

CANDY

0.1.0

Generated by Doxygen 1.9.1

1 Introduction	1
1.1 data format	1
1.2 Built-in name tags	1
1.2.1 Of index approaches (Please go to class @ref IndexTable for more details)	1
1.2.2 Of data loaders (Please go to class @ref DataLoaderTable for more details)	2
1.3 Built-in benchmarks	2
1.3.1 The online insert benchmark	2
1.3.2 The sequential multiple Read write benchmark	2
1.4 How to extend a index algorithm (pure static c++ based)	3
1.5 How to add a single point test	3
1.6 Python Documents	3
2 Todo List	5
3 Module Index	7
3.1 Modules	7
4 Hierarchical Index	9
4.1 Class Hierarchy	9
5 Class Index	13
5.1 Class List	13
6 File Index	19
6.1 File List	19
7 Module Documentation	23
7.1 The support classes for index approaches	23
7.1.1 Detailed Description	26
7.1.2 Function Documentation	26
7.1.2.1 deleteStringObject()	26
7.1.2.2 deleteTensor()	27
7.1.2.3 endHPC()	27
7.1.2.4 getTensorByIndex()	27
7.1.2.5 inlineMain()	28
7.1.2.6 insertStringObject()	28
7.1.2.7 insertTensor() [1/2]	29
7.1.2.8 insertTensor() [2/2]	29
7.1.2.9 loadInitialStringObject()	30
7.1.2.10 loadInitialTensor()	30
7.1.2.11 offlineBuild()	30
7.1.2.12 rawData()	31
7.1.2.13 reviseTensor()	31
7.1.2.14 searchIndex()	31

7.1.2.15 searchStringObject()	32
7.1.2.16 searchTensor() [1/2]	32
7.1.2.17 searchTensor() [2/2]	33
7.1.2.18 searchTensorAndStringObject()	33
7.1.2.19 setConfig() [1/2]	33
7.1.2.20 setConfig() [2/2]	34
7.1.2.21 setFrozenLevel()	34
7.1.2.22 startHPC()	35
7.2 The data loaders of CANDY	35
7.2.1 Detailed Description	35
7.2.1.1 DataLoader	35
7.2.1.2 DataLoader	35
7.3 The main body of CANDY's indexing approaches	36
7.3.1 Detailed Description	36
7.3.1.1 BODY	36
7.4 The bottom tier of indexing alorithms	36
7.4.1 Detailed Description	39
7.4.2 Macro Definition Documentation	39
7.4.2.1 newAbstractIndex	39
7.5 The upper tier of indexing alorithms, can be container of other indexing ways	39
7.5.1 Detailed Description	41
7.6 Shared Utils	41
7.6.1 Detailed Description	43
7.6.1.1 Utils	43
7.6.2 Function Documentation	43
7.6.2.1 cloneInto()	43
7.6.2.2 edit() [1/4]	44
7.6.2.3 edit() [2/4]	44
7.6.2.4 edit() [3/4]	44
7.6.2.5 edit() [4/4]	45
7.6.2.6 exist()	45
7.6.2.7 existDouble()	45
7.6.2.8 existI64()	46
7.6.2.9 existString()	46
7.6.2.10 existU64()	46
7.6.2.11 fromCArg()	47
7.6.2.12 fromFile()	47
7.6.2.13 fromString()	48
7.6.2.14 getDouble()	48
7.6.2.15 getDoubleMap()	48
7.6.2.16 getI64()	49
7.6.2.17 getI64Map()	49

7.6.2.18 getString()	49
7.6.2.19 getStrMap()	50
7.6.2.20 getU64()	50
7.6.2.21 loadFrom()	50
7.6.2.22 toFile()	51
7.6.2.23 toString()	51
7.6.2.24 tryDouble()	52
7.6.2.25 tryI64()	52
7.6.2.26 tryString()	52
7.6.2.27 tryU64()	53
7.7 Other common class or package under C++20 standard	53
7.7.1 Detailed Description	54
7.8 Configurations	54
7.8.1 Detailed Description	55
7.9 Log utils	55
7.9.1 Detailed Description	56
7.9.2 Function Documentation	56
7.9.2.1 appendLogFile()	56
7.9.2.2 log()	56
7.9.2.3 openLogFile()	57
7.9.2.4 setupLoggingFile()	57
7.10 tensor operations	58
7.10.1 Detailed Description	58
7.11 time stamps	58
7.11.1 Detailed Description	59
7.12 Energy Meter packs	59
7.12.1 Detailed Description	60
7.13 The Micro dataset	60
7.13.1 Detailed Description	61
7.13.2 Function Documentation	61
7.13.2.1 MicroDataSet()	61
7.13.2.2 setSeed()	61
7.14 generic	62
7.14.1 Detailed Description	62
7.14.2 Function Documentation	62
7.14.2.1 genIncrementalAlphabet()	62
7.14.2.2 genRandInt()	63
7.14.2.3 genZipfInt()	64
7.14.2.4 genZipfLut()	64
7.15 time stamp	65
7.15.1 Detailed Description	65
7.15.2 Function Documentation	65

7.15.2.1 genSmoothTimeStamp()	65
7.15.2.2 genZipfTimeStamp()	66
8 Class Documentation	67
8.1 _cl_device_integer_dot_product_acceleration_properties_khr Struct Reference	67
8.2 _cl_device_pci_bus_info_khr Struct Reference	68
8.3 _cl_icd_dispatch Struct Reference	68
8.4 _cl_image_format Struct Reference	72
8.5 _cl_mem_android_native_buffer_host_ptr Struct Reference	72
8.6 _cl_mem_ext_host_ptr Struct Reference	73
8.7 _cl_mem_ion_host_ptr Struct Reference	74
8.8 _cl_motion_estimation_desc_intel Struct Reference	75
8.9 _cl_name_version_khr Struct Reference	75
8.10 _cl_queue_family_properties_intel Struct Reference	76
8.11 INTELLI::AbstractC20Thread Class Reference	77
8.11.1 Detailed Description	78
8.11.2 Member Function Documentation	78
8.11.2.1 inlineMain()	78
8.12 CANDY::AbstractDataLoader Class Reference	78
8.12.1 Detailed Description	79
8.12.2 Member Function Documentation	79
8.12.2.1 getData()	79
8.12.2.2 getQuery()	80
8.12.2.3 hijackConfig()	80
8.12.2.4 setConfig()	80
8.13 CANDY::AbstractIndex Class Reference	81
8.13.1 Detailed Description	84
8.13.2 Member Function Documentation	84
8.13.2.1 deleteStringObject()	84
8.13.2.2 deleteTensor()	85
8.13.2.3 endHPC()	85
8.13.2.4 getTensorByIndex()	86
8.13.2.5 insertStringObject()	86
8.13.2.6 insertTensor()	87
8.13.2.7 loadInitialStringObject()	87
8.13.2.8 loadInitialTensor()	88
8.13.2.9 loadInitialTensorAndQueryDistribution()	88
8.13.2.10 offlineBuild()	89
8.13.2.11 rawData()	89
8.13.2.12 reviseTensor()	89
8.13.2.13 searchIndex()	90
8.13.2.14 searchStringObject()	90

8.13.2.15 searchTensor()	91
8.13.2.16 searchTensorAndStringObject()	91
8.13.2.17 setConfig()	92
8.13.2.18 setConfigClass()	92
8.13.2.19 setFrozenLevel()	93
8.13.2.20 setTier()	93
8.13.2.21 startHPC()	93
8.13.2.22 waitPendingOperations()	94
8.14 DIVERSE_METER::AbstractMeter Class Reference	94
8.14.1 Detailed Description	96
8.14.2 Member Function Documentation	96
8.14.2.1 getStaicEnergyConsumption()	96
8.14.2.2 setConfig()	96
8.14.2.3 setStaticPower()	97
8.14.2.4 testStaticPower()	97
8.15 CANDY::AdSampling Class Reference	97
8.16 BS::blocks< T1, T2, T > Class Template Reference	98
8.16.1 Detailed Description	98
8.16.2 Constructor & Destructor Documentation	98
8.16.2.1 blocks()	98
8.16.3 Member Function Documentation	99
8.16.3.1 end()	99
8.16.3.2 get_num_blocks()	99
8.16.3.3 get_total_size()	100
8.16.3.4 start()	100
8.17 CANDY::FLANN::BranchStruct< T > Class Template Reference	100
8.17.1 Detailed Description	101
8.18 CANDY::BucketedFlatIndex Class Reference	101
8.18.1 Detailed Description	103
8.18.2 Member Function Documentation	104
8.18.2.1 deleteTensor()	104
8.18.2.2 encodeMultiRows()	104
8.18.2.3 insertTensor()	105
8.18.2.4 loadInitialTensor()	105
8.18.2.5 reviseTensor()	105
8.18.2.6 searchSingleRow()	106
8.18.2.7 searchTensor()	106
8.18.2.8 setConfig()	107
8.19 CANDY::BufferedCongestionDropIndex Class Reference	107
8.19.1 Detailed Description	109
8.19.2 Member Function Documentation	111
8.19.2.1 deleteTensor()	111

8.19.2.2 endHPC()	112
8.19.2.3 generateBucketedFlatIndexConfig()	112
8.19.2.4 insertTensor()	112
8.19.2.5 insertTensorInline()	113
8.19.2.6 loadInitialTensor()	113
8.19.2.7 offlineBuild()	113
8.19.2.8 reviseTensor()	114
8.19.2.9 searchTensor()	114
8.19.2.10 setConfig()	115
8.19.2.11 setFrozenLevel()	115
8.19.2.12 startHPC()	116
8.20 INTELLI::C20Buffer< dataType > Class Template Reference	116
8.20.1 Detailed Description	117
8.20.2 Constructor & Destructor Documentation	117
8.20.2.1 C20Buffer()	117
8.20.3 Member Function Documentation	117
8.20.3.1 append() [1/2]	117
8.20.3.2 append() [2/2]	118
8.20.3.3 bufferSize()	118
8.20.3.4 data() [1/2]	119
8.20.3.5 data() [2/2]	119
8.20.3.6 size()	119
8.21 CANDY::CANDYObject Class Reference	120
8.21.1 Detailed Description	121
8.21.2 Member Function Documentation	121
8.21.2.1 getStr()	121
8.21.2.2 setStr()	121
8.22 cl_char16 Union Reference	121
8.23 cl_char2 Union Reference	122
8.24 cl_char4 Union Reference	122
8.25 cl_char8 Union Reference	122
8.26 cl_double16 Union Reference	122
8.27 cl_double2 Union Reference	123
8.28 cl_double4 Union Reference	123
8.29 cl_double8 Union Reference	123
8.30 cl_float16 Union Reference	123
8.31 cl_float2 Union Reference	124
8.32 cl_float4 Union Reference	124
8.33 cl_float8 Union Reference	124
8.34 cl_half16 Union Reference	124
8.35 cl_half2 Union Reference	125
8.36 cl_half4 Union Reference	125

8.37 cl_half8 Union Reference	125
8.38 cl_int16 Union Reference	125
8.39 cl_int2 Union Reference	126
8.40 cl_int4 Union Reference	126
8.41 cl_int8 Union Reference	126
8.42 cl_long16 Union Reference	126
8.43 cl_long2 Union Reference	127
8.44 cl_long4 Union Reference	127
8.45 cl_long8 Union Reference	127
8.46 cl_short16 Union Reference	127
8.47 cl_short2 Union Reference	128
8.48 cl_short4 Union Reference	128
8.49 cl_short8 Union Reference	128
8.50 cl_uchar16 Union Reference	128
8.51 cl_uchar2 Union Reference	129
8.52 cl_uchar4 Union Reference	129
8.53 cl_uchar8 Union Reference	129
8.54 cl_uint16 Union Reference	129
8.55 cl_uint2 Union Reference	130
8.56 cl_uint4 Union Reference	130
8.57 cl_uint8 Union Reference	130
8.58 cl_ulong16 Union Reference	130
8.59 cl_ulong2 Union Reference	131
8.60 cl_ulong4 Union Reference	131
8.61 cl_ulong8 Union Reference	131
8.62 cl_ushort16 Union Reference	131
8.63 cl_ushort2 Union Reference	132
8.64 cl_ushort4 Union Reference	132
8.65 cl_ushort8 Union Reference	132
8.66 TONY_CL_HOST::CLContainer Class Reference	132
8.67 CANDY::Clustering Class Reference	133
8.67.1 Detailed Description	135
8.67.2 Member Function Documentation	135
8.67.2.1 computeCentroids()	135
8.67.2.2 imbalance_factor()	135
8.67.2.3 splitClusters()	136
8.67.2.4 train()	136
8.68 CANDY::ClusteringIterationStats Class Reference	137
8.68.1 Detailed Description	137
8.69 CANDY::ClusteringParameters Class Reference	138
8.69.1 Detailed Description	139
8.70 INTELLI::ConfigMap Class Reference	139

8.70.1 Detailed Description	141
8.71 CANDY::CongestionDropIndex Class Reference	141
8.71.1 Detailed Description	143
8.71.2 Member Function Documentation	144
8.71.2.1 deleteStringObject()	144
8.71.2.2 deleteTensor()	145
8.71.2.3 endHPC()	145
8.71.2.4 getTensorByIndex()	145
8.71.2.5 insertStringObject()	146
8.71.2.6 insertTensor()	146
8.71.2.7 loadInitialStringObject()	147
8.71.2.8 loadInitialTensor()	147
8.71.2.9 offlineBuild()	148
8.71.2.10 rawData()	148
8.71.2.11 reviseTensor()	148
8.71.2.12 searchStringObject()	149
8.71.2.13 searchTensor()	149
8.71.2.14 searchTensorAndStringObject()	150
8.71.2.15 setConfig()	150
8.71.2.16 setFrozenLevel()	151
8.71.2.17 startHPC()	151
8.71.2.18 waitPendingOperations()	152
8.72 CANDY::CongestionDropIndexWorker Class Reference	152
8.72.1 Detailed Description	153
8.73 CANDY::DataLoaderTable Class Reference	154
8.73.1 Detailed Description	155
8.73.2 Constructor & Destructor Documentation	155
8.73.2.1 DataLoaderTable()	155
8.73.3 Member Function Documentation	155
8.73.3.1 findDataLoader()	155
8.73.3.2 registerNewDataLoader()	156
8.74 default_attrs Struct Reference	156
8.74.1 Detailed Description	156
8.75 CANDY::FLANN::DistanceIndex Class Reference	157
8.75.1 Detailed Description	157
8.76 CANDY::DistanceQueryer Class Reference	157
8.76.1 Member Function Documentation	158
8.76.1.1 operator>()	158
8.77 CANDY::DistributedIndexWorker Class Reference	159
8.77.1 Detailed Description	160
8.77.2 Member Function Documentation	161
8.77.2.1 deleteTensor()	161

8.77.2.2 endHPC()	161
8.77.2.3 getUnblockQueryResult()	161
8.77.2.4 insertTensor()	162
8.77.2.5 loadInitialTensor()	162
8.77.2.6 loadInitialTensorUnblocked()	162
8.77.2.7 offlineBuild()	163
8.77.2.8 offlineBuildUnblocked()	163
8.77.2.9 searchTensor()	163
8.77.2.10 searchTensorUnblock()	164
8.77.2.11 setConfig()	164
8.77.2.12 setFrozenLevel()	164
8.77.2.13 startHPC()	165
8.77.2.14 waitPendingOperations()	165
8.78 CANDY::DistributedPartitionIndex Class Reference	166
8.78.1 Detailed Description	167
8.78.2 Member Function Documentation	168
8.78.2.1 deleteTensor()	168
8.78.2.2 endHPC()	168
8.78.2.3 getTensorByIndex()	169
8.78.2.4 insertTensor()	169
8.78.2.5 loadInitialTensor()	170
8.78.2.6 offlineBuild()	170
8.78.2.7 rawData()	170
8.78.2.8 reviseTensor()	171
8.78.2.9 searchTensor()	171
8.78.2.10 setConfig()	172
8.78.2.11 setFrozenLevel()	172
8.78.2.12 startHPC()	172
8.78.2.13 waitPendingOperations()	173
8.79 CANDY::DIW_RayWrapper Class Reference	173
8.79.1 Detailed Description	174
8.79.2 Member Function Documentation	174
8.79.2.1 deleteTensor()	174
8.79.2.2 endHPC()	175
8.79.2.3 insertTensor()	175
8.79.2.4 loadInitialTensor()	175
8.79.2.5 offlineBuild()	176
8.79.2.6 searchTensor()	176
8.79.2.7 setConfig()	177
8.79.2.8 setFrozenLevel()	177
8.79.2.9 startHPC()	177
8.79.2.10 waitPendingOperations()	178

8.80 CANDY::DPGIndex Class Reference	178
8.80.1 Detailed Description	180
8.80.2 Member Function Documentation	181
8.80.2.1 deleteTensor()	181
8.80.2.2 endHPC()	181
8.80.2.3 getTensorByIndex()	181
8.80.2.4 insertTensor()	182
8.80.2.5 loadInitialTensor()	182
8.80.2.6 offlineBuild()	183
8.80.2.7 rawData()	183
8.80.2.8 reviseTensor()	183
8.80.2.9 searchTensor()	184
8.80.2.10 setConfig()	184
8.80.2.11 setFrozenLevel()	185
8.80.2.12 startHPC()	185
8.81 DIVERSE_METER::EspMeterUart Class Reference	186
8.81.1 Detailed Description	187
8.81.2 Member Function Documentation	187
8.81.2.1 setConfig()	187
8.82 CANDY::ExpFamilyDataLoader Class Reference	187
8.82.1 Detailed Description	189
8.82.2 Member Function Documentation	190
8.82.2.1 getData()	190
8.82.2.2 getQuery()	190
8.82.2.3 hijackConfig()	190
8.82.2.4 setConfig()	191
8.83 CANDY::FaissIndex Class Reference	191
8.83.1 Detailed Description	193
8.83.2 Member Function Documentation	193
8.83.2.1 getTensorByIndex()	193
8.83.2.2 insertTensor()	194
8.83.2.3 loadInitialTensor()	194
8.83.2.4 searchIndex()	195
8.83.2.5 searchTensor()	195
8.83.2.6 setConfig()	196
8.84 CANDY::FlannComponent Class Reference	196
8.84.1 Member Function Documentation	197
8.84.1.1 setParams()	197
8.85 CANDY::FlannIndex Class Reference	198
8.85.1 Member Function Documentation	199
8.85.1.1 getTensorByIndex()	199
8.85.1.2 insertTensor()	199

8.85.1.3 loadInitialTensor()	200
8.85.1.4 searchIndex()	200
8.85.1.5 searchTensor()	200
8.85.1.6 setConfig()	201
8.86 CANDY::FlannParam Struct Reference	202
8.87 CANDY::FlatAMMIPIndex Class Reference	202
8.87.1 Detailed Description	204
8.87.2 Member Function Documentation	205
8.87.2.1 deleteTensor()	205
8.87.2.2 getTensorByIndex()	205
8.87.2.3 insertTensor()	206
8.87.2.4 rawData()	207
8.87.2.5 reviseTensor()	207
8.87.2.6 searchIndex()	208
8.87.2.7 searchTensor()	208
8.87.2.8 setConfig()	208
8.87.2.9 size()	209
8.88 CANDY::FlatAMMIPObjIndex Class Reference	209
8.88.1 Detailed Description	211
8.88.2 Member Function Documentation	211
8.88.2.1 deleteStringObject()	211
8.88.2.2 deleteTensor()	212
8.88.2.3 getTensorByIndex()	212
8.88.2.4 insertStringObject()	213
8.88.2.5 insertTensor()	213
8.88.2.6 rawData()	214
8.88.2.7 reviseTensor()	214
8.88.2.8 searchIndex()	214
8.88.2.9 searchStringObject()	215
8.88.2.10 searchTensor()	215
8.88.2.11 setConfig()	216
8.88.2.12 size()	216
8.89 CANDY::FlatIndex Class Reference	216
8.89.1 Detailed Description	218
8.89.2 Member Function Documentation	218
8.89.2.1 deleteTensor()	218
8.89.2.2 getTensorByIndex()	218
8.89.2.3 insertTensor()	219
8.89.2.4 rawData()	219
8.89.2.5 reviseTensor()	219
8.89.2.6 searchIndex()	220
8.89.2.7 searchTensor()	220

8.89.2.8 setConfig()	221
8.89.2.9 size()	221
8.90 CANDY::FVECSDataLoader Class Reference	222
8.90.1 Detailed Description	223
8.90.2 Member Function Documentation	224
8.90.2.1 getData()	224
8.90.2.2 getQuery()	224
8.90.2.3 setConfig()	224
8.90.2.4 tensorFromFVECS()	225
8.91 CANDY::HDF5DataLoader Class Reference	225
8.91.1 Detailed Description	227
8.91.2 Member Function Documentation	227
8.91.2.1 getData()	227
8.91.2.2 getQuery()	228
8.91.2.3 setConfig()	228
8.91.2.4 tensorFromHDF5()	228
8.92 CANDY::FLANN::Heap< T > Class Template Reference	229
8.92.1 Detailed Description	229
8.93 CANDY::HNSW Class Reference	230
8.93.1 Detailed Description	232
8.93.2 Member Function Documentation	232
8.93.2.1 add_links_starting_from()	232
8.93.2.2 add_without_lock()	232
8.93.2.3 cum_nb_neighbors()	233
8.93.2.4 nb_neighbors()	233
8.93.2.5 neighbor_range()	234
8.93.2.6 prepare_level_tab()	234
8.93.2.7 random_level()	234
8.93.2.8 search()	235
8.93.2.9 set_nb_neighbors()	235
8.93.2.10 set_probs()	235
8.93.3 Member Data Documentation	236
8.93.3.1 cum_nneighbor_per_level	236
8.93.3.2 search_bounded_queue	236
8.94 HNSWalter Class Reference	236
8.95 CANDY::HNSWNaiveIndex Class Reference	236
8.95.1 Detailed Description	238
8.95.2 Member Function Documentation	238
8.95.2.1 deleteTensor()	238
8.95.2.2 insertTensor()	239
8.95.2.3 reviseTensor()	239
8.95.2.4 searchTensor()	239

8.95.2.5 setConfig()	240
8.96 CANDY::HNSWVertex Class Reference	240
8.96.1 Detailed Description	241
8.97 HostPara Class Reference	242
8.98 CANDY::IndexTable Class Reference	242
8.98.1 Detailed Description	243
8.98.2 Member Function Documentation	243
8.98.2.1 addIndex()	243
8.98.2.2 getIndex()	244
8.99 INTELLI::IntelliLog Class Reference	244
8.99.1 Detailed Description	244
8.100 INTELLI::IntelliLog_FileProtector Class Reference	245
8.100.1 Detailed Description	245
8.101 INTELLI::IntelliTensorOP Class Reference	245
8.101.1 Member Function Documentation	246
8.101.1.1 appendRows() [1/2]	246
8.101.1.2 appendRows() [2/2]	247
8.101.1.3 appendRowsBufferMode() [1/2]	247
8.101.1.4 appendRowsBufferMode() [2/2]	248
8.101.1.5 deleteRow() [1/2]	248
8.101.1.6 deleteRow() [2/2]	249
8.101.1.7 deleteRowBufferMode() [1/2]	249
8.101.1.8 deleteRowBufferMode() [2/2]	249
8.101.1.9 deleteRows() [1/2]	250
8.101.1.10 deleteRows() [2/2]	250
8.101.1.11 deleteRowsBufferMode() [1/2]	251
8.101.1.12 deleteRowsBufferMode() [2/2]	251
8.101.1.13 editRows() [1/2]	252
8.101.1.14 editRows() [2/2]	252
8.101.1.15 insertRows() [1/2]	253
8.101.1.16 insertRows() [2/2]	254
8.101.1.17 l2Normalize()	254
8.101.1.18 rowSampling()	255
8.101.1.19 tensorFromFile()	255
8.101.1.20 tensorFromFlatBin()	255
8.101.1.21 tensorToFile()	256
8.101.1.22 tensorToFlatBin()	256
8.102 INTELLITensorOP Class Reference	257
8.102.1 Detailed Description	257
8.103 INTELLI::IntelliTimeStamp Class Reference	257
8.103.1 Detailed Description	258
8.104 INTELLI::IntelliTimeStampGenerator Class Reference	258

8.104.1 Detailed Description	259
8.104.2 Member Function Documentation	260
8.104.2.1 generateArrival()	260
8.104.2.2 getTimeStamps()	260
8.104.2.3 setConfig()	260
8.105 DIVERSE_METER::IntelMeter Class Reference	261
8.105.1 Detailed Description	262
8.105.2 Member Function Documentation	262
8.105.2.1 setConfig()	262
8.106 CANDY::IVFListBucket Class Reference	263
8.106.1 Detailed Description	264
8.106.2 Member Function Documentation	264
8.106.2.1 deleteTensor()	264
8.106.2.2 deleteTensorWithEncode()	264
8.106.2.3 getAllTensors()	265
8.106.2.4 getAllTensorsWithEncode()	265
8.106.2.5 getMinimumTensorsUnderHamming()	265
8.106.2.6 insertTensorWithEncode()	266
8.106.2.7 sizeWithEncode()	266
8.107 CANDY::IVFListCell Class Reference	267
8.107.1 Detailed Description	267
8.107.2 Member Function Documentation	268
8.107.2.1 deleteTensor()	268
8.107.2.2 deleteTensorPtr()	268
8.107.2.3 getAllTensors()	268
8.107.2.4 insertTensor()	269
8.107.2.5 insertTensorPtr()	269
8.108 CANDY::IVFTensorEncodingList Class Reference	269
8.108.1 Detailed Description	270
8.108.2 Member Function Documentation	270
8.108.2.1 deleteTensorWithEncode()	270
8.108.2.2 getMinimumNumOfTensors()	271
8.108.2.3 getMinimumNumOfTensorsHamming()	271
8.108.2.4 getMinimumNumOfTensorsInsideBucket()	272
8.108.2.5 getMinimumNumOfTensorsInsideBucketHamming()	273
8.108.2.6 init()	273
8.108.2.7 insertTensorWithEncode()	273
8.109 CANDY::KdTree Class Reference	274
8.109.1 Member Function Documentation	275
8.109.1.1 addPoints()	275
8.109.1.2 addPointToTree()	276
8.109.1.3 divideTree()	276

8.109.1.4	getNeighbors()	276
8.109.1.5	knnSearch()	277
8.109.1.6	meanSplit()	277
8.109.1.7	planeSplit()	278
8.109.1.8	searchLevel()	278
8.109.1.9	selectDivision()	279
8.109.1.10	setConfig()	279
8.109.1.11	setParams()	279
8.110	CANDY::KmeansTree Class Reference	280
8.110.1	Detailed Description	282
8.110.2	Member Function Documentation	282
8.110.2.1	addPoints()	282
8.110.2.2	addPointToTree()	282
8.110.2.3	computeClustering()	282
8.110.2.4	computeNodeStat()	283
8.110.2.5	explore()	283
8.110.2.6	findNN()	283
8.110.2.7	getNeighbors()	284
8.110.2.8	knnSearch()	284
8.110.2.9	setConfig()	285
8.110.2.10	setParams()	285
8.111	INTELLI::MemoryTracker Class Reference	286
8.111.1	Detailed Description	286
8.111.2	Member Function Documentation	287
8.111.2.1	getAvgCpu()	287
8.111.2.2	getAvgMem()	287
8.111.2.3	getCurMem()	287
8.111.2.4	getMaxCpu()	287
8.111.2.5	getMaxMem()	288
8.111.2.6	start()	288
8.112	DIVERSE_METER::MeterTable Class Reference	288
8.112.1	Detailed Description	289
8.112.2	Constructor & Destructor Documentation	290
8.112.2.1	MeterTable()	290
8.112.3	Member Function Documentation	290
8.112.3.1	findMeter()	290
8.112.3.2	registerNewMeter()	290
8.113	INTELLI::MicroDataSet Class Reference	291
8.113.1	Detailed Description	291
8.114	CANDY::HNSW::MinimaxHeap Struct Reference	292
8.114.1	Detailed Description	292
8.115	CANDY::MLPBucketIdxModel Class Reference	293

8.115.1 Member Function Documentation	293
8.115.1.1 hash()	293
8.115.1.2 init()	293
8.115.1.3 trainModel()	294
8.116 CANDY::MLPHashingModel Class Reference	294
8.116.1 Member Function Documentation	295
8.116.1.1 fineTuneModel()	295
8.116.1.2 hash()	295
8.116.1.3 init()	295
8.116.1.4 trainModel()	296
8.117 BS::multi_future< T > Class Template Reference	296
8.117.1 Detailed Description	297
8.117.2 Constructor & Destructor Documentation	297
8.117.2.1 multi_future()	297
8.117.3 Member Function Documentation	297
8.117.3.1 get()	298
8.117.3.2 operator[]()	298
8.117.3.3 push_back()	298
8.117.3.4 size()	298
8.118 CANDY::DPGIndex::Neighbor Struct Reference	299
8.119 CANDY::NNDescentIndex::Neighbor Struct Reference	300
8.120 CANDY::NNDescentIndex::Nhoo Struct Reference	300
8.121 CANDY::DPGIndex::NhooLayer0 Struct Reference	301
8.122 CANDY::DPGIndex::NhooLayer1 Struct Reference	302
8.123 CANDY::NNDescentIndex Class Reference	303
8.123.1 Detailed Description	305
8.123.2 Member Function Documentation	305
8.123.2.1 deleteTensor()	305
8.123.2.2 endHPC()	305
8.123.2.3 getTensorByIndex()	306
8.123.2.4 insertTensor()	306
8.123.2.5 loadInitialTensor()	307
8.123.2.6 offlineBuild()	307
8.123.2.7 rawData()	307
8.123.2.8 reviseTensor()	308
8.123.2.9 searchTensor()	308
8.123.2.10 setConfig()	308
8.123.2.11 setFrozenLevel()	309
8.123.2.12 startHPC()	309
8.124 CANDY::KdTree::Node Struct Reference	310
8.125 CANDY::KmeansTree::Node Struct Reference	310
8.126 CANDY::HNSW::NodeDistCloser Struct Reference	311

8.126.1 Detailed Description	312
8.127 CANDY::HNSW::NodeDistFarther Struct Reference	312
8.127.1 Detailed Description	312
8.128 CANDY::KmeansTree::NodeInfo Struct Reference	313
8.129 CANDY::OnlineIVFL2HIndex Class Reference	313
8.129.1 Detailed Description	315
8.129.2 Member Function Documentation	315
8.129.2.1 loadInitialTensor()	315
8.129.2.2 loadInitialTensorAndQueryDistribution()	316
8.129.2.3 setConfig()	316
8.129.2.4 trainModelWithData()	317
8.130 CANDY::OnlineIVFLSHIndex Class Reference	317
8.130.1 Detailed Description	319
8.130.2 Member Function Documentation	319
8.130.2.1 crsAmm()	319
8.130.2.2 deleteRowsInline()	320
8.130.2.3 deleteTensor()	320
8.130.2.4 insertTensor()	321
8.130.2.5 reviseTensor()	321
8.130.2.6 searchTensor()	321
8.130.2.7 setConfig()	322
8.131 CANDY::OnlinePQIndex Class Reference	322
8.131.1 Detailed Description	324
8.131.2 Member Function Documentation	325
8.131.2.1 deleteRowsInline()	325
8.131.2.2 deleteTensor()	325
8.131.2.3 fineGrainedEncode()	326
8.131.2.4 insertTensor()	326
8.131.2.5 loadInitialTensor()	326
8.131.2.6 offlineBuild()	327
8.131.2.7 reviseTensor()	328
8.131.2.8 searchTensor()	328
8.131.2.9 setConfig()	328
8.131.2.10 setFrozenLevel()	329
8.132 CANDY::ParallelIndexWorker Class Reference	330
8.132.1 Detailed Description	332
8.133 CANDY::ParallelPartitionIndex Class Reference	333
8.133.1 Detailed Description	335
8.133.2 Member Function Documentation	335
8.133.2.1 deleteStringObject()	335
8.133.2.2 deleteTensor()	336
8.133.2.3 endHPC()	336

8.133.2.4	getTensorByIndex()	337
8.133.2.5	insertStringObject()	337
8.133.2.6	insertTensor()	337
8.133.2.7	loadInitialStringObject()	338
8.133.2.8	loadInitialTensor()	338
8.133.2.9	offlineBuild()	339
8.133.2.10	rawData()	339
8.133.2.11	reviseTensor()	339
8.133.2.12	searchStringObject()	340
8.133.2.13	searchTensor()	340
8.133.2.14	searchTensorAndStringObject()	341
8.133.2.15	setConfig()	341
8.133.2.16	setFrozenLevel()	342
8.133.2.17	startHPC()	342
8.133.2.18	waitPendingOperations()	343
8.134	INTELLI::ThreadPerf::PerfPair Class Reference	343
8.134.1	Detailed Description	344
8.135	INTELLI::ThreadPerf::PerfTool Class Reference	344
8.136	CANDY::PQDecoder Class Reference	344
8.136.1	Detailed Description	345
8.136.2	Member Function Documentation	345
8.136.2.1	decode()	345
8.137	CANDY::PQEncoder Class Reference	346
8.137.1	Detailed Description	346
8.137.2	Member Function Documentation	346
8.137.2.1	encode()	346
8.138	CANDY::PQIndex Class Reference	347
8.138.1	Detailed Description	349
8.138.2	Member Function Documentation	349
8.138.2.1	add()	349
8.138.2.2	deleteTensor()	349
8.138.2.3	getTensorByIndex()	351
8.138.2.4	insertTensor()	351
8.138.2.5	loadInitialTensor()	352
8.138.2.6	rawData()	352
8.138.2.7	reviseTensor()	352
8.138.2.8	searchIndex()	353
8.138.2.9	searchTensor()	353
8.138.2.10	setConfig()	354
8.138.2.11	setFrozenLevel()	354
8.138.2.12	train()	355
8.139	CANDY::ProductQuantizer Class Reference	355

8.139.1 Detailed Description	356
8.139.2 Member Function Documentation	356
8.139.2.1 add()	357
8.139.2.2 compute_code()	357
8.139.2.3 compute_codes()	357
8.139.2.4 compute_distance_table()	358
8.139.2.5 compute_distance_tables()	358
8.139.2.6 decode()	358
8.139.2.7 search()	359
8.139.2.8 setCentroidsFrom() [1/2]	359
8.139.2.9 setCentroidsFrom() [2/2]	359
8.139.2.10 train()	360
8.140 CANDY::FLANN::RandomCenterChooser Class Reference	360
8.140.1 Detailed Description	360
8.141 CANDY::RandomDataLoader Class Reference	361
8.141.1 Detailed Description	362
8.141.2 Member Function Documentation	362
8.141.2.1 getData()	362
8.141.2.2 getQuery()	363
8.141.2.3 setConfig()	363
8.142 CANDY::Rania_Python Class Reference	363
8.142.1 Detailed Description	365
8.142.2 Member Function Documentation	365
8.142.2.1 dataLoader_create()	365
8.142.2.2 dataLoader_editCfgDouble()	366
8.142.2.3 dataLoader_editCfgFloat()	366
8.142.2.4 dataLoader_editCfgI64()	367
8.142.2.5 dataLoader_editCfgStr()	367
8.142.2.6 dataLoader_getData()	368
8.142.2.7 dataLoader_getQuery()	368
8.142.2.8 dataLoader_init()	368
8.142.2.9 index_create()	369
8.142.2.10 index_delete()	369
8.142.2.11 index_deleteString()	369
8.142.2.12 index_editCfgDouble()	370
8.142.2.13 index_editCfgI64()	370
8.142.2.14 index_editCfgStr()	371
8.142.2.15 index_endHPC()	371
8.142.2.16 index_init()	371
8.142.2.17 index_insert()	372
8.142.2.18 index_insertString()	372
8.142.2.19 index_loadCfgFromFile()	373

8.142.2.20	index_loadInitial()	373
8.142.2.21	index_loadInitialString()	373
8.142.2.22	index_offlineBuild()	374
8.142.2.23	index_rawData()	374
8.142.2.24	index_reset()	375
8.142.2.25	index_revise()	375
8.142.2.26	index_search()	375
8.142.2.27	index_searchString()	376
8.142.2.28	index_searchTensorAndStringList()	376
8.142.2.29	index_setFrozenLevel()	377
8.142.2.30	index_startHPC()	377
8.142.2.31	index_waitPending()	377
8.142.2.32	tensorFromFile()	378
8.142.2.33	tensorFromFVECS()	378
8.142.2.34	tensorFromHDF5()	378
8.142.2.35	tensorToFile()	379
8.143	DIVERSE_METER::rapl_power_unit Struct Reference	379
8.144	CANDY::FLANN::ResultSet Class Reference	380
8.144.1	Detailed Description	380
8.145	CANDY::SimpleStreamClustering Class Reference	380
8.145.1	Detailed Description	382
8.145.2	Member Function Documentation	382
8.145.2.1	addSingleRow()	382
8.145.2.2	addSingleRowWithIdx()	382
8.145.2.3	buildCentroids()	383
8.145.2.4	classifyMultiRow()	383
8.145.2.5	classifySingleRow()	384
8.145.2.6	deleteSingleRow()	384
8.145.2.7	deleteSingleRowWithIdx()	385
8.145.2.8	euclideanDelete()	385
8.145.2.9	euclideanDistance()	385
8.145.2.10	euclideanInsert()	386
8.145.2.11	exportCentroids()	386
8.145.2.12	loadCentroids() [1/2]	386
8.145.2.13	loadCentroids() [2/2]	387
8.145.2.14	saveCentroidsToFile()	387
8.146	INTELLI::SPSCQueue< T, Allocator > Class Template Reference	388
8.147	BS::synced_stream Class Reference	389
8.147.1	Detailed Description	389
8.147.2	Constructor & Destructor Documentation	390
8.147.2.1	synced_stream()	390
8.147.3	Member Function Documentation	390

8.147.3.1 print()	390
8.147.3.2 println()	390
8.147.4 Member Data Documentation	391
8.147.4.1 endl	391
8.147.4.2 flush	391
8.148 CANDY::TensorIdxPair Class Reference	392
8.148.1 Detailed Description	392
8.149 CANDY::TensorListIdxPair Class Reference	393
8.150 CANDY::TensorStrPair Class Reference	393
8.151 CANDY::TensorStrVecPair Class Reference	394
8.152 BS::thread_pool Class Reference	395
8.152.1 Detailed Description	396
8.152.2 Constructor & Destructor Documentation	396
8.152.2.1 thread_pool()	396
8.152.3 Member Function Documentation	396
8.152.3.1 get_tasks_queued()	396
8.152.3.2 get_tasks_running()	397
8.152.3.3 get_tasks_total()	397
8.152.3.4 get_thread_count()	397
8.152.3.5 is_paused()	397
8.152.3.6 parallelize_loop() [1/2]	398
8.152.3.7 parallelize_loop() [2/2]	398
8.152.3.8 push_loop() [1/2]	399
8.152.3.9 push_loop() [2/2]	400
8.152.3.10 push_task()	400
8.152.3.11 reset()	401
8.152.3.12 submit()	401
8.153 INTELLI::ThreadPerf Class Reference	402
8.153.1 Detailed Description	404
8.153.2 Constructor & Destructor Documentation	404
8.153.2.1 ThreadPerf()	404
8.153.3 Member Function Documentation	405
8.153.3.1 getResultById()	405
8.153.3.2 getResultByName()	405
8.153.3.3 initEventsByCfg()	406
8.153.3.4 resultToConfigMap()	406
8.153.3.5 start()	406
8.154 INTELLI::ThreadPerfPAPI Class Reference	407
8.154.1 Detailed Description	408
8.154.2 Constructor & Destructor Documentation	409
8.154.2.1 ThreadPerfPAPI()	409
8.154.3 Member Function Documentation	409

8.154.3.1 addPapiTag() [1/2]	409
8.154.3.2 addPapiTag() [2/2]	410
8.154.3.3 getResultById()	410
8.154.3.4 getResultByName()	410
8.154.3.5 initEventsByCfg()	411
8.154.3.6 resultToConfigMap()	411
8.154.3.7 start()	412
8.155 BS::timer Class Reference	412
8.155.1 Detailed Description	412
8.155.2 Member Function Documentation	412
8.155.2.1 ms()	412
8.156 CANDY::FLANN::UniqueRandom Class Reference	413
8.156.1 Detailed Description	413
8.157 INTELLI::UtilityFunctions Class Reference	413
8.157.1 Member Function Documentation	413
8.157.1.1 bind2Core()	413
8.157.1.2 calculateRecall()	414
8.157.1.3 getLatencyPercentage()	414
8.157.1.4 saveTimeStampToFile()	415
8.157.1.5 tensorListFromFile()	415
8.157.1.6 tensorListToFile()	415
8.158 VertexComparison Struct Reference	416
8.159 CANDY::FLANN::VisitBitset Class Reference	416
8.160 VisitedBitset Class Reference	417
8.160.1 Detailed Description	417
8.161 CANDY::VisitedTable Class Reference	417
8.161.1 Detailed Description	418
8.162 CANDY::YinYangGraph Class Reference	418
8.162.1 Detailed Description	419
8.162.2 Member Function Documentation	419
8.162.2.1 deleteTensorWithEncode()	419
8.162.2.2 getMinimumNumOfTensors()	419
8.162.2.3 init()	420
8.162.2.4 insertTensorWithEncode()	420
8.163 CANDY::YinYangGraph_DistanceFunctions Class Reference	421
8.164 CANDY::YinYangGraph_ListBucket Class Reference	421
8.164.1 Detailed Description	422
8.164.2 Member Function Documentation	422
8.164.2.1 deleteTensor()	422
8.164.2.2 deleteTensorWithEncode()	423
8.164.2.3 getVertexWithEncode()	423
8.164.2.4 insertTensorWithEncode()	423

8.165 CANDY::YinYangGraph_ListCell Class Reference	424
8.165.1 Detailed Description	425
8.165.2 Member Function Documentation	425
8.165.2.1 deleteTensor()	425
8.165.2.2 getVertex()	425
8.165.2.3 insertTensor()	426
8.166 CANDY::YinYangGraphIndex Class Reference	426
8.166.1 Detailed Description	428
8.166.2 Member Function Documentation	428
8.166.2.1 crsAmm()	428
8.166.2.2 insertTensor()	429
8.166.2.3 searchTensor()	429
8.166.2.4 setConfig()	429
8.167 CANDY::YinYangGraphSimpleIndex Class Reference	430
8.167.1 Detailed Description	431
8.167.2 Member Function Documentation	431
8.167.2.1 insertSingleRowTensor()	431
8.167.2.2 insertTensor()	432
8.167.2.3 searchTensor()	432
8.167.2.4 setConfig()	433
8.168 CANDY::YinYangVertex Class Reference	433
8.168.1 Detailed Description	435
8.168.2 Member Function Documentation	435
8.168.2.1 attachTensor()	435
8.168.2.2 detachTensor()	435
8.168.2.3 greedySearchForKNearestTensor()	436
8.168.2.4 greedySearchForKNearestVertex() [1/2]	436
8.168.2.5 greedySearchForKNearestVertex() [2/2]	437
8.168.2.6 greedySearchForNearestVertex() [1/2]	438
8.168.2.7 greedySearchForNearestVertex() [2/2]	438
8.168.2.8 init()	438
8.168.2.9 setUpperLayer()	439
8.168.2.10 toString()	439
8.168.2.11 tryToConnect()	440
8.169 CANDY::YinYangVertexMap Class Reference	440
8.169.1 Member Function Documentation	441
8.169.1.1 edit()	441
8.169.1.2 erase()	442
8.169.1.3 exist()	442
8.169.1.4 nearestKVertexWithinMap()	442
8.169.1.5 nearestVertexWithinMap()	443
8.169.1.6 nearestVertexWithinMe()	443

8.170 CANDY::ZipfDataLoader Class Reference	444
8.170.1 Detailed Description	445
8.170.2 Member Function Documentation	445
8.170.2.1 getData()	446
8.170.2.2 getQuery()	446
8.170.2.3 setConfig()	446
9 File Documentation	449
9.1 include/CANDY.h File Reference	449
9.2 include/CANDY/AbstractIndex.h File Reference	450
9.3 include/CANDY/BucketedFlatIndex.h File Reference	451
9.4 include/CANDY/BufferedCongestionDropIndex.h File Reference	452
9.5 include/CANDY/CANDYObject.h File Reference	453
9.6 include/CANDY/CongestionDropIndex.h File Reference	454
9.7 include/CANDY/CongestionDropIndex/CongestionDropIndexWorker.h File Reference	455
9.8 include/CANDY/DistributedPartitionIndex.h File Reference	456
9.9 include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h File Reference	457
9.10 include/CANDY/DPGIndex.h File Reference	458
9.11 include/CANDY/FaissIndex.h File Reference	459
9.12 include/CANDY/FlatAMMIPIndex.h File Reference	459
9.13 include/CANDY/FlatAMMIPObjIndex.h File Reference	460
9.14 include/CANDY/FlatIndex.h File Reference	461
9.15 include/CANDY/HNSWNaiveIndex.h File Reference	462
9.16 include/CANDY/IndexTable.h File Reference	463
9.17 include/CANDY/NNDescentIndex.h File Reference	464
9.18 include/CANDY/OnlinePQIndex.h File Reference	465
9.19 include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h File Reference	466
9.20 include/CANDY/OnlinePQIndex/SimpleStreamClustering.h File Reference	467
9.21 include/CANDY/ParallelPartitionIndex.h File Reference	468
9.22 include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h File Reference	469
9.23 include/CANDY/YinYangGraphIndex.h File Reference	471
9.24 include/CANDY/YinYangGraphIndex/YinYangGraph.h File Reference	472
9.25 include/CANDY/YinYangGraphSimpleIndex.h File Reference	473
9.26 include/CANDYPYTHON.h File Reference	474
9.27 include/DataLoader/AbstractDataLoader.h File Reference	475
9.28 include/DataLoader/DataLoaderTable.h File Reference	475
9.29 include/DataLoader/ExpFamilyDataLoader.h File Reference	477
9.30 include/DataLoader/FVECSDDataLoader.h File Reference	478
9.31 include/DataLoader/HDF5DataLoader.h File Reference	479
9.32 include/DataLoader/RandomDataLoader.h File Reference	480
9.33 include/DataLoader/ZipfDataLoader.h File Reference	481
9.34 include/Utils/AbstractC20Thread.hpp File Reference	482

9.35 include/Utils/BS_thread_pool.hpp File Reference	483
9.35.1 Detailed Description	484
9.36 include/Utils/C20Buffers.hpp File Reference	484
9.37 include/Utils/ConfigMap.hpp File Reference	485
9.38 include/Utils/IntelliTensorOP.hpp File Reference	486
9.39 include/Utils/IntelliTimeStampGenerator.h File Reference	487
9.40 include/Utils/Meters/AbstractMeter.hpp File Reference	488
9.41 include/Utils/Meters/EspMeterUart/EspMeterUart.hpp File Reference	489
9.42 include/Utils/ThreadPerf.hpp File Reference	490
9.43 include/Utils/ThreadPerfPAPI.hpp File Reference	492
9.44 src/CANDY/AbstractIndex.cpp File Reference	493
9.45 src/CANDY/BucketedFlatIndex.cpp File Reference	493
9.45.1 Macro Definition Documentation	493
9.45.1.1 BF_NEXT_POW_2	494
9.46 src/CANDY/BufferedCongestionDropIndex.cpp File Reference	494
9.47 src/CANDY/CongestionDropIndex.cpp File Reference	494
9.48 src/CANDY/CongestionDropIndex/CongestionDropIndexWorker.cpp File Reference	495
9.49 src/CANDY/DistributedPartitionIndex.cpp File Reference	495
9.50 src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp File Reference	495
9.51 src/CANDY/DPGIndex.cpp File Reference	496
9.52 src/CANDY/FlatAMMIPIndex.cpp File Reference	496
9.53 src/CANDY/FlatAMMIPObjIndex.cpp File Reference	496
9.54 src/CANDY/FlatIndex.cpp File Reference	497
9.55 src/CANDY/NNDescentIndex.cpp File Reference	497
9.56 src/CANDY/OnlineIVFL2HIndex.cpp File Reference	498
9.56.1 Macro Definition Documentation	498
9.56.1.1 ONLINEIVFL2H_NEXT_POW_2	498
9.57 src/CANDY/OnlineIVFLSHIndex.cpp File Reference	498
9.57.1 Macro Definition Documentation	499
9.57.1.1 ONLINEIVF_NEXT_POW_2	499
9.58 src/CANDY/OnlinePQIndex.cpp File Reference	499
9.59 src/CANDY/ParallelPartitionIndex.cpp File Reference	500
9.60 src/CANDY/ParallelPartitionIndex/ParallelIndexWorker.cpp File Reference	500
9.61 src/CANDY/YinYangGraphIndex.cpp File Reference	500
9.61.1 Macro Definition Documentation	501
9.61.1.1 ONLINEIVF_NEXT_POW_2	501
9.62 src/CANDY/YinYangGraphSimpleIndex.cpp File Reference	501

Chapter 1

Introduction

This project is an index library for online vector management, our key idea is Ranging Augmented Navigation Indexing Approach, while some typical state-of-art is also included

1.1 data format

The api interface is `torch::Tensor` for both c++ and python, and we also include support for loading the following data formats from file

- *.fvecs, (<http://corpus-texmex.irisa.fr/>) using `FVECSDataLoader`, a static public class function `tensorFromFVECS` is also provided
- *.h5, *.hdf5 (<https://github.com/HDFGroup/hdf5>) using `HDF5DataLoader`, a static public class function `tensorFromHDF5` is also provided
 - experimental feature, should using `-DENABLE_HDF5=ON` in cmake
 - not support compression yet

1.2 Built-in name tags

1.2.1 Of index approaches (Please go to class `@ref IndexTable` for more details)

- flat `FlatIndex`
- parallelPartition `ParallelPartitionIndex`
- onlinePQ `OnlinePQIndex`
- onlineIVFLSH `OnlineIVFLSHIndex`
- HNSWNaive `HNSWNaiveIndex`
- faiss `FaissIndex`

1.2.2 Of data loaders (Please go to class `@ref DataLoaderTable` for more details)

- random `RandomDataLoader`
- fvecs `FVECSDataLoader`
- hdf5 `HDF5DataLoader`
- zipf `ZipfDataLoader`
- expFamily `ExpFamilyDataLoader`
- exp, the exponential distribution in `ExpFamilyDataLoader`
- beta, the beta distribution in `ExpFamilyDataLoader`
- gaussian, the beta distribution in `ExpFamilyDataLoader`
- poisson, the poisson distribution in `ExpFamilyDataLoader`

1.3 Built-in benchmarks

1.3.1 The online insert benchmark

This benchmark program evaluates the inserting latency and recall of a specified index, the usage is `./onlineInsert <name of config file>`

Note

required parameters

- `vecDim`, the dimension of vector, I64, default 768,
- `vecVolume`, the volume of row tensors, I64, default value depends on the `DataLoader`
- `eventRateTps`, the event rate of tuples, each tuple is a row, default 100
- `querySize`, the size of your query, I64, default value depends on the `DataLoader`
- `cutOffTimeSeconds`, the setting time to cut off execution after given seconds, default -1 (no cut off), I64
- `batchSize`, the size of batch, I64, default equal to the `vecVolume`
- `staticDataSet`, turn on this to force data to be static and make everything already arrived, I64, default 0
- `indexTag`, the name tag of index class, String, default flat
- `dataLoaderTag`, the name tag of data loader class, String, default random
- `initialRows`, the rows of initially loaded tensors, I64, default 0 (streaming at the begining)
- `waitPendingWrite`, wether or not wait for pending writes before start a query, I64, default 0 (NOT) see also `DataLoaderTable`, `IndexTable`

1.3.2 The sequential multiple Read write benchmark

This benchmark program evaluates the inserting latency and recall of a specified index, but with multiple RW sequences `./multiRW <name of config file>`

Note

additional parameters compared with [The online insert benchmark](#)

- `numberOfRWSeq`, the number of RW sequences, will divide both data base tensor and query tensor by this factor, I64, default 1

1.4 How to extend a index algorithm (pure static c++ based)

- go to the src/CANDY and include/CANDY
- copy the example class, such as FlatIndex, rename it, and implement your own index class
 - copy the cpp and h
 - rename the cpp and h
 - automatically conduct the IDE-full-replace over the template by your own name in cpp and h
 - define your own function
- Note**
 - Please use this copy-and-replace policy rather than creat your own, unless you know the doxygen comment style very well and can always keep it!!!
- Warning**
 - This copy-and-replace policy will also prevent from wrong parameter types of interface functions, please DO KEEP THE INTERFACE PARAMETER UNDER THE SAME TYPE!!!!!!!!!!!!
- register our class with a tag to src/CANDY/IndexTable.cpp
- edit the CMakeList.txt at src/CANDY to include your new algo and recompile
- remember to add a test bench, you can refer to FlatIndexTest.cpp at test/SystemTest for example

1.5 How to add a single point test

- follow and copy the SimpleTest.cpp to create your own, say A.cpp
- register A.cpp to test/CMakeLists.txt, please follow how we deal with the SketchTest.cpp
- assuming you have made A.cpp into a_test, append ./a_test "--success" to the last row of .github/workflows/cmake.yml

1.6 Python Documents

Please find the class named Rania_Python for python APIs

Chapter 2

Todo List

Class `CANDY::CANDYObject`

to finish the functions of setting void * pointers

Class `CANDY::Clustering`

current build of centroids still depends on `IndexFlatL2`, perhaps re-implemented in a total tensor manner

- `train`

Class `CANDY::DistributedPartitionIndex`

consider an unblocked, optimized version of `insertTensor`, as we did in `loadInitialTensor` ?

Class `CANDY::FaissIndex`

more explanation on IVFPQ, NNDecent, LSH, NSG

Member `CANDY::IVFTensorEncodingList::getMinimumNumOfTensorsInsideBucket` (`torch::Tensor &t`, `std::vector< uint8_t > &encode`, `uint64_t bktIdx`, `int64_t minimumNum`)

improve the efficiency of this function in traving lists!

Member `CANDY::IVFTensorEncodingList::getMinimumNumOfTensorsInsideBucketHamming` (`torch::Tensor &t`, `std::vector< uint8_t > &encode`, `uint64_t bktIdx`, `int64_t minimumNum`)

improve the efficiency of this function in traving lists!

Class `CANDY::PQIndex`

delete and revise a tensor may not be feasible for `PQIndex`

- `deleteTensor`
- `reviseTensor`

encode and decode may be verbose for both code tensor and code pointers

- `searchTensor`
- `insertTensor`

Class `CANDY::SimpleStreamClustering`

two functions are extremely slow and costly, needs to be re-implemented

- `buildCentroids`
- `classifyMultiRow`

Class `CANDY::YinYangGraphIndex`

implement the delete and revise later

Class `CANDY::YinYangGraphSimpleIndex`

implement the delete and revise later

Member [CANDY::YinYangVertex::greedySearchForKNearestTensor](#) (torch::Tensor &src, YinYangVertexPtr entryPoint, int64_t k, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)

This one is just NNDecent greedy policy, perhaps can be better

Member [CANDY::YinYangVertex::greedySearchForKNearestVertex](#) (YinYangVertexPtr src, YinYangVertexPtr entryPoint, int64_t k, bool ignoreYin, bool forceTheSameLevel, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)

This one is just NNDecent greedy policy, perhaps can be better

Member [CANDY::YinYangVertex::greedySearchForKNearestVertex](#) (torch::Tensor &src, YinYangVertexPtr entryPoint, int64_t k, bool ignoreYin, bool forceTheSameLevel, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)

This one is just NNDecent greedy policy, perhaps can be better

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

The support classes for index approaches	23
The data loaders of CANDY	35
The main body of CANDY's indexing approaches	36
The bottom tier of indexing alorithms	36
The upper tier of indexing alorithms, can be container of other indexing ways	39
Shared Utils	41
Other common class or package under C++20 standard	53
Configurations	54
Log utils	55
tensor operations	58
time stamps	58
Energy Meter packs	59
The Micro dataset	60
generic	62
time stamp	65

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

_cl_device_integer_dot_product_acceleration_properties_khr	67
_cl_device_pci_bus_info_khr	68
_cl_icd_dispatch	68
_cl_image_format	72
_cl_mem_android_native_buffer_host_ptr	72
_cl_mem_ext_host_ptr	73
_cl_mem_ion_host_ptr	74
_cl_motion_estimation_desc_intel	75
_cl_name_version_khr	75
_cl_queue_family_properties_intel	76
INTELLI::AbstractC20Thread	77
CANDY::ParallelIndexWorker	330
CANDY::CongestionDropIndexWorker	152
CANDY::AbstractDataLoader	78
CANDY::ExpFamilyDataLoader	187
CANDY::FVECSDataLoader	222
CANDY::HDF5DataLoader	225
CANDY::RandomDataLoader	361
CANDY::ZipfDataLoader	444
CANDY::AbstractIndex	81
CANDY::BucketedFlatIndex	101
CANDY::BufferedCongestionDropIndex	107
CANDY::CongestionDropIndex	141
CANDY::DPGIndex	178
CANDY::DistributedPartitionIndex	166
CANDY::FaissIndex	191
CANDY::FlannIndex	198
CANDY::FlatAMMIPIndex	202
CANDY::FlatAMMIPObjIndex	209
CANDY::FlatIndex	216
CANDY::HNSWNaiveIndex	236
CANDY::NNDescentIndex	303
CANDY::OnlineIVFLSHIndex	317
CANDY::OnlineIVFL2HIndex	313

CANDY::OnlinePQIndex	322
CANDY::PQIndex	347
CANDY::ParallelPartitionIndex	333
CANDY::YinYangGraphIndex	426
CANDY::YinYangGraphSimpleIndex	430
DIVERSE_METER::AbstractMeter	94
DIVERSE_METER::EspMeterUart	186
DIVERSE_METER::IntelMeter	261
CANDY::AdSampling	97
BS::blocks< T1, T2, T >	98
CANDY::FLANN::BranchStruct< T >	100
INTELLI::C20Buffer< dataType >	116
CANDY::CANDYObject	120
cl_char16	121
cl_char2	122
cl_char4	122
cl_char8	122
cl_double16	122
cl_double2	123
cl_double4	123
cl_double8	123
cl_float16	123
cl_float2	124
cl_float4	124
cl_float8	124
cl_half16	124
cl_half2	125
cl_half4	125
cl_half8	125
cl_int16	125
cl_int2	126
cl_int4	126
cl_int8	126
cl_long16	126
cl_long2	127
cl_long4	127
cl_long8	127
cl_short16	127
cl_short2	128
cl_short4	128
cl_short8	128
cl_uchar16	128
cl_uchar2	129
cl_uchar4	129
cl_uchar8	129
cl_uint16	129
cl_uint2	130
cl_uint4	130
cl_uint8	130
cl_ulong16	130
cl_ulong2	131
cl_ulong4	131
cl_ulong8	131
cl_ushort16	131
cl_ushort2	132
cl_ushort4	132
cl_ushort8	132
TONY_CL_HOST::CLContainer	132

CANDY::ClusteringIterationStats	137
CANDY::ClusteringParameters	138
CANDY::Clustering	133
INTELLI::ConfigMap	139
CANDY::DataLoaderTable	154
default_attrs	156
CANDY::FLANN::DistanceIndex	157
CANDY::DistanceQueryer	157
CANDY::DistributedIndexWorker	159
CANDY::DIW_RayWrapper	173
CANDY::FlannComponent	196
CANDY::KdTree	274
CANDY::KmeansTree	280
CANDY::FlannParam	202
CANDY::FLANN::Heap< T >	229
CANDY::HNSW	230
HNSWalter	236
CANDY::HNSWVertex	240
HostPara	242
CANDY::IndexTable	242
INTELLI::IntelliLog	244
INTELLI::IntelliLog_FileProtector	245
INTELLI::IntelliTensorOP	245
INTELLITensorOP	257
INTELLI::IntelliTimeStamp	257
INTELLI::IntelliTimeStampGenerator	258
CANDY::IVFListBucket	263
CANDY::IVFListCell	267
CANDY::IVFTensorEncodingList	269
INTELLI::MemoryTracker	286
DIVERSE_METER::MeterTable	288
INTELLI::MicroDataSet	291
CANDY::HNSW::MinimaxHeap	292
CANDY::MLPBucketIdxModel	293
CANDY::MLPHashingModel	294
BS::multi_future< T >	296
CANDY::DPGIndex::Neighbor	299
CANDY::NNDescentIndex::Neighbor	300
CANDY::NNDescentIndex::Nhoo	300
CANDY::DPGIndex::NhooLayer0	301
CANDY::DPGIndex::NhooLayer1	302
CANDY::KdTree::Node	310
CANDY::KmeansTree::Node	310
CANDY::HNSW::NodeDistCloser	311
CANDY::HNSW::NodeDistFarther	312
CANDY::KmeansTree::NodeInfo	313
INTELLI::ThreadPerf::PerfPair	343
INTELLI::ThreadPerf::PerfTool	344
CANDY::PQDecoder	344
CANDY::PQEncoder	346
CANDY::ProductQuantizer	355
CANDY::FLANN::RandomCenterChooser	360
CANDY::Rania_Python	363
DIVERSE_METER::rapl_power_unit	379
CANDY::FLANN::ResultSet	380
CANDY::SimpleStreamClustering	380
INTELLI::SPSCQueue< T, Allocator >	388
BS::syncd_stream	389

CANDY::TensorIdxPair	392
CANDY::TensorListIdxPair	393
CANDY::TensorStrPair	393
CANDY::TensorStrVecPair	394
BS::thread_pool	395
INTELLI::ThreadPerf	402
INTELLI::ThreadPerfPAPI	407
BS::timer	412
CANDY::FLANN::UniqueRandom	413
INTELLI::UtilityFunctions	413
VertexComparison	416
CANDY::FLANN::VisitBitset	416
VisitedBitset	417
CANDY::VisitedTable	417
CANDY::YinYangGraph	418
CANDY::YinYangGraph_DistanceFunctions	421
CANDY::YinYangGraph_ListBucket	421
CANDY::YinYangGraph_ListCell	424
CANDY::YinYangVertex	433
CANDY::YinYangVertexMap	440

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

_cl_device_integer_dot_product_acceleration_properties_khr	67
_cl_device_pci_bus_info_khr	68
_cl_icd_dispatch	68
_cl_image_format	72
_cl_mem_android_native_buffer_host_ptr	72
_cl_mem_ext_host_ptr	73
_cl_mem_ion_host_ptr	74
_cl_motion_estimation_desc_intel	75
_cl_name_version_khr	75
_cl_queue_family_properties_intel	76
INTELLI::AbstractC20Thread	
The base class and abstraction of C++20 thread, and it can be derived into other threads	77
CANDY::AbstractDataLoader	
The abstract class of data loader, parent for all loaders	78
CANDY::AbstractIndex	
The abstract class of an index approach	81
DIVERSE_METER::AbstractMeter	
The abstract class for all meters	94
CANDY::AdSampling	97
BS::blocks< T1, T2, T >	
A helper class to divide a range into blocks. Used by <code>parallelize_loop()</code> and <code>push_loop()</code>	98
CANDY::FLANN::BranchStruct< T >	
The structure representing a branch point when finding neighbors in the tree	100
CANDY::BucketedFlatIndex	
The class of splitting similar vectors into fixed number of buckets, each bucket is managed by FlatIndex	101
CANDY::BufferedCongestionDropIndex	
Similar to CongestionDropIndex , but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under BucketedFlatIndex More detailed description with an image:	107
INTELLI::C20Buffer< dataType >	116
CANDY::CANDYObject	
A generic object class to link string or void * pointers	120
cl_char16	121
cl_char2	122

cl_char4	122
cl_char8	122
cl_double16	122
cl_double2	123
cl_double4	123
cl_double8	123
cl_float16	123
cl_float2	124
cl_float4	124
cl_float8	124
cl_half16	124
cl_half2	125
cl_half4	125
cl_half8	125
cl_int16	125
cl_int2	126
cl_int4	126
cl_int8	126
cl_long16	126
cl_long2	127
cl_long4	127
cl_long8	127
cl_short16	127
cl_short2	128
cl_short4	128
cl_short8	128
cl_uchar16	128
cl_uchar2	129
cl_uchar4	129
cl_uchar8	129
cl_uint16	129
cl_uint2	130
cl_uint4	130
cl_uint8	130
cl_ulong16	130
cl_ulong2	131
cl_ulong4	131
cl_ulong8	131
cl_ushort16	131
cl_ushort2	132
cl_ushort4	132
cl_ushort8	132
TONY_CL_HOST::CLContainer	132
CANDY::Clustering	
Class for naive K-means clustering	133
CANDY::ClusteringIterationStats	
Struct to record performance of clustering during iterations	137
CANDY::ClusteringParameters	
Class for the clustering parameters to be set before training/building	138
INTELLI::ConfigMap	
The unified map structure to store configurations in a key-value style	139
CANDY::CongestionDropIndex	
A container index to evaluate other bottom index, will just drop the data if congestion occurs, also support the data sharding parallelism	141
CANDY::CongestionDropIndexWorker	
A worker class to container bottom indexings, will just drop new element if congestion occurs	152
CANDY::DataLoaderTable	
The table class to index all Data loaders	154

<code>default_attrs</code>	The low-level perf descriptions passed to OS	156
<code>CANDY::FLANN::DistanceIndex</code>	The structure representing a vectors' distance with the query along with its index	157
<code>CANDY::DistanceQueryer</code>		157
<code>CANDY::DistributedIndexWorker</code>	A worker class of parallel index thread	159
<code>CANDY::DistributedPartitionIndex</code>	A basic distributed index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query	166
<code>CANDY::DIW_RayWrapper</code>	Ray wrapper of <code>DistributedIndexWorker</code> , most of its function will be ray-remote	173
<code>CANDY::DPGIndex</code>	A hierarchical algorithm based on a data structure consistent with <code>NNDescentIndex</code> , the sub-graph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic graph still uses the <code>NNDescent</code> algorithm in this implementation	178
<code>DIVERSE_METER::EspMeterUart</code>	Entity of an esp32s2-based power meter, connected by uart 115200	186
<code>CANDY::ExpFamilyDataLoader</code>	The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta	187
<code>CANDY::FaissIndex</code>	The class of converting faiss index api into rania index style	191
<code>CANDY::FlannComponent</code>		196
<code>CANDY::FlannIndex</code>		198
<code>CANDY::FlannParam</code>		202
<code>CANDY::FlatAMMIPIndex</code>	The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance	202
<code>CANDY::FlatAMMIPObjIndex</code>	Similar to <code>FlatAMMIPIndex</code> , but additionally has object storage (currently only string)	209
<code>CANDY::FlatIndex</code>	The class of a flat index approach, using brutal force management	216
<code>CANDY::FVECSDataLoader</code>	The class for loading *.fvecs data	222
<code>CANDY::HDF5DataLoader</code>	The class for loading *.hdf5 or *.h5 file, as specified in https://github.com/HDFGroup/hdf5	225
<code>CANDY::FLANN::Heap< T ></code>	Heap structure used by <code>FlannIndex</code>	229
<code>CANDY::HNSW</code>	The class of a <code>HNSW</code> structure, maintaining parameters and a vertex entry point	230
<code>HNSWalter</code>		236
<code>CANDY::HNSWNaiveIndex</code>	The class of a <code>HNSW</code> index approach, store the data in each vertex	236
<code>CANDY::HNSWVertex</code>	The class of a <code>HNSW</code> vertex, storing the data in each vertex	240
<code>HostPara</code>		242
<code>CANDY::IndexTable</code>	The table to index index algos	242
<code>INTELLI::IntelliLog</code>	The log functions packed in class	244
<code>INTELLI::IntelliLog_FileProtector</code>	The protector for concurrent log on a file	245
<code>INTELLI::IntelliTensorOP</code>		245
<code>INTELLITensorOP</code>	The common tensor functions packed in class	257

INTELLI::IntelliTimeStamp	
The class to define a timestamp	257
INTELLI::IntelliTimeStampGenerator	
The basic class to generate time stamps	258
DIVERSE_METER::IntelMeter	
Entity of intel msr-based power meter, may be not support for some newer architectures	261
CANDY::IVFListBucket	
Bucket of multiple IVFListCell	263
CANDY::IVFListCell	
Cell of row tensor pointers which have the same code	267
CANDY::IVFTensorEncodingList	
The inverted file (IVF) list to organize tensor and its encodings	269
CANDY::KdTree	274
CANDY::KmeansTree	
The structure representing hierarchical k-means tree used in FLANN	280
INTELLI::MemoryTracker	
The top entity to trace current, average and maximum memory foot print	286
DIVERSE_METER::MeterTable	
The table class to index all meters	288
INTELLI::MicroDataSet	
The all-in-one class for the Micro dataset	291
CANDY::HNSW::MinimaxHeap	
Tiny heap that is used during search	292
CANDY::MLPBucketIdxModel	293
CANDY::MLPHashingModel	294
BS::multi_future< T >	
A helper class to facilitate waiting for and/or getting the results of multiple futures at once	296
CANDY::DPGIndex::Neighbor	299
CANDY::NNDescentIndex::Neighbor	300
CANDY::NNDescentIndex::Nhood	300
CANDY::DPGIndex::NhoodLayer0	301
CANDY::DPGIndex::NhoodLayer1	302
CANDY::NNDescentIndex	
An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized	303
CANDY::KdTree::Node	310
CANDY::KmeansTree::Node	310
CANDY::HNSW::NodeDistCloser	
Sort pairs from nearest to farthest by distance	311
CANDY::HNSW::NodeDistFarther	
Sort pairs from farthest to nearest	312
CANDY::KmeansTree::NodeInfo	313
CANDY::OnlineIVFL2HIndex	
A L2H (learning 2 hash) indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The L2H function is using ML to approximate spectral hashing principles (NIPS 2008)	313
CANDY::OnlineIVFLSHIndex	
A LSH indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The LSH function is the vanilla random projection (gaussian or random matrix)	317
CANDY::OnlinePQIndex	
The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers	322
CANDY::ParallelIndexWorker	
A worker class of parallel index thread	330

CANDY::ParallelPartitionIndex	
A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature	333
INTELLI::ThreadPerf::PerfPair	
Record pair of perf events	343
INTELLI::ThreadPerf::PerfTool	
.	344
CANDY::PQDecoder	
Class for decoding from codes, approximated assignment of centroids, to centroids indices	344
CANDY::PQEncoder	
Class for encoding input vectors to codes, standing for approximated assignment of centroids	346
CANDY::PQIndex	
Class for indexing vectors using product quantizations, this is a raw implementation without hierarchical	347
CANDY::ProductQuantizer	
Class for basic product quantization operations on input of tensors	355
CANDY::FLANN::RandomCenterChooser	
The class used in hierarchical kmeans tree to choose center	360
CANDY::RandomDataLoader	
The class of ranom data loader,	361
CANDY::Rania_Python	
The python bounding functions	363
DIVERSE_METER::rapl_power_unit	
.	379
CANDY::FLANN::ResultSet	
Priority queue used in FlannIndex	380
CANDY::SimpleStreamClustering	
Simple class for stream clustering, following online PQ style and using simple linear equations	380
INTELLI::SPSCQueue< T, Allocator >	
.	388
BS::synced_stream	
A helper class to synchronize printing to an output stream by different threads	389
CANDY::TensorIdxPair	
The class to define a tensor along with some idx	392
CANDY::TensorListIdxPair	
.	393
CANDY::TensorStrPair	
.	393
CANDY::TensorStrVecPair	
.	394
BS::thread_pool	
A fast, lightweight, and easy-to-use C++17 thread pool class	395
INTELLI::ThreadPerf	
The top entity to provide perf traces, please use this class only UNLESS you know what you are doing	402
INTELLI::ThreadPerfPAPI	
The top entity to provide perf traces by using PAPI lib	407
BS::timer	
A helper class to measure execution time for benchmarking purposes	412
CANDY::FLANN::UniqueRandom	
The class to output unique random values	413
INTELLI::UtilityFunctions	
.	413
VertexComparison	
.	416
CANDY::FLANN::VisitBitset	
.	416
VisitedBitset	
The visited array of nodes	417
CANDY::VisitedTable	
.	417
CANDY::YinYangGraph	
The top class of yinyang graph, containing ivf list and critical graph information	418
CANDY::YinYangGraph_DistanceFunctions	
.	421
CANDY::YinYangGraph_ListBucket	
Bucket of multiple YinYangGraph_ListCell	421

CANDY::YinYangGraph_ListCell	
Cell of an ending YinYangVertex	424
CANDY::YinYangGraphIndex	
The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph	426
CANDY::YinYangGraphSimpleIndex	
The class of indexing using a simpe yinyang graph,there is no LSH search is only within the linked yinyanggraph	430
CANDY::YinYangVertex	
The class of a YinYangVertex , storing the data in each vertex	433
CANDY::YinYangVertexMap	440
CANDY::ZipfDataLoader	
The class to load zipf data	444

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

include/CANDY.h	449
include/CANDYPYTHON.h	474
include/CANDY/AbstractIndex.h	450
include/CANDY/BucketedFlatIndex.h	451
include/CANDY/BufferedCongestionDropIndex.h	452
include/CANDY/CANDYObject.h	453
include/CANDY/CongestionDropIndex.h	454
include/CANDY/DistributedPartitionIndex.h	456
include/CANDY/DPGIndex.h	458
include/CANDY/FaissIndex.h	459
include/CANDY/FlannIndex.h	??
include/CANDY/FlatAMMIPIndex.h	459
include/CANDY/FlatAMMIPObjIndex.h	460
include/CANDY/FlatIndex.h	461
include/CANDY/HNSWNaiveIndex.h	462
include/CANDY/IndexTable.h	463
include/CANDY/NNDescentIndex.h	464
include/CANDY/OnlineIVFL2HIndex.h	??
include/CANDY/OnlineIVFLSHIndex.h	??
include/CANDY/OnlinePQIndex.h	465
include/CANDY/ParallelPartitionIndex.h	468
include/CANDY/PQIndex.h	??
include/CANDY/YinYangGraphIndex.h	471
include/CANDY/YinYangGraphSimpleIndex.h	473
include/CANDY/CongestionDropIndex/CongestionDropIndexWorker.h	455
include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h	457
include/CANDY/FlannIndex/FlannComponent.h	??
include/CANDY/FlannIndex/FlannUtils.h	??
include/CANDY/FlannIndex/KdTree.h	??
include/CANDY/FlannIndex/Kmeans.h	??
include/CANDY/HashingModels/MLPBucketIdxModel.h	??
include/CANDY/HashingModels/MLPHashingModel.h	??
include/CANDY/HNSWNaive/AdSampling.h	??
include/CANDY/HNSWNaive/DistanceQueryer.h	??
include/CANDY/HNSWNaive/HNSW.h	??

include/CANDY/HNSWNaive/HNSWAlter.h	??
include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h	466
include/CANDY/OnlinePQIndex/SimpleStreamClustering.h	467
include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h	469
include/CANDY/PQIndex/Clustering.h	??
include/CANDY/YinYangGraphIndex/YinYangGraph.h	472
include/CL/cl.h	??
include/CL/cl_d3d10.h	??
include/CL/cl_d3d11.h	??
include/CL/cl_dx9_media_sharing.h	??
include/CL/cl_dx9_media_sharing_intel.h	??
include/CL/cl_egl.h	??
include/CL/cl_ext.h	??
include/CL/cl_ext_intel.h	??
include/CL/cl_gl.h	??
include/CL/cl_gl_ext.h	??
include/CL/cl_half.h	??
include/CL/cl_icd.h	??
include/CL/cl_layer.h	??
include/CL/cl_platform.h	??
include/CL/cl_va_api_media_sharing_intel.h	??
include/CL/cl_version.h	??
include/CL/CLContainer.hpp	??
include/CL/opencl.h	??
include/DataLoader/AbstractDataLoader.h	475
include/DataLoader/DataLoaderTable.h	475
include/DataLoader/ExpFamilyDataLoader.h	477
include/DataLoader/FVECSDataLoader.h	478
include/DataLoader/HDF5DataLoader.h	479
include/DataLoader/RandomDataLoader.h	480
include/DataLoader/ZipfDataLoader.h	481
include/Utils/AbstractC20Thread.hpp	482
include/Utils/BS_thread_pool.hpp	
BS::thread_pool: a fast, lightweight, and easy-to-use C++17 thread pool library. This header file contains the entire library, including the main BS::thread_pool class and the helper classes BS::multi_future, BS::blocks, BS::synced_stream, and BS::timer	
	483
include/Utils/C20Buffers.hpp	484
include/Utils/ConfigMap.hpp	485
include/Utils/IntelliLog.h	??
include/Utils/IntelliTensorOP.hpp	486
include/Utils/IntelliTimeStampGenerator.h	487
include/Utils/MemTracker.h	??
include/Utils/MicroDataSet.hpp	??
include/Utils/SPSCQueue.hpp	??
include/Utils/ThreadPerf.hpp	490
include/Utils/ThreadPerfPAPI.hpp	492
include/Utils/UtilityFunctions.h	??
include/Utils/Meters/AbstractMeter.hpp	488
include/Utils/Meters/MeterTable.h	??
include/Utils/Meters/EspMeterUart/EspMeterUart.hpp	489
include/Utils/Meters/IntelMeter/IntelMeter.hpp	??
src/CANDY/AbstractIndex.cpp	493
src/CANDY/BucketedFlatIndex.cpp	493
src/CANDY/BufferedCongestionDropIndex.cpp	494
src/CANDY/CongestionDropIndex.cpp	494
src/CANDY/DistributedPartitionIndex.cpp	495
src/CANDY/DPGIndex.cpp	496
src/CANDY/FlatAMMIPIndex.cpp	496

src/CANDY/FlatAMMIPObjIndex.cpp	496
src/CANDY/FlatIndex.cpp	497
src/CANDY/NNDescentIndex.cpp	497
src/CANDY/OnlineVFL2HIndex.cpp	498
src/CANDY/OnlineVFLSHIndex.cpp	498
src/CANDY/OnlinePQIndex.cpp	499
src/CANDY/ParallelPartitionIndex.cpp	500
src/CANDY/YinYangGraphIndex.cpp	500
src/CANDY/YinYangGraphSimpleIndex.cpp	501
src/CANDY/CongestionDropIndex/CongestionDropIndexWorker.cpp	495
src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp	495
src/CANDY/ParallelPartitionIndex/ParallelIndexWorker.cpp	500

Chapter 7

Module Documentation

7.1 The support classes for index approaches

Classes

- class [CANDY::CongestionDropIndexWorker](#)
A worker class to container bottom indexings, will just drop new element if congestion occurs.
- class [CANDY::MLPBucketIdxModel](#)
- class [CANDY::MLPHashingModel](#)
- class [CANDY::ParallelIndexWorker](#)
A worker class of parallel index thread.
- class [CANDY::DIW_RayWrapper](#)
the ray wrapper of [DistributedIndexWorker](#), most of its function will be ray-remote
- class [CANDY::IVFListCell](#)
a cell of row tensor pointers which have the same code
- class [CANDY::SimpleStreamClustering](#)
a simple class for stream clustering, following online PQ style and using simple linear equations

Macros

- `#define newYinYangVertex make_shared<CANDY::YinYangVertex>`
(Macro) To creat a new [YinYangVertex](#) under shared pointer.
- `#define newYinYangGraph_ListCell make_shared<CANDY::YinYangGraph_ListCell>`
(Macro) To creat a new [newYinYangGraph_ListCell](#) under shared pointer.
- `#define newYinYangGraph_ListBucket make_shared<CANDY::YinYangGraph_ListBucket>`
(Macro) To creat a new [YinYangGraph_ListBucket](#) under shared pointer.

Typedefs

- `typedef std::shared_ptr< CANDY::YinYangVertex > CANDY::YinYangVertexPtr`
The class to describe a shared pointer to [YinYangVertex](#).
- `typedef std::shared_ptr< CANDY::YinYangGraph_ListCell > CANDY::YinYangGraph_ListCellPtr`
The class to describe a shared pointer to [YinYangGraph_ListCell](#).
- `typedef std::shared_ptr< CANDY::YinYangGraph_ListBucket > CANDY::YinYangGraph_ListBucketPtr`
The class to describe a shared pointer to [YinYangGraph_ListBucket](#).

Functions

- virtual bool [CANDY::CongestionDropIndexWorker::insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [CANDY::CongestionDropIndexWorker::setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual std::vector< torch::Tensor > [CANDY::CongestionDropIndexWorker::searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- **CANDY::TensorIdxPair::TensorIdxPair** (torch::Tensor _t, int64_t _idx)
- **CANDY::TensorListIdxPair::TensorListIdxPair** (std::vector< torch::Tensor > &t, int64_t _idx, int64_t _↔ seq)
- **CANDY::TensorStrPair::TensorStrPair** (torch::Tensor _t, int64_t _idx)
- **CANDY::TensorStrPair::TensorStrPair** (torch::Tensor _t, int64_t _idx, std::vector< std::string > &str)
- **CANDY::TensorStrVecPair::TensorStrVecPair** (std::vector< torch::Tensor > &t, int64_t _idx, int64_t _↔ seq, std::vector< std::vector< std::string >> str)
- **CANDY::TensorStrVecPair::TensorStrVecPair** (std::vector< torch::Tensor > &t, int64_t _idx, int64_t _↔ seq)
- virtual void [CANDY::ParallelIndexWorker::inlineMain](#) ()
The inline "main" function of thread, as an interface.
- virtual void **CANDY::ParallelIndexWorker::setReduceQueue** (TensorListIdxQueuePtr rq)
- virtual void **CANDY::ParallelIndexWorker::setReduceStrQueue** (TensorStrVecQueuePtr rq)
- virtual void **CANDY::ParallelIndexWorker::setId** (int64_t _id)
- virtual bool **CANDY::ParallelIndexWorker::waitPendingOperations** ()
- virtual bool [CANDY::ParallelIndexWorker::loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [CANDY::ParallelIndexWorker::reset](#) ()
reset this index to initated status
- virtual bool [CANDY::ParallelIndexWorker::setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [CANDY::ParallelIndexWorker::startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [CANDY::ParallelIndexWorker::insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [CANDY::ParallelIndexWorker::deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [CANDY::ParallelIndexWorker::reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > [CANDY::ParallelIndexWorker::searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [CANDY::ParallelIndexWorker::getTensorByIndex](#) (std::vector< faiss::↔ :idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [CANDY::ParallelIndexWorker::rawData](#) ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > [CANDY::ParallelIndexWorker::searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [CANDY::ParallelIndexWorker::endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [CANDY::ParallelIndexWorker::setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [CANDY::ParallelIndexWorker::offlineBuild](#) (torch::Tensor &t)

offline build phase

- virtual void **CANDY::ParallelIndexWorker::pushSearch** (torch::Tensor q, int64_t k)
- virtual void **CANDY::ParallelIndexWorker::pushSearchStr** (torch::Tensor q, int64_t k)
- virtual bool **CANDY::ParallelIndexWorker::loadInitialStringObject** (torch::Tensor &t, std::vector< std::string > &strs)

load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)

- virtual bool **CANDY::ParallelIndexWorker::insertStringObject** (torch::Tensor &t, std::vector< std::string > &strs)

insert a string object

- virtual bool **CANDY::ParallelIndexWorker::deleteStringObject** (torch::Tensor &t, int64_t k=1)

delete tensor along with its corresponding string object

- virtual std::vector< std::vector< std::string > > **CANDY::ParallelIndexWorker::searchStringObject** (torch::Tensor &q, int64_t k)

search the k-NN of a query tensor, return the linked string objects

- virtual std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > **CANDY::ParallelIndexWorker::searchStringObjectAndTensors** (torch::Tensor &q, int64_t k)

search the k-NN of a query tensor, return the linked string objects and original tensors

Variables

- int64_t **CANDY::CongestionDropIndexWorker::forceDrop** = 1
- TensorListIdxQueuePtr **CANDY::CongestionDropIndexWorker::reduceQueue**
- torch::Tensor **CANDY::TensorIdxPair::t**
- int64_t **CANDY::TensorIdxPair::idx**
- std::vector< torch::Tensor > **CANDY::TensorListIdxPair::t**
- int64_t **CANDY::TensorListIdxPair::idx**
- int64_t **CANDY::TensorListIdxPair::querySeq**
- torch::Tensor **CANDY::TensorStrPair::t**
- int64_t **CANDY::TensorStrPair::idx**
- std::vector< std::string > **CANDY::TensorStrPair::strObj**
- std::vector< torch::Tensor > **CANDY::TensorStrVecPair::t**
- int64_t **CANDY::TensorStrVecPair::idx**
- int64_t **CANDY::TensorStrVecPair::querySeq**
- std::vector< std::vector< std::string > > **CANDY::TensorStrVecPair::strObjs**
- TensorQueuePtr **CANDY::ParallelIndexWorker::insertQueue**
- TensorQueuePtr **CANDY::ParallelIndexWorker::reviseQueue0**
- TensorQueuePtr **CANDY::ParallelIndexWorker::reviseQueue1**
- TensorQueuePtr **CANDY::ParallelIndexWorker::buildQueue**
- TensorQueuePtr **CANDY::ParallelIndexWorker::initialLoadQueue**
- TensorIdxQueuePtr **CANDY::ParallelIndexWorker::deleteQueue**
- TensorIdxQueuePtr **CANDY::ParallelIndexWorker::queryQueue**
- TensorIdxQueuePtr **CANDY::ParallelIndexWorker::deleteStrQueue**
- TensorStrQueuePtr **CANDY::ParallelIndexWorker::initialStrQueue**
- TensorStrQueuePtr **CANDY::ParallelIndexWorker::insertStrQueue**
- TensorIdxQueuePtr **CANDY::ParallelIndexWorker::queryStrQueue**
- CmdQueuePtr **CANDY::ParallelIndexWorker::cmdQueue**
- int64_t **CANDY::ParallelIndexWorker::myId** = 0
- int64_t **CANDY::ParallelIndexWorker::vecDim** = 0
- int64_t **CANDY::ParallelIndexWorker::congestionDrop** = 1
- int64_t **CANDY::ParallelIndexWorker::ingestedVectors** = 0
- int64_t **CANDY::ParallelIndexWorker::singleWorkerOpt**
- std::mutex **CANDY::ParallelIndexWorker::m_mut**
- [AbstractIndexPtr](#) **CANDY::ParallelIndexWorker::myIndexAlgo** = nullptr

- TensorListIdxQueuePtr **CANDY::ParallelIndexWorker::reduceQueue**
- TensorStrVecQueuePtr **CANDY::ParallelIndexWorker::reduceStrQueue**
- typedef std::shared_ptr< class **CANDY::MLPBucketIdxModel** > **CANDY::MLPBucketIdxModelPtr**
*The class to describe a shared pointer to **MLPBucketIdxModel**.*
- #define **newMLPBucketIdxModel** std::make_shared<**CANDY::MLPBucketIdxModel**>
(Macro) To creat a new MLPBucketIdxModel shared pointer.
- typedef std::shared_ptr< class **CANDY::MLPHashingModel** > **CANDY::MLPHashingModelPtr**
*The class to describe a shared pointer to **MLPHashingModel**.*
- #define **newMLPHashingModel** std::make_shared<**CANDY::MLPHashingModel**>
(Macro) To creat a new MLPHashingModel shared pointer.
- typedef std::shared_ptr< **CANDY::IVFListCell** > **CANDY::IVFListCellPtr**
*The class to describe a shared pointer to **IVFListCell**.*
- typedef std::shared_ptr< **CANDY::IVFListBucket** > **CANDY::IVFListBucketPtr**
*The class to describe a shared pointer to **IVFListBucket**.*
- #define **newIVFListCell** make_shared<**CANDY::IVFListCell**>
*(Macro) To creat a new **newIVFListCell** under shared pointer.*
- #define **newIVFListBucket** make_shared<**CANDY::IVFListBucket**>
(Macro) To creat a new IVFListBucket under shared pointer.
- typedef std::shared_ptr< **CANDY::SimpleStreamClustering** > **CANDY::SimpleStreamClusteringPtr**
*The class to describe a shared pointer to **SimpleStreamClustering**.*
- #define **newSimpleStreamClustering** make_shared<**CANDY::SimpleStreamClustering**>
(Macro) To creat a new SimpleStreamClustering under shared pointer.

7.1.1 Detailed Description

7.1.2 Function Documentation

7.1.2.1 deleteStringObject()

```
bool CANDY::ParallelIndexWorker::deleteStringObject (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete tensor along with its corresponding string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the delet is successful

7.1.2.2 deleteTensor()

```
bool CANDY::ParallelIndexWorker::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

7.1.2.3 endHPC()

```
bool CANDY::ParallelIndexWorker::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

7.1.2.4 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::ParallelIndexWorker::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

7.1.2.5 inlineMain()

```
void CANDY::ParallelIndexWorker::inlineMain ( ) [protected], [virtual]
```

The inline 'main' function of thread, as an interface.

Note

Normally re-write this in derived classes

0. offline stages

1. insert first
2. revise
3. delete first
4. query
5. terminate

Reimplemented from [INTELLI::AbstractC20Thread](#).

7.1.2.6 insertStringObject()

```
bool CANDY::ParallelIndexWorker::insertStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

insert a string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>strs</i>	the corresponding list of strings

Returns

bool whether the insertion is successful

7.1.2.7 insertTensor() [1/2]

```
bool CANDY::CongestionDropIndexWorker::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::ParallelIndexWorker](#).

7.1.2.8 insertTensor() [2/2]

```
bool CANDY::ParallelIndexWorker::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented in [CANDY::CongestionDropIndexWorker](#).

7.1.2.9 loadInitialStringObject()

```
bool CANDY::ParallelIndexWorker::loadInitialStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row •
<i>strs</i>	the corresponding list of strings

Returns

bool whether the loading is successful

7.1.2.10 loadInitialTensor()

```
bool CANDY::ParallelIndexWorker::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

7.1.2.11 offlineBuild()

```
bool CANDY::ParallelIndexWorker::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

7.1.2.12 rawData()

```
torch::Tensor CANDY::ParallelIndexWorker::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

7.1.2.13 reviseTensor()

```
bool CANDY::ParallelIndexWorker::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

7.1.2.14 searchIndex()

```
std::vector< faiss::idx_t > CANDY::ParallelIndexWorker::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<faiss::idx_t>` the index, follow faiss's order

7.1.2.15 searchStringObject()

```
std::vector< std::vector< std::string > > CANDY::ParallelIndexWorker::searchStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<std::vector<std::string>>` the result object for each row of query

7.1.2.16 searchTensor() [1/2]

```
std::vector< torch::Tensor > CANDY::CongestionDropIndexWorker::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

Reimplemented from [CANDY::ParallelIndexWorker](#).

7.1.2.17 searchTensor() [2/2]

```
std::vector< torch::Tensor > CANDY::ParallelIndexWorker::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented in [CANDY::CongestionDropIndexWorker](#).

7.1.2.18 searchTensorAndStringObject()

```
std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > > CANDY↔
::ParallelIndexWorker::searchTensorAndStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects and original tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::tuple<std::vector<torch::Tensor>,std::vector<std::vector<std::string>>>

7.1.2.19 setConfig() [1/2]

```
bool CANDY::CongestionDropIndexWorker::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

1. find the index algo
2. set up the queues

Reimplemented from [CANDY::ParallelIndexWorker](#).

7.1.2.20 setConfig() [2/2]

```
bool CANDY::ParallelIndexWorker::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

1. find the index algo
2. set up the queues

Reimplemented in [CANDY::CongestionDropIndexWorker](#).

7.1.2.21 setFrozenLevel()

```
bool CANDY::ParallelIndexWorker::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

7.1.2.22 startHPC()

```
bool CANDY::ParallelIndexWorker::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

7.2 The data loaders of CANDY**Classes**

- class [CANDY::AbstractDataLoader](#)
The abstract class of data loader, parent for all loaders.
- class [CANDY::DataLoaderTable](#)
The table class to index all Data loaders.
- class [CANDY::ExpFamilyDataLoader](#)
The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta.
- class [CANDY::FVECSDataLoader](#)
*The class for loading *.fvecs data.*
- class [CANDY::HDF5DataLoader](#)
*The class for loading *.hdf5 or *.h5 file, as specified in <https://github.com/HDFGroup/hdf5>.*
- class [CANDY::RandomDataLoader](#)
The class of ranom data loader,.
- class [CANDY::ZipfDataLoader](#)
The class to load zipf data.

7.2.1 Detailed Description**7.2.1.1 DataLoader**

This folder contains the loader under different generation rules

We define the generation classes of DATA. here

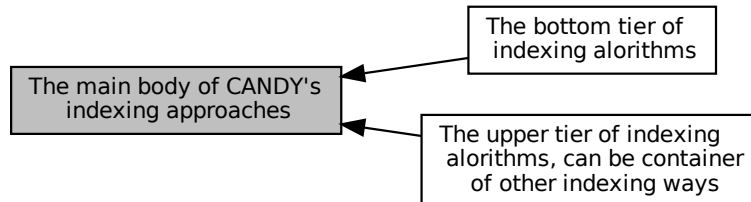
7.2.1.2 DataLoader

This folder contains the dataloader

We define the data loader classes . here

7.3 The main body of CANDY's indexing approaches

Collaboration diagram for The main body of CANDY's indexing approaches:



Modules

- [The bottom tier of indexing alorithms](#)
- [The upper tier of indexing alorithms, can be container of other indexing ways](#)

Classes

- class [CANDY::IndexTable](#)
The table to index index algos.

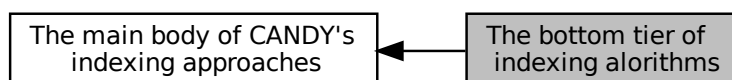
7.3.1 Detailed Description

7.3.1.1 BODY

This folder contains the main body

7.4 The bottom tier of indexing alorithms

Collaboration diagram for The bottom tier of indexing alorithms:



Classes

- class [CANDY::AbstractIndex](#)
The abstract class of an index approach.
- class [CANDY::BucketedFlatIndex](#)
The class of splitting similar vectors into fixed number of buckets, each bucket is managed by [FlatIndex](#).
- class [CANDY::CANDYObject](#)
*A generic object class to link string or void * pointers.*
- class [CANDY::FaissIndex](#)
The class of converting faiss index api into rania index style.
- class [CANDY::FlatAMMIPIndex](#)
The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance.
- class [CANDY::FlatAMMIPObjIndex](#)
Similar to [FlatAMMIPIndex](#), but additionally has object storage (currently only string)
- class [CANDY::FlatIndex](#)
The class of a flat index approach, using brutal force management.
- class [CANDY::OnlineIVFL2HIndex](#)
A L2H (learning 2 hash) indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The L2H function is using ML to approximate spectral hashing principles (NIPS 2008)
- class [CANDY::OnlineIVFLSHIndex](#)
A LSH indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The LSH function is the vanilla random projection (gaussian or random matrix).
- class [CANDY::OnlinePQIndex](#)
The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers.
- class [CANDY::PQIndex](#)
class for indexing vectors using product quantizations, this is a raw implementation without hierarchical
- class [CANDY::YinYangGraphIndex](#)
The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph.
- class [CANDY::YinYangGraphSimpleIndex](#)
The class of indexing using a simple yinyang graph, there is no LSH search is only within the linked yinyanggraph.

Macros

- `#define newParallelIndexWorker std::make_shared<CANDY::ParallelIndexWorker>`
(Macro) To create a new ParallelIndexWorker shared pointer.
- `#define newPQIndex std::make_shared<CANDY::PQIndex>`
(Macro) To create a new PQIndex shared pointer.

Typedefs

- `typedef std::shared_ptr< class CANDY::ParallelIndexWorker > CANDY::ParallelIndexWorkerPtr`
The class to describe a shared pointer to [ParallelIndexWorker](#).
- `typedef std::shared_ptr< class CANDY::PQIndex > CANDY::PQIndexPtr`
The class to describe a shared pointer to [PQIndex](#).
- `#define newAbstractIndex std::make_shared<CANDY::AbstractIndex>`

- (Macro) To creat a new AbstractIndex shared pointer.*

 - typedef std::shared_ptr< class [CANDY::CANDYObject](#) > [CANDY::CANDYObjectPtr](#)
The class to describe a shared pointer to [CANDYObject](#).
- #define [newBucketedFlatIndex](#) std::make_shared<[CANDY::BucketedFlatIndex](#)>
(Macro) To creat a new BucketedFlatIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::BucketedFlatIndex](#) > [CANDY::BucketedFlatIndexPtr](#)
The class to describe a shared pointer to [BucketedFlatIndex](#).
- #define [newDistributedIndexWorker](#) std::make_shared<[CANDY::DistributedIndexWorker](#)>
(Macro) To creat a new DistributedIndexWorker shared pointer.

 - typedef std::shared_ptr< class [CANDY::DistributedIndexWorker](#) > [CANDY::DistributedIndexWorkerPtr](#)
The class to describe a shared pointer to [DistributedIndexWorker](#).
- #define [newFaissIndex](#) std::make_shared<[CANDY::FaissIndex](#)>
(Macro) To creat a new FaissIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::FaissIndex](#) > [CANDY::FaissIndexPtr](#)
The class to describe a shared pointer to [FaissIndexPtr](#).
- #define [newFlatAMMIPIndex](#) std::make_shared<[CANDY::FlatAMMIPIndex](#)>
(Macro) To creat a new FlatAMMIPIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::FlatAMMIPIndex](#) > [CANDY::FlatAMMIPIndexPtr](#)
The class to describe a shared pointer to [FlatAMMIPIndex](#).
- #define [newFlatAMMIPObjIndex](#) std::make_shared<[CANDY::FlatAMMIPObjIndex](#)>
(Macro) To creat a new FlatAMMIPObjIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::FlatAMMIPObjIndex](#) > [CANDY::FlatAMMIPObjIndexPtr](#)
The class to describe a shared pointer to [FlatAMMIPObjIndex](#).
- #define [newFlatIndex](#) std::make_shared<[CANDY::FlatIndex](#)>
(Macro) To creat a new FlatIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::FlatIndex](#) > [CANDY::FlatIndexPtr](#)
The class to describe a shared pointer to [FlatIndex](#).
- #define [newOnlineIVFL2HIndex](#) std::make_shared<[CANDY::OnlineIVFL2HIndex](#)>
(Macro) To creat a new OnlineIVFL2HIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::OnlineIVFL2HIndex](#) > [CANDY::OnlineIVFL2HIndexPtr](#)
The class to describe a shared pointer to [OnlineIVFL2HIndex](#).
- #define [newOnlineIVFLSHIndex](#) std::make_shared<[CANDY::OnlineIVFLSHIndex](#)>
(Macro) To creat a new OnlineIVFLSHIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::OnlineIVFLSHIndex](#) > [CANDY::OnlineIVFLSHIndexPtr](#)
The class to describe a shared pointer to [OnlineIVFLSHIndex](#).
- #define [newOnlinePQIndex](#) std::make_shared<[CANDY::OnlinePQIndex](#)>
(Macro) To creat a new OnlinePQIndex shared pointer.

 - typedef std::shared_ptr< class [CANDY::OnlinePQIndex](#) > [CANDY::OnlinePQIndexPtr](#)

The class to describe a shared pointer to [OnlinePQIndex](#).

- `#define newYinYangGraphIndex std::make_shared<CANDY::YinYangGraphIndex>`
(Macro) To creat a new `YinYangGraphIndex` shared pointer.
- `typedef std::shared_ptr< class CANDY::YinYangGraphIndex > CANDY::YinYangGraphIndexPtr`
The class to describe a shared pointer to `YinYangGraphIndex`.
- `#define newYinYangGraphSimpleIndex std::make_shared<CANDY::YinYangGraphSimpleIndex>`
(Macro) To creat a new `YinYangGraphSimpleIndex` shared pointer.
- `typedef std::shared_ptr< class CANDY::YinYangGraphSimpleIndex > CANDY::YinYangGraphSimpleIndexPtr`
The class to describe a shared pointer to `YinYangGraphSimpleIndex`.
- `typedef std::shared_ptr< class CANDY::AbstractIndex > CANDY::AbstractIndexPtr`
The class to describe a shared pointer to `AbstractIndex`.

7.4.1 Detailed Description

7.4.2 Macro Definition Documentation

7.4.2.1 newAbstractIndex

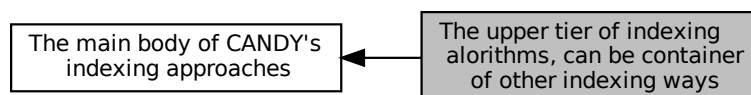
```
#define newAbstractIndex std::make_shared<CANDY::AbstractIndex>
```

(Macro) To creat a new `AbstractIndex` shared pointer.

(Macro) To creat a new `CANDYObject` shared pointer.

7.5 The upper tier of indexing alorithms, can be container of other indexing ways

Collaboration diagram for The upper tier of indexing alorithms, can be container of other indexing ways:



Classes

- class [CANDY::BufferedCongestionDropIndex](#)
Similar to [CongestionDropIndex](#), but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under [BucketedFlatIndex](#) More detailed description with an image:
- class [CANDY::CongestionDropIndex](#)
A container index to evaluate other bottom index, will just drop the data if congestion occurs, also support the data sharding parallelism.
- class [CANDY::DistributedPartitionIndex](#)
A basic distributed index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query.
- class [CANDY::DPGIndex](#)
A hierarchical algorithm based on a data structure consistent with [NNDescentIndex](#), the subgraph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic graph still uses the NNDescent algorithm in this implementation.
- class [CANDY::NNDescentIndex](#)
An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized.
- class [CANDY::ParallelPartitionIndex](#)
A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature.

Macros

- `#define newCongestionDropIndexWorker std::make_shared<CANDY::CongestionDropIndexWorker>`
(Macro) To creat a new CongestionDropIndexWorker shared pointer.

Typedefs

- `typedef std::shared_ptr< class CANDY::CongestionDropIndexWorker > CANDY::CongestionDropIndexWorkerPtr`
The class to describe a shared pointer to [CongestionDropIndexWorker](#).
- `#define newBufferedCongestionDropIndex std::make_shared<CANDY::BufferedCongestionDropIndex>`
(Macro) To creat a new BufferedCongestionDropIndex shared pointer.
- `typedef std::shared_ptr< class CANDY::BufferedCongestionDropIndex > CANDY::BufferedCongestionDropIndexPtr`
The class to describe a shared pointer to [BufferedCongestionDropIndex](#).
- `#define newCongestionDropIndex std::make_shared<CANDY::CongestionDropIndex>`
(Macro) To creat a new CongestionDropIndex shared pointer.
- `typedef std::shared_ptr< class CANDY::CongestionDropIndex > CANDY::CongestionDropIndexPtr`
The class to describe a shared pointer to [CongestionDropIndex](#).
- `#define newDistributedPartitionIndex std::make_shared<CANDY::DistributedPartitionIndex>`
(Macro) To creat a new DistributedPartitionIndex shared pointer.
- `typedef std::shared_ptr< class CANDY::DistributedPartitionIndex > CANDY::DistributedPartitionIndexPtr`
The class to describe a shared pointer to [DistributedPartitionIndex](#).
- `#define newDPGIndex std::make_shared<CANDY::DPGIndex>`

- (Macro) To creat a new *DPGIndex* shared pointer.

 - typedef std::shared_ptr< class [CANDY::DPGIndex](#) > [CANDY::DPGIndexPtr](#)
 - The class to describe a shared pointer to *DPGIndex*.
- #define [newNNDescentIndex](#) std::make_shared<[CANDY::NNDescentIndex](#)>

(Macro) To creat a new *NNDescentIndex* shared pointer.

 - typedef std::shared_ptr< class [CANDY::NNDescentIndex](#) > [CANDY::NNDescentIndexPtr](#)
 - The class to describe a shared pointer to *NNDescentIndex*.
- #define [newParallelPartitionIndex](#) std::make_shared<[CANDY::ParallelPartitionIndex](#)>

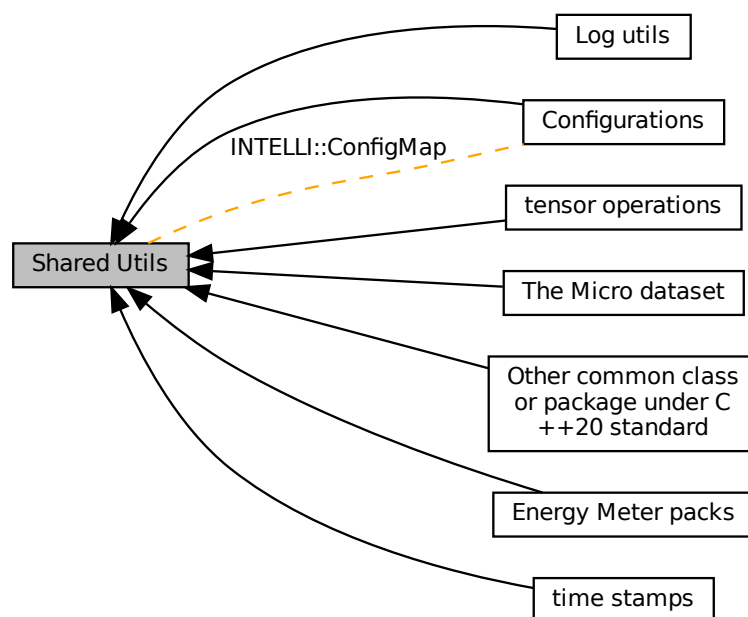
(Macro) To creat a new *ParallelPartitionIndex* shared pointer.

 - typedef std::shared_ptr< class [CANDY::ParallelPartitionIndex](#) > [CANDY::ParallelPartitionIndexPtr](#)
 - The class to describe a shared pointer to *ParallelPartitionIndex*.

7.5.1 Detailed Description

7.6 Shared Utils

Collaboration diagram for Shared Utils:



Modules

- [Other common class or package under C++20 standard](#)
- [Configurations](#)
- [Log utils](#)
- [tensor operations](#)
- [time stamps](#)
- [Energy Meter packs](#)
- [The Micro dataset](#)

Classes

- class [INTELLI::ConfigMap](#)

The unified map structure to store configurations in a key-value style.

Functions

- static void [INTELLI::ConfigMap::spilt](#) (const std::string s, const std::string &c, vector< std::string > &v)
- void [INTELLI::ConfigMap::smartParase](#) (std::string key, std::string value)
- void [INTELLI::ConfigMap::edit](#) (const std::string &key, uint64_t value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void [INTELLI::ConfigMap::edit](#) (const std::string &key, int64_t value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void [INTELLI::ConfigMap::edit](#) (const std::string &key, double value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void [INTELLI::ConfigMap::edit](#) (const std::string &key, std::string value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- bool [INTELLI::ConfigMap::existU64](#) (const std::string &key)
To detect whether the key exists and related to a U64.
- bool [INTELLI::ConfigMap::existI64](#) (const std::string &key)
To detect whether the key exists and related to a I64.
- bool [INTELLI::ConfigMap::existDouble](#) (const std::string &key)
To detect whether the key exists and related to a double.
- bool [INTELLI::ConfigMap::existString](#) (const std::string &key)
To detect whether the key exists and related to a std::string.
- bool [INTELLI::ConfigMap::exist](#) (const std::string &key)
To detect whether the key exists.
- uint64_t [INTELLI::ConfigMap::getU64](#) (const std::string &key)
To get a U64 value by key.
- int64_t [INTELLI::ConfigMap::getI64](#) (const std::string &key)
To get a I64 value by key.
- double [INTELLI::ConfigMap::getDouble](#) (const std::string &key)
To get a double value by key.
- std::string [INTELLI::ConfigMap::getString](#) (const std::string &key)
To get a std::string value by key.
- std::string [INTELLI::ConfigMap::toString](#) (const std::string &separator="\t", std::string newLine="\n")
convert the whole map to std::string and retuen
- bool [INTELLI::ConfigMap::fromString](#) (const std::string src, const std::string &separator="\t", std::string newLine="\n")
load the map from some external string
- void [INTELLI::ConfigMap::cloneInto](#) ([ConfigMap](#) &dest)
clone this config into destination
- void [INTELLI::ConfigMap::loadFrom](#) ([ConfigMap](#) &src)
load some information an external one
- bool [INTELLI::ConfigMap::toFile](#) (const std::string &fname, const std::string &separator=",", std::string newLine="\n")
convert the whole map to file
- bool [INTELLI::ConfigMap::fromFile](#) (const std::string &fname, std::string separator=",", std::string newLine="\n")
update the whole map from file
- bool [INTELLI::ConfigMap::fromCArg](#) (const int argc, char **argv)

- update the whole map from c/c++ program's args*
 - `int64_t INTELLI::ConfigMap::tryI64 (const string &key, int64_t defaultValue=0, bool showWarning=false)`
Try to get an I64 from config map, if not exist, use default value instead.
 - `std::map< std::string, std::string > INTELLI::ConfigMap::getStrMap ()`
return the map of string
 - `std::map< std::string, int64_t > INTELLI::ConfigMap::getI64Map ()`
return the map of I64
 - `std::map< std::string, double > INTELLI::ConfigMap::getDoubleMap ()`
return the map of I64
 - `uint64_t INTELLI::ConfigMap::tryU64 (const string &key, uint64_t defaultValue=0, bool showWarning=false)`
Try to get an U64 from config map, if not exist, use default value instead.
 - `double INTELLI::ConfigMap::tryDouble (const string &key, double defaultValue=0, bool showWarning=false)`
Try to get a double from config map, if not exist, use default value instead.
 - `string INTELLI::ConfigMap::tryString (const string &key, const string &defaultValue="", bool showWarning=false)`
Try to get an String from config map, if not exist, use default value instead.

Variables

- `std::map< std::string, uint64_t > INTELLI::ConfigMap::u64Map`
- `std::map< std::string, int64_t > INTELLI::ConfigMap::i64Map`
- `std::map< std::string, double > INTELLI::ConfigMap::doubleMap`
- `std::map< std::string, std::string > INTELLI::ConfigMap::strMap`

7.6.1 Detailed Description

7.6.1.1 Utils

This folder contains the public utils shared by INTELISTREAM team and some third party dependencies.

This group provides common functions to support the Intelli Stream programs.

7.6.2 Function Documentation

7.6.2.1 cloneInto()

```
void INTELLI::ConfigMap::cloneInto (
    ConfigMap & dest ) [inline]
```

clone this config into destination

Parameters

<i>dest</i>	The clone destination
-------------	-----------------------

7.6.2.2 `edit()` [1/4]

```
void INTELLI::ConfigMap::edit (
    const std::string & key,
    double value ) [inline]
```

Edit the config map. If not exit the config, will create new, or will overwrite.

Parameters

<i>key</i>	The look up key in std::string
<i>value</i>	The double value

7.6.2.3 `edit()` [2/4]

```
void INTELLI::ConfigMap::edit (
    const std::string & key,
    int64_t value ) [inline]
```

Edit the config map. If not exit the config, will create new, or will overwrite.

Parameters

<i>key</i>	The look up key in std::string
<i>value</i>	The i64 value

7.6.2.4 `edit()` [3/4]

```
void INTELLI::ConfigMap::edit (
    const std::string & key,
    std::string value ) [inline]
```

Edit the config map. If not exit the config, will create new, or will overwrite.

Parameters

<i>key</i>	The look up key in std::string
<i>value</i>	The std::string value

7.6.2.5 edit() [4/4]

```
void INTELLI::ConfigMap::edit (
    const std::string & key,
    uint64_t value ) [inline]
```

Edit the config map. If not exist the config, will create new, or will overwrite.

Parameters

<i>key</i>	The look up key in std::string
<i>value</i>	The u64 value

7.6.2.6 exist()

```
bool INTELLI::ConfigMap::exist (
    const std::string & key ) [inline]
```

To detect whether the key exists.

Parameters

<i>key</i>	
------------	--

Returns

bool for the result

7.6.2.7 existDouble()

```
bool INTELLI::ConfigMap::existDouble (
    const std::string & key ) [inline]
```

To detect whether the key exists and related to a double.

Parameters

<i>key</i>	
------------	--

Returns

bool for the result

7.6.2.8 existI64()

```
bool INTELLI::ConfigMap::existI64 (
    const std::string & key ) [inline]
```

To detect whether the key exists and related to a I64.

Parameters

<i>key</i>	
------------	--

Returns

bool for the result

7.6.2.9 existString()

```
bool INTELLI::ConfigMap::existString (
    const std::string & key ) [inline]
```

To detect whether the key exists and related to a std::string.

Parameters

<i>key</i>	
------------	--

Returns

bool for the result

7.6.2.10 existU64()

```
bool INTELLI::ConfigMap::existU64 (
    const std::string & key ) [inline]
```

To detect whether the key exists and related to a U64.

Parameters

<i>key</i>	
------------	--

Returns

bool for the result

7.6.2.11 fromCArg()

```
bool INTELLI::ConfigMap::fromCArg (
    const int argc,
    char ** argv ) [inline]
```

update the whole map from c/c++ program's args

Parameters

<i>argc</i>	the count of input args
<i>argv</i>	the arg list in chars

Note

Will automatically detect int64, double, and string

Returns

bool, whether the file is loaded

7.6.2.12 fromFile()

```
bool INTELLI::ConfigMap::fromFile (
    const std::string & fname,
    std::string separator = ",",
    std::string newLine = "\n" ) [inline]
```

update the whole map from file

Parameters

<i>fname</i>	The file name
<i>separator</i>	The separator std::string, default ",", for csv style
<i>newLine</i>	The newline std::string, default "\n"

Returns

bool, whether the file is loaded

7.6.2.13 fromString()

```
bool INTELLI::ConfigMap::fromString (
    const std::string src,
    const std::string & separator = "\t",
    std::string newLine = "\n" ) [inline]
```

load the map from some external string

Parameters

<i>src,the</i>	string
<i>separator</i>	The separator std::string, default "\t"
<i>newLine</i>	The newline std::string, default "\n"

Returns

bool whether successful

7.6.2.14 getDouble()

```
double INTELLI::ConfigMap::getDouble (
    const std::string & key ) [inline]
```

To get a double value by key.

Parameters

<i>key</i>	
------------	--

Returns

value

Warning

the key must exist!!

7.6.2.15 getDoubleMap()

```
std::map<std::string, double> INTELLI::ConfigMap::getDoubleMap ( ) [inline]
```

return the map of l64

Returns

the doubleMap variable

7.6.2.16 getI64()

```
int64_t INTELLI::ConfigMap::getI64 (
    const std::string & key ) [inline]
```

To get a I64 value by key.

Parameters

<i>key</i>	
------------	--

Returns

value

Warning

the key must exist!!

7.6.2.17 getI64Map()

```
std::map<std::string, int64_t> INTELLI::ConfigMap::getI64Map ( ) [inline]
```

return the map of I64

Returns

the i64Map variable

7.6.2.18 getString()

```
std::string INTELLI::ConfigMap::getString (
    const std::string & key ) [inline]
```

To get a std::string value by key.

Parameters

<i>key</i>	
------------	--

Returns

value

Warning

the key must exist!!

7.6.2.19 getStrMap()

```
std::map<std::string, std::string> INTELLI::ConfigMap::getStrMap ( ) [inline]
```

return the map of string

Returns

the strMap variable

7.6.2.20 getU64()

```
uint64_t INTELLI::ConfigMap::getU64 (
    const std::string & key ) [inline]
```

To get a U64 value by key.

Parameters

<i>key</i>	
------------	--

Returns

value

Warning

the key must exist!!

7.6.2.21 loadFrom()

```
void INTELLI::ConfigMap::loadFrom (
    ConfigMap & src ) [inline]
```

load some information an external one

Parameters

<i>src</i>	The clone destination
------------	-----------------------

7.6.2.22 toFile()

```
bool INTELLI::ConfigMap::toFile (
    const std::string & fname,
    const std::string & separator = ",",
    std::string newLine = "\n" ) [inline]
```

convert the whole map to file

Parameters

<i>fname</i>	The file name
<i>separator</i>	The separator std::string, default "," for csv style
<i>newLine</i>	The newline std::string, default "\n"

Returns

bool, whether the file is created

7.6.2.23 toString()

```
std::string INTELLI::ConfigMap::toString (
    const std::string & separator = "\t",
    std::string newLine = "\n" ) [inline]
```

convert the whole map to std::string and return

Parameters

<i>separator</i>	The separator std::string, default "\t"
<i>newLine</i>	The newline std::string, default "\n"

Returns

the result

7.6.2.24 tryDouble()

```
double INTELLI::ConfigMap::tryDouble (
    const string & key,
    double defaultValue = 0,
    bool showWarning = false ) [inline]
```

Try to get a double from config map, if not exist, use default value instead.

Parameters

<i>key</i>	The key
<i>defaultValue</i>	The default
<i>showWarning</i>	Whether show warning logs if not found

Returns

The returned value

7.6.2.25 tryI64()

```
int64_t INTELLI::ConfigMap::tryI64 (
    const string & key,
    int64_t defaultValue = 0,
    bool showWarning = false ) [inline]
```

Try to get an I64 from config map, if not exist, use default value instead.

Parameters

<i>key</i>	The key
<i>defaultValue</i>	The default
<i>showWarning</i>	Whether show warning logs if not found

Returns

The returned value

7.6.2.26 tryString()

```
string INTELLI::ConfigMap::tryString (
    const string & key,
    const string & defaultValue = "",
    bool showWarning = false ) [inline]
```

Try to get an String from config map, if not exist, use default value instead.

Parameters

<i>key</i>	The key
<i>defaultValue</i>	The default
<i>showWarning</i>	Whether show warning logs if not found

Returns

The returned value

7.6.2.27 tryU64()

```
uint64_t INTELLI::ConfigMap::tryU64 (
    const string & key,
    uint64_t defaultValue = 0,
    bool showWarning = false ) [inline]
```

Try to get an U64 from config map, if not exist, use default value instead.

Parameters

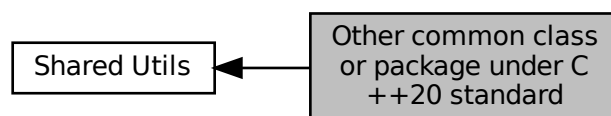
<i>key</i>	The key
<i>defaultValue</i>	The default
<i>showWarning</i>	Whether show warning logs if not found

Returns

The returned value

7.7 Other common class or package under C++20 standard

Collaboration diagram for Other common class or package under C++20 standard:



Classes

- class [INTELLI::AbstractC20Thread](#)
The base class and abstraction of C++20 thread, and it can be derived into other threads.
- class [INTELLI::C20Buffer< dataType >](#)
- class [INTELLI::MemoryTracker](#)
The top entity to trace current, average and maximum memory foot print.
- class [INTELLI::ThreadPerf](#)
The top entity to provide perf traces, please use this class only UNLESS you know what you are doing.
- class [INTELLI::ThreadPerfPAPI](#)
The top entity to provide perf traces by using PAPI lib.

Macros

- `#define newAbstractC20Thread std::make_shared<INTELLI::AbstractC20Thread>`
(Macro) To creat a new [newAbstractC20Thread](#) under shared pointer.
- `#define newThreadPerf std::make_shared<INTELLI::ThreadPerf>`
(Macro) To creat a new [ThreadPerf](#) under shared pointer.
- `#define newThreadPerfPAPI std::make_shared<INTELLI::ThreadPerfPAPI>`
(Macro) To creat a new [ThreadPerfPAPI](#) under shared pointer.

Typedefs

- `typedef std::shared_ptr< AbstractC20Thread > INTELLI::AbstractC20ThreadPtr`
The class to describe a shared pointer to [AbstractC20Thread](#).
- `typedef std::shared_ptr< INTELLI::ThreadPerf > INTELLI::ThreadPerfPtr`
The class to describe a shared pointer to [ThreadPerf](#).
- `typedef std::shared_ptr< INTELLI::ThreadPerfPAPI > INTELLI::ThreadPerfPAPIPtr`
The class to describe a shared pointer to [ThreadPerfPAPI](#).

7.7.1 Detailed Description

This package covers some common C++20 new features, such as `std::thread` to ease the programming

7.8 Configurations

Collaboration diagram for Configurations:



Classes

- class [INTELLI::ConfigMap](#)
The unified map structure to store configurations in a key-value style.

Macros

- `#define newConfigMap make_shared<INTELLI::ConfigMap>`
(Macro) To creat a new ConfigMap under shared pointer.

Typedefs

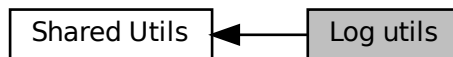
- `typedef std::shared_ptr< ConfigMap > INTELLI::ConfigMapPtr`
The class to describe a shared pointer to [ConfigMap](#).

7.8.1 Detailed Description

This package is used to store configuration information in an unified map and get away from too many stand-alone functtions

7.9 Log utils

Collaboration diagram for Log utils:



Classes

- class [INTELLI::IntelliLog](#)
The log functions packed in class.
- class [INTELLI::IntelliLog_FileProtector](#)
The protector for concurrent log on a file.

Macros

- `#define INTELLI_INFO(n) INTELLI::IntelliLog::log("INFO",n)`
(Macro) To log something as information
- `#define INTELLI_ERROR(n) INTELLI::IntelliLog::log("ERROR",n)`
(Macro) To log something as error
- `#define INTELLI_WARNING(n) INTELLI::IntelliLog::log("WARNING",n)`
- `#define INTELLI_DEBUG(n) INTELLI::IntelliLog::log("DEBUG",n)`
(Macro) To log something as debug

Functions

- static void `INTELLI::IntelliLog::log` (std::string level, std::string_view message, std::source_location const source=std::source_location::current())
Produce a log.
- static void `INTELLI::IntelliLog::setupLoggingFile` (string fname)
set up the logging file by its name
- void `INTELLI::IntelliLog_FileProtector::lock` ()
lock this protector
- void `INTELLI::IntelliLog_FileProtector::unlock` ()
unlock this protector
- void `INTELLI::IntelliLog_FileProtector::openLogFile` (const string &fname)
try to open a file
- void `INTELLI::IntelliLog_FileProtector::appendLogFile` (const string &msg)
try to appened something to the file, if it's opened

7.9.1 Detailed Description

This package is used for logging

7.9.2 Function Documentation

7.9.2.1 appendLogFile()

```
void INTELLI::IntelliLog_FileProtector::appendLogFile (
    const string & msg ) [inline]
```

try to appened something to the file, if it's opened

Parameters

<i>msg</i>	The message to appened
------------	------------------------

7.9.2.2 log()

```
void INTELLI::IntelliLog::log (
    std::string level,
    std::string_view message,
    std::source_location const source = std::source_location::current() ) [static]
```

Produce a log.

Parameters

<i>level</i>	The log level you want to indicate
<i>message</i>	The log message you want to indicate
<i>source</i>	reserved

Note

message is automatically appended with a "\n"

7.9.2.3 openLogFile()

```
void INTELLI::IntelliLog_FileProtector::openLogFile (
    const string & fname ) [inline]
```

try to open a file

Parameters

<i>fname</i>	The name of file
--------------	------------------

7.9.2.4 setupLoggingFile()

```
void INTELLI::IntelliLog::setupLoggingFile (
    string fname ) [static]
```

set up the logging file by its name

Parameters

<i>fname</i>	the name of file
--------------	------------------

7.10 tensor operations

Collaboration diagram for tensor operations:



Classes

- class [INTELLITensorOP](#)
The common tensor functions packed in class.

Macros

- `#define newTensor make_shared<torch::Tensor>`
(Macro) To creat a new Tensor under shared pointer.

Typedefs

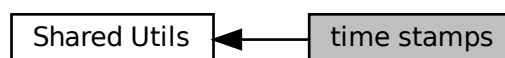
- `typedef std::shared_ptr< torch::Tensor > INTELLI::TensorPtr`
The class to describe a shared pointer to torch::Tensor.

7.10.1 Detailed Description

This package is used for some common tensor operations

7.11 time stamps

Collaboration diagram for time stamps:



Classes

- class [INTELLI::IntelliTimeStamp](#)
The class to define a timestamp.
- class [INTELLI::IntelliTimeStampGenerator](#)
The basic class to generate time stamps.

Macros

- #define [newIntelliTimeStamp](#) std::make_shared<[INTELLI::IntelliTimeStamp](#)>
(Macro) To creat a new IntelliTimeStamp under shared pointer.

Typedefs

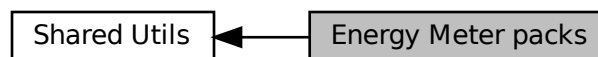
- typedef std::shared_ptr< [INTELLI::IntelliTimeStamp](#) > [INTELLI::IntelliTimeStampPtr](#)
The class to describe a shared pointer to [IntelliTimeStamp](#).

7.11.1 Detailed Description

This package is used for basic time stamp functions

7.12 Energy Meter packs

Collaboration diagram for Energy Meter packs:



Classes

- class [DIVERSE_METER::AbstractMeter](#)
The abstract class for all meters.
- class [DIVERSE_METER::EspMeterUart](#)
the entity of an esp32s2-based power meter, connected by uart 115200
- class [DIVERSE_METER::IntelMeter](#)
the entity of intel msr-based power meter, may be not support for some newer architectures
- class [DIVERSE_METER::MeterTable](#)
The table class to index all meters.

Macros

- `#define newMeterTable std::make_shared<DIVERSE_METER::MeterTable>`
(Macro) To creat a new MeterTable under shared pointer.

Typedefs

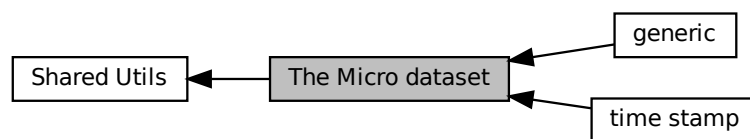
- `typedef std::shared_ptr< class DIVERSE_METER::MeterTable > DIVERSE_METER::MeterTable::MeterTablePtr`
The class to describe a shared pointer to [MeterTable](#).
- `typedef std::shared_ptr< DIVERSE_METER::AbstractMeter > DIVERSE_METER::AbstractMeterPtr`

7.12.1 Detailed Description

This package is used for energy meter

7.13 The Micro dataset

Collaboration diagram for The Micro dataset:



Modules

- [generic](#)
- [time stamp](#)

Classes

- class [INTELLI::MicroDataSet](#)
The all-in-one class for the Micro dataset.

Functions

- [INTELLI::MicroDataSet::MicroDataSet](#) ()=default
default construction, with auto random generator
- [INTELLI::MicroDataSet::MicroDataSet](#) (uint64_t _seed)
construction with seed
- void [INTELLI::MicroDataSet::setSeed](#) (uint64_t _seed)
construction with seed

7.13.1 Detailed Description

Note

The STL and static headers will be named as *.hpp, while *.h means there are real, fixed classes

Warning

Please use this file ONLY as STL, it may not work if you turn it into *.cpp!!!!

This is the synthetic dataset Micro, firstly introduced in our SIGMOD 2021 paper

```
@article{IntraWJoin21,
  author = {Zhang, Shuhao and Mao, Yancan and He, Jiong and Grulich, Philipp M and Zeuch, Steffen and He, Bing},
  title = {Parallelizing Intra-Window Join on Multicores: An Experimental Study},
  booktitle = {Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 18--22, 2021, Denver, Colorado, USA},
  series = {SIGMOD '21},
  year={2021},
  isbn = {978-1-4503-8343-1/21/06},
  url = {https://doi.org/10.1145/3448016.3452793},
  doi = {10.1145/3448016.3452793},
}
```

7.13.2 Function Documentation

7.13.2.1 MicroDataSet()

```
INTELLI::MicroDataSet::MicroDataSet (
    uint64_t _seed ) [inline], [explicit]
```

construction with seed

Parameters

<i>seed</i>	The seed for random generator
-------------	-------------------------------

7.13.2.2 setSeed()

```
void INTELLI::MicroDataSet::setSeed (
    uint64_t _seed ) [inline]
```

construction with seed

Parameters

<i>seed</i>	The seed for random generator
-------------	-------------------------------

7.14 generic

Collaboration diagram for generic:



Functions

- `template<class dType = uint32_t>`
`vector< dType > INTELLI::MicroDataSet::genIncrementalAlphabet (size_t len)`
To generate incremental alphabet, starting from 0 and end at len.
- `template<class tsType = size_t>`
`vector< tsType > INTELLI::MicroDataSet::genZipfInt (size_t len, tsType maxV, double fac)`
The function to generate a vector of integers which has zipf distribution.
- `template<class tsType = uint32_t, class genType = std::mt19937>`
`vector< tsType > INTELLI::MicroDataSet::genRandInt (size_t len, tsType maxV, tsType minV=0)`
generate the vector of random integer
- `template<class dType = double>`
`vector< dType > INTELLI::MicroDataSet::genZipfLut (size_t len, dType fac)`
To generate the zipf Lut.

7.14.1 Detailed Description

The functions for general generation of Micro

7.14.2 Function Documentation

7.14.2.1 genIncrementalAlphabet()

```

template<class dType = uint32_t>
vector<dType> INTELLI::MicroDataSet::genIncrementalAlphabet (
    size_t len ) [inline]
  
```

To generate incremental alphabet, starting from 0 and end at len.

Template Parameters

<i>dType</i>	The data type in the alphabet, default uint32_t
--------------	---

Parameters

<i>len</i>	The length of alphabet
------------	------------------------

Returns

The output vector alphabet

7.14.2.2 genRandInt()

```
template<class tsType = uint32_t, class genType = std::mt19937>
vector<tsType> INTELLI::MicroDataSet::genRandInt (
    size_t len,
    tsType maxV,
    tsType minV = 0 ) [inline]
```

generate the vector of random integer

Template Parameters

<i>tsType</i>	The data type, default uint32_t
<i>genType</i>	The generator type, default mt19937 (32 bit rand)

Parameters

<i>len</i>	The length of output vector
<i>maxV</i>	The maximum value of output
<i>minV</i>	The minimum value of output

Returns

The output vector

Note

Both signed and unsigned int are support, just make sure you have right tsType
Other options for genType:

- mt19937_64: 64 bit rand
- ranlux24: 24 bit
- ranlux48: 48 bit

7.14.2.3 genZipfInt()

```
template<class tsType = size_t>
vector<tsType> INTELLI::MicroDataSet::genZipfInt (
    size_t len,
    tsType maxV,
    double fac ) [inline]
```

The function to generate a vector of integers which has zipf distribution.

Parameters

<i>tsType</i>	The data type of int, default is size_t
<i>len</i>	The length of output vector
<i>maxV</i>	The maximum value of integer
<i>fac</i>	The zipf factor, in [0,1]

Returns

the output vector

7.14.2.4 genZipfLut()

```
template<class dType = double>
vector<dType> INTELLI::MicroDataSet::genZipfLut (
    size_t len,
    dType fac ) [inline]
```

To generate the zipf Lut.

Template Parameters

<i>dType</i>	The data type in the alphabet, default double
--------------	---

Parameters

<i>len</i>	The length of alphabet
<i>fac</i>	The zipf factor, in [0,1]

Returns

The output vector lut

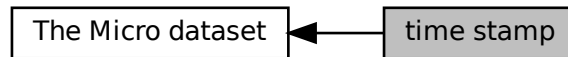
Compute scaling factor such that

$\text{sum}(\text{lut}[i], i=1..\text{alphabet_size}) = 1.0$

Generate the lookup table

7.15 time stamp

Collaboration diagram for time stamp:



Functions

- `template<class tsType = size_t>`
`vector< tsType > INTELLI::MicroDataSet::genSmoothTimeStamp (size_t len, size_t step, size_t interval)`
The function to generate a vector of timestamp which grows smoothly.
- `template<class tsType = size_t>`
`vector< tsType > INTELLI::MicroDataSet::genSmoothTimeStamp (size_t len, size_t maxTime)`
- `template<class tsType = size_t>`
`vector< tsType > INTELLI::MicroDataSet::genZipfTimeStamp (size_t len, tsType maxTime, double fac)`
The function to generate a vector of timestamp which has zipf distribution.

7.15.1 Detailed Description

This group is specialized for time stamps, as they should follow an incremental order

7.15.2 Function Documentation

7.15.2.1 genSmoothTimeStamp()

```

template<class tsType = size_t>
vector<tsType> INTELLI::MicroDataSet::genSmoothTimeStamp (
    size_t len,
    size_t step,
    size_t interval ) [inline]
  
```

The function to generate a vector of timestamp which grows smoothly.

Template Parameters

<i>tsType</i>	The data type of time stamp, default is <code>size_t</code>
---------------	---

Parameters

<i>len</i>	The length of output vector
<i>step</i>	Within the step, timestamp will remain the same
<i>interval</i>	The incremental value between two steps

Returns

The vector of time stamp

7.15.2.2 genZipfTimeStamp()

```
template<class tsType = size_t>
vector<tsType> INTELLI::MicroDataSet::genZipfTimeStamp (
    size_t len,
    tsType maxTime,
    double fac ) [inline]
```

The function to generate a vector of timestamp which has zipf distribution.

Parameters

<i>tsType</i>	The data type of time stamp, default is <code>size_t</code>
<i>len</i>	The length of output vector
<i>maxTime</i>	The maximum value of time stamp
<i>fac</i>	The zipf factor, in [0,1]

Returns

the output vector

See also

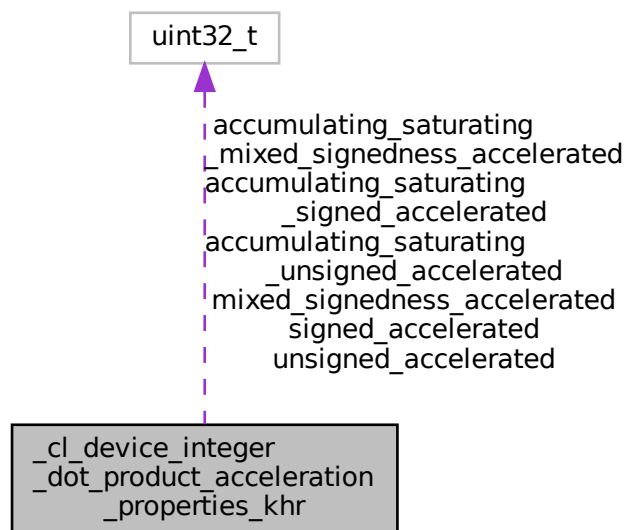
[genZipfInt](#)

Chapter 8

Class Documentation

8.1 `_cl_device_integer_dot_product_acceleration_properties_khr` Struct Reference

Collaboration diagram for `_cl_device_integer_dot_product_acceleration_properties_khr`:



Public Attributes

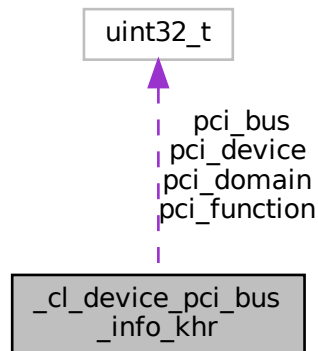
- `cl_bool` **signed_accelerated**
- `cl_bool` **unsigned_accelerated**
- `cl_bool` **mixed_signedness_accelerated**
- `cl_bool` **accumulating_saturating_signed_accelerated**
- `cl_bool` **accumulating_saturating_unsigned_accelerated**
- `cl_bool` **accumulating_saturating_mixed_signedness_accelerated**

The documentation for this struct was generated from the following file:

- `include/CL/cl_ext.h`

8.2 _cl_device_pci_bus_info_khr Struct Reference

Collaboration diagram for _cl_device_pci_bus_info_khr:



Public Attributes

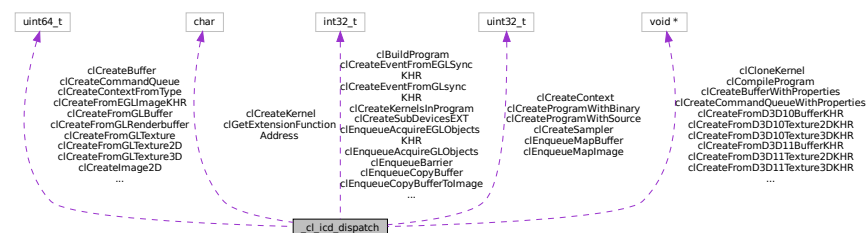
- `cl_uint pci_domain`
- `cl_uint pci_bus`
- `cl_uint pci_device`
- `cl_uint pci_function`

The documentation for this struct was generated from the following file:

- `include/CL/cl_ext.h`

8.3 _cl_icd_dispatch Struct Reference

Collaboration diagram for _cl_icd_dispatch:



Public Attributes

- cl_api_clGetPlatformIDs **clGetPlatformIDs**
- cl_api_clGetPlatformInfo **clGetPlatformInfo**
- cl_api_clGetDeviceIDs **clGetDeviceIDs**
- cl_api_clGetDeviceInfo **clGetDeviceInfo**
- cl_api_clCreateContext **clCreateContext**
- cl_api_clCreateContextFromType **clCreateContextFromType**
- cl_api_clRetainContext **clRetainContext**
- cl_api_clReleaseContext **clReleaseContext**
- cl_api_clGetContextInfo **clGetContextInfo**
- cl_api_clCreateCommandQueue **clCreateCommandQueue**
- cl_api_clRetainCommandQueue **clRetainCommandQueue**
- cl_api_clReleaseCommandQueue **clReleaseCommandQueue**
- cl_api_clGetCommandQueueInfo **clGetCommandQueueInfo**
- cl_api_clSetCommandQueueProperty **clSetCommandQueueProperty**
- cl_api_clCreateBuffer **clCreateBuffer**
- cl_api_clCreateImage2D **clCreateImage2D**
- cl_api_clCreateImage3D **clCreateImage3D**
- cl_api_clRetainMemObject **clRetainMemObject**
- cl_api_clReleaseMemObject **clReleaseMemObject**
- cl_api_clGetSupportedImageFormats **clGetSupportedImageFormats**
- cl_api_clGetMemObjectInfo **clGetMemObjectInfo**
- cl_api_clGetImageInfo **clGetImageInfo**
- cl_api_clCreateSampler **clCreateSampler**
- cl_api_clRetainSampler **clRetainSampler**
- cl_api_clReleaseSampler **clReleaseSampler**
- cl_api_clGetSamplerInfo **clGetSamplerInfo**
- cl_api_clCreateProgramWithSource **clCreateProgramWithSource**
- cl_api_clCreateProgramWithBinary **clCreateProgramWithBinary**
- cl_api_clRetainProgram **clRetainProgram**
- cl_api_clReleaseProgram **clReleaseProgram**
- cl_api_clBuildProgram **clBuildProgram**
- cl_api_clUnloadCompiler **clUnloadCompiler**
- cl_api_clGetProgramInfo **clGetProgramInfo**
- cl_api_clGetProgramBuildInfo **clGetProgramBuildInfo**
- cl_api_clCreateKernel **clCreateKernel**
- cl_api_clCreateKernelsInProgram **clCreateKernelsInProgram**
- cl_api_clRetainKernel **clRetainKernel**
- cl_api_clReleaseKernel **clReleaseKernel**
- cl_api_clSetKernelArg **clSetKernelArg**
- cl_api_clGetKernelInfo **clGetKernelInfo**
- cl_api_clGetKernelWorkGroupInfo **clGetKernelWorkGroupInfo**
- cl_api_clWaitForEvents **clWaitForEvents**
- cl_api_clGetEventInfo **clGetEventInfo**
- cl_api_clRetainEvent **clRetainEvent**
- cl_api_clReleaseEvent **clReleaseEvent**
- cl_api_clGetEventProfilingInfo **clGetEventProfilingInfo**
- cl_api_clFlush **clFlush**
- cl_api_clFinish **clFinish**
- cl_api_clEnqueueReadBuffer **clEnqueueReadBuffer**
- cl_api_clEnqueueWriteBuffer **clEnqueueWriteBuffer**
- cl_api_clEnqueueCopyBuffer **clEnqueueCopyBuffer**
- cl_api_clEnqueueReadImage **clEnqueueReadImage**
- cl_api_clEnqueueWriteImage **clEnqueueWriteImage**

- `cl_api_clEnqueueCopyImage` **clEnqueueCopyImage**
- `cl_api_clEnqueueCopyImageToBuffer` **clEnqueueCopyImageToBuffer**
- `cl_api_clEnqueueCopyBufferToImage` **clEnqueueCopyBufferToImage**
- `cl_api_clEnqueueMapBuffer` **clEnqueueMapBuffer**
- `cl_api_clEnqueueMapImage` **clEnqueueMapImage**
- `cl_api_clEnqueueUnmapMemObject` **clEnqueueUnmapMemObject**
- `cl_api_clEnqueueNDRangeKernel` **clEnqueueNDRangeKernel**
- `cl_api_clEnqueueTask` **clEnqueueTask**
- `cl_api_clEnqueueNativeKernel` **clEnqueueNativeKernel**
- `cl_api_clEnqueueMarker` **clEnqueueMarker**
- `cl_api_clEnqueueWaitForEvents` **clEnqueueWaitForEvents**
- `cl_api_clEnqueueBarrier` **clEnqueueBarrier**
- `cl_api_clGetExtensionFunctionAddress` **clGetExtensionFunctionAddress**
- `cl_api_clCreateFromGLBuffer` **clCreateFromGLBuffer**
- `cl_api_clCreateFromGLTexture2D` **clCreateFromGLTexture2D**
- `cl_api_clCreateFromGLTexture3D` **clCreateFromGLTexture3D**
- `cl_api_clCreateFromGLRenderbuffer` **clCreateFromGLRenderbuffer**
- `cl_api_clGetGLObjectInfo` **clGetGLObjectInfo**
- `cl_api_clGetGLTextureInfo` **clGetGLTextureInfo**
- `cl_api_clEnqueueAcquireGLObjects` **clEnqueueAcquireGLObjects**
- `cl_api_clEnqueueReleaseGLObjects` **clEnqueueReleaseGLObjects**
- `cl_api_clGetGLContextInfoKHR` **clGetGLContextInfoKHR**
- `cl_api_clGetDeviceIDsFromD3D10KHR` **clGetDeviceIDsFromD3D10KHR**
- `cl_api_clCreateFromD3D10BufferKHR` **clCreateFromD3D10BufferKHR**
- `cl_api_clCreateFromD3D10Texture2DKHR` **clCreateFromD3D10Texture2DKHR**
- `cl_api_clCreateFromD3D10Texture3DKHR` **clCreateFromD3D10Texture3DKHR**
- `cl_api_clEnqueueAcquireD3D10ObjectsKHR` **clEnqueueAcquireD3D10ObjectsKHR**
- `cl_api_clEnqueueReleaseD3D10ObjectsKHR` **clEnqueueReleaseD3D10ObjectsKHR**
- `cl_api_clSetEventCallback` **clSetEventCallback**
- `cl_api_clCreateSubBuffer` **clCreateSubBuffer**
- `cl_api_clSetMemObjectDestructorCallback` **clSetMemObjectDestructorCallback**
- `cl_api_clCreateUserEvent` **clCreateUserEvent**
- `cl_api_clSetUserEventStatus` **clSetUserEventStatus**
- `cl_api_clEnqueueReadBufferRect` **clEnqueueReadBufferRect**
- `cl_api_clEnqueueWriteBufferRect` **clEnqueueWriteBufferRect**
- `cl_api_clEnqueueCopyBufferRect` **clEnqueueCopyBufferRect**
- `cl_api_clCreateSubDevicesEXT` **clCreateSubDevicesEXT**
- `cl_api_clRetainDeviceEXT` **clRetainDeviceEXT**
- `cl_api_clReleaseDeviceEXT` **clReleaseDeviceEXT**
- `cl_api_clCreateEventFromGLsyncKHR` **clCreateEventFromGLsyncKHR**
- `cl_api_clCreateSubDevices` **clCreateSubDevices**
- `cl_api_clRetainDevice` **clRetainDevice**
- `cl_api_clReleaseDevice` **clReleaseDevice**
- `cl_api_clCreateImage` **clCreateImage**
- `cl_api_clCreateProgramWithBuiltInKernels` **clCreateProgramWithBuiltInKernels**
- `cl_api_clCompileProgram` **clCompileProgram**
- `cl_api_clLinkProgram` **clLinkProgram**
- `cl_api_clUnloadPlatformCompiler` **clUnloadPlatformCompiler**
- `cl_api_clGetKernelArgInfo` **clGetKernelArgInfo**
- `cl_api_clEnqueueFillBuffer` **clEnqueueFillBuffer**
- `cl_api_clEnqueueFillImage` **clEnqueueFillImage**
- `cl_api_clEnqueueMigrateMemObjects` **clEnqueueMigrateMemObjects**
- `cl_api_clEnqueueMarkerWithWaitList` **clEnqueueMarkerWithWaitList**
- `cl_api_clEnqueueBarrierWithWaitList` **clEnqueueBarrierWithWaitList**
- `cl_api_clGetExtensionFunctionAddressForPlatform` **clGetExtensionFunctionAddressForPlatform**

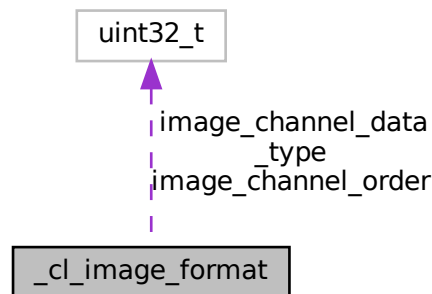
- `cl_api_clCreateFromGLTexture` **clCreateFromGLTexture**
- `cl_api_clGetDeviceIDsFromD3D11KHR` **clGetDeviceIDsFromD3D11KHR**
- `cl_api_clCreateFromD3D11BufferKHR` **clCreateFromD3D11BufferKHR**
- `cl_api_clCreateFromD3D11Texture2DKHR` **clCreateFromD3D11Texture2DKHR**
- `cl_api_clCreateFromD3D11Texture3DKHR` **clCreateFromD3D11Texture3DKHR**
- `cl_api_clCreateFromDX9MediaSurfaceKHR` **clCreateFromDX9MediaSurfaceKHR**
- `cl_api_clEnqueueAcquireD3D11ObjectsKHR` **clEnqueueAcquireD3D11ObjectsKHR**
- `cl_api_clEnqueueReleaseD3D11ObjectsKHR` **clEnqueueReleaseD3D11ObjectsKHR**
- `cl_api_clGetDeviceIDsFromDX9MediaAdapterKHR` **clGetDeviceIDsFromDX9MediaAdapterKHR**
- `cl_api_clEnqueueAcquireDX9MediaSurfacesKHR` **clEnqueueAcquireDX9MediaSurfacesKHR**
- `cl_api_clEnqueueReleaseDX9MediaSurfacesKHR` **clEnqueueReleaseDX9MediaSurfacesKHR**
- `cl_api_clCreateFromEGLImageKHR` **clCreateFromEGLImageKHR**
- `cl_api_clEnqueueAcquireEGLObjectsKHR` **clEnqueueAcquireEGLObjectsKHR**
- `cl_api_clEnqueueReleaseEGLObjectsKHR` **clEnqueueReleaseEGLObjectsKHR**
- `cl_api_clCreateEventFromEGLSyncKHR` **clCreateEventFromEGLSyncKHR**
- `cl_api_clCreateCommandQueueWithProperties` **clCreateCommandQueueWithProperties**
- `cl_api_clCreatePipe` **clCreatePipe**
- `cl_api_clGetPipeInfo` **clGetPipeInfo**
- `cl_api_clSVMAlloc` **clSVMAlloc**
- `cl_api_clSVMFree` **clSVMFree**
- `cl_api_clEnqueueSVMFree` **clEnqueueSVMFree**
- `cl_api_clEnqueueSVMMemcpy` **clEnqueueSVMMemcpy**
- `cl_api_clEnqueueSVMMemFill` **clEnqueueSVMMemFill**
- `cl_api_clEnqueueSVMMap` **clEnqueueSVMMap**
- `cl_api_clEnqueueSVMUnmap` **clEnqueueSVMUnmap**
- `cl_api_clCreateSamplerWithProperties` **clCreateSamplerWithProperties**
- `cl_api_clSetKernelArgSVMPointer` **clSetKernelArgSVMPointer**
- `cl_api_clSetKernelExecInfo` **clSetKernelExecInfo**
- `cl_api_clGetKernelSubGroupInfoKHR` **clGetKernelSubGroupInfoKHR**
- `cl_api_clCloneKernel` **clCloneKernel**
- `cl_api_clCreateProgramWithIL` **clCreateProgramWithIL**
- `cl_api_clEnqueueSVMigrateMem` **clEnqueueSVMigrateMem**
- `cl_api_clGetDeviceAndHostTimer` **clGetDeviceAndHostTimer**
- `cl_api_clGetHostTimer` **clGetHostTimer**
- `cl_api_clGetKernelSubGroupInfo` **clGetKernelSubGroupInfo**
- `cl_api_clSetDefaultDeviceCommandQueue` **clSetDefaultDeviceCommandQueue**
- `cl_api_clSetProgramReleaseCallback` **clSetProgramReleaseCallback**
- `cl_api_clSetProgramSpecializationConstant` **clSetProgramSpecializationConstant**
- `cl_api_clCreateBufferWithProperties` **clCreateBufferWithProperties**
- `cl_api_clCreateImageWithProperties` **clCreateImageWithProperties**
- `cl_api_clSetContextDestructorCallback` **clSetContextDestructorCallback**

The documentation for this struct was generated from the following file:

- `include/CL/cl_icd.h`

8.4 `_cl_image_format` Struct Reference

Collaboration diagram for `_cl_image_format`:



Public Attributes

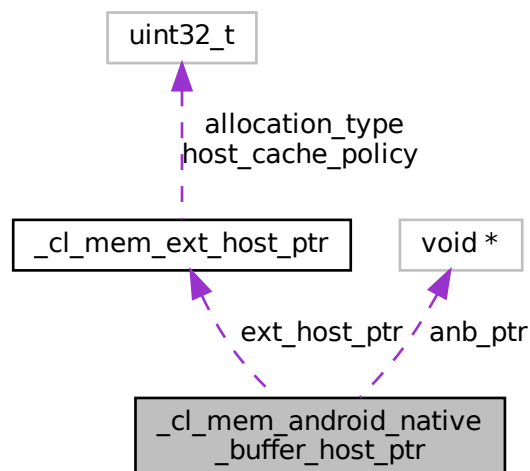
- `cl_channel_order` **`image_channel_order`**
- `cl_channel_type` **`image_channel_data_type`**

The documentation for this struct was generated from the following file:

- `include/CL/cl.h`

8.5 `_cl_mem_android_native_buffer_host_ptr` Struct Reference

Collaboration diagram for `_cl_mem_android_native_buffer_host_ptr`:



Public Attributes

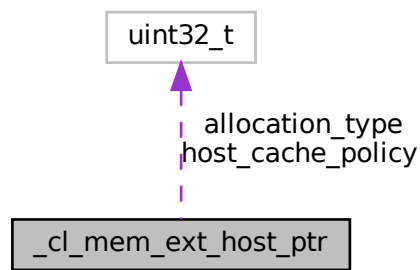
- [cl_mem_ext_host_ptr](#) **ext_host_ptr**
- void * **anb_ptr**

The documentation for this struct was generated from the following file:

- include/CL/cl_ext.h

8.6 _cl_mem_ext_host_ptr Struct Reference

Collaboration diagram for _cl_mem_ext_host_ptr:



Public Attributes

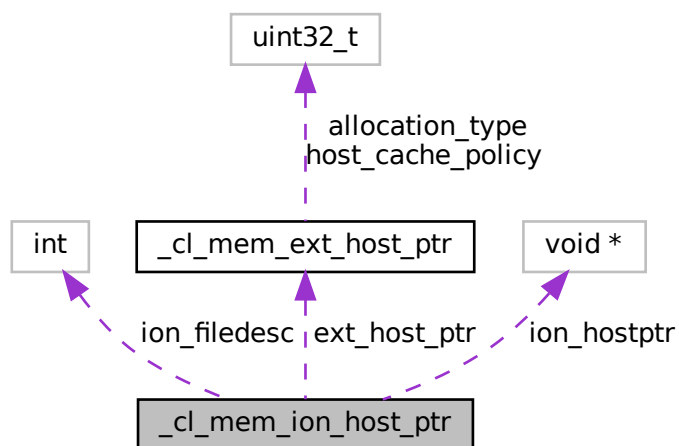
- cl_uint **allocation_type**
- cl_uint **host_cache_policy**

The documentation for this struct was generated from the following file:

- include/CL/cl_ext.h

8.7 _cl_mem_ion_host_ptr Struct Reference

Collaboration diagram for _cl_mem_ion_host_ptr:



Public Attributes

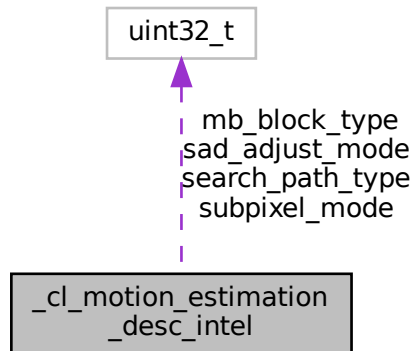
- `cl_mem_ext_host_ptr` `ext_host_ptr`
- `int` `ion_filedesc`
- `void *` `ion_hostptr`

The documentation for this struct was generated from the following file:

- `include/CL/cl_ext.h`

8.8 _cl_motion_estimation_desc_intel Struct Reference

Collaboration diagram for _cl_motion_estimation_desc_intel:



Public Attributes

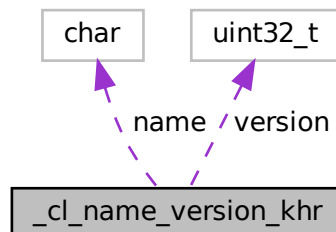
- `cl_uint` **mb_block_type**
- `cl_uint` **subpixel_mode**
- `cl_uint` **sad_adjust_mode**
- `cl_uint` **search_path_type**

The documentation for this struct was generated from the following file:

- `include/CL/cl_ext.h`

8.9 _cl_name_version_khr Struct Reference

Collaboration diagram for _cl_name_version_khr:



Public Attributes

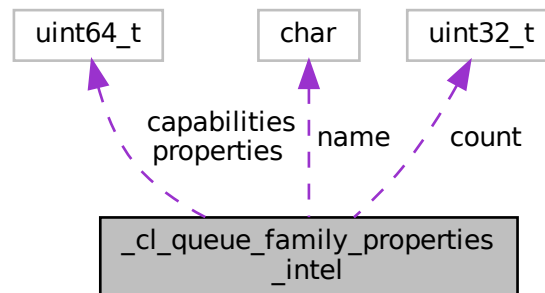
- `cl_version_khr` **version**
- `char` **name** [CL_NAME_VERSION_MAX_NAME_SIZE_KHR]

The documentation for this struct was generated from the following file:

- `include/CL/cl_ext.h`

8.10 `_cl_queue_family_properties_intel` Struct Reference

Collaboration diagram for `_cl_queue_family_properties_intel`:



Public Attributes

- `cl_command_queue_properties` **properties**
- `cl_command_queue_capabilities_intel` **capabilities**
- `cl_uint` **count**
- `char` **name** [CL_QUEUE_FAMILY_MAX_NAME_SIZE_INTEL]

The documentation for this struct was generated from the following file:

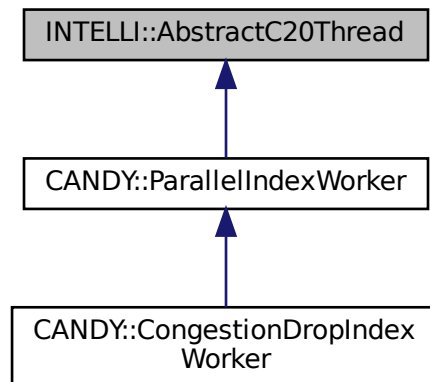
- `include/CL/cl_ext.h`

8.11 INTELLI::AbstractC20Thread Class Reference

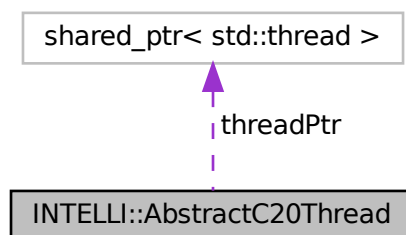
The base class and abstraction of C++20 thread, and it can be derived into other threads.

```
#include <Utils/AbstractC20Thread.hpp>
```

Inheritance diagram for INTELLI::AbstractC20Thread:



Collaboration diagram for INTELLI::AbstractC20Thread:



Public Member Functions

- void `startThread` ()
 to start this thread
- void `joinThread` ()
 the thread join function

Protected Member Functions

- virtual void [inlineMain](#) ()
The inline 'main' function of thread, as an interface.

Protected Attributes

- std::shared_ptr< std::thread > **threadPtr**

8.11.1 Detailed Description

The base class and abstraction of C++20 thread, and it can be derived into other threads.

8.11.2 Member Function Documentation

8.11.2.1 inlineMain()

```
virtual void INTELLI::AbstractC20Thread::inlineMain ( ) [inline], [protected], [virtual]
```

The inline 'main' function of thread, as an interface.

Note

Normally re-write this in derived classes

Reimplemented in [CANDY::ParallelIndexWorker](#).

The documentation for this class was generated from the following file:

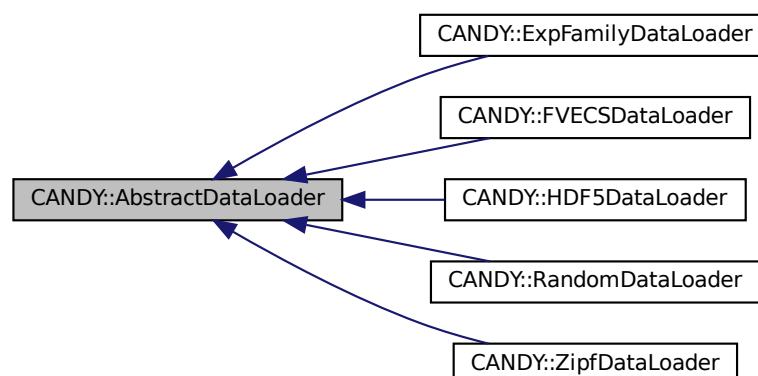
- include/Utils/[AbstractC20Thread.hpp](#)

8.12 CANDY::AbstractDataLoader Class Reference

The abstract class of data loader, parent for all loaders.

```
#include <DataLoader/AbstractDataLoader.h>
```

Inheritance diagram for CANDY::AbstractDataLoader:



Public Member Functions

- virtual bool [hijackConfig](#) (INTELLI::ConfigMapPtr cfg)
To hijack some configurations inline.
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
Set the GLOBAL config map related to this loader.
- virtual torch::Tensor [getData](#) ()
get the data tensor
- virtual torch::Tensor [getQuery](#) ()
get the query tensor

8.12.1 Detailed Description

The abstract class of data loader, parent for all loaders.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

8.12.2 Member Function Documentation

8.12.2.1 [getData\(\)](#)

```
torch::Tensor CANDY::AbstractDataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented in [CANDY::ZipfDataLoader](#), [CANDY::RandomDataLoader](#), [CANDY::HDF5DataLoader](#), [CANDY::FVECSDataLoader](#), and [CANDY::ExpFamilyDataLoader](#).

8.12.2.2 `getQuery()`

```
torch::Tensor CANDY::AbstractDataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented in [CANDY::ZipfDataLoader](#), [CANDY::RandomDataLoader](#), [CANDY::HDF5DataLoader](#), [CANDY::FVECSDataLoader](#), and [CANDY::ExpFamilyDataLoader](#).

8.12.2.3 `hijackConfig()`

```
bool CANDY::AbstractDataLoader::hijackConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

To hijack some configurations inline.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented in [CANDY::ExpFamilyDataLoader](#).

8.12.2.4 `setConfig()`

```
bool CANDY::AbstractDataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented in [CANDY::ZipfDataLoader](#), [CANDY::RandomDataLoader](#), [CANDY::HDF5DataLoader](#), [CANDY::FVECSDataLoader](#), and [CANDY::ExpFamilyDataLoader](#).

The documentation for this class was generated from the following files:

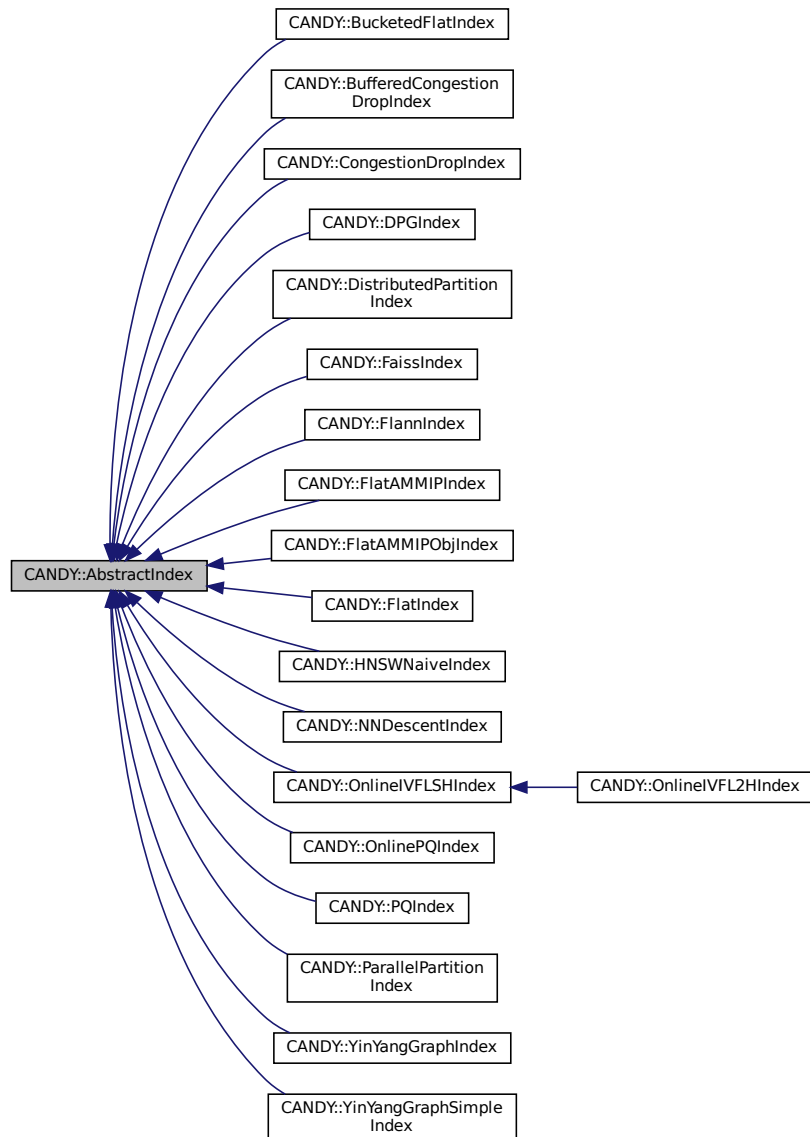
- include/DataLoader/[AbstractDataLoader.h](#)
- src/DataLoader/AbstractDataLoader.cpp

8.13 CANDY::AbstractIndex Class Reference

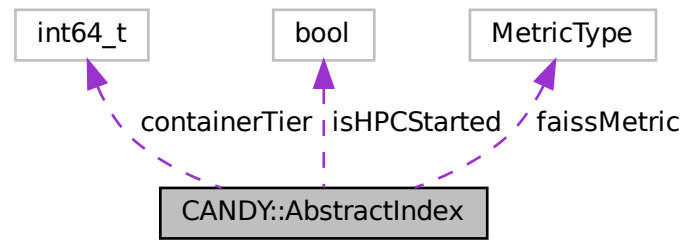
The abstract class of an index approach.

```
#include <CANDY/AbstractIndex.h>
```

Inheritance diagram for CANDY::AbstractIndex:



Collaboration diagram for CANDY::AbstractIndex:



Public Member Functions

- virtual void [setTier](#) (int64_t tie)
set the tier of this indexing, 0 refers the entry indexing
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfigClass](#) (INTELLI::ConfigMap cfg)
set the index-specific config related to one index
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor, also online function
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > [searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC features
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (torch::Tensor &t)
offline build phase

- virtual bool [waitPendingOperations](#) ()
a busy waiting for all pending operations to be done
- virtual bool [loadInitialStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)
- virtual bool [insertStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
insert a string object
- virtual bool [deleteStringObject](#) (torch::Tensor &t, int64_t k=1)
delete tensor along with its corresponding string object
- virtual std::vector< std::vector< std::string > > [searchStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects
- virtual std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > [searchTensorAndStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects and original tensors
- virtual bool [loadInitialTensorAndQueryDistribution](#) (torch::Tensor &t, torch::Tensor &query)
load the initial tensors and query distributions of a data base, use this BEFORE [insertTensor](#)

Public Attributes

- bool **isHPCStarted** = false

Protected Attributes

- faiss::MetricType **faissMetric** = faiss::METRIC_L2
- int64_t **containerTier** = 0

8.13.1 Detailed Description

The abstract class of an index approach.

8.13.2 Member Function Documentation

8.13.2.1 deleteStringObject()

```
bool CANDY::AbstractIndex::deleteStringObject (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete tensor along with its corresponding string object

Note

This is majorly an online function

Parameters

t	the tensor, some index need to be single row
k	the number of nearest neighbors

Returns

bool whether the delet is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::FlatAMMIPObjIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.2 deleteTensor()

```
bool CANDY::AbstractIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor, also online function

Parameters

t	the tensor, some index needs to be single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFLSHIndex](#), [CANDY::NNDescentIndex](#), [CANDY::HNSWNaiveIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.3 endHPC()

```
bool CANDY::AbstractIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC features

Returns

bool whether the HPC termination is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::NNDescentIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), and [CANDY::BufferedCongestionDropIndex](#).

8.13.2.4 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::AbstractIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::NNDescentIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::FlannIndex](#), [CANDY::FaissIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.5 insertStringObject()

```
bool CANDY::AbstractIndex::insertStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

insert a string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>strs</i>	the corresponding list of strings

Returns

bool whether the insertion is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::FlatAMMIPObjIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.6 insertTensor()

```
bool CANDY::AbstractIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented in [CANDY::YinYangGraphSimpleIndex](#), [CANDY::YinYangGraphIndex](#), [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFLSHIndex](#), [CANDY::NNDescentIndex](#), [CANDY::HNSWNaiveIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::FlannIndex](#), [CANDY::FaissIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.7 loadInitialStringObject()

```
bool CANDY::AbstractIndex::loadInitialStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row •
<i>strs</i>	the corresponding list of strings

Returns

bool whether the loading is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.8 loadInitialTensor()

```
bool CANDY::AbstractIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFL2HIndex](#), [CANDY::NNDescentIndex](#), [CANDY::FlannIndex](#), [CANDY::FaissIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.9 loadInitialTensorAndQueryDistribution()

```
bool CANDY::AbstractIndex::loadInitialTensorAndQueryDistribution (
    torch::Tensor & t,
    torch::Tensor & query ) [virtual]
```

load the initial tensors and query distributions of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the data tensor
<i>query</i>	the example query tensor

Returns

bool whether the loading is successful

Reimplemented in [CANDY::OnlineIVFL2HIndex](#).

8.13.2.10 offlineBuild()

```
bool CANDY::AbstractIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Note

This is to generate some offline data structures, NOT load offline tensors

Please use [loadInitialTensor](#) for loading initial tensors

Returns

whether the building is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::NNDescentIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), and [CANDY::BufferedCongestionDropIndex](#).

8.13.2.11 rawData()

```
torch::Tensor CANDY::AbstractIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::NNDescentIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.12 reviseTensor()

```
bool CANDY::AbstractIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFLSHIndex](#), [CANDY::NNDescentIndex](#), [CANDY::HNSWNaiveIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.13 searchIndex()

```
std::vector< faiss::idx_t > CANDY::AbstractIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented in [CANDY::PQIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::FlannIndex](#), and [CANDY::FaissIndex](#).

8.13.2.14 searchStringObject()

```
std::vector< std::vector< std::string > > CANDY::AbstractIndex::searchStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<std::vector<std::string>>` the result object for each row of query

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::FlatAMMIPObjIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.15 searchTensor()

```
std::vector< torch::Tensor > CANDY::AbstractIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

Reimplemented in [CANDY::YinYangGraphSimpleIndex](#), [CANDY::YinYangGraphIndex](#), [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFLSHIndex](#), [CANDY::NNDescentIndex](#), [CANDY::HNSWNaiveIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIPObjIndex](#), [CANDY::FlatAMMIPIIndex](#), [CANDY::FlannIndex](#), [CANDY::FaissIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.16 searchTensorAndStringObject()

```
std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > CANDY↵
::AbstractIndex::searchTensorAndStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects and original tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::tuple<std::vector<torch::Tensor>,std::vector<std::vector<std::string>>>`

Reimplemented in [CANDY::ParallelPartitionIndex](#), and [CANDY::CongestionDropIndex](#).

8.13.2.17 setConfig()

```
bool CANDY::AbstractIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Note

If there is any pre-built data structures, please load it in implementing this

If there is any initial tensors to be stored, please load it after this by [loadInitialTensor](#)

Returns

bool whether the configuration is successful

Reimplemented in [CANDY::YinYangGraphSimpleIndex](#), [CANDY::YinYangGraphIndex](#), [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::OnlineIVFLSHIndex](#), [CANDY::OnlineIVFL2HIndex](#), [CANDY::NNDescentIndex](#), [CANDY::HNSWNaiveIndex](#), [CANDY::FlatIndex](#), [CANDY::FlatAMMIObjIndex](#), [CANDY::FlatAMMIPIndex](#), [CANDY::FlannIndex](#), [CANDY::FaissIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), [CANDY::BufferedCongestionDropIndex](#), and [CANDY::BucketedFlatIndex](#).

8.13.2.18 setConfigClass()

```
bool CANDY::AbstractIndex::setConfigClass (
    INTELLI::ConfigMap cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class, using raw class
------------	---

Note

If there is any pre-built data structures, please load it in implementing this

If there is any initial tensors to be stored, please load it after this by [loadInitialTensor](#)

Returns

bool whether the configuration is successful

8.13.2.19 setFrozenLevel()

```
bool CANDY::AbstractIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented in [CANDY::PQIndex](#), [CANDY::ParallelPartitionIndex](#), [CANDY::OnlinePQIndex](#), [CANDY::NNDescentIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), and [CANDY::BufferedCongestionDropIndex](#).

8.13.2.20 setTier()

```
virtual void CANDY::AbstractIndex::setTier (
    int64_t tie ) [inline], [virtual]
```

set the tier of this indexing, 0 refers the entry indexing

Parameters

<i>tie</i>	the setting of tier number
------------	----------------------------

Note

The parameter of tier idx affects nothing now, but will do something later

8.13.2.21 startHPC()

```
bool CANDY::AbstractIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::NNDescentIndex](#), [CANDY::DPGIndex](#), [CANDY::DistributedPartitionIndex](#), [CANDY::CongestionDropIndex](#), and [CANDY::BufferedCongestionDropIndex](#).

8.13.2.22 waitPendingOperations()

```
bool CANDY::AbstractIndex::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Returns

bool, whether the waiting is actually done;

Reimplemented in [CANDY::ParallelPartitionIndex](#), [CANDY::DistributedPartitionIndex](#), and [CANDY::CongestionDropIndex](#).

The documentation for this class was generated from the following files:

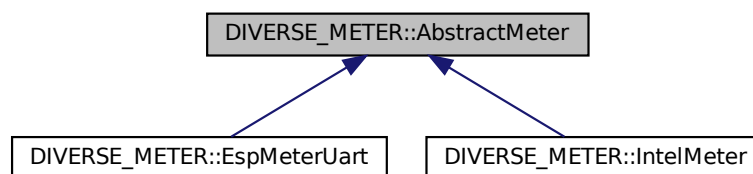
- include/CANDY/[AbstractIndex.h](#)
- src/CANDY/[AbstractIndex.cpp](#)

8.14 DIVERSE_METER::AbstractMeter Class Reference

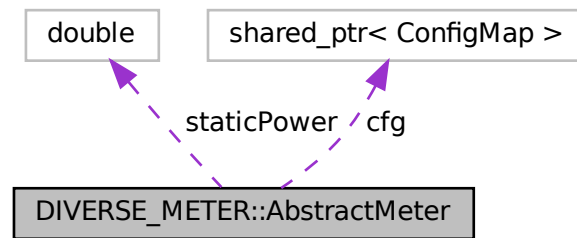
The abstract class for all meters.

```
#include <Utils/Meters/AbstractMeter.hpp>
```

Inheritance diagram for DIVERSE_METER::AbstractMeter:



Collaboration diagram for DIVERSE_METER::AbstractMeter:



Public Member Functions

- virtual void [setConfig](#) (INTELLI::ConfigMapPtr _cfg)
to set the configmap
- void [setStaticPower](#) (double _sp)
to manually set the static power
- void [testStaticPower](#) (uint64_t sleepingSecond)
to test the static power of a system by sleeping
- virtual void [startMeter](#) ()
to start the meter into some measuring tasks
- virtual void [stopMeter](#) ()
to stop the meter into some measuring tasks
- virtual double [getE](#) ()
to get the energy in J, including static energy consumption of system
- virtual double [getPeak](#) ()
to get the peak power in W, including static power of system
- virtual bool [isValid](#) ()
- double [getStaticPower](#) ()
to return the tested static power return the [staticPower](#)
- double [getStaicEnergyConsumption](#) (uint64_t runningUs)
to return the static energy consumption of a system under several us

Protected Attributes

- double [staticPower](#) = 0
static power of a system in W
- INTELLI::ConfigMapPtr [cfg](#) = nullptr

8.14.1 Detailed Description

The abstract class for all meters.

Note

default behaviors:

- create
- call [setConfig\(\)](#) to config this meter
- (optional) call [testStaticPower\(\)](#) to automatically test the static power of a device or [setStaticPower](#) to manually set the static power, if you want to exclude it
- call [startMeter\(\)](#) to start measurement
- (run your program)
- call [stopMeter\(\)](#) to stop measurement
- call [getE\(\)](#), [getPeak\(\)](#), etc to get the measurement results

8.14.2 Member Function Documentation

8.14.2.1 getStaicEnergyConsumption()

```
double DIVERSE_METER::AbstractMeter::getStaicEnergyConsumption (
    uint64_t runningUs )
```

to return the static energy consumption of a system under several us

Parameters

<i>runningUs</i>	The time in us of a running return the staticPower
------------------	--

8.14.2.2 setConfig()

```
virtual void DIVERSE_METER::AbstractMeter::setConfig (
    INTELLI::ConfigMapPtr _cfg ) [inline], [virtual]
```

to set the configmap

Parameters

<i>cfg</i>	the config map
------------	----------------

Reimplemented in [DIVERSE_METER::IntelMeter](#), and [DIVERSE_METER::EspMeterUart](#).

8.14.2.3 setStaticPower()

```
void DIVERSE_METER::AbstractMeter::setStaticPower (
    double _sp ) [inline]
```

to manually set the static power

Parameters

<code>_sp</code>	
------------------	--

8.14.2.4 testStaticPower()

```
void DIVERSE_METER::AbstractMeter::testStaticPower (
    uint64_t sleepingSecond )
```

to test the static power of a system by sleeping

Parameters

<code>sleepingSecond</code>	The seconds for sleep
-----------------------------	-----------------------

The documentation for this class was generated from the following files:

- include/Utils/Meters/[AbstractMeter.hpp](#)
- src/Utils/Meters/AbstractMeter.cpp

8.15 CANDY::AdSampling Class Reference

Public Member Functions

- **AdSampling** (int64_t d)
- void **set_transformed** (torch::Tensor *tm)
- torch::Tensor **transform** (torch::Tensor ta)
- void **set_threshold** (float threshold)
- void **set_step** (size_t step, float epsilon)
- float **distanceCompute_L2** (torch::Tensor ta, torch::Tensor tb)

Static Public Member Functions

- static torch::Tensor **getTransformMatrix** (int64_t dim)

The documentation for this class was generated from the following file:

- include/CANDY/HNSWNaive/AdSampling.h

8.16 BS::blocks< T1, T2, T > Class Template Reference

A helper class to divide a range into blocks. Used by `parallelize_loop()` and `push_loop()`.

```
#include <BS_thread_pool.hpp>
```

Public Member Functions

- `blocks` (const T1 first_index_, const T2 index_after_last_, const size_t num_blocks_)
Construct a blocks object with the given specifications.
- `T start` (const size_t i) const
Get the first index of a block.
- `T end` (const size_t i) const
Get the index after the last index of a block.
- `size_t get_num_blocks` () const
Get the number of blocks. Note that this may be different than the desired number of blocks that was passed to the constructor.
- `size_t get_total_size` () const
Get the total number of indices in the range.

8.16.1 Detailed Description

```
template<typename T1, typename T2, typename T = std::common_type_t<T1, T2>>
class BS::blocks< T1, T2, T >
```

A helper class to divide a range into blocks. Used by `parallelize_loop()` and `push_loop()`.

Template Parameters

<i>T1</i>	The type of the first index in the range. Should be a signed or unsigned integer.
<i>T2</i>	The type of the index after the last index in the range. Should be a signed or unsigned integer. If T1 is not the same as T2, a common type will be automatically inferred.
<i>T</i>	The common type of T1 and T2.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 blocks()

```
template<typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
BS::blocks< T1, T2, T >::blocks (
    const T1 first_index_,
    const T2 index_after_last_,
    const size_t num_blocks_ ) [inline]
```

Construct a blocks object with the given specifications.

Parameters

<i>first_index_</i>	The first index in the range.
<i>index_after_↵ last_</i>	The index after the last index in the range.
<i>num_blocks_</i>	The desired number of blocks to divide the range into.

8.16.3 Member Function Documentation

8.16.3.1 end()

```
template<typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
T BS::blocks< T1, T2, T >::end (
    const size_t i ) const [inline]
```

Get the index after the last index of a block.

Parameters

<i>i</i>	The block number.
----------	-------------------

Returns

The index after the last index.

8.16.3.2 get_num_blocks()

```
template<typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
size_t BS::blocks< T1, T2, T >::get_num_blocks ( ) const [inline]
```

Get the number of blocks. Note that this may be different than the desired number of blocks that was passed to the constructor.

Returns

The number of blocks.

8.16.3.3 `get_total_size()`

```
template<typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
size_t BS::blocks< T1, T2, T >::get_total_size ( ) const [inline]
```

Get the total number of indices in the range.

Returns

The total number of indices.

8.16.3.4 `start()`

```
template<typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
T BS::blocks< T1, T2, T >::start (
    const size_t i ) const [inline]
```

Get the first index of a block.

Parameters

<i>i</i>	The block number.
----------	-------------------

Returns

The first index.

The documentation for this class was generated from the following file:

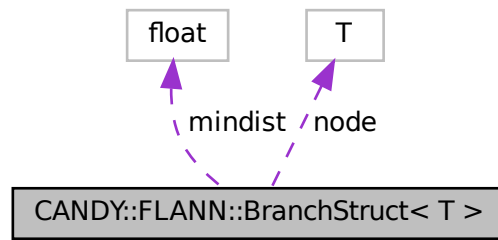
- [include/Utils/BS_thread_pool.hpp](#)

8.17 CANDY::FLANN::BranchStruct< T > Class Template Reference

The structure representing a branch point when finding neighbors in the tree.

```
#include <CANDY/FlannIndex/FlannUtils.h>
```


Collaboration diagram for CANDY::FLANN::BranchStruct< T >:



Public Member Functions

- **BranchStruct** (const T &n, float dist)
- bool **operator**< (const [BranchStruct](#)< T > &right) const

Public Attributes

- T **node**
- float **mindist**

8.17.1 Detailed Description

```
template<typename T>
class CANDY::FLANN::BranchStruct< T >
```

The structure representing a branch point when finding neighbors in the tree.

The documentation for this class was generated from the following file:

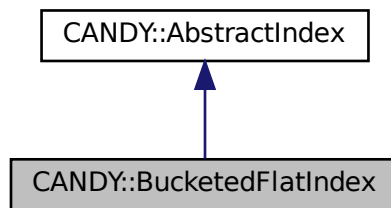
- include/CANDY/FlannIndex/FlannUtils.h

8.18 CANDY::BucketedFlatIndex Class Reference

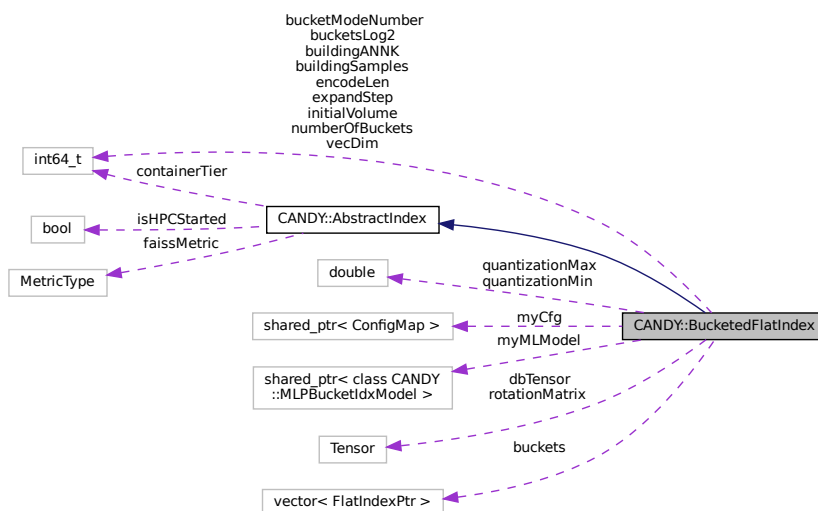
The class of splitting similar vectors into fixed number of buckets, each bucket is managed by [FlatIndex](#).

```
#include <CANDY/BucketedFlatIndex.h>
```

Inheritance diagram for CANDY::BucketedFlatIndex:



Collaboration diagram for CANDY::BucketedFlatIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Protected Member Functions

- uint64_t **encodeSingleRowMean** (torch::Tensor &tensor)
- uint64_t **encodeSingleRowLsh** (torch::Tensor &tensor)
- std::vector< uint64_t > **encodeMultiRows** (torch::Tensor &tensor)
- torch::Tensor **searchSingleRow** (torch::Tensor &q, uint64_t bkt, int64_t k)

search the k-NN of a query tensor, return the result tensors

Protected Attributes

- [INTELLI::ConfigMapPtr](#) **myCfg** = nullptr
- torch::Tensor **dbTensor**
- int64_t **vecDim** = 0
- int64_t **initialVolume** = 1000
- int64_t **expandStep** = 100
- int64_t **numberOfBuckets** = 1
- int64_t **buildingSamples** = -1
- int64_t **buildingANNK** = 10
- int64_t **bucketModeNumber**
- int64_t **bucketsLog2** = 0
- std::vector< [FlatIndexPtr](#) > **buckets**
- double **quantizationMax**
- double **quantizationMin**
- int64_t **encodeLen**
- torch::Tensor **rotationMatrix**
- [MLPBucketIdxModelPtr](#) **myMLModel** = nullptr

Additional Inherited Members

8.18.1 Detailed Description

The class of splitting similar vectors into fixed number of buckets, each bucket is managed by [FlatIndex](#).

Note

currently single thread

config parameters

- vecDim, the dimension of vectors, default 768, l64
- initialVolume, the initial volume of inline database tensor, default 1000, l64
- expandStep, the step of expanding inline database, default 100, l64
- numberOfBuckets, the number of titer buckets, default 1, l64, suggest 2^n
- bucketMode, the mode of assigning buckets, default 'mean', String, allow the following with its own parameters
 - 'mean': the bucket is assigned by uniform quantization of the mean, the quantization step is assigned by numberOfBuckets require following parameters
 - * quantizationMax the max value used for quantization, default 1, Double
 - * quantizationMin the min value used for quantization, default -1, Double
 - 'LSH': the bucket is assigned by LSH, and raw LSH encoding will be aggregated according to numberOfBuckets
 - * encodeLen, the length of LSH encoding, in bytes, default 1, l64

- * metricType, the type of AKNN metric, default L2, String
- * lshMatrixType, the type of lsh matrix, default gaussian, String
 - gaussian means a $N(0,1)$ LSH matrix
 - random means a random matrix where each value ranges from -0.5~0.5
- 'ML': the bucket is assigned by machine learning to generate bucket indices
 - encodeLen, the length of LSH encoding, in bytes, default 1, l64
 - metricType, the type of AKNN metric, default L2, String
 - cudaBuild whether or not use cuda to build model, l64, default 0
 - learningRate the learning rate for training, Double, default 0.01
 - hiddenLayerDim the dimension of hidden layer, l64, default the same as output layer
 - MLTrainBatchSize the batch size of ML training, l64, default 64
 - MLTrainMargin the margin value used in training, Double, default 2*0.1
 - MLTrainEpochs the number of epochs in training, l64, default 10

8.18.2 Member Function Documentation

8.18.2.1 deleteTensor()

```
bool CANDY::BucketedFlatIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.18.2.2 encodeMultiRows()

```
std::vector< uint64_t > CANDY::BucketedFlatIndex::encodeMultiRows (
    torch::Tensor & tensor ) [protected]
```

mean

lsh

8.18.2.3 insertTensor()

```
bool CANDY::BucketedFlatIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.18.2.4 loadInitialTensor()

```
bool CANDY::BucketedFlatIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.18.2.5 reviseTensor()

```
bool CANDY::BucketedFlatIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.18.2.6 searchSingleRow()

```
torch::Tensor CANDY::BucketedFlatIndex::searchSingleRow (
    torch::Tensor & q,
    uint64_t bkt,
    int64_t k ) [protected]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow only single rows
<i>bkt</i>	the bucket number which fits best
<i>k</i>	the returned neighbors

Returns

the result tensor

1. test whether the buckets[idx] has enough tensors,
2. if not, try to expand

search on the expanded dbTensor

8.18.2.7 searchTensor()

```
std::vector< torch::Tensor > CANDY::BucketedFlatIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.18.2.8 setConfig()

```
bool CANDY::BucketedFlatIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

@brief 1. common init

@brief 2.a init of mean mode

@brief 2.a init of lsh mode

Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

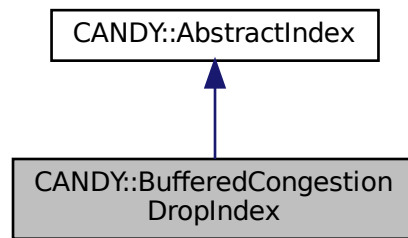
- include/CANDY/[BucketedFlatIndex.h](#)
- src/CANDY/[BucketedFlatIndex.cpp](#)

8.19 CANDY::BufferedCongestionDropIndex Class Reference

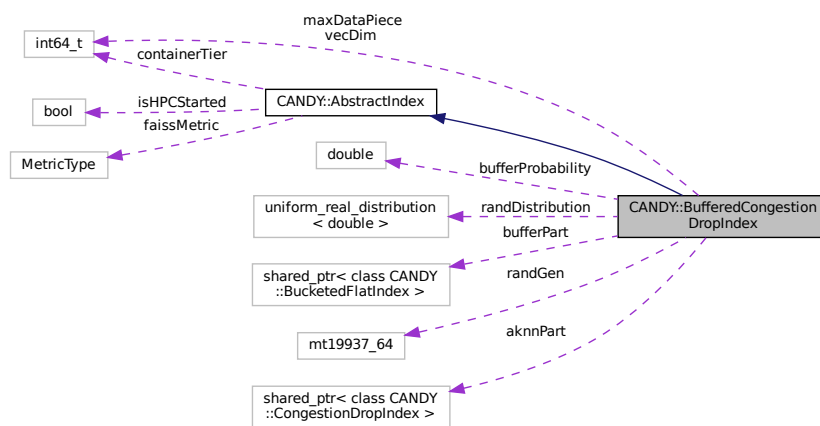
Similar to [CongestionDropIndex](#), but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under [BucketedFlatIndex](#) More detailed description with an image:

```
#include <CANDY/BufferedCongestionDropIndex.h>
```

Inheritance diagram for CANDY::BufferedCongestionDropIndex:



Collaboration diagram for CANDY::BufferedCongestionDropIndex:



Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)

- *search the k-NN of a query tensor, return the result tensors*
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (torch::Tensor &t)
offline build phase

Protected Member Functions

- [INTELLI::ConfigMapPtr generateBucketedFlatIndexConfig](#) (INTELLI::ConfigMapPtr cfg)
to generate the config map of inside [BucketedFlatIndex](#) from the top config
- virtual bool [insertTensorInline](#) (torch::Tensor &t)
insert a tensor to either bufferPart or aknnPart

Protected Attributes

- std::mt19937_64 **randGen**
- [BucketedFlatIndexPtr](#) **bufferPart** = nullptr
- [CongestionDropIndexPtr](#) **aknnPart** = nullptr
- std::uniform_real_distribution< double > **randDistribution**
- double **bufferProbability** = 0.5
- int64_t **maxDataPiece** = -1
- int64_t **vecDim**

Additional Inherited Members

8.19.1 Detailed Description

Similar to [CongestionDropIndex](#), but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under [BucketedFlatIndex](#) More detailed description with an image:

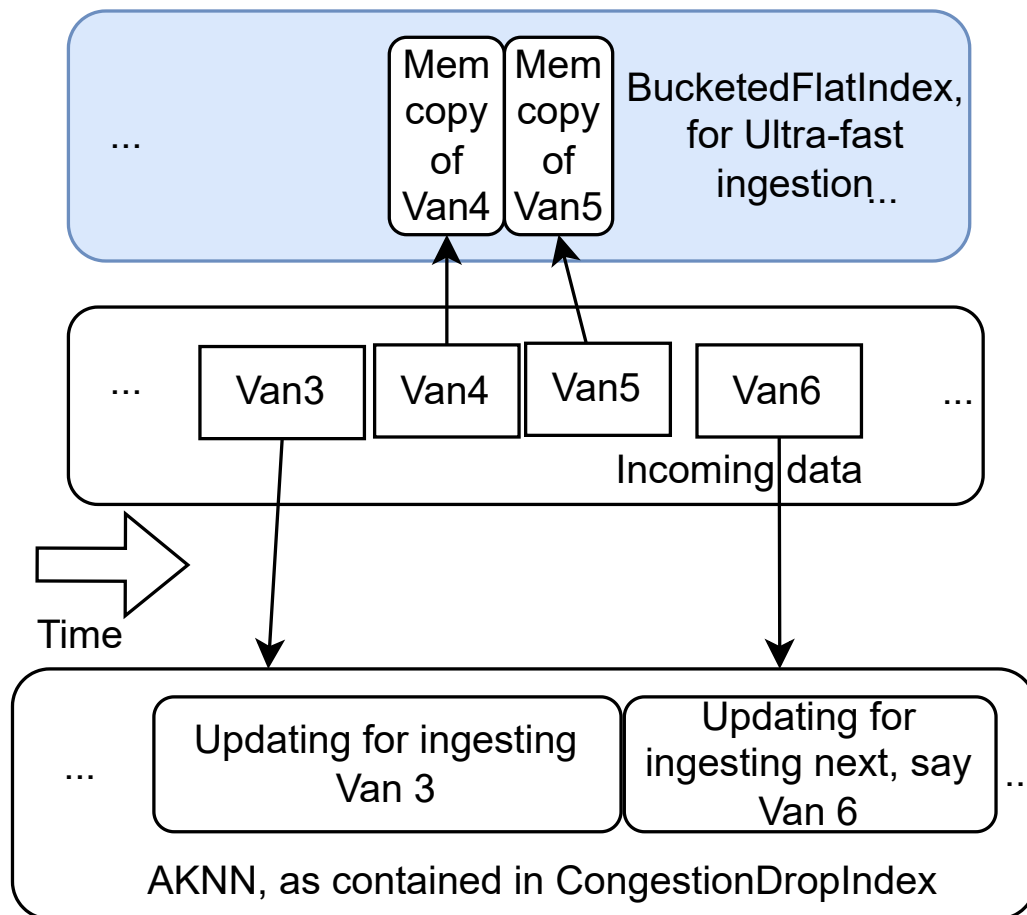


Figure 8.1 An overview of `BufferedCongestionDropIndex`

under [BucketedFlatIndex](#)

Note

The current decision of where to put data is just by probability parameters

- `vecDim`, the dimension of vectors, default 768, I64
- `bufferProbability`, the probability of ingesting data into buffer, default 0.5, Double
- `maxDataPiece`, the max piece of one data throwing into [CongestionDropIndex](#) or [BucketedFlatIndex](#), default -1 (full piece for each insert), I64

special parameters (For configuring the inside [CongestionDropIndex](#))

- `congestionDropWorker_algoTag` The algo tag of this worker, String, default flat
- `congestionDropWorker_queueSize` The input queue size of this worker, I64, default 10
- `parallelWorks` The number of parallel workers, I64, default 1 (set this to less than 0 will use `max hardware_concurrency`);
- `fineGrainedParallelInsert`, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, I64, default 0
- `congestionDrop`, whether or not drop the data when congestion occurs, I64, default 1
- `sharedBuild` whether let all sharding using shared build, 1, I64
- `singleWorkerOpt` whether optimize the searching under single worker, 1 I64

special parameters (For configuring the inside [BucketedFlatIndex](#))

- `buffer_initialVolume`, the initial volume of inline database tensor, default 1000, l64
- `buffer_expandStep`, the step of expanding inline database, default 100, l64
- `buffer_numberOfBuckets`, the number of titer buckets, default 1, l64, suggest 2^n
- `buffer_bucketMode`, the mode of assigning buckets, default 'mean', String, allow the following with its own parameters
 - 'mean': the bucket is assigned by uniform quantization of the mean, the quantization step is assigned by `numberOfBuckets` require following parameters
 - * `buffer_quantizationMax` the max value used for quantization, default 1, Double
 - * `buffer_quantizationMin` the min value used for quantization, default -1, Double
 - 'LSH': the bucket is assigned by LSH, and raw LSH encoding will be aggregated according to `numberOfBuckets`
 - * `buffer_encodeLen`, the length of LSH encoding, in bytes, default 1, l64
 - * `buffer_metricType`, the type of AKNN metric, default L2, String
 - * `buffer_lshMatrixType`, the type of lsh matrix, default gaussian, String
 - gaussian means a $N(0,1)$ LSH matrix
 - random means a random matrix where each value ranges from -0.5~0.5 @warning Make sure you are using 2D tensors!

8.19.2 Member Function Documentation

8.19.2.1 deleteTensor()

```
bool CANDY::BufferedCongestionDropIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

1. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.2 endHPC()

```
bool CANDY::BufferedCongestionDropIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.3 generateBucketedFlatIndexConfig()

```
INTELLI::ConfigMapPtr CANDY::BufferedCongestionDropIndex::generateBucketedFlatIndexConfig (
    INTELLI::ConfigMapPtr cfg ) [protected]
```

to generate the config map of inside [BucketedFlatIndex](#) from the top config

Parameters

<i>cfg</i>	the top config of this index
------------	------------------------------

Returns

the config for inside [BucketedFlatIndex](#)

8.19.2.4 insertTensor()

```
bool CANDY::BufferedCongestionDropIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.5 insertTensorInline()

```
bool CANDY::BufferedCongestionDropIndex::insertTensorInline (
    torch::Tensor & t ) [protected], [virtual]
```

insert a tensor to either bufferPart or aknnPart

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

8.19.2.6 loadInitialTensor()

```
bool CANDY::BufferedCongestionDropIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.7 offlineBuild()

```
bool CANDY::BufferedCongestionDropIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.8 reviseTensor()

```
bool CANDY::BufferedCongestionDropIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

only allow to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::BufferedCongestionDropIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.10 setConfig()

```
bool CANDY::BufferedCongestionDropIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.11 setFrozenLevel()

```
bool CANDY::BufferedCongestionDropIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.19.2.12 startHPC()

```
bool CANDY::BufferedCongestionDropIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented from [CANDY::AbstractIndex](#).

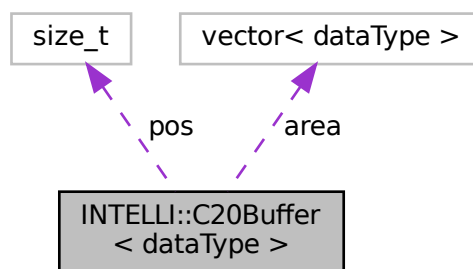
The documentation for this class was generated from the following files:

- [include/CANDY/BufferedCongestionDropIndex.h](#)
- [src/CANDY/BufferedCongestionDropIndex.cpp](#)

8.20 INTELLI::C20Buffer< dataType > Class Template Reference

```
#include <Utils/C20Buffers.hpp>
```

Collaboration diagram for INTELLI::C20Buffer< dataType >:



Public Member Functions

- void [reset](#) ()
reset this buffer, set pos back to 0
- [C20Buffer](#) (size_t len)
Init with original length of buffer.
- size_t [bufferSize](#) ()
To get how many elements are allowed in the buffer.
- size_t [size](#) ()
To get how many VALID elements are existed in the buffer.
- dataType * [data](#) ()
To get the original memory area ponter of data.
- dataType * [data](#) (size_t offset)
To get the original memory area ponter of data, with offset.
- size_t [append](#) (dataType da)
Append the data to the buffer.
- size_t [append](#) (dataType *da, size_t len)
Append the data to the buffer.

Public Attributes

- `std::vector< dataType > area`

Protected Attributes

- `size_t pos = 0`

8.20.1 Detailed Description

```
template<typename dataType>
class INTELLI::C20Buffer< dataType >
```

Template Parameters

<i>dataType</i>	The type of your buffering element
-----------------	------------------------------------

8.20.2 Constructor & Destructor Documentation

8.20.2.1 C20Buffer()

```
template<typename dataType >
INTELLI::C20Buffer< dataType >::C20Buffer (
    size_t len ) [inline]
```

Init with original length of buffer.

Parameters

<i>len</i>	The original length of buffer
------------	-------------------------------

8.20.3 Member Function Documentation

8.20.3.1 append() [1/2]

```
template<typename dataType >
size_t INTELLI::C20Buffer< dataType >::append (
    dataType * da,
    size_t len ) [inline]
```

Append the data to the buffer.

Parameters

<i>da</i>	Data to be appended, a buffer
<i>len</i>	the length of data

Note

Exceed length will lead to a `push_back` in vector

Returns

The valid size after this `append`

8.20.3.2 `append()` [2/2]

```
template<typename dataType >
size_t INTELLI::C20Buffer< dataType >::append (
    dataType da ) [inline]
```

Append the data to the buffer.

Parameters

<i>da</i>	Data to be appended
-----------	---------------------

Note

Exceed length will lead to a `push_back` in vector

Returns

The valid size after this `append`

8.20.3.3 `bufferSize()`

```
template<typename dataType >
size_t INTELLI::C20Buffer< dataType >::bufferSize ( ) [inline]
```

To get how many elements are allowed in the buffer.

Returns

The size of buffer area, i.e., `area.size()`

Note

: This is NOT the size of valid data

See also

[size](#)

8.20.3.4 data() [1/2]

```
template<typename dataType >
dataType* INTELLI::C20Buffer< dataType >::data ( ) [inline]
```

To get the original memory area pointer of data.

Returns

The memory area address (pointer) that stores the data

8.20.3.5 data() [2/2]

```
template<typename dataType >
dataType* INTELLI::C20Buffer< dataType >::data (
    size_t offset ) [inline]
```

To get the original memory area pointer of data, with offset.

Parameters

<i>offset</i>	Offset of data
---------------	----------------

Returns

The memory area address (pointer) that stores the data

Warning

Please ensure the offset is NOT larger than the `area.size()-1`

8.20.3.6 size()

```
template<typename dataType >
size_t INTELLI::C20Buffer< dataType >::size ( ) [inline]
```

To get how many VALID elements are existed in the buffer.

Returns

The size of VALID elements

Note

: This is NOT the size of total buffer

See also

[bufferSize](#)

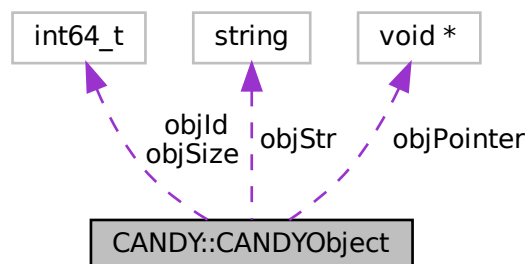
The documentation for this class was generated from the following file:

- [include/Utils/C20Buffers.hpp](#)

8.21 CANDY::CANDYObject Class Reference

A generic object class to link string or void * pointers.

Collaboration diagram for CANDY::CANDYObject:



Public Member Functions

- void [setStr](#) (std::string str)
to set the string
- std::string [getStr](#) ()
to get the string

Public Attributes

- std::string **objStr**
- void * **objPointer** = nullptr
- int64_t **objSize** = 0
- int64_t **objId** = -1

8.21.1 Detailed Description

A generic object class to link string or void * pointers.

Todo to finish the functions of setting void * pointers

8.21.2 Member Function Documentation

8.21.2.1 getStr()

```
std::string CANDY::CANDYObject::getStr ( )
```

to get the string

Returns

the objStr

8.21.2.2 setStr()

```
void CANDY::CANDYObject::setStr (
    std::string str )
```

to set the string

Parameters

<i>str</i>	the string
------------	------------

Returns

void

The documentation for this class was generated from the following files:

- include/CANDY/[CANDYObject.h](#)
- src/CANDY/CANDYObject.cpp

8.22 cl_char16 Union Reference

Public Member Functions

- cl_char **CL_ALIGNED** (16) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.23 cl_char2 Union Reference

Public Member Functions

- cl_char **CL_ALIGNED** (2) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.24 cl_char4 Union Reference

Public Member Functions

- cl_char **CL_ALIGNED** (4) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.25 cl_char8 Union Reference

Public Member Functions

- cl_char **CL_ALIGNED** (8) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.26 cl_double16 Union Reference

Public Member Functions

- cl_double **CL_ALIGNED** (128) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.27 cl_double2 Union Reference

Public Member Functions

- cl_double **CL_ALIGNED** (16) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.28 cl_double4 Union Reference

Public Member Functions

- cl_double **CL_ALIGNED** (32) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.29 cl_double8 Union Reference

Public Member Functions

- cl_double **CL_ALIGNED** (64) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.30 cl_float16 Union Reference

Public Member Functions

- cl_float **CL_ALIGNED** (64) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.31 cl_float2 Union Reference

Public Member Functions

- cl_float **CL_ALIGNED** (8) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.32 cl_float4 Union Reference

Public Member Functions

- cl_float **CL_ALIGNED** (16) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.33 cl_float8 Union Reference

Public Member Functions

- cl_float **CL_ALIGNED** (32) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.34 cl_half16 Union Reference

Public Member Functions

- cl_half **CL_ALIGNED** (32) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.35 cl_half2 Union Reference

Public Member Functions

- cl_half **CL_ALIGNED** (4) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.36 cl_half4 Union Reference

Public Member Functions

- cl_half **CL_ALIGNED** (8) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.37 cl_half8 Union Reference

Public Member Functions

- cl_half **CL_ALIGNED** (16) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.38 cl_int16 Union Reference

Public Member Functions

- cl_int **CL_ALIGNED** (64) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.39 cl_int2 Union Reference

Public Member Functions

- cl_int **CL_ALIGNED** (8) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.40 cl_int4 Union Reference

Public Member Functions

- cl_int **CL_ALIGNED** (16) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.41 cl_int8 Union Reference

Public Member Functions

- cl_int **CL_ALIGNED** (32) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.42 cl_long16 Union Reference

Public Member Functions

- cl_long **CL_ALIGNED** (128) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.43 cl_long2 Union Reference

Public Member Functions

- cl_long **CL_ALIGNED** (16) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.44 cl_long4 Union Reference

Public Member Functions

- cl_long **CL_ALIGNED** (32) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.45 cl_long8 Union Reference

Public Member Functions

- cl_long **CL_ALIGNED** (64) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.46 cl_short16 Union Reference

Public Member Functions

- cl_short **CL_ALIGNED** (32) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.47 cl_short2 Union Reference

Public Member Functions

- cl_short **CL_ALIGNED** (4) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.48 cl_short4 Union Reference

Public Member Functions

- cl_short **CL_ALIGNED** (8) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.49 cl_short8 Union Reference

Public Member Functions

- cl_short **CL_ALIGNED** (16) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.50 cl_uchar16 Union Reference

Public Member Functions

- cl_uchar **CL_ALIGNED** (16) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.51 cl_uchar2 Union Reference

Public Member Functions

- cl_uchar **CL_ALIGNED** (2) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.52 cl_uchar4 Union Reference

Public Member Functions

- cl_uchar **CL_ALIGNED** (4) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.53 cl_uchar8 Union Reference

Public Member Functions

- cl_uchar **CL_ALIGNED** (8) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.54 cl_uint16 Union Reference

Public Member Functions

- cl_uint **CL_ALIGNED** (64) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.55 cl_uint2 Union Reference

Public Member Functions

- cl_uint **CL_ALIGNED** (8) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.56 cl_uint4 Union Reference

Public Member Functions

- cl_uint **CL_ALIGNED** (16) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.57 cl_uint8 Union Reference

Public Member Functions

- cl_uint **CL_ALIGNED** (32) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.58 cl_ulong16 Union Reference

Public Member Functions

- cl_ulong **CL_ALIGNED** (128) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.59 cl_ulong2 Union Reference

Public Member Functions

- cl_ulong **CL_ALIGNED** (16) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.60 cl_ulong4 Union Reference

Public Member Functions

- cl_ulong **CL_ALIGNED** (32) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.61 cl_ulong8 Union Reference

Public Member Functions

- cl_ulong **CL_ALIGNED** (64) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.62 cl_ushort16 Union Reference

Public Member Functions

- cl_ushort **CL_ALIGNED** (32) s[16]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.63 cl_ushort2 Union Reference

Public Member Functions

- cl_ushort **CL_ALIGNED** (4) s[2]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.64 cl_ushort4 Union Reference

Public Member Functions

- cl_ushort **CL_ALIGNED** (8) s[4]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.65 cl_ushort8 Union Reference

Public Member Functions

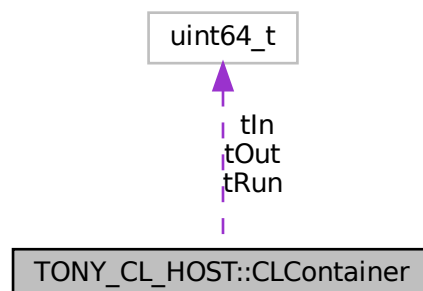
- cl_ushort **CL_ALIGNED** (16) s[8]

The documentation for this union was generated from the following file:

- include/CL/cl_platform.h

8.66 TONY_CL_HOST::CLContainer Class Reference

Collaboration diagram for TONY_CL_HOST::CLContainer:



Public Member Functions

- **CLContainer** (cl_uint id, cl_device_type type, string kernelName)
- **CLContainer** (cl_uint id, cl_device_type type, string kernelName, string clName)
- **CLContainer** (cl_uint id, cl_device_type type, string kernelName, char *filenameFull)
- void **setWorkDimension** (int nd)
- void **saveProgram** (char *outName)
- void **addHostOutPara** ([HostPara](#) par)
- void **addHostInPara** ([HostPara](#) par)
- void **resetHostIn** (size_t idx, [HostPara](#) par)
- void **resetHostOut** (size_t idx, [HostPara](#) par)
- void **clearPar** ()
- void **addBoundaryValue** (uint64_t bnd)
- void **resetBoundary** (size_t idx, uint64_t bnd)
- void **execute** (size_t globalSize, size_t localSize)
- void **execute** (std::vector< size_t > gs, std::vector< size_t > ls)

Public Attributes

- uint64_t **tIn**
- uint64_t **tRun**
- uint64_t **tOut**

The documentation for this class was generated from the following files:

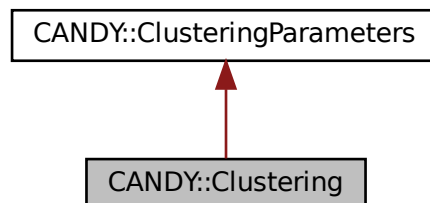
- include/CL/CLContainer.hpp
- src/CLContainer.cpp

8.67 CANDY::Clustering Class Reference

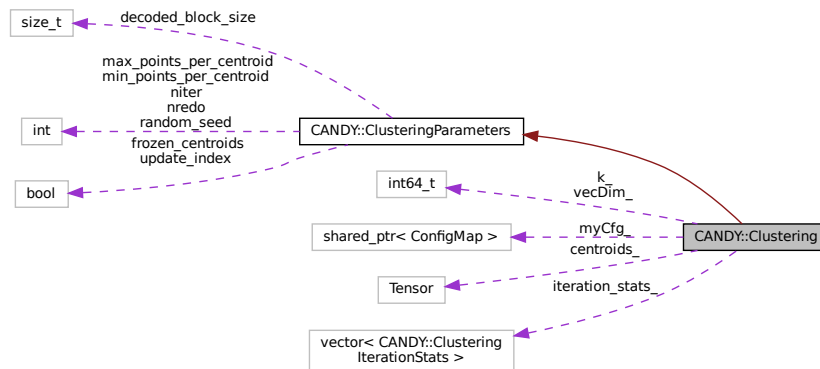
class for naive K-means clustering

```
#include <CANDY/PQIndex/Clustering.h>
```

Inheritance diagram for CANDY::Clustering:



Collaboration diagram for CANDY::Clustering:



Public Member Functions

- **Clustering** (int64_t vecDim, int64_t k)
- void **reset** ()
- auto **getCentroids** () -> torch::Tensor
- void **train** (size_t nx, const torch::Tensor x_in, faiss::IndexFlatL2 *index, const torch::Tensor *weights)
train the clustering using tensor based on IndexFlatL2 with weights
- double **imbalance_factor** (size_t n, int64_t k, int64_t *assign)
compute the imbalance factor of an assignment
- void **computeCentroids** (int64_t d, int64_t k, size_t n, int64_t k_frozen, const torch::Tensor x_in, const int64_t *assign, const torch::Tensor *weights, torch::Tensor *hassign, torch::Tensor *centroids)
compute the centroids of input vectors
- int **splitClusters** (int64_t d, int64_t k, size_t n, int64_t k_frozen, torch::Tensor *hassign, torch::Tensor *centroids)
balance the assignment by averaging between a big cluster and a null cluster

Public Attributes

- std::vector< [ClusteringIterationStats](#) > **iteration_stats_**

Protected Attributes

- [INTELLI::ConfigMapPtr](#) **myCfg_** = nullptr
- int64_t **vecDim_** = 0
dimension of vectors
- int64_t **k_** = 256
number of centroids
- torch::Tensor **centroids_**
*centroids vector size : (k * d)*

8.67.1 Detailed Description

class for naive K-means clustering

Todo current build of centroids still depends on IndexFlatL2, perhaps re-implemented in a total tensor manner

- [train](#)

8.67.2 Member Function Documentation

8.67.2.1 computeCentroids()

```
void CANDY::Clustering::computeCentroids (
    int64_t d,
    int64_t k,
    size_t n,
    int64_t k_frozen,
    const torch::Tensor x_in,
    const int64_t * assign,
    const torch::Tensor * weights,
    torch::Tensor * hassign,
    torch::Tensor * centroids )
```

compute the centroids of input vectors

Parameters

<i>d</i>	dim of vectors
<i>k</i>	number of centroids
<i>n</i>	number of input vectors
<i>k_frozen</i>	number of frozen centroids which remain intact in this computation
<i>x_in</i>	input vectors as Tensor
<i>assign</i>	assignment array for n vectors
<i>weights</i>	weights to compute centroids
<i>hassign</i>	histogram of k centroids
<i>centroids</i>	centroids after computation

8.67.2.2 imbalance_factor()

```
double CANDY::Clustering::imbalance_factor (
    size_t n,
    int64_t k,
    int64_t * assign )
```

compute the imbalance factor of an assignment

Parameters

<i>n</i>	number of input vectors
<i>k</i>	number of centroids
<i>assign</i>	assignment of centroid clustering

Returns

imbalance factor of the assignment

8.67.2.3 splitClusters()

```
int CANDY::Clustering::splitClusters (
    int64_t d,
    int64_t k,
    size_t n,
    int64_t k_frozen,
    torch::Tensor * hassign,
    torch::Tensor * centroids )
```

balance the assignment by averaging between a big cluster and a null cluster

Parameters

<i>d</i>	dim of vectors
<i>k</i>	number of centroids
<i>n</i>	number of input vectors
<i>k_frozen</i>	number of frozen centroids which remain intact
<i>hassign</i>	histogram of k centroids
<i>centroids</i>	centroids after computation

Returns

8.67.2.4 train()

```
void CANDY::Clustering::train (
    size_t nx,
    const torch::Tensor x_in,
    faiss::IndexFlatL2 * index,
    const torch::Tensor * weights )
```

train the clustering using tensor based on IndexFlatL2 with weights

Parameters

<i>nx</i>	number of input vectors
<i>x_in</i>	input vectors as Tensor
<i>index</i>	index upon which to search and evaluate during clustering
<i>weights</i>	weights to compute centroids after assignment

The documentation for this class was generated from the following files:

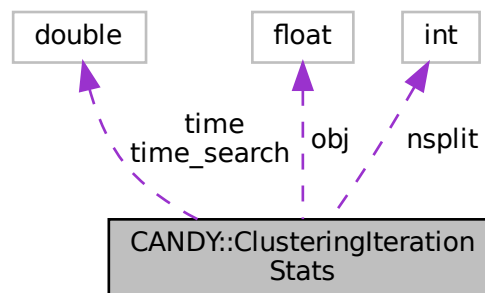
- include/CANDY/PQIndex/Clustering.h
- src/CANDY/PQIndex/Clustering.cpp

8.68 CANDY::ClusteringIterationStats Class Reference

struct to record performance of clustering during iterations

```
#include <CANDY/PQIndex/Clustering.h>
```

Collaboration diagram for CANDY::ClusteringIterationStats:



Public Attributes

- float **obj**
- double **time**
- double **time_search**
- int **nsplit**

8.68.1 Detailed Description

struct to record performance of clustering during iterations

The documentation for this class was generated from the following file:

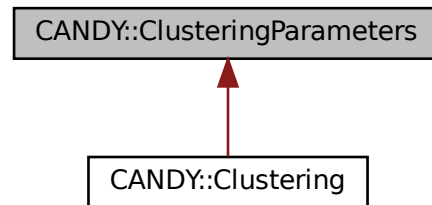
- include/CANDY/PQIndex/Clustering.h

8.69 CANDY::ClusteringParameters Class Reference

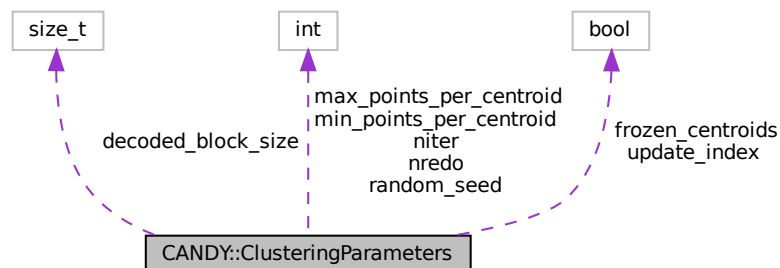
Class for the clustering parameters to be set before training/building.

```
#include <CANDY/PQIndex/Clustering.h>
```

Inheritance diagram for CANDY::ClusteringParameters:



Collaboration diagram for CANDY::ClusteringParameters:



Public Attributes

- int `niter` = 25
number of clustering iterations
- int `nredo` = 1
number of redoes
- bool `update_index` = false
re=train index after each iteration
- bool `frozen_centroids` = false
whether subset of centroids remain intact during each iteration
- int `min_points_per_centroid` = 39
- int `max_points_per_centroid` = 256
- int `random_seed` = 1919810
- size_t `decoded_block_size` = 32768
training batch size of codec decoder

8.69.1 Detailed Description

Class for the clustering parameters to be set before training/building.

The documentation for this class was generated from the following file:

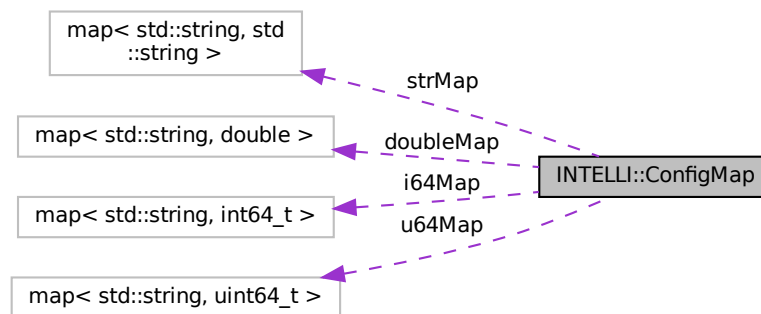
- include/CANDY/PQIndex/Clustering.h

8.70 INTELLI::ConfigMap Class Reference

The unified map structure to store configurations in a key-value style.

```
#include <Utils/ConfigMap.hpp>
```

Collaboration diagram for INTELLI::ConfigMap:



Public Member Functions

- void `edit` (const std::string &key, uint64_t value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void `edit` (const std::string &key, int64_t value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void `edit` (const std::string &key, double value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- void `edit` (const std::string &key, std::string value)
Edit the config map. If not exit the config, will create new, or will overwrite.
- bool `existU64` (const std::string &key)
To detect whether the key exists and related to a U64.
- bool `existI64` (const std::string &key)
To detect whether the key exists and related to a I64.
- bool `existDouble` (const std::string &key)
To detect whether the key exists and related to a double.
- bool `existString` (const std::string &key)
To detect whether the key exists and related to a std::string.

- bool **exist** (const std::string &key)
To detect whether the key exists.
- uint64_t **getU64** (const std::string &key)
To get a U64 value by key.
- int64_t **getI64** (const std::string &key)
To get a I64 value by key.
- double **getDouble** (const std::string &key)
To get a double value by key.
- std::string **getString** (const std::string &key)
To get a std::string value by key.
- std::string **toString** (const std::string &separator="\t", std::string newLine="\n")
convert the whole map to std::string and return
- bool **fromString** (const std::string src, const std::string &separator="\t", std::string newLine="\n")
load the map from some external string
- void **cloneInto** (ConfigMap &dest)
clone this config into destination
- void **loadFrom** (ConfigMap &src)
load some information an external one
- bool **toFile** (const std::string &fname, const std::string &separator=",", std::string newLine="\n")
convert the whole map to file
- bool **fromFile** (const std::string &fname, std::string separator=",", std::string newLine="\n")
update the whole map from file
- bool **fromCArg** (const int argc, char **argv)
update the whole map from c/c++ program's args
- int64_t **tryI64** (const string &key, int64_t defaultValue=0, bool showWarning=false)
Try to get an I64 from config map, if not exist, use default value instead.
- std::map< std::string, std::string > **getStrMap** ()
return the map of string
- std::map< std::string, int64_t > **getI64Map** ()
return the map of I64
- std::map< std::string, double > **getDoubleMap** ()
return the map of I64
- uint64_t **tryU64** (const string &key, uint64_t defaultValue=0, bool showWarning=false)
Try to get an U64 from config map, if not exist, use default value instead.
- double **tryDouble** (const string &key, double defaultValue=0, bool showWarning=false)
Try to get a double from config map, if not exist, use default value instead.
- string **tryString** (const string &key, const string &defaultValue="", bool showWarning=false)
Try to get an String from config map, if not exist, use default value instead.

Protected Member Functions

- void **smartParase** (std::string key, std::string value)

Static Protected Member Functions

- static void **spilt** (const std::string s, const std::string &c, vector< std::string > &v)

Protected Attributes

- `std::map< std::string, uint64_t > u64Map`
- `std::map< std::string, int64_t > i64Map`
- `std::map< std::string, double > doubleMap`
- `std::map< std::string, std::string > strMap`

8.70.1 Detailed Description

The unified map structure to store configurations in a key-value style.

Note

Require [IntelliLog](#) Util package

The documentation for this class was generated from the following file:

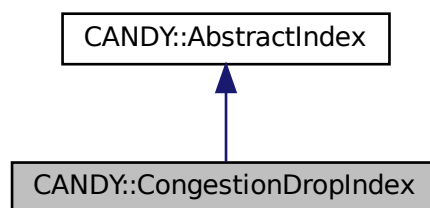
- `include/Utils/ConfigMap.hpp`

8.71 CANDY::CongestionDropIndex Class Reference

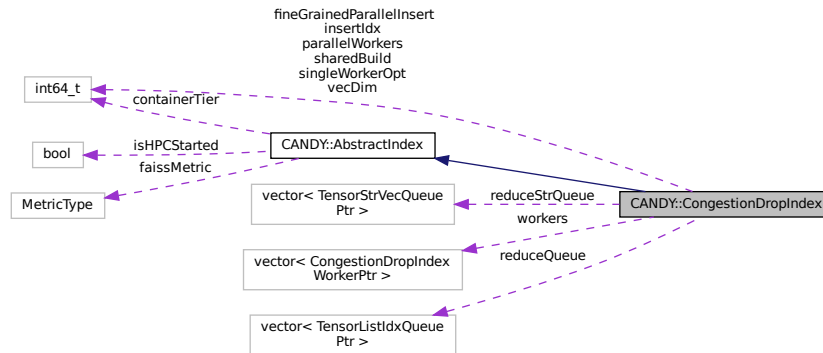
A container index to evaluate other bottom index, will just drop the data if congestion occurs, also support the data sharding parallelism.

```
#include <CANDY/CongestionDropIndex.h>
```

Inheritance diagram for CANDY::CongestionDropIndex:



Collaboration diagram for CANDY::CongestionDropIndex:



Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (torch::Tensor &t)
offline build phase
- virtual bool [waitPendingOperations](#) ()
a busy waiting for all pending operations to be done
- virtual bool [loadInitialStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)
- virtual bool [insertStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
insert a string object

- virtual bool [deleteStringObject](#) (torch::Tensor &t, int64_t k=1)
delete tensor along with its corresponding string object
- virtual std::vector< std::vector< std::string > > [searchStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects
- virtual std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > [searchTensorAndStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects and original tensors

Public Attributes

- std::vector< TensorListIdxQueuePtr > **reduceQueue**
- std::vector< TensorStrVecQueuePtr > **reduceStrQueue**

Protected Member Functions

- void **insertTensorInline** (torch::Tensor &t)
- void **partitionBuildInLine** (torch::Tensor &t)
- void **partitionLoadInLine** (torch::Tensor &t)
- void **insertStringInline** (torch::Tensor &t, std::vector< string > &s)
- void **partitionLoadStringInLine** (torch::Tensor &t, std::vector< string > &s)

Protected Attributes

- int64_t **parallelWorkers**
- int64_t **insertIdx**
- std::vector< [CongestionDropIndexWorkerPtr](#) > **workers**
- int64_t **vecDim**
- int64_t **fineGrainedParallelInsert**
- int64_t **sharedBuild**
- int64_t **singleWorkerOpt**

8.71.1 Detailed Description

A container index to evaluate other bottom index, will just drop the data if congestion occurs, also support the data sharding parallelism.

Note

When there is only one worker, will only R/W lock for concurrency control, no sequential guarantee, different from [ParallelPartitionIndex](#)

Warning

Don't mix the usage of tensor-only I/O and tensor-string hybrid I/O in one indexing class
remember to call `starHPC` and [endHPC](#)

Note

special parameters

- `congestionDropWorker_algoTag` The algo tag of this worker, String, default flat
- `congestionDropWorker_queueSize` The input queue size of this worker, l64, default 10
- `parallelWorks` The number of paraller workers, l64, default 1 (set this to less than 0 will use max hardware_concurrency);
- `vecDim`, the dimension of vectors, default 768, l64
- `fineGrainedParallelInsert`, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, l64, default 0
- `congestionDrop`, whether or not drop the data when congestion occurs, l64, default 1
- `sharedBuild` whether let all sharding using shared build, 1, l64
- `singleWorkerOpt` whether optimize the searching under single worker, 1 l64 @warning Make sure you are using 2D tensors!

8.71.2 Member Function Documentation

8.71.2.1 deleteStringObject()

```
bool CANDY::CongestionDropIndex::deleteStringObject (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete tensor along with its corresponding string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the delet is successful

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.2 deleteTensor()

```
bool CANDY::CongestionDropIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.3 endHPC()

```
bool CANDY::CongestionDropIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.4 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::CongestionDropIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.5 insertStringObject()

```
bool CANDY::CongestionDropIndex::insertStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

insert a string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>strs</i>	the corresponding list of strings

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.6 insertTensor()

```
bool CANDY::CongestionDropIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.7 loadInitialStringObject()

```
bool CANDY::CongestionDropIndex::loadInitialStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row •
<i>strs</i>	the corresponding list of strings

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.8 loadInitialTensor()

```
bool CANDY::CongestionDropIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.9 offlineBuild()

```
bool CANDY::CongestionDropIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.10 rawData()

```
torch::Tensor CANDY::CongestionDropIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.11 reviseTensor()

```
bool CANDY::CongestionDropIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

only allow to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.12 searchStringObject()

```
std::vector< std::vector< std::string > > CANDY::CongestionDropIndex::searchStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<std::vector<std::string>> the result object for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.13 searchTensor()

```
std::vector< torch::Tensor > CANDY::CongestionDropIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.14 searchTensorAndStringObject()

```
std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > CANDY↔
::CongestionDropIndex::searchTensorAndStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects and original tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::tuple<std::vector<torch::Tensor>,std::vector<std::vector<std::string>>>`

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.15 setConfig()

```
bool CANDY::CongestionDropIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.16 setFrozenLevel()

```
bool CANDY::CongestionDropIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.17 startHPC()

```
bool CANDY::CongestionDropIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.71.2.18 waitPendingOperations()

```
bool CANDY::CongestionDropIndex::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Note

in this index, there are may be some un-commited write due to the parallel queues

Returns

bool, whether the waiting is actually done;

Reimplemented from [CANDY::AbstractIndex](#).

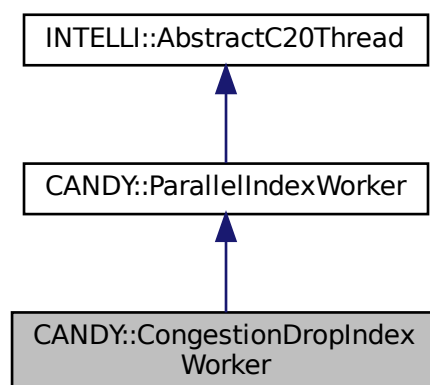
The documentation for this class was generated from the following files:

- include/CANDY/[CongestionDropIndex.h](#)
- src/CANDY/[CongestionDropIndex.cpp](#)

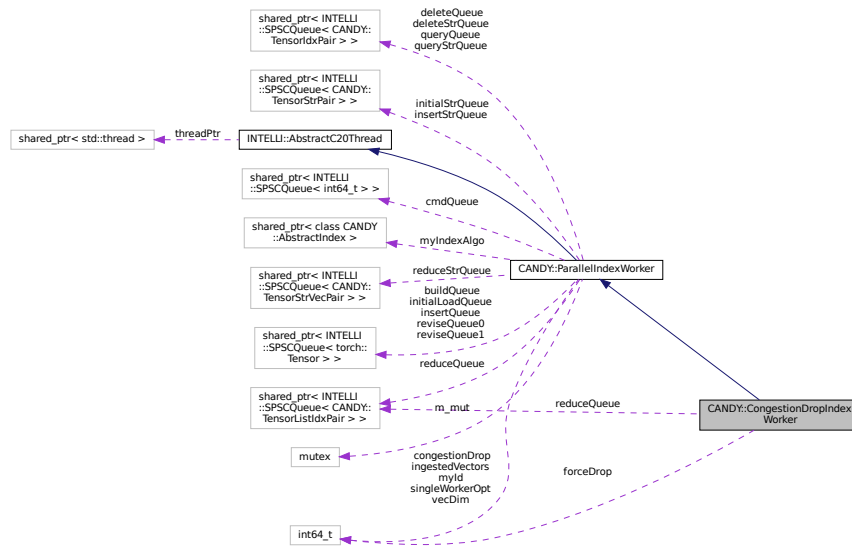
8.72 CANDY::CongestionDropIndexWorker Class Reference

A worker class to container bottom indexings, will just drop new element if congestion occurs.

Inheritance diagram for CANDY::CongestionDropIndexWorker:



Collaboration diagram for CANDY::CongestionDropIndexWorker:



Public Member Functions

- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors

Public Attributes

- TensorListIdxQueuePtr [reduceQueue](#)

Protected Attributes

- int64_t [forceDrop](#) = 1

Additional Inherited Members

8.72.1 Detailed Description

A worker class to container bottom indexings, will just drop new element if congestion occurs.

Note

special parameters

- congestionDropWorker_algoTag The algo tag of this worker, String, default flat
- congestionDropWorker_queueSize The input queue size of this worker, I64, default 10
- congestionDrop, whether or not drop the data when congestion occurs, I64, default 1 -vecDim the dimension of vectors, I674, default 768

The documentation for this class was generated from the following files:

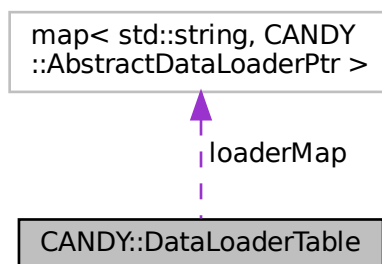
- include/CANDY/CongestionDropIndex/[CongestionDropIndexWorker.h](#)
- src/CANDY/CongestionDropIndex/[CongestionDropIndexWorker.cpp](#)

8.73 CANDY::DataLoaderTable Class Reference

The table class to index all Data loaders.

```
#include <DataLoader/DataLoaderTable.h>
```

Collaboration diagram for CANDY::DataLoaderTable:



Public Types

- typedef std::shared_ptr< class [CANDY::DataLoaderTable](#) > [DataLoaderTablePtr](#)
The class to describe a shared pointer to [DataLoaderTable](#).

Public Member Functions

- [DataLoaderTable](#) ()
The constructing function.
- void [registerNewDataLoader](#) ([CANDY::AbstractDataLoaderPtr](#) dnew, std::string tag)
To register a new loader.
- [CANDY::AbstractDataLoaderPtr](#) [findDataLoader](#) (std::string name)
find a dataloader in the table according to its name

Protected Attributes

- `std::map< std::string, CANDY::AbstractDataLoaderPtr >` `loaderMap`

8.73.1 Detailed Description

The table class to index all Data loaders.

Note

Default behavior

- create
- (optional) call [registerNewDataLoader](#) for new loader
- find a loader by [findDataLoader](#) using its tag

default tags

- random [RandomDataLoader](#)
- fvecs [FVECSDataLoader](#)
- hdf5 [HDF5DataLoader](#)
- zipf [ZipfDataLoader](#)
- expFamily [ExpFamilyDataLoader](#)
- exp, the exponential distribution in [ExpFamilyDataLoader](#)
- beta, the beta distribution in [ExpFamilyDataLoader](#)
- gaussian, the beta distribution in [ExpFamilyDataLoader](#)
- poisson, the poisson distribution in [ExpFamilyDataLoader](#)

8.73.2 Constructor & Destructor Documentation

8.73.2.1 DataLoaderTable()

```
CANDY::DataLoaderTable::DataLoaderTable ( )
```

The constructing function.

Note

If new DataLoader wants to be included by default, please revise the following in *.cpp
revise me if you need new loader

more specific loader oin exp family

8.73.3 Member Function Documentation

8.73.3.1 findDataLoader()

```
CANDY::AbstractDataLoaderPtr CANDY::DataLoaderTable::findDataLoader (
    std::string name ) [inline]
```

find a dataloader in the table according to its name

Parameters

<i>name</i>	The nameTag of loader
-------------	-----------------------

Returns

The DataLoader, nullptr if not found

8.73.3.2 registerNewDataLoader()

```
void CANDY::DataLoaderTable::registerNewDataLoader (
    CANDY::AbstractDataLoaderPtr dnew,
    std::string tag ) [inline]
```

To register a new loader.

Parameters

<i>onew</i>	The new operator
<i>tag</i>	The name tag

The documentation for this class was generated from the following files:

- include/DataLoader/[DataLoaderTable.h](#)
- src/DataLoader/DataLoaderTable.cpp

8.74 default_attrs Struct Reference

The low-level perf descriptions passed to OS.

```
#include <ThreadPerf.hpp>
```

8.74.1 Detailed Description

The low-level perf descriptions passed to OS.

The low-level perf events send to OS call, don't touch me.

The documentation for this struct was generated from the following file:

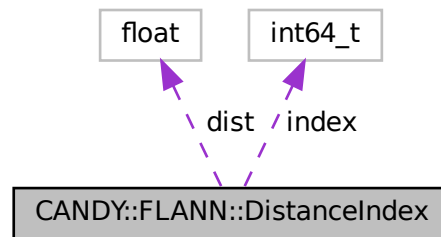
- include/Utils/[ThreadPerf.hpp](#)

8.75 CANDY::FLANN::DistanceIndex Class Reference

The structure representing a vectors' distance with the query along with its index.

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

Collaboration diagram for CANDY::FLANN::DistanceIndex:



Public Member Functions

- **DistanceIndex** (float d, int64_t i)
- bool **operator<** (const [DistanceIndex](#) &right) const

Public Attributes

- float **dist**
- int64_t **index**

8.75.1 Detailed Description

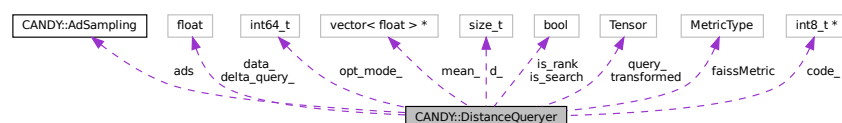
The structure representing a vectors' distance with the query along with its index.

The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.76 CANDY::DistanceQueryer Class Reference

Collaboration diagram for CANDY::DistanceQueryer:



Public Types

- typedef int64_t **opt_mode_t**

Public Member Functions

- **DistanceQueryer** (size_t d)
- float **operator()** (INTELLI::TensorPtr idx)
compute the distance between given idx's vector and query vector
- float **operator()** (const int8_t *code)
- float **lvq_first_level** (const float *x, const size_t len, int8_t *codes)
- void **lvq_second_level** (const float *x, const size_t len, int8_t *codes, float delta)
- float **symmetric_dis** (INTELLI::TensorPtr i, INTELLI::TensorPtr j)
- void **set_query** (torch::Tensor &x)
- int8_t * **compute_code** (INTELLI::TensorPtr idx)
- torch::Tensor **compute_transformed** (INTELLI::TensorPtr idx)
- void **set_mode** (opt_mode_t opt_mode, faiss::MetricType metric)
- void **set_rank** (bool rank)
- void **set_search** (bool search)
- float **int8vec_IP** (const int8_t *x, const int8_t *y, size_t d)
- float **fvec_IP** (const float *x, const float *y, size_t d)
- float **int8vec_L2** (const int8_t *x, const int8_t *y, size_t d)
- float **fvec_L2** (const float *x, const float *y, size_t d)

Public Attributes

- opt_mode_t **opt_mode_** = OPT_VANILLA
- faiss::MetricType **faissMetric** = faiss::METRIC_L2
- size_t **d_**
- torch::Tensor **query_**
- float * **data_**
- std::vector< float > * **mean_**
used for LVQ
- bool **is_rank** = false
- bool **is_search** = false
- int8_t * **code_** = nullptr
- float **delta_query_** = 0.0
- AdSampling * **ads** = nullptr
used for AdSAMPLING
- torch::Tensor **transformed**

8.76.1 Member Function Documentation

8.76.1.1 operator()

```
float CANDY::DistanceQueryer::operator() (
    INTELLI::TensorPtr idx ) [inline]
```

compute the distance between given idx's vector and query vector

Parameters

<code>idx</code>	the target vector to be computed with query vector
------------------	--

Returns

L2 Distance

The documentation for this class was generated from the following file:

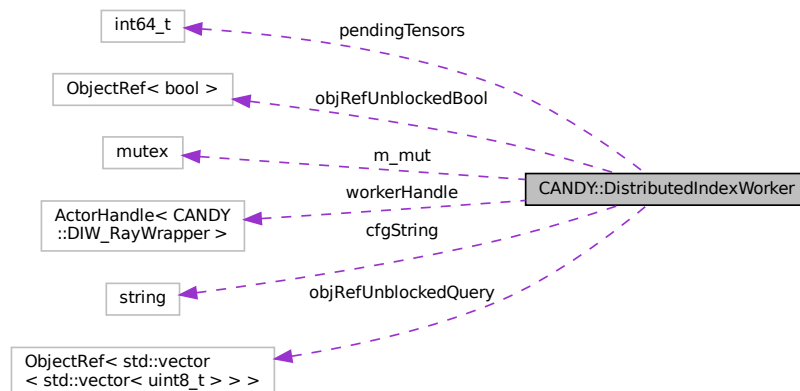
- include/CANDY/HNSWNaive/DistanceQueryer.h

8.77 CANDY::DistributedIndexWorker Class Reference

A worker class of parallel index thread.

```
#include <CANDY/DistributedPartitionIndex/DistributedIndexWorker.h>
```

Collaboration diagram for CANDY::DistributedIndexWorker:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor

- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual void [searchTensorUnblock](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, without blocking the reset process
- virtual std::vector< torch::Tensor > [getUnblockQueryResult](#) (void)
search the k-NN of a query tensor, return the result tensors
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (torch::Tensor &t)
offline build phase
- virtual void [offlineBuildUnblocked](#) (torch::Tensor &t)
offline build phase in unblocked model
- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [loadInitialTensorUnblocked](#) (torch::Tensor &t)
load initial tensor in unblocked model
- virtual bool [waitPendingOperations](#) ()
a busy waitting for all pending operations to be done
- bool [waitPendingBool](#) (void)
wait for the pending bool results, which are previously launched by unblocked manner

Protected Member Functions

- void [lock](#) ()
lock this worker
- void [unlock](#) ()
unlock this worker

Protected Attributes

- ray::ActorHandle< [DIW_RayWrapper](#) > **workerHandle**
- std::string **cfgString**
- std::mutex **m_mut**
- ray::ObjectRef< std::vector< std::vector< uint8_t > > > **objRefUnblockedQuery**
- ray::ObjectRef< bool > **objRefUnblockedBool**
- int64_t **pendingTensors** = 0

8.77.1 Detailed Description

A worker class of parallel index thread.

Note

special parameters

- parallelWorker_algoTag The algo tag of this worker, String, default flat
- parallelWorker_queueSize The input queue size of this worker, l64, default 10

8.77.2 Member Function Documentation

8.77.2.1 deleteTensor()

```
bool CANDY::DistributedIndexWorker::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

8.77.2.2 endHPC()

```
bool CANDY::DistributedIndexWorker::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

8.77.2.3 getUnblockQueryResult()

```
std::vector< torch::Tensor > CANDY::DistributedIndexWorker::getUnblockQueryResult (
    void ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>q</i>	the tensor, packed in std::vector<uint8_t> allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<std::vector<uint8_t>>` the packed result tensor for each row of query

8.77.2.4 insertTensor()

```
bool CANDY::DistributedIndexWorker::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<code>t</code>	the tensor, some index need to be single row
----------------	--

Returns

bool whether the insertion is successful

8.77.2.5 loadInitialTensor()

```
bool CANDY::DistributedIndexWorker::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<code>t</code>	the tensor, some index need to be single row
----------------	--

Returns

bool whether the loading is successful

8.77.2.6 loadInitialTensorUnblocked()

```
void CANDY::DistributedIndexWorker::loadInitialTensorUnblocked (
    torch::Tensor & t ) [virtual]
```

load initial tensor in unblocked model

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

8.77.2.7 offlineBuild()

```
bool CANDY::DistributedIndexWorker::offlineBuild (  
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

8.77.2.8 offlineBuildUnblocked()

```
void CANDY::DistributedIndexWorker::offlineBuildUnblocked (  
    torch::Tensor & t ) [virtual]
```

offline build phase in unblocked model

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

8.77.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::DistributedIndexWorker::searchTensor (  
    torch::Tensor & q,  
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

8.77.2.10 searchTensorUnblock()

```
void CANDY::DistributedIndexWorker::searchTensorUnblock (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, without blocking the reset process

Parameters

<i>q</i>	the tensor, packed in <code>std::vector<uint8_t></code> allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<std::vector<uint8_t>>` the packed result tensor for each row of query

8.77.2.11 setConfig()

```
bool CANDY::DistributedIndexWorker::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

`bool` whether the configuration is successful

8.77.2.12 setFrozenLevel()

```
bool CANDY::DistributedIndexWorker::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

8.77.2.13 startHPC()

```
bool CANDY::DistributedIndexWorker::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

8.77.2.14 waitPendingOperations()

```
bool CANDY::DistributedIndexWorker::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Note

in this index, there are may be some un-commited write due to the parallel queues

Returns

bool, whether the waitting is actually done;

The documentation for this class was generated from the following files:

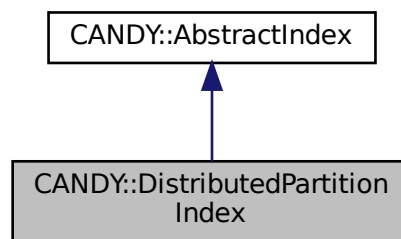
- include/CANDY/DistributedPartitionIndex/[DistributedIndexWorker.h](#)
- src/CANDY/DistributedPartitionIndex/[DistributedIndexWorker.cpp](#)

8.78 CANDY::DistributedPartitionIndex Class Reference

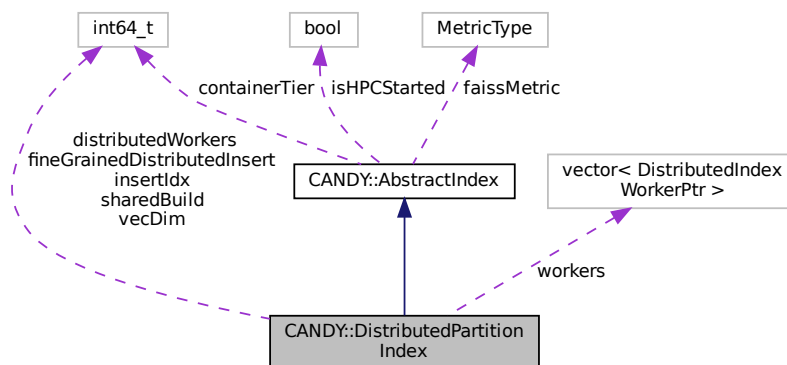
A basic distributed index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query.

```
#include <CANDY/DistributedPartitionIndex.h>
```

Inheritance diagram for CANDY::DistributedPartitionIndex:



Collaboration diagram for CANDY::DistributedPartitionIndex:



Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)

- insert a tensor*
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
- delete a tensor*
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
- revise a tensor*
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
- return a vector of tensors according to some index*
- virtual torch::Tensor [rawData](#) ()
- return the rawData of tensor*
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
- search the k-NN of a query tensor, return the result tensors*
- virtual bool [startHPC](#) ()
- some extra set-ups if the index has HPC fetures*
- virtual bool [endHPC](#) ()
- some extra termination if the index has HPC fetures*
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
- set the frozen level of online updating internal state*
- virtual bool [offlineBuild](#) (torch::Tensor &t)
- offline build phase*
- virtual bool [waitPendingOperations](#) ()
- a busy waiting for all pending operations to be done*

Protected Member Functions

- void [insertTensorInline](#) (torch::Tensor t)
- void [partitionBuildInLine](#) (torch::Tensor &t)
- void [partitionLoadInLine](#) (torch::Tensor &t)

Protected Attributes

- int64_t [distributedWorkers](#)
- int64_t [insertIdx](#)
- std::vector< [DistributedIndexWorkerPtr](#) > [workers](#)
- int64_t [vecDim](#)
- int64_t [fineGrainedDistributedInsert](#)
- int64_t [sharedBuild](#)

Additional Inherited Members

8.78.1 Detailed Description

A basic distributed index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query.

Todo consider an unblocked, optimized version of [insertTensor](#), as we did in [loadInitialTensor](#) ?

Note

special parameters

- `distributedWorker_algoTag` The algo tag of this worker, String, default flat
- `distributedWorker_queueSize` The input queue size of this worker, l64, default 10
- `distributedWorkers` The number of paraller workers, l64, default 1;
- `vecDim`, the dimension of vectors, default 768, l64
- `fineGrainedDistributedInsert`, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, l64, default 0
- `sharedBuild` whether let all sharding using shared build, 1, l64

Warning

Make sure you are using 2D tensors! Not works well with python API

8.78.2 Member Function Documentation**8.78.2.1 deleteTensor()**

```
bool CANDY::DistributedPartitionIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

1. map
2. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.2 endHPC()

```
bool CANDY::DistributedPartitionIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.3 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::DistributedPartitionIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.4 insertTensor()

```
bool CANDY::DistributedPartitionIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.5 loadInitialTensor()

```
bool CANDY::DistributedPartitionIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.6 offlineBuild()

```
bool CANDY::DistributedPartitionIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.7 rawData()

```
torch::Tensor CANDY::DistributedPartitionIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.8 reviseTensor()

```
bool CANDY::DistributedPartitionIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

only allow to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::DistributedPartitionIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

1. map
2. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.10 setConfig()

```
bool CANDY::DistributedPartitionIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.11 setFrozenLevel()

```
bool CANDY::DistributedPartitionIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.12 startHPC()

```
bool CANDY::DistributedPartitionIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.78.2.13 waitPendingOperations()

```
bool CANDY::DistributedPartitionIndex::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Note

in this index, there are may be some un-commited write due to the parallel queues

Returns

bool, whether the waiting is actually done;

Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

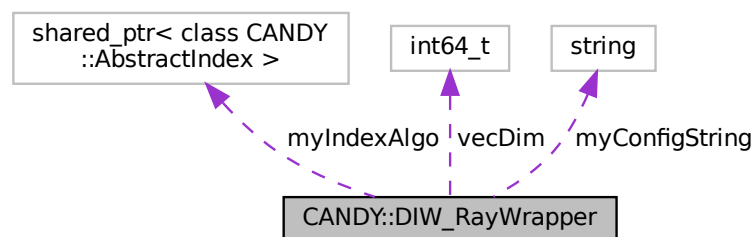
- include/CANDY/[DistributedPartitionIndex.h](#)
- src/CANDY/[DistributedPartitionIndex.cpp](#)

8.79 CANDY::DIW_RayWrapper Class Reference

the ray wrapper of [DistributedIndexWorker](#), most of its function will be ray-remote

```
#include <CANDY/DistributedPartitionIndex/DistributedIndexWorker.h>
```

Collaboration diagram for CANDY::DIW_RayWrapper:



Public Member Functions

- bool [setConfig](#) (std::string cfs)
set the config by using raw string
- virtual bool [insertTensor](#) (std::vector< uint8_t > t)
insert a tensor
- virtual bool [deleteTensor](#) (std::vector< uint8_t > t, int64_t k=1)
delete a tensor
- virtual std::vector< std::vector< uint8_t > > [searchTensor](#) (std::vector< uint8_t > t, int64_t k)
search the k-NN of a query tensor, return the result tensors
- bool [reset](#) ()
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (std::vector< uint8_t > t)
offline build phase
- virtual bool [loadInitialTensor](#) (std::vector< uint8_t > t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual bool [waitPendingOperations](#) ()
a busy waiting for all pending operations to be done

Static Public Member Functions

- static [DIW_RayWrapper](#) * [FactoryCreate](#) ()

Protected Attributes

- [AbstractIndexPtr](#) [myIndexAlgo](#) = nullptr
- std::string [myConfigString](#) = ""
- int64_t [vecDim](#) = 0

8.79.1 Detailed Description

the ray wrapper of [DistributedIndexWorker](#), most of its function will be ray-remote

- distributedWorker_algoTag The algo tag of this worker, String, default flat
- vecDim the dimension of vectors, l674, default 768

8.79.2 Member Function Documentation

8.79.2.1 deleteTensor()

```
bool CANDY::DIW_RayWrapper::deleteTensor (
    std::vector< uint8_t > t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, packed in <code>std::vector<uint8_t></code>
<i>k</i>	the number packed in <code>std::vector<uint8_t></code>

Returns

bool whether the deleting is successful

8.79.2.2 endHPC()

```
bool CANDY::DIW_RayWrapper::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

8.79.2.3 insertTensor()

```
bool CANDY::DIW_RayWrapper::insertTensor (
    std::vector< uint8_t > t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor packed in <code>std::vector<uint8_t></code>
----------	--

Returns

bool whether the insertion is successful

8.79.2.4 loadInitialTensor()

```
bool CANDY::DIW_RayWrapper::loadInitialTensor (
    std::vector< uint8_t > t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

8.79.2.5 offlineBuild()

```
bool CANDY::DIW_RayWrapper::offlineBuild (
    std::vector< uint8_t > t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

8.79.2.6 searchTensor()

```
std::vector< std::vector< uint8_t > > CANDY::DIW_RayWrapper::searchTensor (
    std::vector< uint8_t > t,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>q</i>	the tensor, packed in std::vector<uint8_t> allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<std::vector<uint8_t>> the packed result tensor for each row of query

8.79.2.7 setConfig()

```
bool CANDY::DIW_RayWrapper::setConfig (
    std::string cfs )
```

set the config by using raw string

Parameters

<i>cfs</i>	the raw string
------------	----------------

Returns

bool

1. find the index algo

8.79.2.8 setFrozenLevel()

```
bool CANDY::DIW_RayWrapper::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

8.79.2.9 startHPC()

```
bool CANDY::DIW_RayWrapper::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

8.79.2.10 waitPendingOperations()

```
bool CANDY::DIW_RayWrapper::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Note

in this index, there are may be some un-committed write due to the parallel queues

Returns

bool, whether the waiting is actually done;

The documentation for this class was generated from the following files:

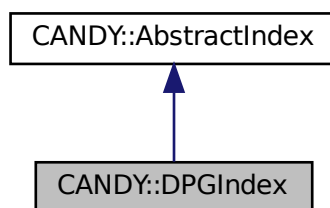
- include/CANDY/DistributedPartitionIndex/[DistributedIndexWorker.h](#)
- src/CANDY/DistributedPartitionIndex/[DistributedIndexWorker.cpp](#)

8.80 CANDY::DPGIndex Class Reference

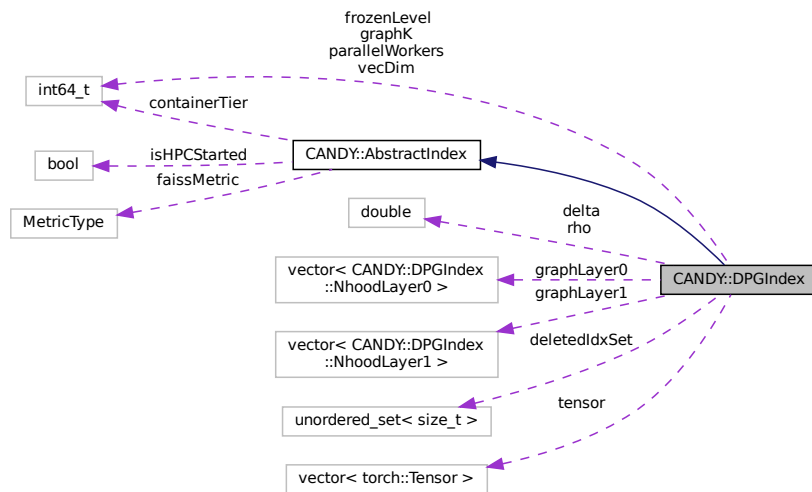
A hierarchical algorithm based on a data structure consistent with [NNDescentIndex](#), the subgraph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic graph still uses the NNDescent algorithm in this implementation.

```
#include <CANDY/DPGIndex.h>
```

Inheritance diagram for CANDY::DPGIndex:



Collaboration diagram for CANDY::DPGIndex:



Classes

- struct [Neighbor](#)
- struct [NhoodLayer0](#)
- struct [NhoodLayer1](#)

Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [endHPC](#) ()

- *some extra termination if the index has HPC fetures*
- virtual bool **setFrozenLevel** (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool **offlineBuild** (torch::Tensor &t)
offline build phase

Protected Member Functions

- void **nnDescent** ()
- void **randomSample** (std::mt19937 &rng, std::vector< size_t > &vec, size_t n, size_t sampledCount)
- bool **updateLayer0Neighbor** (size_t i, size_t j, double dist)
- void **addLayer1Neighbor** (size_t i, size_t j)
- void **removeLayer1Neighbor** (size_t i, size_t j)
- double **calcDist** (const torch::Tensor &ta, const torch::Tensor &tb)
- torch::Tensor **searchOnce** (torch::Tensor q, int64_t k)
- std::vector< std::pair< double, size_t > > **searchOnceInner** (torch::Tensor q, int64_t k)
- bool **insertOnce** (vector< std::pair< double, size_t > > &neighbors, torch::Tensor t)
- bool **deleteOnce** (torch::Tensor t, int64_t k)
- void **parallelFor** (size_t idxSize, std::function< void(size_t)> action)
- void **buildLayer1** (size_t i)

Protected Attributes

- int64_t **graphK**
- int64_t **parallelWorkers**
- int64_t **vecDim**
- int64_t **frozenLevel**
- double **rho**
- double **delta**
- std::vector< NhoodLayer0 > **graphLayer0**
- std::vector< NhoodLayer1 > **graphLayer1**
- std::vector< torch::Tensor > **tensor**
- std::unordered_set< size_t > **deletedIdxSet**

Additional Inherited Members

8.80.1 Detailed Description

A hierarchical algorithm based on a data structure consistent with [NNDescentIndex](#), the subgraph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic graph still uses the NNDescent algorithm in this implementation.

Note

special parameters

- **parallelWorkers** The number of paraller workers, l64, default 1 (set this to less than 0 will use max hardware_concurrency);
- **vecDim**, the dimension of vectors, default 768, l64
- **graphK**, the neighbors of every node in internal data struct, default 20, l64
- **rho**, sample proportion in NNDescent algorithm which takes effect in offline build only (larger is higher accuracy but lower speed), default 1.0, F64
- **delta**, loop termination condition in NNDescent algorithm which takes effect in offline build only (smaller is higher accuracy but lower speed), default 0.01, F64 @warning Make sure you are using 2D tensors!

8.80.2 Member Function Documentation

8.80.2.1 deleteTensor()

```
bool CANDY::DPGIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.2 endHPC()

```
bool CANDY::DPGIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.3 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::DPGIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.4 insertTensor()

```
bool CANDY::DPGIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.5 loadInitialTensor()

```
bool CANDY::DPGIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.6 offlineBuild()

```
bool CANDY::DPGIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.7 rawData()

```
torch::Tensor CANDY::DPGIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.8 reviseTensor()

```
bool CANDY::DPGIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::DPGIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.10 setConfig()

```
bool CANDY::DPGIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.11 setFrozenLevel()

```
bool CANDY::DPGIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.80.2.12 startHPC()

```
bool CANDY::DPGIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented from [CANDY::AbstractIndex](#).

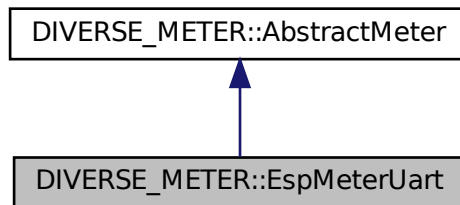
The documentation for this class was generated from the following files:

- [include/CANDY/DPGIndex.h](#)
- [src/CANDY/DPGIndex.cpp](#)

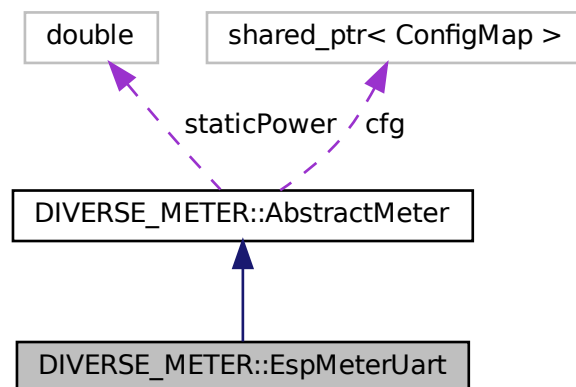
8.81 DIVERSE_METER::EspMeterUart Class Reference

the entity of an esp32s2-based power meter, connected by uart 115200

Inheritance diagram for DIVERSE_METER::EspMeterUart:



Collaboration diagram for DIVERSE_METER::EspMeterUart:



Public Member Functions

- virtual void [setConfig](#) (INTELLI::ConfigMapPtr _cfg)
to set the configmap
- void [startMeter](#) ()
to start the meter into some measuring tasks
- void [stopMeter](#) ()
to stop the meter into some measuring tasks
- double [getE](#) ()
to get the energy in J, including static energy consumption of system
- double [getPeak](#) ()
to get the peak power in W, including static power of system
- bool **isValid** ()

Additional Inherited Members

8.81.1 Detailed Description

the entity of an esp32s2-based power meter, connected by uart 115200

Note

default behaviors:

- create
- call [setConfig\(\)](#) to config this meter
- (optional) call [testStaticPower\(\)](#) to test the static power of a device, if you want to exclude it
- call [startMeter\(\)](#) to start measurement
- (run your program)
- call [stopMeter\(\)](#) to stop measurement
- call [getE\(\)](#), [getPeak\(\)](#), etc to get the measurement results

config parameters:

- meterAddress, String, The file system path of meter, default "/dev/ttyUSB0";

tag is "espUart"

8.81.2 Member Function Documentation

8.81.2.1 setConfig()

```
void EspMeterUart::setConfig (
    INTELLI::ConfigMapPtr _cfg ) [virtual]
```

to set the configmap

Parameters

<i>cfg</i>	the config map
------------	----------------

Reimplemented from [DIVERSE_METER::AbstractMeter](#).

The documentation for this class was generated from the following files:

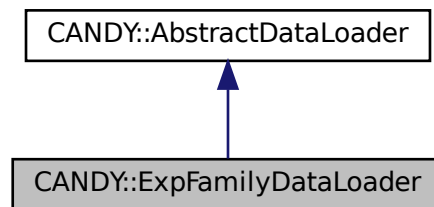
- include/Utils/Meters/EspMeterUart/[EspMeterUart.hpp](#)
- src/Utils/Meters/EspMeterUart/[EspMeterUart.cpp](#)

8.82 CANDY::ExpFamilyDataLoader Class Reference

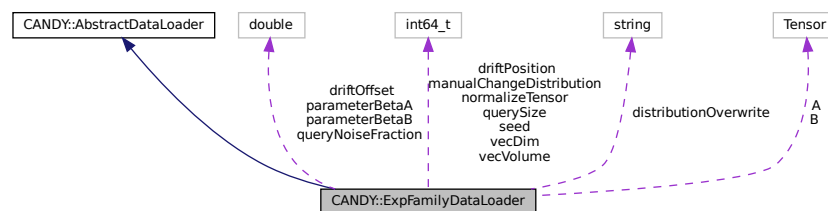
The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta.

```
#include <DataLoader/ExpFamilyDataLoader.h>
```

Inheritance diagram for CANDY::ExpFamilyDataLoader:



Collaboration diagram for CANDY::ExpFamilyDataLoader:



Public Member Functions

- virtual bool [hijackConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
To hijack some configurations inline.
- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
Set the GLOBAL config map related to this loader.
- virtual [torch::Tensor](#) [getData](#) ()
get the data tensor
- virtual [torch::Tensor](#) [getQuery](#) ()
get the query tensor

Protected Member Functions

- [torch::Tensor](#) [generateExp](#) ()
- [torch::Tensor](#) [generateGaussian](#) ()
- [torch::Tensor](#) [generateBinomial](#) ()
- [torch::Tensor](#) [generatePoisson](#) ()
- [torch::Tensor](#) [generateBeta](#) ()
- [torch::Tensor](#) [generateData](#) ()

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- int64_t **vecDim**
- int64_t **vecVolume**
- int64_t **querySize**
- int64_t **seed**
- int64_t **driftPosition**
- int64_t **manualChangeDistribution**
- std::string **distributionOverwrite**
- double **driftOffset**
- double **queryNoiseFraction**
- int64_t **normalizeTensor**
- double **parameterBetaA**
- double **parameterBetaB**

8.82.1 Detailed Description

The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

parameters of config

- vecDim, the dimension of vectors, default 768, I64
- vecVolume, the volume of vectors, default 1000, I64
- driftPosition, the position of starting some 'concept drift', default 0 (no drift), I64
- parameterBetaA, the a parameter in beta distribution, default 2.0, double
- parameterBetaB, the b parameter in beta distribution, default 2.0, double
- normalizeTensor, whether or not additionally normalize the tensors in L2, 0 (no), I64
 - driftOffset, the offset value of concept drift, default 0.5, Double
 - queryNoiseFraction, the fraction of noise in query, default 0, allow 0~1, Double
- querySize, the size of query, default 10, I64
- manualChangeDistribution, open this to manually change the distribution, default 0, I64
- distributionOverwrite, the string indicator to manually overwrite the distribution tag, default exponential, String, can be any one of
 - poisson
 - gaussian
 - exp
 - beta
- seed, the ExpFamily seed, default 7758258, I64

: default name tags "ExpFamily": [ExpFamilyDataLoader](#)

8.82.2 Member Function Documentation

8.82.2.1 `getData()`

```
torch::Tensor CANDY::ExpFamilyDataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.82.2.2 `getQuery()`

```
torch::Tensor CANDY::ExpFamilyDataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.82.2.3 `hijackConfig()`

```
bool CANDY::ExpFamilyDataLoader::hijackConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

To hijack some configurations inline.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

8.82.2.4 setConfig()

```
bool CANDY::ExpFamilyDataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

The documentation for this class was generated from the following files:

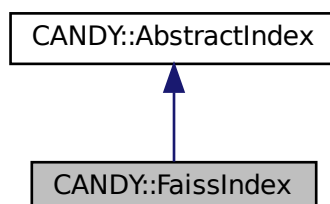
- include/DataLoader/[ExpFamilyDataLoader.h](#)
- src/DataLoader/ExpFamilyDataLoader.cpp

8.83 CANDY::FaissIndex Class Reference

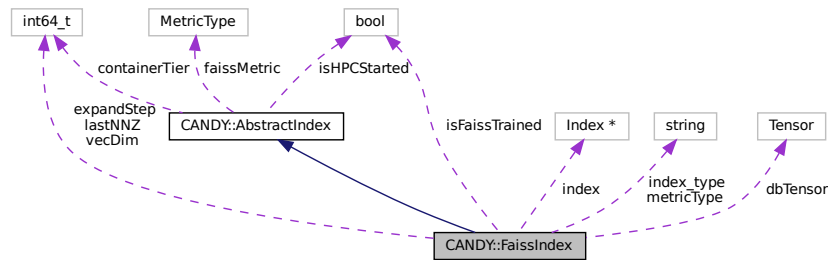
The class of converting faiss index api into rania index style.

```
#include <CANDY/FaissIndex.h>
```

Inheritance diagram for CANDY::FaissIndex:



Collaboration diagram for CANDY::FaissIndex:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
set the index-specific config related to one index
- virtual bool [loadInitialTensor](#) ([torch::Tensor](#) &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual bool [insertTensor](#) ([torch::Tensor](#) &t)
insert a tensor
- virtual [std::vector< faiss::idx_t >](#) [searchIndex](#) ([torch::Tensor](#) q, [int64_t](#) k)
search the k-NN of a query tensor, return their index
- virtual [std::vector< torch::Tensor >](#) [searchTensor](#) ([torch::Tensor](#) &q, [int64_t](#) k)
search the k-NN of a query tensor, return the result tensors
- virtual [std::vector< torch::Tensor >](#) [getTensorByIndex](#) ([std::vector< faiss::idx_t >](#) &idx, [int64_t](#) k)
return a vector of tensors according to some index

Protected Types

- typedef [std::string](#) **index_type_t**
- typedef [std::string](#) **metric_type_t**

Protected Attributes

- bool **isFaissTrained** = false
- [faiss::Index *](#) **index** = nullptr
- [index_type_t](#) **index_type**
- [metric_type_t](#) **metricType**
- [int64_t](#) **vecDim**
- [torch::Tensor](#) **dbTensor**
- [int64_t](#) **lastNNZ**
- [int64_t](#) **expandStep**

Additional Inherited Members

8.83.1 Detailed Description

The class of converting faiss index api into rania index style.

Note

currently single thread

Todo more explanation on IVFPQ, NNDecent, LSH, NSG

Note

config parameters

- vecDim, the dimension of vectors, default 768, l64
- faissIndexTag, the internal tag of loading faiss index approaches, String can be either one of the following
 - flat (default), using faiss::IndexFlat
 - [HNSW](#), using faiss::IndexHNSWFlat, additional config as follows
 - * maxConnection, l64, default 32, the max number of neighbor connections in hns
 - PQ, using faiss::IndexPQ, additional config as follows
 - * encodeLen, the encoding length in bytes, l64, default 1
 - * encodeLenBits, the encoding length in bits, l64, default encodeLen*8 (will overwrite encodeLen if manually set)
 - * subQuantizers, the number of subquantizers used, l64, default 8
 - IVFPQ, using faiss::IndexIVFPQ, additional config as follows
 - * encodeLen, the encoding length in bytes, l64, default 1
 - * encodeLenBits, the encoding length in bits, l64, default encodeLen*8 (will overwrite encodeLen if manually set)
 - * subQuantizers, the number of subquantizers used, l64, default 8
 - * lists, the number of lists used, l64, default 1000
 - LSH, using faiss::IndexLSH, additional config as follows
 - * encodeLen, the encoding length in bytes, l64, default 1
 - * encodeLenBits, the encoding length in bits, l64, default encodeLen*8 (will overwrite encodeLen if manually set)
 - NNDescent, using faiss::IndexNNDescentFlat, still some missing functions like [insertTensor](#)
 - NSG, using faiss::IndexNSGFlat, still some missing functions like [insertTensor](#)

8.83.2 Member Function Documentation

8.83.2.1 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::FaissIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.83.2.2 insertTensor()

```
bool CANDY::FaissIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.83.2.3 loadInitialTensor()

```
bool CANDY::FaissIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.83.2.4 searchIndex()

```
std::vector< faiss::idx_t > CANDY::FaissIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.83.2.5 searchTensor()

```
std::vector< torch::Tensor > CANDY::FaissIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.83.2.6 setConfig()

```
bool CANDY::FaissIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

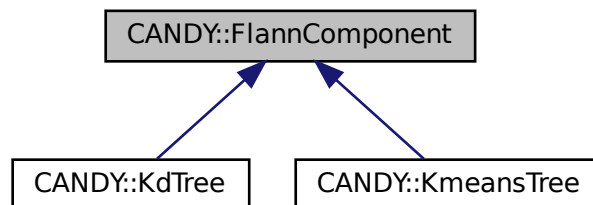
Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

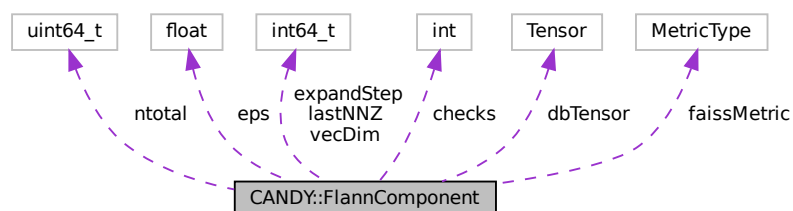
- include/CANDY/[FaissIndex.h](#)
- src/CANDY/FaissIndex.cpp

8.84 CANDY::FlannComponent Class Reference

Inheritance diagram for CANDY::FlannComponent:



Collaboration diagram for CANDY::FlannComponent:



Public Member Functions

- virtual void **addPoints** (torch::Tensor &t)
- virtual int **knnSearch** (torch::Tensor &q, int64_t *idx, float *distances, int64_t aknn)
- virtual bool **setConfig** ([INTELLI::ConfigMapPtr](#) cfg)
- virtual bool **setParams** ([FlannParam](#) param)
set the params from auto-tuning

Public Attributes

- int64_t **vecDim**
- uint64_t **ntotal**
- int **checks** = 32
- float **eps** = 0.0
- int64_t **lastNNZ**
- int64_t **expandStep**
- torch::Tensor [dbTensor](#)
Pointer dataset.
- faiss::MetricType **faissMetric** = faiss::METRIC_L2

8.84.1 Member Function Documentation

8.84.1.1 setParams()

```
virtual bool CANDY::FlannComponent::setParams (
    FlannParam param ) [inline], [virtual]
```

set the params from auto-tuning

Parameters

<i>param</i>	best param
--------------	------------

Returns

true if success

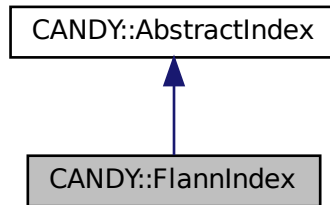
Reimplemented in [CANDY::KmeansTree](#), and [CANDY::KdTree](#).

The documentation for this class was generated from the following file:

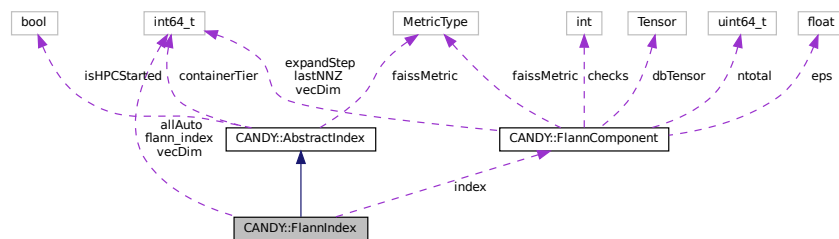
- include/CANDY/FlannIndex/FlannComponent.h

8.85 CANDY::FlannIndex Class Reference

Inheritance diagram for CANDY::FlannIndex:



Collaboration diagram for CANDY::FlannIndex:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
set the index-specific config related to one index
- virtual bool [loadInitialTensor](#) ([torch::Tensor](#) &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual bool [insertTensor](#) ([torch::Tensor](#) &t)
insert a tensor
- virtual [std::vector< faiss::idx_t >](#) [searchIndex](#) ([torch::Tensor](#) q, [int64_t](#) k)
search the k-NN of a query tensor, return their index
- virtual [std::vector< torch::Tensor >](#) [searchTensor](#) ([torch::Tensor](#) &q, [int64_t](#) k)
search the k-NN of a query tensor, return the result tensors
- virtual [std::vector< torch::Tensor >](#) [getTensorByIndex](#) ([std::vector< faiss::idx_t >](#) &idx, [int64_t](#) k)
return a vector of tensors according to some index

Public Attributes

- [flann_index_t](#) **flann_index** = `FLANN_KMEANS`
- [FlannComponent](#) * **index**
- [int64_t](#) **vecDim**
- [int64_t](#) **allAuto** = 0

Additional Inherited Members

8.85.1 Member Function Documentation

8.85.1.1 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::FlannIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.85.1.2 insertTensor()

```
bool CANDY::FlannIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.85.1.3 loadInitialTensor()

```
bool CANDY::FlannIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.85.1.4 searchIndex()

```
std::vector< faiss::idx_t > CANDY::FlannIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.85.1.5 searchTensor()

```
std::vector< torch::Tensor > CANDY::FlannIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.85.1.6 setConfig()

```
bool CANDY::FlannIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

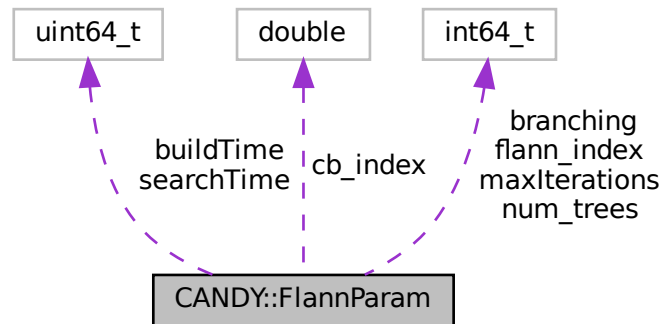
Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

- include/CANDY/FlannIndex.h
- src/CANDY/FlannIndex.cpp

8.86 CANDY::FlannParam Struct Reference

Collaboration diagram for CANDY::FlannParam:



Public Attributes

- `flann_index_t` **flann_index**
- `int64_t` **num_trees**
- `double` **cb_index**
- `int64_t` **branching**
- `int64_t` **maxIterations**
- `uint64_t` **searchTime**
- `uint64_t` **buildTime**

The documentation for this struct was generated from the following file:

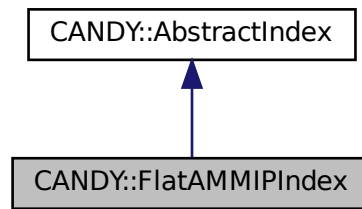
- `include/CANDY/FlannIndex/FlannComponent.h`

8.87 CANDY::FlatAMMIPIndex Class Reference

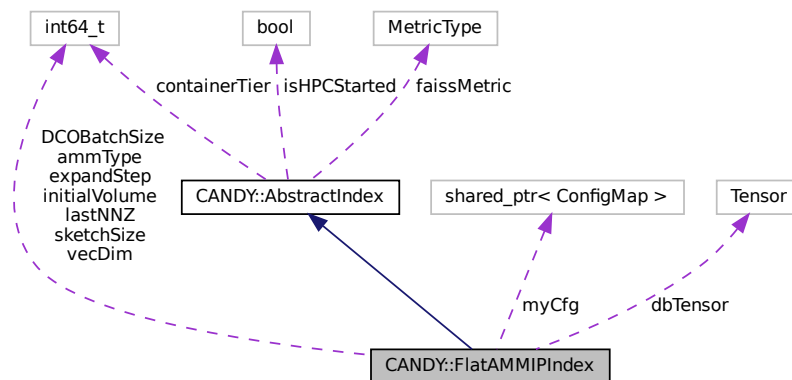
The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance.

```
#include <CANDY/FlatAMMIPIndex.h>
```

Inheritance diagram for CANDY::FlatAMMIPIndex:



Collaboration diagram for CANDY::FlatAMMIPIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > [searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)

- return a vector of tensors according to some index*
- virtual torch::Tensor `rawData` ()
return the rawData of tensor
- virtual int64_t `size` ()
return the size of ingested tensors

Protected Member Functions

- torch::Tensor `myMMInline` (torch::Tensor &a, torch::Tensor &b, int64_t ss=10)
- std::vector< faiss::idx_t > `knnInline` (torch::Tensor &query, int64_t k, int64_t distanceBatch=-1)

Protected Attributes

- `INTELLI::ConfigMapPtr myCfg` = nullptr
- torch::Tensor `dbTensor`
- int64_t `lastNNZ` = 0
- int64_t `vecDim` = 0
- int64_t `initialVolume` = 1000
- int64_t `expandStep` = 100
- int64_t `ammType` = 0
- int64_t `sketchSize` = 10
- int64_t `DCOBatchSize` = -1

Additional Inherited Members

8.87.1 Detailed Description

The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance.

Note

Only support inner product distance

currently single thread

config parameters

- `vecDim`, the dimension of vectors, default 768, l64
- `initialVolume`, the initial volume of inline database tensor, default 1000, l64
- `expandStep`, the step of expanding inline database, default 100, l64
- `sketchSize`, the sketch size of amm, default 10, l64
- `DCOBatchSize`, the batch size of internal distance comparison operation (DCO), default -1 (full data once), l64
- `ammAlgo`, the amm algorithm used for compute distance, default mm, String, can be the following
 - mm the original torch::matmul
 - crs column row sampling
 - smp-pca the smp-pca algorithm

8.87.2 Member Function Documentation

8.87.2.1 deleteTensor()

```
bool CANDY::FlatAMMIPIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.2 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::FlatAMMIPIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.3 insertTensor()

```
bool CANDY::FlatAMMIPIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.4 rawData()

```
torch::Tensor CANDY::FlatAMMIPIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.5 reviseTensor()

```
bool CANDY::FlatAMMIPIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.6 searchIndex()

```
std::vector< faiss::idx_t > CANDY::FlatAMMIPIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.7 searchTensor()

```
std::vector< torch::Tensor > CANDY::FlatAMMIPIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.8 setConfig()

```
bool CANDY::FlatAMMIPIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.87.2.9 size()

```
virtual int64_t CANDY::FlatAMMIPIndex::size ( ) [inline], [virtual]
```

return the size of ingested tensors

Returns

The documentation for this class was generated from the following files:

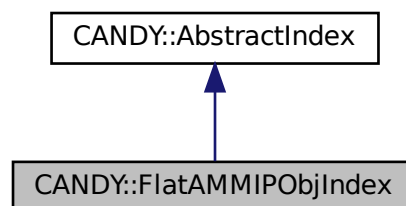
- include/CANDY/[FlatAMMIPIndex.h](#)
- src/CANDY/[FlatAMMIPIndex.cpp](#)

8.88 CANDY::FlatAMMIPObjIndex Class Reference

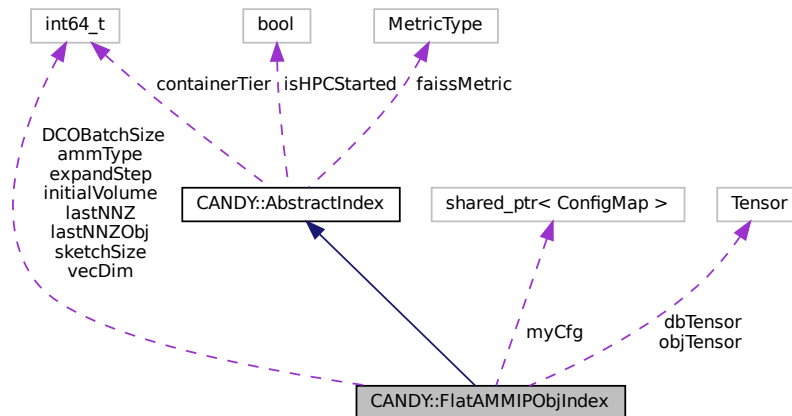
Similar to [FlatAMMIPIndex](#), but additionally has object storage (currently only string)

```
#include <CANDY/FlatAMMIPObjIndex.h>
```

Inheritance diagram for CANDY::FlatAMMIPObjIndex:



Collaboration diagram for CANDY::FlatAMMIObjIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > [searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual int64_t [size](#) ()
return the size of ingested tensors
- virtual bool [insertStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
insert a string object
- virtual bool [deleteStringObject](#) (torch::Tensor &t, int64_t k=1)
delete tensor along with its corresponding string object
- virtual std::vector< std::vector< std::string > > [searchStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects

Protected Member Functions

- torch::Tensor **myMMInline** (torch::Tensor &a, torch::Tensor &b, int64_t ss=10)
- std::vector< faiss::idx_t > **knnInline** (torch::Tensor &query, int64_t k, int64_t distanceBatch=-1)

Protected Attributes

- [INTELLI::ConfigMapPtr](#) **myCfg** = nullptr
- torch::Tensor **dbTensor**
- torch::Tensor **objTensor**
- int64_t **lastNNZ** = 0
- int64_t **lastNNZObj** = 0
- int64_t **vecDim** = 0
- int64_t **initialVolume** = 1000
- int64_t **expandStep** = 100
- int64_t **ammType** = 0
- int64_t **sketchSize** = 10
- int64_t **DCOBatchSize** = -1

Additional Inherited Members

8.88.1 Detailed Description

Similar to [FlatAMMIPIndex](#), but additionally has object storage (currently only string)

Note

Only support inner product distance
currently single thread
config parameters

- vecDim, the dimension of vectors, default 768, l64
- initialVolume, the initial volume of inline database tensor, default 1000, l64
- expandStep, the step of expanding inline database, default 100, l64
- sketchSize, the sketch size of amm, default 10, l64
- DCOBatchSize, the batch size of internal distance comparison operation (DCO), default -1 (full data once), l64
- ammAlgo, the amm algorithm used for compute distance, default mm, String, can be the following
 - mm the original torch::matmul
 - crs column row sampling
 - smp-pca the smp-pca algorithm

8.88.2 Member Function Documentation

8.88.2.1 deleteStringObject()

```
bool CANDY::FlatAMMIPObjIndex::deleteStringObject (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete tensor along with its corresponding string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the delet is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.2 deleteTensor()

```
bool CANDY::FlatAMMIPObjIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.3 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::FlatAMMIPObjIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.4 insertStringObject()

```
bool CANDY::FlatAMMIPObjIndex::insertStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

insert a string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>strs</i>	the corresponding list of strings

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.5 insertTensor()

```
bool CANDY::FlatAMMIPObjIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.6 rawData()

```
torch::Tensor CANDY::FlatAMMIPObjIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.7 reviseTensor()

```
bool CANDY::FlatAMMIPObjIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.8 searchIndex()

```
std::vector< faiss::idx_t > CANDY::FlatAMMIPObjIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.9 searchStringObject()

```
std::vector< std::vector< std::string > > CANDY::FlatAMMIPObjIndex::searchStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<std::vector<std::string>> the result object for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.10 searchTensor()

```
std::vector< torch::Tensor > CANDY::FlatAMMIPObjIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.11 setConfig()

```
bool CANDY::FlatAMMIPObjIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.88.2.12 size()

```
virtual int64_t CANDY::FlatAMMIPObjIndex::size ( ) [inline], [virtual]
```

return the size of ingested tensors

Returns

The documentation for this class was generated from the following files:

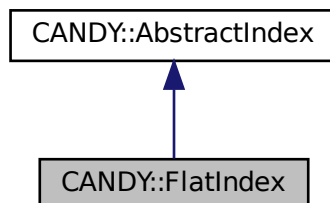
- include/CANDY/[FlatAMMIPObjIndex.h](#)
- src/CANDY/[FlatAMMIPObjIndex.cpp](#)

8.89 CANDY::FlatIndex Class Reference

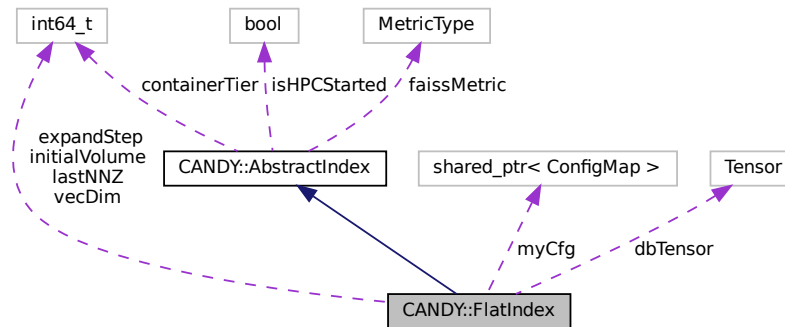
The class of a flat index approach, using brutal force management.

```
#include <CANDY/FlatIndex.h>
```

Inheritance diagram for CANDY::FlatIndex:



Collaboration diagram for CANDY::FlatIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > [searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual int64_t [size](#) ()
return the size of ingested tensors

Protected Attributes

- INTELLI::ConfigMapPtr **myCfg** = nullptr
- torch::Tensor **dbTensor**
- int64_t **lastNNZ** = 0
- int64_t **vecDim** = 0
- int64_t **initialVolume** = 1000
- int64_t **expandStep** = 100

Additional Inherited Members

8.89.1 Detailed Description

The class of a flat index approach, using brutal force management.

Note

currently single thread

config parameters

- vecDim, the dimension of vectors, default 768, l64
- initialVolume, the initial volume of inline database tensor, default 1000, l64
- expandStep, the step of expanding inline database, default 100, l64

8.89.2 Member Function Documentation

8.89.2.1 deleteTensor()

```
bool CANDY::FlatIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.2 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::FlatIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.3 insertTensor()

```
bool CANDY::FlatIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.4 rawData()

```
torch::Tensor CANDY::FlatIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.5 reviseTensor()

```
bool CANDY::FlatIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.6 searchIndex()

```
std::vector< faiss::idx_t > CANDY::FlatIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.7 searchTensor()

```
std::vector< torch::Tensor > CANDY::FlatIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.8 setConfig()

```
bool CANDY::FlatIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.89.2.9 size()

```
virtual int64_t CANDY::FlatIndex::size ( ) [inline], [virtual]
```

return the size of ingested tensors

Returns

The documentation for this class was generated from the following files:

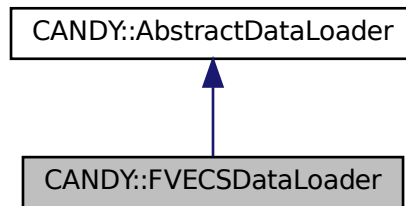
- include/CANDY/[FlatIndex.h](#)
- src/CANDY/[FlatIndex.cpp](#)

8.90 CANDY::FVECSDataLoader Class Reference

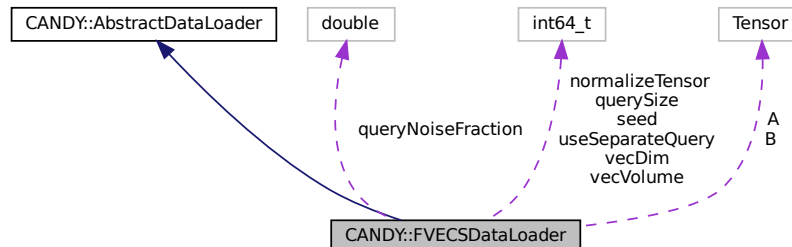
The class for loading *.fvecs data.

```
#include <DataLoader/FVECSDataLoader.h>
```

Inheritance diagram for CANDY::FVECSDataLoader:



Collaboration diagram for CANDY::FVECSDataLoader:



Public Member Functions

- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
Set the GLOBAL config map related to this loader.
- virtual torch::Tensor [getData](#) ()
get the data tensor
- virtual torch::Tensor [getQuery](#) ()
get the query tensor

Static Public Member Functions

- static torch::Tensor [tensorFromFVECS](#) (std::string fname)
the inline function to load tensor from fvecs file

Protected Member Functions

- bool **generateData** (std::string fname)
- bool **generateQuery** (std::string fname)

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- int64_t **vecDim**
- int64_t **vecVolume**
- int64_t **querySize**
- int64_t **seed**
- int64_t **normalizeTensor**
- double **queryNoiseFraction**
- int64_t **useSeparateQuery**

8.90.1 Detailed Description

The class for loading *.fvecs data.

*.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

parameters of config

- vecDim, the dimension of vectors, default 128, l64
- vecVolume, the volume of vectors, default 10000, l64
- dataPath, the path to the data file, datasets/fvecs/sift10K/siftsmall_base.fvecs, String
- normalizeTensor, whether or not normalize the tensors in L2, 1 (yes), l64
- useSeparateQuery, whether or not load query separately, 1, l64
- queryPath, the path to query file, datasets/fvecs/sift10K/siftsmall_query.fvecs. String
- queryNoiseFraction, the fraction of noise in query, default 0, allow 0~1, Double
 - no effect when query is loaded from separate file
- querySize, the size of query, default 10, l64
- seed, the random seed, default 7758258, l64

: default name tags

- "fvecs": [FVECSDataLoader](#)

8.90.2 Member Function Documentation

8.90.2.1 `getData()`

```
torch::Tensor CANDY::FVECSDataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.90.2.2 `getQuery()`

```
torch::Tensor CANDY::FVECSDataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.90.2.3 `setConfig()`

```
bool CANDY::FVECSDataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

8.90.2.4 tensorFromFVECS()

```
torch::Tensor CANDY::FVECSDataLoader::tensorFromFVECS (
    std::string fname ) [static]
```

the inline function to load tensor from fvecs file

Parameters

<i>fname</i>	the name of file
--------------	------------------

Returns

the generated tensor

The documentation for this class was generated from the following files:

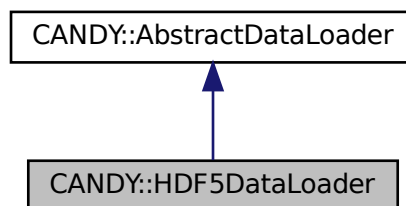
- include/DataLoader/[FVECSDataLoader.h](#)
- src/DataLoader/FVECSDataLoader.cpp

8.91 CANDY::HDF5DataLoader Class Reference

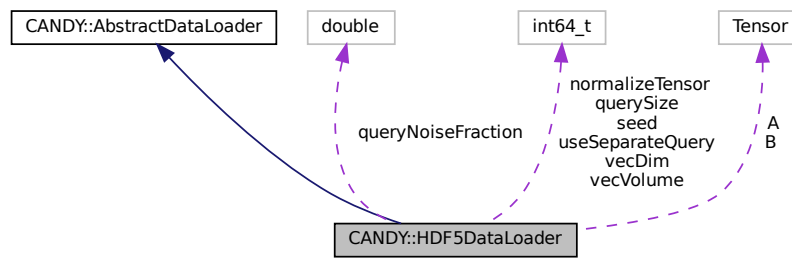
The class for loading *.hdf5 or *.h5 file, as specified in <https://github.com/HDFGroup/hdf5>.

```
#include <DataLoader/HDF5DataLoader.h>
```

Inheritance diagram for CANDY::HDF5DataLoader:



Collaboration diagram for CANDY::HDF5DataLoader:



Public Member Functions

- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
Set the GLOBAL config map related to this loader.
- virtual torch::Tensor [getData](#) ()
get the data tensor
- virtual torch::Tensor [getQuery](#) ()
get the query tensor

Static Public Member Functions

- static torch::Tensor [tensorFromHDF5](#) (std::string fname, std::string attr)
*the inline function to load tensor from *.h5 or *.hdf5 file*

Protected Member Functions

- bool **generateData** (std::string fname)
- bool **generateQuery** (std::string fname)

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- int64_t **vecDim**
- int64_t **vecVolume**
- int64_t **querySize**
- int64_t **seed**
- int64_t **normalizeTensor**
- double **queryNoiseFraction**
- int64_t **useSeparateQuery**

8.91.1 Detailed Description

The class for loading *.hdf5 or *.h5 file, as specified in <https://github.com/HDFGroup/hdf5>.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

parameters of config

- vecDim, the dimension of vectors, default 512 (for sun dataset), l64
- vecVolume, the volume of vectors, default 10000, l64
- normalizeTensor, whether or not normalize the tensors in L2, 1 (yes), l64
- dataPath, the path to the data file, datasets/hdf5/sun/sun.hdf5, String
- useSeparateQuery, whether or not load query separately, 1, l64
- queryNoiseFraction, the fraction of noise in query, default 0, allow 0~1, Double
 - no effect when query is loaded from separate file
- querySize, the size of query, default 10, l64
- seed, the random seed, default 7758258, l64

: default name tags

- hdf5: [HDF5DataLoader](#)

8.91.2 Member Function Documentation

8.91.2.1 getData()

```
torch::Tensor CANDY::HDF5DataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.91.2.2 `getQuery()`

```
torch::Tensor CANDY::HDF5DataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.91.2.3 `setConfig()`

```
bool CANDY::HDF5DataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

8.91.2.4 `tensorFromHDF5()`

```
torch::Tensor CANDY::HDF5DataLoader::tensorFromHDF5 (
    std::string fname,
    std::string attr ) [static]
```

the inline function to load tensor from *.h5 or *.hdf5 file

Parameters

<i>fname</i>	the name of file
<i>attr</i>	the attribute in hdf5 file

Returns

the generated tensor

The documentation for this class was generated from the following files:

- include/DataLoader/HDF5DataLoader.h
- src/DataLoader/HDF5DataLoader.cpp

8.92 CANDY::FLANN::Heap< T > Class Template Reference

heap structure used by [FlannIndex](#)

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

Public Member Functions

- **Heap** (int64_t size)
- int64_t **size** ()
- bool **empty** ()
- void **clear** ()
- void **insert** (const T &t)
- bool **popMin** (T &value)

8.92.1 Detailed Description

```
template<typename T>  
class CANDY::FLANN::Heap< T >
```

heap structure used by [FlannIndex](#)

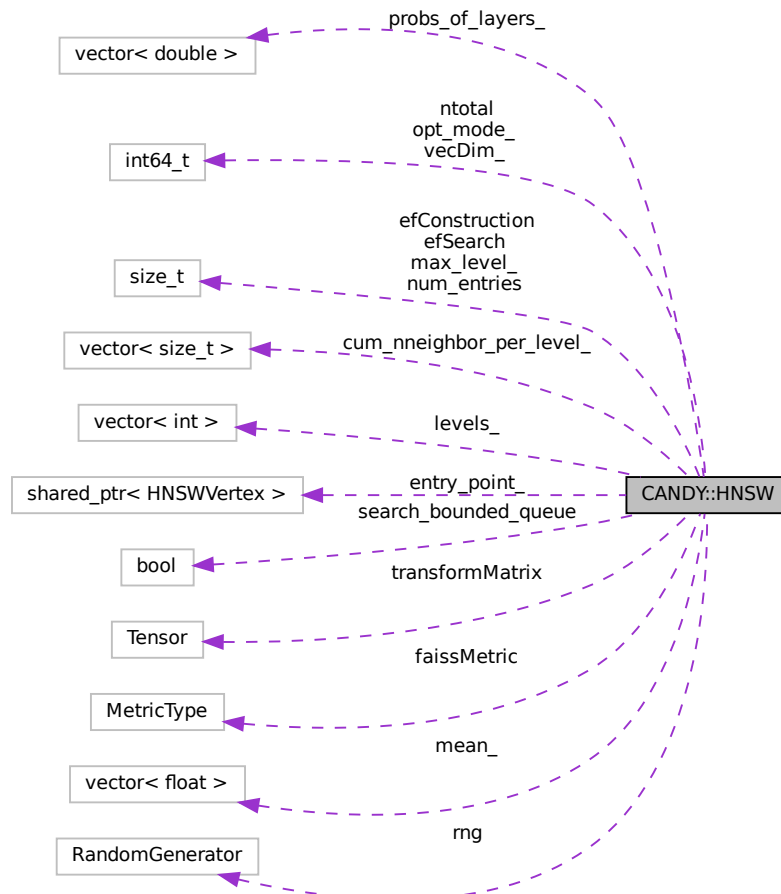
The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.93 CANDY::HNSW Class Reference

The class of a [HNSW](#) structure, maintaining parameters and a vertex entry point.

Collaboration diagram for CANDY::HNSW:



Classes

- struct [MinimaxHeap](#)
a tiny heap that is used during search
- struct [NodeDistCloser](#)
sort pairs from nearest to farthest by distance
- struct [NodeDistFarther](#)
sort pairs from farthest to nearest

Public Types

- typedef `std::pair< float, INTELLI::TensorPtr >` **Node**
- typedef `int64_t` **opt_mode_t**

Public Member Functions

- [HNSW](#) (int64_t vecDim, int64_t M)
Init [HNSW](#) structure with M neighbors.
- void [search](#) ([DistanceQueryer](#) &qdis, int k, std::vector< VertexPtr > &I, float *D, [VisitedTable](#) &vt)
search topK neighbors using qdis and store the results in I and D
- int [getLevelsByTensor](#) (torch::Tensor &t)
- int [getLevelsByPtr](#) ([INTELLI::TensorPtr](#) idx)
- void [set_nb_neighbors](#) (size_t layer_no, size_t nb)
update the number of neighbors of a layer. Not used currently
- size_t [nb_neighbors](#) (size_t layer_no)
number of neighbors for layer layer_no
- size_t [cum_nb_neighbors](#) (size_t layer_no)
cumulated number of neighbors up to layer layer_no excluded
- int [prepare_level_tab](#) (torch::Tensor &x, bool preset_levels, bool is_NSW)
assign levels to new vectors
- int [random_level](#) ()
generate a random level
- void [set_probs](#) (int64_t M, float levelMult)
set probabilities of a level to be assigned for new vectors
- void [neighbor_range](#) (int level, size_t *begin, size_t *end)
set the boundaries within neighbors_[TensorPtr] on a level
- void [add_links_starting_from](#) ([DistanceQueryer](#) &disq, VertexPtr pt_id, VertexPtr nearest, float d_nearest, int level, [VisitedTable](#) &vt)
called when add vertices. Add links to new vector according to its nearest vector's neighbor
- void [add_without_lock](#) ([DistanceQueryer](#) &disq, int assigned_level, VertexPtr pt_id, [VisitedTable](#) &vt)
add neighbors to a vertex single-threaded
- void [set_mode](#) (opt_mode_t opt_mode, faiss::MetricType metric)
- string [transform_from_tensor](#) ([INTELLI::TensorPtr](#) idx)

Public Attributes

- std::vector< int > [levels_](#)
For Tensor t, its assigned levels.
- int64_t [vecDim_](#)
- int64_t [ntotal](#)
- std::vector< size_t > [cum_nneighbor_per_level_](#)
- std::vector< double > [probs_of_layers_](#)
assigned probabilities for each layer (sum=1)
- faiss::RandomGenerator [rng](#)
- VertexPtr [entry_point_](#) = nullptr
entry point on the top level
- opt_mode_t [opt_mode_](#) = OPT_VANILLA
- faiss::MetricType [faissMetric](#) = faiss::METRIC_L2
- std::vector< float > [mean_](#)
used for LVQ encoding
- torch::Tensor [transformMatrix](#)
used for ADsampling
- size_t [max_level_](#) = -1
max level of [HNSW](#) structure
- size_t [num_entries](#) = 1

- entry_point numbers, default as 1*
- size_t [efConstruction](#) = 40
expansion factor at construction time
- size_t [efSearch](#) = 15
expansion factor during search
- bool [search_bounded_queue](#) = true

8.93.1 Detailed Description

The class of a [HNSW](#) structure, maintaining parameters and a vertex entry point.

Note

now each vertex storing each vertex's neighbors, visited number and level, with a pointer to the vector;
The [HNSW](#) structure does not store actual data of the graph except the entry point

8.93.2 Member Function Documentation

8.93.2.1 [add_links_starting_from\(\)](#)

```
void CANDY::HNSW::add_links_starting_from (
    CANDY::DistanceQueryer & disq,
    CANDY::VertexPtr pt_id,
    CANDY::VertexPtr nearest,
    float d_nearest,
    int level,
    CANDY::VisitedTable & vt )
```

called when add vertices. Add links to new vector according to its nearest vector's neighbor

Parameters

<i>disq</i>	DistanceQuery whose query is set as the new vertex to insert
<i>pt_id</i>	new vector ptr
<i>nearest</i>	greedy-searched nearest vector to new vector. Search starting from entry point
<i>d_nearest</i>	distance between nearest vector and query
<i>level</i>	assigned level
<i>vt</i>	VisitedTable

8.93.2.2 [add_without_lock\(\)](#)

```
void CANDY::HNSW::add_without_lock (
    CANDY::DistanceQueryer & disq,
```

```
int assigned_level,  
CANDY::VertexPtr pt_id,  
CANDY::VisitedTable & vt )
```

add neighbors to a vertex single-threaded

Parameters

<i>qdis</i>	distance queryer init with the query
<i>assigned_level</i>	the query's assigned level from which to add links
<i>pt_id</i>	query's vertex pointer
<i>vt</i>	visited table

8.93.2.3 cum_nb_neighbors()

```
size_t CANDY::HNSW::cum_nb_neighbors (   
    size_t layer_no )
```

cumulated number of neighbors up to layer layer_no excluded

Parameters

<i>layer_no</i>	layer number
-----------------	--------------

Returns

number of neighbors for layer layer_no

8.93.2.4 nb_neighbors()

```
size_t CANDY::HNSW::nb_neighbors (   
    size_t layer_no )
```

number of neighbors for layer layer_no

Parameters

<i>layer_no</i>	layer number
-----------------	--------------

Returns

number of neighbors for layer layer_no

8.93.2.5 neighbor_range()

```
void CANDY::HNSW::neighbor_range (
    int level,
    size_t * begin,
    size_t * end )
```

set the boundaries within neighbors_[TensorPtr] on a level

Parameters

<i>level</i>	level to be searched
<i>begin</i>	begin index
<i>end</i>	end index

8.93.2.6 prepare_level_tab()

```
int CANDY::HNSW::prepare_level_tab (
    torch::Tensor & x,
    bool preset_levels,
    bool is_NSW )
```

assign levels to new vectors

Parameters

<i>x</i>	new vectors to be assigned
<i>preset_levels</i>	if levels have been init for new vectors
<i>is_NSW</i>	if this is an NSW structure rather than HNSW

Returns

max_level assigned for new vectors

8.93.2.7 random_level()

```
int CANDY::HNSW::random_level ( )
```

generate a random level

Returns

random level

8.93.2.8 search()

```
void CANDY::HNSW::search (
    DistanceQueryer & qdis,
    int k,
    std::vector< VertexPtr > & I,
    float * D,
    VisitedTable & vt )
```

search topK neighbors using qdis and store the results in I and D

Parameters

<i>qdis</i>	distance queryer init with the query to be searched
<i>k</i>	top K neighbors
<i>I</i>	results for vectors
<i>D</i>	results for distances
<i>vt</i>	visited table

8.93.2.9 set_nb_neighbors()

```
void CANDY::HNSW::set_nb_neighbors (
    size_t layer_no,
    size_t nb )
```

update the number of neighbors of a layer. Not used currently

Parameters

<i>layer_no</i>	layer to update
<i>nb</i>	neighbor number to update to

8.93.2.10 set_probs()

```
void CANDY::HNSW::set_probs (
    int64_t M,
    float levelMult )
```

set probabilities of a level to be assigned for new vectors

Parameters

<i>M</i>	number of neighbors
<i>levelMult</i>	1/log(M) to distribute the probability

8.93.3 Member Data Documentation

8.93.3.1 cum_nneighbor_per_level_

```
std::vector<size_t> CANDY::HNSW::cum_nneighbor_per_level_
```

cumulative number of neighbors stored per layer with that layer excluded, should remain intact! cum_nneighbor_↔ per_level_[0] = 0;

8.93.3.2 search_bounded_queue

```
bool CANDY::HNSW::search_bounded_queue = true
```

whether the search process is bounded; now only bounded search is implemented

The documentation for this class was generated from the following files:

- include/CANDY/HNSWNaive/HNSW.h
- src/CANDY/HNSWNaive/HNSW.cpp

8.94 HNSWAlter Class Reference

The documentation for this class was generated from the following file:

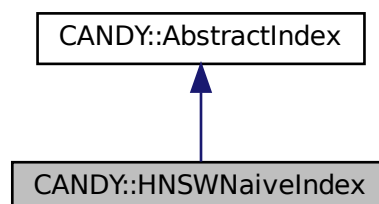
- include/CANDY/HNSWNaive/HNSWAlter.h

8.95 CANDY::HNSWNaiveIndex Class Reference

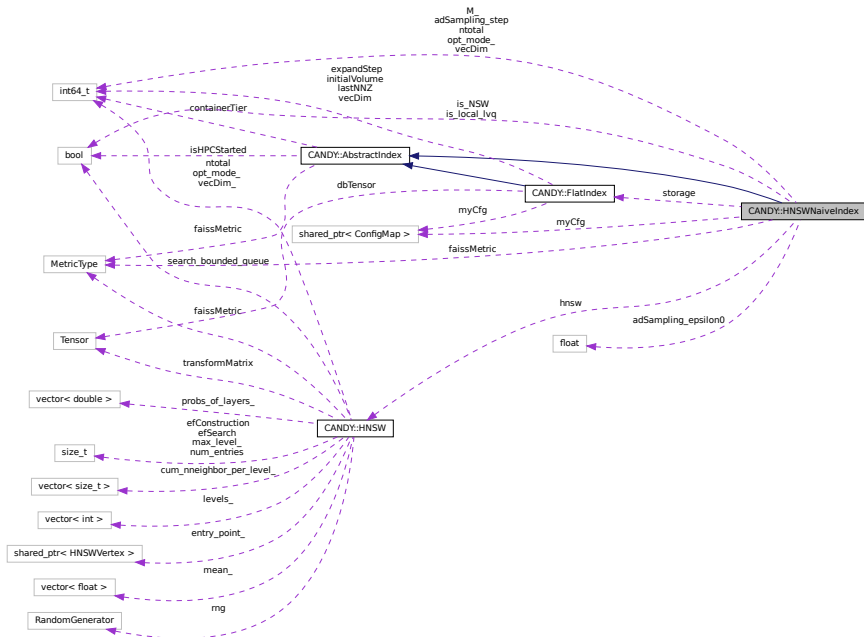
The class of a [HNSW](#) index approach, store the data in each vertex.

```
#include <CANDY/HNSWNaiveIndex.h>
```

Inheritance diagram for CANDY::HNSWNaiveIndex:



Collaboration diagram for CANDY::HNSWNaiveIndex:



Public Types

- typedef int64_t **opt_mode_t**

Public Member Functions

- virtual bool **setConfig** (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool **insertTensor** (torch::Tensor &t)
insert a tensor
- virtual bool **deleteTensor** (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool **reviseTensor** (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > **searchTensor** (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors

Public Attributes

- **HNSW** **hnsw**
- bool **is_NSW**
- bool **is_local_ivq** = true
- **FlatIndex** * **storage** = nullptr
- INTELLI::ConfigMapPtr **myCfg** = nullptr
- opt_mode_t **opt_mode_** = OPT_VANILLA
- faiss::MetricType **faissMetric** = faiss::METRIC_L2

- `int64_t vecDim`
- `int64_t M_ = 32`
Number of neighbors in [HNSW](#) structure.
- `int64_t ntotal = 0`
Number of all vectors.
- `int64_t adSampling_step = 32`
- `float adSampling_epsilon0 = 1.0`

Additional Inherited Members

8.95.1 Detailed Description

The class of a [HNSW](#) index approach, store the data in each vertex.

Note

currently single thread

config parameters

- `vecDim`, the dimension of vectors, default 768, l64
- `maxConnection`, number of maximum neighbor connection at each level, default 32, l64
- `is_NSW`, whether initialized as an NSW index, default 0 (init as [HNSW](#)), l64

8.95.2 Member Function Documentation

8.95.2.1 deleteTensor()

```
bool CANDY::HNSWNaiveIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.95.2.2 insertTensor()

```
bool CANDY::HNSWNaiveIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.95.2.3 reviseTensor()

```
bool CANDY::HNSWNaiveIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.95.2.4 searchTensor()

```
std::vector< torch::Tensor > CANDY::HNSWNaiveIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.95.2.5 setConfig()

```
bool CANDY::HNSWNaiveIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

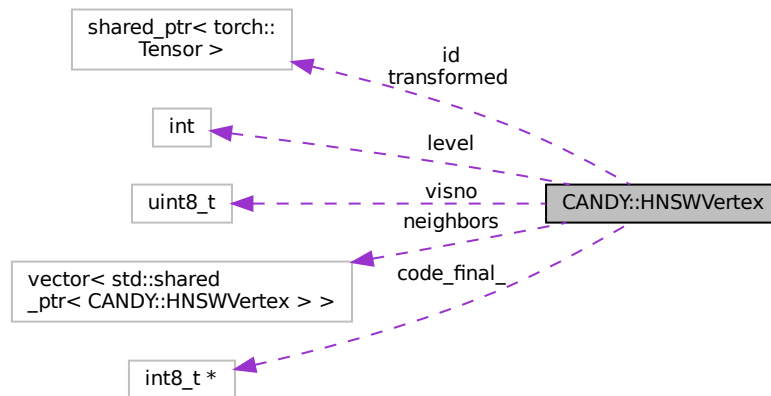
The documentation for this class was generated from the following files:

- include/CANDY/[HNSWNaiveIndex.h](#)
- src/CANDY/HNSWNaiveIndex.cpp

8.96 CANDY::HNSWVertex Class Reference

The class of a [HNSW](#) vertex, storing the data in each vertex.

Collaboration diagram for CANDY::HNSWVertex:



Public Member Functions

- **HNSWVertex** ([INTELLI::TensorPtr](#) id, int level, int num_neighbors)

Public Attributes

- [INTELLI::TensorPtr](#) **id**
- `int8_t * code_final_ = nullptr`
used for LVQ
- [INTELLI::TensorPtr](#) **transformed** = nullptr
used for adsampling
- int **level**
- `std::vector< std::shared_ptr< HNSWVertex > >` **neighbors**
- `uint8_t` **visno**

8.96.1 Detailed Description

The class of a [HNSW](#) vertex, storing the data in each vertex.

Note

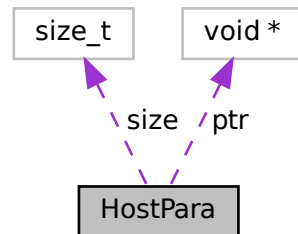
now storing each vertex's neighbors, visited number and level, with a pointer to the vector

The documentation for this class was generated from the following file:

- `include/CANDY/HNSWNaive/HNSW.h`

8.97 HostPara Class Reference

Collaboration diagram for HostPara:



Public Member Functions

- **HostPara** (void *tptr, size_t tsize)

Public Attributes

- void * **ptr**
- size_t **size**

The documentation for this class was generated from the following file:

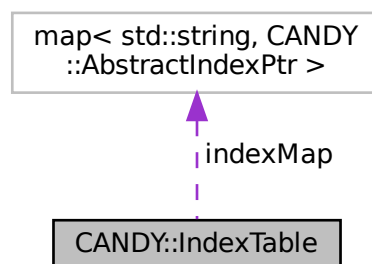
- include/CL/CLContainer.hpp

8.98 CANDY::IndexTable Class Reference

The table to index index algos.

```
#include <CANDY/IndexTable.h>
```

Collaboration diagram for CANDY::IndexTable:



Public Member Functions

- void [addIndex](#) ([CANDY::AbstractIndexPtr](#) anew, std::string tag)
To register a new ALGO.
- [CANDY::AbstractIndexPtr](#) [getIndex](#) (std::string name)
find a dataloader in the table according to its name

Protected Attributes

- std::map< std::string, [CANDY::AbstractIndexPtr](#) > **indexMap**

8.98.1 Detailed Description

The table to index index algos.

ingroup CANDY_lib

Note

Default behavior

- create
- (optional) call [addIndex](#) for new algo
- find a loader by [getIndex](#) using its tag

default tags (String)

- flat [FlatIndex](#)
- parallelPartition [ParallelPartitionIndex](#)
- onlinePQ [OnlinePQIndex](#)
- onlineIVFLSH [OnlineIVFLSHIndex](#)
- HNSWNaive [HNSWNaiveIndex](#)
- faiss [FaissIndex](#)
- congestionDrop [CongestionDropIndex](#)
- bufferedCongestionDrop [BufferedCongestionDropIndex](#)
- flatAMMIP [FlatAMMIPIndex](#)

8.98.2 Member Function Documentation

8.98.2.1 addIndex()

```
void CANDY::IndexTable::addIndex (
    CANDY::AbstractIndexPtr anew,
    std::string tag ) [inline]
```

To register a new ALGO.

Parameters

<i>anew</i>	The new algo
<i>tag</i>	The name tag

8.98.2.2 getIndex()

```
CANDY::AbstractIndexPtr CANDY::IndexTable::getIndex (
    std::string name ) [inline]
```

find a dataloader in the table according to its name

Parameters

<i>name</i>	The nameTag of loader
-------------	-----------------------

Returns

The AbstractIndexPtr, nullptr if not found

The documentation for this class was generated from the following files:

- include/CANDY/[IndexTable.h](#)
- src/CANDY/IndexTable.cpp

8.99 INTELLI::IntelliLog Class Reference

The log functions packed in class.

Static Public Member Functions

- static void [log](#) (std::string level, std::string_view message, std::source_location const source=std::source_location::current())
Produce a log.
- static void [setupLoggingFile](#) (string fname)
set up the logging file by its name

8.99.1 Detailed Description

The log functions packed in class.

The documentation for this class was generated from the following files:

- include/Utils/IntelliLog.h
- src/Utils/IntelliLog.cpp

8.100 INTELLI::IntelliLog_FileProtector Class Reference

The protector for concurrent log on a file.

Public Member Functions

- void [lock](#) ()
lock this protector
- void [unlock](#) ()
unlock this protector
- void [openLogFile](#) (const string &fname)
try to open a file
- void [appendLogFile](#) (const string &msg)
try to appened something to the file, if it's opened

8.100.1 Detailed Description

The protector for concurrent log on a file.

Warning

This class is preserved for internal use only!

The documentation for this class was generated from the following file:

- include/Utils/IntelliLog.h

8.101 INTELLI::IntelliTensorOP Class Reference

Static Public Member Functions

- static bool [deleteRow](#) (torch::Tensor *tensor, int64_t rowIdx)
delete a row of a tensor
- static bool [deleteRow](#) (TensorPtr tp, int64_t rowIdx)
delete a row of a tensor
- static bool [deleteRows](#) (torch::Tensor *tensor, std::vector< int64_t > &rowIdx)
delete rows of a tensor
- static bool [deleteRows](#) (TensorPtr tp, std::vector< int64_t > &rowIdx)
delete rows of a tensor
- static bool [appendRows](#) (torch::Tensor *tHead, torch::Tensor *tTail)
append rows to the head tensor
- static bool [appendRows](#) (TensorPtr tHeadP, TensorPtr tTailP)
append rows to the head tensor
- static bool [insertRows](#) (torch::Tensor *tHead, torch::Tensor *tTail, int64_t startRow)
insert rows to the head tensor
- static bool [insertRows](#) (TensorPtr tHead, TensorPtr tTail, int64_t startRow)

- insert rows to the head tensor*
- static bool [editRows](#) (torch::Tensor *tHead, torch::Tensor *tTail, int64_t startRow)
edit rows in the head tensor
- static bool [editRows](#) (TensorPtr tHead, TensorPtr tTail, int64_t startRow)
edit rows in the head tensor
- static bool [deleteRowBufferMode](#) (torch::Tensor *tensor, int64_t rowIdx, int64_t *lastNNZ)
delete a row of a tensor, shift this row with last nnz, and does not re-create the tensor
- static bool [deleteRowBufferMode](#) (TensorPtr tensor, int64_t rowIdx, int64_t *lastNNZ)
delete a row of a tensor, shift this row with last nnz, and does not re-create the tensor
- static bool [deleteRowsBufferMode](#) (torch::Tensor *tensor, std::vector< int64_t > &rowIdx, int64_t *lastNNZ)
delete rows of a tensor, shift this row with last nnz, and does not re-create the tensor
- static bool [deleteRowsBufferMode](#) (TensorPtr tensor, std::vector< int64_t > &rowIdx, int64_t *lastNNZ)
delete rows of a tensor, shift this row with last nnz, and does not re-create the tensor
- static bool [appendRowsBufferMode](#) (torch::Tensor *tHead, torch::Tensor *tTail, int64_t *lastNNZ, int64_t customExpandSize=0)
append rows to the head tensor, under the buffer mode
- static bool [appendRowsBufferMode](#) (TensorPtr tHead, TensorPtr tTail, int64_t *lastNNZ, int64_t customExpandSize=0)
append rows to the head tensor, under the buffer mode
- static std::vector< uint8_t > [tensorToFlatBin](#) (torch::Tensor *A)
convert a tensor to flat binary form, i.e., <rows> <cols> <flat data>
- static bool [tensorToFile](#) (torch::Tensor *A, std::string fname)
convert a tensor to flat binary form and stored in a file, i.e., <rows> <cols> <flat data>
- static bool [tensorFromFlatBin](#) (torch::Tensor *A, std::vector< uint8_t > &ru)
load a tensor from flat binary form, i.e., <rows> <cols> <flat data>
- static bool [tensorFromFile](#) (torch::Tensor *A, std::string fname)
load a tensor from a file of flat binary form, i.e., <rows> <cols> <flat data>
- static torch::Tensor [rowSampling](#) (torch::Tensor &a, int64_t sampledRows)
to sample some rows of an input tensor and return
- static torch::Tensor [l2Normalize](#) (torch::Tensor &a)
to normalize the tensor in each column, using l2

8.101.1 Member Function Documentation

8.101.1.1 [appendRows\(\)](#) [1/2]

```
static bool INTELLI::IntelliTensorOP::appendRows (
    TensorPtr tHeadP,
    TensorPtr tTailP ) [inline], [static]
```

append rows to the head tensor

Parameters

<i>tHead</i>	the head tensor, using shared pointer
<i>tTail</i>	the tail tensor, using shared pointer

Note

The number of columnnes must be matched

Returns

bool, whether the operation is successful

8.101.1.2 appendRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::appendRows (
    torch::Tensor * tHead,
    torch::Tensor * tTail ) [inline], [static]
```

append rows to the head tensor

Parameters

<i>tHead</i>	the head tensor, using pointer
<i>tTail</i>	the tail tensor, using poniter

Note

The number of columnnes must be matched

Returns

bool, whether the operation is successful

8.101.1.3 appendRowsBufferMode() [1/2]

```
static bool INTELLI::IntelliTensorOP::appendRowsBufferMode (
    TensorPtr tHead,
    TensorPtr tTail,
    int64_t * lastNNZ,
    int64_t customExpandSize = 0 ) [inline], [static]
```

append rows to the head tensor, under the buffer mode

Parameters

<i>tHead</i>	the head tensor, using shared pointer
<i>tTail</i>	the tail tensor, using sahred poniter
* <i>lastNNZ</i>	the original last non zero row in tHead, will be changed
<i>customExpandSize</i>	the customized expansion size of buffer,

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.101.1.4 appendRowsBufferMode() [2/2]

```
static bool INTELLI::IntelliTensorOP::appendRowsBufferMode (
    torch::Tensor * tHead,
    torch::Tensor * tTail,
    int64_t * lastNNZ,
    int64_t customExpandSize = 0 ) [inline], [static]
```

append rows to the head tensor, under the buffer mode

Parameters

<i>tHead</i>	the head tensor, using pointer
<i>tTail</i>	the tail tensor, using poniter
<i>*lastNNZ</i>	the original last non zero row in tHead, will be changed
<i>customExpandSize</i>	the customized expansion size of buffer,

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.101.1.5 deleteRow() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRow (
    TensorPtr tp,
    int64_t rowIdx ) [inline], [static]
```

delete a row of a tensor

Parameters

<i>t</i>	the tensor under shared pointer
<i>rowIdx</i>	the row to be deleted

Returns

bool, whether the operation is successful

8.101.1.6 deleteRow() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRow (
    torch::Tensor * tensor,
    int64_t rowIdx ) [inline], [static]
```

delete a row of a tensor

Parameters

<i>t</i>	the tensor pointer
<i>rowIdx</i>	the row to be deleted

Returns

bool, whether the operation is successful

8.101.1.7 deleteRowBufferMode() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRowBufferMode (
    TensorPtr tensor,
    int64_t rowIdx,
    int64_t * lastNNZ ) [inline], [static]
```

delete a row of a tensor, shift this row with last nnz, and does not re-create the tensor

Parameters

<i>tensor</i>	the tensor shared pointer
<i>rowIdx</i>	the row to be deleted
* <i>lastNNZ</i>	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.101.1.8 deleteRowBufferMode() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRowBufferMode (
    torch::Tensor * tensor,
```

```
int64_t rowIdx,
int64_t * lastNNZ ) [inline], [static]
```

delete a row of a tensor, shift this row with last nnz, and does not re-create the tensor

Parameters

<i>tensor</i>	the tensor pointer
<i>rowIdx</i>	the row to be deleted
<i>*lastNNZ</i>	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.101.1.9 deleteRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRows (
    TensorPtr tp,
    std::vector< int64_t > & rowIdx ) [inline], [static]
```

delete rows of a tensor

Parameters

<i>t</i>	the tensor under shared pointer
<i>rowIdx</i>	the rows to be deleted

Returns

bool, whether the operation is successful

8.101.1.10 deleteRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRows (
    torch::Tensor * tensor,
    std::vector< int64_t > & rowIdx ) [inline], [static]
```

delete rows of a tensor

Parameters

<i>t</i>	the tensor pointer
<i>rowIdx</i>	the rows to be deleted

Returns

bool, whether the operation is successful

8.101.1.11 deleteRowsBufferMode() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRowsBufferMode (
    TensorPtr tensor,
    std::vector< int64_t > & rowIdx,
    int64_t * lastNNZ ) [inline], [static]
```

delete rows of a tensor, shift this row with last nnz, and does not re-create the tensor

Parameters

<i>tensor</i>	the tensor shared pointer
<i>rowIdx</i>	the rows to be deleted
<i>*lastNNZ</i>	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.101.1.12 deleteRowsBufferMode() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRowsBufferMode (
    torch::Tensor * tensor,
    std::vector< int64_t > & rowIdx,
    int64_t * lastNNZ ) [inline], [static]
```

delete rows of a tensor, shift this row with last nnz, and does not re-create the tensor

Parameters

<i>tensor</i>	the tensor pointer
<i>rowIdx</i>	the rows to be deleted
<i>*lastNNZ</i>	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.101.1.13 editRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::editRows (
    TensorPtr tHead,
    TensorPtr tTail,
    int64_t startRow ) [inline], [static]
```

edit rows in the head tensor

Parameters

<i>tHead</i>	the head tensor, using shared pointer
<i>tTail</i>	the tail tensor, using shared pointer
<i>startRow,the</i>	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

8.101.1.14 editRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::editRows (
    torch::Tensor * tHead,
    torch::Tensor * tTail,
    int64_t startRow ) [inline], [static]
```

edit rows in the head tensor

Parameters

<i>tHead</i>	the head tensor, using pointer
<i>tTail</i>	the tail tensor, using pointer
<i>startRow,the</i>	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.101.1.15 insertRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::insertRows (
    TensorPtr tHead,
    TensorPtr tTail,
    int64_t startRow ) [inline], [static]
```

insert rows to the head tensor

Parameters

<i>tHead</i>	the head tensor, using shared pointer
<i>tTail</i>	the tail tensor, using shared pointer
<i>startRow, the</i>	starRow of tTail to be appeared afeter insertion

Returns

bool, whether the operation is successful

8.101.1.16 insertRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::insertRows (
    torch::Tensor * tHead,
    torch::Tensor * tTail,
    int64_t startRow ) [inline], [static]
```

insert rows to the head tensor

Parameters

<i>tHead</i>	the head tensor, using pointer
<i>tTail</i>	the tail tensor, using pointer
<i>startRow, the</i>	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.101.1.17 l2Normalize()

```
static torch::Tensor INTELLI::IntelliTensorOP::l2Normalize (
    torch::Tensor & a ) [inline], [static]
```

to normalize the tensor in each column, using l2

Parameters

<i>a</i>	the input tensor
----------	------------------

Returns

the result tensor

8.101.1.18 rowSampling()

```
static torch::Tensor INTELLI::IntelliTensorOP::rowSampling (
    torch::Tensor & a,
    int64_t sampledRows ) [inline], [static]
```

to sample some rows of an input tensor and return

Parameters

<i>a</i>	the input tensor
<i>sampledRows</i>	the number of rows to be sampled

Returns

the result tensor

8.101.1.19 tensorFromFile()

```
static bool INTELLI::IntelliTensorOP::tensorFromFile (
    torch::Tensor * A,
    std::string fname ) [inline], [static]
```

load a tensor from a file of flat binary form, i.e., <rows> <cols> <flat data>

Parameters

<i>A</i>	the tensor
<i>fname</i>	the name of file

Returns

bool, the load is successful or not

8.101.1.20 tensorFromFlatBin()

```
static bool INTELLI::IntelliTensorOP::tensorFromFlatBin (
    torch::Tensor * A,
    std::vector< uint8_t > & ru ) [inline], [static]
```

load a tensor from flat binary form, i.e., <rows> <cols> <flat data>

Parameters

<i>A</i>	the tensor
<i>ru</i>	the binart in <code>std::vector<uint8_t></code>

Returns

`bool`, the load is successful or not

8.101.1.21 tensorToFile()

```
static bool INTELLI::IntelliTensorOP::tensorToFile (
    torch::Tensor * A,
    std::string fname ) [inline], [static]
```

convert a tensor to flat binary form and stored in a file, i.e., <rows> <cols> <flat data>

Parameters

<i>A</i>	the tensor
<i>fname</i>	the name of file

Returns

`bool`, the output is successful or not

8.101.1.22 tensorToFlatBin()

```
static std::vector<uint8_t> INTELLI::IntelliTensorOP::tensorToFlatBin (
    torch::Tensor * A ) [inline], [static]
```

convert a tensor to flat binary form, i.e., <rows> <cols> <flat data>

Parameters

<i>A</i>	the tensor
----------	------------

Returns

`std::vector<uint8_t>` the binary form

The documentation for this class was generated from the following file:

- `include/Utils/IntelliTensorOP.hpp`

8.102 INTELLITensorOP Class Reference

The common tensor functions packed in class.

8.102.1 Detailed Description

The common tensor functions packed in class.

Note

Most are static functions

The documentation for this class was generated from the following file:

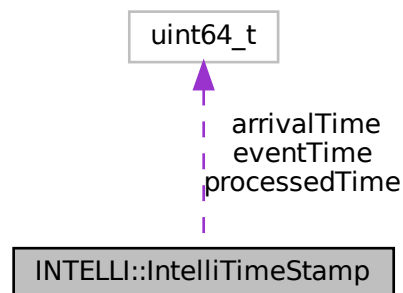
- [include/Utils/IntelliTensorOP.hpp](#)

8.103 INTELLI::IntelliTimeStamp Class Reference

The class to define a timestamp.

```
#include <Utils/IntelliTimeStampGenerator.h>
```

Collaboration diagram for INTELLI::IntelliTimeStamp:



Public Member Functions

- **IntelliTimeStamp** (uint64_t te, uint64_t ta, uint64_t tp)

Public Attributes

- uint64_t [eventTime](#) = 0
The time when the related event (to a row or a column) happen.
- uint64_t [arrivalTime](#) = 0
The time when the related event (to a row or a column) arrive to the system.
- uint64_t [processedTime](#) = 0
the time when the related event is fully processed

8.103.1 Detailed Description

The class to define a timestamp.

The documentation for this class was generated from the following file:

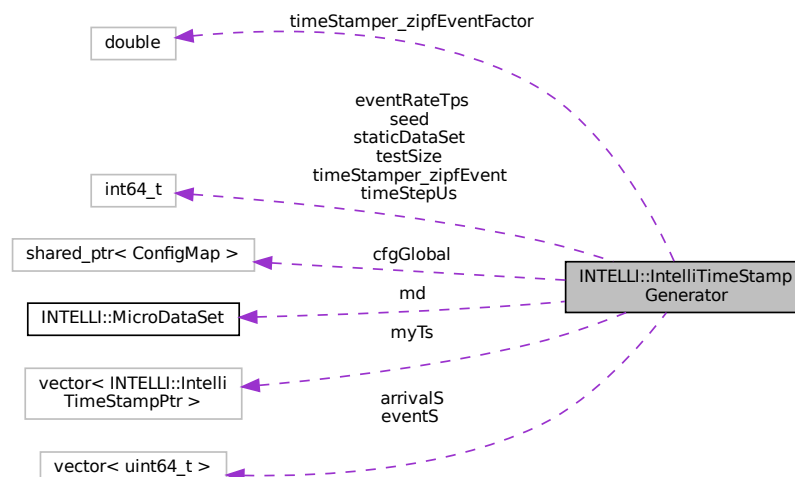
- include/Utils/[IntelliTimeStampGenerator.h](#)

8.104 INTELLI::IntelliTimeStampGenerator Class Reference

The basic class to generate time stamps.

```
#include <Utils/IntelliTimeStampGenerator.h>
```

Collaboration diagram for INTELLI::IntelliTimeStampGenerator:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
Set the GLOBAL config map related to this TimerStamper.
- virtual std::vector< [INTELLI::IntelliTimeStampPtr](#) > [getTimeStamps](#) ()
get the vector of time stamps

Public Attributes

- `std::vector< INTELLI::IntelliTimeStampPtr > myTs`

Protected Member Functions

- `void generateEvent ()`
generate the vector of event
- `void generateArrival ()`
generate the vector of arrival
- `void generateFinal ()`
generate the final result of s and r
- `std::vector< INTELLI::IntelliTimeStampPtr > constructTimeStamps (std::vector< uint64_t > eventS, std::vector< uint64_t > arrivalS)`

Protected Attributes

- `INTELLI::ConfigMapPtr cfgGlobal`
- `INTELLI::MicroDataSet md`
- `int64_t timeStamper_zipfEvent = 0`
- `double timeStamper_zipfEventFactor = 0`
- `int64_t testSize`
- `std::vector< uint64_t > eventS`
- `std::vector< uint64_t > arrivalS`
- `int64_t eventRateTps = 0`
- `int64_t timeStepUs = 40`
- `int64_t seed = 114514`
- `int64_t staticDataSet = 0`

8.104.1 Detailed Description

The basic class to generate time stamps.

Note

require configs:

- `eventRateTps` I64 The real-world rate of spawn event, in Tuples/s
- `streamingTupleCnt` I64 The number of "streaming tuples", can be set to the #rows or #cols of a matrix
- `timeStamper_zipfEvent`, I64, whether or not using the zipf for event rate, default 0
- `timeStamper_zipfEventFactor`, Double, the zipf factor for event rate, default 0.1, should be 0~1
- `staticDataSet`, I64, 0, whether or not treat a dataset as static

Default behavior

- `create`
- call `setConfig` to generate the timestamp under instructions
- call `getTimeStamps` to get the timestamp

8.104.2 Member Function Documentation

8.104.2.1 generateArrival()

```
void INTELLI::IntelliTimeStampGenerator::generateArrival ( ) [protected]
```

generate the vector of arrival

Note

As we do not consider OoO now, this is a dummy function

8.104.2.2 getTimeStamps()

```
std::vector< INTELLI::IntelliTimeStampPtr > INTELLI::IntelliTimeStampGenerator::getTimeStamps  
( ) [virtual]
```

get the vector of time stamps

Returns

the vector

8.104.2.3 setConfig()

```
bool INTELLI::IntelliTimeStampGenerator::setConfig (  
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this TimerStamper.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

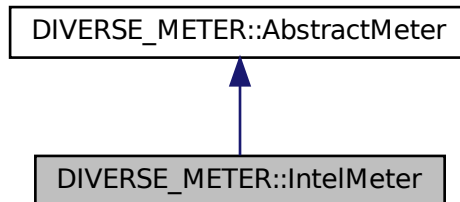
The documentation for this class was generated from the following files:

- include/Utils/[IntelliTimeStampGenerator.h](#)
- src/Utils/[IntelliTimeStampGenerator.cpp](#)

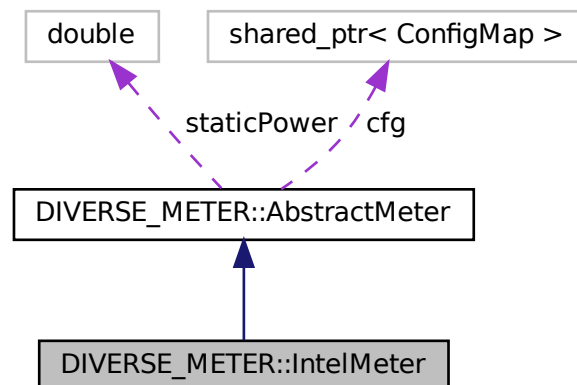
8.105 DIVERSE_METER::IntelMeter Class Reference

the entity of intel msr-based power meter, may be not support for some newer architectures

Inheritance diagram for DIVERSE_METER::IntelMeter:



Collaboration diagram for DIVERSE_METER::IntelMeter:



Public Member Functions

- virtual void `setConfig` (`INTELLI::ConfigMapPtr _cfg`)
to set the configmap
- void `startMeter` ()
to start the meter into some measuring tasks
- void `stopMeter` ()
to stop the meter into some measuring tasks
- double `getE` ()
to get the energy in J, including static energy consumption of system
- bool `isValid` ()

Additional Inherited Members

8.105.1 Detailed Description

the entity of intel msr-based power meter, may be not support for some newer architectures

- create
- call [setConfig\(\)](#) to config this meter
- (optional) call [testStaticPower\(\)](#) to test the static power of a device, if you want to exclude it
- call [startMeter\(\)](#) to start measurement
- (run your program)
- call [stopMeter\(\)](#) to stop measurement
- call [getE\(\)](#), [getPeak\(\)](#), etc to get the measurement results

Warning

: only works for some x64 machines

Note

: no peak power support, tag is "intelMsr"

8.105.2 Member Function Documentation

8.105.2.1 setConfig()

```
void IntelMeter::setConfig (
    INTELLI::ConfigMapPtr _cfg ) [virtual]
```

to set the configmap

Parameters

<i>cfg</i>	the config map
------------	----------------

Reimplemented from [DIVERSE_METER::AbstractMeter](#).

The documentation for this class was generated from the following files:

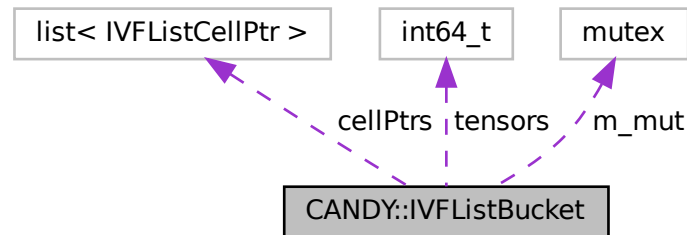
- include/Utils/Meters/IntelMeter/IntelMeter.hpp
- src/Utils/Meters/IntelMeter/IntelMeter.cpp

8.106 CANDY::IVFListBucket Class Reference

a bucket of multiple [IVFListCell](#)

```
#include <CANDY/OnlinePQIndex/IVFTensorEncodingList.h>
```

Collaboration diagram for CANDY::IVFListBucket:



Public Member Functions

- `int64_t size ()`
- `void lock ()`
lock this bucket
- `void unlock ()`
unlock this bucket
- `void insertTensorWithEncode (torch::Tensor &t, std::vector< uint8_t > &encode, bool isConcurrent=false)`
insert a tensor with its encode
- `bool deleteTensorWithEncode (torch::Tensor &t, std::vector< uint8_t > &encode, bool isConcurrent=false)`
delete a tensor with its encode
- `bool deleteTensor (torch::Tensor &t, bool isConcurrent=false)`
delete a tensor
- `torch::Tensor getAllTensors ()`
get all of the tensors in list
- `torch::Tensor getAllTensorsWithEncode (std::vector< uint8_t > &_encode)`
get all of the tensors in list with a specific encode
- `int64_t sizeWithEncode (std::vector< uint8_t > &_encode)`
get teh size in list with a specific encode
- `torch::Tensor getMinimumTensorsUnderHamming (std::vector< uint8_t > &_encode, int64_t minNumber, int64_t _vecDim)`
get a minimum number of tensors under sorted hamming distance

Protected Attributes

- `int64_t tensors = 0`
- `std::list< IVFListCellPtr > cellPtrs`
- `std::mutex m_mut`

8.106.1 Detailed Description

a bucket of multiple [IVFListCell](#)

8.106.2 Member Function Documentation

8.106.2.1 deleteTensor()

```
bool CANDY::IVFListBucket::deleteTensor (
    torch::Tensor & t,
    bool isConcurrent = false )
```

delete a tensor

Note

will check the equal condition by torch::equal

Parameters

<i>t</i>	the tensor
<i>isConcurrent</i>	whether this process is concurrently executed •

Returns

bool whether the tensor is really deleted

8.106.2.2 deleteTensorWithEncode()

```
bool CANDY::IVFListBucket::deleteTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    bool isConcurrent = false )
```

delete a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.106.2.3 getAllTensors()

```
torch::Tensor CANDY::IVFListBucket::getAllTensors (
    void )
```

get all of the tensors in list

Returns

a 2-D tensor contain all, torch::zeros({1,1}) if got nothing

8.106.2.4 getAllTensorsWithEncode()

```
torch::Tensor CANDY::IVFListBucket::getAllTensorsWithEncode (
    std::vector< uint8_t > & _encode )
```

get all of the tensors in list with a specific encode

Parameters

<i>_encode</i>	the specified encode
----------------	----------------------

Returns

a 2-D tensor contain all, torch::zeros({1,1}) if got nothing

8.106.2.5 getMinimumTensorsUnderHamming()

```
torch::Tensor CANDY::IVFListBucket::getMinimumTensorsUnderHamming (
    std::vector< uint8_t > & _encode,
    int64_t minNumber,
    int64_t _vecDim )
```

get a minimum number of tensors under sorted hamming distance

Parameters

<i>_encode</i>	the specified encode
<i>minNumber</i>	the minimum of desired tensors
<i>_vecDim</i>	the dimension of database vectors

Returns

a 2-D tensor or result, `torch::zeros({1,1})` if got nothing

1. try exact match
2. scan
3. sort

8.106.2.6 insertTensorWithEncode()

```
void CANDY::IVFListBucket::insertTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>isConcurrent</i>	whether this process is concurrently executed

8.106.2.7 sizeWithEncode()

```
int64_t CANDY::IVFListBucket::sizeWithEncode (
    std::vector< uint8_t > & _encode )
```

get teh size in list with a specific encode

Parameters

<i>_encode</i>	the specified encode
----------------	----------------------

Returns

the size under *_encode*

The documentation for this class was generated from the following files:

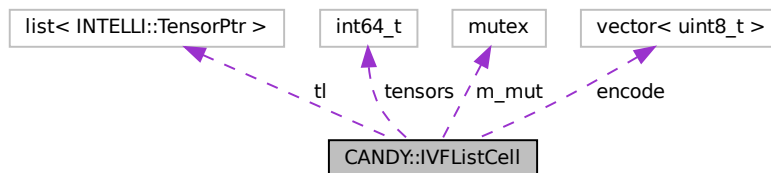
- [include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h](#)
- [src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp](#)

8.107 CANDY::IVFListCell Class Reference

a cell of row tensor pointers which have the same code

```
#include <CANDY/OnlinePQIndex/IVFTensorEncodingList.h>
```

Collaboration diagram for CANDY::IVFListCell:



Public Member Functions

- `int64_t size ()`
- `void lock ()`
lock this cell
- `void unlock ()`
unlock this cell
- `void setEncode (std::vector< uint8_t > _encode)`
- `std::vector< uint8_t > getEncode ()`
- `void insertTensor (torch::Tensor &t)`
insert a tensor
- `void insertTensorPtr (INTELLI::TensorPtr tp)`
insert a tensor pointer
- `bool deleteTensor (torch::Tensor &t)`
delete a tensor
- `bool deleteTensorPtr (INTELLI::TensorPtr tp)`
delete a tensor pointer
- `torch::Tensor getAllTensors ()`
get all of the tensors in list

Protected Attributes

- `int64_t tensors = 0`
- `std::list< INTELLI::TensorPtr > tl`
- `std::mutex m_mut`
- `std::vector< uint8_t > encode`

8.107.1 Detailed Description

a cell of row tensor pointers which have the same code

8.107.2 Member Function Documentation

8.107.2.1 deleteTensor()

```
bool CANDY::IVFListCell::deleteTensor (
    torch::Tensor & t )
```

delete a tensor

Note

will check the equal condition by torch::equal

Parameters

<i>t</i>	the tensor @returen bool whether the tensor is really deleted
----------	---

8.107.2.2 deleteTensorPtr()

```
bool CANDY::IVFListCell::deleteTensorPtr (
    INTELLI::TensorPtr tp )
```

delete a tensor pointer

Note

will check the equal condition by pointer ==

Parameters

<i>tp</i>	the tensor pointer @returen bool whether the tensor is really deleted
-----------	---

8.107.2.3 getAllTensors()

```
torch::Tensor CANDY::IVFListCell::getAllTensors (
    void )
```

get all of the tensors in list

Returns

a 2-D tensor contain all, torch::zeros({1,1}) if got nothing

8.107.2.4 insertTensor()

```
void CANDY::IVFListCell::insertTensor (
    torch::Tensor & t )
```

insert a tensor

Parameters

<i>t</i>	the tensor
----------	------------

8.107.2.5 insertTensorPtr()

```
void CANDY::IVFListCell::insertTensorPtr (
    INTELLI::TensorPtr tp )
```

insert a tensor pointer

Parameters

<i>tp</i>	the tensor pointer
-----------	--------------------

The documentation for this class was generated from the following files:

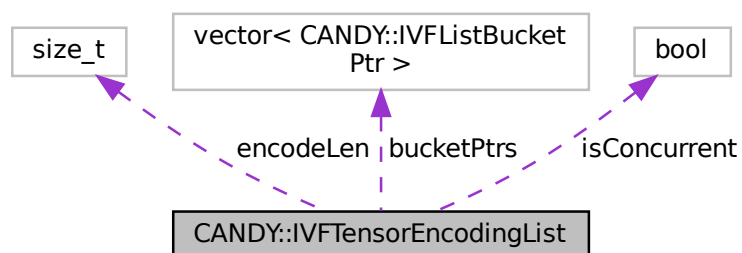
- include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h
- src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp

8.108 CANDY::IVFTensorEncodingList Class Reference

The inverted file (IVF) list to organize tensor and its encodings.

```
#include <CANDY/OnlinePQIndex/IVFTensorEncodingList.h>
```

Collaboration diagram for CANDY::IVFTensorEncodingList:



Public Member Functions

- void [init](#) (size_t bkts, size_t _encodeLen)
init this IVFList
- void [insertTensorWithEncode](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool isConcurrent=false)
insert a tensor with its encode
- bool [deleteTensorWithEncode](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool isConcurrent=false)
delete a tensor with its encode
- torch::Tensor [getMinimumNumOfTensors](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, int64_t minimumNum)
get minimum number of tensors that are candidate to query t
- torch::Tensor [getMinimumNumOfTensorsHamming](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, int64_t minimumNum)
get minimum number of tensors that are candidate to query t, using hamming distance
- torch::Tensor [getMinimumNumOfTensorsInsideBucket](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, int64_t minimumNum)
get minimum number of tensors that are candidate to query t, must inside a bucket
- torch::Tensor [getMinimumNumOfTensorsInsideBucketHamming](#) (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, int64_t minimumNum)
get minimum number of tensors that are candidate to query t, must inside a bucket

Public Attributes

- bool **isConcurrent** = false

Static Protected Member Functions

- static uint8_t [getLeftIdxU8](#) (uint8_t idx, uint8_t leftOffset, bool *reachedLeftMost)
- static uint8_t [getRightIdxU8](#) (uint8_t idx, uint8_t rightOffset, bool *reachedRightMost)

Protected Attributes

- std::vector< [CANDY::IVFListBucketPtr](#) > **bucketPtrs**
- size_t **encodeLen** = 0

8.108.1 Detailed Description

The inverted file (IVF) list to organize tensor and its encodings.

8.108.2 Member Function Documentation

8.108.2.1 deleteTensorWithEncode()

```
bool CANDY::IVFTensorEncodingList::deleteTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    bool isConcurrent = false )
```

delete a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.108.2.2 getMinimumNumOfTensors()

```
torch::Tensor CANDY::IVFTensorEncodingList::getMinimumNumOfTensors (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    int64_t minimumNum )
```

get minimum number of tensors that are candidate to query t

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

- a 2-D tensor contain all, torch::zeros({minimumNum,D}) if got nothing

1. test whether the bucket[idx] has enough tensors,
2. if not, try to expand

8.108.2.3 getMinimumNumOfTensorsHamming()

```
torch::Tensor CANDY::IVFTensorEncodingList::getMinimumNumOfTensorsHamming (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    int64_t minimumNum )
```

get minimum number of tensors that are candidate to query t, using hamming distance

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

- a 2-D tensor contain all, torch::zeros({minimumNum,D}) if got nothing
1. test whether the buckt[idx] has enough tensors,
 2. if not, try to expand

8.108.2.4 getMinimumNumOfTensorsInsideBucket()

```
torch::Tensor CANDY::IVFTensorEncodingList::getMinimumNumOfTensorsInsideBucket (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    int64_t minimumNum )
```

get minimum number of tensors that are candidate to query t, must inside a bucket

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

- a 2-D tensor contain all, torch::zeros({minimumNum,D}) if got nothing

Todo improve the efficiency of this function in travsing lists!

1. get the exact encode

probe the i th byte with left and right expand

left

right

8.108.2.5 getMinimumNumOfTensorsInsideBucketHamming()

```
torch::Tensor CANDY::IVFTensorEncodingList::getMinimumNumOfTensorsInsideBucketHamming (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    int64_t minimumNum )
```

get minimum number of tensors that are candidate to query t, must inside a bucket

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

- a 2-D tensor contain all, torch::zeros({minimumNum,D}) if got nothing

Todo improve the efficiency of this function in traving lists!

8.108.2.6 init()

```
void CANDY::IVFTensorEncodingList::init (
    size_t bkts,
    size_t _encodeLen )
```

init this IVFList

Parameters

<i>bkts</i>	the number of buckets
<i>_encodeLen</i>	the length of tensors' encoding

8.108.2.7 insertTensorWithEncode()

```
void CANDY::IVFTensorEncodingList::insertTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

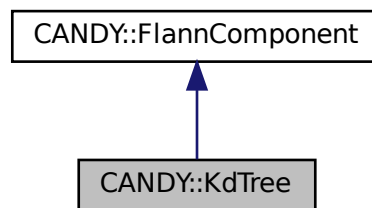
<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

The documentation for this class was generated from the following files:

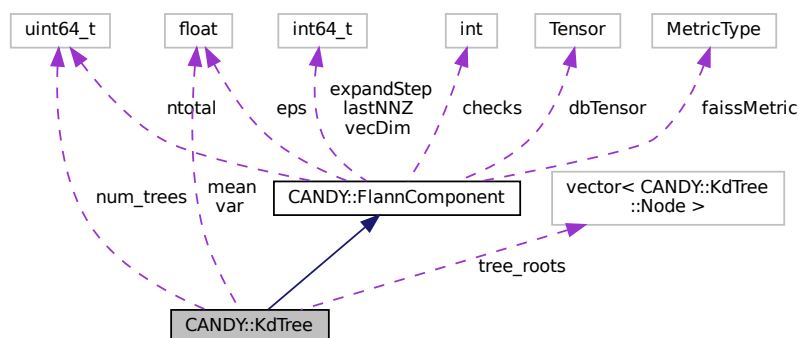
- [include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h](#)
- [src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp](#)

8.109 CANDY::KdTree Class Reference

Inheritance diagram for CANDY::KdTree:



Collaboration diagram for CANDY::KdTree:



Classes

- struct [Node](#)

Public Types

- typedef [Node](#) * **NodePtr**
- typedef [FLANN::BranchStruct](#)< [NodePtr](#) > **BranchSt**
- typedef [BranchSt](#) * **Branch**

Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg) override
set the index-specific config related to one index
- void [addPointToTree](#) ([NodePtr](#) node, int64_t idx)
add dbTensor[idx] to tree with root as node
- virtual void [addPoints](#) (torch::Tensor &t) override
add data into the tree either by reconstruction or appending
- virtual int [knnSearch](#) (torch::Tensor &q, int64_t *idx, float *distances, int64_t aknn) override
perform knn-search on the kdTree structure
- virtual bool [setParams](#) ([FlannParam](#) param) override
set the params from auto-tuning
- void [getNeighbors](#) ([FLANN::ResultSet](#) &result, const float *vec, int maxCheck, float epsError)
- void [searchLevel](#) ([FLANN::ResultSet](#) &result, const float *vec, [NodePtr](#) node, float mindist, int &checkCount, int maxCheck, float epsError, [FLANN::Heap](#)< [BranchSt](#) > *heap, [FLANN::VisitBitset](#) &checked)
search from a given node of the tree
- void [buildTree](#) ()
build the tree from scratch
- [NodePtr](#) [divideTree](#) (int64_t *idx, int count)
create a node that subdivides vectors from data[first] to data[last]. Called recursively on each subset
- void [meanSplit](#) (int64_t *ind, int count, int64_t &index, int64_t &cutfeat, float &cutval)
choose which feature to use to subdivide this subset of vectors by randomly choosing those with highest variance
- int [selectDivision](#) (float *v)
select top RAND_DIM largest values from vector and return index of one of them at random
- void [planeSplit](#) (int64_t *ind, int count, int64_t cutfeat, float cutval, int &lim1, int &lim2)
subdivide the lists by a plane perpendicular on axe corresponding to the cutfeat dimension at cutval position

Public Attributes

- uint64_t [num_trees](#)
Number of randomized trees that are used in forest.
- float * **mean**
- float * **var**
- std::vector< [NodePtr](#) > [tree_roots](#)
array of num_trees to specify roots

8.109.1 Member Function Documentation

8.109.1.1 addPoints()

```
void CANDY::KdTree::addPoints (
    torch::Tensor & t ) [override], [virtual]
```

add data into the tree either by reconstruction or appending

Parameters

<i>t</i>	new data
----------	----------

Reimplemented from [CANDY::FlannComponent](#).

8.109.1.2 addPointToTree()

```
void CANDY::KdTree::addPointToTree (
    NodePtr node,
    int64_t idx )
```

add dbTensor[idx] to tree with root as node

Parameters

<i>node</i>	typically a tree root
<i>idx</i>	index in dbTensor

8.109.1.3 divideTree()

```
CANDY::KdTree::NodePtr CANDY::KdTree::divideTree (
    int64_t * idx,
    int count )
```

create a node that subdivides vectors from data[first] to data[last]. Called recursively on each subset

Parameters

<i>idx</i>	index of this vector
<i>count</i>	number of vectors in this sublist

Returns

8.109.1.4 getNeighbors()

```
void CANDY::KdTree::getNeighbors (
    FLANN::ResultSet & result,
    const float * vec,
    int maxCheck,
    float epsError )
```


Parameters

<i>result</i>	
<i>vec</i>	
<i>maxCheck</i>	
<i>epsError</i>	

8.109.1.5 knnSearch()

```
int CANDY::KdTree::knnSearch (
    torch::Tensor & q,
    int64_t * idx,
    float * distances,
    int64_t aknn ) [override], [virtual]
```

perform knn-search on the kdTree structure

Parameters

<i>q</i>	query data to be searched
<i>idx</i>	result vectors indices
<i>distances</i>	result vectors' distances with query
<i>aknn</i>	number of approximate neighbors

Returns

number of results obtained

Reimplemented from [CANDY::FlannComponent](#).

8.109.1.6 meanSplit()

```
void CANDY::KdTree::meanSplit (
    int64_t * ind,
    int count,
    int64_t & index,
    int64_t & cutfeat,
    float & cutval )
```

choose which feature to use to subdivide this subset of vectors by randomly choosing those with highest variance

Parameters

<i>ind</i>	index of this vector
<i>count</i>	number of vectors in this sublist
<i>index</i>	index where the sublist split
<i>cutfeat</i>	index of highest variance as cut feature
<i>cutval</i>	value of highest variance

8.109.1.7 planeSplit()

```
void CANDY::KdTree::planeSplit (
    int64_t * ind,
    int count,
    int64_t cutfeat,
    float cutval,
    int & lim1,
    int & lim2 )
```

subdivide the lists by a plane perpendicular on axe corresponding to the cutfeat dimension at cutval position

Parameters

<i>ind</i>	index of the list
<i>count</i>	count of the list
<i>cutfeat</i>	the chosen feature
<i>cutval</i>	the threshold value to be compared
<i>lim1</i>	split index candidate for meansplit
<i>lim2</i>	split index candidate for meansplit

8.109.1.8 searchLevel()

```
void CANDY::KdTree::searchLevel (
    FLANN::ResultSet & result,
    const float * vec,
    NodePtr node,
    float mindist,
    int & checkCount,
    int maxCheck,
    float epsError,
    FLANN::Heap< BranchSt > * heap,
    FLANN::VisitBitset & checked )
```

search from a given node of the tree

Parameters

<i>result</i>	priority queue to store results
<i>vec</i>	vector to be searched
<i>node</i>	current node to be traversed
<i>mindist</i>	current minimum distance obtained
<i>checkCount</i>	count of checks on multiple trees
<i>maxCheck</i>	max check on multiple trees
<i>epsError</i>	error to be compared with worst distance
<i>heap</i>	heap structure to store branches
<i>checked</i>	visited bitmap

TODO: not sure + IP

8.109.1.9 selectDivision()

```
int CANDY::KdTree::selectDivision (
    float * v )
```

select top RAND_DIM largest values from vector and return index of one of them at random

Parameters

<i>v</i>	values of variance
----------	--------------------

Returns

the index of randomly chosen highest variance

8.109.1.10 setConfig()

```
bool CANDY::KdTree::setConfig (
    INTELLI::ConfigMapPtr cfg ) [override], [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::FlannComponent](#).

8.109.1.11 setParams()

```
bool CANDY::KdTree::setParams (
    FlannParam param ) [override], [virtual]
```

set the params from auto-tuning

Parameters

<i>param</i>	best param
--------------	------------

Returns

true if success

Reimplemented from [CANDY::FlannComponent](#).

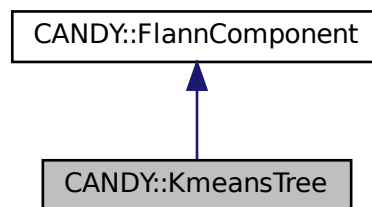
The documentation for this class was generated from the following files:

- include/CANDY/FlannIndex/KdTree.h
- src/CANDY/FlannIndex/KdTree.cpp

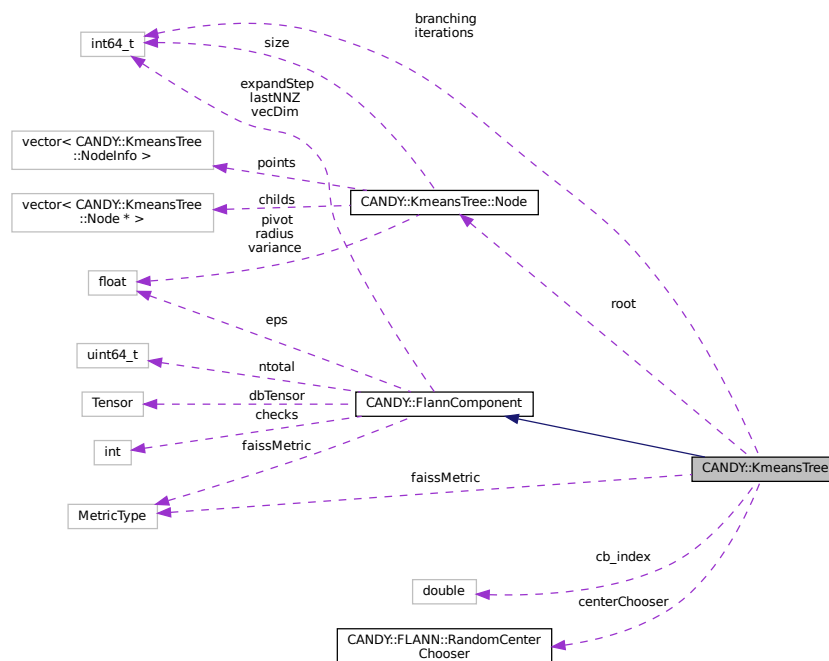
8.110 CANDY::KmeansTree Class Reference

The structure representing hierarchical k-means tree used in FLANN.

Inheritance diagram for CANDY::KmeansTree:



Collaboration diagram for CANDY::KmeansTree:



Classes

- struct [Node](#)
- struct [NodeInfo](#)

Public Types

- typedef [Node](#) * **NodePtr**
- typedef [FLANN::BranchStruct](#)< [NodePtr](#) > **BranchSt**
- typedef [BranchSt](#) * **Branch**

Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg) override
set the index-specific config related to one index
- void [addPointToTree](#) ([NodePtr](#) node, int64_t idx, float dist)
add dbTensor[idx] to tree with root as node
- virtual void [addPoints](#) (torch::Tensor &t) override
add data into the tree either by reconstruction or appending
- void [computeNodeStat](#) ([NodePtr](#) node, std::vector< int64_t > &indices)
compute the radius, variance and mean for this cluster
- void [computeClustering](#) ([NodePtr](#) node, int64_t *indices, int64_t indices_length, int64_t branching)
- virtual int [knnSearch](#) (torch::Tensor &q, int64_t *idx, float *distances, int64_t aknn) override
perform knn-search on the kdTree structure
- virtual bool [setParams](#) ([FlannParam](#) param) override
set the params from auto-tuning
- void [getNeighbors](#) ([FLANN::ResultSet](#) &result, float *vec, int maxCheck)
called by knnSearch, to search the vec within the true
- int64_t [explore](#) ([NodePtr](#) node, float *q, [FLANN::Heap](#)< [BranchSt](#) > *heap)
explore from the node for the closest center
- void [findNN](#) ([NodePtr](#) node, [FLANN::ResultSet](#) &result, float *vec, int &check, int maxCheck, [FLANN::Heap](#)< [BranchSt](#) > *heap)
practice KNN search

Public Attributes

- int64_t [branching](#)
branching factor used in clustering
- int64_t [iterations](#)
number of max iterations when clustering
- double [cb_index](#) = 0.4
Cluster border index used in tree search when choosing the closest cluster to search next;
- [NodePtr](#) [root](#)
root of tree
- [FLANN::RandomCenterChooser](#) * [centerChooser](#)
the center chooser in clustering; currently only implemented randomChooser
- faiss::MetricType [faissMetric](#) = faiss::METRIC_L2

8.110.1 Detailed Description

The structure representing hierarchical k-means tree used in FLANN.

8.110.2 Member Function Documentation

8.110.2.1 addPoints()

```
void CANDY::KmeansTree::addPoints (
    torch::Tensor & t ) [override], [virtual]
```

add data into the tree either by reconstruction or appending

Parameters

<i>t</i>	new data
----------	----------

Reimplemented from [CANDY::FlannComponent](#).

8.110.2.2 addPointToTree()

```
void CANDY::KmeansTree::addPointToTree (
    NodePtr node,
    int64_t idx,
    float dist )
```

add dbTensor[idx] to tree with root as node

Parameters

<i>node</i>	typically a tree root
<i>idx</i>	index in dbTensor
<i>dist</i>	

8.110.2.3 computeClustering()

```
void CANDY::KmeansTree::computeClustering (
    NodePtr node,
    int64_t * indices,
    int64_t indices_length,
    int64_t branching )
```

#brief compute the cluster iteratively

Parameters

<i>node</i>	the node where the cluster starts
<i>indices</i>	indexes to be involved
<i>indices_length</i>	length of indexes to be involved
<i>branching</i>	number of branching in tree

8.110.2.4 computeNodeStat()

```
void CANDY::KmeansTree::computeNodeStat (
    NodePtr node,
    std::vector< int64_t > & indices )
```

compute the radius, variance and mean for this cluster

Parameters

<i>node</i>	the node representing the cluster
<i>indices</i>	the indexes within the cluster

8.110.2.5 explore()

```
int64_t CANDY::KmeansTree::explore (
    NodePtr node,
    float * q,
    FLANN::Heap< BranchSt > * heap )
```

explore from the node for the closest center

Parameters

<i>node</i>	node to be explored
<i>q</i>	query vector
<i>heap</i>	heap set

Returns

the index of center

8.110.2.6 findNN()

```
void CANDY::KmeansTree::findNN (
    NodePtr node,
```

```

    FLANN::ResultSet & result,
    float * vec,
    int & check,
    int maxCheck,
    FLANN::Heap< BranchSt > * heap )

```

practice KNN search

Parameters

<i>node</i>	starting node
<i>result</i>	result set
<i>vec</i>	query vector
<i>check</i>	current check time
<i>maxCheck</i>	max check times
<i>heap</i>	heap set

8.110.2.7 getNeighbors()

```

void CANDY::KmeansTree::getNeighbors (
    FLANN::ResultSet & result,
    float * vec,
    int maxCheck )

```

called by knnSearch, to search the vec within the true

Parameters

<i>result</i>	result set
<i>vec</i>	vector to be searched
<i>maxCheck</i>	max times to check

8.110.2.8 knnSearch()

```

int CANDY::KmeansTree::knnSearch (
    torch::Tensor & q,
    int64_t * idx,
    float * distances,
    int64_t aknn ) [override], [virtual]

```

perform knn-search on the kdTree structure

Parameters

<i>q</i>	query data to be searched
<i>idx</i>	result vectors indices
<i>distances</i>	result vectors' distances with query
<i>aknn</i>	number of approximate neighbors

Returns

number of results obtained

Reimplemented from [CANDY::FlannComponent](#).

8.110.2.9 setConfig()

```
bool CANDY::KmeansTree::setConfig (
    INTELLI::ConfigMapPtr cfg ) [override], [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::FlannComponent](#).

8.110.2.10 setParams()

```
bool CANDY::KmeansTree::setParams (
    FlannParam param ) [override], [virtual]
```

set the params from auto-tuning

Parameters

<i>param</i>	best param
--------------	------------

Returns

true if success

Reimplemented from [CANDY::FlannComponent](#).

The documentation for this class was generated from the following files:

- include/CANDY/FlannIndex/Kmeans.h
- src/CANDY/FlannIndex/Kmeans.cpp

8.111 INTELLI::MemoryTracker Class Reference

The top entity to trace current, average and maximum memory foot print.

Public Member Functions

- void [start](#) (uint64_t sec, uint64_t usec=0)
To start memory usage tracing.
- void **triggerMemorySample** ()
- void [stop](#) ()
To end memory usage tracing.
- size_t [getAvgMem](#) ()
To return the average memory usage during the sampling.
- double [getAvgCpu](#) ()
To return the average Cpu utilization rate during the sampling.
- size_t [getMaxMem](#) ()
To return the max memory usage during the sampling.
- double [getMaxCpu](#) ()
To return the max Cpu utilization rate during the sampling.
- size_t [getCurMem](#) ()
To return the current memory usage when calling this function.

Static Public Member Functions

- static void **setActiveInstance** ([MemoryTracker](#) *ins)

8.111.1 Detailed Description

The top entity to trace current, average and maximum memory foot print.

Note

The default unit is KB, will use Linux timer to keep sampling memory usage usage

- create a class
- call INTELLI::MemoryTracker::setActiveInstance(&xxx) to register this to linux timer
- call [start](#) to start the sampling
- call end to end the sampling
- call [getAvgMem](#), [getMaxMem](#), or [getCurMem](#) to get the result, [getCurMem](#) is a instant function rather than reporting the sampled results

Warning

Never use multiple instance of INTELLI::MemoryTracker::setActiveInstance(&xxx)

8.111.2 Member Function Documentation

8.111.2.1 getAvgCpu()

```
double INTELLI::MemoryTracker::getAvgCpu ( ) [inline]
```

To return the average Cpu utilization rate during the sampling.

Returns

the fractional

8.111.2.2 getAvgMem()

```
size_t INTELLI::MemoryTracker::getAvgMem ( ) [inline]
```

To return the average memory usage during the sampling.

Returns

size_t the memory usage in KB

8.111.2.3 getCurMem()

```
size_t INTELLI::MemoryTracker::getCurMem ( ) [inline]
```

To return the current memory usage when calling this function.

Returns

size_t the memory usage in KB

8.111.2.4 getMaxCpu()

```
double INTELLI::MemoryTracker::getMaxCpu ( ) [inline]
```

To return the max Cpu utilization rate during the sampling.

Returns

the fractional

8.111.2.5 getMaxMem()

```
size_t INTELLI::MemoryTracker::getMaxMem ( ) [inline]
```

To return the max memory usage during the sampling.

Returns

size_t the memory usage in KB

8.111.2.6 start()

```
void INTELLI::MemoryTracker::start (
    uint64_t sec,
    uint64_t usec = 0 ) [inline]
```

To start memory usage tracing.

Parameters

<i>sec</i>	the second of sampling
<i>usec</i>	the micro-second of sampling

Note

call after setPerfList

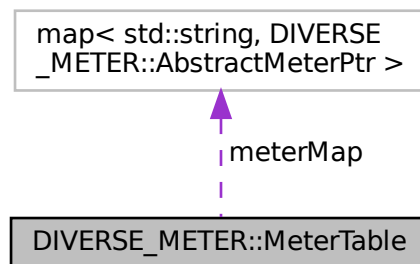
The documentation for this class was generated from the following files:

- include/Utils/MemTracker.h
- src/Utils/MemTracker.cpp

8.112 DIVERSE_METER::MeterTable Class Reference

The table class to index all meters.

Collaboration diagram for DIVERSE_METER::MeterTable:



Public Types

- typedef std::shared_ptr< class [DIVERSE_METER::MeterTable](#) > [MeterTablePtr](#)
The class to describe a shared pointer to [MeterTable](#).

Public Member Functions

- [MeterTable](#) ()
The constructing function.
- void [registerNewMeter](#) (DIVERSE_METER::AbstractMeterPtr dnew, std::string tag)
To register a new meter.
- DIVERSE_METER::AbstractMeterPtr [findMeter](#) (std::string name)
find a meter in the table according to its name

Protected Attributes

- std::map< std::string, DIVERSE_METER::AbstractMeterPtr > **meterMap**

8.112.1 Detailed Description

The table class to index all meters.

Note

Default behavior

- create
- (optional) call [registerNewMeter](#) for new meter
- find a loader by [findMeter](#) using its tag

default tags

- espUart [EspMeterUart](#)
- intelMsr [IntelMeter](#)

8.112.2 Constructor & Destructor Documentation

8.112.2.1 MeterTable()

```
DIVERSE_METER::MeterTable::MeterTable ( )
```

The constructing function.

Note

If new MatrixLoader wants to be included by default, please revise the following in *.cpp
revise me if you need new loader

8.112.3 Member Function Documentation

8.112.3.1 findMeter()

```
DIVERSE_METER::AbstractMeterPtr DIVERSE_METER::MeterTable::findMeter (
    std::string name ) [inline]
```

find a meter in the table according to its name

Parameters

<i>name</i>	The nameTag of loader
-------------	-----------------------

Returns

The Meter, nullptr if not found

8.112.3.2 registerNewMeter()

```
void DIVERSE_METER::MeterTable::registerNewMeter (
    DIVERSE_METER::AbstractMeterPtr dnew,
    std::string tag ) [inline]
```

To register a new meter.

Parameters

<i>onew</i>	The new operator
<i>tag</i>	The name tag

The documentation for this class was generated from the following files:

- include/Utils/Meters/MeterTable.h
- src/Utils/Meters/MeterTable.cpp

8.113 INTELLI::MicroDataSet Class Reference

The all-in-one class for the Micro dataset.

```
#include <Utils/MicroDataSet.hpp>
```

Public Member Functions

- [MicroDataSet](#) ()=default
default construction, with auto random generator
- [MicroDataSet](#) (uint64_t _seed)
construction with seed
- void [setSeed](#) (uint64_t _seed)
construction with seed
- template<class dType = uint32_t>
vector< dType > [genIncrementalAlphabet](#) (size_t len)
To generate incremental alphabet, starting from 0 and end at len.
- template<class tsType = size_t>
vector< tsType > [genZipfInt](#) (size_t len, tsType maxV, double fac)
The function to generate a vector of integers which has zipf distribution.
- template<class tsType = uint32_t, class genType = std::mt19937>
vector< tsType > [genRandInt](#) (size_t len, tsType maxV, tsType minV=0)
generate the vector of random integer
- template<class dType = double>
vector< dType > [genZipfLut](#) (size_t len, dType fac)
To generate the zipf Lut.
- template<class tsType = size_t>
vector< tsType > [genSmoothTimeStamp](#) (size_t len, size_t step, size_t interval)
The function to generate a vector of timestamp which grows smoothly.
- template<class tsType = size_t>
vector< tsType > [genSmoothTimeStamp](#) (size_t len, size_t maxTime)
- template<class tsType = size_t>
vector< tsType > [genZipfTimeStamp](#) (size_t len, tsType maxTime, double fac)
The function to generate a vector of timestamp which has zipf distribution.

8.113.1 Detailed Description

The all-in-one class for the Micro dataset.

The documentation for this class was generated from the following file:

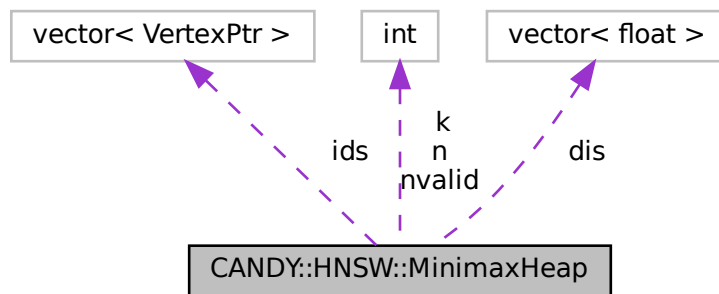
- include/Utils/MicroDataSet.hpp

8.114 CANDY::HNSW::MinimaxHeap Struct Reference

a tiny heap that is used during search

```
#include <HNSW.h>
```

Collaboration diagram for CANDY::HNSW::MinimaxHeap:



Public Types

- typedef faiss::CMax< float, VertexPtr > **HC**

Public Member Functions

- **MinimaxHeap** (int n)
- void **push** (VertexPtr i, float v)
- float **max** () const
- int **size** () const
- void **clear** ()
- VertexPtr **pop_min** (float *vmin_out=nullptr)
- int **count_below** (float thresh)

Public Attributes

- int **n**
- int **k**
- int **nvalid**
- std::vector< VertexPtr > **ids**
- std::vector< float > **dis**

8.114.1 Detailed Description

a tiny heap that is used during search

The documentation for this struct was generated from the following file:

- include/CANDY/HNSWNaive/HNSW.h

8.115 CANDY::MLPBucketIdxModel Class Reference

Public Member Functions

- virtual void [init](#) (int64_t inputDim, int64_t idxMax, [INTELLI::ConfigMapPtr](#) extraConfig)
init the model class
- virtual void [trainModel](#) (torch::Tensor &x1, torch::Tensor &x2, torch::Tensor &labels)
the training function
- virtual torch::Tensor [hash](#) (torch::Tensor input)
the forward hashing function

8.115.1 Member Function Documentation

8.115.1.1 hash()

```
torch::Tensor CANDY::MLPBucketIdxModel::hash (
    torch::Tensor input ) [virtual]
```

the forward hashing function

Parameters

<i>input</i>	The input tensor
--------------	------------------

Returns

the output tensor for encoding

8.115.1.2 init()

```
void CANDY::MLPBucketIdxModel::init (
    int64_t inputDim,
    int64_t idxMax,
    INTELLI::ConfigMapPtr extraConfig ) [virtual]
```

init the model class

Parameters

<i>inputDim</i>	the dimension of model ending input
<i>outputDim</i>	the dimension of model ending output
<i>extraConfig</i>	optional extra configs

Note

accepted configurations

- `cudaBuild` whether or not use cuda to build model, I64, default 0
- `learningRate` the learning rate for training, Double, default 0.01
- `hiddenLayerDim` the dimension of hidden layer, I64, default the same as output layer
- `MLTrainBatchSize` the batch size of ML training, I64, default 64
- `MLTrainMargin` the margin value used in training, Double, default 2*0.1
- `MLTrainEpochs` the number of epochs in training, I64, default 10

8.115.1.3 `trainModel()`

```
void CANDY::MLPBucketIdxModel::trainModel (
    torch::Tensor & x1,
    torch::Tensor & x2,
    torch::Tensor & labels ) [virtual]
```

the training function

Parameters

<i>x1</i>	an 2D tensor sized [n*d]
<i>x2</i>	an 2D tensor sized [n*d]
<i>labels</i>	an 1D integer tensor sized n, indicating whether x1[i] is similar to x2[i]

The documentation for this class was generated from the following files:

- `include/CANDY/HashingModels/MLPBucketIdxModel.h`
- `src/CANDY/HashingModels/MLPBucketIdxModel.cpp`

8.116 `CANDY::MLPHashingModel` Class Reference

Public Member Functions

- virtual void `init` (int64_t inputDim, int64_t outputDim, [INTELLI::ConfigMapPtr](#) extraConfig)
init the model class
- virtual void `trainModel` (torch::Tensor &x1, torch::Tensor &x2, torch::Tensor &labels)
the training function
- virtual void `fineTuneModel` (torch::Tensor &x1, torch::Tensor &x2, torch::Tensor &labels, int64_t epochs, double lr)
the fine tune function
- virtual torch::Tensor `hash` (torch::Tensor input)
the forward hashing function

8.116.1 Member Function Documentation

8.116.1.1 fineTuneModel()

```
void CANDY::MLPHashingModel::fineTuneModel (
    torch::Tensor & x1,
    torch::Tensor & x2,
    torch::Tensor & labels,
    int64_t epochs,
    double lr ) [virtual]
```

the fine tune function

Parameters

<i>x1</i>	an 2D tensor sized [n*d]
<i>x2</i>	an 2D tensor sized [n*d]
<i>labels</i>	an 1D integer tensor sized n, indicating whether x1[i] is similar to x2[i]
<i>epochs</i>	the number of epoches
<i>lr</i>	the learning rate

8.116.1.2 hash()

```
torch::Tensor CANDY::MLPHashingModel::hash (
    torch::Tensor input ) [virtual]
```

the forward hashing function

Parameters

<i>input</i>	The input tensor
--------------	------------------

Returns

the output tensor for encoding

8.116.1.3 init()

```
void CANDY::MLPHashingModel::init (
    int64_t inputDim,
    int64_t outputDim,
    INTELLI::ConfigMapPtr extraConfig ) [virtual]
```

init the model class

Parameters

<i>inputDim</i>	the dimension of model ending input
<i>outputDim</i>	the dimension of model ending output
<i>extraConfig</i>	optional extra configs

Note

accepted configurations

- `cudaBuild` whether or not use cuda to build model, `l64`, default 0
- `learningRate` the learning rate for training, `Double`, default 0.01
- `hiddenLayerDim` the dimension of hidden layer, `l64`, default the same as output layer
- `MLTrainBatchSize` the batch size of ML training, `l64`, default 64
- `MLTrainMargin` the margin value in regulating variance used in training, `Double`, default 0
- `MLTrainEpochs` the number of epochs in training, `l64`, default 10

8.116.1.4 trainModel()

```
void CANDY::MLPHashingModel::trainModel (
    torch::Tensor & x1,
    torch::Tensor & x2,
    torch::Tensor & labels ) [virtual]
```

the training function

Parameters

<i>x1</i>	an 2D tensor sized [n*d]
<i>x2</i>	an 2D tensor sized [n*d]
<i>labels</i>	an 1D integer tensor sized n, indicating whether <code>x1[i]</code> is similar to <code>x2[i]</code>

The documentation for this class was generated from the following files:

- `include/CANDY/HashingModels/MLPHashingModel.h`
- `src/CANDY/HashingModels/MLPHashingModel.cpp`

8.117 BS::multi_future< T > Class Template Reference

A helper class to facilitate waiting for and/or getting the results of multiple futures at once.

```
#include <BS_thread_pool.hpp>
```

Public Member Functions

- [multi_future](#) (const size_t num_futures_=0)
Construct a [multi_future](#) object with the given number of futures.
- std::conditional_t< std::is_void_v< T >, void, std::vector< T > > [get](#) ()
Get the results from all the futures stored in this [multi_future](#) object, rethrowing any stored exceptions.
- std::future< T > & [operator\[\]](#) (const size_t i)
Get a reference to one of the futures stored in this [multi_future](#) object.
- void [push_back](#) (std::future< T > future)
Append a future to this [multi_future](#) object.
- size_t [size](#) () const
Get the number of futures stored in this [multi_future](#) object.
- void [wait](#) () const
Wait for all the futures stored in this [multi_future](#) object.

8.117.1 Detailed Description

```
template<typename T>
class BS::multi_future< T >
```

A helper class to facilitate waiting for and/or getting the results of multiple futures at once.

Template Parameters

<i>T</i>	The return type of the futures.
----------	---------------------------------

8.117.2 Constructor & Destructor Documentation

8.117.2.1 multi_future()

```
template<typename T >
BS::multi_future< T >::multi_future (
    const size_t num_futures_ = 0 ) [inline]
```

Construct a [multi_future](#) object with the given number of futures.

Parameters

<i>num_futures_↵</i>	The desired number of futures to store.
----------------------	---

8.117.3 Member Function Documentation

8.117.3.1 get()

```
template<typename T >
std::conditional_t<std::is_void_v<T>, void, std::vector<T> > BS::multi_future< T >::get ( )
[inline]
```

Get the results from all the futures stored in this [multi_future](#) object, rethrowing any stored exceptions.

Returns

If the futures return void, this function returns void as well. Otherwise, it returns a vector containing the results.

8.117.3.2 operator[]()

```
template<typename T >
std::future<T>& BS::multi_future< T >::operator[] (
    const size_t i ) [inline]
```

Get a reference to one of the futures stored in this [multi_future](#) object.

Parameters

<i>i</i>	The index of the desired future.
----------	----------------------------------

Returns

The future.

8.117.3.3 push_back()

```
template<typename T >
void BS::multi_future< T >::push_back (
    std::future< T > future ) [inline]
```

Append a future to this [multi_future](#) object.

Parameters

<i>future</i>	The future to append.
---------------	-----------------------

8.117.3.4 size()

```
template<typename T >
```

```
size_t BS::multi_future< T >::size ( ) const [inline]
```

Get the number of futures stored in this [multi_future](#) object.

Returns

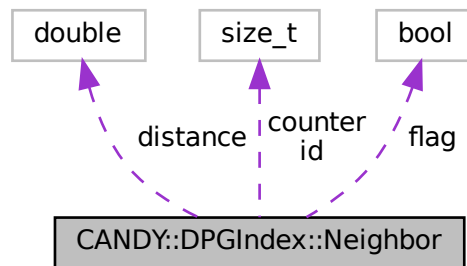
The number of futures.

The documentation for this class was generated from the following file:

- [include/Utils/BS_thread_pool.hpp](#)

8.118 CANDY::DPGIndex::Neighbor Struct Reference

Collaboration diagram for CANDY::DPGIndex::Neighbor:



Public Member Functions

- **Neighbor** (size_t id, double distance, bool f)
- bool **operator**< (const [Neighbor](#) &other) const

Public Attributes

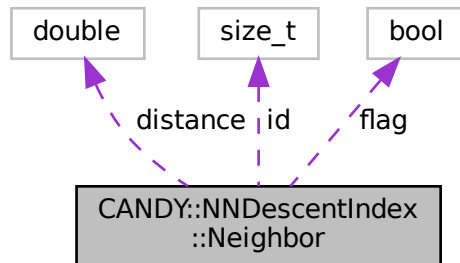
- size_t **id**
- double **distance**
- bool **flag**
- size_t **counter**

The documentation for this struct was generated from the following file:

- [include/CANDY/DPGIndex.h](#)

8.119 CANDY::NNDescentIndex::Neighbor Struct Reference

Collaboration diagram for CANDY::NNDescentIndex::Neighbor:



Public Member Functions

- **Neighbor** (size_t id, double distance, bool f)
- bool **operator**< (const [Neighbor](#) &other) const

Public Attributes

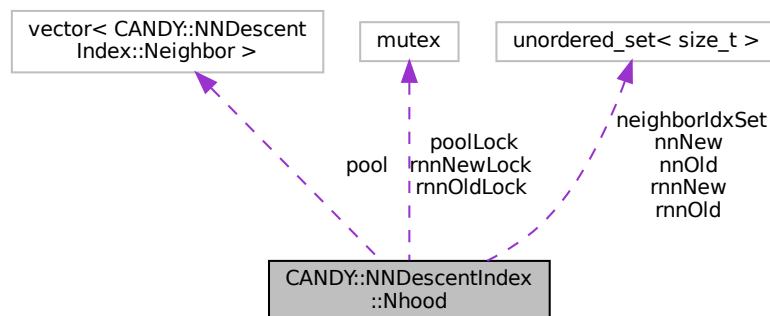
- size_t **id**
- double **distance**
- bool **flag**

The documentation for this struct was generated from the following file:

- include/CANDY/[NNDescentIndex.h](#)

8.120 CANDY::NNDescentIndex::Nhod Struct Reference

Collaboration diagram for CANDY::NNDescentIndex::Nhod:



Public Member Functions

- **Nhood** (const [Nhood](#) &other)

Public Attributes

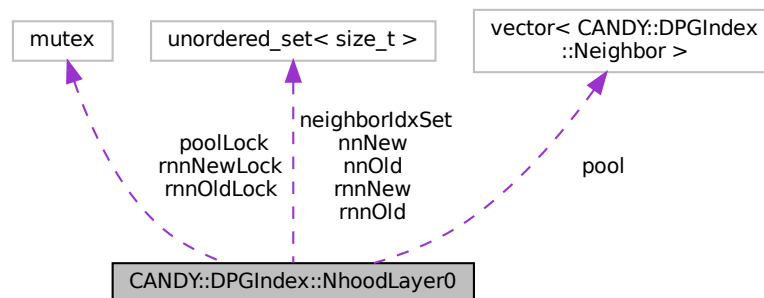
- std::mutex **poolLock**
- std::vector< [Neighbor](#) > **pool**
- std::unordered_set< size_t > **neighborIdxSet**
- std::unordered_set< size_t > **nnOld**
- std::unordered_set< size_t > **nnNew**
- std::mutex **rnnOldLock**
- std::unordered_set< size_t > **rnnOld**
- std::mutex **rnnNewLock**
- std::unordered_set< size_t > **rnnNew**

The documentation for this struct was generated from the following file:

- include/CANDY/[NNDescentIndex.h](#)

8.121 CANDY::DPGIndex::NhoodLayer0 Struct Reference

Collaboration diagram for CANDY::DPGIndex::NhoodLayer0:



Public Member Functions

- **NhoodLayer0** (const [NhoodLayer0](#) &other)

Public Attributes

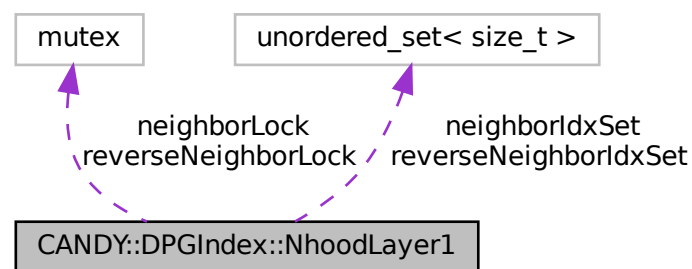
- std::mutex **poolLock**
- std::vector< [Neighbor](#) > **pool**
- std::unordered_set< size_t > **neighborIdxSet**
- std::unordered_set< size_t > **nnOld**
- std::unordered_set< size_t > **nnNew**
- std::mutex **rnnOldLock**
- std::unordered_set< size_t > **rnnOld**
- std::mutex **rnnNewLock**
- std::unordered_set< size_t > **rnnNew**

The documentation for this struct was generated from the following file:

- include/CANDY/[DPGIndex.h](#)

8.122 CANDY::DPGIndex::NhhoodLayer1 Struct Reference

Collaboration diagram for CANDY::DPGIndex::NhhoodLayer1:



Public Member Functions

- **NhhoodLayer1** (const [NhhoodLayer1](#) &other)

Public Attributes

- std::mutex **neighborLock**
- std::mutex **reverseNeighborLock**
- std::unordered_set< size_t > **neighborIdxSet**
- std::unordered_set< size_t > **reverseNeighborIdxSet**

The documentation for this struct was generated from the following file:

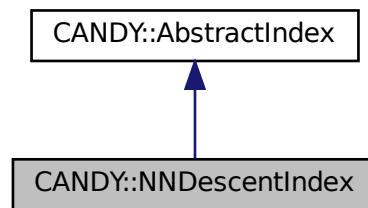
- include/CANDY/[DPGIndex.h](#)

8.123 CANDY::NNDescentIndex Class Reference

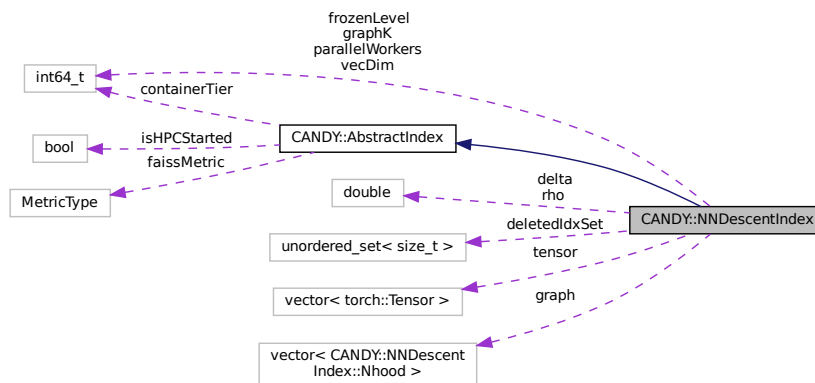
An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized.

```
#include <CANDY/NNDescentIndex.h>
```

Inheritance diagram for CANDY::NNDescentIndex:



Collaboration diagram for CANDY::NNDescentIndex:



Classes

- struct [Neighbor](#)
- struct [Nhood](#)

Public Member Functions

- virtual bool **loadInitialTensor** (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void **reset** ()
reset this index to initied status
- virtual bool **setConfig** (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool **insertTensor** (torch::Tensor &t)
insert a tensor
- virtual bool **deleteTensor** (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool **reviseTensor** (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > **getTensorByIndex** (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor **rawData** ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > **searchTensor** (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool **startHPC** ()
some extra set-ups if the index has HPC fetures
- virtual bool **endHPC** ()
some extra termination if the index has HPC fetures
- virtual bool **setFrozenLevel** (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool **offlineBuild** (torch::Tensor &t)
offline build phase

Protected Member Functions

- void **nnDescent** ()
- void **randomSample** (std::mt19937 &rng, std::vector< size_t > &vec, size_t n, size_t sampledCount)
- bool **updateNN** (size_t i, size_t j, double dist)
- double **calcDist** (const torch::Tensor &ta, const torch::Tensor &tb)
- torch::Tensor **searchOnce** (torch::Tensor q, int64_t k)
- std::vector< std::pair< double, size_t > > **searchOnceInner** (torch::Tensor q, int64_t k)
- bool **insertOnce** (vector< std::pair< double, size_t > > &neighbors, torch::Tensor t)
- bool **deleteOnce** (torch::Tensor t, int64_t k)
- void **parallelFor** (size_t idxSize, std::function< void(size_t)> action)

Protected Attributes

- int64_t **graphK**
- int64_t **parallelWorkers**
- int64_t **vecDim**
- int64_t **frozenLevel**
- double **rho**
- double **delta**
- std::vector< [Nhoo](#)d > **graph**
- std::vector< torch::Tensor > **tensor**
- std::unordered_set< size_t > **deletedIdxSet**

Additional Inherited Members

8.123.1 Detailed Description

An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized.

Note

special parameters

- `parallelWorkers` The number of paraller workers, l64, default 1 (set this to less than 0 will use max hardware_concurrency);
- `vecDim`, the dimension of vectors, default 768, l64
- `graphK`, the neighbors of every node in internal data struct, default 20, l64
- `rho`, sample proportion in NNDescent algorithm which takes effect in offline build only (larger is higher accuracy but lower speed), default 1.0, F64
- `delta`, loop termination condition in NNDescent algorithm which takes effect in offline build only (smaller is higher accuracy but lower speed), default 0.01, F64 @warning Make sure you are using 2D tensors!

8.123.2 Member Function Documentation

8.123.2.1 deleteTensor()

```
bool CANDY::NNDescentIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.2 endHPC()

```
bool CANDY::NNDescentIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.3 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::NNDescentIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.4 insertTensor()

```
bool CANDY::NNDescentIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.5 loadInitialTensor()

```
bool CANDY::NNDescentIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.6 offlineBuild()

```
bool CANDY::NNDescentIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.7 rawData()

```
torch::Tensor CANDY::NNDescentIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.8 reviseTensor()

```
bool CANDY::NNDescentIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::NNDescentIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.10 setConfig()

```
bool CANDY::NNDescentIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.11 setFrozenLevel()

```
bool CANDY::NNDescentIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.123.2.12 startHPC()

```
bool CANDY::NNDescentIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

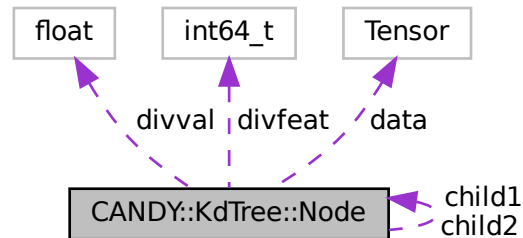
Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

- [include/CANDY/NNDescentIndex.h](#)
- [src/CANDY/NNDescentIndex.cpp](#)

8.124 CANDY::KdTree::Node Struct Reference

Collaboration diagram for CANDY::KdTree::Node:



Public Attributes

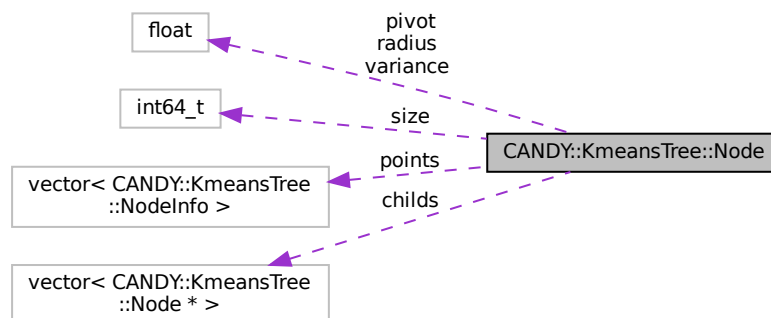
- `int64_t` [divfeat](#)
index used for subdivision.
- `float` [divval](#)
The value used for subdivision.
- `torch::Tensor` [data](#)
Node data.
- `Node *` [child1](#)
- `Node *` [child2](#)

The documentation for this struct was generated from the following file:

- `include/CANDY/FlannIndex/KdTree.h`

8.125 CANDY::KmeansTree::Node Struct Reference

Collaboration diagram for CANDY::KmeansTree::Node:



Public Attributes

- float * [pivot](#)
Cluster center.
- float [radius](#)
Cluster radius.
- float [variance](#)
Cluster variance.
- int64_t [size](#)
Cluster size.
- std::vector< [Node](#) * > [childs](#)
child nodes
- std::vector< [NodeInfo](#) > [points](#)
node points

The documentation for this struct was generated from the following file:

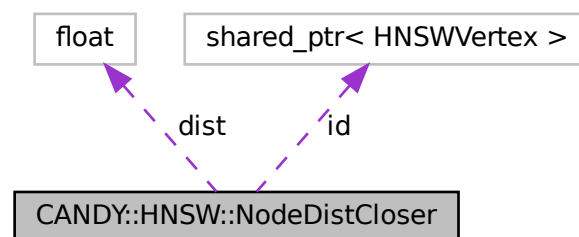
- include/CANDY/FlannIndex/Kmeans.h

8.126 CANDY::HNSW::NodeDistCloser Struct Reference

sort pairs from nearest to farthest by distance

```
#include <HNSW.h>
```

Collaboration diagram for CANDY::HNSW::NodeDistCloser:



Public Member Functions

- **NodeDistCloser** (float dist, VertexPtr id)
- bool **operator**< (const [NodeDistCloser](#) &obj1) const

Public Attributes

- float **dist**
- VertexPtr **id**

8.126.1 Detailed Description

sort pairs from nearest to farthest by distance

The documentation for this struct was generated from the following file:

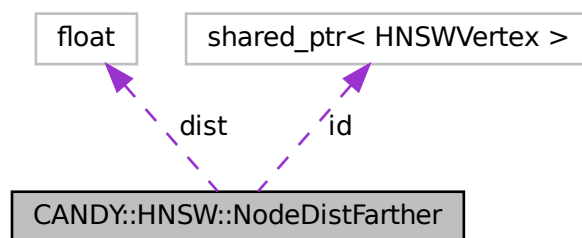
- include/CANDY/HNSWNaive/HNSW.h

8.127 CANDY::HNSW::NodeDistFarther Struct Reference

sort pairs from farthest to nearest

```
#include <HNSW.h>
```

Collaboration diagram for CANDY::HNSW::NodeDistFarther:



Public Member Functions

- **NodeDistFarther** (float dist, VertexPtr id)
- bool **operator**< (const [NodeDistFarther](#) &obj1) const

Public Attributes

- float **dist**
- VertexPtr **id**

8.127.1 Detailed Description

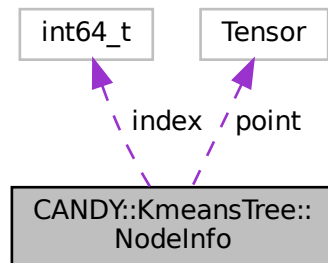
sort pairs from farthest to nearest

The documentation for this struct was generated from the following file:

- include/CANDY/HNSWNaive/HNSW.h

8.128 CANDY::KmeansTree::NodeInfo Struct Reference

Collaboration diagram for CANDY::KmeansTree::NodeInfo:



Public Attributes

- `int64_t` **index**
- `torch::Tensor` **point**

The documentation for this struct was generated from the following file:

- `include/CANDY/FlannIndex/Kmeans.h`

8.129 CANDY::OnlineIVFL2HIndex Class Reference

A L2H (learning 2 hash) indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The L2H function is using ML to approximate spectral hashing principles (NIPS 2008)

```
#include <CANDY/OnlineIVFL2HIndex.h>
```


Protected Attributes

- [MLPHashingModelPtr](#) **myMLModel** = nullptr
- int64_t **buildingSamples** = -1
- int64_t **buildingANNK** = 10
- [FlatIndex](#) **trainIndex**
- double **positiveSampleRatio** = 0.1

Additional Inherited Members

8.129.1 Detailed Description

A L2H (learning 2 hash) indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The L2H function is using ML to approximate spectral hashing principles (NIPS 2008)

Note

currently single thread

using hamming L2H function defined in faiss

config parameters

- vecDim, the dimension of vectors, default 768, I64
- candidateTimes, the times of k to determine minimum candidates, default 1, I64
- numberOfBuckets, the number of first tier buckets, default 1, I64, suggest 2^n
- encodeLen, the length of L2H encoding, in bytes, default 1, I64
- metricType, the type of AKNN metric, default L2, String
- buildingSamples, the number of samples for building internal ML model during initial loading, default -1, I64
- buildingANNK, the ANNK for labeling data as input, default 10, I64

machine learning extra configs

- cudaBuild whether or not use cuda to build model, I64, default 0
- learningRate the learning rate for training, Double, default 0.1
- hiddenLayerDim the dimension of hidden layer, I64, default the same as output layer
- MLTrainBatchSize the batch size of ML training, I64, default 128
- MLTrainMargin the margin value in regulating variance used in training, Double, default 2.0
- MLTrainEpochs the number of epochs in training, I64, default 30
- positiveSampleRatio the ratio of positive samples during self-supervised learning, Double, default 0.1 (should be 0~1)

8.129.2 Member Function Documentation

8.129.2.1 loadInitialTensor()

```
bool CANDY::OnlineIVFL2HIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.129.2.2 loadInitialTensorAndQueryDistribution()

```
bool CANDY::OnlineIVFL2HIndex::loadInitialTensorAndQueryDistribution (
    torch::Tensor & t,
    torch::Tensor & query ) [virtual]
```

load the initial tensors and query distributions of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the data tensor
<i>query</i>	the example query tensor

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.129.2.3 setConfig()

```
bool CANDY::OnlineIVFL2HIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

generate the rotation matrix for random projection

Reimplemented from [CANDY::OnlineIVFLSHIndex](#).

8.129.2.4 trainModelWithData()

```
void CANDY::OnlineIVFL2HIndex::trainModelWithData (
    torch::Tensor & t ) [protected]
```

self-supervised learning on data, including automatic labeling

Parameters

<i>t</i>	the input tensor
----------	------------------

The documentation for this class was generated from the following files:

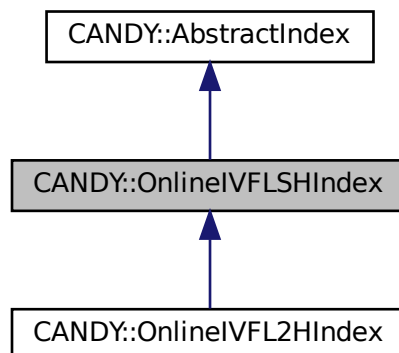
- include/CANDY/OnlineIVFL2HIndex.h
- src/CANDY/[OnlineIVFL2HIndex.cpp](#)

8.130 CANDY::OnlineIVFLSHIndex Class Reference

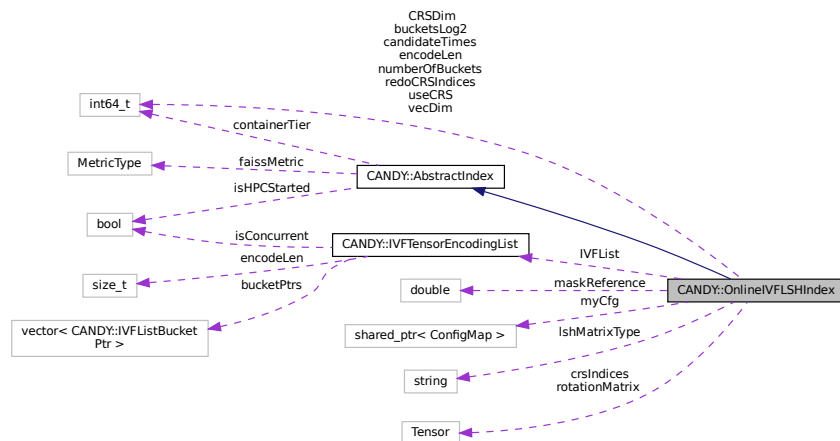
A LSH indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The LSH function is the vanilla random projection (gaussian or random matrix).

```
#include <CANDY/OnlineIVFLSHIndex.h>
```

Inheritance diagram for CANDY::OnlineIVFLSHIndex:



Collaboration diagram for CANDY::OnlineIVFLSHIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors

Static Public Member Functions

- static void [fvecs2bitvecs](#) (const float *x, uint8_t *b, size_t d, size_t n, float ref)
- static void [fvec2bitvec](#) (const float *x, uint8_t *b, size_t d, float ref)
- static torch::Tensor [crsAmm](#) (torch::Tensor &A, torch::Tensor &B, torch::Tensor &indices)
thw column row sampling to compute approximate matrix multiplication

Protected Member Functions

- std::vector< uint8_t > [encodeSingleRow](#) (torch::Tensor &tensor, uint64_t *bucket)
- virtual torch::Tensor [randomProjection](#) (torch::Tensor &a)
- bool [deleteRowsInline](#) (torch::Tensor &t)
the inline function of deleting rows
- void [genCrsIndices](#) (void)
to generate the sampling indices of crs

Protected Attributes

- `INTELLI::ConfigMapPtr myCfg = nullptr`
- `int64_t vecDim = 0`
- `int64_t numberOfBuckets = 1`
- `int64_t encodeLen = 1`
- `int64_t candidateTimes = 1`
- `int64_t useCRS = 0`
- `int64_t CRSDim = 1`
- `int64_t bucketsLog2 = 0`
- `int64_t redoCRSIndices = 0`
- `std::string lshMatrixType = "gaussian"`
- `double maskReference = 0.5`
- `IVFTensorEncodingList IVFList`
- `torch::Tensor rotationMatrix`
- `torch::Tensor crsIndices`

Additional Inherited Members

8.130.1 Detailed Description

A LSH indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The LSH function is the vanilla random projection (gaussian or random matrix).

Note

currently single thread

using hamming LSH function defined in faiss

config parameters

- `vecDim`, the dimension of vectors, default 768, l64
- `candidateTimes`, the times of k to determine minimum candidates, default 1, l64
- `numberOfBuckets`, the number of first tier buckets, default 1, l64, suggest 2^n
- `encodeLen`, the length of LSH encoding, in bytes, default 1, l64
- `metricType`, the type of AKNN metric, default L2, String
- `lshMatrixType`, the type of lsh matrix, default gaussian, String
 - gaussian means a $N(0,1)$ LSH matrix
 - random means a random matrix where each value ranges from $-0.5 \sim 0.5$
- `useCRS`, whether or not use column row sampling in projecting the vector, 0 (No), l64
 - further trade off of accuracy v.s. efficiency
- `CRSDim`, the dimension which are not pruned by crs, 1/10 of `vecDim`, l64
- `redoCRSIndices`, whether or not re-generate the indices of CRS, 0 (No), l64

8.130.2 Member Function Documentation

8.130.2.1 `crsAmm()`

```
torch::Tensor CANDY::OnlineIVFLSHIndex::crsAmm (
    torch::Tensor & A,
    torch::Tensor & B,
    torch::Tensor & indices ) [static]
```

thw column row sampling to compute approximate matrix multiplication

Parameters

<i>A</i>	the left side matrix
<i>B</i>	the right side matrix
<i>idx</i>	the indices of sampling
<i>_crsDim</i>	the dimension of preserved dimensions

8.130.2.2 deleteRowsInline()

```
bool CANDY::OnlineIVFLSHIndex::deleteRowsInline (
    torch::Tensor & t ) [protected]
```

the inline function of deleting rows

Parameters

<i>t</i>	the tensor, multiple rows
----------	---------------------------

Returns

bool whether the deleting is successful

8.130.2.3 deleteTensor()

```
bool CANDY::OnlineIVFLSHIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.130.2.4 insertTensor()

```
bool CANDY::OnlineIVFLSHIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.130.2.5 reviseTensor()

```
bool CANDY::OnlineIVFLSHIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.130.2.6 searchTensor()

```
std::vector< torch::Tensor > CANDY::OnlineIVFLSHIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.130.2.7 setConfig()

```
bool CANDY::OnlineIVFLSHIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

generate the rotation matrix for random projection

Reimplemented from [CANDY::AbstractIndex](#).

Reimplemented in [CANDY::OnlineIVFL2HIndex](#).

The documentation for this class was generated from the following files:

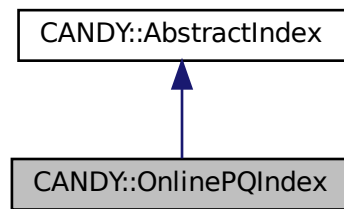
- include/CANDY/OnlineIVFLSHIndex.h
- src/CANDY/[OnlineIVFLSHIndex.cpp](#)

8.131 CANDY::OnlinePQIndex Class Reference

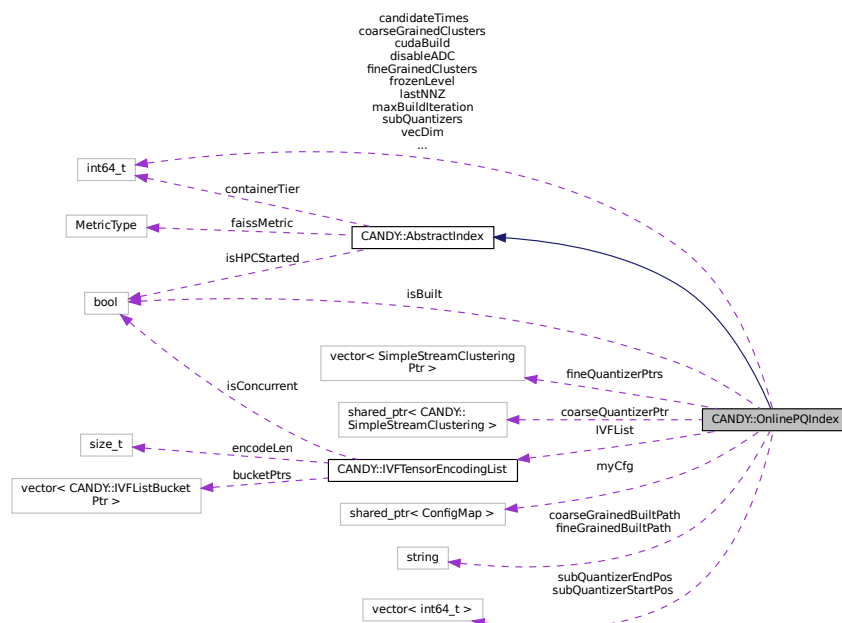
The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers.

```
#include <CANDY/OnlinePQIndex.h>
```

Inheritance diagram for CANDY::OnlinePQIndex:



Collaboration diagram for CANDY::OnlinePQIndex:



Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)

- *delete a tensor*
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
- *revise a tensor*
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
- *search the k-NN of a query tensor, return the result tensors*
- virtual bool [offlineBuild](#) (torch::Tensor &t)
- *offline build phase*
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
- *set the frozen level of online updating internal state*

Protected Member Functions

- bool [tryLoadQuantizers](#) (void)
- std::vector< int64_t > [coarseGrainedEncode](#) (torch::Tensor &t, torch::Tensor *residential)
- std::vector< std::vector< uint8_t > > [fineGrainedEncode](#) (torch::Tensor &residential)
- bool [deleteRowsInline](#) (torch::Tensor &t)
- *the inline function of deleting rows*

Protected Attributes

- [INTELLI::ConfigMapPtr](#) **myCfg** = nullptr
- int64_t **lastNNZ** = 0
- int64_t **vecDim** = 0
- int64_t **coarseGrainedClusters** = 4096
- int64_t **subQuantizers** = 8
- int64_t **fineGrainedClusters** = 256
- int64_t **cudaBuild** = 0
- int64_t **maxBuildIteration** = 1000
- int64_t **candidateTimes** = 1
- int64_t **disableADC** = 0
- bool **isBuilt** = false
- std::string **coarseGrainedBuiltPath**
- std::string **fineGrainedBuiltPath**
- [SimpleStreamClusteringPtr](#) **coarseQuantizerPtr**
- std::vector< [SimpleStreamClusteringPtr](#) > **fineQuantizerPtrs**
- std::vector< int64_t > **subQuantizerStartPos**
- std::vector< int64_t > **subQuantizerEndPos**
- int64_t **frozenLevel** = 0
- [IVFTensorEncodingList](#) **IVFList**

Additional Inherited Members

8.131.1 Detailed Description

The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers.

Note

currently single thread

config parameters

- vecDim, the dimension of vectors, default 768, l64
- coarseGrainedClusters, the number of coarse-grained clusters, default 4096, l64
- fineGrainedClusters, the number of fine-grained clusters in each sub quantizer, default 256, 1~256 l64
- subQuantizers, the number of sub quantizers, default 8, l64
- coarseGrainedBuiltPath, the path of built coarse grained centroids, default "OnlinePQIndex_coarse.rbt", String
- fineGrainedBuiltPath, the path of built fine grained centroids, default "OnlinePQIndex_fine.tbt", String
- cudaBuild, whether using cuda in building phase, default 0, l64
- maxBuildIteration, the maxium iterations of building, default 1000, l64
- candidateTimes, the times of k to determine minimum candidates, default 1 ,l64
- disableADC, set this to 1 will disable ADC or residential computing and go back to IVFPQ, default 0 (means IVFADC mode), l64

8.131.2 Member Function Documentation**8.131.2.1 deleteRowsInline()**

```
bool CANDY::OnlinePQIndex::deleteRowsInline (
    torch::Tensor & t ) [protected]
```

the inline function of deleting rows

Parameters

<i>t</i>	the tensor, multiple rows
----------	---------------------------

Returns

bool whether the deleting is successful

8.131.2.2 deleteTensor()

```
bool CANDY::OnlinePQIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.3 fineGrainedEncode()

```
std::vector< std::vector< uint8_t > > CANDY::OnlinePQIndex::fineGrainedEncode (
    torch::Tensor & residential ) [protected]
```

1. get the output of each subquantizer

8.131.2.4 insertTensor()

```
bool CANDY::OnlinePQIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.5 loadInitialTensor()

```
bool CANDY::OnlinePQIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and is different from [insertTensor](#) for this one:

- The frozen level is forced to be 0 since the data is initial data
- Will try to build clusters from scratch if they are not successfully loaded

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.6 offlineBuild()

```
bool CANDY::OnlinePQIndex::offlineBuild (
    torch::Tensor & t ) [virtual]
```

offline build phase

Note

In this index, call offlineBuild will do the following'

- Build cluster centroids of both coarse grained and fine grained quantizers from t
- Save the centroids to raw binary tensor files, the names are as specified in 'coarseGrainedBuiltPath' and 'fineGrainedBuiltPath'

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

1. build the coarse-grained clusters
2. calculate the residential
3. build sub quantizers

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.7 reviseTensor()

```
bool CANDY::OnlinePQIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.8 searchTensor()

```
std::vector< torch::Tensor > CANDY::OnlinePQIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.9 setConfig()

```
bool CANDY::OnlinePQIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

create cluster instances

Reimplemented from [CANDY::AbstractIndex](#).

8.131.2.10 setFrozenLevel()

```
bool CANDY::OnlinePQIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Note

the frozen levels

- 0 frozen everything
- 1 frozen fine-grained clusters
- 2 frozen coarse-grained clusters
- ≥ 3 frozen nothing

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

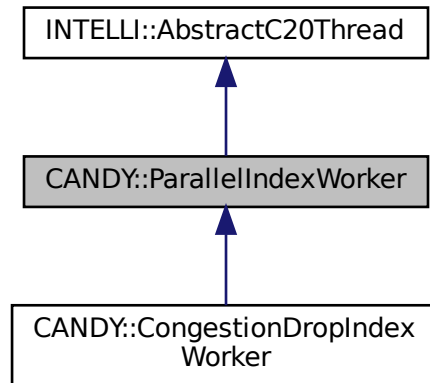
- include/CANDY/[OnlinePQIndex.h](#)
- src/CANDY/[OnlinePQIndex.cpp](#)

8.132 CANDY::ParallelIndexWorker Class Reference

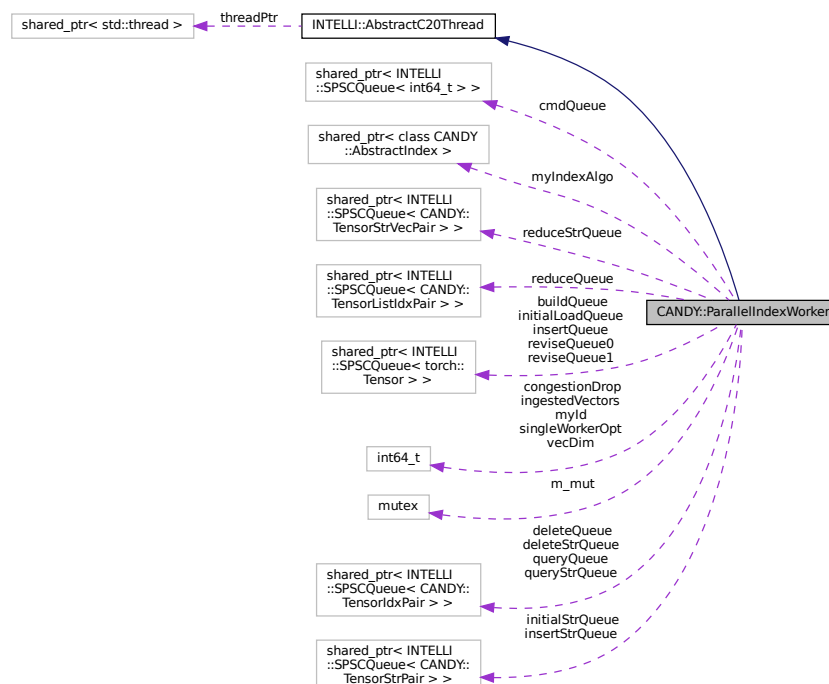
A worker class of parallel index thread.

```
#include <CANDY/ParallelPartitionIndex/ParallelIndexWorker.h>
```

Inheritance diagram for CANDY::ParallelIndexWorker:



Collaboration diagram for CANDY::ParallelIndexWorker:



Public Member Functions

- virtual void **setReduceQueue** (TensorListIdxQueuePtr rq)
- virtual void **setReduceStrQueue** (TensorStrVecQueuePtr rq)
- virtual void **setId** (int64_t _id)
- virtual bool **waitPendingOperations** ()
- virtual bool **loadInitialTensor** (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void **reset** ()
reset this index to initied status
- virtual bool **setConfig** (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool **startHPC** ()
some extra set-ups if the index has HPC fetures
- virtual bool **insertTensor** (torch::Tensor &t)
insert a tensor
- virtual bool **deleteTensor** (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool **reviseTensor** (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< faiss::idx_t > **searchIndex** (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > **getTensorByIndex** (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor **rawData** ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > **searchTensor** (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool **endHPC** ()
some extra termination if the index has HPC fetures
- virtual bool **setFrozenLevel** (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool **offlineBuild** (torch::Tensor &t)
offline build phase
- virtual void **pushSearch** (torch::Tensor q, int64_t k)
- virtual void **pushSearchStr** (torch::Tensor q, int64_t k)
- virtual bool **loadInitialStringObject** (torch::Tensor &t, std::vector< std::string > &strs)
load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)
- virtual bool **insertStringObject** (torch::Tensor &t, std::vector< std::string > &strs)
insert a string object
- virtual bool **deleteStringObject** (torch::Tensor &t, int64_t k=1)
delete tensor along with its corresponding string object
- virtual std::vector< std::vector< std::string > > **searchStringObject** (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects
- virtual std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > **searchTensorAndStringObject** (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects and original tensors

Public Attributes

- TensorListIdxQueuePtr **reduceQueue**
- TensorStrVecQueuePtr **reduceStrQueue**

Protected Member Functions

- virtual void [inlineMain](#) ()

The inline 'main' function of thread, as an interface.

Protected Attributes

- TensorQueuePtr **insertQueue**
- TensorQueuePtr **reviseQueue0**
- TensorQueuePtr **reviseQueue1**
- TensorQueuePtr **buildQueue**
- TensorQueuePtr **initialLoadQueue**
- TensorIdxQueuePtr **deleteQueue**
- TensorIdxQueuePtr **queryQueue**
- TensorIdxQueuePtr **deleteStrQueue**
- TensorStrQueuePtr **initialStrQueue**
- TensorStrQueuePtr **insertStrQueue**
- TensorIdxQueuePtr **queryStrQueue**
- CmdQueuePtr **cmdQueue**
- int64_t **myId** = 0
- int64_t **vecDim** = 0
- int64_t **congestionDrop** = 1
- int64_t **ingestedVectors** = 0
- int64_t **singleWorkerOpt**
- std::mutex **m_mut**
- [AbstractIndexPtr](#) **myIndexAlgo** = nullptr

8.132.1 Detailed Description

A worker class of parallel index thread.

Note

Concurrency policy is strictly read after write
special parameters

- parallelWorker_algoTag The algo tag of this worker, String, default flat
- parallelWorker_queueSize The input queue size of this worker, l64, default 10
- vecDim the dimension of vectors, l674, default 768
- congestionDrop, whether or not drop the data when congestion occurs, l64, default 0

The documentation for this class was generated from the following files:

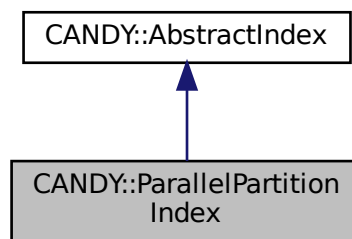
- include/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.h](#)
- src/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.cpp](#)

8.133 CANDY::ParallelPartitionIndex Class Reference

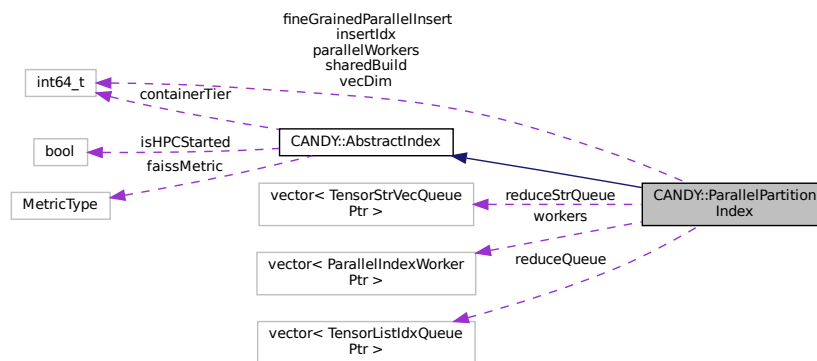
A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature.

```
#include <CANDY/ParallelPartitionIndex.h>
```

Inheritance diagram for CANDY::ParallelPartitionIndex:



Collaboration diagram for CANDY::ParallelPartitionIndex:



Public Member Functions

- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor

- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual bool [startHPC](#) ()
some extra set-ups if the index has HPC fetures
- virtual bool [endHPC](#) ()
some extra termination if the index has HPC fetures
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state
- virtual bool [offlineBuild](#) (torch::Tensor &t)
offline build phase
- virtual bool [waitPendingOperations](#) ()
a busy waitting for all pending operations to be done
- virtual bool [loadInitialStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)
- virtual bool [insertStringObject](#) (torch::Tensor &t, std::vector< std::string > &strs)
insert a string object
- virtual bool [deleteStringObject](#) (torch::Tensor &t, int64_t k=1)
delete tensor along with its corresponding string object
- virtual std::vector< std::vector< std::string > > [searchStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects
- virtual std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > [searchTensorAndStringObject](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the linked string objects and original tensors

Public Attributes

- std::vector< TensorListIdxQueuePtr > **reduceQueue**
- std::vector< TensorStrVecQueuePtr > **reduceStrQueue**

Protected Member Functions

- void [insertTensorInline](#) (torch::Tensor &t)
- void [partitionBuildInLine](#) (torch::Tensor &t)
- void [partitionLoadInLine](#) (torch::Tensor &t)
- void [insertStringInline](#) (torch::Tensor &t, std::vector< string > &s)
- void [partitionLoadStringInLine](#) (torch::Tensor &t, std::vector< string > &s)

Protected Attributes

- int64_t **parallelWorkers**
- int64_t **insertIdx**
- std::vector< [ParallelIndexWorkerPtr](#) > **workers**
- int64_t **vecDim**
- int64_t **fineGrainedParallelInsert**
- int64_t **sharedBuild**

8.133.1 Detailed Description

A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature.

Note

Concurrency policy is strictly read after write

Warning

Don't mix the usage of tensor-only I/O and tensor-string hybrid I/O in one indexing class
remember to call `starHPC` and `endHPC`

Note

special parameters

- `parallelWorker_algoTag` The algo tag of this worker, String, default flat
- `parallelWorker_queueSize` The input queue size of this worker, l64, default 10
- `parallelWorkers` The number of paraller workers, l64, default 1 (set this to less than 0 will use max hardware_concurrency);
- `vecDim`, the dimension of vectors, default 768, l64
- `fineGrainedParallelInsert`, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, l64, default 0
- `sharedBuild` whether let all sharding using shared build, 1, l64
- `congestionDrop`, whether or not drop the data when congestion occurs, l64, default 0 @warning Make sure you are using 2D tensors!

8.133.2 Member Function Documentation

8.133.2.1 deleteStringObject()

```
bool CANDY::ParallelPartitionIndex::deleteStringObject (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete tensor along with its corresponding string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the delet is successful

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.2 deleteTensor()

```
bool CANDY::ParallelPartitionIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor

Parameters

<i>t</i>	the tensor, some index needs to be single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.3 endHPC()

```
bool CANDY::ParallelPartitionIndex::endHPC ( ) [virtual]
```

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.4 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::ParallelPartitionIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.5 insertStringObject()

```
bool CANDY::ParallelPartitionIndex::insertStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

insert a string object

Note

This is majorly an online function

Parameters

<i>t</i>	the tensor, some index need to be single row
<i>strs</i>	the corresponding list of strings

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.6 insertTensor()

```
bool CANDY::ParallelPartitionIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.7 loadInitialStringObject()

```
bool CANDY::ParallelPartitionIndex::loadInitialStringObject (
    torch::Tensor & t,
    std::vector< std::string > & strs ) [virtual]
```

load the initial tensors of a data base along with its string objects, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row •
<i>strs</i>	the corresponding list of strings

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.8 loadInitialTensor()

```
bool CANDY::ParallelPartitionIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and may be different from [insertTensor](#) for some indexes

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.9 offlineBuild()

```
bool CANDY::ParallelPartitionIndex::offlineBuild (  
    torch::Tensor & t ) [virtual]
```

offline build phase

Parameters

<i>t</i>	the tensor for offline build
----------	------------------------------

Returns

whether the building is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.10 rawData()

```
torch::Tensor CANDY::ParallelPartitionIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.11 reviseTensor()

```
bool CANDY::ParallelPartitionIndex::reviseTensor (  
    torch::Tensor & t,  
    torch::Tensor & w ) [virtual]
```

revise a tensor

Parameters

<i>t</i>	the tensor to be revised
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

only allow to delete and insert, no straightforward revision

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.12 searchStringObject()

```
std::vector< std::vector< std::string > > CANDY::ParallelPartitionIndex::searchStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<std::vector<std::string>> the result object for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.13 searchTensor()

```
std::vector< torch::Tensor > CANDY::ParallelPartitionIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.14 searchTensorAndStringObject()

```
std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > CANDY↔
::ParallelPartitionIndex::searchTensorAndStringObject (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the linked string objects and original tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::tuple<std::vector<torch::Tensor>,std::vector<std::vector<std::string>>>`

1. broadcast the query
2. prepare to collect
3. reduce

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.15 setConfig()

```
bool CANDY::ParallelPartitionIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.16 setFrozenLevel()

```
bool CANDY::ParallelPartitionIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.17 startHPC()

```
bool CANDY::ParallelPartitionIndex::startHPC ( ) [virtual]
```

some extra set-ups if the index has HPC fetures

Returns

bool whether the HPC set-up is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.133.2.18 waitPendingOperations()

```
bool CANDY::ParallelPartitionIndex::waitPendingOperations ( ) [virtual]
```

a busy waiting for all pending operations to be done

Note

in this index, there are may be some un-commited write due to the parallel queues

Returns

bool, whether the waiting is actually done;

Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

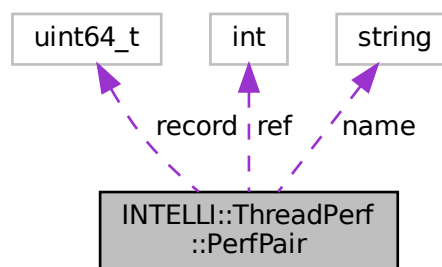
- include/CANDY/[ParallelPartitionIndex.h](#)
- src/CANDY/[ParallelPartitionIndex.cpp](#)

8.134 INTELLI::ThreadPerf::PerfPair Class Reference

a record pair of perf events

```
#include <Utils/ThreadPerf.hpp>
```

Collaboration diagram for INTELLI::ThreadPerf::PerfPair:

**Public Member Functions**

- **PerfPair** (int _ref, std::string _name)

Public Attributes

- int **ref**
- std::string **name**
- uint64_t **record**

8.134.1 Detailed Description

a record pair of perf events

The documentation for this class was generated from the following file:

- include/Utils/[ThreadPerf.hpp](#)

8.135 INTELLI::ThreadPerf::PerfTool Class Reference

Public Member Functions

- **PerfTool** (pid_t pid, int cpu)
- uint64_t **readPerf** (size_t ch)
- int **startPerf** (size_t ch)
- int **stopPerf** (size_t ch)
- bool **isValidChannel** (size_t ch)

The documentation for this class was generated from the following file:

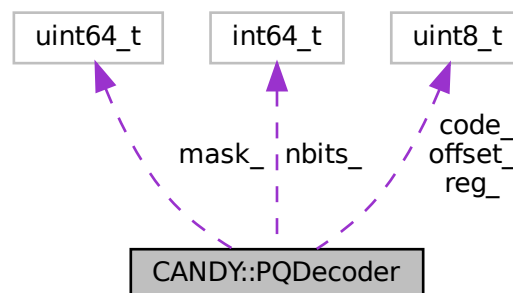
- include/Utils/[ThreadPerf.hpp](#)

8.136 CANDY::PQDecoder Class Reference

class for decoding from codes, approximated assignment of centroids, to centroids indices

```
#include <CANDY/PQIndex.h>
```

Collaboration diagram for CANDY::PQDecoder:



Public Member Functions

- **PQDecoder** (const uint8_t *code, int64_t nbits)
- uint64_t **decode** ()
decode from codes to the actual index of a centroid in sub-vector

Public Attributes

- const uint8_t * **code_**
- uint8_t **offset_**
- const int64_t **nbits_**
- const uint64_t **mask_**
- uint8_t **reg_**

8.136.1 Detailed Description

class for decoding from codes, approximated assignment of centroids, to centroids indices

8.136.2 Member Function Documentation

8.136.2.1 decode()

```
uint64_t CANDY::PQDecoder::decode ( ) [inline]
```

decode from codes to the actual index of a centroid in sub-vector

Returns

the centroid assignment

The documentation for this class was generated from the following file:

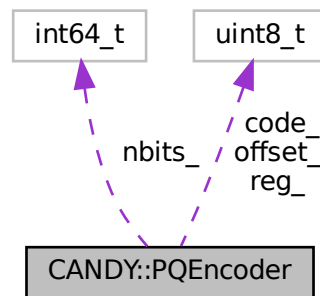
- include/CANDY/PQIndex.h

8.137 CANDY::PQEncoder Class Reference

class for encoding input vectors to codes, standing for approximated assignment of centroids

```
#include <CANDY/PQIndex.h>
```

Collaboration diagram for CANDY::PQEncoder:



Public Member Functions

- **PQEncoder** (uint8_t *code, int64_t nbits, uint8_t offset)
- void **encode** (uint64_t x)
encode assignment x to code

Public Attributes

- uint8_t * **code_**
- int64_t **nbits_**
number of bits per subquantizer index
- uint8_t **offset_**
- uint8_t **reg_**

8.137.1 Detailed Description

class for encoding input vectors to codes, standing for approximated assignment of centroids

8.137.2 Member Function Documentation

8.137.2.1 encode()

```
void CANDY::PQEncoder::encode (
    uint64_t x ) [inline]
```

encode assignment x to code

Parameters

x	centroids assignment of a vector for its part of sub-vector
---	---

The documentation for this class was generated from the following file:

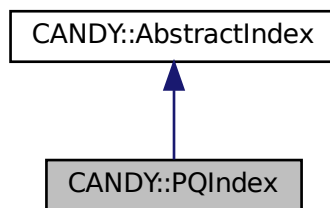
- include/CANDY/PQIndex.h

8.138 CANDY::PQIndex Class Reference

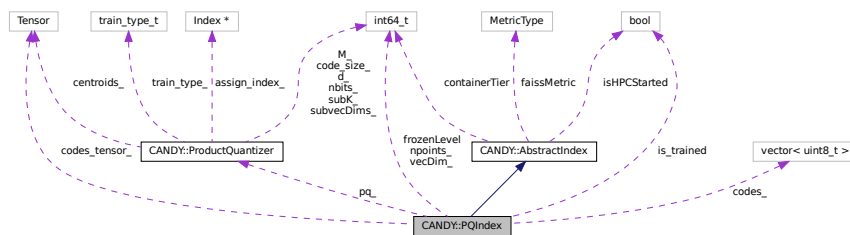
class for indexing vectors using product quantizations, this is a raw implementation without hierachical

```
#include <CANDY/PQIndex.h>
```

Inheritance diagram for CANDY::PQIndex:



Collaboration diagram for CANDY::PQIndex:



Public Member Functions

- virtual void [reset](#) ()
reset this index to initied status
- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor. In [PQIndex](#) setting it requires to re-train on new data
- virtual bool [loadInitialTensor](#) (torch::Tensor &t)
load the initial tensors of a data base, use this BEFORE [insertTensor](#)
- virtual bool [deleteTensor](#) (torch::Tensor &t, int64_t k=1)
delete a tensor. In [PQIndex](#) setting it requires to re-train on new data
- virtual bool [reviseTensor](#) (torch::Tensor &t, torch::Tensor &w)
revise a tensor. In [PQIndex](#) setting it requires to re-train on new data
- virtual std::vector< faiss::idx_t > [searchIndex](#) (torch::Tensor q, int64_t k)
search the k-NN of a query tensor, return their index
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors
- virtual std::vector< torch::Tensor > [getTensorByIndex](#) (std::vector< faiss::idx_t > &idx, int64_t k)
return a vector of tensors according to some index
- virtual torch::Tensor [rawData](#) ()
return the rawData of tensor
- virtual bool [setFrozenLevel](#) (int64_t frozenLv)
set the frozen level of online updating internal state

Protected Member Functions

- void [add](#) (int64_t nx, torch::Tensor x)
add a batch of vectors into [PQIndex](#) which would serve as the base to modify
- void [train](#) (int64_t nx, torch::Tensor x)
train the [PQIndex](#) upon input vectors. Should be called after [add\(\)](#)

Protected Attributes

- [ProductQuantizer](#) **pq_**
- std::vector< uint8_t > **codes_**
*encoded dataset npoints_ * pq_.code_size_*
- torch::Tensor **codes_tensor_**
- int64_t **npoints_** = 0
- int64_t **vecDim_** = 0
- bool **is_trained** = false
- int64_t **frozenLevel** = 0

Additional Inherited Members

8.138.1 Detailed Description

class for indexing vectors using product quantizations, this is a raw implementation without hierachical

Todo delete and revise a tensor may not be feasible for [PQIndex](#)

- [deleteTensor](#)
- [reviseTensor](#)

encode and decode may be verbose for both code tensor and code pointers

- [searchTensor](#)
- [insertTensor](#)

Note

config parameters

- vecDim, the dimension of vectors, default 768, l64
- subQuantizers, the number of sub quantizers, default 8, l64
- nBits, the number of bits in each sub quantizer, default 8, l64

8.138.2 Member Function Documentation

8.138.2.1 add()

```
void CANDY::PQIndex::add (
    int64_t nx,
    torch::Tensor x ) [protected]
```

add a batch of vectors into [PQIndex](#) which would serve as the base to modify

Parameters

<i>nx</i>	number of input x vectors
<i>x</i>	input vectors as tensors

8.138.2.2 deleteTensor()

```
bool CANDY::PQIndex::deleteTensor (
    torch::Tensor & t,
    int64_t k = 1 ) [virtual]
```

delete a tensor. In [PQIndex](#) setting it requires to re-train on new data

Parameters

<i>t</i>	the tensor, recommend single row
<i>k</i>	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.3 getTensorByIndex()

```
std::vector< torch::Tensor > CANDY::PQIndex::getTensorByIndex (
    std::vector< faiss::idx_t > & idx,
    int64_t k ) [virtual]
```

return a vector of tensors according to some index

Parameters

<i>idx</i>	the index, follow faiss's style, allow the KNN index of multiple queries
<i>k</i>	the returned neighbors, i.e., will be the number of rows of each returned tensor

Returns

a vector of tensors, each tensor represent KNN results of one query in idx

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.4 insertTensor()

```
bool CANDY::PQIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor. In [PQIndex](#) setting it requires to re-train on new data

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.5 loadInitialTensor()

```
bool CANDY::PQIndex::loadInitialTensor (
    torch::Tensor & t ) [virtual]
```

load the initial tensors of a data base, use this BEFORE [insertTensor](#)

Note

This is majorly an offline function, and is different from [insertTensor](#) for this one:

- Will firstly try to build clusters from scratch using t

Parameters

<i>t</i>	the tensor, some index need to be single row
----------	--

Returns

bool whether the loading is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.6 rawData()

```
torch::Tensor CANDY::PQIndex::rawData ( ) [virtual]
```

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.7 reviseTensor()

```
bool CANDY::PQIndex::reviseTensor (
    torch::Tensor & t,
    torch::Tensor & w ) [virtual]
```

revise a tensor. In [PQIndex](#) setting it requires to re-train on new data

Parameters

<i>t</i>	the tensor to be revised, recommend single row
<i>w</i>	the revised value

Returns

bool whether the revising is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.8 searchIndex()

```
std::vector< faiss::idx_t > CANDY::PQIndex::searchIndex (
    torch::Tensor q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return their index

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<faiss::idx_t> the index, follow faiss's order

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.9 searchTensor()

```
std::vector< torch::Tensor > CANDY::PQIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

`std::vector<torch::Tensor>` the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.10 setConfig()

```
bool CANDY::PQIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

`bool` whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.11 setFrozenLevel()

```
bool CANDY::PQIndex::setFrozenLevel (
    int64_t frozenLv ) [virtual]
```

set the frozen level of online updating internal state

Parameters

<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state
-----------------	---

Note

the frozen levels

- 0 frozen everything
- ≥ 1 frozen nothing

Returns

whether the setting is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.138.2.12 train()

```
void CANDY::PQIndex::train (
    int64_t nx,
    torch::Tensor x ) [protected]
```

train the [PQIndex](#) upon input vectors. Should be called after [add\(\)](#)

Parameters

<i>nx</i>	number of input x vectors
<i>x</i>	input vectors as tensors

The documentation for this class was generated from the following files:

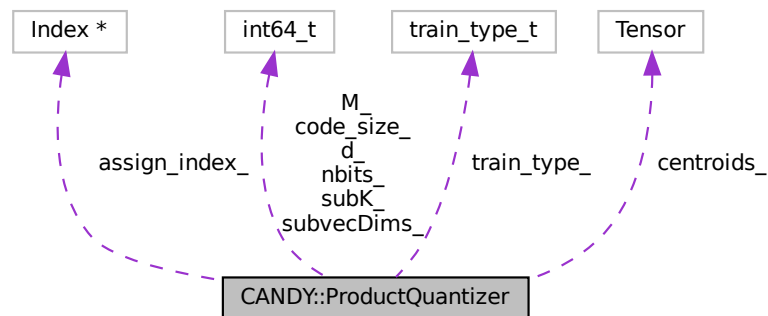
- include/CANDY/PQIndex.h
- src/CANDY/PQIndex.cpp

8.139 CANDY::ProductQuantizer Class Reference

class for basic product quantization operations on input of tensors

```
#include <CANDY/PQIndex.h>
```

Collaboration diagram for CANDY::ProductQuantizer:

**Public Types**

- enum **train_type_t** { **Train_default** , **Train_shared** }

Public Member Functions

- void [setCentroidsFrom](#) ([Clustering](#) cls) const
Set centroids from trained clustering.
- void [setCentroidsFrom](#) ([Clustering](#) cls, int64_t M) const
Set centroids[M] for the Mth subquantizer from trained clustering.
- void [train](#) (int64_t n, torch::Tensor t)
- void [search](#) (const torch::Tensor x, int64_t nx, const uint8_t *codes, const int64_t ncodes, faiss::float_↵ maxheap_array_t *res, bool init_finalize_heap)
- void [add](#) (int64_t nx, const torch::Tensor x)
add vectors to current PQ Index, which would append codes and drift the centroids according to input
- void [compute_code](#) (const float *x, uint8_t *code) const
compute a single vector to codes
- void [compute_codes](#) (const float *x, uint8_t *codes, int64_t n) const
compute vectors to codes
- void [decode](#) (const torch::Tensor code, torch::Tensor *x) const
decode from codes
- void [compute_distance_table](#) (const torch::Tensor x, torch::Tensor *dis_table, int64_t nx) const
*Compute the distance between single x vector and M*subK centroids.*
- void [compute_distance_tables](#) (const torch::Tensor x, torch::Tensor *dis_table, int64_t nx) const
*Compute the distance between nx vectors and M*subK centroids.*
- **ProductQuantizer** (int64_t d, int64_t M, int64_t nbits)

Public Attributes

- int64_t [d_](#)
total dim;
- int64_t [M_](#)
number of subquantizers
- int64_t [nbits_](#)
number of bits per quantization index
- int64_t [subvecDims_](#)
dimensionality of each subvector
- int64_t [subK_](#)
number of centroids of each subquantizer
- int64_t [code_size_](#)
- train_type_t [train_type_](#) = Train_default
- faiss::Index * [assign_index_](#)
- torch::Tensor [centroids_](#)
(M, subK_, subvecDims_)

8.139.1 Detailed Description

class for basic product quantization operations on input of tensors

8.139.2 Member Function Documentation

8.139.2.1 add()

```
void CANDY::ProductQuantizer::add (
    int64_t nx,
    const torch::Tensor x )
```

add vectors to current PQ Index, which would append codes and drift the centroids according to input

Parameters

<i>nx</i>	number of input vectors
<i>x</i>	input vectors to be added

8.139.2.2 compute_code()

```
void CANDY::ProductQuantizer::compute_code (
    const float * x,
    uint8_t * code ) const
```

compute a single vector to codes

Parameters

<i>x</i>	vectors to be encoded
<i>codes</i>	destination codes

8.139.2.3 compute_codes()

```
void CANDY::ProductQuantizer::compute_codes (
    const float * x,
    uint8_t * codes,
    int64_t n ) const
```

compute vectors to codes

Parameters

<i>x</i>	input vectors
<i>codes</i>	store computation results from x, pointer target is a torch::Tensor size of (nx, code_size_);
<i>nx</i>	number of input vectors
<i>start</i>	pointers denoting where it starts

8.139.2.4 compute_distance_table()

```
void CANDY::ProductQuantizer::compute_distance_table (
    const torch::Tensor x,
    torch::Tensor * dis_table,
    int64_t nx ) const
```

Compute the distance between single x vector and M*subK centroids.

Parameters

<i>x</i>	
<i>dis_table</i>	

Returns

distance table tensor size of M*subK

8.139.2.5 compute_distance_tables()

```
void CANDY::ProductQuantizer::compute_distance_tables (
    const torch::Tensor x,
    torch::Tensor * dis_table,
    int64_t nx ) const
```

Compute the distance between nx vectors and M*subK centroids.

Parameters

<i>x</i>	tensor of nx * d
<i>dis_table</i>	output table sizeof nx*M*subK
<i>nx</i>	number of vectors

8.139.2.6 decode()

```
void CANDY::ProductQuantizer::decode (
    const torch::Tensor code,
    torch::Tensor * x ) const
```

decode from codes

Parameters

<i>code</i>	codes to be decoded
<i>x</i>	destination vectors

8.139.2.7 search()

```
void CANDY::ProductQuantizer::search (
    const torch::Tensor x,
    int64_t nx,
    const uint8_t * codes,
    const int64_t ncodes,
    faiss::float_maxheap_array_t * res,
    bool init_finalize_heap )
```

Parameters

<i>x</i>	vectors to be searched
<i>nx</i>	number of input vectors
<i>codes</i>	codes to be consulted during search
<i>ncodes</i>	number of codes
<i>res</i>	result heap
<i>init_finalize_heap</i>	whether at the end of searching each vector to reorder the heap

8.139.2.8 setCentroidsFrom() [1/2]

```
void CANDY::ProductQuantizer::setCentroidsFrom (
    Clustering cls ) const
```

Set centroids from trained clustering.

Parameters

<i>cls</i>	Clustering
------------	------------

8.139.2.9 setCentroidsFrom() [2/2]

```
void CANDY::ProductQuantizer::setCentroidsFrom (
    Clustering cls,
    int64_t M ) const
```

Set centroids[M] for the Mth subquantizer from trained clustering.

Parameters

<i>cls</i>	Clustering
<i>M</i>	subquantizer identifier

8.139.2.10 train()

```
void CANDY::ProductQuantizer::train (
    int64_t n,
    torch::Tensor t )
```

train PQ with given tensor

Parameters

<i>n</i>	number of inputs
<i>t</i>	input vectors as tensor

The documentation for this class was generated from the following files:

- include/CANDY/PQIndex.h
- src/CANDY/PQIndex.cpp

8.140 CANDY::FLANN::RandomCenterChooser Class Reference

The class used in hierarchical kmeans tree to choose center.

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

Public Member Functions

- **RandomCenterChooser** (torch::Tensor *p, int64_t v)
- void **operator()** (int64_t k, int64_t *indices, int64_t indices_length, int64_t *centers, int64_t ¢ers_length)

8.140.1 Detailed Description

The class used in hierarchical kmeans tree to choose center.

The documentation for this class was generated from the following file:

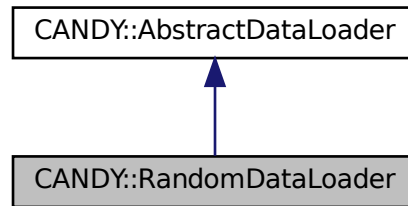
- include/CANDY/FlannIndex/FlannUtils.h

8.141 CANDY::RandomDataLoader Class Reference

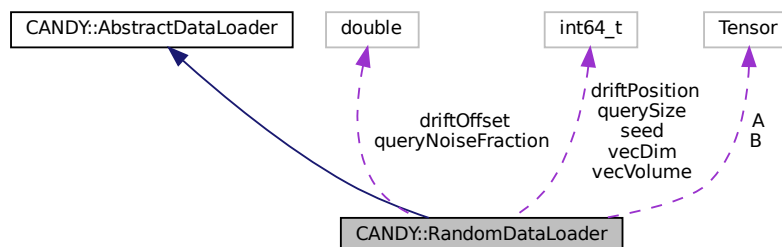
The class of ranom data loader,.

```
#include <DataLoader/RandomDataLoader.h>
```

Inheritance diagram for CANDY::RandomDataLoader:



Collaboration diagram for CANDY::RandomDataLoader:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
Set the GLOBAL config map related to this loader.
- virtual [torch::Tensor](#) [getData](#) ()
get the data tensor
- virtual [torch::Tensor](#) [getQuery](#) ()
get the query tensor

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- int64_t **vecDim**
- int64_t **vecVolume**
- int64_t **querySize**
- int64_t **seed**
- int64_t **driftPosition**
- double **driftOffset**
- double **queryNoiseFraction**

8.141.1 Detailed Description

The class of ranom data loader,.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

parameters of config

- vecDim, the dimension of vectors, default 768, l64
- vecVolume, the volume of vectors, default 1000, l64
- driftPosition, the position of starting some 'concept drift', default 0 (no drift), l64
 - driftOffset, the offset value of concept drift, default 0.5, Double
 - queryNoiseFraction, the fraction of noise in query, default 0, allow 0~1, Double
- querySize, the size of query, default 10, l64
- seed, the random seed, default 7758258, l64

: default name tags "random": [RandomDataLoader](#)

8.141.2 Member Function Documentation

8.141.2.1 `getData()`

```
torch::Tensor CANDY::RandomDataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.141.2.2 `getQuery()`

```
torch::Tensor CANDY::RandomDataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.141.2.3 `setConfig()`

```
bool CANDY::RandomDataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

The documentation for this class was generated from the following files:

- include/DataLoader/[RandomDataLoader.h](#)
- src/DataLoader/RandomDataLoader.cpp

8.142 CANDY::Rania_Python Class Reference

The python bounding functions.

```
#include <CANDYPYTHON.h>
```

Public Member Functions

- torch::Tensor [index_create](#) (string name, string type)
The c++ bindings to creat an index at backend.
- torch::Tensor [index_loadCfgFromFile](#) (string name, string fname)
The c++ bindings to load the config map related to a specific index from file.
- torch::Tensor [index_editCfgDouble](#) (string name, string key, double value)
The c++ bindings to change the config map related to a specific index.
- torch::Tensor [index_editCfgStr](#) (string name, string key, string value)
The c++ bindings to change the config map related to a specific index.
- torch::Tensor [index_editCfgI64](#) (string name, string key, int64_t value)
The c++ bindings to change the config map related to a specific index.
- torch::Tensor [index_init](#) (string name)
The c++ bindings to init an index with its bounded config.
- torch::Tensor [index_insert](#) (string name, torch::Tensor t)
The c++ bindings to insert tensor to an index.
- std::vector< torch::Tensor > [index_search](#) (string name, torch::Tensor t, int64_t k)
The c++ bindings to search tensor.
- torch::Tensor [index_delete](#) (string name, torch::Tensor t, int64_t k)
The c++ bindings to delete tensor to an index.
- torch::Tensor [index_revise](#) (string name, torch::Tensor t, torch::Tensor &w)
The c++ bindings to revise tensor to an index.
- torch::Tensor [index_rawData](#) (string name)
The c++ bindings to return rawData.
- torch::Tensor [index_reset](#) (string name)
The c++ bindings to creat an index at backend.
- torch::Tensor [index_startHPC](#) (string name)
The c++ bindings to start HPC features.
- torch::Tensor [index_endHPC](#) (string name)
The c++ bindings to end HPC features.
- torch::Tensor [tensorToFile](#) (torch::Tensor A, std::string fname)
The c++ bindings to save a tensor into file.
- torch::Tensor [tensorFromFile](#) (std::string fname)
The c++ bindings to load a tensor from file.
- torch::Tensor [index_setFrozenLevel](#) (string name, int64_t frozenLV)
The c++ bindings to set the frozen level of online updating internal state.
- torch::Tensor [index_offlineBuild](#) (string name, torch::Tensor t)
The c++ bindings to offlineBuild.
- torch::Tensor [index_loadInitial](#) (string name, torch::Tensor t)
The c++ bindings to load initial tensor.
- torch::Tensor [index_waitPending](#) (string name)
The c++ bindings to wait pending operations features.
- torch::Tensor [dataLoader_create](#) (string name, string type)
The c++ bindings to creat an dataLoader at backend.
- torch::Tensor [dataLoader_editCfgDouble](#) (string name, string key, double value)
The c++ bindings to change the config map related to a specific dataLoader.
- torch::Tensor [dataLoader_editCfgFloat](#) (string name, string key, float value)
The c++ bindings to change the config map related to a specific dataLoader.
- torch::Tensor [dataLoader_editCfgStr](#) (string name, string key, string value)
The c++ bindings to change the config map related to a specific dataLoader.
- torch::Tensor [dataLoader_editCfgI64](#) (string name, string key, int64_t value)

The c++ bindings to change the config map related to a specific dataLoader.

- torch::Tensor [dataLoader_init](#) (string name)

The c++ bindings to init an dataLoader with its bounded config.

- torch::Tensor [dataLoader_getData](#) (string name)

The c++ bindings to get data tensor from the specified data loader.

- torch::Tensor [dataLoader_getQuery](#) (string name)

The c++ bindings to get query tensor from the specified data loader.

- torch::Tensor [tensorFromFVECS](#) (string name)

The c++ bindings to load tensor from fvecs file.

- torch::Tensor [tensorFromHDF5](#) (string name, string attr)

The c++ bindings to load tensor from HDF5 file.

- torch::Tensor [index_loadInitialString](#) (string name, torch::Tensor t, std::vector< std::string > s)

The c++ bindings to load initial tensor along with string objects.

- torch::Tensor [index_insertString](#) (string name, torch::Tensor t, std::vector< std::string > s)

The c++ bindings to insert tensor to an index with its binded strings.

- torch::Tensor [index_deleteString](#) (string name, torch::Tensor t, int64_t k)

The c++ bindings to delete tensor to an index and its string object.

- std::vector< std::vector< std::string > > [index_searchString](#) (string name, torch::Tensor &q, int64_t k)

The c++ bindings to search binded string of given tensor.

- std::tuple< std::vector< torch::Tensor >, std::vector< std::vector< std::string > > > [index_searchTensorAndStringList](#) (string name, torch::Tensor &q, int64_t k)

The c++ bindings to search tensor and binded string of given tensor.

8.142.1 Detailed Description

The python bounding functions.

Note

- Please first run torch.ops.load_library("<the path of CANDY's library>")
- In this simple bounding, we just access CANDY index class and its configuration by name tag, there is some c++ hash table in the backend to do this
- Please add the prefix "torch.ops.CANDY." when calling the following fucntions, see also benchmark/python↵ Test.py

8.142.2 Member Function Documentation

8.142.2.1 dataLoader_create()

```
torch::Tensor CANDY::Rania_Python::dataLoader_create (
    string name,
    string type ) [inline]
```

The c++ bindings to creat an dataLoader at backend.

Parameters

<i>name</i>	the name of this dataLoader
<i>type</i>	the type of this dataLoader, keep the same as that in CANDY::IndexTable

Returns

tensor 1x1, [1] for success

8.142.2.2 dataLoader_editCfgDouble()

```
torch::Tensor CANDY::Rania_Python::dataLoader_editCfgDouble (
    string name,
    string key,
    double value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

<i>name</i>	the name of the dataLoader
<i>key</i>	the key in the cfg
<i>value</i>	the double value

Returns

tensor 1x1, [1] for success

8.142.2.3 dataLoader_editCfgFloat()

```
torch::Tensor CANDY::Rania_Python::dataLoader_editCfgFloat (
    string name,
    string key,
    float value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

<i>name</i>	the name of the dataLoader
<i>key</i>	the key in the cfg
<i>value</i>	the float value

Returns

tensor 1x1, [1] for success

8.142.2.4 dataLoader_editCfgI64()

```
torch::Tensor CANDY::Rania_Python::dataLoader_editCfgI64 (
    string name,
    string key,
    int64_t value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

<i>name</i>	the name of the dataLoader
<i>key</i>	the key in the cfg
<i>value</i>	the I64 value

Returns

tensor 1x1, [1] for success

8.142.2.5 dataLoader_editCfgStr()

```
torch::Tensor CANDY::Rania_Python::dataLoader_editCfgStr (
    string name,
    string key,
    string value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

<i>name</i>	the name of the dataLoader
<i>key</i>	the key in the cfg
<i>value</i>	the string value

Returns

tensor 1x1, [1] for success

8.142.2.6 `dataLoader_getData()`

```
torch::Tensor CANDY::Rania_Python::dataLoader_getData (
    string name ) [inline]
```

The c++ bindings to get data tensor from the specified data loader.

Parameters

<i>name</i>	the name of the dataLoader
<i>t</i>	the tensor

Returns

tensor 1x1, [1] for success

8.142.2.7 `dataLoader_getQuery()`

```
torch::Tensor CANDY::Rania_Python::dataLoader_getQuery (
    string name ) [inline]
```

The c++ bindings to get query tensor from the specified data loader.

Parameters

<i>name</i>	the name of the dataLoader
-------------	----------------------------

Returns

the first result tensor

8.142.2.8 `dataLoader_init()`

```
torch::Tensor CANDY::Rania_Python::dataLoader_init (
    string name ) [inline]
```

The c++ bindings to init an dataLoader with its bounded config.

Parameters

<i>name</i>	the name of the dataLoader
-------------	----------------------------

Returns

tensor 1x1, [1] for success

8.142.2.9 index_create()

```
torch::Tensor CANDY::Rania_Python::index_create (
    string name,
    string type ) [inline]
```

The c++ bindings to creat an index at backend.

Parameters

<i>name</i>	the name of this index
<i>type</i>	the type of this index, keep the same as that in CANDY::IndexTable

Returns

tensor 1x1, [1] for success

8.142.2.10 index_delete()

```
torch::Tensor CANDY::Rania_Python::index_delete (
    string name,
    torch::Tensor t,
    int64_t k ) [inline]
```

The c++ bindings to delete tensor to an index.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>k</i>	the NNS

Returns

tensor 1x1, [1] for success

8.142.2.11 index_deleteString()

```
torch::Tensor CANDY::Rania_Python::index_deleteString (
    string name,
```

```
torch::Tensor t,
int64_t k ) [inline]
```

The c++ bindings to delete tensor to an index and its string object.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>k</i>	the NNS

Returns

tensor 1x1, [1] for success

8.142.2.12 index_editCfgDouble()

```
torch::Tensor CANDY::Rania_Python::index_editCfgDouble (
    string name,
    string key,
    double value ) [inline]
```

The c++ bindings to change the config map related to a specific index.

Parameters

<i>name</i>	the name of the index
<i>key</i>	the key in the cfg
<i>value</i>	the double value

Returns

tensor 1x1, [1] for success

8.142.2.13 index_editCfgI64()

```
torch::Tensor CANDY::Rania_Python::index_editCfgI64 (
    string name,
    string key,
    int64_t value ) [inline]
```

The c++ bindings to change the config map related to a specific index.

Parameters

<i>name</i>	the name of the index
<i>key</i>	the key in the cfg
<i>value</i>	the I64 value

Returns

tensor 1x1, [1] for success

8.142.2.14 index_editCfgStr()

```
torch::Tensor CANDY::Rania_Python::index_editCfgStr (
    string name,
    string key,
    string value ) [inline]
```

The c++ bindings to change the config map related to a specific index.

Parameters

<i>name</i>	the name of the index
<i>key</i>	the key in the cfg
<i>value</i>	the string value

Returns

tensor 1x1, [1] for success

8.142.2.15 index_endHPC()

```
torch::Tensor CANDY::Rania_Python::index_endHPC (
    string name ) [inline]
```

The c++ bindings to end HPC features.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor 1x1, [1] for success

8.142.2.16 index_init()

```
torch::Tensor CANDY::Rania_Python::index_init (
    string name ) [inline]
```

The c++ bindings to init an index with its bounded config.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor 1x1, [1] for success

8.142.2.17 index_insert()

```
torch::Tensor CANDY::Rania_Python::index_insert (
    string name,
    torch::Tensor t ) [inline]
```

The c++ bindings to insert tensor to an index.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor

Returns

tensor 1x1, [1] for success

8.142.2.18 index_insertString()

```
torch::Tensor CANDY::Rania_Python::index_insertString (
    string name,
    torch::Tensor t,
    std::vector< std::string > s ) [inline]
```

The c++ bindings to insert tensor to an index with its binded strings.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>s</i>	the vector of string, List[str] in python

Returns

tensor 1x1, [1] for success

8.142.2.19 index_loadCfgFromFile()

```
torch::Tensor CANDY::Rania_Python::index_loadCfgFromFile (
    string name,
    string fname ) [inline]
```

The c++ bindings to load the config map related to a specific index from file.

Parameters

<i>name</i>	the name of the index
<i>fname</i>	the name of file

Returns

tensor 1x1, [1] for success

8.142.2.20 index_loadInitial()

```
torch::Tensor CANDY::Rania_Python::index_loadInitial (
    string name,
    torch::Tensor t ) [inline]
```

The c++ bindings to load initial tensor.

Note

This is majorly an offline function, and may be different from [index_insert](#) for some indexes

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor

Returns

tensor 1x1, [1] for success

8.142.2.21 index_loadInitialString()

```
torch::Tensor CANDY::Rania_Python::index_loadInitialString (
    string name,
    torch::Tensor t,
    std::vector< std::string > s ) [inline]
```

The c++ bindings to load initial tensor along with string objects.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor

Returns

tensor 1x1, [1] for success

8.142.2.22 index_offlineBuild()

```
torch::Tensor CANDY::Rania_Python::index_offlineBuild (  
    string name,  
    torch::Tensor t ) [inline]
```

The c++ bindings to offlineBuild.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor

Returns

tensor 1x1, [1] for success

8.142.2.23 index_rawData()

```
torch::Tensor CANDY::Rania_Python::index_rawData (  
    string name ) [inline]
```

The c++ bindings to return rawData.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor of rawData

8.142.2.24 index_reset()

```
torch::Tensor CANDY::Rania_Python::index_reset (
    string name ) [inline]
```

The c++ bindings to creat an index at backend.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor 1x1, [1] for success

8.142.2.25 index_revise()

```
torch::Tensor CANDY::Rania_Python::index_revise (
    string name,
    torch::Tensor t,
    torch::Tensor & w ) [inline]
```

The c++ bindings to revise tensor to an index.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor to be revised
<i>w</i>	the revision

Returns

tensor 1x1, [1] for success

8.142.2.26 index_search()

```
std::vector<torch::Tensor> CANDY::Rania_Python::index_search (
    string name,
    torch::Tensor t,
    int64_t k ) [inline]
```

The c++ bindings to search tensor.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>k</i>	the NNS

Returns

the list of result tensors

8.142.2.27 index_searchString()

```
std::vector<std::vector<std::string> > CANDY::Rania_Python::index_searchString (
    string name,
    torch::Tensor & q,
    int64_t k ) [inline]
```

The c++ bindings to search binded string of given tensor.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>k</i>	the NNS

Returns

List[List[str]], for each rows

8.142.2.28 index_searchTensorAndStringList()

```
std::tuple<std::vector<torch::Tensor>, std::vector<std::vector<std::string> > > CANDY::↵
Rania_Python::index_searchTensorAndStringList (
    string name,
    torch::Tensor & q,
    int64_t k ) [inline]
```

The c++ bindings to search tensor and binded string of given tensor.

Parameters

<i>name</i>	the name of the index
<i>t</i>	the tensor
<i>k</i>	the NNS

Returns

[List[Tensor],List[List[str]]], for each rows

8.142.2.29 index_setFrozenLevel()

```
torch::Tensor CANDY::Rania_Python::index_setFrozenLevel (
    string name,
    int64_t frozenLv ) [inline]
```

The c++ bindings to set the frozen level of online updating internal state.

Parameters

<i>name</i>	the name of the index
<i>frozenLv</i>	the level of frozen, 0 means freeze any online update in internal state

Returns

tensor 1x1, [1] for success

8.142.2.30 index_startHPC()

```
torch::Tensor CANDY::Rania_Python::index_startHPC (
    string name ) [inline]
```

The c++ bindings to start HPC features.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor 1x1, [1] for success

8.142.2.31 index_waitPending()

```
torch::Tensor CANDY::Rania_Python::index_waitPending (
    string name ) [inline]
```

The c++ bindings to wait pending operations features.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

tensor 1x1, [1] for success

8.142.2.32 tensorFromFile()

```
torch::Tensor CANDY::Rania_Python::tensorFromFile (
    std::string fname ) [inline]
```

The c++ bindings to load a tensor from file.

Parameters

<i>name</i>	the name of the index
-------------	-----------------------

Returns

the tensor result

8.142.2.33 tensorFromFVECS()

```
torch::Tensor CANDY::Rania_Python::tensorFromFVECS (
    string name ) [inline]
```

The c++ bindings to load tensor from fvecs file.

Parameters

<i>name</i>	the name of file return the result tensor
-------------	---

8.142.2.34 tensorFromHDF5()

```
torch::Tensor CANDY::Rania_Python::tensorFromHDF5 (
    string name,
    string attr ) [inline]
```

The c++ bindings to load tensor from HDF5 file.

Parameters

<i>name</i>	the name of file
<i>attr</i>	the attribute return the result tensor

8.142.2.35 tensorToFile()

```
torch::Tensor CANDY::Rania_Python::tensorToFile (
    torch::Tensor A,
    std::string fname ) [inline]
```

The c++ bindings to save a tensor into file.

Parameters

<i>A</i>	the tensor
<i>name</i>	the name of the index

Returns

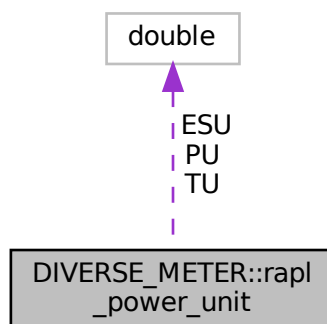
tensor 1x1, [1] for success

The documentation for this class was generated from the following file:

- include/CANDYPYTHON.h

8.143 DIVERSE_METER::rapl_power_unit Struct Reference

Collaboration diagram for DIVERSE_METER::rapl_power_unit:



Public Attributes

- double **PU**
- double **ESU**
- double **TU**

The documentation for this struct was generated from the following file:

- include/Utils/Meters/IntelMeter/IntelMeter.hpp

8.144 CANDY::FLANN::ResultSet Class Reference

a priority queue used in [FlannIndex](#)

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

Public Member Functions

- **ResultSet** (int64_t capacity)
- size_t **size** () const
- bool **isFull** ()
- float **worstDist** ()
- void **add** (float dist, int64_t index)
- void **copy** (int64_t *indices, float *dists, int64_t q, int64_t num_elements)

8.144.1 Detailed Description

a priority queue used in [FlannIndex](#)

The documentation for this class was generated from the following file:

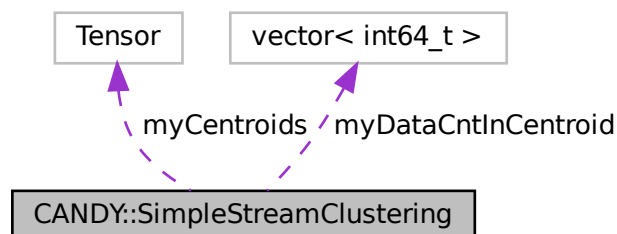
- include/CANDY/FlannIndex/FlannUtils.h

8.145 CANDY::SimpleStreamClustering Class Reference

a simple class for stream clustering, following online PQ style and using simple linear equations

```
#include <CANDY/OnlinePQIndex/SimpleStreamClustering.h>
```

Collaboration diagram for CANDY::SimpleStreamClustering:



Public Member Functions

- bool [buildCentroids](#) (torch::Tensor &trainSet, int64_t k, int64_t maxIterations, DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#), bool usingCuda=true)
to build the centroids from trainset
- torch::Tensor [exportCentroids](#) (void)
export the inside tensor of centroids to outside
- bool [saveCentroidsToFile](#) (std::string fname)
save the centroids to file
- bool [loadCentroids](#) (torch::Tensor &externCentroid)
to load centroids from external tensor
- bool [loadCentroids](#) (std::string fname)
to load centroids from external file
- int64_t [classifySingleRow](#) (torch::Tensor &rowTensor, DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
classify a single row of tensor
- std::vector< int64_t > [classifyMultiRow](#) (torch::Tensor &rowTensor, DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
classify multi rows of tensor
- bool [addSingleRow](#) (torch::Tensor &rowTensor, int64_t frozenLevel=0, UpdateFunction_t insertFunc=[SimpleStreamClustering::euclideanInsert](#), DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
add a single row of tensor
- bool [addSingleRowWithIdx](#) (torch::Tensor &rowTensor, int64_t clusterIdx, int64_t frozenLevel=0, UpdateFunction_t insertFunc=[SimpleStreamClustering::euclideanInsert](#), DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
add a single row of tensor, with specifying its cluster index
- bool [deleteSingleRow](#) (torch::Tensor &rowTensor, int64_t frozenLevel=0, UpdateFunction_t deleteFunc=[SimpleStreamClustering::euclideanDelete](#), DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
delete a single row of tensor
- bool [deleteSingleRowWithIdx](#) (torch::Tensor &rowTensor, int64_t clusterIdx, int64_t frozenLevel=0, UpdateFunction_t deleteFunc=[SimpleStreamClustering::euclideanDelete](#), DistanceFunction_t distanceFunc=[SimpleStreamClustering::euclideanDistance](#))
delete a single row of tensor, with specifying its cluster index

Static Public Member Functions

- static torch::Tensor [euclideanDistance](#) (const torch::Tensor &a, const torch::Tensor &b)
the default euclidean distance function
- static void [euclideanInsert](#) (const torch::Tensor *a, torch::Tensor *b, const int64_t c)
the default euclidean insert function
- static void [euclideanDelete](#) (const torch::Tensor *a, torch::Tensor *b, const int64_t c)
the default euclidean delete function

Protected Attributes

- torch::Tensor **myCentroids**
- std::vector< int64_t > **myDataCntInCentroid**

8.145.1 Detailed Description

a simple class for stream clustering, following online PQ style and using simple linear equations

Todo two functions are extremely slow and costly, needs to be re-implemented

- [buildCentroids](#)
- [classifyMultiRow](#)

8.145.2 Member Function Documentation

8.145.2.1 addSingleRow()

```
bool CANDY::SimpleStreamClustering::addSingleRow (
    torch::Tensor & rowTensor,
    int64_t frozenLevel = 0,
    CANDY::UpdateFunction_t insertFunc = SimpleStreamClustering::euclideanInsert,
    DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance )
```

add a single row of tensor

Parameters

<i>rowTensor</i>	the 1*D row tensor
<i>insertFunc</i>	the insert function
<i>frozenLevel</i>	the level of frozen, 0 means freeze any online update in internal state @ param distanceFunc the distance function

Returns

whether the add is successful

8.145.2.2 addSingleRowWithIdx()

```
bool CANDY::SimpleStreamClustering::addSingleRowWithIdx (
    torch::Tensor & rowTensor,
    int64_t clusterIdx,
    int64_t frozenLevel = 0,
    CANDY::UpdateFunction_t insertFunc = SimpleStreamClustering::euclideanInsert,
    CANDY::DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance
)
```

add a single row of tensor, with specifying its cluster index

Parameters

<i>rowTensor</i>	the 1*D row tensor
<i>clusterIdx</i>	the cluster index
<i>insertFunc</i>	the insert function
<i>frozenLevel</i>	the level of frozen, 0 means freeze any online update in internal state @ param distanceFunc the distance function

Returns

whether the add is successful

8.145.2.3 buildCentroids()

```
bool CANDY::SimpleStreamClustering::buildCentroids (
    torch::Tensor & trainSet,
    int64_t k,
    int64_t maxIterations,
    DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance,
    bool usingCuda = true )
```

to build the centroids from trainset

Parameters

<i>trainSet</i>	the trainset, N*D
<i>k</i>	the number of centroids
<i>maxIterations</i>	the max iterations for setting up clusters
<i>distanceFunc</i>	the distance function
<i>usingCuda</i>	whether or not using cuda

Returns

whether the build is successful

8.145.2.4 classifyMultiRow()

```
std::vector< int64_t > CANDY::SimpleStreamClustering::classifyMultiRow (
    torch::Tensor & rowTensor,
    DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance )
```

classify multi rows of tensor

Parameters

<i>rowsTensor</i>	the N*D row tensor
<i>distanceFunc</i>	the distance function

Returns

the idx of cluster it belongs to

8.145.2.5 classifySingleRow()

```
int64_t CANDY::SimpleStreamClustering::classifySingleRow (
    torch::Tensor & rowTensor,
    DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance )
```

classify a single row of tensor

Parameters

<i>rowTensor</i>	the 1*D row tensor
<i>distanceFunc</i>	the distance function

Returns

the idx of cluster it belongs to

8.145.2.6 deleteSingleRow()

```
bool CANDY::SimpleStreamClustering::deleteSingleRow (
    torch::Tensor & rowTensor,
    int64_t frozenLevel = 0,
    CANDY::UpdateFunction_t deleteFunc = SimpleStreamClustering::euclideanDelete,
    DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance )
```

delete a single row of tensor

Parameters

<i>rowTensor</i>	the 1*D row tensor
<i>deleteFunc</i>	the delete function
<i>frozenLevel</i>	the level of frozen, 0 means freeze any online update in internal state @ param distanceFunc the distance function

Returns

whether the delete is successful

8.145.2.7 deleteSingleRowWithIdx()

```
bool CANDY::SimpleStreamClustering::deleteSingleRowWithIdx (
    torch::Tensor & rowTensor,
    int64_t clusterIdx,
    int64_t frozenLevel = 0,
    CANDY::UpdateFunction_t deleteFunc = SimpleStreamClustering::euclideanInsert,
    CANDY::DistanceFunction_t distanceFunc = SimpleStreamClustering::euclideanDistance
)
```

delete a single row of tensor, with specifying its cluster index

Parameters

<i>rowTensor</i>	the 1*D row tensor
<i>clusterIdx</i>	the cluster index
<i>deleteFunc</i>	the delete function
<i>frozenLevel</i>	the level of frozen, 0 means freeze any online update in internal state @ param distanceFunc the distance function

Returns

whether the deletion is successful

8.145.2.8 euclideanDelete()

```
static void CANDY::SimpleStreamClustering::euclideanDelete (
    const torch::Tensor * a,
    torch::Tensor * b,
    const int64_t c ) [inline], [static]
```

the default euclidean delete function

Parameters

<i>a</i>	the data tensor
<i>b</i>	the centroids
<i>c</i>	the number of points in this centroid

Returns

the aligned distance tensor

8.145.2.9 euclideanDistance()

```
static torch::Tensor CANDY::SimpleStreamClustering::euclideanDistance (
    const torch::Tensor & a,
    const torch::Tensor & b ) [inline], [static]
```

the default euclidean distance function

Parameters

<i>a</i>	the data tensor
<i>b</i>	the centroids

Returns

the aligned distance tensor

8.145.2.10 euclideanInsert()

```
static void CANDY::SimpleStreamClustering::euclideanInsert (
    const torch::Tensor * a,
    torch::Tensor * b,
    const int64_t c ) [inline], [static]
```

the default euclidean insert function

Parameters

<i>a</i>	the data tensor
<i>b</i>	the centroids
<i>c</i>	the number of points in this centroid

Returns

the aligned distance tensor

8.145.2.11 exportCentroids()

```
torch::Tensor CANDY::SimpleStreamClustering::exportCentroids (
    void ) [inline]
```

export the inside tensor of centroids to outside

Returns

the myCentroids tensor

8.145.2.12 loadCentroids() [1/2]

```
bool CANDY::SimpleStreamClustering::loadCentroids (
    std::string fname )
```

to load centroids from external file

Parameters

<i>fname</i>	the file name of external tensor
--------------	----------------------------------

Returns

whether the load is successful

8.145.2.13 loadCentroids() [2/2]

```
bool CANDY::SimpleStreamClustering::loadCentroids (
    torch::Tensor & externCentroid )
```

to load centroids from external tensor

Parameters

<i>externCentroid</i>	the external tensor of centroid
-----------------------	---------------------------------

Returns

whether the load is successful

8.145.2.14 saveCentroidsToFile()

```
bool CANDY::SimpleStreamClustering::saveCentroidsToFile (
    std::string fname ) [inline]
```

save the centroids to file

Parameters

<i>fname</i>	the name of file
--------------	------------------

Returns

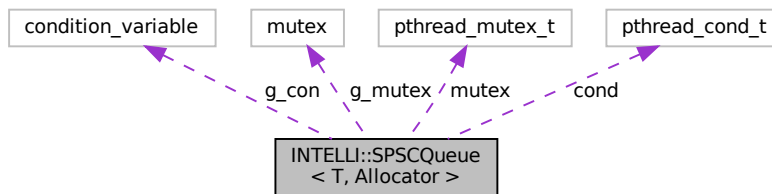
whether the saving is successful

The documentation for this class was generated from the following files:

- include/CANDY/OnlinePQIndex/[SimpleStreamClustering.h](#)
- src/CANDY/OnlinePQIndex/SimpleStreamClustering.cpp

8.146 INTELLI::SPSCQueue< T, Allocator > Class Template Reference

Collaboration diagram for INTELLI::SPSCQueue< T, Allocator >:



Public Member Functions

- **SPSCQueue** (const size_t capacity, const Allocator &allocator=Allocator())
- **SPSCQueue** (const [SPSCQueue](#) &)=delete
- **SPSCQueue** & **operator=** (const [SPSCQueue](#) &)=delete
- void **wakeUpSink** (void)
- void **waitForSource** (void)
- template<typename... Args>
void **emplace** (Args &&...args) noexcept(std::is_nothrow_constructible< T, Args &&... >::value)
- template<typename... Args>
bool **try_emplace** (Args &&...args) noexcept(std::is_nothrow_constructible< T, Args &&... >::value)
- void **push** (const T &v) noexcept(std::is_nothrow_copy_constructible< T >::value)
- template<typename P, typename = typename std::enable_if< std::is_constructible<T, P &&>::value>::type>
void **push** (P &&v) noexcept(std::is_nothrow_constructible< T, P && >::value)
- bool **try_push** (const T &v) noexcept(std::is_nothrow_copy_constructible< T >::value)
- template<typename P, typename = typename std::enable_if< std::is_constructible<T, P &&>::value>::type>
bool **try_push** (P &&v) noexcept(std::is_nothrow_constructible< T, P && >::value)
- T * **front** () noexcept
- void **pop** () noexcept
- size_t **size** () const noexcept
- bool **empty** () const noexcept
- size_t **capacity** () const noexcept

Public Attributes

- pthread_cond_t **cond**
- pthread_mutex_t **mutex**
- std::mutex **g_mutex**
- condition_variable **g_con**

The documentation for this class was generated from the following file:

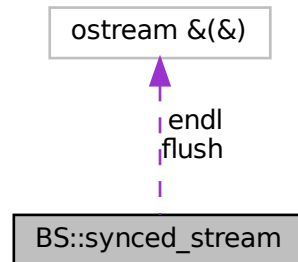
- include/Utils/SPSCQueue.hpp

8.147 BS::synced_stream Class Reference

A helper class to synchronize printing to an output stream by different threads.

```
#include <BS_thread_pool.hpp>
```

Collaboration diagram for BS::synced_stream:



Public Member Functions

- [synced_stream](#) (std::ostream &out_stream_=std::cout)
Construct a new synced stream.
- template<typename... T>
void [print](#) (T &&...items)
Print any number of items into the output stream. Ensures that no other threads print to this stream simultaneously, as long as they all exclusively use the same [synced_stream](#) object to print.
- template<typename... T>
void [println](#) (T &&...items)
Print any number of items into the output stream, followed by a newline character. Ensures that no other threads print to this stream simultaneously, as long as they all exclusively use the same [synced_stream](#) object to print.

Static Public Attributes

- static std::ostream &(&) [endl](#) (std::ostream &)
A stream manipulator to pass to a [synced_stream](#) (an explicit cast of std::endl). Prints a newline character to the stream, and then flushes it. Should only be used if flushing is desired, otherwise ' ' should be used instead.
- static std::ostream &(&) [flush](#) (std::ostream &)
A stream manipulator to pass to a [synced_stream](#) (an explicit cast of std::flush). Used to flush the stream.

8.147.1 Detailed Description

A helper class to synchronize printing to an output stream by different threads.

8.147.2 Constructor & Destructor Documentation

8.147.2.1 `synced_stream()`

```
BS::synced_stream::synced_stream (
    std::ostream & out_stream_ = std::cout ) [inline]
```

Construct a new synced stream.

Parameters

<i>out_↵ stream_</i>	The output stream to print to. The default value is <code>std::cout</code> .
--------------------------	--

8.147.3 Member Function Documentation

8.147.3.1 `print()`

```
template<typename... T>
void BS::synced_stream::print (
    T &&... items ) [inline]
```

Print any number of items into the output stream. Ensures that no other threads print to this stream simultaneously, as long as they all exclusively use the same [synced_stream](#) object to print.

Template Parameters

<i>T</i>	The types of the items
----------	------------------------

Parameters

<i>items</i>	The items to print.
--------------	---------------------

8.147.3.2 `println()`

```
template<typename... T>
void BS::synced_stream::println (
    T &&... items ) [inline]
```

Print any number of items into the output stream, followed by a newline character. Ensures that no other threads print to this stream simultaneously, as long as they all exclusively use the same [synced_stream](#) object to print.

Template Parameters

<i>T</i>	The types of the items
----------	------------------------

Parameters

<i>items</i>	The items to print.
--------------	---------------------

8.147.4 Member Data Documentation

8.147.4.1 endl

```
std::ostream& (&) BS::synced_stream::endl(std::ostream &) [inline], [static]
```

Initial value:

```
=  
static_cast<std::ostream & (&) (std::ostream &)>(std::endl)
```

A stream manipulator to pass to a [synced_stream](#) (an explicit cast of `std::endl`). Prints a newline character to the stream, and then flushes it. Should only be used if flushing is desired, otherwise '`'` should be used instead.

8.147.4.2 flush

```
std::ostream& (&) BS::synced_stream::flush(std::ostream &) [inline], [static]
```

Initial value:

```
=  
static_cast<std::ostream & (&) (std::ostream &)>(std::flush)
```

A stream manipulator to pass to a [synced_stream](#) (an explicit cast of `std::flush`). Used to flush the stream.

The documentation for this class was generated from the following file:

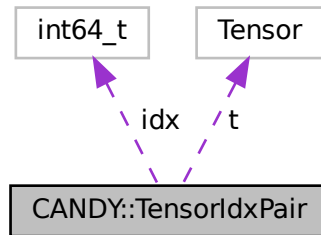
- [include/Utils/BS_thread_pool.hpp](#)

8.148 CANDY::TensorIdxPair Class Reference

The class to define a tensor along with some idx.

```
#include <ParallelIndexWorker.h>
```

Collaboration diagram for CANDY::TensorIdxPair:



Public Member Functions

- **TensorIdxPair** (torch::Tensor _t, int64_t _idx)

Public Attributes

- torch::Tensor **t**
- int64_t **idx**

8.148.1 Detailed Description

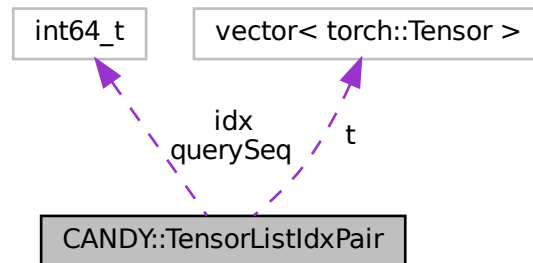
The class to define a tensor along with some idx.

The documentation for this class was generated from the following file:

- include/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.h](#)

8.149 CANDY::TensorListIdxPair Class Reference

Collaboration diagram for CANDY::TensorListIdxPair:



Public Member Functions

- **TensorListIdxPair** (std::vector< torch::Tensor > &t, int64_t_idx, int64_t_seq)

Public Attributes

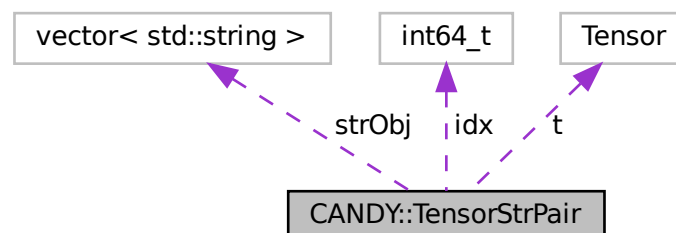
- std::vector< torch::Tensor > **t**
- int64_t **idx**
- int64_t **querySeq**

The documentation for this class was generated from the following file:

- include/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.h](#)

8.150 CANDY::TensorStrPair Class Reference

Collaboration diagram for CANDY::TensorStrPair:



Public Member Functions

- **TensorStrPair** (torch::Tensor _t, int64_t _idx)
- **TensorStrPair** (torch::Tensor _t, int64_t _idx, std::vector< std::string > &str)

Public Attributes

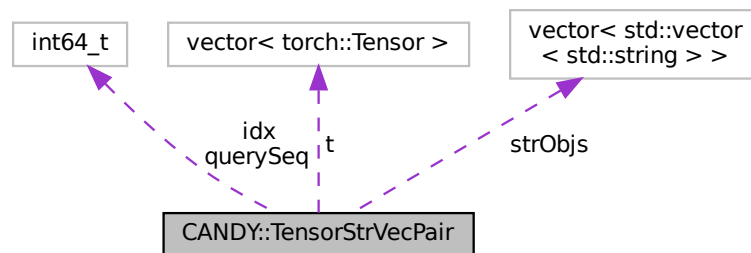
- torch::Tensor **t**
- int64_t **idx**
- std::vector< std::string > **strObj**

The documentation for this class was generated from the following file:

- include/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.h](#)

8.151 CANDY::TensorStrVecPair Class Reference

Collaboration diagram for CANDY::TensorStrVecPair:



Public Member Functions

- **TensorStrVecPair** (std::vector< torch::Tensor > &_t, int64_t _idx, int64_t _seq, std::vector< std::vector< std::string > > str)
- **TensorStrVecPair** (std::vector< torch::Tensor > &_t, int64_t _idx, int64_t _seq)

Public Attributes

- std::vector< torch::Tensor > **t**
- int64_t **idx**
- int64_t **querySeq**
- std::vector< std::vector< std::string > > **strObj**

The documentation for this class was generated from the following file:

- include/CANDY/ParallelPartitionIndex/[ParallelIndexWorker.h](#)

8.152 BS::thread_pool Class Reference

A fast, lightweight, and easy-to-use C++17 thread pool class.

```
#include <BS_thread_pool.hpp>
```

Public Member Functions

- [thread_pool](#) (const [concurrency_t](#) thread_count_=0)
Construct a new thread pool.
- [~thread_pool](#) ()
Destruct the thread pool. Waits for all tasks to complete, then destroys all threads. Note that if the pool is paused, then any tasks still in the queue will never be executed.
- [size_t get_tasks_queued](#) () const
Get the number of tasks currently waiting in the queue to be executed by the threads.
- [size_t get_tasks_running](#) () const
Get the number of tasks currently being executed by the threads.
- [size_t get_tasks_total](#) () const
Get the total number of unfinished tasks: either still in the queue, or running in a thread. Note that [get_tasks_total\(\)](#) == [get_tasks_queued\(\)](#) + [get_tasks_running\(\)](#).
- [concurrency_t get_thread_count](#) () const
Get the number of threads in the pool.
- [bool is_paused](#) () const
Check whether the pool is currently paused.
- [template<typename F, typename T1, typename T2, typename T = std::common_type_t<T1, T2>, typename R = std::invoke_result_t<std::decay_t<F>, T, T>>>](#)
[multi_future](#)< R > [parallelize_loop](#) (const T1 first_index, const T2 index_after_last, F &&loop, const [size_t](#) num_blocks=0)
Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Returns a [multi_future](#) object that contains the futures for all of the blocks.
- [template<typename F, typename T, typename R = std::invoke_result_t<std::decay_t<F>, T, T>>>](#)
[multi_future](#)< R > [parallelize_loop](#) (const T index_after_last, F &&loop, const [size_t](#) num_blocks=0)
Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Returns a [multi_future](#) object that contains the futures for all of the blocks. This overload is used for the special case where the first index is 0.
- [void pause](#) ()
Pause the pool. The workers will temporarily stop retrieving new tasks out of the queue, although any tasks already executed will keep running until they are finished.
- [template<typename F, typename T1, typename T2, typename T = std::common_type_t<T1, T2>>>](#)
[void push_loop](#) (const T1 first_index, const T2 index_after_last, F &&loop, const [size_t](#) num_blocks=0)
Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Does not return a [multi_future](#), so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the loop finishes executing, otherwise bad things will happen.
- [template<typename F, typename T>](#)
[void push_loop](#) (const T index_after_last, F &&loop, const [size_t](#) num_blocks=0)
Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Does not return a [multi_future](#), so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the loop finishes executing, otherwise bad things will happen. This overload is used for the special case where the first index is 0.
- [template<typename F, typename... A>](#)
[void push_task](#) (F &&task, A &&...args)
Push a function with zero or more arguments, but no return value, into the task queue. Does not return a future, so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the task finishes executing, otherwise bad things will happen.

- void `reset` (const `concurrency_t` `thread_count_`=0)
Reset the number of threads in the pool. Waits for all currently running tasks to be completed, then destroys all threads in the pool and creates a new thread pool with the new number of threads. Any tasks that were waiting in the queue before the pool was reset will then be executed by the new threads. If the pool was paused before resetting it, the new pool will be paused as well.
- template<typename F, typename... A, typename R = std::invoke_result_t<std::decay_t<F>, std::decay_t<A>...>>
`std::future< R >` `submit` (F &&task, A &&...args)
Submit a function with zero or more arguments into the task queue. If the function has a return value, get a future for the eventual returned value. If the function has no return value, get an `std::future<void>` which can be used to wait until the task finishes.
- void `unpause` ()
Unpause the pool. The workers will resume retrieving new tasks out of the queue.
- void `wait_for_tasks` ()
Wait for tasks to be completed. Normally, this function waits for all tasks, both those that are currently running in the threads and those that are still waiting in the queue. However, if the pool is paused, this function only waits for the currently running tasks (otherwise it would wait forever). Note: To wait for just one specific task, use `submit()` instead, and call the `wait()` member function of the generated future.

8.152.1 Detailed Description

A fast, lightweight, and easy-to-use C++17 thread pool class.

8.152.2 Constructor & Destructor Documentation

8.152.2.1 `thread_pool()`

```
BS::thread_pool::thread_pool (
    const concurrency_t thread_count_ = 0 ) [inline]
```

Construct a new thread pool.

Parameters

<code>thread_count_ ↔</code>	The number of threads to use. The default value is the total number of hardware threads available, as reported by the implementation. This is usually determined by the number of cores in the CPU. If a core is hyperthreaded, it will count as two threads.
------------------------------	---

8.152.3 Member Function Documentation

8.152.3.1 `get_tasks_queued()`

```
size_t BS::thread_pool::get_tasks_queued ( ) const [inline]
```

Get the number of tasks currently waiting in the queue to be executed by the threads.

Returns

The number of queued tasks.

8.152.3.2 get_tasks_running()

```
size_t BS::thread_pool::get_tasks_running ( ) const [inline]
```

Get the number of tasks currently being executed by the threads.

Returns

The number of running tasks.

8.152.3.3 get_tasks_total()

```
size_t BS::thread_pool::get_tasks_total ( ) const [inline]
```

Get the total number of unfinished tasks: either still in the queue, or running in a thread. Note that `get_tasks_total() == get_tasks_queued() + get_tasks_running()`.

Returns

The total number of tasks.

8.152.3.4 get_thread_count()

```
concurrency_t BS::thread_pool::get_thread_count ( ) const [inline]
```

Get the number of threads in the pool.

Returns

The number of threads.

8.152.3.5 is_paused()

```
bool BS::thread_pool::is_paused ( ) const [inline]
```

Check whether the pool is currently paused.

Returns

true if the pool is paused, false if it is not paused.

8.152.3.6 `parallelize_loop()` [1/2]

```
template<typename F , typename T , typename R = std::invoke_result_t<std::decay_t<F>, T, T>>
multi_future<R> BS::thread_pool::parallelize_loop (
    const T index_after_last,
    F && loop,
    const size_t num_blocks = 0 ) [inline]
```

Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Returns a [multi_future](#) object that contains the futures for all of the blocks. This overload is used for the special case where the first index is 0.

Template Parameters

<i>F</i>	The type of the function to loop through.
<i>T</i>	The type of the loop indices. Should be a signed or unsigned integer.
<i>R</i>	The return value of the loop function F (can be void).

Parameters

<i>index_after_last</i>	The index after the last index in the loop. The loop will iterate from 0 to (index_after_last - 1) inclusive. In other words, it will be equivalent to "for (T i = 0; i < index_after_last; ++i)". Note that if index_after_last == 0, no blocks will be submitted.
<i>loop</i>	The function to loop through. Will be called once per block. Should take exactly two arguments: the first index in the block and the index after the last index in the block. loop(start, end) should typically involve a loop of the form "for (T i = start; i < end; ++i)".
<i>num_blocks</i>	The maximum number of blocks to split the loop into. The default is to use the number of threads in the pool.

Returns

A [multi_future](#) object that can be used to wait for all the blocks to finish. If the loop function returns a value, the [multi_future](#) object can also be used to obtain the values returned by each block.

8.152.3.7 `parallelize_loop()` [2/2]

```
template<typename F , typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>,
typename R = std::invoke_result_t<std::decay_t<F>, T, T>>
multi_future<R> BS::thread_pool::parallelize_loop (
    const T1 first_index,
    const T2 index_after_last,
    F && loop,
    const size_t num_blocks = 0 ) [inline]
```

Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Returns a [multi_future](#) object that contains the futures for all of the blocks.

Template Parameters

<i>F</i>	The type of the function to loop through.
----------	---

Template Parameters

<i>T1</i>	The type of the first index in the loop. Should be a signed or unsigned integer.
<i>T2</i>	The type of the index after the last index in the loop. Should be a signed or unsigned integer. If <i>T1</i> is not the same as <i>T2</i> , a common type will be automatically inferred.
<i>T</i>	The common type of <i>T1</i> and <i>T2</i> .
<i>R</i>	The return value of the loop function <i>F</i> (can be void).

Parameters

<i>first_index</i>	The first index in the loop.
<i>index_after_last</i>	The index after the last index in the loop. The loop will iterate from <i>first_index</i> to (<i>index_after_last</i> - 1) inclusive. In other words, it will be equivalent to "for (T i = <i>first_index</i> ; i < <i>index_after_last</i> ; ++i)". Note that if <i>index_after_last</i> == <i>first_index</i> , no blocks will be submitted.
<i>loop</i>	The function to loop through. Will be called once per block. Should take exactly two arguments: the first index in the block and the index after the last index in the block. <i>loop</i> (start, end) should typically involve a loop of the form "for (T i = start; i < end; ++i)".
<i>num_blocks</i>	The maximum number of blocks to split the loop into. The default is to use the number of threads in the pool.

Returns

A [multi_future](#) object that can be used to wait for all the blocks to finish. If the loop function returns a value, the [multi_future](#) object can also be used to obtain the values returned by each block.

8.152.3.8 push_loop() [1/2]

```
template<typename F , typename T >
void BS::thread_pool::push_loop (
    const T index_after_last,
    F && loop,
    const size_t num_blocks = 0 ) [inline]
```

Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Does not return a [multi_future](#), so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the loop finishes executing, otherwise bad things will happen. This overload is used for the special case where the first index is 0.

Template Parameters

<i>F</i>	The type of the function to loop through.
<i>T</i>	The type of the loop indices. Should be a signed or unsigned integer.

Parameters

<i>index_after_last</i>	The index after the last index in the loop. The loop will iterate from 0 to (<i>index_after_last</i> - 1) inclusive. In other words, it will be equivalent to "for (T i = 0; i < <i>index_after_last</i> ; ++i)". Note that if <i>index_after_last</i> == 0, no blocks will be submitted.
-------------------------	--

Parameters

<i>loop</i>	The function to loop through. Will be called once per block. Should take exactly two arguments: the first index in the block and the index after the last index in the block. loop(start, end) should typically involve a loop of the form "for (T i = start; i < end; ++i)".
<i>num_blocks</i>	The maximum number of blocks to split the loop into. The default is to use the number of threads in the pool.

8.152.3.9 push_loop() [2/2]

```
template<typename F , typename T1 , typename T2 , typename T = std::common_type_t<T1, T2>>
void BS::thread_pool::push_loop (
    const T1 first_index,
    const T2 index_after_last,
    F && loop,
    const size_t num_blocks = 0 ) [inline]
```

Parallelize a loop by automatically splitting it into blocks and submitting each block separately to the queue. Does not return a [multi_future](#), so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the loop finishes executing, otherwise bad things will happen.

Template Parameters

<i>F</i>	The type of the function to loop through.
<i>T1</i>	The type of the first index in the loop. Should be a signed or unsigned integer.
<i>T2</i>	The type of the index after the last index in the loop. Should be a signed or unsigned integer. If T1 is not the same as T2, a common type will be automatically inferred.
<i>T</i>	The common type of T1 and T2.

Parameters

<i>first_index</i>	The first index in the loop.
<i>index_after_last</i>	The index after the last index in the loop. The loop will iterate from first_index to (index_after_last - 1) inclusive. In other words, it will be equivalent to "for (T i = first_index; i < index_after_last; ++i)". Note that if index_after_last == first_index, no blocks will be submitted.
<i>loop</i>	The function to loop through. Will be called once per block. Should take exactly two arguments: the first index in the block and the index after the last index in the block. loop(start, end) should typically involve a loop of the form "for (T i = start; i < end; ++i)".
<i>num_blocks</i>	The maximum number of blocks to split the loop into. The default is to use the number of threads in the pool.

8.152.3.10 push_task()

```
template<typename F , typename... A>
void BS::thread_pool::push_task (
```

```
F && task,
A &&... args ) [inline]
```

Push a function with zero or more arguments, but no return value, into the task queue. Does not return a future, so the user must use [wait_for_tasks\(\)](#) or some other method to ensure that the task finishes executing, otherwise bad things will happen.

Template Parameters

<i>F</i>	The type of the function.
<i>A</i>	The types of the arguments.

Parameters

<i>task</i>	The function to push.
<i>args</i>	The zero or more arguments to pass to the function. Note that if the task is a class member function, the first argument must be a pointer to the object, i.e. &object (or this), followed by the actual arguments.

8.152.3.11 reset()

```
void BS::thread_pool::reset (
    const concurrency_t thread_count_ = 0 ) [inline]
```

Reset the number of threads in the pool. Waits for all currently running tasks to be completed, then destroys all threads in the pool and creates a new thread pool with the new number of threads. Any tasks that were waiting in the queue before the pool was reset will then be executed by the new threads. If the pool was paused before resetting it, the new pool will be paused as well.

Parameters

<i>thread_↵ count_</i>	The number of threads to use. The default value is the total number of hardware threads available, as reported by the implementation. This is usually determined by the number of cores in the CPU. If a core is hyperthreaded, it will count as two threads.
----------------------------	---

8.152.3.12 submit()

```
template<typename F , typename... A, typename R = std::invoke_result_t<std::decay_t<F>,
std::decay_t<A>...>>
std::future<R> BS::thread_pool::submit (
    F && task,
    A &&... args ) [inline]
```

Submit a function with zero or more arguments into the task queue. If the function has a return value, get a future for the eventual returned value. If the function has no return value, get an `std::future<void>` which can be used to wait until the task finishes.

Template Parameters

<i>F</i>	The type of the function.
<i>A</i>	The types of the zero or more arguments to pass to the function.
<i>R</i>	The return type of the function (can be void).

Parameters

<i>task</i>	The function to submit.
<i>args</i>	The zero or more arguments to pass to the function. Note that if the task is a class member function, the first argument must be a pointer to the object, i.e. &object (or this), followed by the actual arguments.

Returns

A future to be used later to wait for the function to finish executing and/or obtain its returned value if it has one.

The documentation for this class was generated from the following file:

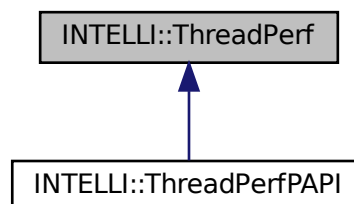
- [include/Utils/BS_thread_pool.hpp](#)

8.153 INTELLI::ThreadPerf Class Reference

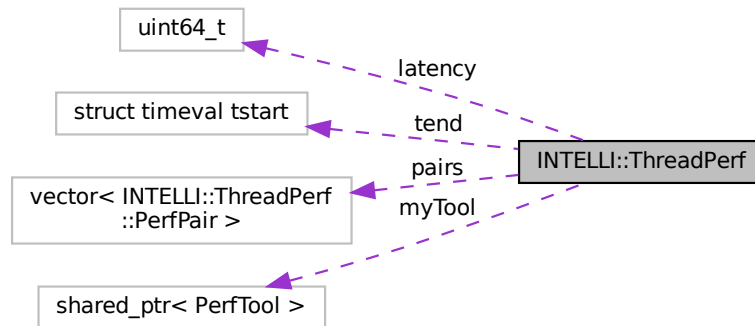
The top entity to provide perf traces, please use this class only UNLESS you know what you are doing.

```
#include <Utils/ThreadPerf.hpp>
```

Inheritance diagram for INTELLI::ThreadPerf:



Collaboration diagram for INTELLI::ThreadPerf:



Classes

- class [PerfPair](#)
a record pair of perf events
- class [PerfTool](#)

Public Member Functions

- [ThreadPerf](#) (int cpu)
To setup this perf to specific cpu.
- virtual void [setPerfList](#) ()
To set up all your interest perf events.
- virtual void [start](#) ()
To start perf tracing.
- virtual void [end](#) ()
To end a perf tracing.
- virtual uint64_t [getResultByld](#) (size_t idx)
Get the perf result by its index of [PerfPair](#).
- virtual uint64_t [getResultByName](#) (string name)
Get the perf result by its name of [PerfPair](#).
- size_t [timeLastUs](#) (struct timeval ts, struct timeval te)
- virtual [ConfigMapPtr](#) [resultToConfigMap](#) ()
convert the perf result into a [ConfigMap](#)
- virtual void [initEventsByCfg](#) ([ConfigMapPtr](#) cfg)
init the perf events according to configmap

Protected Types

- typedef std::shared_ptr< [PerfTool](#) > [PerfToolPtr](#)

Protected Member Functions

- `std::string` **getChValueAsString** (`size_t` idx)

Protected Attributes

- `PerfToolPtr` **myTool**
- `std::vector< PerfPair >` **pairs**
To contain all of your interested perf events.
- `struct timeval` tstart **tend**
- `uint64_t` **latency**

8.153.1 Detailed Description

The top entity to provide perf traces, please use this class only UNLESS you know what you are doing.

Note

You may overwrite the `setPerfList` function for your own interested events

Warning

only works in Linux, and make sure you have opened perf in your kernel and have the access

Note

Requires the [ConfigMap](#) Util

General set up

- create the class
- call [setPerfList](#) or [initEventsByCfg](#), You may overwrite the `setPerfList` function in child classes for your own interested events
- call [start](#)
- run your own process
- call [end](#)
- get the results, by [getResultById](#), [getResultByName](#), or [resultToConfigMap](#)

8.153.2 Constructor & Destructor Documentation

8.153.2.1 ThreadPerf()

```
INTELLI::ThreadPerf::ThreadPerf (
    int cpu ) [inline]
```

To setup this perf to specific cpu.

Parameters

<i>cpu</i>	>=0 for any specific cpu, =-1 for all cpu that may run this process
------------	---

8.153.3 Member Function Documentation

8.153.3.1 getResultById()

```
virtual uint64_t INTELLI::ThreadPerf::getResultById (
    size_t idx ) [inline], [virtual]
```

Get the perf result by its index of [PerfPair](#).

Parameters

<i>idx</i>	The index
------------	-----------

Returns

The value

Reimplemented in [INTELLI::ThreadPerfPAPI](#).

8.153.3.2 getResultByName()

```
virtual uint64_t INTELLI::ThreadPerf::getResultByName (
    string name ) [inline], [virtual]
```

Get the perf result by its name of of [PerfPair](#).

Parameters

<i>idx</i>	The index
------------	-----------

Returns

The value

Reimplemented in [INTELLI::ThreadPerfPAPI](#).

8.153.3.3 `initEventsByCfg()`

```
virtual void INTELLI::ThreadPerf::initEventsByCfg (
    ConfigMapPtr cfg ) [inline], [virtual]
```

init the perf events according to configmap

Parameters

<i>cfg</i>	tyhe configmap
------------	----------------

Reimplemented in [INTELLI::ThreadPerfPAPI](#).

8.153.3.4 `resultToConfigMap()`

```
virtual ConfigMapPtr INTELLI::ThreadPerf::resultToConfigMap ( ) [inline], [virtual]
```

convert the perf result into a [ConfigMap](#)

Returns

The key-value store of configMap, in shared pointer

Note

must stop after calling stop

Reimplemented in [INTELLI::ThreadPerfPAPI](#).

8.153.3.5 `start()`

```
virtual void INTELLI::ThreadPerf::start ( ) [inline], [virtual]
```

To start perf tracing.

Note

call after [setPerfList](#)

Reimplemented in [INTELLI::ThreadPerfPAPI](#).

The documentation for this class was generated from the following file:

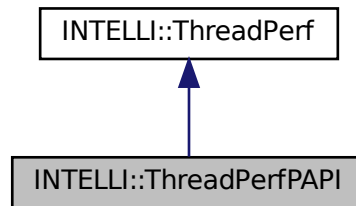
- [include/Utils/ThreadPerf.hpp](#)

8.154 INTELLI::ThreadPerfPAPI Class Reference

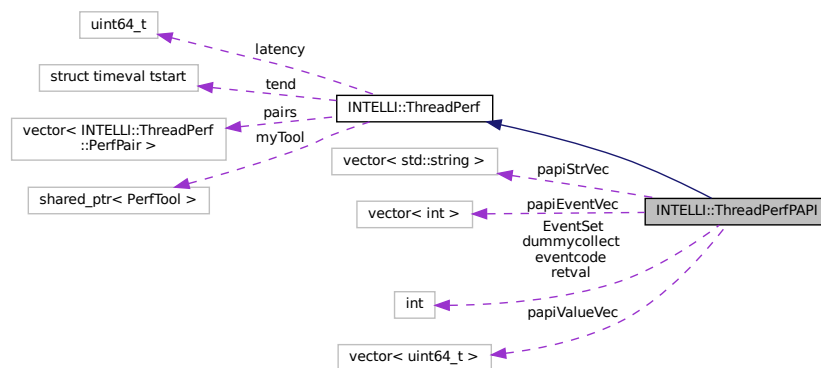
The top entity to provide perf traces by using PAPI lib.

```
#include <Utils/ThreadPerfPAPI.hpp>
```

Inheritance diagram for INTELLI::ThreadPerfPAPI:



Collaboration diagram for INTELLI::ThreadPerfPAPI:



Public Member Functions

- `ThreadPerfPAPI` (int cpu)
To setup this perf to specific cpu.
- void `addPapiTag` (std::string displayTag, int code)
to add a paipi event to be detected
- void `addPapiTag` (std::string displayTag, std::string papiTag)
to add a paipi event to be detected
- virtual void `setPerfList` ()
To set up all your interest perf events.
- virtual void `start` ()
To start perf tracing.

- virtual void `end` ()
To end a perf tracing.
- virtual uint64_t `getResultByld` (size_t idx)
Get the perf result by its index of PerfPair.
- virtual uint64_t `getResultByName` (string name)
Get the perf result by its name of of PerfPair.
- virtual `ConfigMapPtr` `resultToConfigMap` ()
convert the perf result into a ConfigMap
- void `initEventsByCfg` (`ConfigMapPtr` cfg)
init the perf events according to configmap

Protected Member Functions

- void `initPapiLib` ()
- void `clearPapiLib` ()
- void `addPapiEventInline` (int ecode)

Protected Attributes

- std::vector< std::string > `papiStrVec`
- std::vector< uint64_t > `papiValueVec`
- std::vector< int > `papiEventVec`
- int `retval`
- int `EventSet` = PAPI_NULL
- int `dummycollect` = 0
- int `eventcode`

Additional Inherited Members

8.154.1 Detailed Description

The top entity to provide perf traces by using PAPI lib.

Note

You may overwrite the `setPerfList` function for your own interested events

Warning

only works in Linux, and make sure you have opened perf in your kernel and have the access

Note

Requires the [ConfigMap](#) Util
require configs of perf

- perfInstructions, whether or not profile instructions, 1
- perfCycles, to record cpu cycles, 0
- perfMemRead, to record the memory read times, 0
- perfMemWrite, to record the memory write times, 0

General set up

- create the class
- call [initEventsByCfg](#), You may overwrite it function in child classes for your own interested events
- call [start](#)
- run your own process
- call [end](#)
- get the results, by [getResultById](#), [getResultByName](#), or [resultToConfigMap](#)

8.154.2 Constructor & Destructor Documentation**8.154.2.1 ThreadPerfPAPI()**

```
INTELLI::ThreadPerfPAPI::ThreadPerfPAPI (
    int cpu ) [inline]
```

To setup this perf to specific cpu.

Parameters

<i>cpu</i>	>=0 for any specific cpu, =-1 for all cpu that may run this process
------------	---

8.154.3 Member Function Documentation**8.154.3.1 addPapiTag() [1/2]**

```
void INTELLI::ThreadPerfPAPI::addPapiTag (
    std::string displayTag,
    int code ) [inline]
```

to add a paipi event to be detected

Parameters

<i>displayTag</i>	the tag to be displayed in your results
<i>code</i>	the papi lib event code

8.154.3.2 addPapiTag() [2/2]

```
void INTELLI::ThreadPerfPAPI::addPapiTag (
    std::string displayTag,
    std::string papiTag ) [inline]
```

to add a paipi event to be detected

Parameters

<i>displayTag</i>	the tag to be displayed in your results
<i>papiTag</i>	the built-in tag of papi lib

8.154.3.3 getResultById()

```
virtual uint64_t INTELLI::ThreadPerfPAPI::getResultById (
    size_t idx ) [inline], [virtual]
```

Get the perf result by its index of PerfPair.

Parameters

<i>idx</i>	The index
------------	-----------

Returns

The value

Reimplemented from [INTELLI::ThreadPerf](#).

8.154.3.4 getResultByName()

```
virtual uint64_t INTELLI::ThreadPerfPAPI::getResultByName (
    string name ) [inline], [virtual]
```

Get the perf result by its name of of PerfPair.

Parameters

<i>idx</i>	The index
------------	-----------

Returns

The value

Reimplemented from [INTELLI::ThreadPerf](#).

8.154.3.5 initEventsByCfg()

```
void INTELLI::ThreadPerfPAPI::initEventsByCfg (
    ConfigMapPtr cfg ) [inline], [virtual]
```

init the perf events according to configmap

Parameters

<i>cfg</i>	tyhe configmap
------------	----------------

Reimplemented from [INTELLI::ThreadPerf](#).

8.154.3.6 resultToConfigMap()

```
virtual ConfigMapPtr INTELLI::ThreadPerfPAPI::resultToConfigMap ( ) [inline], [virtual]
```

convert the perf result into a [ConfigMap](#)

Returns

The key-value store of configMap, in shared pointer

Note

must stop after calling stop

Reimplemented from [INTELLI::ThreadPerf](#).

8.154.3.7 start()

```
virtual void INTELLI::ThreadPerfPAPI::start ( ) [inline], [virtual]
```

To start perf tracing.

Note

call after [setPerfList](#)

Reimplemented from [INTELLI::ThreadPerf](#).

The documentation for this class was generated from the following file:

- [include/Utils/ThreadPerfPAPI.hpp](#)

8.155 BS::timer Class Reference

A helper class to measure execution time for benchmarking purposes.

```
#include <BS_thread_pool.hpp>
```

Public Member Functions

- void [start](#) ()
Start (or restart) measuring time.
- void [stop](#) ()
Stop measuring time and store the elapsed time since [start\(\)](#).
- `std::chrono::milliseconds::rep` [ms](#) () const
Get the number of milliseconds that have elapsed between [start\(\)](#) and [stop\(\)](#).

8.155.1 Detailed Description

A helper class to measure execution time for benchmarking purposes.

8.155.2 Member Function Documentation

8.155.2.1 ms()

```
std::chrono::milliseconds::rep BS::timer::ms ( ) const [inline]
```

Get the number of milliseconds that have elapsed between [start\(\)](#) and [stop\(\)](#).

Returns

The number of milliseconds.

The documentation for this class was generated from the following file:

- [include/Utils/BS_thread_pool.hpp](#)

8.156 CANDY::FLANN::UniqueRandom Class Reference

The class to output unique random values.

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

Public Member Functions

- void **init** (int64_t n)
- **UniqueRandom** (int64_t n)
- int64_t **next** ()

8.156.1 Detailed Description

The class to output unique random values.

The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.157 INTELLI::UtilityFunctions Class Reference

Static Public Member Functions

- static size_t **timeLast** (struct timeval past, struct timeval now)
- static size_t **timeLastUs** (struct timeval past)
- static int **bind2Core** (int id)
- static std::vector< size_t > **avgPartitionSizeFinal** (size_t inS, std::vector< size_t > partitionWeight)
- static std::vector< size_t > **weightedPartitionSizeFinal** (size_t inS, std::vector< size_t > partitionWeight)
- static size_t **to_periodical** (size_t val, size_t period)
- static double **getLatencyPercentage** (double fraction, std::vector< INTELLI::IntelliTimeStampPtr > &myTs)
get the latency percentile from a time stamp vector
- static bool **saveTimeStampToFile** (std::string fname, std::vector< INTELLI::IntelliTimeStampPtr > &myTs, bool skipZero=true)
save the time stamps to csv file
- static bool **existRow** (torch::Tensor base, torch::Tensor row)
- static double **calculateRecall** (std::vector< torch::Tensor > groundTruth, std::vector< torch::Tensor > prob)
calculate the recall by comparing with ground truth
- static bool **tensorListToFile** (std::vector< torch::Tensor > &tensorVec, std::string folderName)
convert a list of tensors to a folder with multiple flat binary form files, i.e., <rows> <cols> <flat data> for each
- static std::vector< torch::Tensor > **tensorListFromFile** (std::string folderName, uint64_t tensors)
convert a list of tensors to a folder with multiple flat binary form files, i.e., <rows> <cols> <flat data> for each

8.157.1 Member Function Documentation

8.157.1.1 bind2Core()

```
int INTELLI::UtilityFunctions::bind2Core (
    int id ) [static]
```

bind to CPU

- bind the thread to core according to id

Parameters

<i>id</i>	the core you plan to bind, -1 means let os decide
-----------	---

Returns

cpuld, the real core that bind to

fixed some core bind bugs

8.157.1.2 calculateRecall()

```
static double INTELLI::UtilityFunctions::calculateRecall (
    std::vector< torch::Tensor > groundTruth,
    std::vector< torch::Tensor > prob ) [inline], [static]
```

calculate the recall by comparing with ground truth

Parameters

<i>groundTruth</i>	The ground truth
<i>prob</i>	The tensor result to be validated

Returns

the recall in 0~1

8.157.1.3 getLatencyPercentage()

```
static double INTELLI::UtilityFunctions::getLatencyPercentage (
    double fraction,
    std::vector< INTELLI::IntelliTimeStampPtr > & myTs ) [inline], [static]
```

get the latency percentile from a time stamp vector

Parameters

<i>fraction</i>	the percentile in 0~1
<i>myTs</i>	the time stamp vector

Returns

the latency value

8.157.1.4 saveTimeStampToFile()

```
static bool INTELLI::UtilityFunctions::saveTimeStampToFile (
    std::string fname,
    std::vector< INTELLI::IntelliTimeStampPtr > & myTs,
    bool skipZero = true ) [inline], [static]
```

save the time stamps to csv file

Parameters

<i>fname</i>	the name of output file
<i>myTs</i>	the time stamp vector
<i>skipZero</i>	whether skip zero time

Returns

whether the output is successful

8.157.1.5 tensorListFromFile()

```
static std::vector<torch::Tensor> INTELLI::UtilityFunctions::tensorListFromFile (
    std::string folderName,
    uint64_t tensors ) [inline], [static]
```

convert a list of tensors to a folder with multiple flat binary form files, i.e., <rows> <cols> <flat data> for each

Parameters

<i>folderName</i>	the name of folder
<i>tensors</i>	the number of tensors to be loaded

Note

this will overwrite the whole folder!

Returns

the vector of tensors

8.157.1.6 tensorListToFile()

```
static bool INTELLI::UtilityFunctions::tensorListToFile (
    std::vector< torch::Tensor > & tensorVec,
    std::string folderName ) [inline], [static]
```

convert a list of tensors to a folder with multiple flat binary form files, i.e., <rows> <cols> <flat data> for each

Parameters

<i>A</i>	the list of tensors
<i>folderName</i>	the name of folder

Note

this will overwrite the whole folder!

Returns

bool, the output is successful or not

The documentation for this class was generated from the following files:

- include/Utils/UtilityFunctions.h
- src/Utils/UtilityFunctions.cpp

8.158 VertexComparison Struct Reference

Public Member Functions

- bool **operator**() (const std::pair< double, [YinYangVertexPtr](#) > &a, const std::pair< double, [YinYangVertexPtr](#) > &b) const

The documentation for this struct was generated from the following file:

- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.159 CANDY::FLANN::VisitBitset Class Reference

Public Member Functions

- **VisitBitset** (size_t s)
- void **clear** ()
- bool **empty** ()
- void **reset** (int64_t index)
- void **reset_block** (int64_t index)
- void **resize** (size_t s)
- void **set** (int64_t index)
- size_t **getSize** ()
- bool **test** (int64_t index)

The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.160 VisitedBitset Class Reference

The visited array of nodes.

```
#include <CANDY/FlannIndex/FlannUtils.h>
```

8.160.1 Detailed Description

The visited array of nodes.

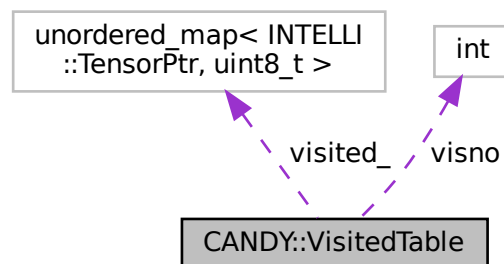
The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.161 CANDY::VisitedTable Class Reference

```
#include <HNSW.h>
```

Collaboration diagram for CANDY::VisitedTable:



Public Member Functions

- void **set** (VertexPtr idx)
- bool **get** (VertexPtr idx)
- void **set** (INTELLI::TensorPtr idx)
- bool **get** (INTELLI::TensorPtr idx)
- void **advance** ()

Public Attributes

- std::unordered_map< INTELLI::TensorPtr, uint8_t > **visited_**
For Tensor *t*, use `visited[TensorPtr]` to define if its visited;.
- int **visno**

8.161.1 Detailed Description

Table to store visited iteration number during search and insert; Now only update the number and store nothing

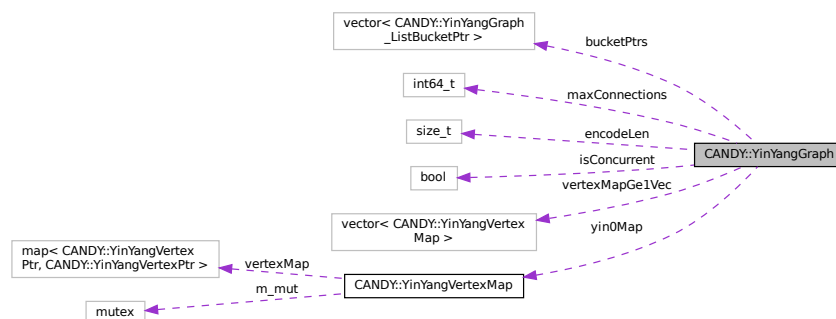
The documentation for this class was generated from the following file:

- include/CANDY/HNSWNaive/HNSW.h

8.162 CANDY::YinYangGraph Class Reference

The top class of yinyang graph, containing ivf list and critical graph information.

Collaboration diagram for CANDY::YinYangGraph:



Public Member Functions

- void **init** (size_t bkts, size_t _encodeLen, int64_t _maxCon)
init this YinYangGraph_List
- void **insertTensorWithEncode** (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool isConcurrent=false)
insert a tensor with its encode
- bool **deleteTensorWithEncode** (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool isConcurrent=false)
delete a tensor with its encode
- torch::Tensor **getMinimumNumOfTensors** (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, int64_t minimumNum)
get minimum number of tensors that are candidate to query t

Public Attributes

- bool **isConcurrent** = false

Static Protected Member Functions

- static uint8_t **getLeftIdxU8** (uint8_t idx, uint8_t leftOffset, bool *reachedLeftMost)
- static uint8_t **getRightIdxU8** (uint8_t idx, uint8_t rightOffset, bool *reachedRightMost)

Protected Attributes

- `std::vector< CANDY::YinYangGraph_ListBucketPtr > bucketPtrs`
- `int64_t maxConnections = 0`
- `size_t encodeLen = 0`
- `YinYangVertexMap yin0Map`
- `std::vector< YinYangVertexMap > vertexMapGe1Vec`

8.162.1 Detailed Description

The top class of yinyang graph, containing ivf list and critical graph information.

- This is a hybrid structure, using encoding-based ranging to assist in graph navigation
- This is a hierarchical structure, using high layer yin vertex to summarize data points (marked as yang)

8.162.2 Member Function Documentation

8.162.2.1 deleteTensorWithEncode()

```
bool CANDY::YinYangGraph::deleteTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    bool isConcurrent = false )
```

delete a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.162.2.2 getMinimumNumOfTensors()

```
torch::Tensor CANDY::YinYangGraph::getMinimumNumOfTensors (
    torch::Tensor & t,
```

```
std::vector< uint8_t > & encode,
uint64_t bktIdx,
int64_t minimumNum )
```

get minimum number of tensors that are candidate to query t

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktIdx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

- a 2-D tensor contain all, torch::zeros({minimumNum,D}) if got nothing

1. set to the top tier like [HNSW](#)
2. set to the related cell

8.162.2.3 init()

```
void CANDY::YinYangGraph::init (
    size_t bkts,
    size_t _encodeLen,
    int64_t _maxCon )
```

init this YinYangGraph_List

Parameters

<i>bkts</i>	the number of buckets
<i>_encodeLen</i>	the length of tensors' encoding
<i>_maxCon</i>	the maximum number of connections in graph vertex

8.162.2.4 insertTensorWithEncode()

```
void CANDY::YinYangGraph::insertTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    uint64_t bktIdx,
    bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>bktidx</i>	the index number of bucket
<i>isConcurrent</i>	whether this process is concurrently executed

The documentation for this class was generated from the following files:

- [include/CANDY/YinYangGraphIndex/YinYangGraph.h](#)
- [src/CANDY/YinYangGraphIndex/YinYangGraph.cpp](#)

8.163 CANDY::YinYangGraph_DistanceFunctions Class Reference

Static Public Member Functions

- static float **L2Distance** (const torch::Tensor &tensor1, const torch::Tensor &tensor2)

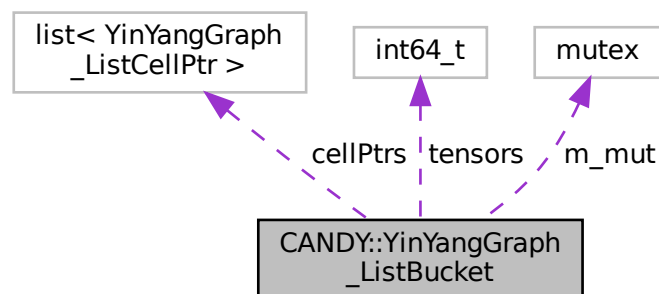
The documentation for this class was generated from the following file:

- [include/CANDY/YinYangGraphIndex/YinYangGraph.h](#)

8.164 CANDY::YinYangGraph_ListBucket Class Reference

a bucket of multiple [YinYangGraph_ListCell](#)

Collaboration diagram for CANDY::YinYangGraph_ListBucket:



Public Member Functions

- `int64_t size ()`
- `void lock ()`
lock this bucket
- `void unlock ()`
unlock this bucket
- `void insertTensorWithEncode (torch::Tensor &t, int64_t maxNeighborCnt, std::vector< uint8_t > &encode, YinYangVertexMap &yin0Map, std::vector< YinYangVertexMap > &vertexMapGe1Vec, bool isConcurrent=false)`
insert a tensor with its encode
- `bool deleteTensorWithEncode (torch::Tensor &t, std::vector< uint8_t > &encode, bool isConcurrent=false)`
delete a tensor with its encode
- `bool deleteTensor (torch::Tensor &t, bool isConcurrent=false)`
delete a tensor
- `YinYangVertexPtr getVertexWithEncode (std::vector< uint8_t > &encode)`
to get the vertex which is linked to an encode, first try exact match, then just return the first one

Protected Attributes

- `int64_t tensors = 0`
- `std::list< YinYangGraph_ListCellPtr > cellPtrs`
- `std::mutex m_mut`

8.164.1 Detailed Description

a bucket of multiple [YinYangGraph_ListCell](#)

8.164.2 Member Function Documentation

8.164.2.1 deleteTensor()

```
bool CANDY::YinYangGraph_ListBucket::deleteTensor (
    torch::Tensor & t,
    bool isConcurrent = false )
```

delete a tensor

Note

will check the equal condition by `torch::equal`

Parameters

<i>t</i>	the tensor
<i>isConcurrent</i>	whether this process is concurrently executed
	.

Returns

bool whether the tensor is really deleted

8.164.2.2 deleteTensorWithEncode()

```
bool CANDY::YinYangGraph_ListBucket::deleteTensorWithEncode (
    torch::Tensor & t,
    std::vector< uint8_t > & encode,
    bool isConcurrent = false )
```

delete a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>encode</i>	the corresponding encode
<i>isConcurrent</i>	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.164.2.3 getVertexWithEncode()

```
YinYangVertexPtr CANDY::YinYangGraph_ListBucket::getVertexWithEncode (
    std::vector< uint8_t > & encode )
```

to get the vertex which is linked to an encode, first try exact match, then just return the first one

Returns

the vertex

8.164.2.4 insertTensorWithEncode()

```
void CANDY::YinYangGraph_ListBucket::insertTensorWithEncode (
    torch::Tensor & t,
    int64_t maxNeighborCnt,
    std::vector< uint8_t > & encode,
    CANDY::YinYangVertexMap & yin0Map,
    std::vector< YinYangVertexMap > & vertexMapGelVec,
    bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

<i>t</i>	the tensor
<i>maxNeighborCnt</i>	
<i>encode</i>	the corresponding encode
<i>yin0Map</i>	the map of yin vertex at level 0
<i>vertexMapGe1Vec</i>	the vector of vertexMap in all level greater or equal to 1
<i>isConcurrent</i>	whether this process is concurrently executed

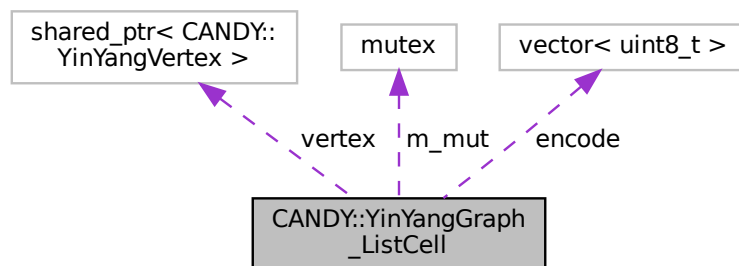
The documentation for this class was generated from the following files:

- include/CANDY/YinYangGraphIndex/YinYangGraph.h
- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.165 CANDY::YinYangGraph_ListCell Class Reference

a cell of an ending [YinYangVertex](#)

Collaboration diagram for CANDY::YinYangGraph_ListCell:



Public Member Functions

- void [lock](#) ()
lock this cell
- void [unlock](#) ()
unlock this cell
- void **setEncode** (std::vector< uint8_t > _encode)
- std::vector< uint8_t > **getEncode** ()
- void [insertTensor](#) (torch::Tensor &t, int64_t maxNeighborCnt, [YinYangVertexMap](#) &yin0Map, std::vector< [YinYangVertexMap](#) > &vertexMapGe1Vec)
insert a tensor
- bool [deleteTensor](#) (torch::Tensor &t)
delete a tensor
- [YinYangVertexPtr](#) [getVertex](#) ()
to get the vertex

Protected Attributes

- [YinYangVertexPtr](#) **vertex** = nullptr
- std::mutex **m_mut**
- std::vector< uint8_t > **encode**

8.165.1 Detailed Description

a cell of an ending [YinYangVertex](#)

8.165.2 Member Function Documentation

8.165.2.1 deleteTensor()

```
bool CANDY::YinYangGraph_ListCell::deleteTensor (
    torch::Tensor & t )
```

delete a tensor

Note

will check the equal condition by torch::equal

Parameters

<i>t</i>	the tensor @return bool whether the tensor is really deleted
----------	--

8.165.2.2 getVertex()

```
YinYangVertexPtr CANDY::YinYangGraph_ListCell::getVertex ( ) [inline]
```

to get the vertex

Returns

the vertex member

8.165.2.3 insertTensor()

```
void CANDY::YinYangGraph_ListCell::insertTensor (
    torch::Tensor & t,
    int64_t maxNeighborCnt,
    YinYangVertexMap & yin0Map,
    std::vector< YinYangVertexMap > & vertexMapGe1Vec )
```

insert a tensor

handling the listcell

Parameters

<i>t</i>	the tensor
<i>maxNeighborCnt</i>	the maximum count of neighbors
<i>yin0Map</i>	the map of yin vertex at level 0
<i>vertexMapGe1Vec</i>	the vector of vertexMap in all level greater or equal to 1

1. a new vertex if vertex==nullptr, create a yin vertex

@broef to connect it with other level 0 yin vertex

to create a new yang vertex and connect it with vertex member

The documentation for this class was generated from the following files:

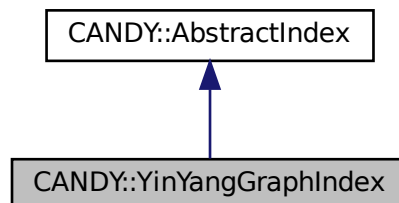
- include/CANDY/YinYangGraphIndex/YinYangGraph.h
- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.166 CANDY::YinYangGraphIndex Class Reference

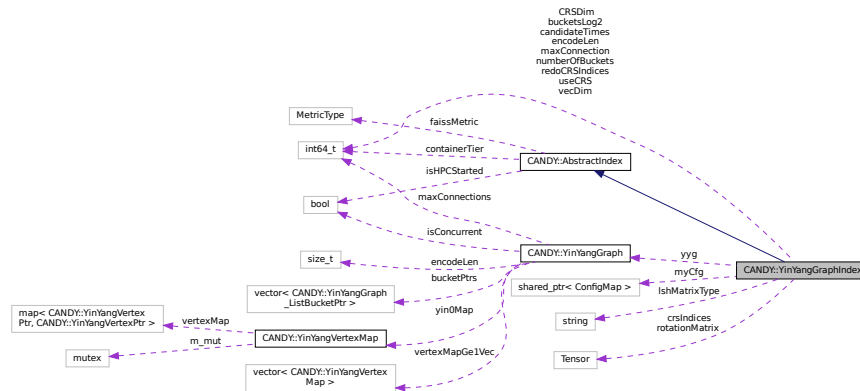
The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph.

```
#include <CANDY/YinYangGraphIndex.h>
```

Inheritance diagram for CANDY::YinYangGraphIndex:



Collaboration diagram for CANDY::YinYangGraphIndex:



Public Member Functions

- virtual bool [setConfig](#) (INTELLI::ConfigMapPtr cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) (torch::Tensor &t)
insert a tensor
- virtual std::vector< torch::Tensor > [searchTensor](#) (torch::Tensor &q, int64_t k)
search the k-NN of a query tensor, return the result tensors

Static Public Member Functions

- static torch::Tensor [crsAmm](#) (torch::Tensor &A, torch::Tensor &B, torch::Tensor &indices)
thw column row sampling to compute approximate matrix multiplication

Protected Member Functions

- std::vector< uint8_t > [encodeSingleRow](#) (torch::Tensor &tensor, uint64_t *bucket)
- torch::Tensor [randomProjection](#) (torch::Tensor &a)
- void [genCrsIndices](#) (void)
to generate the sampling indices of crs

Protected Attributes

- INTELLI::ConfigMapPtr [myCfg](#) = nullptr
- CANDY::YinYangGraph [yyg](#)
- int64_t [vecDim](#) = 0
- int64_t [maxConnection](#) = 0
- int64_t [numberOfBuckets](#) = 4096
- int64_t [encodeLen](#) = 1
- int64_t [candidateTimes](#) = 1
- int64_t [useCRS](#) = 0
- int64_t [CRSDim](#) = 1
- int64_t [bucketsLog2](#) = 0
- int64_t [redoCrsIndices](#) = 0
- std::string [IshMatrixType](#) = "gaussian"
- torch::Tensor [rotationMatrix](#)
- torch::Tensor [crsIndices](#)

Additional Inherited Members

8.166.1 Detailed Description

The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph.

Todo implement the delete and revise later

Note

currently single thread

config parameters

- vecDim, the dimension of vectors, default 768, l64
- maxConnection, the max number of connections in the yinyang graph (for yang vertex of data), default 256, l64
- candidateTimes, the times of k to determine minimum candidates, default 1, l64
- numberOfBuckets, the number of first titer buckets, default 4096, l64, suggest 2^n
- encodeLen, the length of LSH encoding, in bytes, default 1, l64
- metricType, the type of AKNN metric, default L2, String
- lshMatrixType, the type of lsh matrix, default gaussian, String
 - gaussian means a $N(0,1)$ LSH matrix
 - random means a random matrix where each value ranges from -0.5~0.5
- useCRS, whether or not use column row sampling in projecting the vector, 0 (No), l64
 - further trade off of accuracy v.s. efficiency
- CRSDim, the dimension which are not pruned by crs, 1/10 of vecDim, l64
- redoCRSIndices, whether or not re-generate the indices of CRS, 0 (No), l64

8.166.2 Member Function Documentation

8.166.2.1 crsAmm()

```
torch::Tensor CANDY::YinYangGraphIndex::crsAmm (
    torch::Tensor & A,
    torch::Tensor & B,
    torch::Tensor & indices ) [static]
```

thw column row sampling to compute approximate matrix multiplication

Parameters

<i>A</i>	the left side matrix
<i>B</i>	the right side matrix
<i>idx</i>	the indices of sampling
<i>_crsDim</i>	the dimension of preserved dimensions

8.166.2.2 insertTensor()

```
bool CANDY::YinYangGraphIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.166.2.3 searchTensor()

```
std::vector< torch::Tensor > CANDY::YinYangGraphIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.166.2.4 setConfig()

```
bool CANDY::YinYangGraphIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

generate the rotation matrix for random projection

Reimplemented from [CANDY::AbstractIndex](#).

The documentation for this class was generated from the following files:

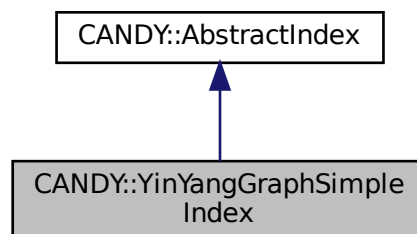
- [include/CANDY/YinYangGraphIndex.h](#)
- [src/CANDY/YinYangGraphIndex.cpp](#)

8.167 CANDY::YinYangGraphSimpleIndex Class Reference

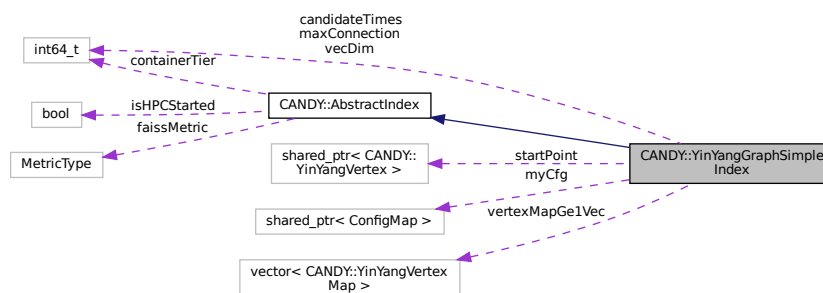
The class of indexing using a simple yinyang graph, there is no LSH search is only within the linked yinyanggraph.

```
#include <CANDY/YinYangGraphSimpleIndex.h>
```

Inheritance diagram for CANDY::YinYangGraphSimpleIndex:



Collaboration diagram for CANDY::YinYangGraphSimpleIndex:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
set the index-specific config related to one index
- virtual bool [insertTensor](#) ([torch::Tensor](#) &t)
insert a tensor
- virtual [std::vector](#)< [torch::Tensor](#) > [searchTensor](#) ([torch::Tensor](#) &q, [int64_t](#) k)
search the k-NN of a query tensor, return the result tensors

Protected Member Functions

- virtual bool [insertSingleRowTensor](#) ([torch::Tensor](#) &t)
insert a tensor

Protected Attributes

- [INTELLI::ConfigMapPtr](#) **myCfg** = nullptr
- [std::vector](#)< [YinYangVertexMap](#) > **vertexMapGe1Vec**
- [int64_t](#) **vecDim** = 0
- [int64_t](#) **maxConnection** = 0
- [int64_t](#) **candidateTimes** = 1
- [YinYangVertexPtr](#) **startPoint** = nullptr

Additional Inherited Members

8.167.1 Detailed Description

The class of indexing using a simpe yinyang graph,there is no LSH search is only within the linked yinyanggraph.

Todo implement the delete and revise later

Note

currently single thread

config parameters

- vecDim, the dimension of vectors, default 768, 164
- maxConnection, the max number of connections in the yinyang graph (for yang vertex of data), default 256, 164
- candidateTimes, the times of k to determine minimum candidates, default 1 ,164
- metricType, the type of AKNN metric, default L2, String

8.167.2 Member Function Documentation

8.167.2.1 insertSingleRowTensor()

```
bool CANDY::YinYangGraphSimpleIndex::insertSingleRowTensor (
    torch::Tensor & t ) [protected], [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, single row
----------	------------------------

Returns

bool whether the insertion is successful

8.167.2.2 insertTensor()

```
bool CANDY::YinYangGraphSimpleIndex::insertTensor (
    torch::Tensor & t ) [virtual]
```

insert a tensor

Parameters

<i>t</i>	the tensor, accept multiple rows
----------	----------------------------------

Returns

bool whether the insertion is successful

Reimplemented from [CANDY::AbstractIndex](#).

8.167.2.3 searchTensor()

```
std::vector< torch::Tensor > CANDY::YinYangGraphSimpleIndex::searchTensor (
    torch::Tensor & q,
    int64_t k ) [virtual]
```

search the k-NN of a query tensor, return the result tensors

Parameters

<i>t</i>	the tensor, allow multiple rows
<i>k</i>	the returned neighbors

Returns

std::vector<torch::Tensor> the result tensor for each row of query

Reimplemented from [CANDY::AbstractIndex](#).

8.167.2.4 setConfig()

```
bool CANDY::YinYangGraphSimpleIndex::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

set the index-specific config related to one index

Parameters

<i>cfg</i>	the config of this class
------------	--------------------------

Returns

bool whether the configuration is successful

Reimplemented from [CANDY::AbstractIndex](#).

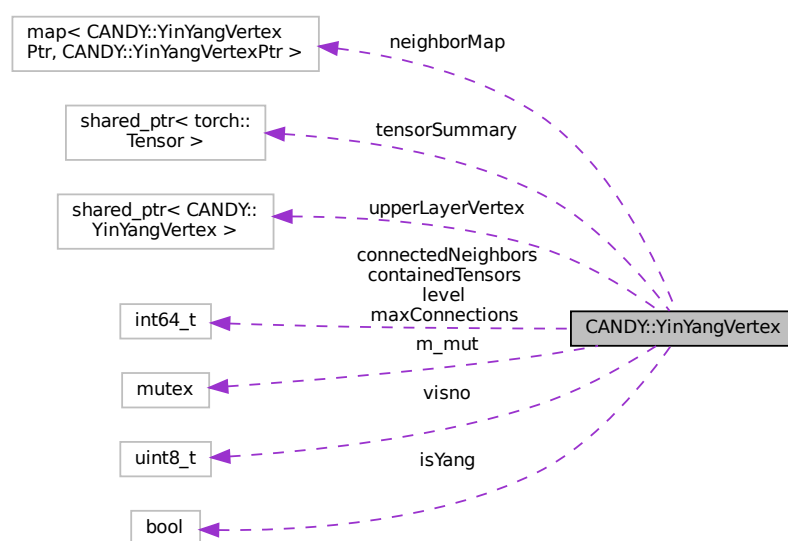
The documentation for this class was generated from the following files:

- [include/CANDY/YinYangGraphSimpleIndex.h](#)
- [src/CANDY/YinYangGraphSimpleIndex.cpp](#)

8.168 CANDY::YinYangVertex Class Reference

The class of a [YinYangVertex](#), storing the data in each vertex.

Collaboration diagram for CANDY::YinYangVertex:



Public Member Functions

- void [init](#) (torch::Tensor &ts, int64_t _level, int64_t maxNumOfNeighbor, int64_t _containedTensors, bool _isYang)
init a yinyang vertex
- void [lock](#) ()
lock this vertex
- void [unlock](#) ()
unlock this vertex
- void [attachTensor](#) (torch::Tensor &ts)
attach a tensor with this vertex
- void [detachTensor](#) (torch::Tensor &ts)
detach a tensor with this vertex
- void [setUpperLayer](#) (YinYangVertexPtr upv)
- std::string [toString](#) (bool shortInfo=true)

Static Public Member Functions

- static YinYangVertexPtr [greedySearchForNearestVertex](#) (YinYangVertexPtr src, YinYangVertexPtr entryPoint, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
to get the nearest vertex of src, start at entryPoint
- static YinYangVertexPtr [greedySearchForNearestVertex](#) (torch::Tensor &src, YinYangVertexPtr entryPoint, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
to get the nearest vertex of src, start at entryPoint
- static torch::Tensor [greedySearchForKNearestTensor](#) (torch::Tensor &src, YinYangVertexPtr entryPoint, int64_t k, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
to get k nearest tesnor of src, start at entryPoint
- static std::vector< YinYangVertexPtr > [greedySearchForKNearestVertex](#) (YinYangVertexPtr src, YinYangVertexPtr entryPoint, int64_t k, bool ignoreYin, bool forceTheSameLevel, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
to get k nearest vertex of src, start at entryPoint
- static std::vector< YinYangVertexPtr > [greedySearchForKNearestVertex](#) (torch::Tensor &src, YinYangVertexPtr entryPoint, int64_t k, bool ignoreYin, bool forceTheSameLevel, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
to get k nearest vertex of src, start at entryPoint
- static bool [tryToConnect](#) (YinYangVertexPtr a, YinYangVertexPtr b, std::vector< YinYangVertexMap > &vertexMapGe1Vec, floatDistanceFunction_t df=YinYangGraph_DistanceFunctions::L2Distance)
try to connect vertex a and b with each other

Public Attributes

- [INTELLI::TensorPtr](#) **tensorSummary**
- int64_t **containedTensors** = 0
- int64_t **level** = 0
- int64_t **connectedNeighbors** = 0
- int64_t **maxConnections** = 0
- bool **isYang** = false
- std::map< [CANDY::YinYangVertexPtr](#), [CANDY::YinYangVertexPtr](#) > **neighborMap**
- YinYangVertexPtr **upperLayerVertex** = nullptr
- uint8_t **visno**

Protected Attributes

- `std::mutex m_mut`

8.168.1 Detailed Description

The class of a [YinYangVertex](#), storing the data in each vertex.

Note

now storing each vertex's neighbors, visited number and level, with a pointer to the vector

- yin: means this is a summarizing or bridge tensor, not a really data point
 - yin vertex will only be used for navigation, not output to result
 - yin vertex will be less likely to be deleted, completely changed compared with yang
 - yin vertex can be a summary of multiple tensors
- yang: means the real data point

8.168.2 Member Function Documentation

8.168.2.1 attachTensor()

```
void CANDY::YinYangVertex::attachTensor (
    torch::Tensor & ts )
```

attach a tensor with this vertex

Parameters

<i>ts</i>	the tensor to be attached
-----------	---------------------------

Note

assume ts is a single row

8.168.2.2 detachTensor()

```
void CANDY::YinYangVertex::detachTensor (
    torch::Tensor & ts )
```

detach a tensor with this vertex

Parameters

<i>ts</i>	the tensor to be detached
-----------	---------------------------

Note

assume *ts* is a single row

8.168.2.3 `greedySearchForKNearestTensor()`

```
torch::Tensor CANDY::YinYangVertex::greedySearchForKNearestTensor (
    torch::Tensor & src,
    CANDY::YinYangVertexPtr entryPoint,
    int64_t k,
    CANDY::floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance )
[static]
```

to get k nearest tesnor of src, start at entryPoint

Parameters

<i>src</i>	the source tensor to be used as reference
<i>entryPoint</i>	the entryPoint to start greedy search
<i>k</i>	the number @parm df the distance calculate function

Todo This one is just NNDecent greedy policy, perhaps can be better

Returns

the result tensor

8.168.2.4 `greedySearchForKNearestVertex()` [1/2]

```
std::vector< YinYangVertexPtr > CANDY::YinYangVertex::greedySearchForKNearestVertex (
    torch::Tensor & src,
    CANDY::YinYangVertexPtr entryPoint,
    int64_t k,
    bool ignoreYin,
    bool forceTheSameLevel,
    CANDY::floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance )
[static]
```

to get k nearest vertex of src, start at entryPoint

Parameters

<i>src</i>	the source tensor
<i>entryPoint</i>	the entryPoint to start greedy search
<i>k</i>	the number
<i>ignoreYin</i>	whether or not ignore YinVertex
<i>forceTheSameLevel</i>	whether or not force to find it at the same level @parm df the distance calculate function

Todo This one is just NNDecent greedy policy, perhaps can be better

Returns

the nearest vertex

8.168.2.5 greedySearchForKNearestVertex() [2/2]

```
std::vector< YinYangVertexPtr > CANDY::YinYangVertex::greedySearchForKNearestVertex (
    YinYangVertexPtr src,
    YinYangVertexPtr entryPoint,
    int64_t k,
    bool ignoreYin,
    bool forceTheSameLevel,
    floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance ) [static]
```

to get k nearest vertex of src, start at entryPoint

Parameters

<i>src</i>	the source vertex to be used as reference
<i>entryPoint</i>	the entryPoint to start greedy search
<i>k</i>	the number
<i>ignoreYin</i>	whether or not ignore YinVertex
<i>forceTheSameLevel</i>	whether or not force to find it at the same level @parm df the distance calculate function

Todo This one is just NNDecent greedy policy, perhaps can be better

Returns

the nearest vertex

8.168.2.6 greedySearchForNearestVertex() [1/2]

```
CANDY::YinYangVertexPtr CANDY::YinYangVertex::greedySearchForNearestVertex (
    torch::Tensor & src,
    YinYangVertexPtr entryPoint,
    floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance ) [static]
```

to get the nearest vertex of src, start at entryPoint

Parameters

<i>src</i>	the source tensor to be used as reference
<i>entryPoint</i>	the entryPoint to start greedy search @parm df the distance calculate function

Returns

the nearest vertex

1. scan the distance of neighbors
2. if this one is optimal, return

switch into the new optimal ones

8.168.2.7 greedySearchForNearestVertex() [2/2]

```
CANDY::YinYangVertexPtr CANDY::YinYangVertex::greedySearchForNearestVertex (
    YinYangVertexPtr src,
    YinYangVertexPtr entryPoint,
    floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance ) [static]
```

to get the nearest vertex of src, start at entryPoint

Parameters

<i>src</i>	the source vertex to be used as reference
<i>entryPoint</i>	the entryPoint to start greedy search @parm df the distance calculate function

Returns

the nearest vertex

8.168.2.8 init()

```
void CANDY::YinYangVertex::init (
    torch::Tensor & ts,
    int64_t _level,
```

```
int64_t  maxNumOfNeighbor,  
int64_t  _containedTensors,  
bool  _isYang )
```

init a yinyang vertex

handling the vertex

Parameters

<i>ts</i>	the tensor linked to this vertex
<i>_level</i>	the level of this one
<i>maxNumOfNeighbor</i>	the maximum number of neighbors
<i>_containedTensors</i>	the number of contained tensors
<i>_isYang</i>	whether this is a yang vertex

8.168.2.9 setUpperLayer()

```
void CANDY::YinYangVertex::setUpperLayer (  
    YinYangVertexPtr upv ) [inline]
```

@breif to set the upper layer vertex of this one

Parameters

<i>upv</i>	the upper layer vertex
------------	------------------------

8.168.2.10 toString()

```
std::string CANDY::YinYangVertex::toString (  
    bool shortInfo = true )
```

@breif convert this vertex into string

Parameters

<i>shortInfo</i>	whether or not shorten the information of tensor filed
------------------	--

Returns

the converted string

8.168.2.11 tryToConnect()

```
bool CANDY::YinYangVertex::tryToConnect (
    CANDY::YinYangVertexPtr a,
    CANDY::YinYangVertexPtr b,
    std::vector< YinYangVertexMap > & vertexMapGe1Vec,
    CANDY::floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance )
[static]
```

try to connect vertex a and b with each other

Parameters

<i>a</i>	the new vertex
<i>b</i>	some existing vertex
<i>vertexMapGe1Vec</i>	the vector of vertexMap in all level greater or equal to 1

Returns

whether the connection is established

1. if b is not fully connected, than just connect
2. try to create an upper layer of b

2.1 create upper layer links

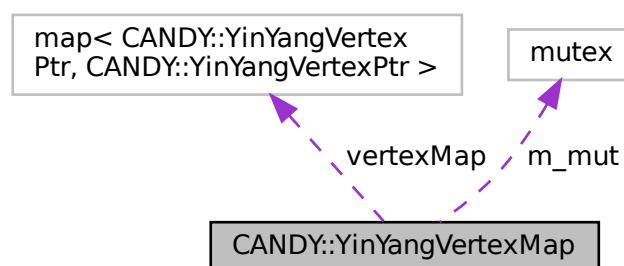
1. shrink connection of b

The documentation for this class was generated from the following files:

- include/CANDY/YinYangGraphIndex/YinYangGraph.h
- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.169 CANDY::YinYangVertexMap Class Reference

Collaboration diagram for CANDY::YinYangVertexMap:



Public Member Functions

- **YinYangVertexMap** (const [YinYangVertexMap](#) &other)
- void [lock](#) ()
lock this map
- void [unlock](#) ()
unlock this map
- bool [exist](#) ([CANDY::YinYangVertexPtr](#) key)
To detect whether a vertex existis in the map.
- void [edit](#) ([CANDY::YinYangVertexPtr](#) kv)
To edit, i.e., mark the existence of a vertex.
- void [erase](#) ([CANDY::YinYangVertexPtr](#) kv)
To erase, i.e., mark the absence of a vertex.
- [YinYangVertexPtr](#) [nearestVertexWithinMe](#) ([YinYangVertexPtr](#) src)
to get the nearest vertex of src, from the map of this class

Static Public Member Functions

- static [YinYangVertexPtr](#) [nearestVertexWithinMap](#) ([YinYangVertexPtr](#) src, [YinYangVertexMap](#) &vmap, float↔
DistanceFunction_t df=[YinYangGraph_DistanceFunctions::L2Distance](#))
to get the nearest vertex of src, from a map
- static std::vector< [YinYangVertexPtr](#) > [nearestKVertexWithinMap](#) ([YinYangVertexPtr](#) src, [YinYangVertexMap](#) &vmap, int64_t k, floatDistanceFunction_t df=[YinYangGraph_DistanceFunctions::L2Distance](#))
to get the nearest vertex k of src, from a map

Public Attributes

- std::map< [CANDY::YinYangVertexPtr](#), [CANDY::YinYangVertexPtr](#) > **vertexMap**

Protected Attributes

- std::mutex **m_mut**

8.169.1 Member Function Documentation

8.169.1.1 [edit\(\)](#)

```
void CANDY::YinYangVertexMap::edit (
    CANDY::YinYangVertexPtr kv ) [inline]
```

To edit, i.e., mark the existence of a vertex.

Parameters

<i>kv</i>	the vertex pointer as key
-----------	---------------------------

Returns

bool for the result

8.169.1.2 erase()

```
void CANDY::YinYangVertexMap::erase (
    CANDY::YinYangVertexPtr kv ) [inline]
```

To erase, i.e., mark the absence of a vertex.

Parameters

<i>kv</i>	the vertex pointer as key
-----------	---------------------------

Returns

bool for the result

8.169.1.3 exist()

```
bool CANDY::YinYangVertexMap::exist (
    CANDY::YinYangVertexPtr key ) [inline]
```

To detect whether a vertex exists in the map.

Parameters

<i>key</i>	the vertex pointer as key
------------	---------------------------

Returns

bool for the result

8.169.1.4 nearestKVertexWithinMap()

```
std::vector< CANDY::YinYangVertexPtr > CANDY::YinYangVertexMap::nearestKVertexWithinMap (
    CANDY::YinYangVertexPtr src,
    CANDY::YinYangVertexMap & vmap,
    int64_t k,
    CANDY::floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance )
[static]
```

to get the nearest vertex k of src, from a map

Parameters

<i>src</i>	the source vertex to be used as reference
<i>vmap,the</i>	vertex map
<i>k,the</i>	number of nearest vertex to be found
<i>df</i>	the distance function

Returns

the nearest vertex

1. traverse
2. sort

8.169.1.5 nearestVertexWithinMap()

```
CANDY::YinYangVertexPtr CANDY::YinYangVertexMap::nearestVertexWithinMap (  
    CANDY::YinYangVertexPtr src,  
    CANDY::YinYangVertexMap & vmap,  
    floatDistanceFunction_t df = YinYangGraph_DistanceFunctions::L2Distance ) [static]
```

to get the nearest vertex of src, from a map

Parameters

<i>src</i>	the source vertex to be used as reference
<i>vmap,the</i>	vertex map
<i>df,the</i>	distance function

Returns

the nearest vertex

8.169.1.6 nearestVertexWithinMe()

```
YinYangVertexPtr CANDY::YinYangVertexMap::nearestVertexWithinMe (  
    YinYangVertexPtr src )
```

to get the nearest vertex of src, from the map of this class

Parameters

<i>src</i>	the source vertex to be used as reference
------------	---

Returns

the nearest vertex

The documentation for this class was generated from the following files:

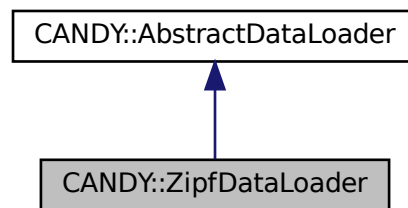
- [include/CANDY/YinYangGraphIndex/YinYangGraph.h](#)
- [src/CANDY/YinYangGraphIndex/YinYangGraph.cpp](#)

8.170 CANDY::ZipfDataLoader Class Reference

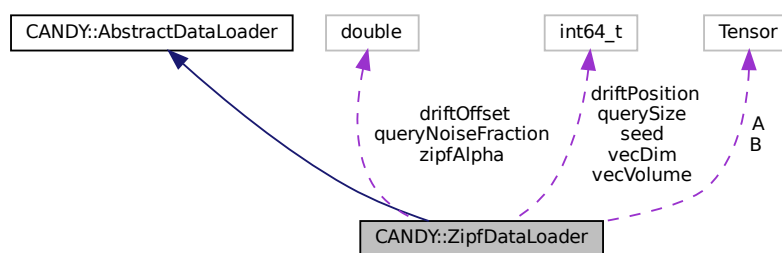
The class to load zipf data.

```
#include <DataLoader/ZipfDataLoader.h>
```

Inheritance diagram for CANDY::ZipfDataLoader:



Collaboration diagram for CANDY::ZipfDataLoader:



Public Member Functions

- virtual bool [setConfig](#) ([INTELLI::ConfigMapPtr](#) cfg)
Set the GLOBAL config map related to this loader.
- virtual torch::Tensor [getData](#) ()
get the data tensor
- virtual torch::Tensor [getQuery](#) ()
get the query tensor

Protected Member Functions

- torch::Tensor **generateZipfDistribution** (int64_t n, int64_t m, double alpha)

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- int64_t **vecDim**
- int64_t **vecVolume**
- int64_t **querySize**
- int64_t **seed**
- int64_t **driftPosition**
- double **driftOffset**
- double **queryNoiseFraction**
- double **zipfAlpha**

8.170.1 Detailed Description

The class to load zipf data.

Note

:

- Must have a global config by [setConfig](#)

Default behavior

- create
- call [setConfig](#), this function will also generate the tensor A and B correspondingly
- call [getData](#) to get the raw data
- call [getQuery](#) to get the query

parameters of config

- vecDim, the dimension of vectors, default 768, I64
- vecVolume, the volume of vectors, default 1000, I64
- normalizeTensor, whether or not normalize the tensors in L2, 1 (yes), I64
- "zipfAlpha" The zipf factor for, Double, 0-highly skewed value. 1- uniform dist.
- driftPosition, the position of starting some 'concept drift', default 0 (no drift), I64
 - driftOffset, the offset value of concept drift, default 0.5, Double
 - queryNoiseFraction, the fraction of noise in query, default 0, allow 0~1, Double
- querySize, the size of query, default 10, I64
- seed, the Zipf seed, default 7758258, I64

: default name tags "Zipf": [ZipfDataLoader](#)

8.170.2 Member Function Documentation

8.170.2.1 `getData()`

```
torch::Tensor CANDY::ZipfDataLoader::getData ( ) [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.170.2.2 `getQuery()`

```
torch::Tensor CANDY::ZipfDataLoader::getQuery ( ) [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from [CANDY::AbstractDataLoader](#).

8.170.2.3 `setConfig()`

```
bool CANDY::ZipfDataLoader::setConfig (
    INTELLI::ConfigMapPtr cfg ) [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

<i>cfg</i>	The config map
------------	----------------

Returns

bool whether the config is successfully set

Note

Reimplemented from [CANDY::AbstractDataLoader](#).

The documentation for this class was generated from the following files:

- [include/DataLoader/ZipfDataLoader.h](#)
- [src/DataLoader/ZipfDataLoader.cpp](#)

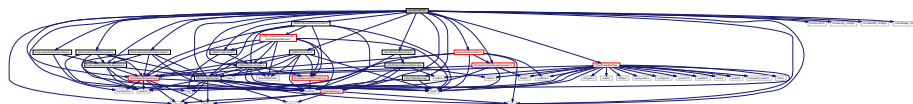
Chapter 9

File Documentation

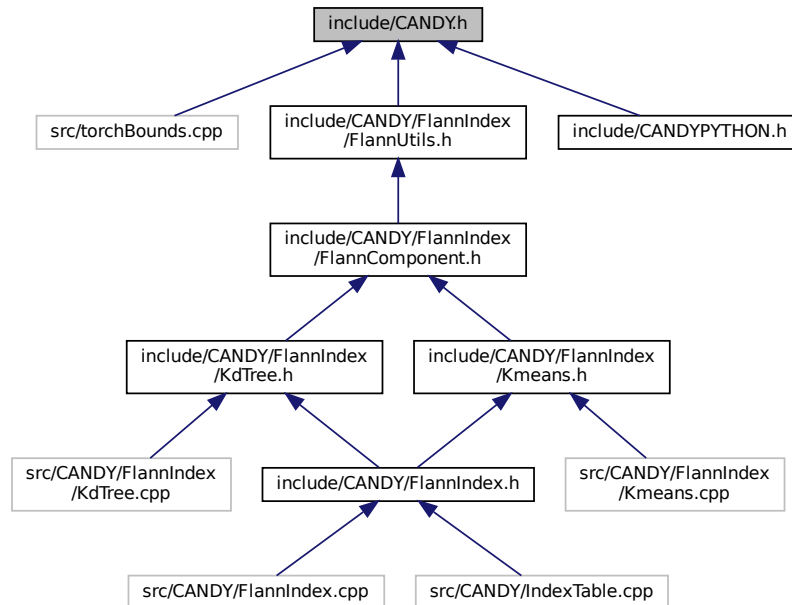
9.1 include/CANDY.h File Reference

```
#include <torch/torch.h>
#include <iostream>
#include <torch/script.h>
#include <string>
#include <memory>
#include <CANDY/IndexTable.h>
#include <CANDY/AbstractIndex.h>
#include <CANDY/FlatIndex.h>
#include <CANDY/ParallelPartitionIndex.h>
#include <include/ray_config.h>
#include <DataLoader/AbstractDataLoader.h>
#include <DataLoader/DataLoaderTable.h>
#include <DataLoader/RandomDataLoader.h>
#include <DataLoader/FVECSDataLoader.h>
#include <include/hdf5_config.h>
#include <Utils/ConfigMap.hpp>
#include <Utils/Meters/MeterTable.h>
#include <Utils/C20Buffers.hpp>
#include <Utils/ThreadPerf.hpp>
#include <include/papi_config.h>
#include <Utils/IntelliLog.h>
#include <Utils/UtilityFunctions.h>
#include <Utils/IntelliTensorOP.hpp>
#include <Utils/IntelliTimeStampGenerator.h>
```

Include dependency graph for CANDY.h:



This graph shows which files directly or indirectly include this file:



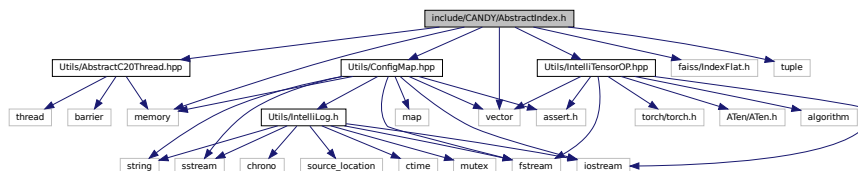
9.2 include/CANDY/AbstractIndex.h File Reference

```

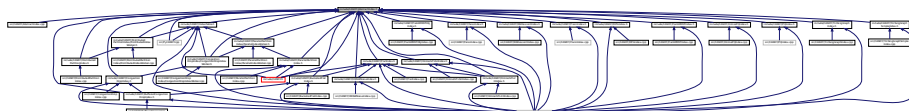
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
#include <tuple>

```

Include dependency graph for AbstractIndex.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::AbstractIndex](#)
The abstract class of an index approach.

Macros

- #define [newAbstractIndex](#) std::make_shared<[CANDY::AbstractIndex](#)>
(Macro) To creat a new *AbstractIndex* shared pointer.

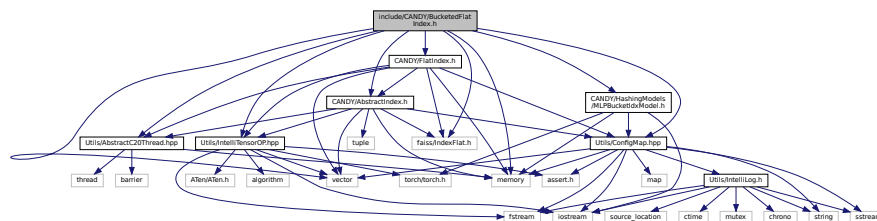
Typedefs

- typedef std::shared_ptr< class [CANDY::AbstractIndex](#) > [CANDY::AbstractIndexPtr](#)
The class to describe a shared pointer to *AbstractIndex*.

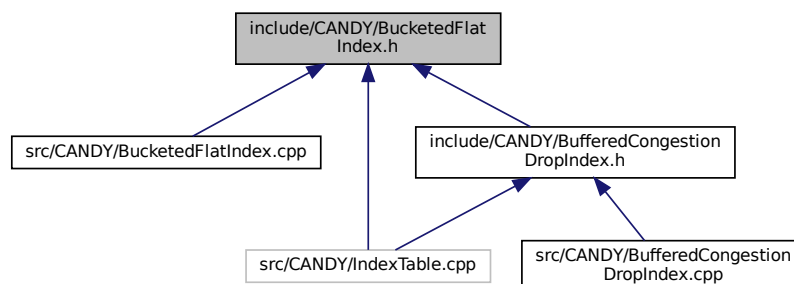
9.3 include/CANDY/BucketedFlatIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
#include <CANDY/AbstractIndex.h>
#include <CANDY/FlatIndex.h>
#include <CANDY/HashingModels/MLPBucketIdxModel.h>
```

Include dependency graph for `BucketedFlatIndex.h`:



This graph shows which files directly or indirectly include this file:



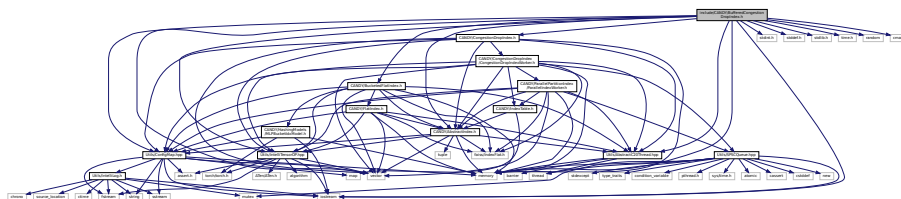
Classes

- class [CANDY::BucketedFlatIndex](#)
The class of splitting similar vectors into fixed number of buckets, each bucket is managed by [FlatIndex](#).
- #define [newBucketedFlatIndex](#) std::make_shared<[CANDY::BucketedFlatIndex](#)>
(Macro) To creat a new [BucketedFlatIndex](#) shared pointer.
- typedef std::shared_ptr< class [CANDY::BucketedFlatIndex](#) > [CANDY::BucketedFlatIndexPtr](#)
The class to describe a shared pointer to [BucketedFlatIndex](#).

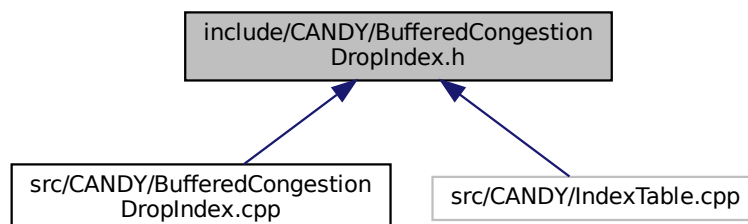
9.4 include/CANDY/BufferedCongestionDropIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/AbstractIndex.h>
#include <CANDY/CongestionDropIndex.h>
#include <CANDY/BucketedFlatIndex.h>
#include <stdint.h>
#include <stddef.h>
#include <stdlib.h>
#include <time.h>
#include <random>
#include <cmath>
#include <iostream>
```

Include dependency graph for BufferedCongestionDropIndex.h:



This graph shows which files directly or indirectly include this file:



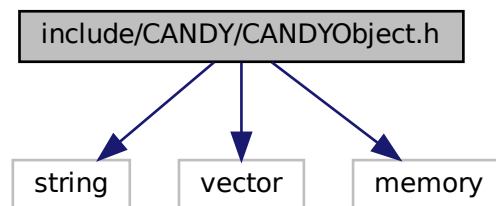
Classes

- class [CANDY::BufferedCongestionDropIndex](#)
Similar to [CongestionDropIndex](#), but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under [BucketedFlatIndex](#) More detailed description with an image:
- #define [newBufferedCongestionDropIndex](#) std::make_shared<[CANDY::BufferedCongestionDropIndex](#)>
(Macro) To creat a new [BufferedCongestionDropIndex](#) shared pointer.
- typedef std::shared_ptr< class [CANDY::BufferedCongestionDropIndex](#) > [CANDY::BufferedCongestionDropIndexPtr](#)
The class to describe a shared pointer to [BufferedCongestionDropIndex](#).

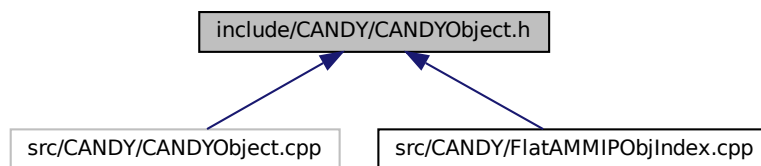
9.5 include/CANDY/CANDYObject.h File Reference

```
#include <string>
#include <vector>
#include <memory>
```

Include dependency graph for CANDYObject.h:



This graph shows which files directly or indirectly include this file:

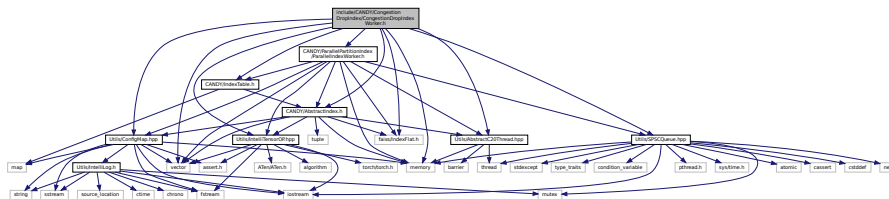


Classes

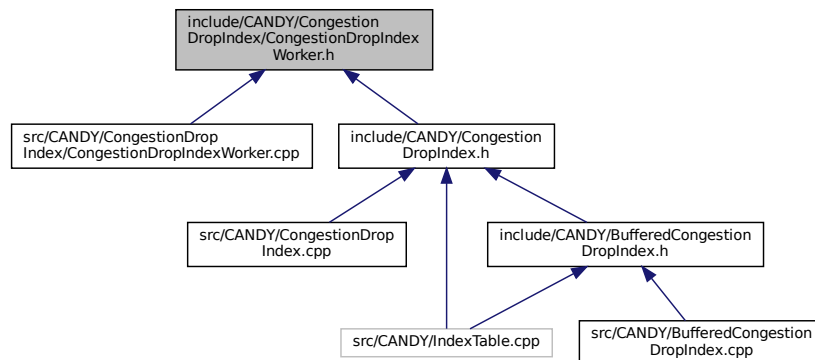
- class [CANDY::CANDYObject](#)
*A generic object class to link string or void * pointers.*
- #define [newCANDYObject](#) std::make_shared<[CANDY::CANDYObject](#)>
- typedef std::shared_ptr< class [CANDY::CANDYObject](#) > [CANDY::CANDYObjectPtr](#)
The class to describe a shared pointer to [CANDYObject](#).

9.7 include/CANDY/CongestionDropIndex/CongestionDropIndexWorker.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/IndexTable.h>
#include <Utils/SPSCQueue.hpp>
#include <CANDY/AbstractIndex.h>
#include <faiss/IndexFlat.h>
#include <CANDY/ParallelPartitionIndex/ParallelIndexWorker.h>
Include dependency graph for CongestionDropIndexWorker.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::CongestionDropIndexWorker](#)

A worker class to container bottom indexings, will just drop new element if congestion occurs.

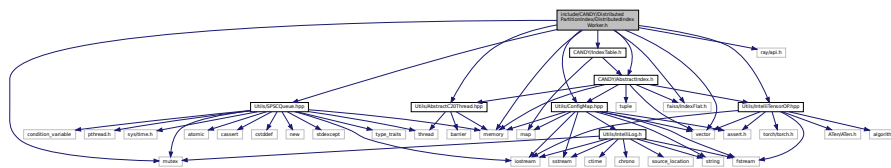
Macros

- #define [newCongestionDropIndexWorker](#) std::make_shared<[CANDY::CongestionDropIndexWorker](#)>
(Macro) To creat a new CongestionDropIndexWorker shared pointer.

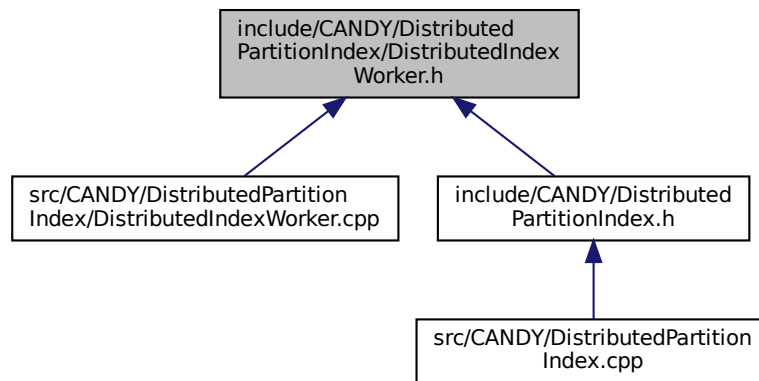
9.9 include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/IndexTable.h>
#include <Utils/SPSCQueue.hpp>
#include <CANDY/AbstractIndex.h>
#include <faiss/IndexFlat.h>
#include <ray/api.h>
#include <mutex>
```

Include dependency graph for DistributedIndexWorker.h:



This graph shows which files directly or indirectly include this file:



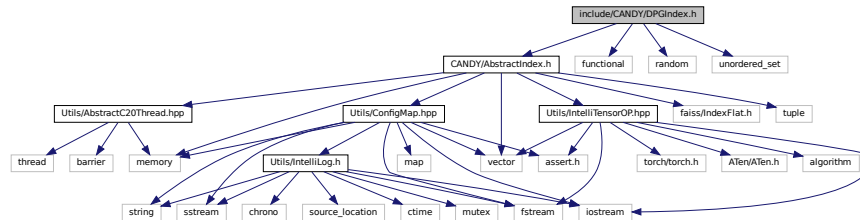
Classes

- class [CANDY::DIW_RayWrapper](#)
the ray wrapper of [DistributedIndexWorker](#), most of its function will be ray-remote
- class [CANDY::DistributedIndexWorker](#)
A worker class of parallel index thread.
- #define [newDistributedIndexWorker](#) std::make_shared<[CANDY::DistributedIndexWorker](#)>
(Macro) To creat a new [DistributedIndexWorker](#) shared pointer.
- typedef std::shared_ptr< class [CANDY::DistributedIndexWorker](#) > [CANDY::DistributedIndexWorkerPtr](#)
The class to describe a shared pointer to [DistributedIndexWorker](#).

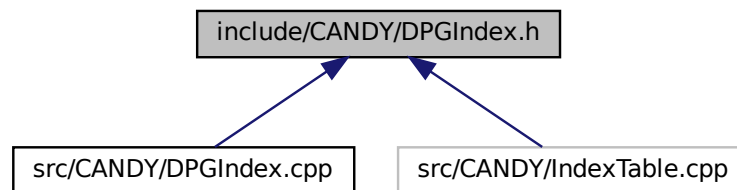
9.10 include/CANDY/DPGIndex.h File Reference

```
#include <CANDY/AbstractIndex.h>
#include <functional>
#include <random>
#include <unordered_set>
```

Include dependency graph for DPGIndex.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::DPGIndex](#)

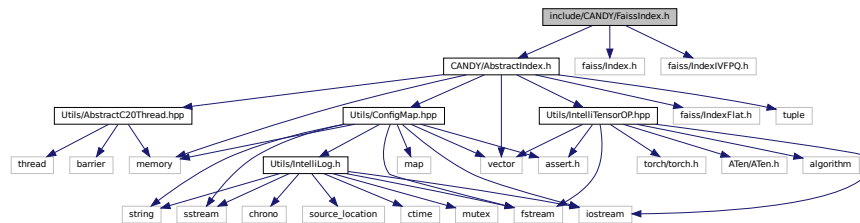
A hierarchical algorithm based on a data structure consistent with [NNDescentIndex](#), the subgraph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic graph still uses the NNDescent algorithm in this implementation.

- struct [CANDY::DPGIndex::Neighbor](#)
- struct [CANDY::DPGIndex::NhoodLayer0](#)
- struct [CANDY::DPGIndex::NhoodLayer1](#)

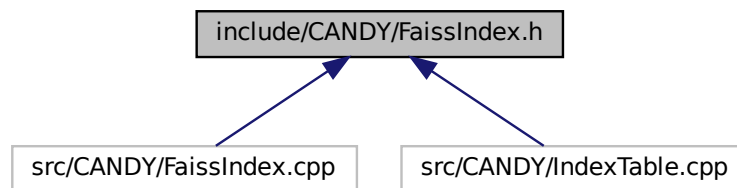
- #define [newDPGIndex](#) std::make_shared<[CANDY::DPGIndex](#)>
(Macro) To creat a new DPGIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::DPGIndex](#) > [CANDY::DPGIndexPtr](#)
The class to describe a shared pointer to [DPGIndex](#).

9.11 include/CANDY/FaissIndex.h File Reference

```
#include <CANDY/AbstractIndex.h>
#include <faiss/Index.h>
#include <faiss/IndexIVFPQ.h>
Include dependency graph for FaissIndex.h:
```



This graph shows which files directly or indirectly include this file:



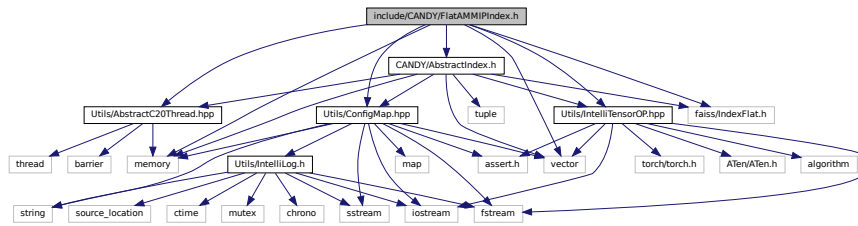
Classes

- class [CANDY::FaissIndex](#)
The class of converting faiss index api into rania index style.
- #define [newFaissIndex](#) std::make_shared<[CANDY::FaissIndex](#)>
(Macro) To creat a new FaissIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::FaissIndex](#) > [CANDY::FaissIndexPtr](#)
The class to describe a shared pointer to FaissIndexPtr.

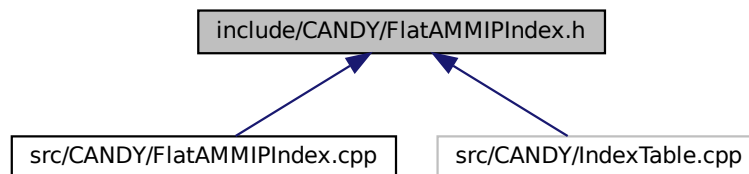
9.12 include/CANDY/FlatAMMIPIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
```

```
#include <faiss/IndexFlat.h>
#include <CANDY/AbstractIndex.h>
Include dependency graph for FlatAMMIPIndex.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::FlatAMMIPIndex](#)

The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance.

- #define [newFlatAMMIPIndex](#) std::make_shared<CANDY::FlatAMMIPIndex>
(Macro) To creat a new FlatAMMIPIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::FlatAMMIPIndex](#) > [CANDY::FlatAMMIPIndexPtr](#)
The class to describe a shared pointer to [FlatAMMIPIndex](#).

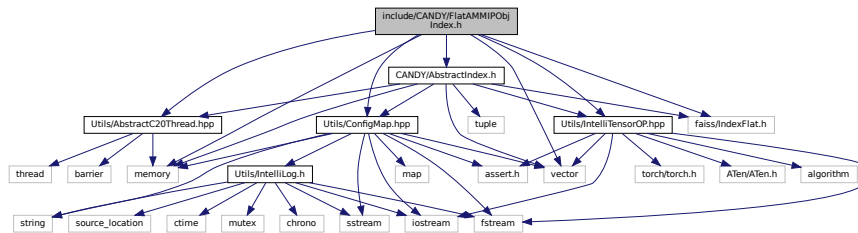
9.13 include/CANDY/FlatAMMIPObjIndex.h File Reference

```
#include <Uutils/AbstractC20Thread.hpp>
#include <Uutils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Uutils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
```

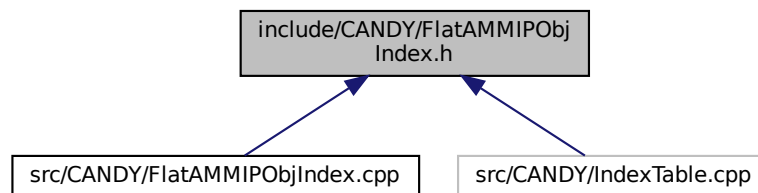


```
#include <CANDY/AbstractIndex.h>
```

Include dependency graph for FlatAMMIObjIndex.h:



This graph shows which files directly or indirectly include this file:



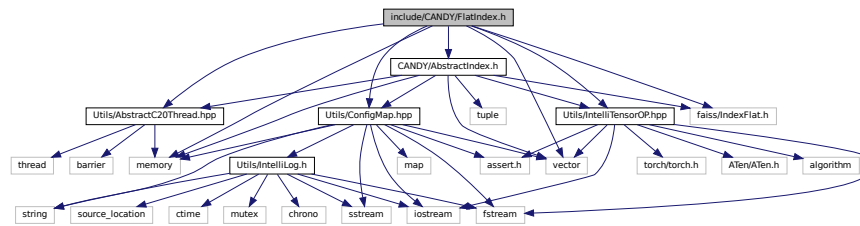
Classes

- class [CANDY::FlatAMMIObjIndex](#)
Similar to [FlatAMMIIndex](#), but additionally has object storage (currently only string)
- #define [newFlatAMMIObjIndex](#) std::make_shared<CANDY::FlatAMMIObjIndex>
(Macro) To creat a new FlatAMMIObjIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::FlatAMMIObjIndex](#) > [CANDY::FlatAMMIObjIndexPtr](#)
The class to describe a shared pointer to [FlatAMMIObjIndex](#).

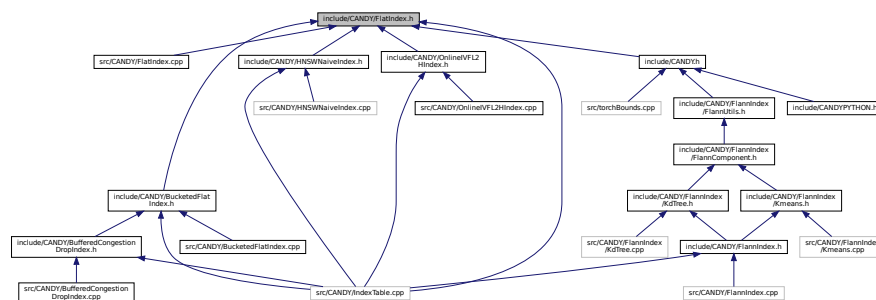
9.14 include/CANDY/FlatIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
```

```
#include <CANDY/AbstractIndex.h>
Include dependency graph for FlatIndex.h:
```



This graph shows which files directly or indirectly include this file:

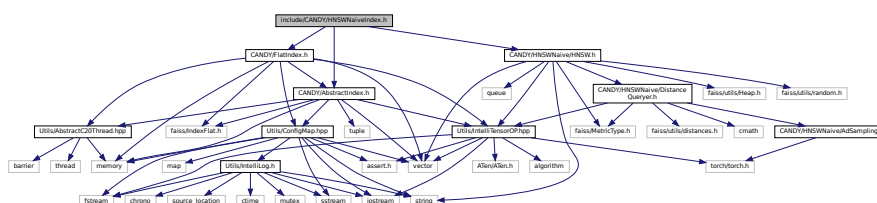


Classes

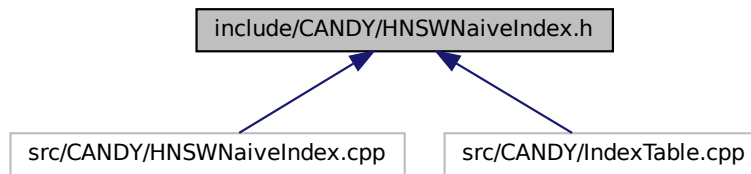
- class `CANDY::FlatIndex`
The class of a flat index approach, using brutal force management.
- #define `newFlatIndex` `std::make_shared<CANDY::FlatIndex>`
(Macro) To create a new FlatIndex shared pointer.
- typedef `std::shared_ptr< class CANDY::FlatIndex > CANDY::FlatIndexPtr`
The class to describe a shared pointer to FlatIndex.

9.15 include/CANDY/HNSWNaiveIndex.h File Reference

```
#include <CANDY/AbstractIndex.h>
#include <CANDY/FlatIndex.h>
#include <CANDY/HNSWNaive/HNSW.h>
Include dependency graph for HNSWNaiveIndex.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::HNSWNaiveIndex](#)

The class of a *HNSW* index approach, store the data in each vertex.

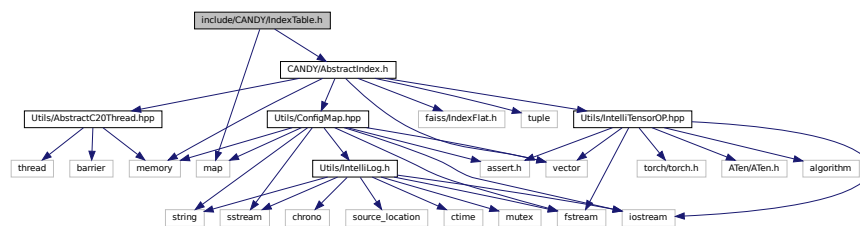
Macros

- #define **newHNSWNaiveIndex** std::make_shared<[CANDY::HNSWNaiveIndex](#)>
- #define **newNSWIndex** std::make_shared<[CANDY::HNSWNaiveIndex](#)>

9.16 include/CANDY/IndexTable.h File Reference

```
#include <map>
#include <CANDY/AbstractIndex.h>
```

Include dependency graph for IndexTable.h:



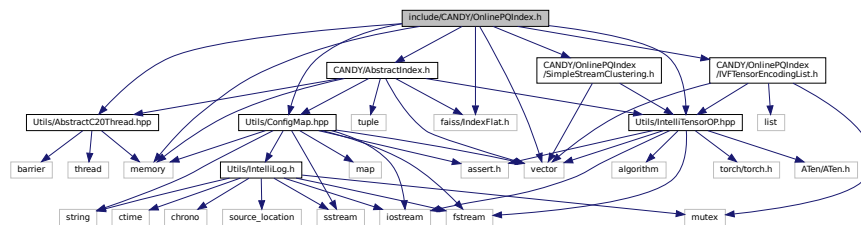
Classes

- class [CANDY::NNDescentIndex](#)
An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized.
- struct [CANDY::NNDescentIndex::Neighbor](#)
- struct [CANDY::NNDescentIndex::Nhoo](#)
- #define [newNNDescentIndex](#) std::make_shared<[CANDY::NNDescentIndex](#)>
(Macro) To creat a new NNDescentIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::NNDescentIndex](#) > [CANDY::NNDescentIndexPtr](#)
The class to describe a shared pointer to [NNDescentIndex](#).

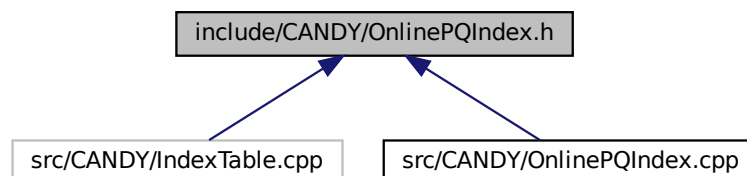
9.18 include/CANDY/OnlinePQIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
#include <CANDY/AbstractIndex.h>
#include <CANDY/OnlinePQIndex/SimpleStreamClustering.h>
#include <CANDY/OnlinePQIndex/IVFTensorEncodingList.h>
```

Include dependency graph for OnlinePQIndex.h:



This graph shows which files directly or indirectly include this file:



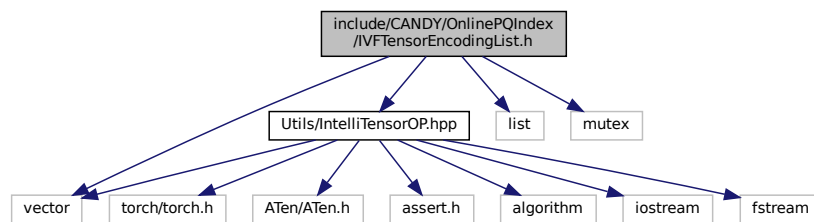
Classes

- class [CANDY::OnlinePQIndex](#)
The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers.
- `#define newOnlinePQIndex std::make_shared<CANDY::OnlinePQIndex>`
(Macro) To creat a new OnlinePQIndex shared pointer.
- `typedef std::shared_ptr< class CANDY::OnlinePQIndex > CANDY::OnlinePQIndexPtr`
The class to describe a shared pointer to [OnlinePQIndex](#).

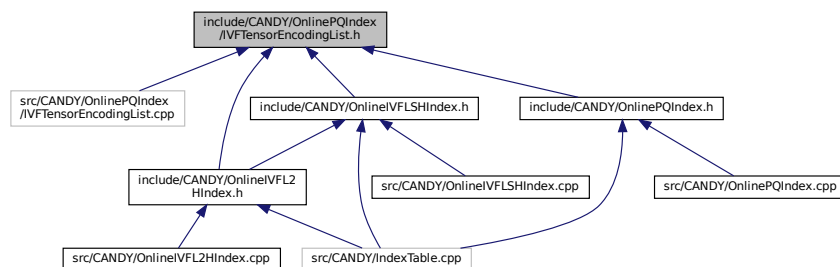
9.19 include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h File Reference

```
#include <Utils/IntelliTensorOP.hpp>
#include <vector>
#include <list>
#include <mutex>
```

Include dependency graph for IVFTensorEncodingList.h:



This graph shows which files directly or indirectly include this file:



Classes

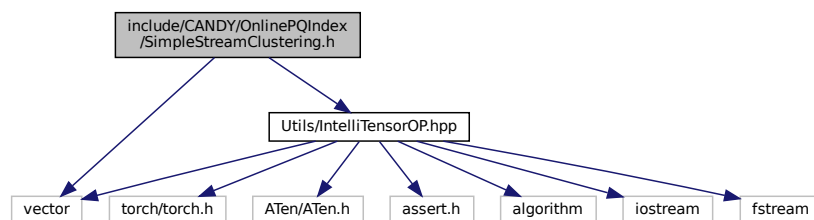
- class [CANDY::IVFListCell](#)
a cell of row tensor pointers which have the same code
- class [CANDY::IVFListBucket](#)
a bucket of multiple [IVFListCell](#)
- class [CANDY::IVFTensorEncodingList](#)
The inverted file (IVF) list to organize tensor and its encodings.
- #define [newIVFListCell](#) `make_shared<CANDY::IVFListCell>`
(Macro) To creat a new [newIVFListCell](#) under shared pointer.
- #define [newIVFListBucket](#) `make_shared<CANDY::IVFListBucket>`
(Macro) To creat a new [IVFListBucket](#) under shared pointer.
- typedef `std::shared_ptr< CANDY::IVFListCell > CANDY::IVFListCellPtr`
The class to describe a shared pointer to [IVFListCell](#).
- typedef `std::shared_ptr< CANDY::IVFListBucket > CANDY::IVFListBucketPtr`
The class to describe a shared pointer to [IVFListBucket](#).

9.20 include/CANDY/OnlinePQIndex/SimpleStreamClustering.h File Reference

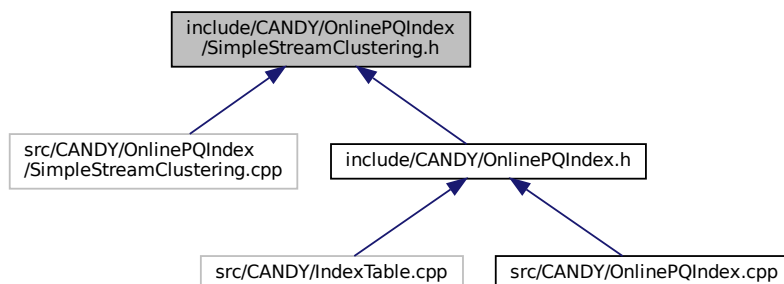
```
#include <Utils/IntelliTensorOP.hpp>
```

```
#include <vector>
```

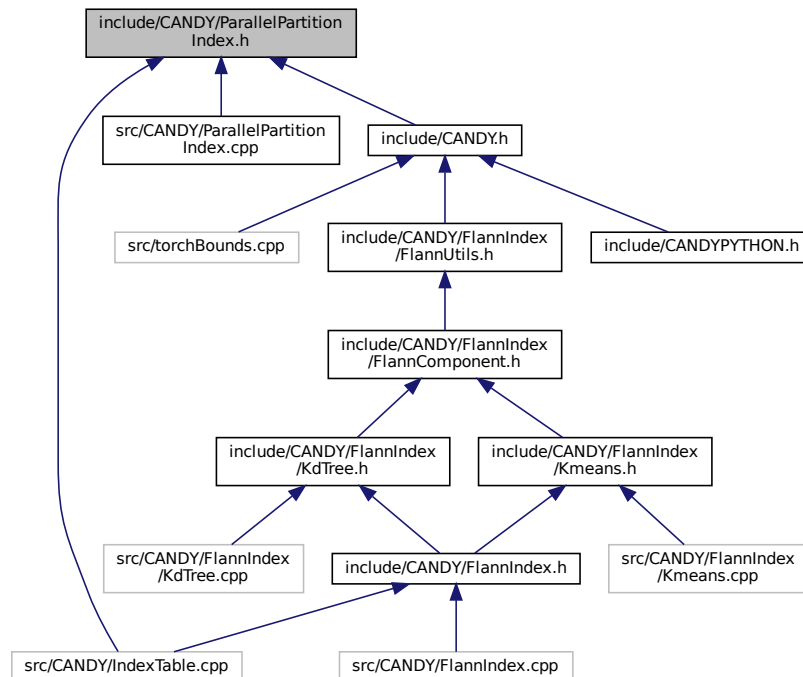
Include dependency graph for SimpleStreamClustering.h:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::ParallelPartitionIndex](#)
A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature.
- #define [newParallelPartitionIndex](#) std::make_shared<[CANDY::ParallelPartitionIndex](#)>
(Macro) To creat a new ParallelPartitionIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::ParallelPartitionIndex](#) > [CANDY::ParallelPartitionIndexPtr](#)
The class to describe a shared pointer to [ParallelPartitionIndex](#).

9.22 include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h File Reference

```

#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/IndexTable.h>
#include <Utils/SPSCQueue.hpp>
#include <CANDY/AbstractIndex.h>

```


Typedefs

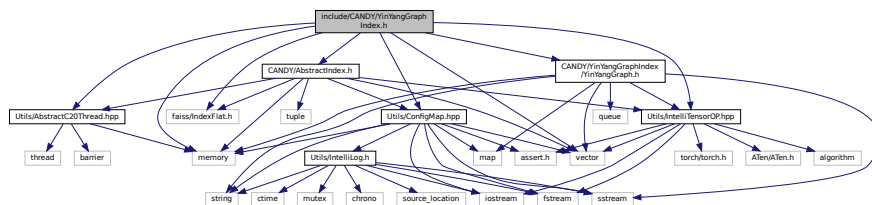
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< torch::Tensor > > **CANDY::TensorQueuePtr**
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< [CANDY::TensorIdxPair](#) > > **CANDY::TensorIdxQueuePtr**
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< [CANDY::TensorListIdxPair](#) > > **CANDY::TensorListIdxQueuePtr**
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< int64_t > > **CANDY::CmdQueuePtr**
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< [CANDY::TensorStrPair](#) > > **CANDY::TensorStrQueuePtr**
- typedef std::shared_ptr< [INTELLI::SPSCQueue](#)< [CANDY::TensorStrVecPair](#) > > **CANDY::TensorStrVecQueuePtr**
- typedef std::shared_ptr< class [CANDY::ParallelIndexWorker](#) > [CANDY::ParallelIndexWorkerPtr](#)

The class to describe a shared pointer to [ParallelIndexWorker](#).

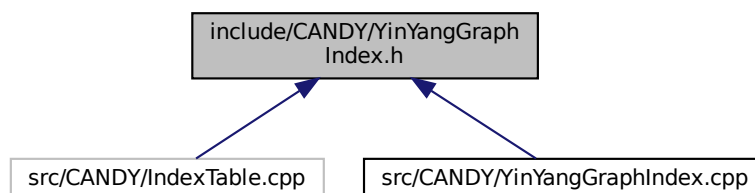
9.23 include/CANDY/YinYangGraphIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
#include <CANDY/AbstractIndex.h>
#include <CANDY/YinYangGraphIndex/YinYangGraph.h>
```

Include dependency graph for YinYangGraphIndex.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::YinYangGraphIndex](#)

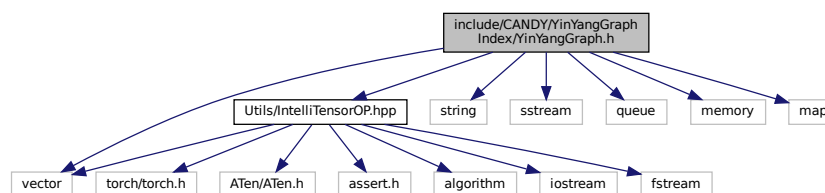
The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph.

- #define [newYinYangGraphIndex](#) std::make_shared<[CANDY::YinYangGraphIndex](#)>
(Macro) To creat a new YinYangGraphIndex shared pointer.
- typedef std::shared_ptr< class [CANDY::YinYangGraphIndex](#) > [CANDY::YinYangGraphIndexPtr](#)
The class to describe a shared pointer to [YinYangGraphIndex](#).

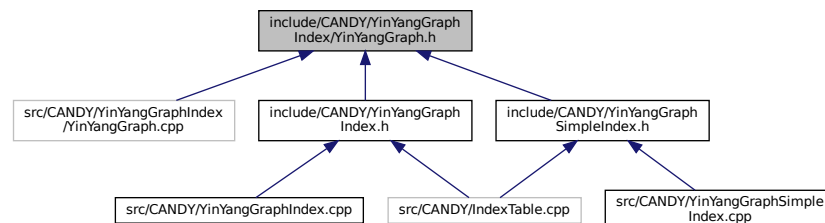
9.24 include/CANDY/YinYangGraphIndex/YinYangGraph.h File Reference

```
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <string>
#include <sstream>
#include <queue>
#include <memory>
#include <map>
```

Include dependency graph for YinYangGraph.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::YinYangGraph_DistanceFunctions](#)
- class [CANDY::YinYangVertex](#)

The class of a *YinYangVertex*, storing the data in each vertex.
- class [CANDY::YinYangVertexMap](#)
- class [CANDY::YinYangGraph_ListCell](#)

a cell of an ending *YinYangVertex*
- class [CANDY::YinYangGraph_ListBucket](#)

a bucket of multiple *YinYangGraph_ListCell*
- class [CANDY::YinYangGraph](#)

The top class of yinyang graph, containing ivf list and critical graph information.

Macros

- #define [newYinYangVertex](#) make_shared<[CANDY::YinYangVertex](#)>

(Macro) To creat a new *YinYangVertex* under shared pointer.
- #define [newYinYangGraph_ListCell](#) make_shared<[CANDY::YinYangGraph_ListCell](#)>

(Macro) To creat a new *newYinYangGraph_ListCell* under shared pointer.
- #define [newYinYangGraph_ListBucket](#) make_shared<[CANDY::YinYangGraph_ListBucket](#)>

(Macro) To creat a new *YinYangGraph_ListBucket* under shared pointer.

Typedefs

- using [CANDY::floatDistanceFunction_t](#) = float(*) (const torch::Tensor &, const torch::Tensor &)
- typedef std::shared_ptr< [CANDY::YinYangVertex](#) > [CANDY::YinYangVertexPtr](#)

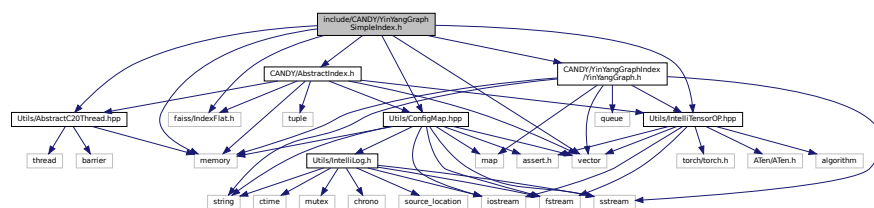
The class to describe a shared pointer to *YinYangVertex*.
- typedef std::shared_ptr< [CANDY::YinYangGraph_ListCell](#) > [CANDY::YinYangGraph_ListCellPtr](#)

The class to describe a shared pointer to *YinYangGraph_ListCell*.
- typedef std::shared_ptr< [CANDY::YinYangGraph_ListBucket](#) > [CANDY::YinYangGraph_ListBucketPtr](#)

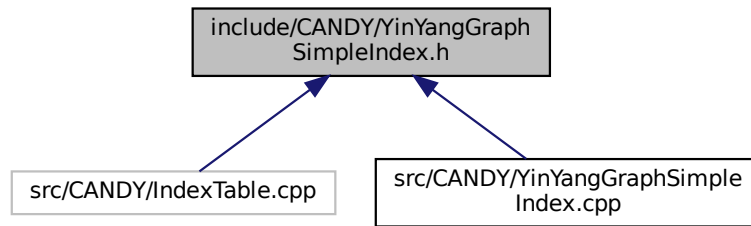
The class to describe a shared pointer to *YinYangGraph_ListBucket*.

9.25 include/CANDY/YinYangGraphSimpleIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
#include <CANDY/AbstractIndex.h>
#include <CANDY/YinYangGraphIndex/YinYangGraph.h>
Include dependency graph for YinYangGraphSimpleIndex.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CANDY::YinYangGraphSimpleIndex](#)

The class of indexing using a simple yinyang graph, there is no LSH search is only within the linked yinyanggraph.

- #define [newYinYangGraphSimpleIndex](#) std::make_shared<[CANDY::YinYangGraphSimpleIndex](#)>

(Macro) To create a new YinYangGraphSimpleIndex shared pointer.

- typedef std::shared_ptr< class [CANDY::YinYangGraphSimpleIndex](#) > [CANDY::YinYangGraphSimpleIndexPtr](#)

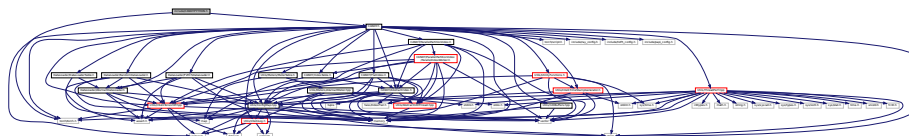
The class to describe a shared pointer to [YinYangGraphSimpleIndex](#).

9.26 include/CANDYPYTHON.h File Reference

```
#include <CANDY.h>
```

```
#include <torch/torch.h>
```

Include dependency graph for CANDYPYTHON.h:



Classes

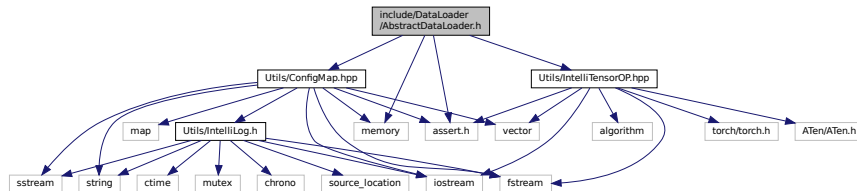
- class [CANDY::Rania_Python](#)

The python bounding functions.

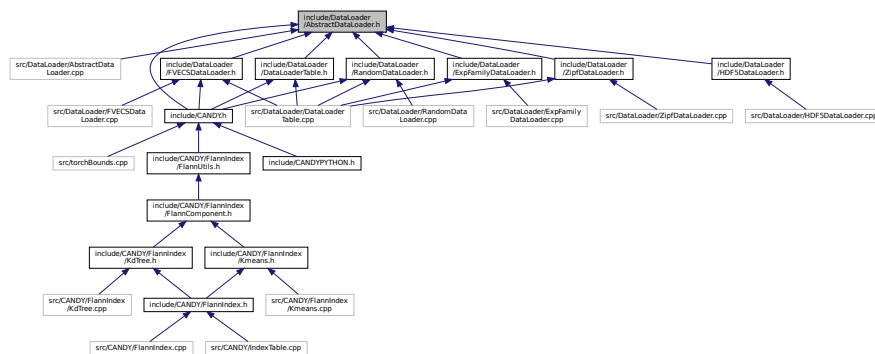
9.27 include/DataLoader/AbstractDataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
```

Include dependency graph for AbstractDataLoader.h:



This graph shows which files directly or indirectly include this file:



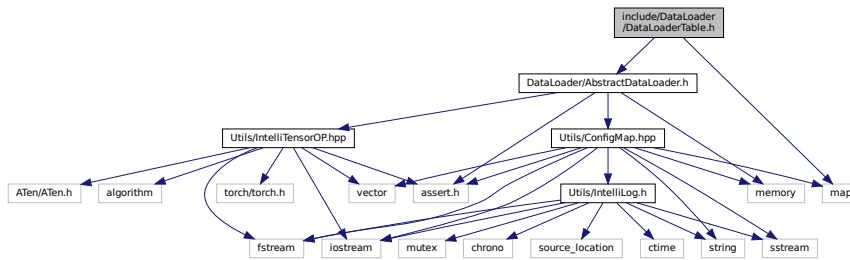
Classes

- class `CANDY::AbstractDataLoader`
The abstract class of data loader, parent for all loaders.
- `#define newAbstractDataLoader std::make_shared<CANDY::AbstractDataLoader>`
(Macro) To create a new AbstractDataLoader under shared pointer.
- `typedef std::shared_ptr< class CANDY::AbstractDataLoader > CANDY::AbstractDataLoaderPtr`
The class to describe a shared pointer to AbstractDataLoader.

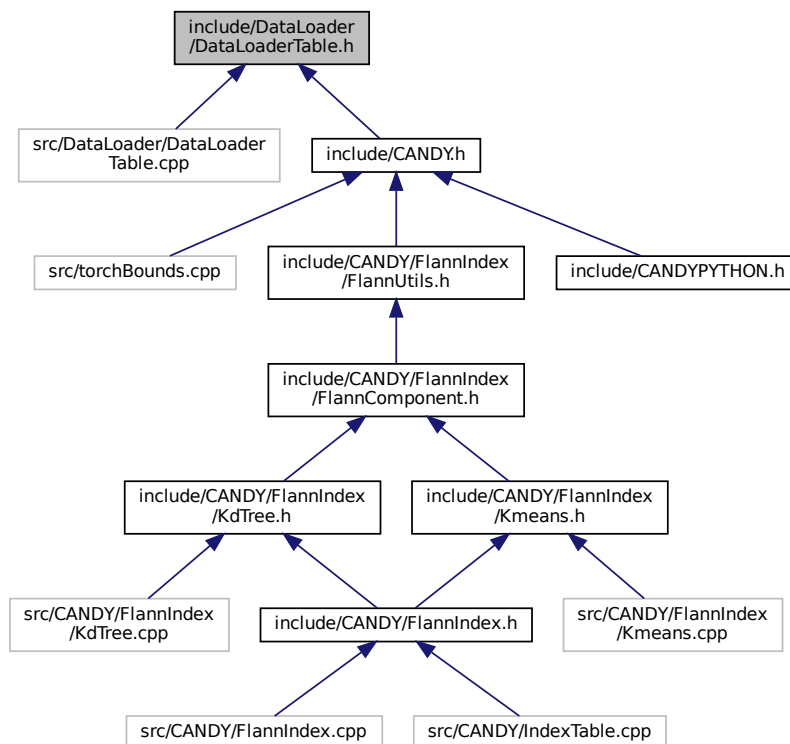
9.28 include/DataLoader/DataLoaderTable.h File Reference

```
#include <map>
#include <DataLoader/AbstractDataLoader.h>
```

Include dependency graph for DataLoaderTable.h:



This graph shows which files directly or indirectly include this file:



Classes

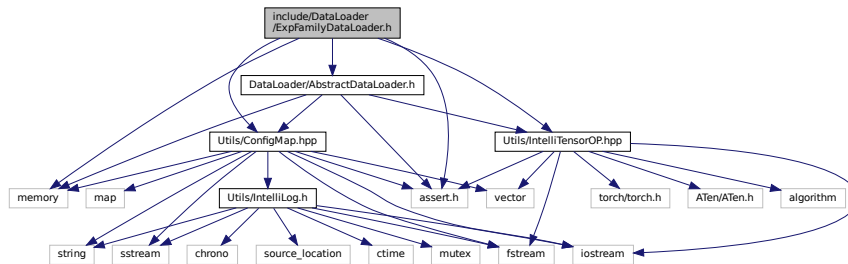
- class [CANDY::DataLoaderTable](#)
The table class to index all Data loaders.

Macros

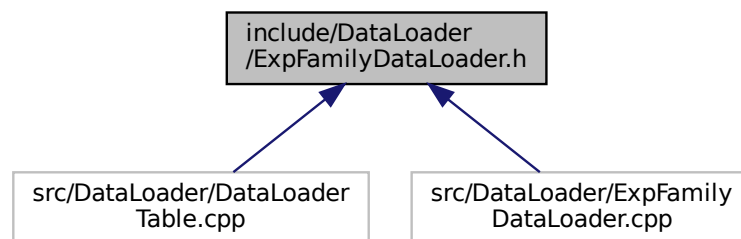
- #define [newDataLoaderTable](#) std::make_shared<[CANDY::DataLoaderTable](#)>
(Macro) To creat a new DataLoaderTable under shared pointer.

9.29 include/DataLoader/ExpFamilyDataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
#include <DataLoader/AbstractDataLoader.h>
Include dependency graph for ExpFamilyDataLoader.h:
```



This graph shows which files directly or indirectly include this file:



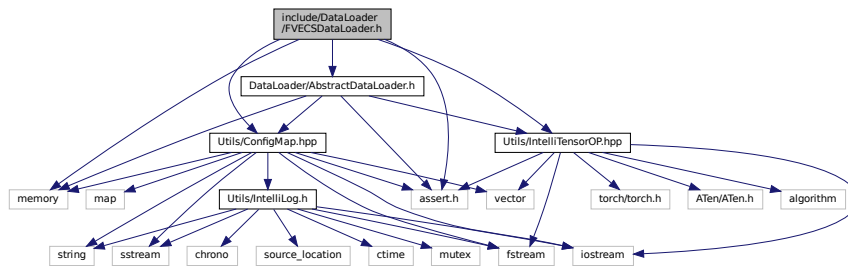
Classes

- class [CANDY::ExpFamilyDataLoader](#)
The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta.
- #define [newExpFamilyDataLoader](#) std::make_shared<CANDY::ExpFamilyDataLoader>
(Macro) To creat a new ExpFamilyDataLoader under shared pointer.
- typedef std::shared_ptr< class [CANDY::ExpFamilyDataLoader](#) > [CANDY::ExpFamilyDataLoaderPtr](#)
The class to describe a shared pointer to [ExpFamilyDataLoader](#).

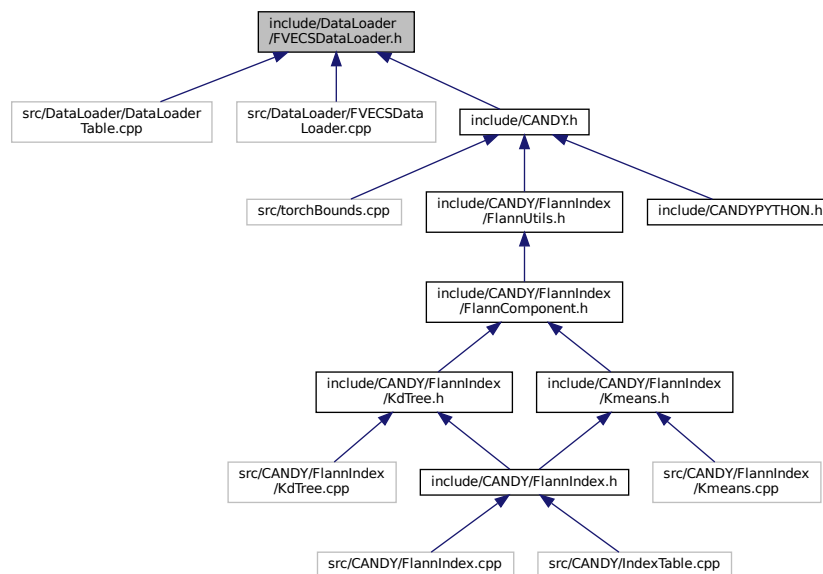
9.30 include/DataLoader/FVECSDataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
#include <DataLoader/AbstractDataLoader.h>
```

Include dependency graph for FVECSDataLoader.h:



This graph shows which files directly or indirectly include this file:

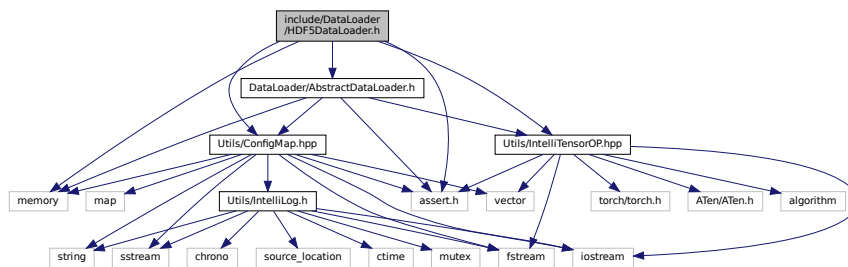


Classes

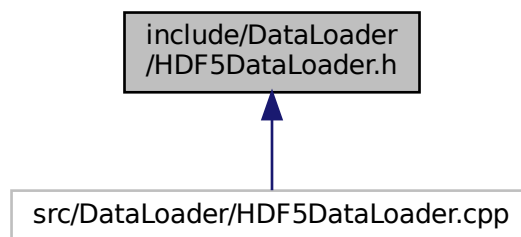
- class [CANDY::FVECSDataLoader](#)
The class for loading *.fvecs data.
- `#define newFVECSDataLoader std::make_shared<CANDY::FVECSDataLoader>`
(Macro) To creat a new FVECSDataLoader under shared pointer.
- typedef `std::shared_ptr< class CANDY::FVECSDataLoader > CANDY::FVECSDataLoaderPtr`
The class to describe a shared pointer to FVECSDataLoader.

9.31 include/DataLoader/HDF5DataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
#include <DataLoader/AbstractDataLoader.h>
Include dependency graph for HDF5DataLoader.h:
```



This graph shows which files directly or indirectly include this file:



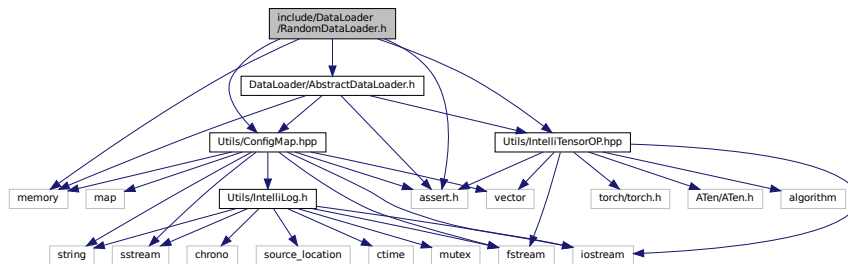
Classes

- class [CANDY::HDF5DataLoader](#)
The class for loading *.hdf5 or *.h5 file, as specified in <https://github.com/HDFGroup/hdf5>.
- #define [newHDF5DataLoader](#) std::make_shared<[CANDY::HDF5DataLoader](#)>
(Macro) To creat a new HDF5DataLoader under shared pointer.
- typedef std::shared_ptr< class [CANDY::HDF5DataLoader](#) > [CANDY::HDF5DataLoaderPtr](#)
The class to describe a shared pointer to [HDF5DataLoader](#).

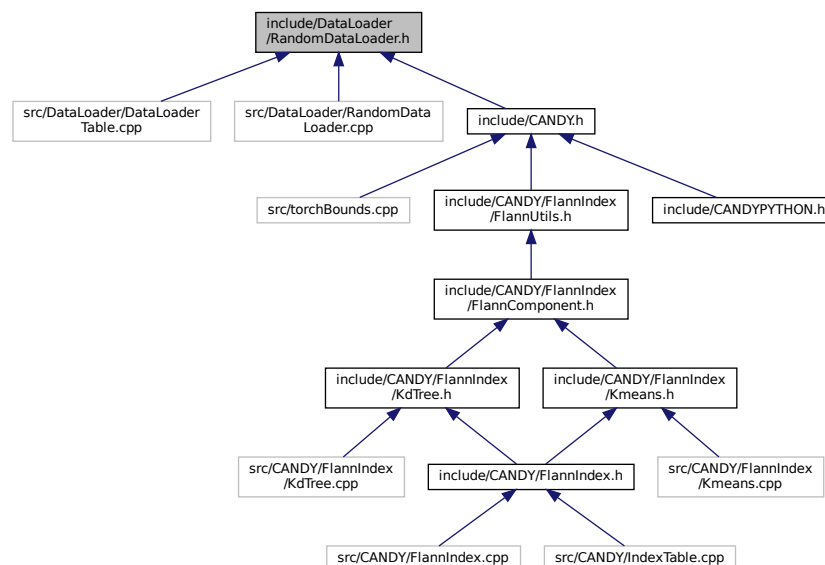
9.32 include/DataLoader/RandomDataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
#include <DataLoader/AbstractDataLoader.h>
```

Include dependency graph for RandomDataLoader.h:



This graph shows which files directly or indirectly include this file:

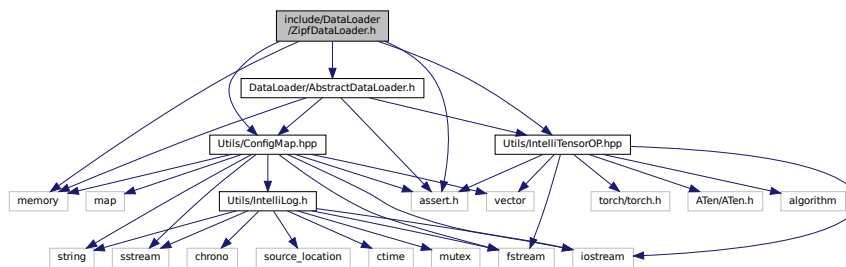


Classes

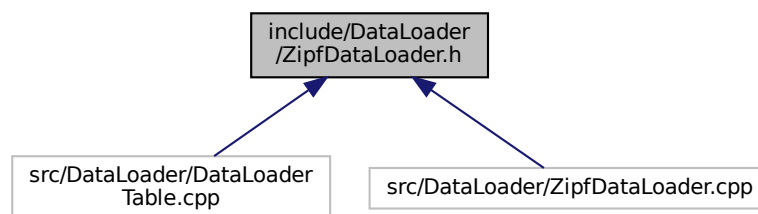
- class [CANDY::RandomDataLoader](#)
The class of random data loader.
- `#define newRandomDataLoader std::make_shared<CANDY::RandomDataLoader>`
(Macro) To create a new RandomDataLoader under shared pointer.
- `typedef std::shared_ptr< class CANDY::RandomDataLoader > CANDY::RandomDataLoaderPtr`
The class to describe a shared pointer to [RandomDataLoader](#).

9.33 include/DataLoader/ZipfDataLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliTensorOP.hpp>
#include <assert.h>
#include <memory>
#include <DataLoader/AbstractDataLoader.h>
Include dependency graph for ZipfDataLoader.h:
```



This graph shows which files directly or indirectly include this file:



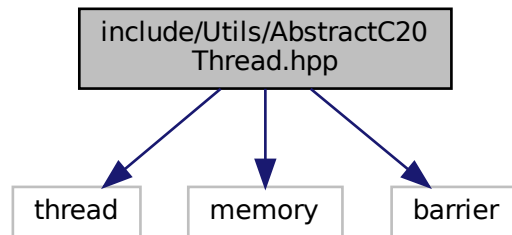
Classes

- class [CANDY::ZipfDataLoader](#)
The class to load zipf data.
- #define [newZipfDataLoader](#) std::make_shared<[CANDY::ZipfDataLoader](#)>
(Macro) To creat a new ZipfDataLoader under shared pointer.
- typedef std::shared_ptr< class [CANDY::ZipfDataLoader](#) > [CANDY::ZipfDataLoaderPtr](#)
The class to describe a shared pointer to [ZipfDataLoader](#).

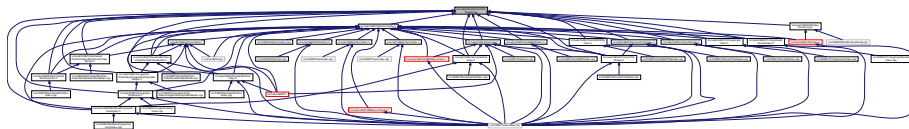
9.34 include/Utils/AbstractC20Thread.hpp File Reference

```
#include <thread>
#include <memory>
#include <barrier>
```

Include dependency graph for AbstractC20Thread.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [INTELLI::AbstractC20Thread](#)

The base class and abstraction of C++20 thread, and it can be derived into other threads.

Macros

- #define [newAbstractC20Thread](#) `std::make_shared<INTELLI::AbstractC20Thread>`

(Macro) To creat a new [newAbstractC20Thread](#) under shared pointer.

Typedefs

- typedef `std::shared_ptr< AbstractC20Thread >` [INTELLI::AbstractC20ThreadPtr](#)

The class to describe a shared pointer to [AbstractC20Thread](#).

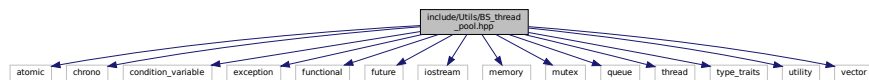
- typedef `std::shared_ptr< std::barrier<> >` [INTELLI::BarrierPtr](#)

9.35 include/Utils/BS_thread_pool.hpp File Reference

BS::thread_pool: a fast, lightweight, and easy-to-use C++17 thread pool library. This header file contains the entire library, including the main **BS::thread_pool** class and the helper classes **BS::multi_future**, **BS::blocks**, **BS::syncd_stream**, and **BS::timer**.

```
#include <atomic>
#include <chrono>
#include <condition_variable>
#include <exception>
#include <functional>
#include <future>
#include <iostream>
#include <memory>
#include <mutex>
#include <queue>
#include <thread>
#include <type_traits>
#include <utility>
#include <vector>
```

Include dependency graph for BS_thread_pool.hpp:



Classes

- class **BS::multi_future**< T >
A helper class to facilitate waiting for and/or getting the results of multiple futures at once.
- class **BS::blocks**< T1, T2, T >
A helper class to divide a range into blocks. Used by `parallelize_loop()` and `push_loop()`.
- class **BS::thread_pool**
A fast, lightweight, and easy-to-use C++17 thread pool class.
- class **BS::syncd_stream**
A helper class to synchronize printing to an output stream by different threads.
- class **BS::timer**
A helper class to measure execution time for benchmarking purposes.

Macros

- `#define BS_THREAD_POOL_VERSION "v3.3.0 (2022-08-03)"`

Typedefs

- using **BS::concurrency_t** = `std::invoke_result_t< decltype(std::thread::hardware_concurrency)>`
A convenient shorthand for the type of `std::thread::hardware_concurrency()`. Should evaluate to unsigned int.
- typedef `std::shared_ptr< thread_pool >` **BS::thread_pool_ptr**

9.35.1 Detailed Description

`BS::thread_pool`: a fast, lightweight, and easy-to-use C++17 thread pool library. This header file contains the entire library, including the main `BS::thread_pool` class and the helper classes `BS::multi_future`, `BS::blocks`, `BS::syncd_stream`, and `BS::timer`.

Author

Barak Shoshany (baraksh@gmail.com) (<http://baraksh.com>)

Version

3.3.0

Date

2022-08-03

Copyright

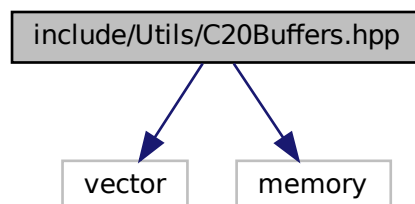
Copyright (c) 2022 Barak Shoshany. Licensed under the MIT license. If you found this project useful, please consider starring it on GitHub! If you use this library in software of any kind, please provide a link to the GitHub repository <https://github.com/bshoshany/thread-pool> in the source code and documentation. If you use this library in published research, please cite it as follows: Barak Shoshany, "A C++17 Thread Pool for High-Performance Scientific Computing", doi:10.5281/zenodo.4742687, arXiv:2105.00613 (May 2021)

9.36 include/Utils/C20Buffers.hpp File Reference

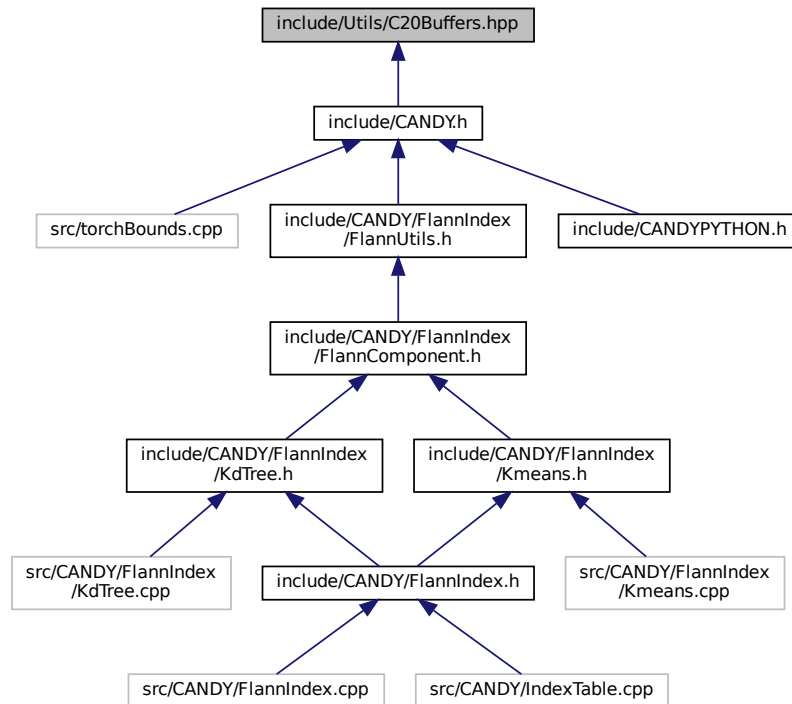
```
#include <vector>
```

```
#include <memory>
```

Include dependency graph for C20Buffers.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [INTELLI::C20Buffer< dataType >](#)

Macros

- `#define _UTILS_C20BUFFERS_HPP_`
- `#define ADB_memcpy(dst, src, size) memcpy(dst, src, size)`

9.37 include/Utils/ConfigMap.hpp File Reference

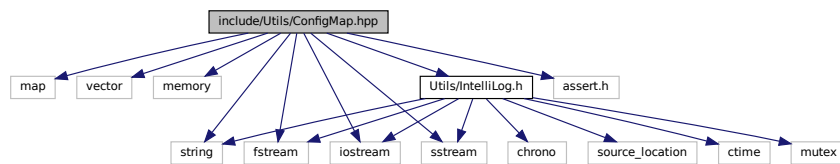
```

#include <map>
#include <vector>
#include <memory>
#include <string>
#include <fstream>
#include <iostream>
#include <sstream>
#include <assert.h>

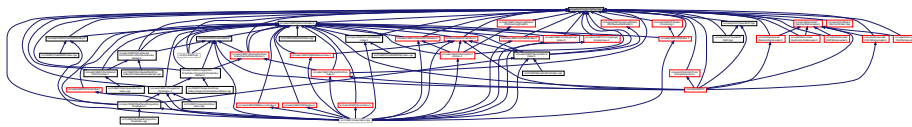
```

```
#include <Utils/IntelliLog.h>
```

Include dependency graph for ConfigMap.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [INTELLI::ConfigMap](#)

The unified map structure to store configurations in a key-value style.

Macros

- `#define _UTILS_CONFIGMAP_HPP_`
- `#define newConfigMap make_shared<INTELLI::ConfigMap>`

(Macro) To creat a new ConfigMap under shared pointer.

Typedefs

- `typedef std::shared_ptr< ConfigMap > INTELLI::ConfigMapPtr`

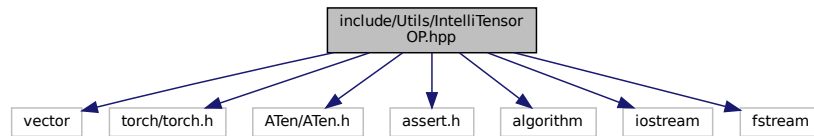
The class to describe a shared pointer to [ConfigMap](#).

9.38 include/Utils/IntelliTensorOP.hpp File Reference

```
#include <vector>
#include <torch/torch.h>
#include <ATen/ATen.h>
#include <assert.h>
#include <algorithm>
#include <iostream>
```

```
#include <fstream>
```

Include dependency graph for IntelliTensorOP.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [INTELLI::IntelliTensorOP](#)

Macros

- #define [newTensor](#) `make_shared<torch::Tensor>`
(Macro) To creat a new Tensor under shared pointer.

Typedefs

- typedef `std::shared_ptr< torch::Tensor >` [INTELLI::TensorPtr](#)
The class to describe a shared pointer to `torch::Tensor`.

9.39 include/Utils/IntelliTimeStampGenerator.h File Reference

```
#include <stdint.h>
```

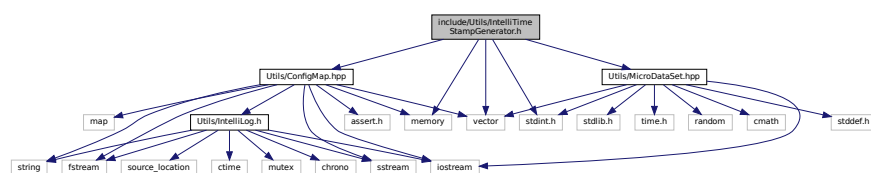
```
#include <vector>
```

```
#include <memory>
```

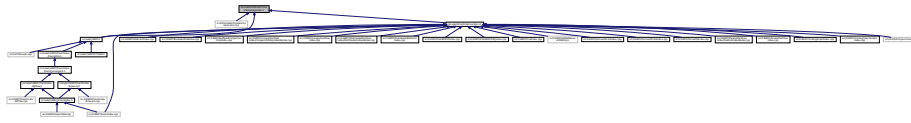
```
#include <Utils/ConfigMap.hpp>
```

```
#include <Utils/MicroDataSet.hpp>
```

Include dependency graph for IntelliTimeStampGenerator.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [INTELLI::IntelliTimeStamp](#)
The class to define a timestamp.
- class [INTELLI::IntelliTimeStampGenerator](#)
The basic class to generate time stamps.

Macros

- `#define newIntelliTimeStamp std::make_shared<INTELLI::IntelliTimeStamp>`
(Macro) To creat a new *IntelliTimeStamp* under shared pointer.

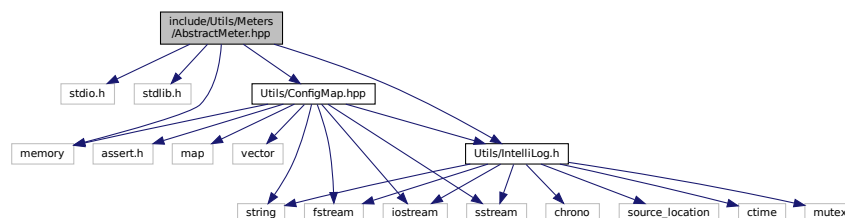
Typedefs

- `typedef std::shared_ptr< INTELLI::IntelliTimeStamp > INTELLI::IntelliTimeStampPtr`
The class to describe a shared pointer to *IntelliTimeStamp*.

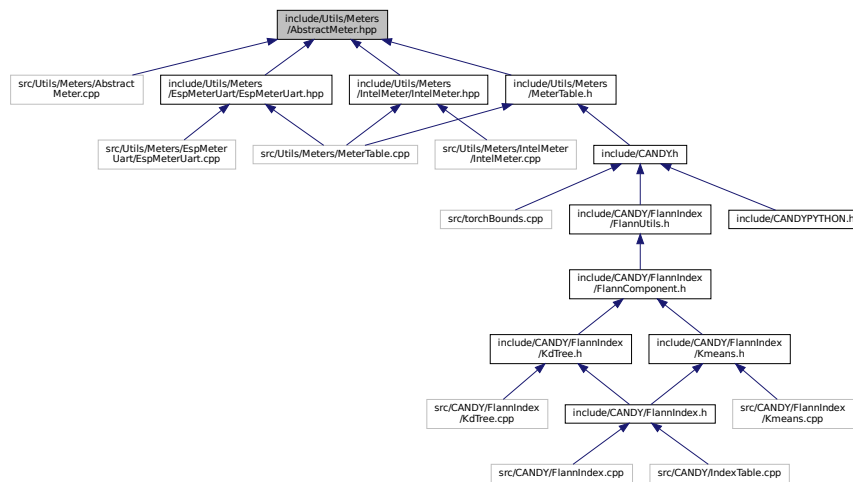
9.40 include/Utils/Meters/AbstractMeter.hpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliLog.h>
#include <memory>
```

Include dependency graph for AbstractMeter.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [DIVERSE_METER::AbstractMeter](#)

The abstract class for all meters.

Macros

- #define [METER_ERROR\(n\)](#) [INTELLI_ERROR\(n\)](#)

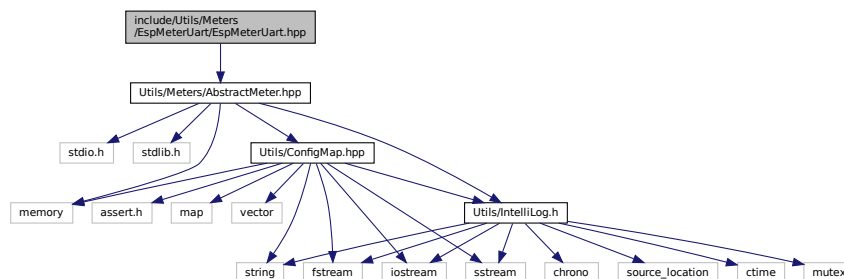
Typedefs

- typedef std::shared_ptr< [DIVERSE_METER::AbstractMeter](#) > [DIVERSE_METER::AbstractMeterPtr](#)

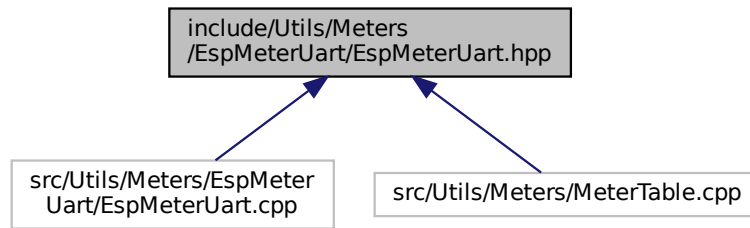
9.41 include/Utils/Meters/EspMeterUart/EspMeterUart.hpp File Reference

```
#include <Utils/Meters/AbstractMeter.hpp>
```

Include dependency graph for EspMeterUart.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [DIVERSE_METER::EspMeterUart](#)
the entity of an esp32s2-based power meter, connected by uart 115200

Macros

- `#define ADB_INCLUDE_UTILS_EspMeterUartUART_HPP_`
- `#define newEspMeterUart() std::make_shared<EspMeterUart>();`

Typedefs

- `typedef std::shared_ptr< DIVERSE_METER::EspMeterUart > DIVERSE_METER::EspMeterUartPtr`

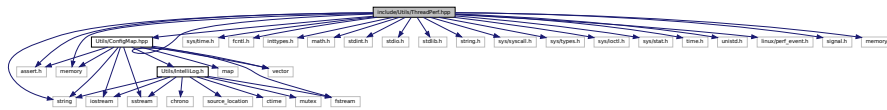
9.42 include/Utils/ThreadPerf.hpp File Reference

```

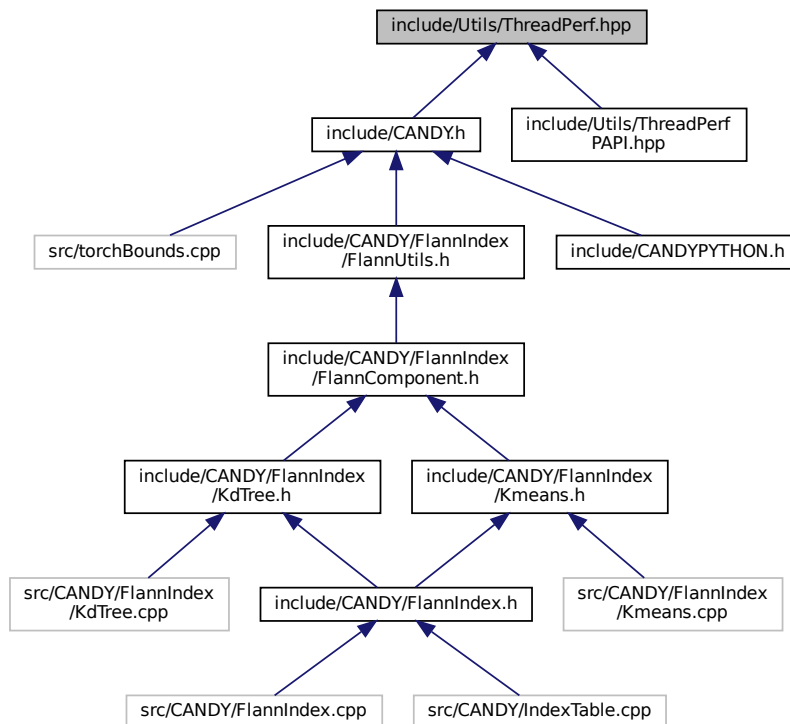
#include <string>
#include <sys/time.h>
#include <assert.h>
#include <fcntl.h>
#include <inttypes.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <time.h>
#include <unistd.h>
#include <linux/perf_event.h>
#include <signal.h>
#include <memory.h>

```

```
#include <memory>
#include <vector>
#include <Utils/ConfigMap.hpp>
Include dependency graph for ThreadPerf.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `INTELLI::ThreadPerf`
The top entity to provide perf traces, please use this class only *UNLESS* you know what you are doing.
- class `INTELLI::ThreadPerf::PerfPair`
a record pair of perf events
- class `INTELLI::ThreadPerf::PerfTool`

Macros

- `#define PERF_ERROR(n) printf(n)`
- `#define LIBPERF_ARRAY_SIZE(x) (sizeof(x)/sizeof(x[0]))`
- `#define newThreadPerf std::make_shared<INTELLI::ThreadPerf>`
(Macro) To creat a new ThreadPerf under shared pointer.

Typedefs

- typedef std::shared_ptr< [INTELLI::ThreadPerf](#) > [INTELLI::ThreadPerfPtr](#)

The class to describe a shared pointer to [ThreadPerf](#).

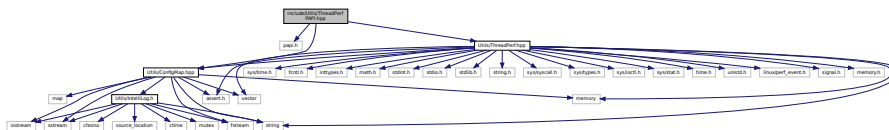
Enumerations

- enum [INTELLI::perfTrace](#) {
COUNT_SW_CPU_CLOCK = 0 , **COUNT_SW_TASK_CLOCK** = 1 , **COUNT_SW_CONTEXT_SWITCHES** = 2 , **COUNT_SW_CPU_MIGRATIONS** = 3 ,
COUNT_SW_PAGE_FAULTS = 4 , **COUNT_SW_PAGE_FAULTS_MIN** = 5 , **COUNT_SW_PAGE_FAULTS_MAJ** = 6 , **COUNT_HW_CPU_CYCLES** = 7 ,
COUNT_HW_INSTRUCTIONS = 8 , **COUNT_HW_CACHE_REFERENCES** = 9 , **COUNT_HW_CACHE_MISSES** = 10 , **COUNT_HW_BRANCH_INSTRUCTIONS** = 11 ,
COUNT_HW_BRANCH_MISSES = 12 , **COUNT_HW_BUS_CYCLES** = 13 , **COUNT_HW_CACHE_L1D_LOADS** = 14 , **COUNT_HW_CACHE_L1D_LOADS_MISSES** = 15 ,
COUNT_HW_CACHE_L1D_STORES = 16 , **COUNT_HW_CACHE_L1D_STORES_MISSES** = 17 ,
COUNT_HW_CACHE_L1D_PREFETCHES = 18 , **COUNT_HW_CACHE_L1I_LOADS** = 19 ,
COUNT_HW_CACHE_L1I_LOADS_MISSES = 20 , **COUNT_HW_CACHE_LL_LOADS** = 21 , **COUNT_HW_CACHE_LL_LOADS_MISSES** = 22 , **COUNT_HW_CACHE_LL_STORES** = 23 ,
COUNT_HW_CACHE_LL_STORES_MISSES = 24 , **COUNT_HW_CACHE_DTLB_LOADS** = 25 ,
COUNT_HW_CACHE_DTLB_LOADS_MISSES = 26 , **COUNT_HW_CACHE_DTLB_STORES** = 27 ,
COUNT_HW_CACHE_DTLB_STORES_MISSES = 28 , **COUNT_HW_CACHE_ITLB_LOADS** = 29 ,
COUNT_HW_CACHE_ITLB_LOADS_MISSES = 30 , **COUNT_HW_CACHE_BPU_LOADS** = 31 ,
COUNT_HW_CACHE_BPU_LOADS_MISSES = 32 }

The low level description of perf events, used inside, don't touch me **UNLESS** you know what you are doing.

9.43 include/Utils/ThreadPerfPAPI.hpp File Reference

```
#include <papi.h>
#include <Utils/ConfigMap.hpp>
#include <Utils/ThreadPerf.hpp>
Include dependency graph for ThreadPerfPAPI.hpp:
```



Classes

- class [INTELLI::ThreadPerfPAPI](#)
The top entity to provide perf traces by using PAPI lib.

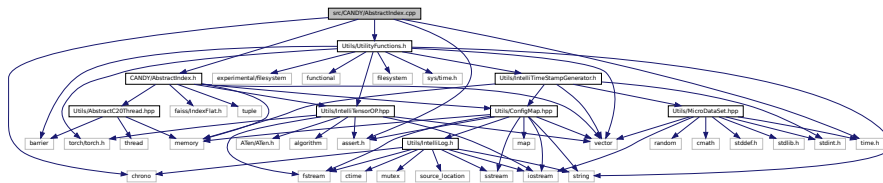
Macros

- #define **ERROR_RETURN**(retval) { fprintf(stderr, "Error %d %s:line %d: \n", retval, __FILE__, __LINE__); }
- #define [newThreadPerfPAPI](#) std::make_shared<[INTELLI::ThreadPerfPAPI](#)>
(Macro) To creat a new ThreadPerfPAPI under shared pointer.

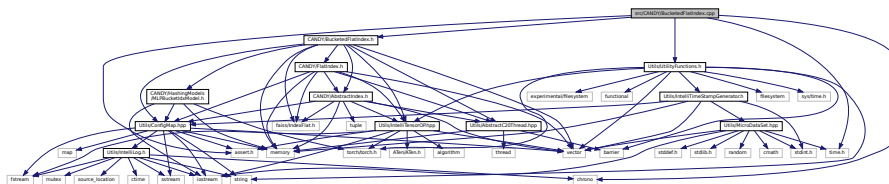
- `typedef std::shared_ptr< INTELLI::ThreadPerfPAPI > INTELLI::ThreadPerfPAPIPtr`
The class to describe a shared pointer to [ThreadPerfPAPI](#).

```
#include <CANDY/AbstractIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for AbstractIndex.cpp:



```
#include "CANDY/BucketedFlatIndex.h"
#include "Utils/UtilityFunctions.h"
#include <time.h>
#include <chrono>
#include <assert.h>
Include dependency graph for BucketedFlatIndex.cpp:
```



- `#define BF_NEXT_POW_2(V)`
- `#define HASH(X, MASK, SKIP) (((X) & MASK) >> SKIP)`

Generated by Doxygen

9.45.1.1 BF_NEXT_POW_2

```
#define BF_NEXT_POW_2(  
    V )
```

Value:

```
do {  
    V--;  
    V |= V » 1;  
    V |= V » 2;  
    V |= V » 4;  
    V |= V » 8;  
    V |= V » 16;  
    V++;  
} while(0)
```

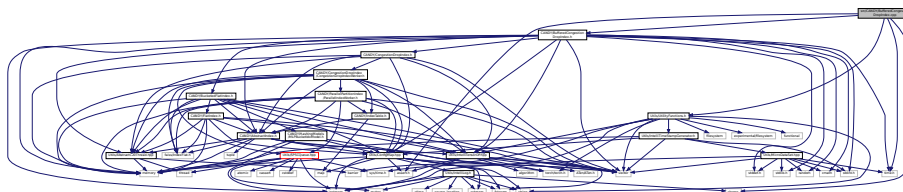
///

compute the next number, greater than or equal to 32-bit unsigned v. taken from "bit twiddling hacks": <http://graphics.stanford.edu/~seander/bithacks.html>

9.46 src/CANDY/BufferedCongestionDropIndex.cpp File Reference

```
#include <CANDY/BufferedCongestionDropIndex.h>  
#include <Utils/UtilityFunctions.h>  
#include <time.h>  
#include <chrono>  
#include <assert.h>
```

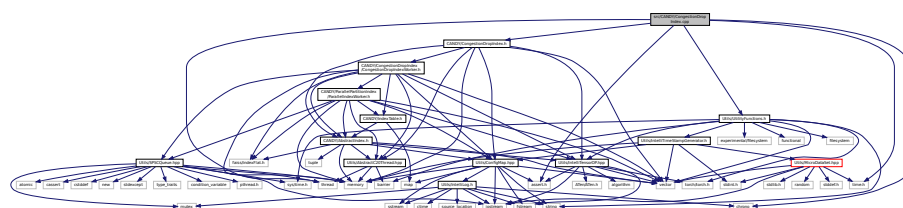
Include dependency graph for BufferedCongestionDropIndex.cpp:



9.47 src/CANDY/CongestionDropIndex.cpp File Reference

```
#include <CANDY/CongestionDropIndex.h>  
#include <Utils/UtilityFunctions.h>  
#include <time.h>  
#include <chrono>  
#include <assert.h>  
#include <thread>
```

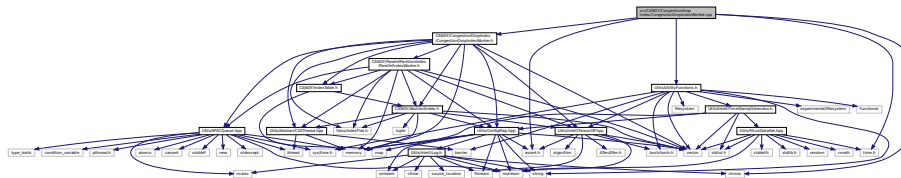
Include dependency graph for CongestionDropIndex.cpp:



9.48 src/CANDY/CongestionDropIndex/CongestionDropIndexWorker.cpp File Reference

```
#include <CANDY/CongestionDropIndex/CongestionDropIndexWorker.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

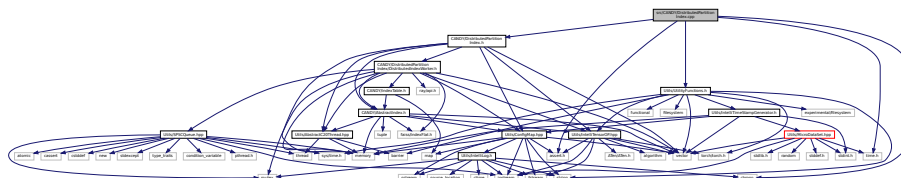
Include dependency graph for CongestionDropIndexWorker.cpp:



9.49 src/CANDY/DistributedPartitionIndex.cpp File Reference

```
#include <CANDY/DistributedPartitionIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

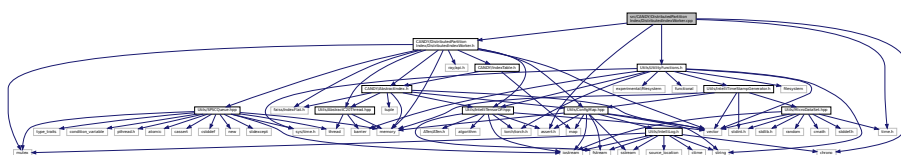
Include dependency graph for DistributedPartitionIndex.cpp:



9.50 src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp File Reference

```
#include <CANDY/DistributedPartitionIndex/DistributedIndexWorker.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for DistributedIndexWorker.cpp:



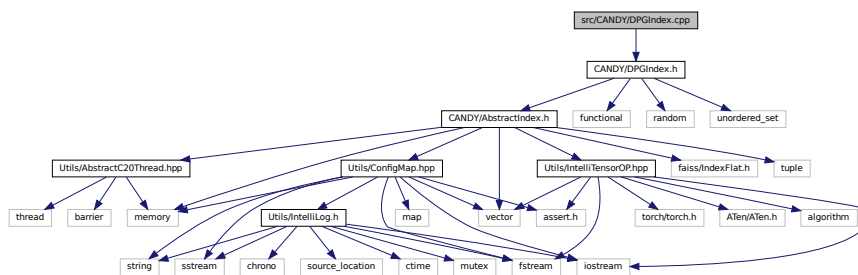
Functions

- **RAY_REMOTE** (CANDY::DIW_RayWrapper::FactoryCreate, &CANDY::DIW_RayWrapper::setConfig, &CANDY::DIW_RayWrapper::insertTensor, &CANDY::DIW_RayWrapper::deleteTensor, &CANDY::DIW_RayWrapper::searchTensor, &CANDY::DIW_RayWrapper::reset, &CANDY::DIW_RayWrapper::startHPC, &CANDY::DIW_RayWrapper::endHPC, &CANDY::DIW_RayWrapper::setFrozenLevel, &CANDY::DIW_RayWrapper::offlineBuild, &CANDY::DIW_RayWrapper::loadInit, &CANDY::DIW_RayWrapper::waitPendingOperations)

9.51 src/CANDY/DPGIndex.cpp File Reference

```
#include <CANDY/DPGIndex.h>
```

Include dependency graph for DPGIndex.cpp:



9.52 src/CANDY/FlatAMMIPIndex.cpp File Reference

```
#include <CANDY/FlatAMMIPIndex.h>
```

```
#include <UtiIs/UtilityFunctions.h>
```

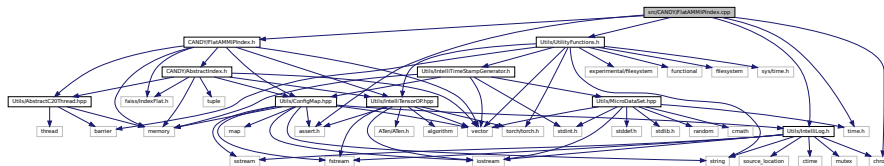
```
#include <time.h>
```

```
#include <chrono>
```

```
#include <assert.h>
```

```
#include <UtiIs/IntelliLog.h>
```

Include dependency graph for FlatAMMIPIndex.cpp:



9.53 src/CANDY/FlatAMMIPObjIndex.cpp File Reference

```
#include <CANDY/FlatAMMIPObjIndex.h>
```

```
#include <UtiIs/UtilityFunctions.h>
```

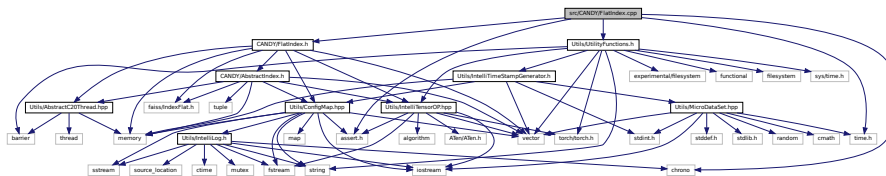
```
#include <time.h>
```

```
#include <chrono>
```

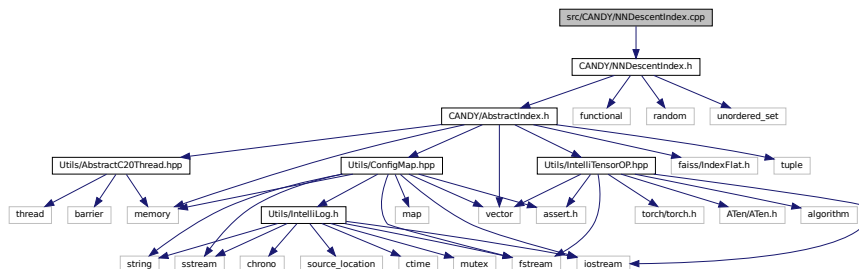
[illegible]

```
#include <CANDY/FlatIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for FlatIndex.cpp:



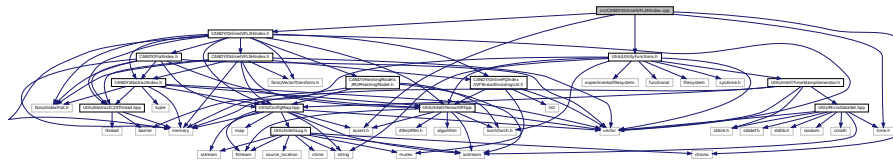
```
#include <CANDY/NNDescentIndex.h>
Include dependency graph for NNDescentIndex.cpp:
```



9.56 src/CANDY/OnlineIVFL2HIndex.cpp File Reference

```
#include <CANDY/OnlineIVFL2HIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for OnlineIVFL2HIndex.cpp:



Macros

- #define [ONLINEIVFL2H_NEXT_POW_2\(V\)](#)

9.56.1 Macro Definition Documentation

9.56.1.1 ONLINEIVFL2H_NEXT_POW_2

```
#define ONLINEIVFL2H_NEXT_POW_2(  
    V )
```

Value:

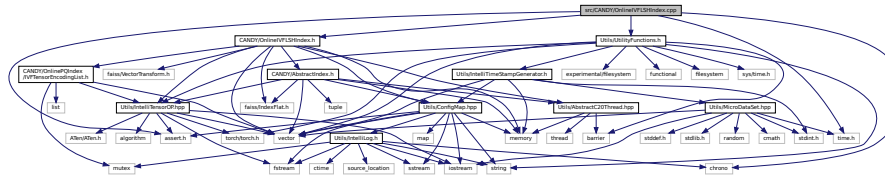
```
do {  
    V--;  
    V |= V >> 1;  
    V |= V >> 2;  
    V |= V >> 4;  
    V |= V >> 8;  
    V |= V >> 16;  
    V++;  
} while(0)
```

compute the next number, greater than or equal to 32-bit unsigned v. taken from "bit twiddling hacks": <http://graphics.stanford.edu/~seander/bithacks.html>

9.57 src/CANDY/OnlineIVFLSHIndex.cpp File Reference

```
#include <CANDY/OnlineIVFLSHIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
```

Include dependency graph for OnlineIVFLSHIndex.cpp:



- #define **ONLINEIVF_NEXT_POW_2**(V)
- #define **HASH**(X, MASK, SKIP) (((X) & MASK) >> SKIP)

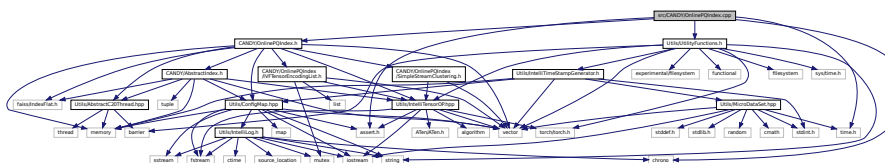
9.57.1.1 ONLINEIVF_NEXT_POW_2

Value:

///

http←

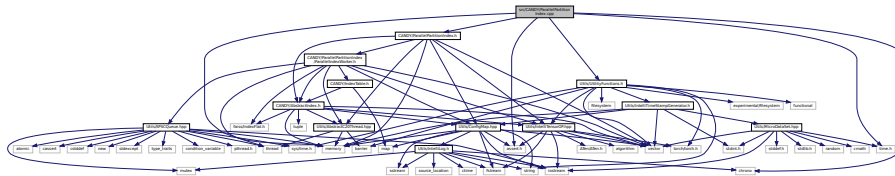
Include dependency graph for OnlinePQIndex.cpp:



9.59 src/CANDY/ParallelPartitionIndex.cpp File Reference

```
#include <CANDY/ParallelPartitionIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
#include <thread>
```

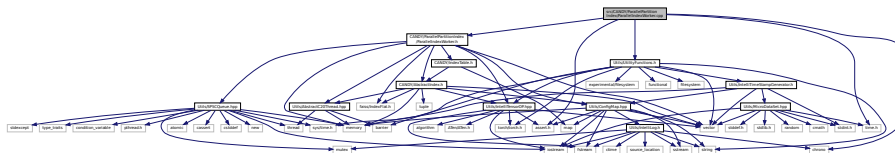
Include dependency graph for ParallelPartitionIndex.cpp:



9.60 src/CANDY/ParallelPartitionIndex/ParallelIndexWorker.cpp File Reference

```
#include <CANDY/ParallelPartitionIndex/ParallelIndexWorker.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

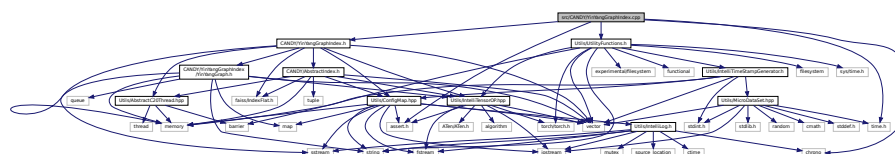
Include dependency graph for ParallelIndexWorker.cpp:



9.61 src/CANDY/YinYangGraphIndex.cpp File Reference

```
#include <CANDY/YinYangGraphIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for YinYangGraphIndex.cpp:



Macros

- `#define ONLINEIVF_NEXT_POW_2(V)`
- `#define HASH(X, MASK, SKIP) (((X) & MASK) >> SKIP)`

9.61.1 Macro Definition Documentation

9.61.1.1 ONLINEIVF_NEXT_POW_2

```
#define ONLINEIVF_NEXT_POW_2(  
    V )
```

Value:

```
do {  
    V--;  
    V |= V >> 1;  
    V |= V >> 2;  
    V |= V >> 4;  
    V |= V >> 8;  
    V |= V >> 16;  
    V++;  
} while(0)
```

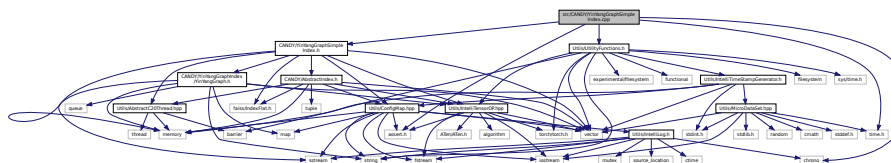
///

compute the next number, greater than or equal to 32-bit unsigned v. taken from "bit twiddling hacks": <http://graphics.stanford.edu/~seander/bithacks.html>

9.62 src/CANDY/YinYangGraphSimpleIndex.cpp File Reference

```
#include <CANDY/YinYangGraphSimpleIndex.h>  
#include <Utils/UtilityFunctions.h>  
#include <time.h>  
#include <chrono>  
#include <assert.h>
```

Include dependency graph for YinYangGraphSimpleIndex.cpp:



Index

- [_cl_device_integer_dot_product_acceleration_properties_intel](#), [67](#)
- [_cl_device_pci_bus_info_khr](#), [68](#)
- [_cl_icd_dispatch](#), [68](#)
- [_cl_image_format](#), [72](#)
- [_cl_mem_android_native_buffer_host_ptr](#), [72](#)
- [_cl_mem_ext_host_ptr](#), [73](#)
- [_cl_mem_ion_host_ptr](#), [74](#)
- [_cl_motion_estimation_desc_intel](#), [75](#)
- [_cl_name_version_khr](#), [75](#)
- [_cl_queue_family_properties_intel](#), [76](#)
- [add](#)
 - [CANDY::PQIndex](#), [349](#)
 - [CANDY::ProductQuantizer](#), [356](#)
- [add_links_starting_from](#)
 - [CANDY::HNSW](#), [232](#)
- [add_without_lock](#)
 - [CANDY::HNSW](#), [232](#)
- [addIndex](#)
 - [CANDY::IndexTable](#), [243](#)
- [addPapiTag](#)
 - [INTELLI::ThreadPerfPAPI](#), [409](#), [410](#)
- [addPoints](#)
 - [CANDY::KdTree](#), [275](#)
 - [CANDY::KmeansTree](#), [282](#)
- [addPointToTree](#)
 - [CANDY::KdTree](#), [276](#)
 - [CANDY::KmeansTree](#), [282](#)
- [addSingleRow](#)
 - [CANDY::SimpleStreamClustering](#), [382](#)
- [addSingleRowWithIdx](#)
 - [CANDY::SimpleStreamClustering](#), [382](#)
- [append](#)
 - [INTELLI::C20Buffer< dataType >](#), [117](#), [118](#)
- [appendLogFile](#)
 - [Log utils](#), [56](#)
- [appendRows](#)
 - [INTELLI::IntelliTensorOP](#), [246](#), [247](#)
- [appendRowsBufferMode](#)
 - [INTELLI::IntelliTensorOP](#), [247](#), [248](#)
- [attachTensor](#)
 - [CANDY::YinYangVertex](#), [435](#)
- [BF_NEXT_POW_2](#)
 - [BucketedFlatIndex.cpp](#), [493](#)
- [bind2Core](#)
 - [INTELLI::UtilityFunctions](#), [413](#)
- [blocks](#)
 - [BS::blocks< T1, T2, T >](#), [98](#)
 - [blocks](#), [98](#)
 - [end](#), [99](#)
 - [get_num_blocks](#), [99](#)
 - [get_total_size](#), [99](#)
 - [start](#), [100](#)
- [BS::multi_future< T >](#), [296](#)
 - [get](#), [297](#)
 - [multi_future](#), [297](#)
 - [operator\[\]](#), [298](#)
 - [push_back](#), [298](#)
 - [size](#), [298](#)
- [BS::synced_stream](#), [389](#)
 - [endl](#), [391](#)
 - [flush](#), [391](#)
 - [print](#), [390](#)
 - [println](#), [390](#)
 - [synced_stream](#), [390](#)
- [BS::thread_pool](#), [395](#)
 - [get_tasks_queued](#), [396](#)
 - [get_tasks_running](#), [397](#)
 - [get_tasks_total](#), [397](#)
 - [get_thread_count](#), [397](#)
 - [is_paused](#), [397](#)
 - [parallelize_loop](#), [397](#), [398](#)
 - [push_loop](#), [399](#), [400](#)
 - [push_task](#), [400](#)
 - [reset](#), [401](#)
 - [submit](#), [401](#)
 - [thread_pool](#), [396](#)
- [BS::timer](#), [412](#)
 - [ms](#), [412](#)
- [BucketedFlatIndex.cpp](#)
 - [BF_NEXT_POW_2](#), [493](#)
- [bufferSize](#)
 - [INTELLI::C20Buffer< dataType >](#), [118](#)
- [buildCentroids](#)
 - [CANDY::SimpleStreamClustering](#), [383](#)
- [C20Buffer](#)
 - [INTELLI::C20Buffer< dataType >](#), [117](#)
- [calculateRecall](#)
 - [INTELLI::UtilityFunctions](#), [414](#)
- [CANDY::AbstractDataLoader](#), [78](#)
 - [getData](#), [79](#)
 - [getQuery](#), [79](#)
 - [hijackConfig](#), [80](#)
 - [setConfig](#), [80](#)
- [CANDY::AbstractIndex](#), [81](#)
 - [deleteStringObject](#), [84](#)

- deleteTensor, 85
- endHPC, 85
- getTensorByIndex, 85
- insertStringObject, 86
- insertTensor, 86
- loadInitialStringObject, 87
- loadInitialTensor, 88
- loadInitialTensorAndQueryDistribution, 88
- offlineBuild, 89
- rawData, 89
- reviseTensor, 89
- searchIndex, 90
- searchStringObject, 90
- searchTensor, 91
- searchTensorAndStringObject, 91
- setConfig, 92
- setConfigClass, 92
- setFrozenLevel, 93
- setTier, 93
- startHPC, 93
- waitPendingOperations, 94
- CANDY::AdSampling, 97
- CANDY::BucketedFlatIndex, 101
 - deleteTensor, 104
 - encodeMultiRows, 104
 - insertTensor, 104
 - loadInitialTensor, 105
 - reviseTensor, 105
 - searchSingleRow, 106
 - searchTensor, 106
 - setConfig, 107
- CANDY::BufferedCongestionDropIndex, 107
 - deleteTensor, 111
 - endHPC, 111
 - generateBucketedFlatIndexConfig, 112
 - insertTensor, 112
 - insertTensorInline, 112
 - loadInitialTensor, 113
 - offlineBuild, 113
 - reviseTensor, 114
 - searchTensor, 114
 - setConfig, 115
 - setFrozenLevel, 115
 - startHPC, 115
- CANDY::CANDYObject, 120
 - getStr, 121
 - setStr, 121
- CANDY::Clustering, 133
 - computeCentroids, 135
 - imbalance_factor, 135
 - splitClusters, 136
 - train, 136
- CANDY::ClusteringIterationStats, 137
- CANDY::ClusteringParameters, 138
- CANDY::CongestionDropIndex, 141
 - deleteStringObject, 144
 - deleteTensor, 144
 - endHPC, 145
 - getTensorByIndex, 145
 - insertStringObject, 146
 - insertTensor, 146
 - loadInitialStringObject, 147
 - loadInitialTensor, 147
 - offlineBuild, 148
 - rawData, 148
 - reviseTensor, 148
 - searchStringObject, 149
 - searchTensor, 149
 - searchTensorAndStringObject, 150
 - setConfig, 150
 - setFrozenLevel, 151
 - startHPC, 151
 - waitPendingOperations, 151
- CANDY::CongestionDropIndexWorker, 152
- CANDY::DataLoaderTable, 154
 - DataLoaderTable, 155
 - findDataLoader, 155
 - registerNewDataLoader, 156
- CANDY::DistanceQueryer, 157
 - operator(), 158
- CANDY::DistributedIndexWorker, 159
 - deleteTensor, 161
 - endHPC, 161
 - getUnblockQueryResult, 161
 - insertTensor, 162
 - loadInitialTensor, 162
 - loadInitialTensorUnblocked, 162
 - offlineBuild, 163
 - offlineBuildUnblocked, 163
 - searchTensor, 163
 - searchTensorUnblock, 164
 - setConfig, 164
 - setFrozenLevel, 164
 - startHPC, 165
 - waitPendingOperations, 165
- CANDY::DistributedPartitionIndex, 166
 - deleteTensor, 168
 - endHPC, 168
 - getTensorByIndex, 169
 - insertTensor, 169
 - loadInitialTensor, 169
 - offlineBuild, 170
 - rawData, 170
 - reviseTensor, 170
 - searchTensor, 171
 - setConfig, 171
 - setFrozenLevel, 172
 - startHPC, 172
 - waitPendingOperations, 172
- CANDY::DIW_RayWrapper, 173
 - deleteTensor, 174
 - endHPC, 175
 - insertTensor, 175
 - loadInitialTensor, 175
 - offlineBuild, 176
 - searchTensor, 176

- setConfig, 176
- setFrozenLevel, 177
- startHPC, 177
- waitPendingOperations, 177
- CANDY::DPGIndex, 178
 - deleteTensor, 181
 - endHPC, 181
 - getTensorByIndex, 181
 - insertTensor, 182
 - loadInitialTensor, 182
 - offlineBuild, 183
 - rawData, 183
 - reviseTensor, 183
 - searchTensor, 184
 - setConfig, 184
 - setFrozenLevel, 185
 - startHPC, 185
- CANDY::DPGIndex::Neighbor, 299
- CANDY::DPGIndex::NhoodLayer0, 301
- CANDY::DPGIndex::NhoodLayer1, 302
- CANDY::ExpFamilyDataLoader, 187
 - getData, 190
 - getQuery, 190
 - hijackConfig, 190
 - setConfig, 191
- CANDY::FaissIndex, 191
 - getTensorByIndex, 193
 - insertTensor, 194
 - loadInitialTensor, 194
 - searchIndex, 195
 - searchTensor, 195
 - setConfig, 195
- CANDY::FLANN::BranchStruct< T >, 100
- CANDY::FLANN::DistanceIndex, 157
- CANDY::FLANN::Heap< T >, 229
- CANDY::FLANN::RandomCenterChooser, 360
- CANDY::FLANN::ResultSet, 380
- CANDY::FLANN::UniqueRandom, 413
- CANDY::FLANN::VisitBitset, 416
- CANDY::FlannComponent, 196
 - setParams, 197
- CANDY::FlannIndex, 198
 - getTensorByIndex, 199
 - insertTensor, 199
 - loadInitialTensor, 199
 - searchIndex, 200
 - searchTensor, 200
 - setConfig, 201
- CANDY::FlannParam, 202
- CANDY::FlatAMMIPIndex, 202
 - deleteTensor, 205
 - getTensorByIndex, 205
 - insertTensor, 205
 - rawData, 207
 - reviseTensor, 207
 - searchIndex, 207
 - searchTensor, 208
 - setConfig, 208
 - size, 209
- CANDY::FlatAMMIPObjIndex, 209
 - deleteStringObject, 211
 - deleteTensor, 212
 - getTensorByIndex, 212
 - insertStringObject, 213
 - insertTensor, 213
 - rawData, 213
 - reviseTensor, 214
 - searchIndex, 214
 - searchStringObject, 215
 - searchTensor, 215
 - setConfig, 215
 - size, 216
- CANDY::FlatIndex, 216
 - deleteTensor, 218
 - getTensorByIndex, 218
 - insertTensor, 219
 - rawData, 219
 - reviseTensor, 219
 - searchIndex, 220
 - searchTensor, 220
 - setConfig, 221
 - size, 221
- CANDY::FVECSDataLoader, 222
 - getData, 224
 - getQuery, 224
 - setConfig, 224
 - tensorFromFVECS, 225
- CANDY::HDF5DataLoader, 225
 - getData, 227
 - getQuery, 227
 - setConfig, 228
 - tensorFromHDF5, 228
- CANDY::HNSW, 230
 - add_links_starting_from, 232
 - add_without_lock, 232
 - cum_nb_neighbors, 233
 - cum_nneighbor_per_level_, 236
 - nb_neighbors, 233
 - neighbor_range, 233
 - prepare_level_tab, 234
 - random_level, 234
 - search, 234
 - search_bounded_queue, 236
 - set_nb_neighbors, 235
 - set_probs, 235
- CANDY::HNSW::MinimaxHeap, 292
- CANDY::HNSW::NodeDistCloser, 311
- CANDY::HNSW::NodeDistFarther, 312
- CANDY::HNSWNaiveIndex, 236
 - deleteTensor, 238
 - insertTensor, 238
 - reviseTensor, 239
 - searchTensor, 239
 - setConfig, 240
- CANDY::HNSWVertex, 240
- CANDY::IndexTable, 242

- addIndex, [243](#)
- getIndex, [244](#)
- CANDY::IVFLListBucket, [263](#)
 - deleteTensor, [264](#)
 - deleteTensorWithEncode, [264](#)
 - getAllTensors, [265](#)
 - getAllTensorsWithEncode, [265](#)
 - getMinimumTensorsUnderHamming, [265](#)
 - insertTensorWithEncode, [266](#)
 - sizeWithEncode, [266](#)
- CANDY::IVFLListCell, [267](#)
 - deleteTensor, [268](#)
 - deleteTensorPtr, [268](#)
 - getAllTensors, [268](#)
 - insertTensor, [268](#)
 - insertTensorPtr, [269](#)
- CANDY::IVFTensorEncodingList, [269](#)
 - deleteTensorWithEncode, [270](#)
 - getMinimumNumOfTensors, [271](#)
 - getMinimumNumOfTensorsHamming, [271](#)
 - getMinimumNumOfTensorsInsideBucket, [272](#)
 - getMinimumNumOfTensorsInsideBucketHamming, [272](#)
 - init, [273](#)
 - insertTensorWithEncode, [273](#)
- CANDY::KdTree, [274](#)
 - addPoints, [275](#)
 - addPointToTree, [276](#)
 - divideTree, [276](#)
 - getNeighbors, [276](#)
 - knnSearch, [277](#)
 - meanSplit, [277](#)
 - planeSplit, [278](#)
 - searchLevel, [278](#)
 - selectDivision, [279](#)
 - setConfig, [279](#)
 - setParams, [279](#)
- CANDY::KdTree::Node, [310](#)
- CANDY::KmeansTree, [280](#)
 - addPoints, [282](#)
 - addPointToTree, [282](#)
 - computeClustering, [282](#)
 - computeNodeStat, [283](#)
 - explore, [283](#)
 - findNN, [283](#)
 - getNeighbors, [284](#)
 - knnSearch, [284](#)
 - setConfig, [285](#)
 - setParams, [285](#)
- CANDY::KmeansTree::Node, [310](#)
- CANDY::KmeansTree::NodeInfo, [313](#)
- CANDY::MLPBucketIdxModel, [293](#)
 - hash, [293](#)
 - init, [293](#)
 - trainModel, [294](#)
- CANDY::MLPHashingModel, [294](#)
 - fineTuneModel, [295](#)
 - hash, [295](#)
 - init, [295](#)
 - trainModel, [296](#)
- CANDY::NNDescentIndex, [303](#)
 - deleteTensor, [305](#)
 - endHPC, [305](#)
 - getTensorByIndex, [306](#)
 - insertTensor, [306](#)
 - loadInitialTensor, [306](#)
 - offlineBuild, [307](#)
 - rawData, [307](#)
 - reviseTensor, [307](#)
 - searchTensor, [308](#)
 - setConfig, [308](#)
 - setFrozenLevel, [309](#)
 - startHPC, [309](#)
- CANDY::NNDescentIndex::Neighbor, [300](#)
- CANDY::NNDescentIndex::Nhoo, [300](#)
- CANDY::OnlineIVFL2HIndex, [313](#)
 - loadInitialTensor, [315](#)
 - loadInitialTensorAndQueryDistribution, [316](#)
 - setConfig, [316](#)
 - trainModelWithData, [317](#)
- CANDY::OnlineIVFLSHIndex, [317](#)
 - crsAmm, [319](#)
 - deleteRowsInline, [320](#)
 - deleteTensor, [320](#)
 - insertTensor, [320](#)
 - reviseTensor, [321](#)
 - searchTensor, [321](#)
 - setConfig, [322](#)
- CANDY::OnlinePQIndex, [322](#)
 - deleteRowsInline, [325](#)
 - deleteTensor, [325](#)
 - fineGrainedEncode, [326](#)
 - insertTensor, [326](#)
 - loadInitialTensor, [326](#)
 - offlineBuild, [327](#)
 - reviseTensor, [327](#)
 - searchTensor, [328](#)
 - setConfig, [328](#)
 - setFrozenLevel, [329](#)
- CANDY::ParallelIndexWorker, [330](#)
- CANDY::ParallelPartitionIndex, [333](#)
 - deleteStringObject, [335](#)
 - deleteTensor, [336](#)
 - endHPC, [336](#)
 - getTensorByIndex, [336](#)
 - insertStringObject, [337](#)
 - insertTensor, [337](#)
 - loadInitialStringObject, [338](#)
 - loadInitialTensor, [338](#)
 - offlineBuild, [339](#)
 - rawData, [339](#)
 - reviseTensor, [339](#)
 - searchStringObject, [340](#)
 - searchTensor, [340](#)
 - searchTensorAndStringObject, [341](#)
 - setConfig, [341](#)

- setFrozenLevel, [342](#)
 - startHPC, [342](#)
 - waitPendingOperations, [342](#)
- CANDY::PQDecoder, [344](#)
 - decode, [345](#)
- CANDY::PQEncoder, [346](#)
 - encode, [346](#)
- CANDY::PQIndex, [347](#)
 - add, [349](#)
 - deleteTensor, [349](#)
 - getTensorByIndex, [351](#)
 - insertTensor, [351](#)
 - loadInitialTensor, [352](#)
 - rawData, [352](#)
 - reviseTensor, [352](#)
 - searchIndex, [353](#)
 - searchTensor, [353](#)
 - setConfig, [354](#)
 - setFrozenLevel, [354](#)
 - train, [354](#)
- CANDY::ProductQuantizer, [355](#)
 - add, [356](#)
 - compute_code, [357](#)
 - compute_codes, [357](#)
 - compute_distance_table, [357](#)
 - compute_distance_tables, [358](#)
 - decode, [358](#)
 - search, [359](#)
 - setCentroidsFrom, [359](#)
 - train, [360](#)
- CANDY::RandomDataLoader, [361](#)
 - getData, [362](#)
 - getQuery, [362](#)
 - setConfig, [363](#)
- CANDY::Rania_Python, [363](#)
 - dataLoader_create, [365](#)
 - dataLoader_editCfgDouble, [366](#)
 - dataLoader_editCfgFloat, [366](#)
 - dataLoader_editCfgI64, [367](#)
 - dataLoader_editCfgStr, [367](#)
 - dataLoader_getData, [367](#)
 - dataLoader_getQuery, [368](#)
 - dataLoader_init, [368](#)
 - index_create, [369](#)
 - index_delete, [369](#)
 - index_deleteString, [369](#)
 - index_editCfgDouble, [370](#)
 - index_editCfgI64, [370](#)
 - index_editCfgStr, [371](#)
 - index_endHPC, [371](#)
 - index_init, [371](#)
 - index_insert, [372](#)
 - index_insertString, [372](#)
 - index_loadCfgFromFile, [372](#)
 - index_loadInitial, [373](#)
 - index_loadInitialString, [373](#)
 - index_offlineBuild, [374](#)
 - index_rawData, [374](#)
 - index_reset, [374](#)
 - index_revise, [375](#)
 - index_search, [375](#)
 - index_searchString, [376](#)
 - index_searchTensorAndStringList, [376](#)
 - index_setFrozenLevel, [376](#)
 - index_startHPC, [377](#)
 - index_waitPending, [377](#)
 - tensorFromFile, [378](#)
 - tensorFromFVECS, [378](#)
 - tensorFromHDF5, [378](#)
 - tensorToFile, [379](#)
- CANDY::SimpleStreamClustering, [380](#)
 - addSingleRow, [382](#)
 - addSingleRowWithIdx, [382](#)
 - buildCentroids, [383](#)
 - classifyMultiRow, [383](#)
 - classifySingleRow, [384](#)
 - deleteSingleRow, [384](#)
 - deleteSingleRowWithIdx, [384](#)
 - euclideanDelete, [385](#)
 - euclideanDistance, [385](#)
 - euclideanInsert, [386](#)
 - exportCentroids, [386](#)
 - loadCentroids, [386](#), [387](#)
 - saveCentroidsToFile, [387](#)
- CANDY::TensorIdxPair, [392](#)
- CANDY::TensorListIdxPair, [393](#)
- CANDY::TensorStrPair, [393](#)
- CANDY::TensorStrVecPair, [394](#)
- CANDY::VisitedTable, [417](#)
- CANDY::YinYangGraph, [418](#)
 - deleteTensorWithEncode, [419](#)
 - getMinimumNumOfTensors, [419](#)
 - init, [420](#)
 - insertTensorWithEncode, [420](#)
- CANDY::YinYangGraph_DistanceFunctions, [421](#)
- CANDY::YinYangGraph_ListBucket, [421](#)
 - deleteTensor, [422](#)
 - deleteTensorWithEncode, [423](#)
 - getVertexWithEncode, [423](#)
 - insertTensorWithEncode, [423](#)
- CANDY::YinYangGraph_ListCell, [424](#)
 - deleteTensor, [425](#)
 - getVertex, [425](#)
 - insertTensor, [425](#)
- CANDY::YinYangGraphIndex, [426](#)
 - crsAmm, [428](#)
 - insertTensor, [429](#)
 - searchTensor, [429](#)
 - setConfig, [429](#)
- CANDY::YinYangGraphSimpleIndex, [430](#)
 - insertSingleRowTensor, [431](#)
 - insertTensor, [432](#)
 - searchTensor, [432](#)
 - setConfig, [432](#)
- CANDY::YinYangVertex, [433](#)
 - attachTensor, [435](#)

- detachTensor, [435](#)
- greedySearchForKNearestTensor, [436](#)
- greedySearchForKNearestVertex, [436](#), [437](#)
- greedySearchForNearestVertex, [437](#), [438](#)
- init, [438](#)
- setUpLayer, [439](#)
- toString, [439](#)
- tryToConnect, [439](#)
- CANDY::YinYangVertexMap, [440](#)
 - edit, [441](#)
 - erase, [442](#)
 - exist, [442](#)
 - nearestKVertexWithinMap, [442](#)
 - nearestVertexWithinMap, [443](#)
 - nearestVertexWithinMe, [443](#)
- CANDY::ZipfDataLoader, [444](#)
 - getData, [445](#)
 - getQuery, [446](#)
 - setConfig, [446](#)
- cl_char16, [121](#)
- cl_char2, [122](#)
- cl_char4, [122](#)
- cl_char8, [122](#)
- cl_double16, [122](#)
- cl_double2, [123](#)
- cl_double4, [123](#)
- cl_double8, [123](#)
- cl_float16, [123](#)
- cl_float2, [124](#)
- cl_float4, [124](#)
- cl_float8, [124](#)
- cl_half16, [124](#)
- cl_half2, [125](#)
- cl_half4, [125](#)
- cl_half8, [125](#)
- cl_int16, [125](#)
- cl_int2, [126](#)
- cl_int4, [126](#)
- cl_int8, [126](#)
- cl_long16, [126](#)
- cl_long2, [127](#)
- cl_long4, [127](#)
- cl_long8, [127](#)
- cl_short16, [127](#)
- cl_short2, [128](#)
- cl_short4, [128](#)
- cl_short8, [128](#)
- cl_uchar16, [128](#)
- cl_uchar2, [129](#)
- cl_uchar4, [129](#)
- cl_uchar8, [129](#)
- cl_uint16, [129](#)
- cl_uint2, [130](#)
- cl_uint4, [130](#)
- cl_uint8, [130](#)
- cl_ulong16, [130](#)
- cl_ulong2, [131](#)
- cl_ulong4, [131](#)
- cl_ulong8, [131](#)
- cl_ushort16, [131](#)
- cl_ushort2, [132](#)
- cl_ushort4, [132](#)
- cl_ushort8, [132](#)
- classifyMultiRow
 - CANDY::SimpleStreamClustering, [383](#)
- classifySingleRow
 - CANDY::SimpleStreamClustering, [384](#)
- cloneInto
 - Shared Utils, [43](#)
- compute_code
 - CANDY::ProductQuantizer, [357](#)
- compute_codes
 - CANDY::ProductQuantizer, [357](#)
- compute_distance_table
 - CANDY::ProductQuantizer, [357](#)
- compute_distance_tables
 - CANDY::ProductQuantizer, [358](#)
- computeCentroids
 - CANDY::Clustering, [135](#)
- computeClustering
 - CANDY::KmeansTree, [282](#)
- computeNodeStat
 - CANDY::KmeansTree, [283](#)
- Configurations, [54](#)
- crsAmm
 - CANDY::OnlineIVFLSHIndex, [319](#)
 - CANDY::YinYangGraphIndex, [428](#)
- cum_nb_neighbors
 - CANDY::HNSW, [233](#)
- cum_nneighbor_per_level_
 - CANDY::HNSW, [236](#)
- data
 - INTELLI::C20Buffer< dataType >, [119](#)
- dataLoader_create
 - CANDY::Rania_Python, [365](#)
- dataLoader_editCfgDouble
 - CANDY::Rania_Python, [366](#)
- dataLoader_editCfgFloat
 - CANDY::Rania_Python, [366](#)
- dataLoader_editCfgI64
 - CANDY::Rania_Python, [367](#)
- dataLoader_editCfgStr
 - CANDY::Rania_Python, [367](#)
- dataLoader_getData
 - CANDY::Rania_Python, [367](#)
- dataLoader_getQuery
 - CANDY::Rania_Python, [368](#)
- dataLoader_init
 - CANDY::Rania_Python, [368](#)
- DataLoaderTable
 - CANDY::DataLoaderTable, [155](#)
- decode
 - CANDY::PQDecoder, [345](#)
 - CANDY::ProductQuantizer, [358](#)
- default_attrs, [156](#)
- deleteRow

- INTELLI::IntelliTensorOP, [248](#), [249](#)
- deleteRowBufferMode
 - INTELLI::IntelliTensorOP, [249](#)
- deleteRows
 - INTELLI::IntelliTensorOP, [250](#)
- deleteRowsBufferMode
 - INTELLI::IntelliTensorOP, [251](#)
- deleteRowsInline
 - CANDY::OnlineIVFLSHIndex, [320](#)
 - CANDY::OnlinePQIndex, [325](#)
- deleteSingleRow
 - CANDY::SimpleStreamClustering, [384](#)
- deleteSingleRowWithIdx
 - CANDY::SimpleStreamClustering, [384](#)
- deleteStringObject
 - CANDY::AbstractIndex, [84](#)
 - CANDY::CongestionDropIndex, [144](#)
 - CANDY::FlatAMMIPObjIndex, [211](#)
 - CANDY::ParallelPartitionIndex, [335](#)
 - The support classes for index approaches, [26](#)
- deleteTensor
 - CANDY::AbstractIndex, [85](#)
 - CANDY::BucketedFlatIndex, [104](#)
 - CANDY::BufferedCongestionDropIndex, [111](#)
 - CANDY::CongestionDropIndex, [144](#)
 - CANDY::DistributedIndexWorker, [161](#)
 - CANDY::DistributedPartitionIndex, [168](#)
 - CANDY::DIW_RayWrapper, [174](#)
 - CANDY::DPGIndex, [181](#)
 - CANDY::FlatAMMIPIndex, [205](#)
 - CANDY::FlatAMMIPObjIndex, [212](#)
 - CANDY::FlatIndex, [218](#)
 - CANDY::HNSWNaiveIndex, [238](#)
 - CANDY::IVFLListBucket, [264](#)
 - CANDY::IVFLListCell, [268](#)
 - CANDY::NNDescentIndex, [305](#)
 - CANDY::OnlineIVFLSHIndex, [320](#)
 - CANDY::OnlinePQIndex, [325](#)
 - CANDY::ParallelPartitionIndex, [336](#)
 - CANDY::PQIndex, [349](#)
 - CANDY::YinYangGraph_ListBucket, [422](#)
 - CANDY::YinYangGraph_ListCell, [425](#)
 - The support classes for index approaches, [27](#)
- deleteTensorPtr
 - CANDY::IVFLListCell, [268](#)
- deleteTensorWithEncode
 - CANDY::IVFLListBucket, [264](#)
 - CANDY::IVFTensorEncodingList, [270](#)
 - CANDY::YinYangGraph, [419](#)
 - CANDY::YinYangGraph_ListBucket, [423](#)
- detachTensor
 - CANDY::YinYangVertex, [435](#)
- DIVERSE_METER::AbstractMeter, [94](#)
- getStaicEnergyConsumption, [96](#)
- setConfig, [96](#)
- setStaticPower, [96](#)
- testStaticPower, [97](#)
- DIVERSE_METER::EspMeterUart, [186](#)
- setConfig, [187](#)
- DIVERSE_METER::IntelMeter, [261](#)
- setConfig, [262](#)
- DIVERSE_METER::MeterTable, [288](#)
- findMeter, [290](#)
- MeterTable, [290](#)
- registerNewMeter, [290](#)
- DIVERSE_METER::rapl_power_unit, [379](#)
- divideTree
 - CANDY::KdTree, [276](#)
- edit
 - CANDY::YinYangVertexMap, [441](#)
 - Shared Utils, [44](#)
- editRows
 - INTELLI::IntelliTensorOP, [251](#), [252](#)
- encode
 - CANDY::PQEncoder, [346](#)
- encodeMultiRows
 - CANDY::BucketedFlatIndex, [104](#)
- end
 - BS::blocks< T1, T2, T >, [99](#)
- endHPC
 - CANDY::AbstractIndex, [85](#)
 - CANDY::BufferedCongestionDropIndex, [111](#)
 - CANDY::CongestionDropIndex, [145](#)
 - CANDY::DistributedIndexWorker, [161](#)
 - CANDY::DistributedPartitionIndex, [168](#)
 - CANDY::DIW_RayWrapper, [175](#)
 - CANDY::DPGIndex, [181](#)
 - CANDY::NNDescentIndex, [305](#)
 - CANDY::ParallelPartitionIndex, [336](#)
 - The support classes for index approaches, [27](#)
- endl
 - BS::synced_stream, [391](#)
- Energy Meter packs, [59](#)
- erase
 - CANDY::YinYangVertexMap, [442](#)
- euclideanDelete
 - CANDY::SimpleStreamClustering, [385](#)
- euclideanDistance
 - CANDY::SimpleStreamClustering, [385](#)
- euclideanInsert
 - CANDY::SimpleStreamClustering, [386](#)
- exist
 - CANDY::YinYangVertexMap, [442](#)
 - Shared Utils, [45](#)
- existDouble
 - Shared Utils, [45](#)
- existI64
 - Shared Utils, [45](#)
- existString
 - Shared Utils, [46](#)
- existU64
 - Shared Utils, [46](#)
- explore
 - CANDY::KmeansTree, [283](#)
- exportCentroids
 - CANDY::SimpleStreamClustering, [386](#)

- findDataLoader
 - CANDY::DataLoaderTable, 155
- findMeter
 - DIVERSE_METER::MeterTable, 290
- findNN
 - CANDY::KmeansTree, 283
- fineGrainedEncode
 - CANDY::OnlinePQIndex, 326
- fineTuneModel
 - CANDY::MLPHashingModel, 295
- flush
 - BS::synced_stream, 391
- fromCArg
 - Shared Utils, 47
- fromFile
 - Shared Utils, 47
- fromString
 - Shared Utils, 47
- generateArrival
 - INTELLI::IntelliTimeStampGenerator, 260
- generateBucketedFlatIndexConfig
 - CANDY::BufferedCongestionDropIndex, 112
- generic, 62
 - genIncrementalAlphabet, 62
 - genRandInt, 63
 - genZipfInt, 63
 - genZipfLut, 64
- genIncrementalAlphabet
 - generic, 62
- genRandInt
 - generic, 63
- genSmoothTimeStamp
 - time stamp, 65
- genZipfInt
 - generic, 63
- genZipfLut
 - generic, 64
- genZipfTimeStamp
 - time stamp, 66
- get
 - BS::multi_future< T >, 297
- get_num_blocks
 - BS::blocks< T1, T2, T >, 99
- get_tasks_queued
 - BS::thread_pool, 396
- get_tasks_running
 - BS::thread_pool, 397
- get_tasks_total
 - BS::thread_pool, 397
- get_thread_count
 - BS::thread_pool, 397
- get_total_size
 - BS::blocks< T1, T2, T >, 99
- getAllTensors
 - CANDY::IVFListBucket, 265
 - CANDY::IVFListCell, 268
- getAllTensorsWithEncode
 - CANDY::IVFListBucket, 265
- getAvgCpu
 - INTELLI::MemoryTracker, 287
- getAvgMem
 - INTELLI::MemoryTracker, 287
- getCurMem
 - INTELLI::MemoryTracker, 287
- getData
 - CANDY::AbstractDataLoader, 79
 - CANDY::ExpFamilyDataLoader, 190
 - CANDY::FVECSDataLoader, 224
 - CANDY::HDF5DataLoader, 227
 - CANDY::RandomDataLoader, 362
 - CANDY::ZipfDataLoader, 445
- getDouble
 - Shared Utils, 48
- getDoubleMap
 - Shared Utils, 48
- getI64
 - Shared Utils, 48
- getI64Map
 - Shared Utils, 49
- getIndex
 - CANDY::IndexTable, 244
- getLatencyPercentage
 - INTELLI::UtilityFunctions, 414
- getMaxCpu
 - INTELLI::MemoryTracker, 287
- getMaxMem
 - INTELLI::MemoryTracker, 287
- getMinimumNumOfTensors
 - CANDY::IVFTensorEncodingList, 271
 - CANDY::YinYangGraph, 419
- getMinimumNumOfTensorsHamming
 - CANDY::IVFTensorEncodingList, 271
- getMinimumNumOfTensorsInsideBucket
 - CANDY::IVFTensorEncodingList, 272
- getMinimumNumOfTensorsInsideBucketHamming
 - CANDY::IVFTensorEncodingList, 272
- getMinimumTensorsUnderHamming
 - CANDY::IVFListBucket, 265
- getNeighbors
 - CANDY::KdTree, 276
 - CANDY::KmeansTree, 284
- getQuery
 - CANDY::AbstractDataLoader, 79
 - CANDY::ExpFamilyDataLoader, 190
 - CANDY::FVECSDataLoader, 224
 - CANDY::HDF5DataLoader, 227
 - CANDY::RandomDataLoader, 362
 - CANDY::ZipfDataLoader, 446
- getResultById
 - INTELLI::ThreadPerf, 405
 - INTELLI::ThreadPerfPAPI, 410
- getResultByName
 - INTELLI::ThreadPerf, 405
 - INTELLI::ThreadPerfPAPI, 410
- getStaicEnergyConsumption
 - DIVERSE_METER::AbstractMeter, 96

- getStr
 - CANDY::CANDYObject, 121
- getString
 - Shared Utils, 49
- getStrMap
 - Shared Utils, 50
- getTensorByIndex
 - CANDY::AbstractIndex, 85
 - CANDY::CongestionDropIndex, 145
 - CANDY::DistributedPartitionIndex, 169
 - CANDY::DPGIndex, 181
 - CANDY::FaissIndex, 193
 - CANDY::FlannIndex, 199
 - CANDY::FlatAMMIPIndex, 205
 - CANDY::FlatAMMIPObjIndex, 212
 - CANDY::FlatIndex, 218
 - CANDY::NNDescentIndex, 306
 - CANDY::ParallelPartitionIndex, 336
 - CANDY::PQIndex, 351
 - The support classes for index approaches, 27
- getTimeStamps
 - INTELLI::IntelliTimeStampGenerator, 260
- getU64
 - Shared Utils, 50
- getUnblockQueryResult
 - CANDY::DistributedIndexWorker, 161
- getVertex
 - CANDY::YinYangGraph_ListCell, 425
- getVertexWithEncode
 - CANDY::YinYangGraph_ListBucket, 423
- greedySearchForKNearestTensor
 - CANDY::YinYangVertex, 436
- greedySearchForKNearestVertex
 - CANDY::YinYangVertex, 436, 437
- greedySearchForNearestVertex
 - CANDY::YinYangVertex, 437, 438
- hash
 - CANDY::MLPBucketIdxModel, 293
 - CANDY::MLPHashingModel, 295
- hijackConfig
 - CANDY::AbstractDataLoader, 80
 - CANDY::ExpFamilyDataLoader, 190
- HNSWalter, 236
- HostPara, 242
- imbalance_factor
 - CANDY::Clustering, 135
- include/CANDY.h, 449
- include/CANDY/AbstractIndex.h, 450
- include/CANDY/BucketedFlatIndex.h, 451
- include/CANDY/BufferedCongestionDropIndex.h, 452
- include/CANDY/CANDYObject.h, 453
- include/CANDY/CongestionDropIndex.h, 454
- include/CANDY/CongestionDropIndex/CongestionDropIndexWorker.h, 455
- include/CANDY/DistributedPartitionIndex.h, 456
- include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h, 457
- include/CANDY/DPGIndex.h, 458
- include/CANDY/FaissIndex.h, 459
- include/CANDY/FlatAMMIPIndex.h, 459
- include/CANDY/FlatAMMIPObjIndex.h, 460
- include/CANDY/FlatIndex.h, 461
- include/CANDY/HNSWNaiveIndex.h, 462
- include/CANDY/IndexTable.h, 463
- include/CANDY/NNDescentIndex.h, 464
- include/CANDY/OnlinePQIndex.h, 465
- include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h, 466
- include/CANDY/OnlinePQIndex/SimpleStreamClustering.h, 467
- include/CANDY/ParallelPartitionIndex.h, 468
- include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h, 469
- include/CANDY/YinYangGraphIndex.h, 471
- include/CANDY/YinYangGraphIndex/YinYangGraph.h, 472
- include/CANDY/YinYangGraphSimpleIndex.h, 473
- include/CANDYPYTHON.h, 474
- include/DataLoader/AbstractDataLoader.h, 475
- include/DataLoader/DataLoaderTable.h, 475
- include/DataLoader/ExpFamilyDataLoader.h, 477
- include/DataLoader/FVECSDataLoader.h, 478
- include/DataLoader/HDF5DataLoader.h, 479
- include/DataLoader/RandomDataLoader.h, 480
- include/DataLoader/ZipfDataLoader.h, 481
- include/Utils/AbstractC20Thread.hpp, 482
- include/Utils/BS_thread_pool.hpp, 483
- include/Utils/C20Buffers.hpp, 484
- include/Utils/ConfigMap.hpp, 485
- include/Utils/IntelliTensorOP.hpp, 486
- include/Utils/IntelliTimeStampGenerator.h, 487
- include/Utils/Meters/AbstractMeter.hpp, 488
- include/Utils/Meters/EspMeterUart/EspMeterUart.hpp, 489
- include/Utils/ThreadPerf.hpp, 490
- include/Utils/ThreadPerfPAPI.hpp, 492
- index_create
 - CANDY::Rania_Python, 369
- index_delete
 - CANDY::Rania_Python, 369
- index_deleteString
 - CANDY::Rania_Python, 369
- index_editCfgDouble
 - CANDY::Rania_Python, 370
- index_editCfgI64
 - CANDY::Rania_Python, 370
- index_editCfgStr
 - CANDY::Rania_Python, 371
- index_endHPC
 - CANDY::Rania_Python, 371
- index_init
 - CANDY::Rania_Python, 371
- index_insert
 - CANDY::Rania_Python, 372
- index_insertString

- CANDY::Rania_Python, 372
- index_loadCfgFromFile
 - CANDY::Rania_Python, 372
- index_loadInitial
 - CANDY::Rania_Python, 373
- index_loadInitialString
 - CANDY::Rania_Python, 373
- index_offlineBuild
 - CANDY::Rania_Python, 374
- index_rawData
 - CANDY::Rania_Python, 374
- index_reset
 - CANDY::Rania_Python, 374
- index_revise
 - CANDY::Rania_Python, 375
- index_search
 - CANDY::Rania_Python, 375
- index_searchString
 - CANDY::Rania_Python, 376
- index_searchTensorAndStringList
 - CANDY::Rania_Python, 376
- index_setFrozenLevel
 - CANDY::Rania_Python, 376
- index_startHPC
 - CANDY::Rania_Python, 377
- index_waitPending
 - CANDY::Rania_Python, 377
- init
 - CANDY::IVFTensorEncodingList, 273
 - CANDY::MLPBucketIdxModel, 293
 - CANDY::MLPHashingModel, 295
 - CANDY::YinYangGraph, 420
 - CANDY::YinYangVertex, 438
- initEventsByCfg
 - INTELLI::ThreadPerf, 405
 - INTELLI::ThreadPerfPAPI, 411
- inlineMain
 - INTELLI::AbstractC20Thread, 78
 - The support classes for index approaches, 28
- insertRows
 - INTELLI::IntelliTensorOP, 252, 254
- insertSingleRowTensor
 - CANDY::YinYangGraphSimpleIndex, 431
- insertStringObject
 - CANDY::AbstractIndex, 86
 - CANDY::CongestionDropIndex, 146
 - CANDY::FlatAMMIPObjIndex, 213
 - CANDY::ParallelPartitionIndex, 337
 - The support classes for index approaches, 28
- insertTensor
 - CANDY::AbstractIndex, 86
 - CANDY::BucketedFlatIndex, 104
 - CANDY::BufferedCongestionDropIndex, 112
 - CANDY::CongestionDropIndex, 146
 - CANDY::DistributedIndexWorker, 162
 - CANDY::DistributedPartitionIndex, 169
 - CANDY::DIW_RayWrapper, 175
 - CANDY::DPGIndex, 182
 - CANDY::FaissIndex, 194
 - CANDY::FlannIndex, 199
 - CANDY::FlatAMMIPObjIndex, 205
 - CANDY::FlatAMMIPObjIndex, 213
 - CANDY::FlatIndex, 219
 - CANDY::HNSWNaiveIndex, 238
 - CANDY::IVFListCell, 268
 - CANDY::NNDescentIndex, 306
 - CANDY::OnlineIVFLSHIndex, 320
 - CANDY::OnlinePQIndex, 326
 - CANDY::ParallelPartitionIndex, 337
 - CANDY::PQIndex, 351
 - CANDY::YinYangGraph_ListCell, 425
 - CANDY::YinYangGraphIndex, 429
 - CANDY::YinYangGraphSimpleIndex, 432
 - The support classes for index approaches, 29
- insertTensorInline
 - CANDY::BufferedCongestionDropIndex, 112
- insertTensorPtr
 - CANDY::IVFListCell, 269
- insertTensorWithEncode
 - CANDY::IVFListBucket, 266
 - CANDY::IVFTensorEncodingList, 273
 - CANDY::YinYangGraph, 420
 - CANDY::YinYangGraph_ListBucket, 423
- INTELLI::AbstractC20Thread, 77
 - inlineMain, 78
- INTELLI::C20Buffer< dataType >, 116
 - append, 117, 118
 - bufferSize, 118
 - C20Buffer, 117
 - data, 119
 - size, 119
- INTELLI::ConfigMap, 139
- INTELLI::IntelliLog, 244
- INTELLI::IntelliLog_FileProtector, 245
- INTELLI::IntelliTensorOP, 245
 - appendRows, 246, 247
 - appendRowsBufferMode, 247, 248
 - deleteRow, 248, 249
 - deleteRowBufferMode, 249
 - deleteRows, 250
 - deleteRowsBufferMode, 251
 - editRows, 251, 252
 - insertRows, 252, 254
 - l2Normalize, 254
 - rowSampling, 255
 - tensorFromFile, 255
 - tensorFromFlatBin, 255
 - tensorToFile, 256
 - tensorToFlatBin, 256
- INTELLI::IntelliTimeStamp, 257
- INTELLI::IntelliTimeStampGenerator, 258
 - generateArrival, 260
 - getTimeStamps, 260
 - setConfig, 260
- INTELLI::MemoryTracker, 286
 - getAvgCpu, 287

- getAvgMem, [287](#)
- getCurMem, [287](#)
- getMaxCpu, [287](#)
- getMaxMem, [287](#)
- start, [288](#)
- INTELLI::MicroDataSet, [291](#)
- INTELLI::SPSCQueue< T, Allocator >, [388](#)
- INTELLI::ThreadPerf, [402](#)
 - getResultById, [405](#)
 - getResultByName, [405](#)
 - initEventsByCfg, [405](#)
 - resultToConfigMap, [406](#)
 - start, [406](#)
 - ThreadPerf, [404](#)
- INTELLI::ThreadPerf::PerfPair, [343](#)
- INTELLI::ThreadPerf::PerfTool, [344](#)
- INTELLI::ThreadPerfPAPI, [407](#)
 - addPapiTag, [409](#), [410](#)
 - getResultById, [410](#)
 - getResultByName, [410](#)
 - initEventsByCfg, [411](#)
 - resultToConfigMap, [411](#)
 - start, [411](#)
 - ThreadPerfPAPI, [409](#)
- INTELLI::UtilityFunctions, [413](#)
 - bind2Core, [413](#)
 - calculateRecall, [414](#)
 - getLatencyPercentage, [414](#)
 - saveTimeStampToFile, [414](#)
 - tensorListFromFile, [415](#)
 - tensorListToFile, [415](#)
- INTELLITensorOP, [257](#)
- is_paused
 - BS::thread_pool, [397](#)
- knnSearch
 - CANDY::KdTree, [277](#)
 - CANDY::KmeansTree, [284](#)
- l2Normalize
 - INTELLI::IntelliTensorOP, [254](#)
- loadCentroids
 - CANDY::SimpleStreamClustering, [386](#), [387](#)
- loadFrom
 - Shared Utils, [50](#)
- loadInitialStringObject
 - CANDY::AbstractIndex, [87](#)
 - CANDY::CongestionDropIndex, [147](#)
 - CANDY::ParallelPartitionIndex, [338](#)
 - The support classes for index approaches, [29](#)
- loadInitialTensor
 - CANDY::AbstractIndex, [88](#)
 - CANDY::BucketedFlatIndex, [105](#)
 - CANDY::BufferedCongestionDropIndex, [113](#)
 - CANDY::CongestionDropIndex, [147](#)
 - CANDY::DistributedIndexWorker, [162](#)
 - CANDY::DistributedPartitionIndex, [169](#)
 - CANDY::DIW_RayWrapper, [175](#)
 - CANDY::DPGIndex, [182](#)
 - CANDY::FaissIndex, [194](#)
 - CANDY::FlannIndex, [199](#)
 - CANDY::NNDescentIndex, [306](#)
 - CANDY::OnlineIVFL2HIndex, [315](#)
 - CANDY::OnlinePQIndex, [326](#)
 - CANDY::ParallelPartitionIndex, [338](#)
 - CANDY::PQIndex, [352](#)
 - The support classes for index approaches, [30](#)
- loadInitialTensorAndQueryDistribution
 - CANDY::AbstractIndex, [88](#)
 - CANDY::OnlineIVFL2HIndex, [316](#)
- loadInitialTensorUnblocked
 - CANDY::DistributedIndexWorker, [162](#)
- log
 - Log utils, [56](#)
- Log utils, [55](#)
 - appendLogFile, [56](#)
 - log, [56](#)
 - openLogFile, [57](#)
 - setupLoggingFile, [57](#)
- meanSplit
 - CANDY::KdTree, [277](#)
- MeterTable
 - DIVERSE_METER::MeterTable, [290](#)
- MicroDataSet
 - The Micro dataset, [61](#)
- ms
 - BS::timer, [412](#)
- multi_future
 - BS::multi_future< T >, [297](#)
- nb_neighbors
 - CANDY::HNSW, [233](#)
- nearestKVertexWithinMap
 - CANDY::YinYangVertexMap, [442](#)
- nearestVertexWithinMap
 - CANDY::YinYangVertexMap, [443](#)
- nearestVertexWithinMe
 - CANDY::YinYangVertexMap, [443](#)
- neighbor_range
 - CANDY::HNSW, [233](#)
- newAbstractIndex
 - The bottom tier of indexing alorithms, [39](#)
- offlineBuild
 - CANDY::AbstractIndex, [89](#)
 - CANDY::BufferedCongestionDropIndex, [113](#)
 - CANDY::CongestionDropIndex, [148](#)
 - CANDY::DistributedIndexWorker, [163](#)
 - CANDY::DistributedPartitionIndex, [170](#)
 - CANDY::DIW_RayWrapper, [176](#)
 - CANDY::DPGIndex, [183](#)
 - CANDY::NNDescentIndex, [307](#)
 - CANDY::OnlinePQIndex, [327](#)
 - CANDY::ParallelPartitionIndex, [339](#)
 - The support classes for index approaches, [30](#)
- offlineBuildUnblocked
 - CANDY::DistributedIndexWorker, [163](#)

- ONLINEIVF_NEXT_POW_2
 - OnlineIVFLSHIndex.cpp, [499](#)
 - YinYangGraphIndex.cpp, [501](#)
- ONLINEIVFL2H_NEXT_POW_2
 - OnlineIVFL2HIndex.cpp, [498](#)
- OnlineIVFL2HIndex.cpp
 - ONLINEIVFL2H_NEXT_POW_2, [498](#)
- OnlineIVFLSHIndex.cpp
 - ONLINEIVF_NEXT_POW_2, [499](#)
- openLogFile
 - Log utils, [57](#)
- operator()
 - CANDY::DistanceQueryer, [158](#)
- operator[]
 - BS::multi_future< T >, [298](#)
- Other common class or package under C++20 standard, [53](#)
- parallelize_loop
 - BS::thread_pool, [397](#), [398](#)
- planeSplit
 - CANDY::KdTree, [278](#)
- prepare_level_tab
 - CANDY::HNSW, [234](#)
- print
 - BS::synced_stream, [390](#)
- println
 - BS::synced_stream, [390](#)
- push_back
 - BS::multi_future< T >, [298](#)
- push_loop
 - BS::thread_pool, [399](#), [400](#)
- push_task
 - BS::thread_pool, [400](#)
- random_level
 - CANDY::HNSW, [234](#)
- rawData
 - CANDY::AbstractIndex, [89](#)
 - CANDY::CongestionDropIndex, [148](#)
 - CANDY::DistributedPartitionIndex, [170](#)
 - CANDY::DPGIndex, [183](#)
 - CANDY::FlatAMMIPIndex, [207](#)
 - CANDY::FlatAMMIPObjIndex, [213](#)
 - CANDY::FlatIndex, [219](#)
 - CANDY::NNDescentIndex, [307](#)
 - CANDY::ParallelPartitionIndex, [339](#)
 - CANDY::PQIndex, [352](#)
 - The support classes for index approaches, [31](#)
- registerNewDataLoader
 - CANDY::DataLoaderTable, [156](#)
- registerNewMeter
 - DIVERSE_METER::MeterTable, [290](#)
- reset
 - BS::thread_pool, [401](#)
- resultToConfigMap
 - INTELLI::ThreadPerf, [406](#)
 - INTELLI::ThreadPerfPAPI, [411](#)
- reviseTensor
 - CANDY::AbstractIndex, [89](#)
 - CANDY::BucketedFlatIndex, [105](#)
 - CANDY::BufferedCongestionDropIndex, [114](#)
 - CANDY::CongestionDropIndex, [148](#)
 - CANDY::DistributedPartitionIndex, [170](#)
 - CANDY::DPGIndex, [183](#)
 - CANDY::FlatAMMIPIndex, [207](#)
 - CANDY::FlatAMMIPObjIndex, [214](#)
 - CANDY::FlatIndex, [219](#)
 - CANDY::HNSWNaiveIndex, [239](#)
 - CANDY::NNDescentIndex, [307](#)
 - CANDY::OnlineIVFLSHIndex, [321](#)
 - CANDY::OnlinePQIndex, [327](#)
 - CANDY::ParallelPartitionIndex, [339](#)
 - CANDY::PQIndex, [352](#)
 - The support classes for index approaches, [31](#)
- rowSampling
 - INTELLI::IntelliTensorOP, [255](#)
- saveCentroidsToFile
 - CANDY::SimpleStreamClustering, [387](#)
- saveTimeStampToFile
 - INTELLI::UtilityFunctions, [414](#)
- search
 - CANDY::HNSW, [234](#)
 - CANDY::ProductQuantizer, [359](#)
- search_bounded_queue
 - CANDY::HNSW, [236](#)
- searchIndex
 - CANDY::AbstractIndex, [90](#)
 - CANDY::FaissIndex, [195](#)
 - CANDY::FlannIndex, [200](#)
 - CANDY::FlatAMMIPIndex, [207](#)
 - CANDY::FlatAMMIPObjIndex, [214](#)
 - CANDY::FlatIndex, [220](#)
 - CANDY::PQIndex, [353](#)
 - The support classes for index approaches, [31](#)
- searchLevel
 - CANDY::KdTree, [278](#)
- searchSingleRow
 - CANDY::BucketedFlatIndex, [106](#)
- searchStringObject
 - CANDY::AbstractIndex, [90](#)
 - CANDY::CongestionDropIndex, [149](#)
 - CANDY::FlatAMMIPObjIndex, [215](#)
 - CANDY::ParallelPartitionIndex, [340](#)
 - The support classes for index approaches, [32](#)
- searchTensor
 - CANDY::AbstractIndex, [91](#)
 - CANDY::BucketedFlatIndex, [106](#)
 - CANDY::BufferedCongestionDropIndex, [114](#)
 - CANDY::CongestionDropIndex, [149](#)
 - CANDY::DistributedIndexWorker, [163](#)
 - CANDY::DistributedPartitionIndex, [171](#)
 - CANDY::DIW_RayWrapper, [176](#)
 - CANDY::DPGIndex, [184](#)
 - CANDY::FaissIndex, [195](#)
 - CANDY::FlannIndex, [200](#)
 - CANDY::FlatAMMIPIndex, [208](#)

- CANDY::FlatAMMIPObjIndex, 215
- CANDY::FlatIndex, 220
- CANDY::HNSWNaiveIndex, 239
- CANDY::NNDescentIndex, 308
- CANDY::OnlineIVFLSHIndex, 321
- CANDY::OnlinePQIndex, 328
- CANDY::ParallelPartitionIndex, 340
- CANDY::PQIndex, 353
- CANDY::YinYangGraphIndex, 429
- CANDY::YinYangGraphSimpleIndex, 432
- The support classes for index approaches, 32
- searchTensorAndStringObject
 - CANDY::AbstractIndex, 91
 - CANDY::CongestionDropIndex, 150
 - CANDY::ParallelPartitionIndex, 341
 - The support classes for index approaches, 33
- searchTensorUnblock
 - CANDY::DistributedIndexWorker, 164
- selectDivision
 - CANDY::KdTree, 279
- set_nb_neighbors
 - CANDY::HNSW, 235
- set_probs
 - CANDY::HNSW, 235
- setCentroidsFrom
 - CANDY::ProductQuantizer, 359
- setConfig
 - CANDY::AbstractDataLoader, 80
 - CANDY::AbstractIndex, 92
 - CANDY::BucketedFlatIndex, 107
 - CANDY::BufferedCongestionDropIndex, 115
 - CANDY::CongestionDropIndex, 150
 - CANDY::DistributedIndexWorker, 164
 - CANDY::DistributedPartitionIndex, 171
 - CANDY::DIW_RayWrapper, 176
 - CANDY::DPGIndex, 184
 - CANDY::ExpFamilyDataLoader, 191
 - CANDY::FaissIndex, 195
 - CANDY::FlannIndex, 201
 - CANDY::FlatAMMIPIndex, 208
 - CANDY::FlatAMMIPObjIndex, 215
 - CANDY::FlatIndex, 221
 - CANDY::FVECSDataLoader, 224
 - CANDY::HDF5DataLoader, 228
 - CANDY::HNSWNaiveIndex, 240
 - CANDY::KdTree, 279
 - CANDY::KmeansTree, 285
 - CANDY::NNDescentIndex, 308
 - CANDY::OnlineIVFL2HIndex, 316
 - CANDY::OnlineIVFLSHIndex, 322
 - CANDY::OnlinePQIndex, 328
 - CANDY::ParallelPartitionIndex, 341
 - CANDY::PQIndex, 354
 - CANDY::RandomDataLoader, 363
 - CANDY::YinYangGraphIndex, 429
 - CANDY::YinYangGraphSimpleIndex, 432
 - CANDY::ZipfDataLoader, 446
 - DIVERSE_METER::AbstractMeter, 96
 - DIVERSE_METER::EspMeterUart, 187
 - DIVERSE_METER::IntelMeter, 262
 - INTELLI::IntelliTimeStampGenerator, 260
 - The support classes for index approaches, 33, 34
- setConfigClass
 - CANDY::AbstractIndex, 92
- setFrozenLevel
 - CANDY::AbstractIndex, 93
 - CANDY::BufferedCongestionDropIndex, 115
 - CANDY::CongestionDropIndex, 151
 - CANDY::DistributedIndexWorker, 164
 - CANDY::DistributedPartitionIndex, 172
 - CANDY::DIW_RayWrapper, 177
 - CANDY::DPGIndex, 185
 - CANDY::NNDescentIndex, 309
 - CANDY::OnlinePQIndex, 329
 - CANDY::ParallelPartitionIndex, 342
 - CANDY::PQIndex, 354
 - The support classes for index approaches, 34
- setParams
 - CANDY::FlannComponent, 197
 - CANDY::KdTree, 279
 - CANDY::KmeansTree, 285
- setSeed
 - The Micro dataset, 61
- setStaticPower
 - DIVERSE_METER::AbstractMeter, 96
- setStr
 - CANDY::CANDYObject, 121
- setTier
 - CANDY::AbstractIndex, 93
- setUpLayer
 - CANDY::YinYangVertex, 439
- setupLoggingFile
 - Log utils, 57
- Shared Utils, 41
 - cloneInto, 43
 - edit, 44
 - exist, 45
 - existDouble, 45
 - existI64, 45
 - existString, 46
 - existU64, 46
 - fromCArg, 47
 - fromFile, 47
 - fromString, 47
 - getDouble, 48
 - getDoubleMap, 48
 - getI64, 48
 - getI64Map, 49
 - getString, 49
 - getStrMap, 50
 - getU64, 50
 - loadFrom, 50
 - toFile, 51
 - toString, 51
 - tryDouble, 51
 - tryI64, 52

- tryString, [52](#)
- tryU64, [53](#)
- size
 - BS::multi_future< T >, [298](#)
 - CANDY::FlatAMMIPIndex, [209](#)
 - CANDY::FlatAMMIPObjIndex, [216](#)
 - CANDY::FlatIndex, [221](#)
 - INTELLI::C20Buffer< dataType >, [119](#)
- sizeWithEncode
 - CANDY::IVFLListBucket, [266](#)
- splitClusters
 - CANDY::Clustering, [136](#)
- src/CANDY/AbstractIndex.cpp, [493](#)
- src/CANDY/BucketedFlatIndex.cpp, [493](#)
- src/CANDY/BufferedCongestionDropIndex.cpp, [494](#)
- src/CANDY/CongestionDropIndex.cpp, [494](#)
- src/CANDY/CongestionDropIndex/CongestionDropIndexWorker.cpp, [495](#)
- src/CANDY/DistributedPartitionIndex.cpp, [495](#)
- src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp, [495](#)
- src/CANDY/DPGIndex.cpp, [496](#)
- src/CANDY/FlatAMMIPIndex.cpp, [496](#)
- src/CANDY/FlatAMMIPObjIndex.cpp, [496](#)
- src/CANDY/FlatIndex.cpp, [497](#)
- src/CANDY/NNDescentIndex.cpp, [497](#)
- src/CANDY/OnlineIVFL2HIndex.cpp, [498](#)
- src/CANDY/OnlineIVFLSHIndex.cpp, [498](#)
- src/CANDY/OnlinePQIndex.cpp, [499](#)
- src/CANDY/ParallelPartitionIndex.cpp, [500](#)
- src/CANDY/ParallelPartitionIndex/ParallelIndexWorker.cpp, [500](#)
- src/CANDY/YinYangGraphIndex.cpp, [500](#)
- src/CANDY/YinYangGraphSimpleIndex.cpp, [501](#)
- start
 - BS::blocks< T1, T2, T >, [100](#)
 - INTELLI::MemoryTracker, [288](#)
 - INTELLI::ThreadPerf, [406](#)
 - INTELLI::ThreadPerfPAPI, [411](#)
- startHPC
 - CANDY::AbstractIndex, [93](#)
 - CANDY::BufferedCongestionDropIndex, [115](#)
 - CANDY::CongestionDropIndex, [151](#)
 - CANDY::DistributedIndexWorker, [165](#)
 - CANDY::DistributedPartitionIndex, [172](#)
 - CANDY::DIW_RayWrapper, [177](#)
 - CANDY::DPGIndex, [185](#)
 - CANDY::NNDescentIndex, [309](#)
 - CANDY::ParallelPartitionIndex, [342](#)
 - The support classes for index approaches, [35](#)
- submit
 - BS::thread_pool, [401](#)
- synced_stream
 - BS::synced_stream, [390](#)
- tensor operations, [58](#)
- tensorFromFile
 - CANDY::Rania_Python, [378](#)
 - INTELLI::IntelliTensorOP, [255](#)
- tensorFromFlatBin
 - INTELLI::IntelliTensorOP, [255](#)
- tensorFromFVECS
 - CANDY::FVECSDataLoader, [225](#)
 - CANDY::Rania_Python, [378](#)
- tensorFromHDF5
 - CANDY::HDF5DataLoader, [228](#)
 - CANDY::Rania_Python, [378](#)
- tensorListFromFile
 - INTELLI::UtilityFunctions, [415](#)
- tensorListToFile
 - INTELLI::UtilityFunctions, [415](#)
- tensorToFile
 - CANDY::Rania_Python, [379](#)
 - INTELLI::IntelliTensorOP, [256](#)
- tensorToFlatBin
 - INTELLI::IntelliTensorOP, [256](#)
- testStaticPower
 - DIVERSE_METER::AbstractMeter, [97](#)
- The bottom tier of indexing alorithms, [36](#)
 - newAbstractIndex, [39](#)
- The data loaders of CANDY, [35](#)
- The main body of CANDY's indexing approaches, [36](#)
- The Micro dataset, [60](#)
 - MicroDataSet, [61](#)
 - setSeed, [61](#)
- The support classes for index approaches, [23](#)
 - deleteStringObject, [26](#)
 - deleteTensor, [27](#)
 - endHPC, [27](#)
 - getTensorByIndex, [27](#)
 - inlineMain, [28](#)
 - insertStringObject, [28](#)
 - insertTensor, [29](#)
 - loadInitialStringObject, [29](#)
 - loadInitialTensor, [30](#)
 - offlineBuild, [30](#)
 - rawData, [31](#)
 - reviseTensor, [31](#)
 - searchIndex, [31](#)
 - searchStringObject, [32](#)
 - searchTensor, [32](#)
 - searchTensorAndStringObject, [33](#)
 - setConfig, [33](#), [34](#)
 - setFrozenLevel, [34](#)
 - startHPC, [35](#)
- The upper tier of indexing alorithms, can be container of other indexing ways, [39](#)
- thread_pool
 - BS::thread_pool, [396](#)
- ThreadPerf
 - INTELLI::ThreadPerf, [404](#)
- ThreadPerfPAPI
 - INTELLI::ThreadPerfPAPI, [409](#)
- time stamp, [65](#)
 - genSmoothTimeStamp, [65](#)
 - genZipfTimeStamp, [66](#)
- time stamps, [58](#)

- toFile
 - Shared Utils, [51](#)
- TONY_CL_HOST::CLContainer, [132](#)
- toString
 - CANDY::YinYangVertex, [439](#)
 - Shared Utils, [51](#)
- train
 - CANDY::Clustering, [136](#)
 - CANDY::PQIndex, [354](#)
 - CANDY::ProductQuantizer, [360](#)
- trainModel
 - CANDY::MLPBucketIdxModel, [294](#)
 - CANDY::MLPHashingModel, [296](#)
- trainModelWithData
 - CANDY::OnlineIVFL2HIndex, [317](#)
- tryDouble
 - Shared Utils, [51](#)
- tryI64
 - Shared Utils, [52](#)
- tryString
 - Shared Utils, [52](#)
- tryToConnect
 - CANDY::YinYangVertex, [439](#)
- tryU64
 - Shared Utils, [53](#)
- VertexComparison, [416](#)
- VisitedBitset, [417](#)
- waitPendingOperations
 - CANDY::AbstractIndex, [94](#)
 - CANDY::CongestionDropIndex, [151](#)
 - CANDY::DistributedIndexWorker, [165](#)
 - CANDY::DistributedPartitionIndex, [172](#)
 - CANDY::DIW_RayWrapper, [177](#)
 - CANDY::ParallelPartitionIndex, [342](#)
- YinYangGraphIndex.cpp
 - ONLINEIVF_NEXT_POW_2, [501](#)