CANDY

0.1.0

Generated by Doxygen 1.9.1

Chapter 1

Introduction

This project is an index library and benchmark kit for online vector management, covering various AKNN algos, datasets, online insert benchmark, and examples for more fancy downstream tasks.

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
- congestionDrop CongestionDropIndex
- bufferedCongestionDrop BufferedCongestionDropIndex
- flatAMMIP FlatAMMIPIndex

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 Candy_Python for python APIs (old style)
- Please enable pybind build and install the ∗.so to system path, you can import PyCANDY, see benchmark/scripts/PyCANDY for details

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 travsing 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 travsing 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, YinYangVertex←↩ Ptr entryPoint, int64_t k, bool ignoreYin, bool forceTheSameLevel, floatDistanceFunction_t df=Yin←↩ YangGraph_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=YinYang←↩ Graph_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:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

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

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-specfic 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 inited status
- virtual bool CANDY::ParallelIndexWorker::setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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::searc (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

t	the tensor, some index need to be single row
k	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

t	the tensor, some index needs to be single row
k	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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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

t	the tensor, some index need to be single row
strs	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

t	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

t	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

t	the tensor, some index need to be single row •
strs	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

t	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

t	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

t	the tensor to be revised
w	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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-specfic config related to one index

Parameters

cfg	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-specfic config related to one index

Parameters

cfg	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

frozenLv	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

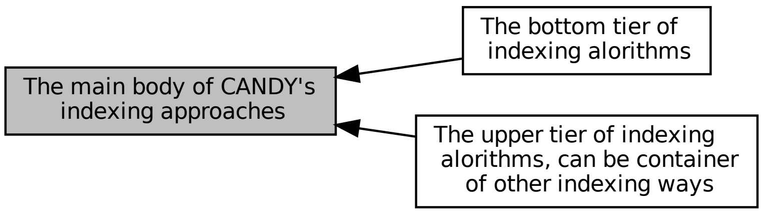
7.2.1.2 DataLoader

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:


The main body of CANDY's indexing approaches
The bottom tier of indexing alorithms
The upper tier of indexing alorithms, can be container of other indexing ways

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.

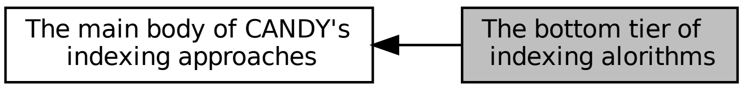
7.3.1 Detailed Description

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:


The main body of CANDY's indexing approaches
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 hierachical
- 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 simpe yinyang graph,there is no LSH search is only within the linked yinyanggraph.

Macros

- #define newParallelIndexWorker std::make_shared<CANDY::ParallelIndexWorker>

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

  (Macro) To creat 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

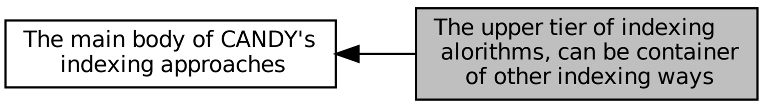
```
#define newAbstractIndex std::make_shared<CANDY::AbstractIndex>
```

(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:


The main body of CANDY's indexing approaches
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>

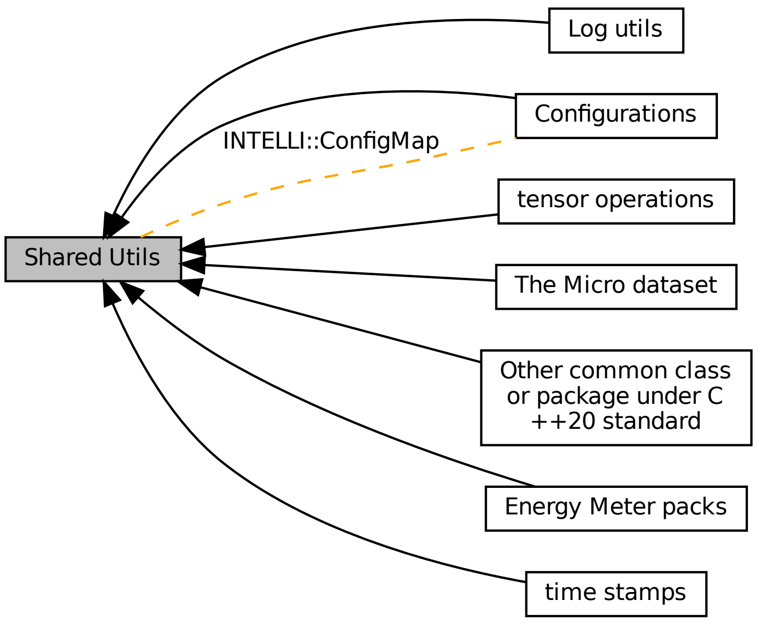
  (Macro) To creat a new ParallelPartitionIndex shared pointer.
- typedef std::shared_ptr< class CANDY::ParallelPartitionIndex > CANDY::ParallelPartitionIndexPtr

  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 new↩ Line="\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 new↩ Line="\n")

  convert the whole map to file
- bool INTELLI::ConfigMap::fromFile (const std::string &fname, std::string separator=",", std::string new↩ Line="\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 show↩ Warning=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

dest	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

key	The look up key in std::string
value	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

key	The look up key in std::string
value	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

key	The look up key in std::string
value	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 exit the config, will create new, or will overwrite.

Parameters

key	The look up key in std::string
value	The u64 value

7.6.2.6 exist()

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

To detect whether the key exists.

Parameters

key	

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

key	

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

key	

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

key	

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

key	

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

argc	the count of input args
argv	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

fname	The file name
separator	The separator std::string, default "," for csv style
newLine	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

src,the	string
separator	The separator std::string, default "\t"
newLine	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

key	

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 I64

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

key	

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

key	

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

key	

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

src	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

fname	The file name
separator	The separator std::string, default "," for csv style
newLine	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 retuen

Parameters

separator	The separator std::string, default "\t"
newLine	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

key	The key
defaultValue	The default
showWarning	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

key	The key
defaultValue	The default
showWarning	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

key	The key
defaultValue	The default
showWarning	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

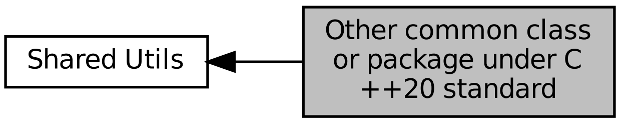
key	The key
defaultValue	The default
showWarning	Whether show warning logs if not found

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

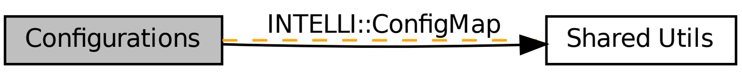
  The class to describe a shared pointer to ThreadPerfPAPI.

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

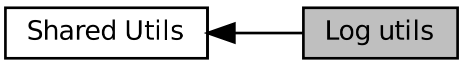
  The class to describe a shared pointer to ConfigMap.

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) 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

msg	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

level	The log level you want to indicate
message	The log message you want to indicate
source	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

fname	The name of file

7.9.2.4 setupLoggingFile()

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

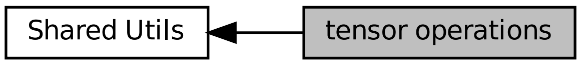
set up the logging file by its name

Parameters

fname	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

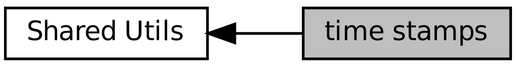
  The class to describe a shared pointer to torch::Tensor.

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

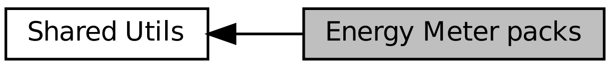
  The class to describe a shared pointer to IntelliTimeStamp.

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.

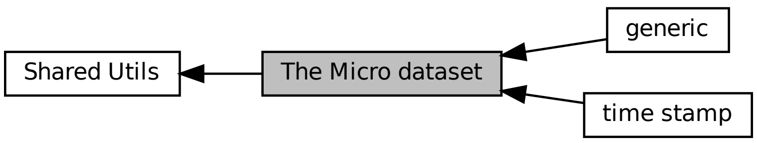
- typedef std::shared_ptr< DIVERSE_METER::AbstractMeter > **DIVERSE_METER::AbstractMeterPtr**

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--2
  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

seed	The seed for random generator

7.13.2.2 setSeed()

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

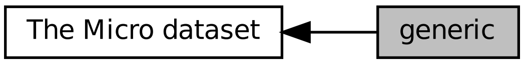
construction with seed

Parameters

seed	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

dType	The data type in the alphabet, default uint32_t

Parameters

len	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

tsType	The data type, default uint32_t
genType	The generator type, default mt19937 (32 bit rand)

Parameters

len	The length of output vector
maxV	The maximum value of output
minV	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

tsType	The data type of int, default is size_t
len	The length of output vector
maxV	The maximum value of integer
fac	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

dType	The data type in the alphabet, default double

Parameters

len	The length of alphabet
fac	The zipf factor, in [0,1]

Returns

The output vector lut

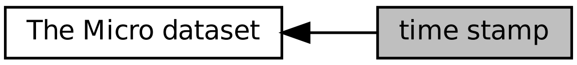
Compute scaling factor such that

sum (lut[i], i=1..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

tsType	The data type of time stamp, default is size_t

Parameters

len	The length of output vector
step	Within the step, timestamp will remain the same
interval	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

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

Returns

the output vector

See also

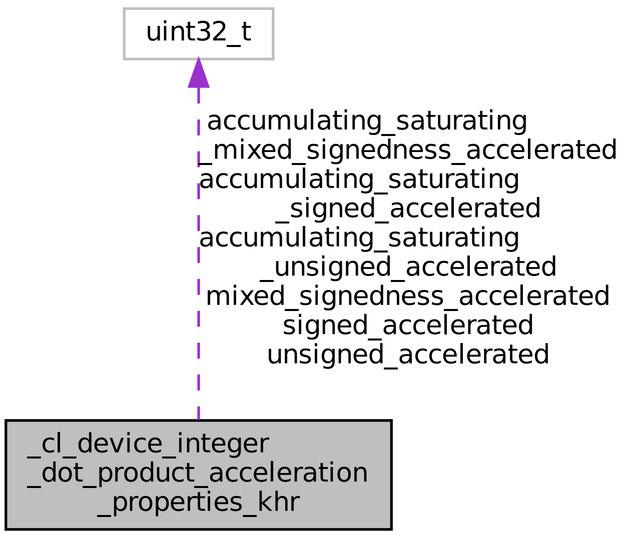
genZipfInt

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

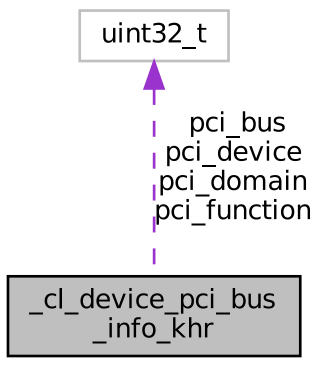
- cl_bool **signed_accelerated**
- cl_bool **unsigned_accelerated**
- cl_bool **mixed_signedness_accelerated**
- cl_bool **accumulating_saturating_signed_accelerated**
- cl_bool **accumulating_saturating_unsigned_accelerated**
- cl_bool **accumulating_saturating_mixed_signedness_accelerated**

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

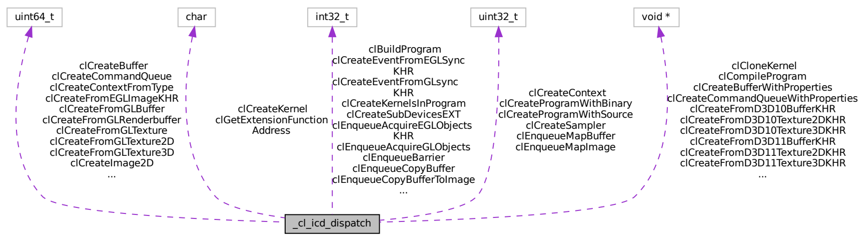
- cl_uint **pci_domain**
- cl_uint **pci_bus**
- cl_uint **pci_device**
- cl_uint **pci_function**

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**

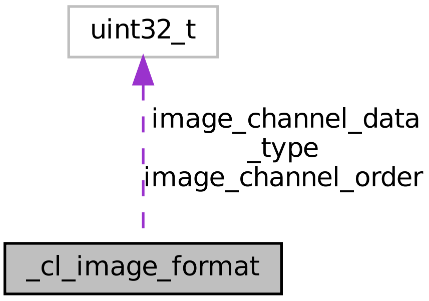
- cl_api_clCreateFromGLTexture **clCreateFromGLTexture**
- cl_api_clGetDeviceIDsFromD3D11KHR **clGetDeviceIDsFromD3D11KHR**
- cl_api_clCreateFromD3D11BufferKHR **clCreateFromD3D11BufferKHR**
- cl_api_clCreateFromD3D11Texture2DKHR **clCreateFromD3D11Texture2DKHR**
- cl_api_clCreateFromD3D11Texture3DKHR **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_clEnqueueSVMMigrateMem **clEnqueueSVMMigrateMem**
- 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

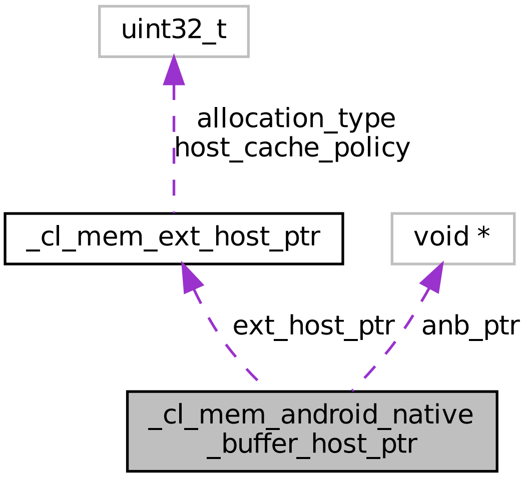
- cl_channel_order **image_channel_order**
- cl_channel_type **image_channel_data_type**

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

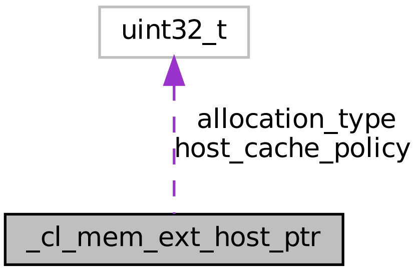
- cl_mem_ext_host_ptr **ext_host_ptr**
- void ∗ **anb_ptr**

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

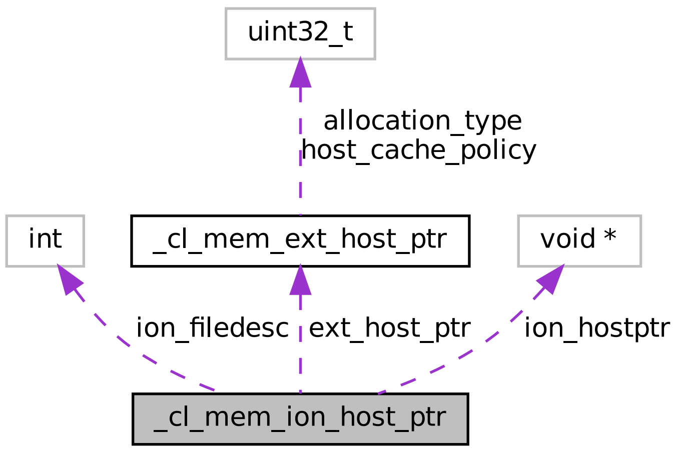
- cl_uint **allocation_type**
- cl_uint **host_cache_policy**

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

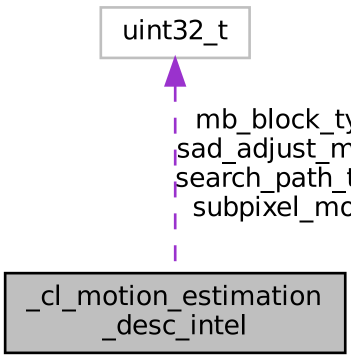
- cl_mem_ext_host_ptr **ext_host_ptr**
- int **ion_filedesc**
- void ∗ **ion_hostptr**

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

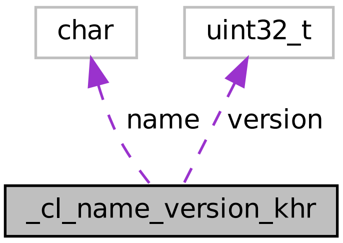
- cl_uint **mb_block_type**
- cl_uint **subpixel_mode**
- cl_uint **sad_adjust_mode**
- cl_uint **search_path_type**

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

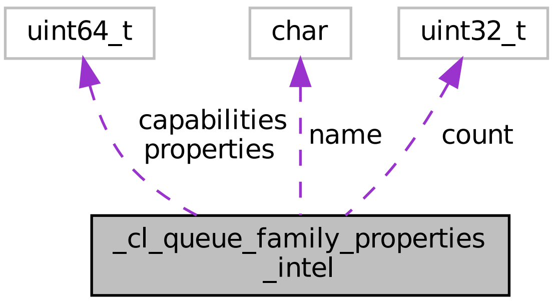
- cl_version_khr **version**
- char **name** [CL_NAME_VERSION_MAX_NAME_SIZE_KHR]

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:

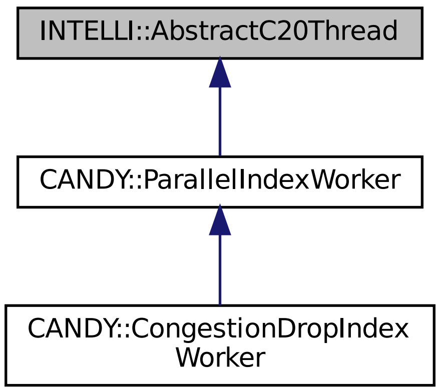
- include/CL/cl_ext.h

8.11 INTELLI::AbstractC20Thread Class Reference

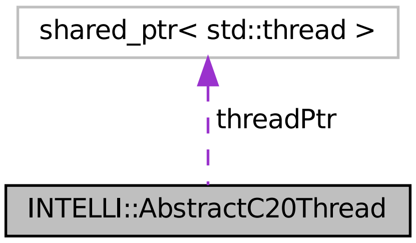
The base class and abstraction of C++20 thread, and it can be derived into other threads.

`#include <Utils/AbstractC20Thread.hpp>`

Inheritance diagram for INTELLI::AbstractC20Thread:


INTELLI::AbstractC20Thread
CANDY::ParallelIndexWorker
CANDY::CongestionDropIndex Worker

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:

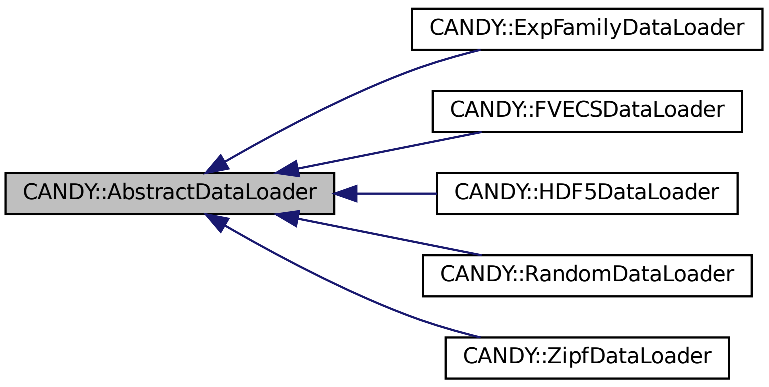
- include/Utils/AbstractC20Thread.hpp

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

cfg	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

cfg	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:

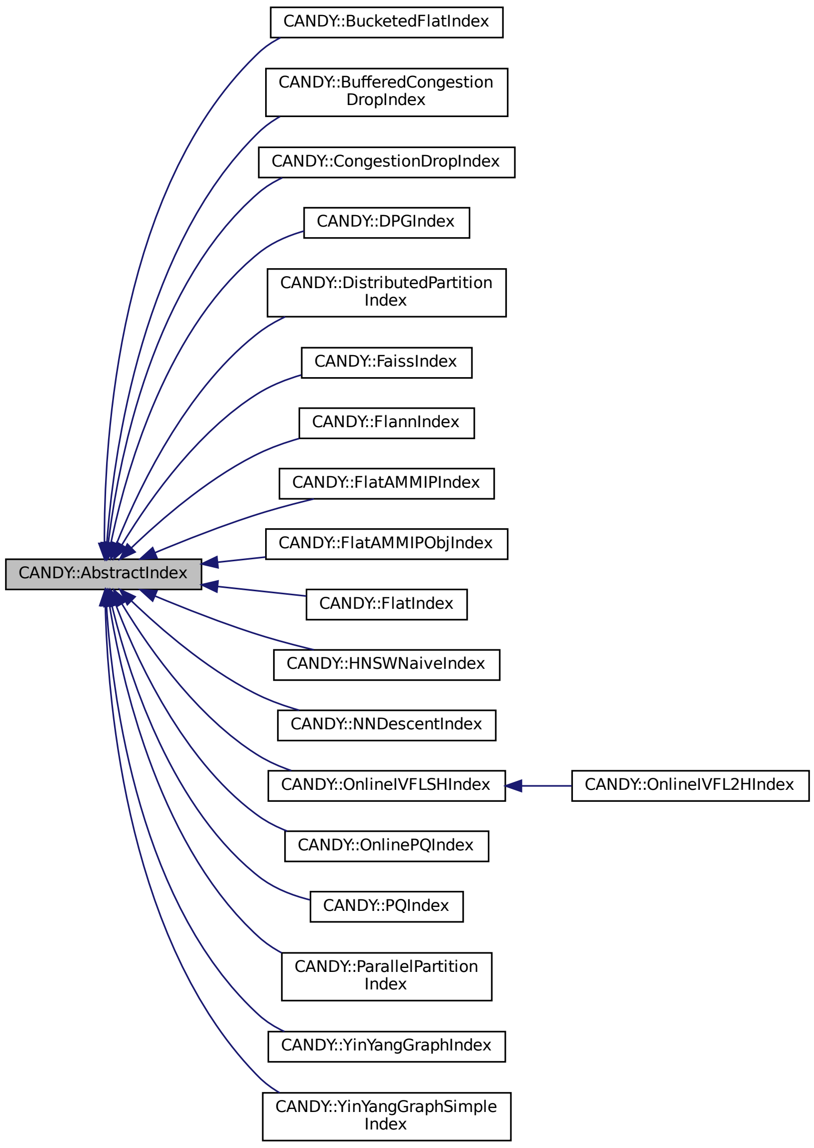
- include/DataLoader/AbstractDataLoader.h
- src/DataLoader/AbstractDataLoader.cpp

8.13 CANDY::AbstractIndex Class Reference

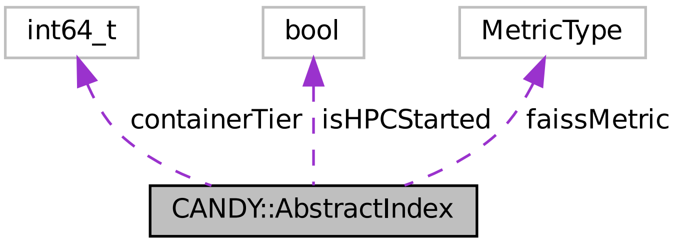
The abstract class of an index approach.

```
#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 inited 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-specfic 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::FlatAMMIPIndex, 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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

t	the tensor, some index need to be single row
strs	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

t	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

t	the tensor, some index need to be single row •
strs	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

t	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

t	the data tensor
query	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

t	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

t	the tensor to be revised
w	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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

t	the tensor, allow multiple rows
k	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::FlatAMMIPIndex, 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

t	the tensor, allow multiple rows
k	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-specfic config related to one index

Parameters

cfg	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::FlatAMMIPObjIndex, 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

cfg	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

frozenLv	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

tie	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:

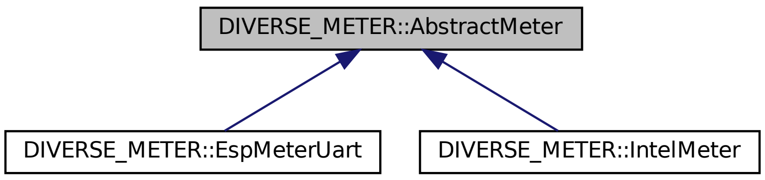
- include/CANDY/AbstractIndex.h
- src/CANDY/AbstractIndex.cpp

8.14 DIVERSE_METER::AbstractMeter Class Reference

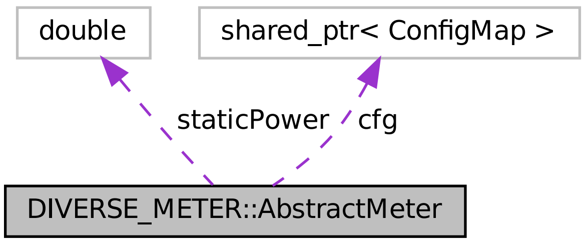
The abstract class for all meters.

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

Inheritance diagram for DIVERSE_METER::AbstractMeter:


DIVERSE_METER::AbstractMeter
DIVERSE_METER::EspMeterUart
DIVERSE_METER::IntelMeter

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 resluts

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

runningUs	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

cfg	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

_sp	

8.14.2.4 testStaticPower()

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

to test the static power of a system by sleeping

Parameters

sleepingSecond	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

T1	The type of the first index in the range. Should be a signed or unsigned integer.
T2	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.
T	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

*first_index_*	The first index in the range.
*index_after_↩ last_*	The index after the last index in the range.
*num_blocks_*	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	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	The block number.

Returns

The first index.

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

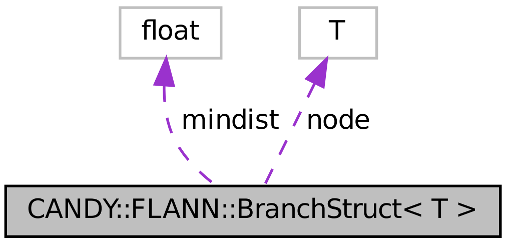
- include/Utils/BS_thread_pool.hpp

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:

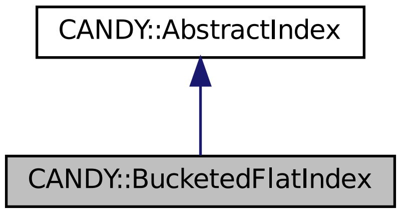
- include/CANDY/FlannIndex/FlannUtils.h

8.18 CANDY::BucketedFlatIndex Class Reference

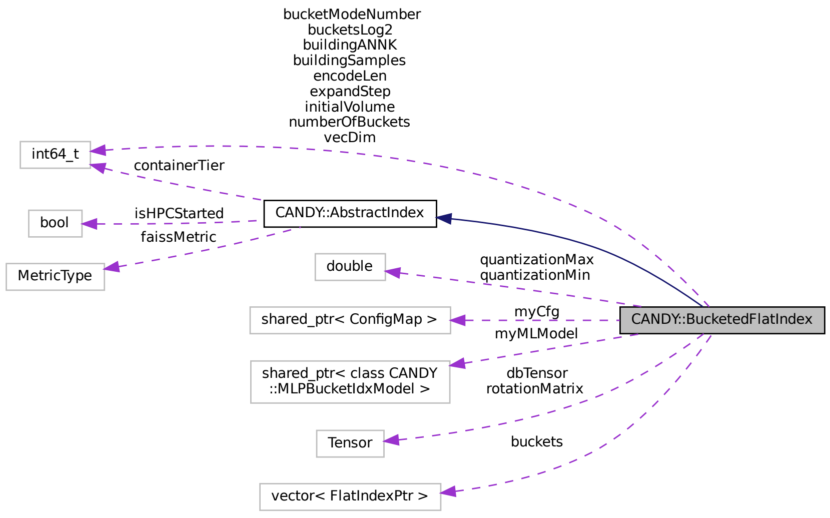
The class of splitting similar vectors into fixed number of buckets, each bucket is managed by FlatIndex.

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

Inheritance diagram for CANDY::BucketedFlatIndex:



Collaboration diagram for CANDY::BucketedFlatIndex:



Public Member Functions

- virtual void reset ()

  reset this index to inited 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, I64
- initialVolume, the initial volume of inline database tensor, default 1000, I64
- expandStep, the step of expanding inline database, default 100, I64
- numberOfBuckets, the number of titer buckets, default 1, I64, 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, I64

        * 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 maching learning to generate bucket indicies
    - encodeLen, the length of LSH encoding, in bytes, default 1, I64
    - metricType, the type of AKNN metric, default L2, String
    - 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

8.18.2 Member Function Documentation

8.18.2.1 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	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

t	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

t	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

t	the tensor to be revised, recommend single row
w	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

t	the tensor, allow only single rows
bkt	the bucket number which fits best
k	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

t	the tensor, allow multiple rows
k	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

cfg	the config of this class

Returns

bool whether the configuration is successful

@breif 1. common init

@breif 2.a init of mean mode

@breif 2.a init of lsh mode

Reimplemented from CANDY::AbstractIndex.

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

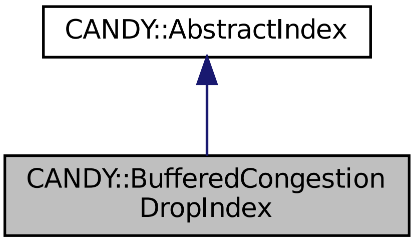
- include/CANDY/BucketedFlatIndex.h
- src/CANDY/BucketedFlatIndex.cpp

8.19 CANDY::BufferedCongestionDropIndex Class Reference

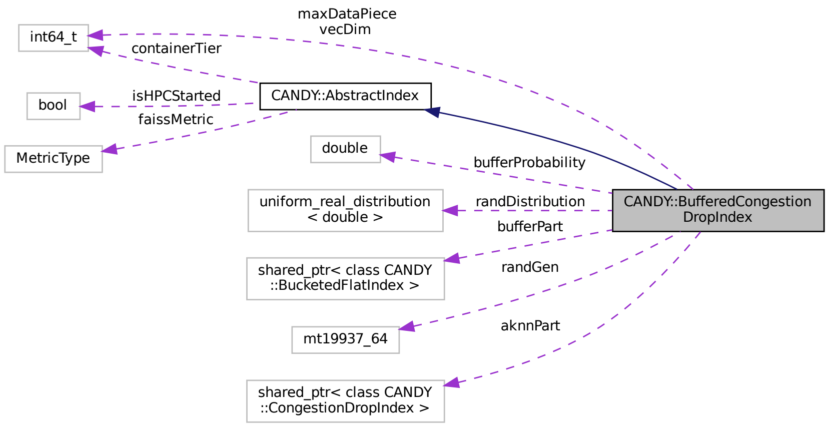
Similar to CongestionDropIndex, but will try to place some of the online data into an ingestion-efficient buffer, the buffer is implemented under BucketedFlatIndex More detailed description with an image:

```
#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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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:

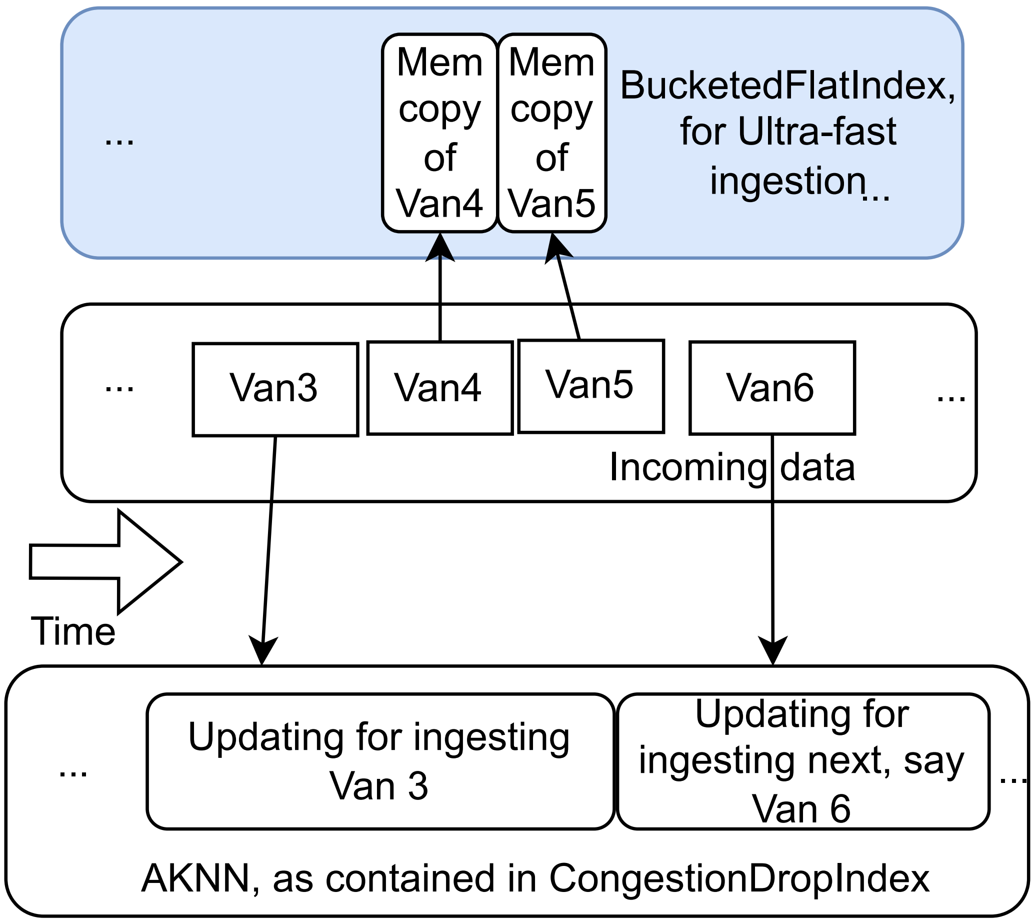

...
Mem copy of Van4
Mem copy of Van5
BucketedFlatIndex, for Ultra-fast ingestion...
...
Van3
Van4
Van5
Van6
...
Incoming data
Time
...
Updating for ingesting Van 3
Updating for ingesting next, say Van 6
...
AKNN, as contained in CongestionDropIndex

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, I64
- buffer_expandStep, the step of expanding inline database, default 100, I64
- buffer_numberOfBuckets, the number of titer buckets, default 1, I64, 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, I64
    - 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 @warnning 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

t	the tensor, some index needs to be single row
k	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

cfg	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

t	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

t	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

t	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

t	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

t	the tensor to be revised
w	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

t	the tensor, allow multiple rows
k	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-specfic config related to one index

Parameters

cfg	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

frozenLv	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.

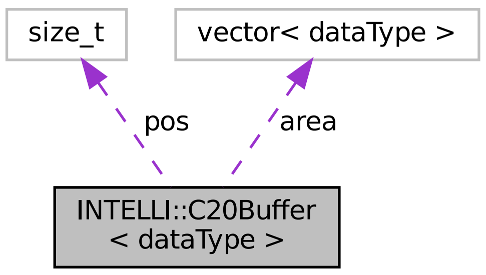
The documentation for this class was generated from the following files:

- 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

dataType	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

len	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

da	Data to be appended, a buffer
len	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

da	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 ponter 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 ponter of data, with offset.

Parameters

offset	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::Candy_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.

8.21.1 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

8.21.2 Member Function Documentation

8.21.2.1 dataLoader_create()

```
torch::Tensor CANDY::Candy_Python::dataLoader_create (
            string name,
            string type ) [inline]
```

The c++ bindings to creat an dataLoader at backend.

Parameters

name	the name of this dataLoader
type	the type of this dataLoader, keep the same as that in CANDY::IndexTable

Returns

tensor 1x1, [1] for success

8.21.2.2 dataLoader_editCfgDouble()

```
torch::Tensor CANDY::Candy_Python::dataLoader_editCfgDouble (
            string name,
            string key,
            double value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

name	the name of the dataLoader
key	the key in the cfg
value	the double value

Returns

tensor 1x1, [1] for success

8.21.2.3 dataLoader_editCfgFloat()

```
torch::Tensor CANDY::Candy_Python::dataLoader_editCfgFloat (
            string name,
            string key,
            float value )  [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

name	the name of the dataLoader
key	the key in the cfg
value	the float value

Returns

tensor 1x1, [1] for success

8.21.2.4 dataLoader_editCfgI64()

```
torch::Tensor CANDY::Candy_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

name	the name of the dataLoader
key	the key in the cfg
value	the I64 value

Returns

tensor 1x1, [1] for success

8.21.2.5 dataLoader_editCfgStr()

```
torch::Tensor CANDY::Candy_Python::dataLoader_editCfgStr (
            string name,
            string key,
            string value ) [inline]
```

The c++ bindings to change the config map related to a specific dataLoader.

Parameters

name	the name of the dataLoader
key	the key in the cfg
value	the string value

Returns

tensor 1x1, [1] for success

8.21.2.6 dataLoader_getData()

```
torch::Tensor CANDY::Candy_Python::dataLoader_getData (
            string name ) [inline]
```

The c++ bindings to get data tensor from the specified data loader.

Parameters

name	the name of the dataLoader
t	the tensor

Returns

tensor 1x1, [1] for success

8.21.2.7 dataLoader_getQuery()

```
torch::Tensor CANDY::Candy_Python::dataLoader_getQuery (
            string name ) [inline]
```

The c++ bindings to get query tensor from the specified data loader.

Parameters

name	the name of the dataLoader

Returns

the first result tensor

8.21.2.8 dataLoader_init()

```
torch::Tensor CANDY::Candy_Python::dataLoader_init (
             string name )  [inline]
```

The c++ bindings to init an dataLoader with its bounded config.

Parameters

name	the name of the dataLoader

Returns

tensor 1x1, [1] for success

8.21.2.9 index_create()

```
torch::Tensor CANDY::Candy_Python::index_create (
             string name,
             string type )  [inline]
```

The c++ bindings to creat an index at backend.

Parameters

name	the name of this index
type	the type of this index, keep the same as that in CANDY::IndexTable

Returns

tensor 1x1, [1] for success

8.21.2.10 index_delete()

```
torch::Tensor CANDY::Candy_Python::index_delete (
             string name,
             torch::Tensor t,
             int64_t k )  [inline]
```

The c++ bindings to delete tensor to an index.

Parameters

name	the name of the index
t	the tensor
k	the NNS

Returns

tensor 1x1, [1] for success

8.21.2.11 index_deleteString()

```
torch::Tensor CANDY::Candy_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

name	the name of the index
t	the tensor
k	the NNS

Returns

tensor 1x1, [1] for success

8.21.2.12 index_editCfgDouble()

```
torch::Tensor CANDY::Candy_Python::index_editCfgDouble (
             string name,
             string key,
             double value )  [inline]
```

The c++ bindings to change the config map related to a specific index.

Parameters

name	the name of the index
key	the key in the cfg
value	the double value

Returns

tensor 1x1, [1] for success

8.21.2.13 index_editCfgI64()

```
torch::Tensor CANDY::Candy_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

name	the name of the index
key	the key in the cfg
value	the I64 value

Returns

tensor 1x1, [1] for success

8.21.2.14 index_editCfgStr()

```
torch::Tensor CANDY::Candy_Python::index_editCfgStr (
            string name,
            string key,
            string value )  [inline]
```

The c++ bindings to change the config map related to a specific index.

Parameters

name	the name of the index
key	the key in the cfg
value	the string value

Returns

tensor 1x1, [1] for success

8.21.2.15 index_endHPC()

```
torch::Tensor CANDY::Candy_Python::index_endHPC (
            string name ) [inline]
```

The c++ bindings to end HPC features.

Parameters

name	the name of the index

Returns

tensor 1x1, [1] for success

8.21.2.16 index_init()

```
torch::Tensor CANDY::Candy_Python::index_init (
            string name ) [inline]
```

The c++ bindings to init an index with its bounded config.

Parameters

name	the name of the index

Returns

tensor 1x1, [1] for success

8.21.2.17 index_insert()

```
torch::Tensor CANDY::Candy_Python::index_insert (
            string name,
            torch::Tensor t ) [inline]
```

The c++ bindings to insert tensor to an index.

Parameters

name	the name of the index
t	the tensor

Returns

tensor 1x1, [1] for success

8.21.2.18 index_insertString()

```
torch::Tensor CANDY::Candy_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

name	the name of the index
t	the tensor
s	the vector of string, List[str] in python

Returns

tensor 1x1, [1] for success

8.21.2.19 index_loadCfgFromFile()

```
torch::Tensor CANDY::Candy_Python::index_loadCfgFromFile (
            string name,
            string fname )  [inline]
```

The c++ bindings to load the config map related to a specific index from file.

Parameters

name	the name of the index
fname	the name of file

Returns

tensor 1x1, [1] for success

8.21.2.20 index_loadInitial()

```
torch::Tensor CANDY::Candy_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

name	the name of the index
t	the tensor

Returns

tensor 1x1, [1] for success

8.21.2.21 index_loadInitialString()

```
torch::Tensor CANDY::Candy_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

name	the name of the index
t	the tensor

Returns

tensor 1x1, [1] for success

8.21.2.22 index_offlineBuild()

```
torch::Tensor CANDY::Candy_Python::index_offlineBuild (
            string name,
            torch::Tensor t )  [inline]
```

The c++ bindings to offlineBuild.

Parameters

name	the name of the index
t	the tensor

Returns

tensor 1x1, [1] for success

8.21.2.23 index_rawData()

```
torch::Tensor CANDY::Candy_Python::index_rawData (
             string name )  [inline]
```

The c++ bindings to return rawData.

Parameters

name	the name of the index

Returns

tensor of rawData

8.21.2.24 index_reset()

```
torch::Tensor CANDY::Candy_Python::index_reset (
             string name )  [inline]
```

The c++ bindings to creat an index at backend.

Parameters

name	the name of the index

Returns

tensor 1x1, [1] for success

8.21.2.25 index_revise()

```
torch::Tensor CANDY::Candy_Python::index_revise (
             string name,
             torch::Tensor t,
             torch::Tensor & w )  [inline]
```

The c++ bindings to revise tensor to an index.

Parameters

name	the name of the index
t	the tensor to be revised
w	the revison

Returns

tensor 1x1, [1] for success

8.21.2.26 index_search()

```
std::vector<torch::Tensor> CANDY::Candy_Python::index_search (
             string name,
             torch::Tensor t,
             int64_t k )  [inline]
```

The c++ bindings to search tensor.

Parameters

name	the name of the index
t	the tensor
k	the NNS

Returns

the list of result tensors

8.21.2.27 index_searchString()

```
std::vector<std::vector<std::string> > CANDY::Candy_Python::index_searchString (
             string name,
             torch::Tensor & q,
             int64_t k )  [inline]
```

The c++ bindings to search binded string of given tensor.

Parameters

name	the name of the index
t	the tensor
k	the NNS

Returns

List[List[str]], for each rows

8.21.2.28 index_searchTensorAndStringList()

```
std::tuple<std::vector<torch::Tensor>, std::vector<std::vector<std::string> > > CANDY::↩
Candy_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

name	the name of the index
t	the tensor
k	the NNS

Returns

[List[Tensor],List[List[str]]], for each rows

8.21.2.29 index_setFrozenLevel()

```
torch::Tensor CANDY::Candy_Python::index_setFrozenLevel (
            string name,
            int64_t frozenLV )  [inline]
```

The c++ bindings to set the frozen level of online updating internal state.

Parameters

name	the name of the index
frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

tensor 1x1, [1] for success

8.21.2.30 index_startHPC()

```
torch::Tensor CANDY::Candy_Python::index_startHPC (
            string name )  [inline]
```

The c++ bindings to start HPC features.

Parameters

name	the name of the index

Returns

tensor 1x1, [1] for success

8.21.2.31 index_waitPending()

```
torch::Tensor CANDY::Candy_Python::index_waitPending (
            string name )  [inline]
```

The c++ bindings to wait pending operations features.

Parameters

name	the name of the index

Returns

tensor 1x1, [1] for success

8.21.2.32 tensorFromFile()

```
torch::Tensor CANDY::Candy_Python::tensorFromFile (
            std::string fname )  [inline]
```

The c++ bindings to load a tensor from file.

Parameters

name	the name of the index

Returns

the tensor result

8.21.2.33 tensorFromFVECS()

```
torch::Tensor CANDY::Candy_Python::tensorFromFVECS (
             string name )  [inline]
```

The c++ bindings to load tensor from fvecs file.

Parameters

name	the name of file return the result tensor

8.21.2.34 tensorFromHDF5()

```
torch::Tensor CANDY::Candy_Python::tensorFromHDF5 (
             string name,
             string attr )  [inline]
```

The c++ bindings to load tensor from HDF5 file.

Parameters

name	the name of file
attr	the attribute return the result tensor

8.21.2.35 tensorToFile()

```
torch::Tensor CANDY::Candy_Python::tensorToFile (
             torch::Tensor A,
             std::string fname )  [inline]
```

The c++ bindings to save a tensor into file.

Parameters

A	the tensor
name	the name of the index

Returns

tensor 1x1, [1] for success

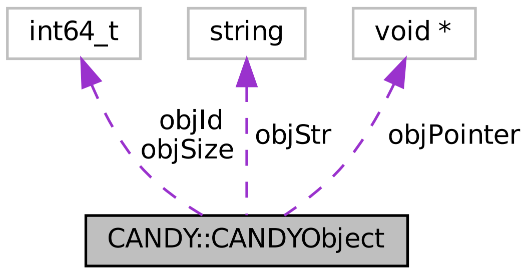
The documentation for this class was generated from the following file:

- include/CANDYPYTHON.h

8.22 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

8.22.1 Detailed Description

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

Todo to finish the functions of setting void ∗ pointers

8.22.2 Member Function Documentation

8.22.2.1 getStr()

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

to get the string

Returns

the objStr

8.22.2.2 setStr()

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

to set the string

Parameters

str	the string

Returns

void

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

- include/CANDY/CANDYObject.h
- src/CANDY/CANDYObject.cpp

8.23 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

8.24 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

8.25 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

8.26 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

8.27 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

8.28 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

8.29 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

8.30 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

8.31 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

8.32 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

8.33 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

8.34 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

8.35 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

8.36 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

8.37 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

8.38 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

8.39 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

8.40 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

8.41 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

8.42 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

8.43 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

8.44 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

8.45 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

8.46 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

8.47 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

8.48 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

8.49 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

8.50 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

8.51 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

8.52 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

8.53 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

8.54 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

8.55 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

8.56 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

8.57 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

8.58 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

8.59 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

8.60 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

8.61 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

8.62 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

8.63 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

8.64 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

8.65 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

8.66 cl_ushort8 Union Reference

Public Member Functions

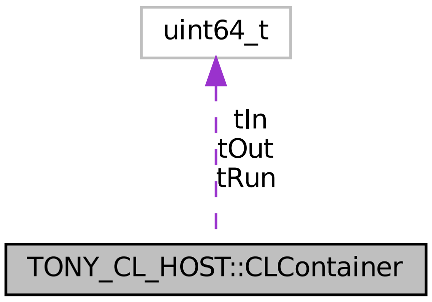
- cl_ushort **CL_ALIGNED** (16) s[8]

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

- include/CL/cl_platform.h

8.67 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:

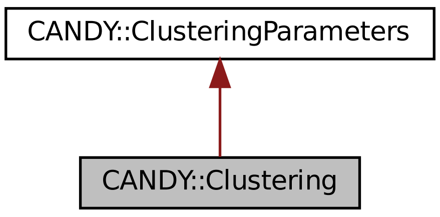
- include/CL/CLContainer.hpp
- src/CLContainer.cpp

8.68 CANDY::Clustering Class Reference

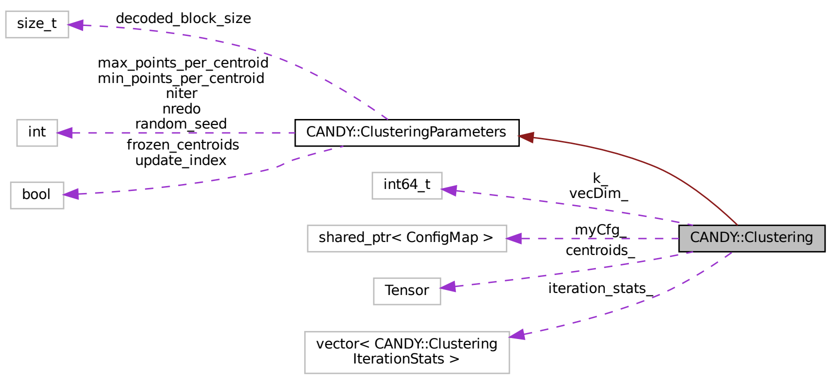
class for naive K-means clustering

```
#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)

8.68.1 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

8.68.2 Member Function Documentation

8.68.2.1 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

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

8.68.2.2 imbalance_factor()

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

compute the imbalance factor of an assignment

Parameters

n	number of input vectors
k	number of centroids
assign	assignment of centroid clustering

Returns

imbalance factor of the assignment

8.68.2.3 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

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

Returns

8.68.2.4 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

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

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

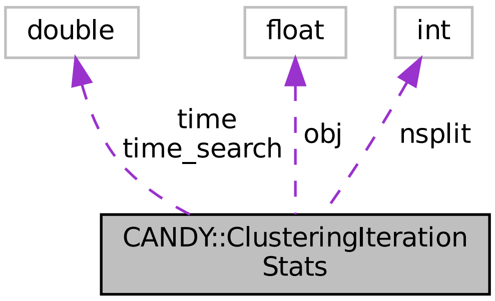
- include/CANDY/PQIndex/Clustering.h
- src/CANDY/PQIndex/Clustering.cpp

8.69 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**

8.69.1 Detailed Description

struct to record performance of clustering during iterations

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

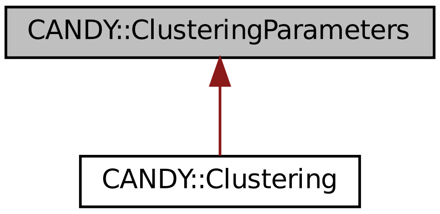
- include/CANDY/PQIndex/Clustering.h

8.70 CANDY::ClusteringParameters Class Reference

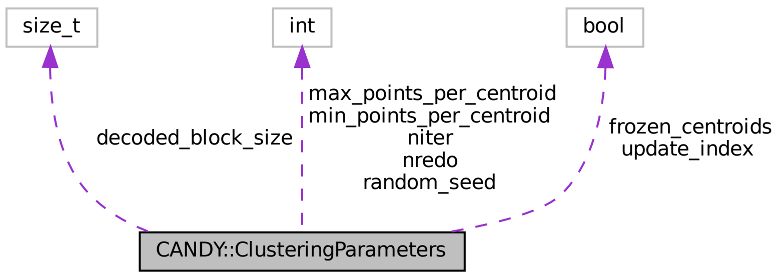
Class for the clustering parameters to be set before training/building.

`#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

8.70.1 Detailed Description

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

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

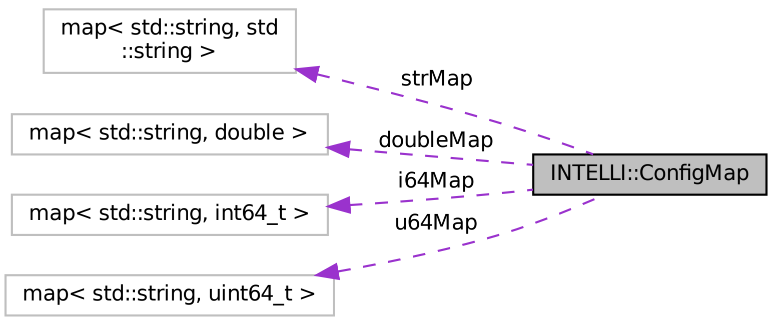
- include/CANDY/PQIndex/Clustering.h

8.71 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:


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

8.71.1 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:

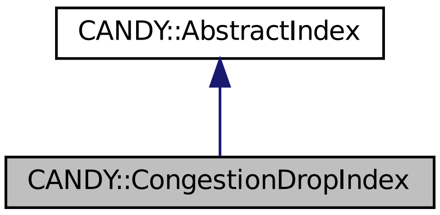
- include/Utils/ConfigMap.hpp

8.72 CANDY::CongestionDropIndex Class Reference

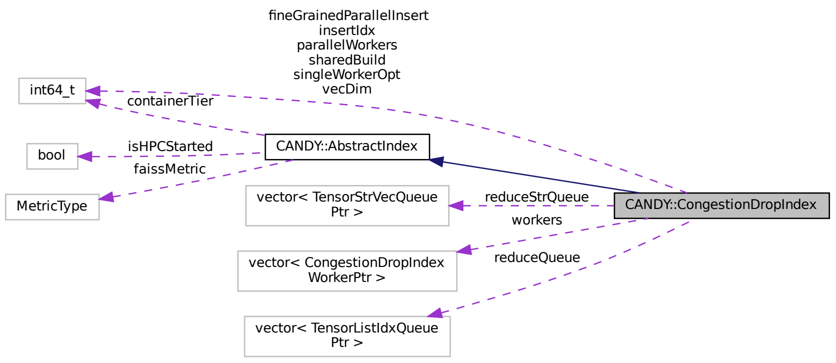
A container index to evaluate other bottom index, will just drop the data if congestion occurs, also support the data sharding parallelism.

```
#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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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**

8.72.1 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, I64, default 10
- parallelWorks The number of paraller workers, I64, default 1 (set this to less than 0 will use max hardware_concurrency);
- vecDim, the dimension of vectors, default 768, I64
- 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 @warnning Make sure you are using 2D tensors!

8.72.2 Member Function Documentation

8.72.2.1 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

t	the tensor, some index need to be single row
k	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.

8.72.2.2 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, some index needs to be single row
k	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.

8.72.2.3 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.

8.72.2.4 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.72.2.5 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

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

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.6 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.7 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

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

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.8 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.9 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.10 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.72.2.11 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised
w	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.

8.72.2.12 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.72.2.13 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

t	the tensor, allow multiple rows
k	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.

8.72.2.14 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

t	the tensor, allow multiple rows
k	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.

8.72.2.15 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.16 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

Reimplemented from CANDY::AbstractIndex.

8.72.2.17 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.

8.72.2.18 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.

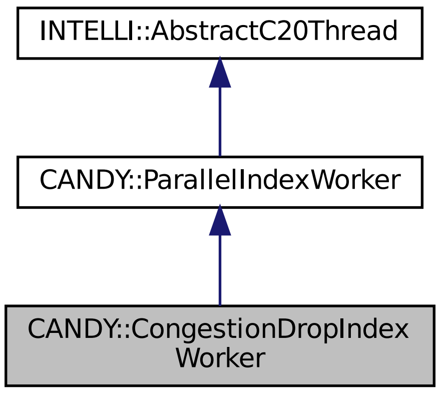
The documentation for this class was generated from the following files:

- include/CANDY/CongestionDropIndex.h
- src/CANDY/CongestionDropIndex.cpp

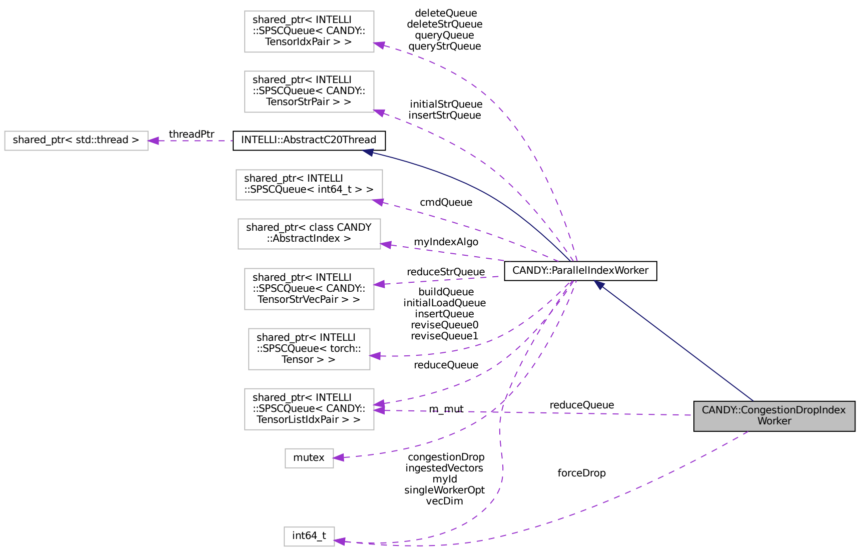
8.73 CANDY::CongestionDropIndexWorker Class Reference

A worker class to container bottom indexings, will just drop new element if congestion occurs.

Inheritance diagram for CANDY::CongestionDropIndexWorker:


INTELLI::AbstractC20Thread
CANDY::ParallelIndexWorker
CANDY::CongestionDropIndex Worker

Collaboration diagram for CANDY::CongestionDropIndexWorker:


shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorIdxPair > >
deleteQueue deleteStrQueue queryQueue queryStrQueue
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorStrPair > >
initialStrQueue insertStrQueue
shared_ptr< std::thread >
threadPtr
INTELLI::AbstractC20Thread
shared_ptr< INTELLI ::SPSCQueue< int64_t > >
cmdQueue
shared_ptr< class CANDY ::AbstractIndex >
myIndexAlgo
CANDY::ParallelIndexWorker
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorStrVecPair > >
reduceStrQueue
buildQueue initialLoadQueue insertQueue reviseQueue0 reviseQueue1
shared_ptr< INTELLI ::SPSCQueue< torch:: Tensor > >
reduceQueue
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorListIdxPair > >
m_mut
reduceQueue
CANDY::CongestionDropIndex Worker
mutex
congestionDrop ingestedVectors myId singleWorkerOpt vecDim
forceDrop
int64_t

Public Member Functions

- virtual bool insertTensor (torch::Tensor &t)

  insert a tensor
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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

8.73.1 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:

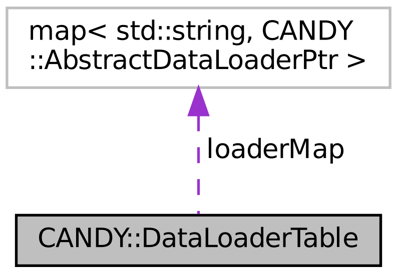
- include/CANDY/CongestionDropIndex/CongestionDropIndexWorker.h
- src/CANDY/CongestionDropIndex/CongestionDropIndexWorker.cpp

8.74 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**

8.74.1 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

8.74.2 Constructor & Destructor Documentation

8.74.2.1 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

8.74.3 Member Function Documentation

8.74.3.1 findDataLoader()

```
CANDY::AbstractDataLoaderPtr CANDY::DataLoaderTable::findDataLoader (
            std::string name )  [inline]
```

find a dataloader in the table according to its name

Parameters

name	The nameTag of loader

Returns

The DataLoader, nullptr if not found

8.74.3.2 registerNewDataLoader()

```
void CANDY::DataLoaderTable::registerNewDataLoader (
            CANDY::AbstractDataLoaderPtr dnew,
            std::string tag )  [inline]
```

To register a new loader.

Parameters

onew	The new operator
tag	THe name tag

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

- include/DataLoader/DataLoaderTable.h
- src/DataLoader/DataLoaderTable.cpp

8.75 default_attrs Struct Reference

The low-level perf descriptions passed to OS.

```
#include <ThreadPerf.hpp>
```

8.75.1 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:

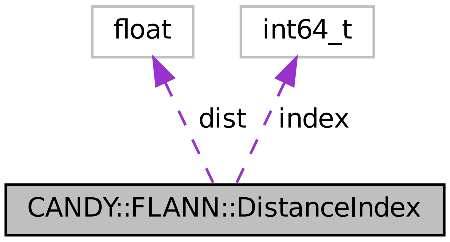
- include/Utils/ThreadPerf.hpp

8.76 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**

8.76.1 Detailed Description

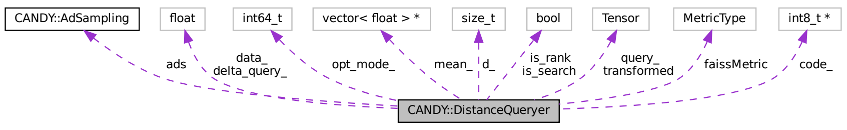
The structure representing a vectors' distance with the query along with its index.

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

- include/CANDY/FlannIndex/FlannUtils.h

8.77 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**

8.77.1 Member Function Documentation

8.77.1.1 operator()()

```
float CANDY::DistanceQueryer::operator() (
            INTELLI::TensorPtr idx )  [inline]
```

compute the distance between given idx's vector and query vector

Parameters

idx	the target vector to be computed with query vector

Returns

L2 Distance

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

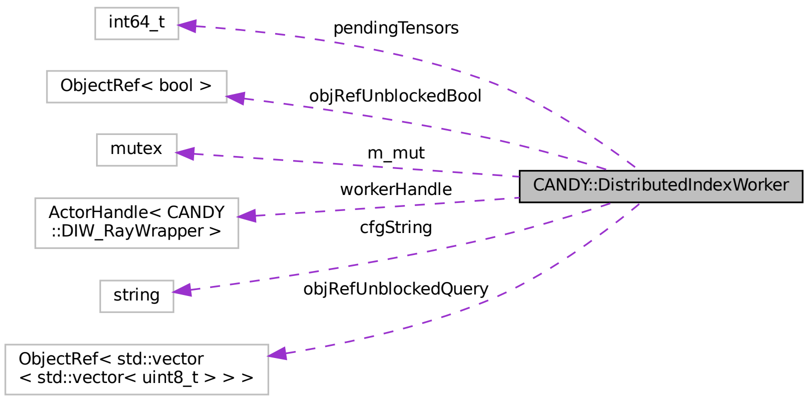
- include/CANDY/HNSWNaive/DistanceQueryer.h

8.78 CANDY::DistributedIndexWorker Class Reference

A worker class of parallel index thread.

```
#include <CANDY/DistributedPartitionIndex/DistributedIndexWorker.h>
```

Collaboration diagram for CANDY::DistributedIndexWorker:


int64_t
pendingTensors
ObjectRef< bool >
objRefUnblockedBool
mutex
m_mut
CANDY::DistributedIndexWorker
workerHandle
ActorHandle< CANDY ::DIW_RayWrapper >
cfgString
string
objRefUnblockedQuery
ObjectRef< std::vector < std::vector< uint8_t > > >

Public Member Functions

- virtual void reset ()

  reset this index to inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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

8.78.1 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, I64, default 10

8.78.2 Member Function Documentation

8.78.2.1 deleteTensor()

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

delete a tensor

Parameters

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

Returns

bool whether the deleting is successful

8.78.2.2 endHPC()

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

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

8.78.2.3 getUnblockQueryResult()

```
std::vector< torch::Tensor > CANDY::DistributedIndexWorker::getUnblockQueryResult (
            void )  [virtual]
```

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

Parameters

q	the tensor, packed in std::vector<uint8_t> allow multiple rows
k	the returned neighbors

Returns

std::vector<std::vector<uint8_t>> the packed result tensor for each row of query

8.78.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

8.78.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

8.78.2.6 loadInitialTensorUnblocked()

```
void CANDY::DistributedIndexWorker::loadInitialTensorUnblocked (
            torch::Tensor & t )  [virtual]
```

load initial tensor in unblocked model

Parameters

t	the tensor for offline build

8.78.2.7 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

8.78.2.8 offlineBuildUnblocked()

```
void CANDY::DistributedIndexWorker::offlineBuildUnblocked (
            torch::Tensor & t )  [virtual]
```

offline build phase in unblocked model

Parameters

t	the tensor for offline build

8.78.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

8.78.2.10 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

q	the tensor, packed in std::vector<uint8_t> allow multiple rows
k	the returned neighbors

Returns

std::vector<std::vector<uint8_t>> the packed result tensor for each row of query

8.78.2.11 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

8.78.2.12 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

8.78.2.13 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

8.78.2.14 waitPendingOperations()

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

a busy waitting 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:

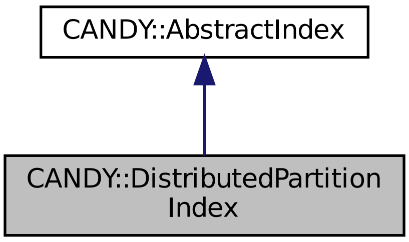
- include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h
- src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp

8.79 CANDY::DistributedPartitionIndex Class Reference

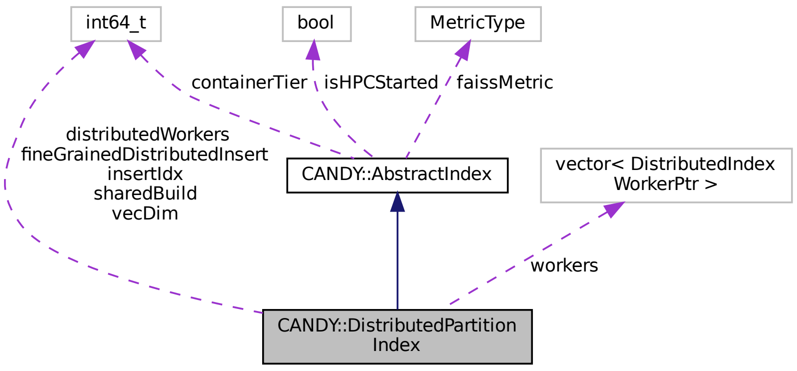
A basic distributed index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query.

```
#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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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

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

8.79.1 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, I64, default 10
- distributedWorkers The number of paraller workers, I64, default 1;
- vecDim, the dimension of vectors, default 768, I64
- fineGrainedDistributedInsert, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, I64, default 0
- sharedBuild whether let all sharding using shared build, 1, I64

Warning

Make sure you are using 2D tensors! Not works well with python API

8.79.2 Member Function Documentation

8.79.2.1 deleteTensor()

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

delete a tensor

Parameters

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

Returns

bool whether the deleting is successful

1. map
2. reduce

Reimplemented from CANDY::AbstractIndex.

8.79.2.2 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.

8.79.2.3 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.79.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.79.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.79.2.6 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

Reimplemented from CANDY::AbstractIndex.

8.79.2.7 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.79.2.8 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised
w	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.

8.79.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

1. map
2. reduce

Reimplemented from CANDY::AbstractIndex.

8.79.2.10 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.79.2.11 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

Reimplemented from CANDY::AbstractIndex.

8.79.2.12 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.

8.79.2.13 waitPendingOperations()

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

a busy waitting 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;

Reimplemented from CANDY::AbstractIndex.

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

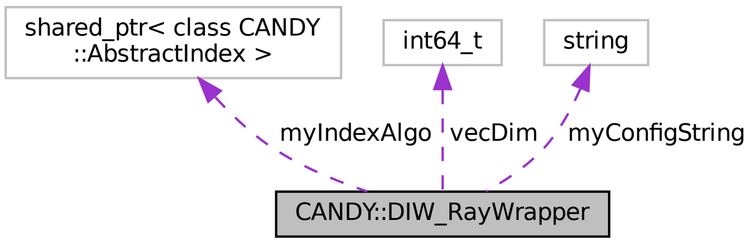
- include/CANDY/DistributedPartitionIndex.h
- src/CANDY/DistributedPartitionIndex.cpp

8.80 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 waitting 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

8.80.1 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, I674, default 768

8.80.2 Member Function Documentation

8.80.2.1 deleteTensor()

```
bool CANDY::DIW_RayWrapper::deleteTensor (
            std::vector< uint8_t > t,
            int64_t k = 1 )  [virtual]
```

delete a tensor

Parameters

t	the tensor, packed in std::vector<uint8_t>
k	the number packed in std::vector<uint8_t>

Returns

bool whether the deleting is successful

8.80.2.2 endHPC()

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

some extra termination if the index has HPC fetures

Returns

bool whether the HPC termination is successful

8.80.2.3 insertTensor()

```
bool CANDY::DIW_RayWrapper::insertTensor (
            std::vector< uint8_t > t )  [virtual]
```

insert a tensor

Parameters

t	the tensor packed in std::vector<uint8_t>

Returns

bool whether the insertion is successful

8.80.2.4 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

t	the tensor for offline build

Returns

whether the building is successful

8.80.2.5 offlineBuild()

```
bool CANDY::DIW_RayWrapper::offlineBuild (
            std::vector< uint8_t > t )  [virtual]
```

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

8.80.2.6 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

q	the tensor, packed in std::vector<uint8_t> allow multiple rows
k	the returned neighbors

Returns

std::vector<std::vector<uint8_t>> the packed result tensor for each row of query

8.80.2.7 setConfig()

```
bool CANDY::DIW_RayWrapper::setConfig (
            std::string cfs )
```

set the config by using raw string

Parameters

cfs	the raw string

Returns

bool

1. find the index algo

8.80.2.8 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

8.80.2.9 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

8.80.2.10 waitPendingOperations()

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

a busy waitting 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:

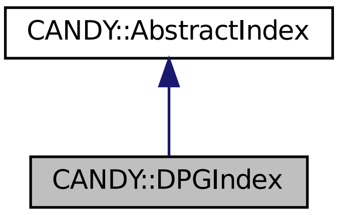
- include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h
- src/CANDY/DistributedPartitionIndex/DistributedIndexWorker.cpp

8.81 CANDY::DPGIndex Class Reference

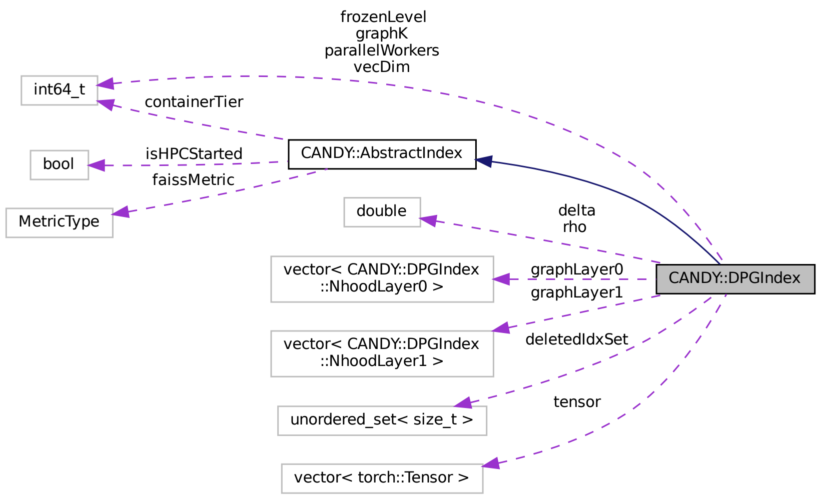
A hierarchical algorithm based on a data structure consistent with NNDescentIndex, the subgraph in the hierarchical graph will retain half of the most directional diversity of edges in the original graph, and expand the unidirectional edges into bidirectional edges. The offline construction of the basic 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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

    set the index-specfic 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

8.81.1 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, I64, default 1 (set this to less than 0 will use max hardware_concurrency);
- vecDim, the dimension of vectors, default 768, I64
- graphK, the neighbors of every node in internal data struct, default 20, I64
- 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 @warnning Make sure you are using 2D tensors!

8.81.2 Member Function Documentation

8.81.2.1 deleteTensor()

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

delete a tensor

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 from CANDY::AbstractIndex.

8.81.2.2 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.

8.81.2.3 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.81.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.81.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.81.2.6 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

Reimplemented from CANDY::AbstractIndex.

8.81.2.7 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.81.2.8 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised
w	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

Reimplemented from CANDY::AbstractIndex.

8.81.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.81.2.10 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.81.2.11 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

Reimplemented from CANDY::AbstractIndex.

8.81.2.12 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.

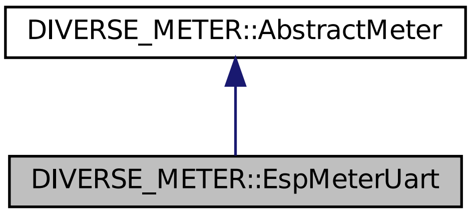
The documentation for this class was generated from the following files:

- include/CANDY/DPGIndex.h
- src/CANDY/DPGIndex.cpp

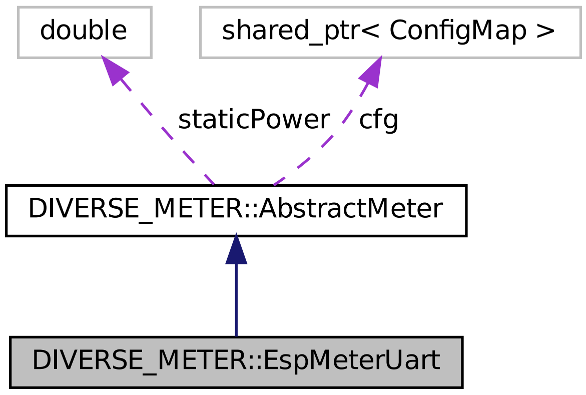
8.82 DIVERSE_METER::EspMeterUart Class Reference

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

8.82.1 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 resluts

config parameters:

- meterAddress, String, The file system path of meter, default "/dev/ttyUSB0";

tag is "espUart"

8.82.2 Member Function Documentation

8.82.2.1 setConfig()

```
void EspMeterUart::setConfig (
            INTELLI::ConfigMapPtr _cfg )  [virtual]
```

to set the configmap

Parameters

cfg	the config map

Reimplemented from DIVERSE_METER::AbstractMeter.

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

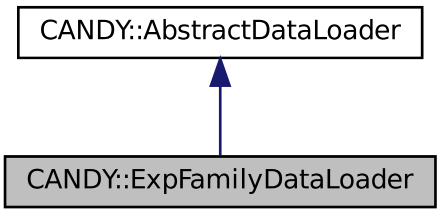
- include/Utils/Meters/EspMeterUart/EspMeterUart.hpp
- src/Utils/Meters/EspMeterUart/EspMeterUart.cpp

8.83 CANDY::ExpFamilyDataLoader Class Reference

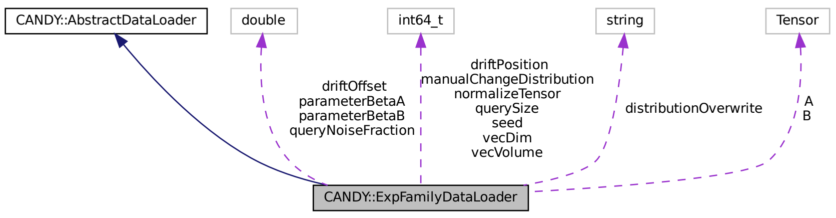
The class to load data from exponential family, i.e., poisson, gaussian, exponential and beta.

```
#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**

8.83.1 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

8.83.2 Member Function Documentation

8.83.2.1 getData()

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

get the data tensor

Returns

the generated data tensor

Reimplemented from CANDY::AbstractDataLoader.

8.83.2.2 getQuery()

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

get the query tensor

Returns

the generated query tensor

Reimplemented from CANDY::AbstractDataLoader.

8.83.2.3 hijackConfig()

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

To hijack some configurations inline.

Parameters

cfg	The config map

Returns

bool whether the config is successfully set

Note

Reimplemented from CANDY::AbstractDataLoader.

8.83.2.4 setConfig()

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

Set the GLOBAL config map related to this loader.

Parameters

cfg	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:

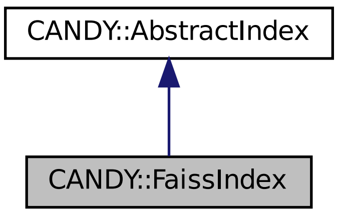
- include/DataLoader/ExpFamilyDataLoader.h
- src/DataLoader/ExpFamilyDataLoader.cpp

8.84 CANDY::FaissIndex Class Reference

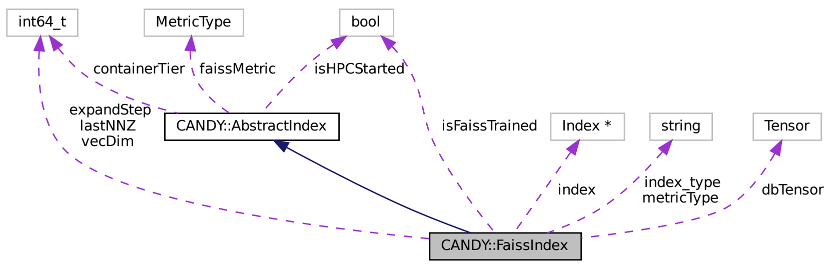
The class of converting faiss index api into rania index style.

`#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

8.84.1 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, I64
- 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, I64, default 32, the max number of neighbor connections in hnsw
  - PQ, using faiss::IndexPQ, additional config as follows
    - encodeLen, the encoding length in bytes, I64, default 1
    - encodeLenBits, the encoding length in bits, I64, default encodeLen∗8 (will overwrite encodeLen if manually set)
    - subQuantizers, the number of subquantizers used, I64, default 8
  - IVFPQ, using faiss::IndexIVFPQ, additional config as follows
    - encodeLen, the encoding length in bytes, I64, default 1
    - encodeLenBits, the encoding length in bits, I64, default encodeLen∗8 (will overwrite encodeLen if manually set)
    - subQuantizers, the number of subquantizers used, I64, default 8
    - lists, the number of lists used, I64, default 1000
  - LSH, using faiss::IndexLSH, additional config as follows
    - encodeLen, the encoding length in bytes, I64, default 1
    - encodeLenBits, the encoding length in bits, I64, 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

8.84.2 Member Function Documentation

8.84.2.1 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.84.2.2 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.84.2.3 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.84.2.4 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.84.2.5 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.84.2.6 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

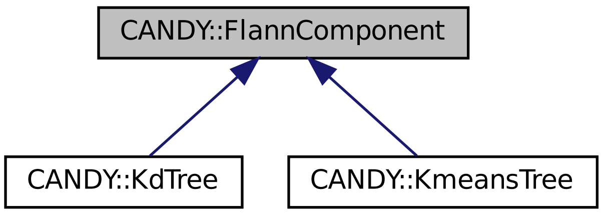
Reimplemented from CANDY::AbstractIndex.

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

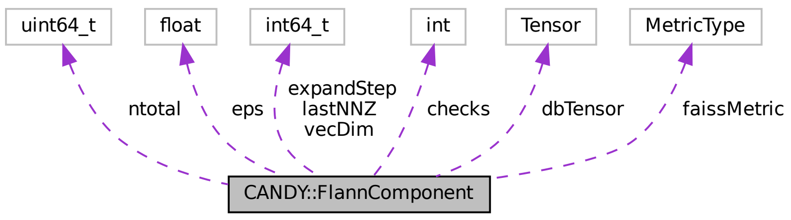
- include/CANDY/FaissIndex.h
- src/CANDY/FaissIndex.cpp

8.85 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

8.85.1 Member Function Documentation

8.85.1.1 setParams()

```
virtual bool CANDY::FlannComponent::setParams (
            FlannParam param )  [inline], [virtual]
```

set the params from auto-tuning

Parameters

param	best param

Returns

true if success

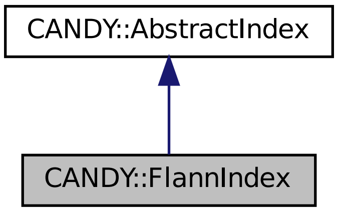
Reimplemented in CANDY::KmeansTree, and CANDY::KdTree.

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

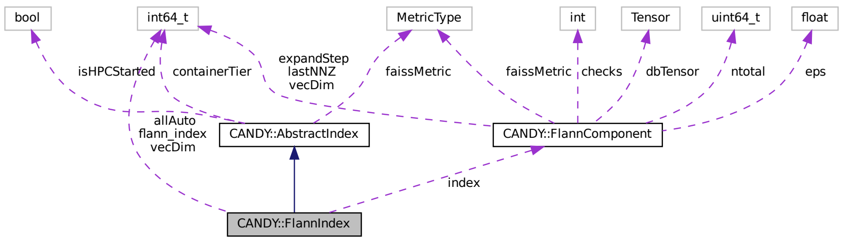
- include/CANDY/FlannIndex/FlannComponent.h

8.86 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

8.86.1 Member Function Documentation

8.86.1.1 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.86.1.2 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.86.1.3 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.86.1.4 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.86.1.5 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.86.1.6 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

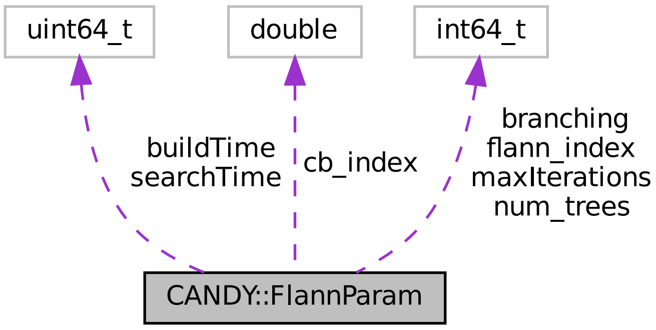
Reimplemented from CANDY::AbstractIndex.

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

- include/CANDY/FlannIndex.h
- src/CANDY/FlannIndex.cpp

8.87 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:

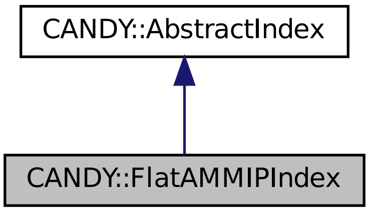
- include/CANDY/FlannIndex/FlannComponent.h

8.88 CANDY::FlatAMMIPIndex Class Reference

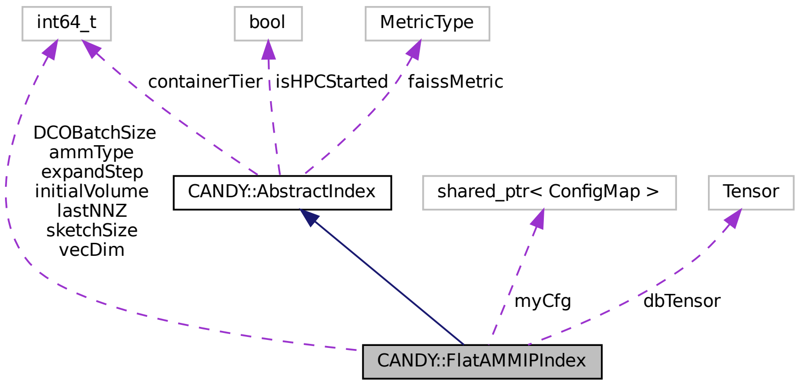
The class of a flat index approach, using brutal force management for data, but approximate matrix multiplication to compute distance.

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

Inheritance diagram for CANDY::FlatAMMIPIndex:



Collaboration diagram for CANDY::FlatAMMIPIndex:



Public Member Functions

- virtual void reset ()

  reset this index to inited 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

8.88.1 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, I64
- initialVolume, the initial volume of inline database tensor, default 1000, I64
- expandStep, the step of expanding inline database, default 100, I64
- sketchSize, the sketch size of amm, default 10, I64
- DCOBatchSize, the batch size of internal distance comparison operation (DCO), default -1 (full data once), I64
- 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 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.88.2.2 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.88.2.3 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.88.2.4 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.88.2.5 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.88.2.6 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.88.2.7 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.88.2.8 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.88.2.9 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:

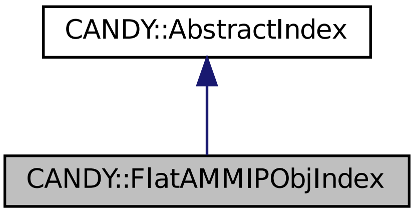
- include/CANDY/FlatAMMIPIndex.h
- src/CANDY/FlatAMMIPIndex.cpp

8.89 CANDY::FlatAMMIPObjIndex Class Reference

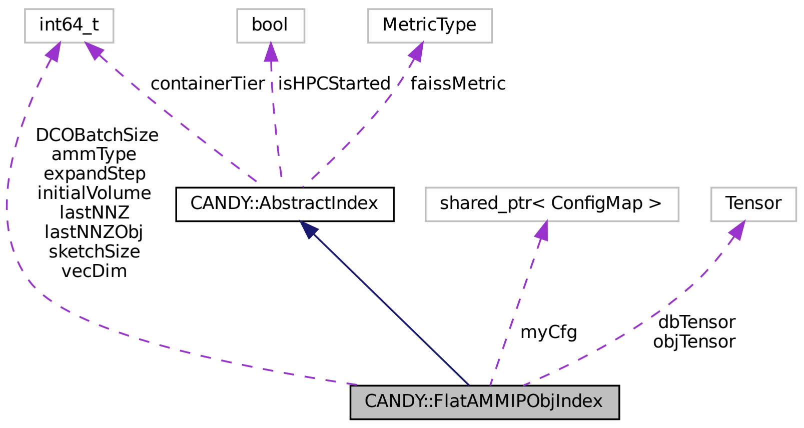
Similar to FlatAMMIPIndex, but additionally has object storage (currently only string)

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

Inheritance diagram for CANDY::FlatAMMIPObjIndex:



Collaboration diagram for CANDY::FlatAMMIPObjIndex:



Public Member Functions

- virtual void reset ()

  reset this index to inited 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

8.89.1 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, I64
- initialVolume, the initial volume of inline database tensor, default 1000, I64
- expandStep, the step of expanding inline database, default 100, I64
- sketchSize, the sketch size of amm, default 10, I64
- DCOBatchSize, the batch size of internal distance comparison operation (DCO), default -1 (full data once), I64
- 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

8.89.2 Member Function Documentation

8.89.2.1 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

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

Returns

bool whether the delet is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.2 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.3 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.89.2.4 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

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

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.5 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.6 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.89.2.7 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.8 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.89.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.89.2.10 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.89.2.11 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.89.2.12 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:

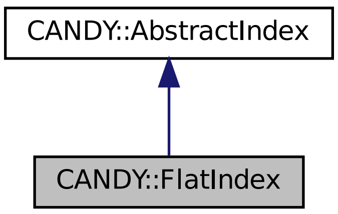
- include/CANDY/FlatAMMIPObjIndex.h
- src/CANDY/FlatAMMIPObjIndex.cpp

8.90 CANDY::FlatIndex Class Reference

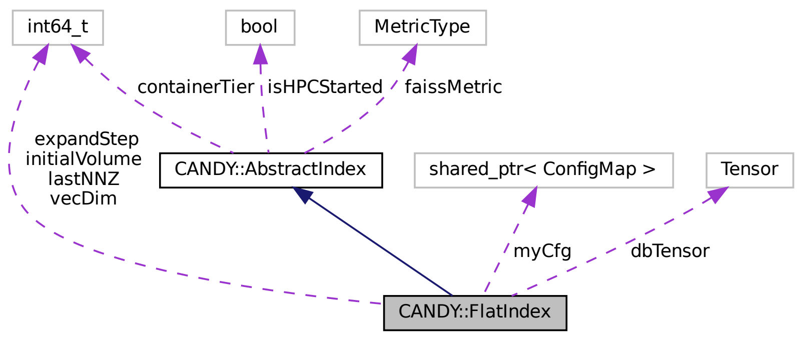
The class of a flat index approach, using brutal force management.

`#include <CANDY/FlatIndex.h>`

Inheritance diagram for CANDY::FlatIndex:



Collaboration diagram for CANDY::FlatIndex:



Public Member Functions

- virtual void reset ()

    reset this index to inited 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

8.90.1 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, I64
- initialVolume, the initial volume of inline database tensor, default 1000, I64
- expandStep, the step of expanding inline database, default 100, I64

8.90.2 Member Function Documentation

8.90.2.1 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.90.2.2 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.90.2.3 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.90.2.4 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.90.2.5 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.90.2.6 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.90.2.7 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.90.2.8 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.90.2.9 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:

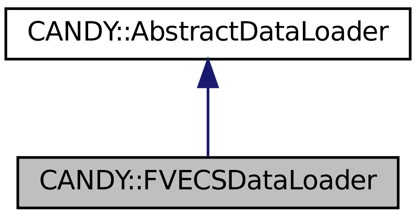
- include/CANDY/FlatIndex.h
- src/CANDY/FlatIndex.cpp

8.91 CANDY::FVECSDataLoader Class Reference

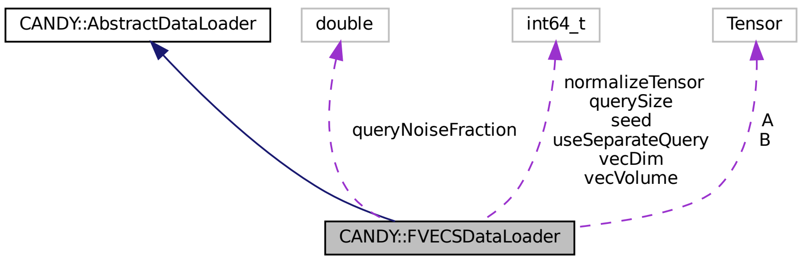
The class for loading ∗.fvecs data.

```
#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**

8.91.1 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, I64
- vecVolume, the volume of vectors, default 10000, I64
- 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), I64
- useSeparateQuery, whether or not load query separately, 1, I64
- 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, I64
- seed, the random seed, default 7758258, I64

: default name tags

- "fvecs": FVECSDataLoader

8.91.2 Member Function Documentation

8.91.2.1 getData()

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

get the data tensor

Returns

the generated data tensor

Reimplemented from CANDY::AbstractDataLoader.

8.91.2.2 getQuery()

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

get the query tensor

Returns

the generated query tensor

Reimplemented from CANDY::AbstractDataLoader.

8.91.2.3 setConfig()

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

Set the GLOBAL config map related to this loader.

Parameters

cfg	The config map

Returns

bool whether the config is successfully set

Note

Reimplemented from CANDY::AbstractDataLoader.

8.91.2.4 tensorFromFVECS()

```
torch::Tensor CANDY::FVECSDataLoader::tensorFromFVECS (
              std::string fname )  [static]
```

the inline function to load tensor from fvecs file

Parameters

fname	the name of file

Returns

the geneared tensor

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

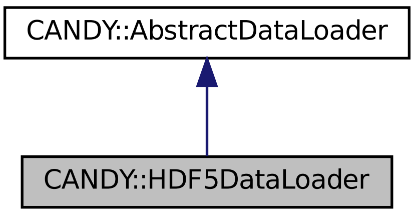
- include/DataLoader/FVECSDataLoader.h
- src/DataLoader/FVECSDataLoader.cpp

8.92 CANDY::HDF5DataLoader Class Reference

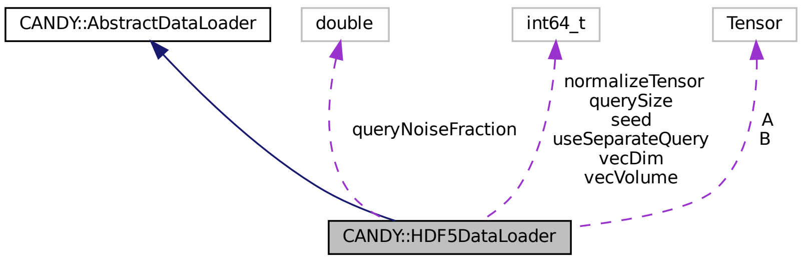
The class for loading ∗.hdf5 or ∗.h5 file, as specified in `https://github.com/HDFGroup/hdf5`.

`#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**

8.92.1 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), I64
- vecVolume, the volume of vectors, default 10000, I64
- normalizeTensor, whether or not normalize the tensors in L2, 1 (yes), I64
- dataPath, the path to the data file, datasets/hdf5/sun/sun.hdf5, String
- useSeparateQuery, whether or not load query separately, 1, I64
- 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, I64
- seed, the random seed, default 7758258, I64

: default name tags

- hdf5: HDF5DataLoader

8.92.2 Member Function Documentation

8.92.2.1 getData()

```
torch::Tensor CANDY::HDF5DataLoader::getData ( )  [virtual]
```

get the data tensor

Returns

the generated data tensor

Reimplemented from CANDY::AbstractDataLoader.

8.92.2.2 getQuery()

```
torch::Tensor CANDY::HDF5DataLoader::getQuery ( )  [virtual]
```

get the query tensor

Returns

the generated query tensor

Reimplemented from CANDY::AbstractDataLoader.

8.92.2.3 setConfig()

```
bool CANDY::HDF5DataLoader::setConfig (
             INTELLI::ConfigMapPtr cfg )  [virtual]
```

Set the GLOBAL config map related to this loader.

Parameters

cfg	The config map

Returns

bool whether the config is successfully set

Note

Reimplemented from CANDY::AbstractDataLoader.

8.92.2.4 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

fname	the name of file
attr	the attribute in hdf5 file

Returns

the geneaetd tensor

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

- include/DataLoader/HDF5DataLoader.h
- src/DataLoader/HDF5DataLoader.cpp

8.93 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)

8.93.1 Detailed Description

template<typename T>
class CANDY::FLANN::Heap< T >

heap structure used by FlannIndex

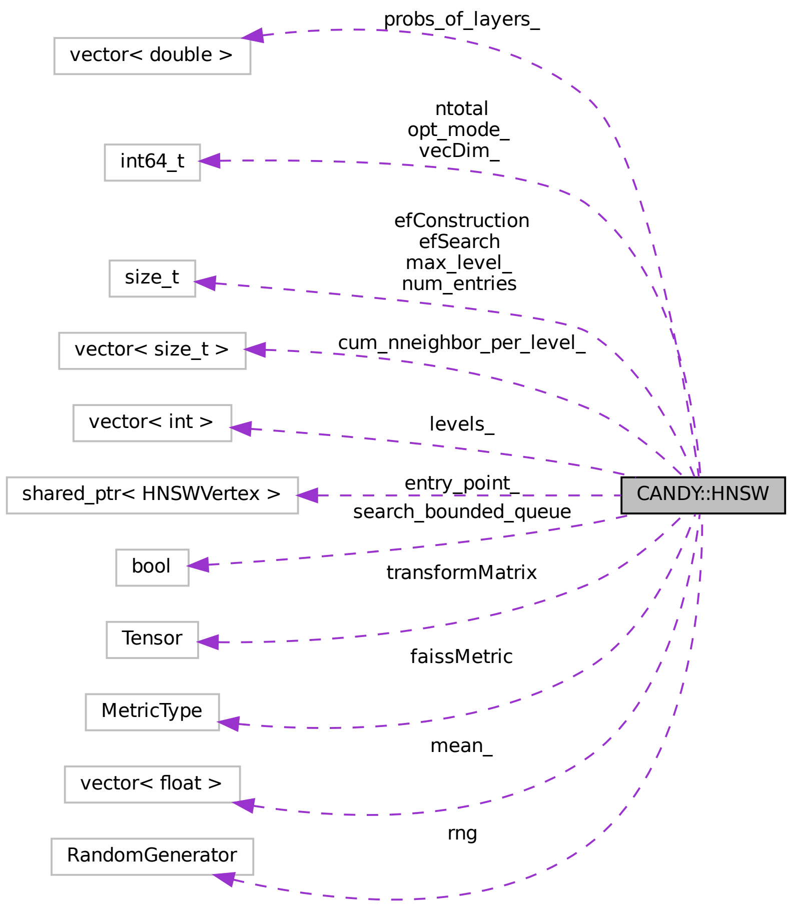
The documentation for this class was generated from the following file:

- include/CANDY/FlannIndex/FlannUtils.h

8.94 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

8.94.1 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

8.94.2 Member Function Documentation

8.94.2.1 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

disq	DistanceQuery whose query is set as the new vertex to insert
pt_id	new vector ptr
nearest	greedy-searched nearest vector to new vector. Search starting from entry point
d_nearest	distance between nearest vector and query
level	assigned level
vt	VisitedTable

8.94.2.2 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

qdis	distance queryer init with the query
assigned_level	the query's assigned level from which to add links
pt_id	query's vertex pointer
vt	visited table

8.94.2.3 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

layer_no	layer number

Returns

number of neighbors for layer layer_no

8.94.2.4 nb_neighbors()

```
size_t CANDY::HNSW::nb_neighbors (
            size_t layer_no )
```

number of neighbors for layer layer_no

Parameters

layer_no	layer number

Returns

number of neighbors for layer layer_no

8.94.2.5 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

level	level to be searched
begin	begin index
end	end index

8.94.2.6 prepare_level_tab()

```
int CANDY::HNSW::prepare_level_tab (
             torch::Tensor & x,
             bool preset_levels,
             bool is_NSW )
```

assign levels to new vectors

Parameters

x	new vectors to be assigned
preset_levels	if levels have been init for new vectors
is_NSW	if this is an NSW structure rather than HNSW

Returns

max_level assigned for new vectors

8.94.2.7 random_level()

```
int CANDY::HNSW::random_level ( )
```

generate a random level

Returns

random level

8.94.2.8 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

qdis	distance queryer init with the query to be searched
k	top K neighbors
I	results for vectors
D	results for distances
vt	vistied table

8.94.2.9 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

layer_no	layer to update
nb	neighbor number to update to

8.94.2.10 set_probs()

```
void CANDY::HNSW::set_probs (
            int64_t M,
            float levelMult )
```

set probabilities of a level to be assigned for new vectors

Parameters

M	number of neighbors
levelMult	1/log(M) to distribute the probability

8.94.3 Member Data Documentation

8.94.3.1 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;

8.94.3.2 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

8.95 HNSWAlter Class Reference

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

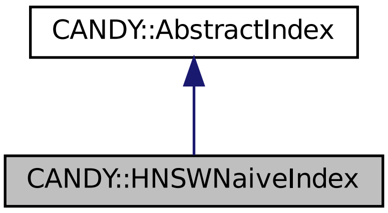
- include/CANDY/HNSWNaive/HNSWAlter.h

8.96 CANDY::HNSWNaiveIndex Class Reference

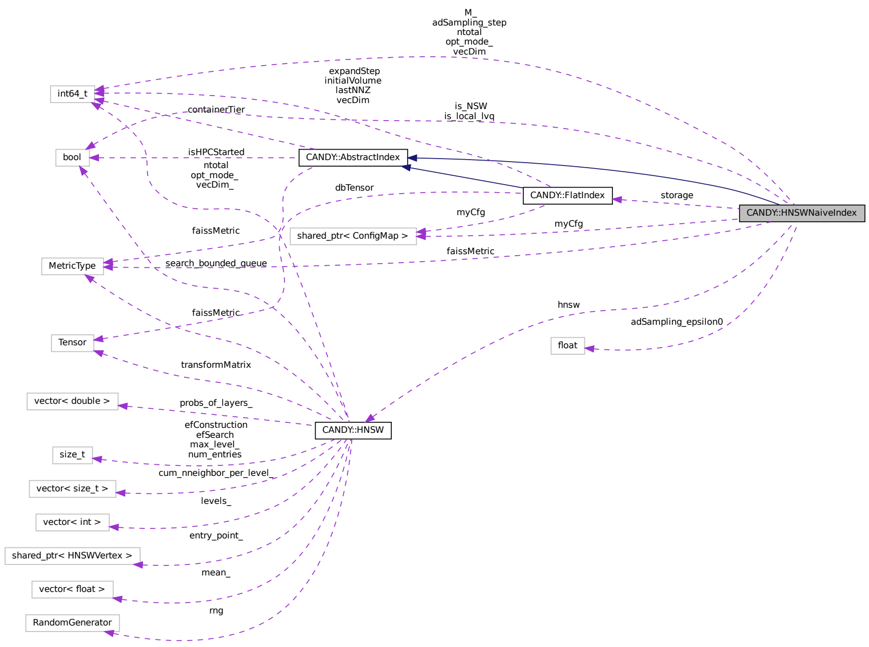
The class of a HNSW index approach, store the data in each vertex.

```
#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_lvq** = 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

8.96.1 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, I64
- maxConnection, number of maximum neighbor connection at each level, default 32, I64
- is_NSW, whether initialized as an NSW index, default 0 (init as HNSW), I64

8.96.2 Member Function Documentation

8.96.2.1 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.96.2.2 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.96.2.3 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.96.2.4 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.96.2.5 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

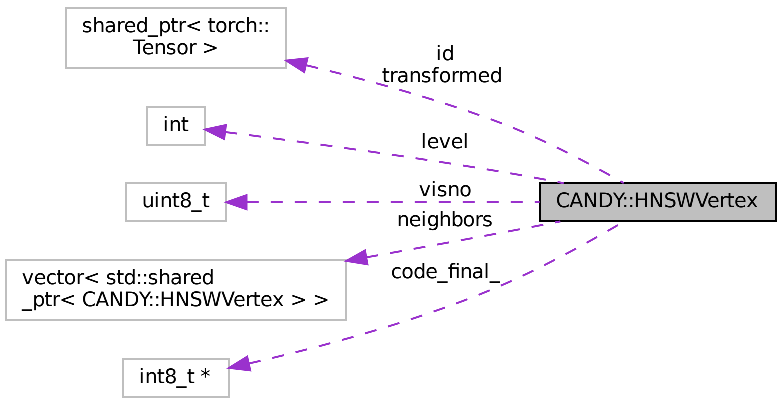
The documentation for this class was generated from the following files:

- include/CANDY/HNSWNaiveIndex.h
- src/CANDY/HNSWNaiveIndex.cpp

8.97 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**

8.97.1 Detailed Description

The class of a HNSW vertex, storing the data in each vertex.

Note

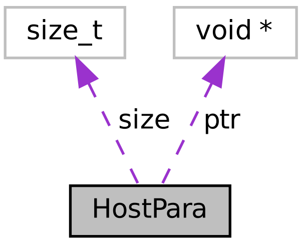
now storing each vertex's neighbors, visited number and level, with a pointer to the vector

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

- include/CANDY/HNSWNaive/HNSW.h

8.98 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:

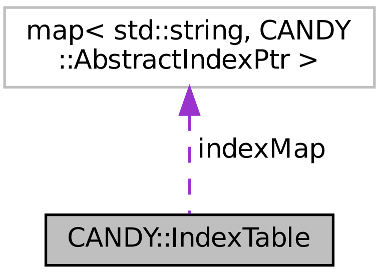
- include/CL/CLContainer.hpp

8.99 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**

8.99.1 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

8.99.2 Member Function Documentation

8.99.2.1 addIndex()

```
void CANDY::IndexTable::addIndex (
            CANDY::AbstractIndexPtr anew,
            std::string tag )  [inline]
```

To register a new ALGO.

Parameters

anew	The new algo
tag	THe name tag

8.99.2.2 getIndex()

```
CANDY::AbstractIndexPtr CANDY::IndexTable::getIndex (
            std::string name )  [inline]
```

find a dataloader in the table according to its name

Parameters

name	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

8.100 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

8.100.1 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

8.101 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

8.101.1 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

8.102 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 custom←
  ExpandSize=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

8.102.1 Member Function Documentation

8.102.1.1 appendRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::appendRows (
            TensorPtr tHeadP,
            TensorPtr tTailP )  [inline], [static]
```

append rows to the head tensor

Parameters

tHead	the head tensor, using shared pointer
tTail	the tail tensor, using shared pointer

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.2 appendRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::appendRows (
            torch::Tensor * tHead,
            torch::Tensor * tTail )  [inline], [static]
```

append rows to the head tensor

Parameters

tHead	the head tensor, using pointer
tTail	the tail tensor, using poniter

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.3 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

tHead	the head tensor, using shared pointer
tTail	the tail tensor, using sahred poniter
∗*lastNNZ*	the original last non zero row in tHead, will be changed
customExpandSize	the customized expansion size of buffer,

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.4 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

tHead	the head tensor, using pointer
tTail	the tail tensor, using poniter
∗*lastNNZ*	the original last non zero row in tHead, will be changed
customExpandSize	the customized expansion size of buffer,

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.5 deleteRow() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRow (
            TensorPtr tp,
            int64_t rowIdx )  [inline], [static]
```

delete a row of a tensor

Parameters

t	the tensor under shared pointer
rowIdx	the row to be deleted

Returns

bool, whether the operation is successful

8.102.1.6 deleteRow() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRow (
            torch::Tensor * tensor,
            int64_t rowIdx ) [inline], [static]
```

delete a row of a tensor

Parameters

t	the tensor pointer
rowIdx	the row to be deleted

Returns

bool, whether the operation is successful

8.102.1.7 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

tensor	the tensor shared pointer
rowIdx	the row to be deleted
∗*lastNNZ*	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.102.1.8 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

tensor	the tensor pointer
rowIdx	the row to be deleted
∗*lastNNZ*	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.102.1.9 deleteRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::deleteRows (
            TensorPtr tp,
            std::vector< int64_t > & rowIdx )  [inline], [static]
```

delete rows of a tensor

Parameters

t	the tensor under shared pointer
rowIdx	the rows to be deleted

Returns

bool, whether the operation is successful

8.102.1.10 deleteRows() [2/2]

```
static bool INTELLI::IntelliTensorOP::deleteRows (
            torch::Tensor * tensor,
            std::vector< int64_t > & rowIdx )  [inline], [static]
```

delete rows of a tensor

Parameters

t	the tensor pointer
rowIdx	the rows to be deleted

Returns

bool, whether the operation is successful

8.102.1.11 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

tensor	the tensor shared pointer
rowIdx	the rows to be deleted
∗*lastNNZ*	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.102.1.12 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

tensor	the tensor pointer
rowIdx	the rows to be deleted
∗*lastNNZ*	the original last non zero row in tensor, will be changed

Returns

bool, whether the operation is successful

8.102.1.13 editRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::editRows (
            TensorPtr tHead,
            TensorPtr tTail,
            int64_t startRow )  [inline], [static]
```

edit rows in the head tensor

Parameters

tHead	the head tensor, using shared pointer
tTail	the tail tensor, using shared poniter
startRow,the	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

8.102.1.14 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

tHead	the head tensor, using pointer
tTail	the tail tensor, using poniter
startRow,the	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.15 insertRows() [1/2]

```
static bool INTELLI::IntelliTensorOP::insertRows (
            TensorPtr tHead,
            TensorPtr tTail,
            int64_t startRow )  [inline], [static]
```

insert rows to the head tensor

Parameters

tHead	the head tensor, using shared pointer
tTail	the tail tensor, using shared poniter
startRow,the	starRow of tTail to be appeared afeter insertion

Returns

bool, whether the operation is successful

8.102.1.16 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

tHead	the head tensor, using pointer
tTail	the tail tensor, using poniter
startRow,the	starRow of tTail to be appeared afeter insertion

Note

The number of columnes must be matched

Returns

bool, whether the operation is successful

8.102.1.17 l2Normalize()

```
static torch::Tensor INTELLI::IntelliTensorOP::l2Normalize (
            torch::Tensor & a )  [inline], [static]
```

to normalize the tensor in each column, using l2

Parameters

a	the input tensor

Returns

the result tensor

8.102.1.18 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

a	the input tensor
sampledRows	the number of rows to be sampled

Returns

the result tensor

8.102.1.19 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

A	the tensor
fname	the name of file

Returns

bool, the load is successful or not

8.102.1.20 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

A	the tensor
ru	the binart in std::vector<uint8_t>

Returns

bool, the load is successful or not

8.102.1.21 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

A	the tensor
fname	the name of file

Returns

bool, the output is successful or not

8.102.1.22 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

A	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

8.103 INTELLITensorOP Class Reference

The common tensor functions packed in class.

8.103.1 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:

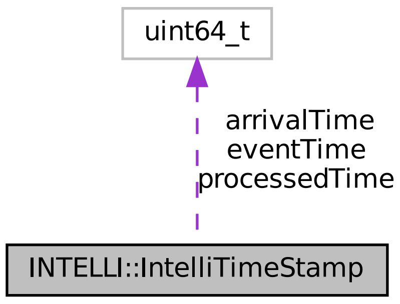
- include/Utils/IntelliTensorOP.hpp

8.104 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

8.104.1 Detailed Description

The class to define a timestamp.

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

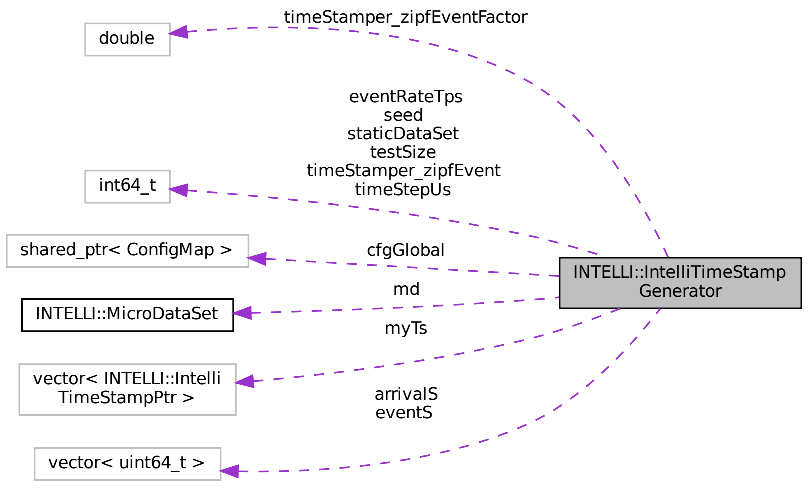
- include/Utils/IntelliTimeStampGenerator.h

8.105 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

8.105.1 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 zpf 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

8.105.2 Member Function Documentation

8.105.2.1 generateArrival()

```
void INTELLI::IntelliTimeStampGenerator::generateArrival ( )  [protected]
```

generate the vector of arrival

Note

As we do not consider OoO now, this is a dummy function

8.105.2.2 getTimeStamps()

```
std::vector< INTELLI::IntelliTimeStampPtr > INTELLI::IntelliTimeStampGenerator::getTimeStamps
( )  [virtual]
```

get the vector of time stamps

Returns

the vector

8.105.2.3 setConfig()

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

Set the GLOBAL config map related to this TimerStamper.

Parameters

cfg	The config map

Returns

bool whether the config is successfully set

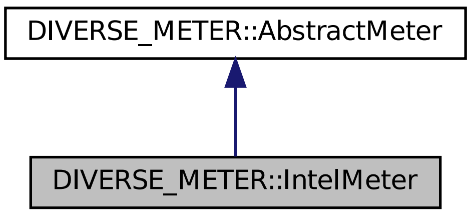
The documentation for this class was generated from the following files:

- include/Utils/IntelliTimeStampGenerator.h
- src/Utils/IntelliTimeStampGenerator.cpp

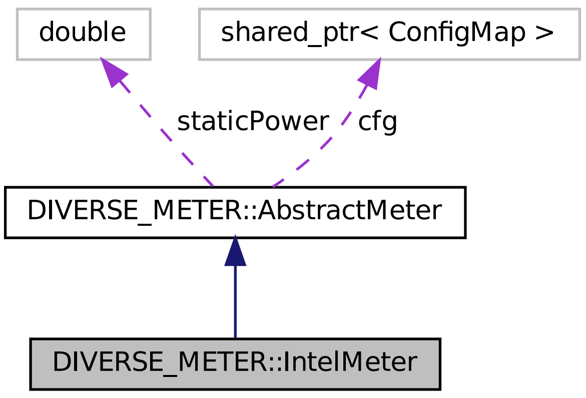
8.106 DIVERSE_METER::IntelMeter Class Reference

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

8.106.1 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 resluts

  Warning

  : only works for some x64 machines

  Note

  : no peak power support, tag is "intelMsr"

8.106.2 Member Function Documentation

8.106.2.1 setConfig()

```
void IntelMeter::setConfig (
            INTELLI::ConfigMapPtr _cfg )   [virtual]
```

to set the configmap

Parameters

cfg	the config map

Reimplemented from DIVERSE_METER::AbstractMeter.

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

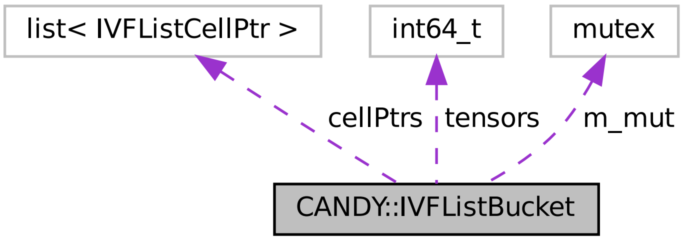
- include/Utils/Meters/IntelMeter/IntelMeter.hpp
- src/Utils/Meters/IntelMeter/IntelMeter.cpp

8.107 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**

8.107.1 Detailed Description

a bucket of multiple IVFListCell

8.107.2 Member Function Documentation

8.107.2.1 deleteTensor()

```
bool CANDY::IVFListBucket::deleteTensor (
            torch::Tensor & t,
            bool isConcurrent = false )
```

delete a tensor

Note

will check the equal condition by torch::equal

Parameters

t	the tensor
isConcurrent	whether this process is concurrently executed •

Returns

bool whether the tensor is really deleted

8.107.2.2 deleteTensorWithEncode()

```
bool CANDY::IVFListBucket::deleteTensorWithEncode (
            torch::Tensor & t,
            std::vector< uint8_t > & encode,
            bool isConcurrent = false )
```

delete a tensor with its encode

Parameters

t	the tensor
encode	the corresponding encode
isConcurrent	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.107.2.3 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

8.107.2.4 getAllTensorsWithEncode()

```
torch::Tensor CANDY::IVFListBucket::getAllTensorsWithEncode (
            std::vector< uint8_t > & _encode )
```

get all of the tensors in list with a specific encode

Parameters

_encode	the specified encode

Returns

a 2-D tensor contain all, torch::zeros({1,1}) if got nothing

8.107.2.5 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

_encode	the specified encode
minNumber	the minimum of desired tensors
_vecDim	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

8.107.2.6 insertTensorWithEncode()

```
void CANDY::IVFListBucket::insertTensorWithEncode (
            torch::Tensor & t,
            std::vector< uint8_t > & encode,
            bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

t	the tensor
encode	the corresponding encode
isConcurrent	whether this process is concurrently executed

8.107.2.7 sizeWithEncode()

```
int64_t CANDY::IVFListBucket::sizeWithEncode (
            std::vector< uint8_t > & _encode )
```

get teh size in list with a specific encode

Parameters

_encode	the specified encode

Returns

the size under _encode

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

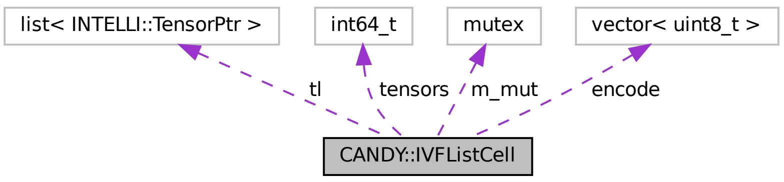
- include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h
- src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp

8.108 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**

8.108.1 Detailed Description

a cell of row tensor pointers which have the same code

8.108.2 Member Function Documentation

8.108.2.1 deleteTensor()

```
bool CANDY::IVFListCell::deleteTensor (
            torch::Tensor & t )
```

delete a tensor

Note

will check the equal condition by torch::equal

Parameters

t	the tensor @returen bool whether the tensor is really deleted

8.108.2.2 deleteTensorPtr()

```
bool CANDY::IVFListCell::deleteTensorPtr (
            INTELLI::TensorPtr tp )
```

delete a tensor pointer

Note

will check the equal condition by pointer ==

Parameters

tp	the tensor pointer @returen bool whether the tensor is realy deleted

8.108.2.3 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

8.108.2.4 insertTensor()

```
void CANDY::IVFListCell::insertTensor (
            torch::Tensor & t )
```

insert a tensor

Parameters

t	the tensor

8.108.2.5 insertTensorPtr()

```
void CANDY::IVFListCell::insertTensorPtr (
            INTELLI::TensorPtr tp )
```

insert a tensor pointer

Parameters

tp	the tensor pointer

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

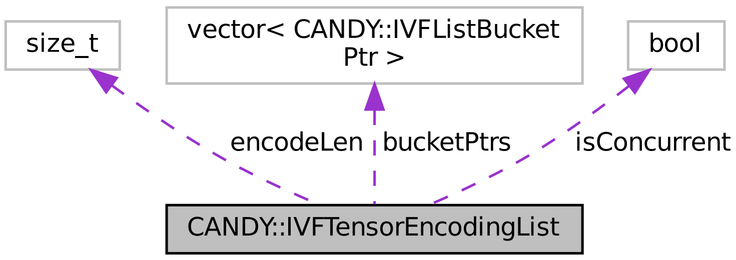
- include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h
- src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp

8.109 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:


size_t
vector< CANDY::IVFListBucket Ptr >
bool
encodeLen
bucketPtrs
isConcurrent
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 is↩
Concurrent=false)

  insert a tensor with its encode
- bool deleteTensorWithEncode (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool is↩
Concurrent=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

8.109.1 Detailed Description

The inverted file (IVF) list to organize tensor and its encodings.

8.109.2 Member Function Documentation

8.109.2.1 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	whether this process is concurrently executed

Returns

bool whether the tensor is really deleted

8.109.2.2 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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

8.109.2.3 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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

8.109.2.4 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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

8.109.2.5 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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!

8.109.2.6 init()

```
void CANDY::IVFTensorEncodingList::init (
            size_t bkts,
            size_t _encodeLen )
```

init this IVFList

Parameters

bkts	the number of buckets
_encodeLen	the length of tensors' encoding

8.109.2.7 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

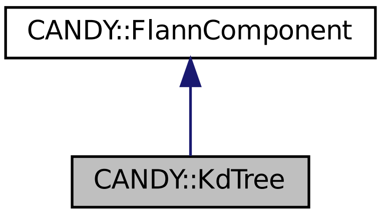
t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	whether this process is concurrently executed

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

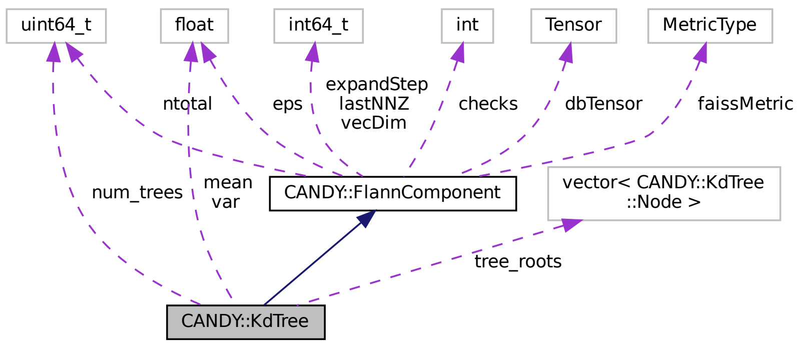
- include/CANDY/OnlinePQIndex/IVFTensorEncodingList.h
- src/CANDY/OnlinePQIndex/IVFTensorEncodingList.cpp

8.110 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

8.110.1 Member Function Documentation

8.110.1.1 addPoints()

```
void CANDY::KdTree::addPoints (
            torch::Tensor & t )  [override], [virtual]
```

add data into the tree either by reconstruction or appending

Parameters

t	new data

Reimplemented from CANDY::FlannComponent.

8.110.1.2 addPointToTree()

```
void CANDY::KdTree::addPointToTree (
             NodePtr node,
             int64_t idx )
```

add dbTensor[idx] to tree with root as node

Parameters

node	typically a tree root
idx	index in dbTensor

8.110.1.3 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

idx	index of this vector
count	number of vectors in this sublist

Returns

8.110.1.4 getNeighbors()

```
void CANDY::KdTree::getNeighbors (
             FLANN::ResultSet & result,
             const float * vec,
             int maxCheck,
             float epsError )
```

Parameters

result	
vec	
maxCheck	
epsError	

8.110.1.5 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

q	query data to be searched
idx	result vectors indices
distances	result vectors' distances with query
aknn	number of approximate neighbors

Returns

number of results obtained

Reimplemented from CANDY::FlannComponent.

8.110.1.6 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

ind	index of this vector
count	number of vectors in this sublist
index	index where the sublist split
cutfeat	index of highest variance as cut feature
cutval	value of highest variance

8.110.1.7 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

ind	index of the list
count	count of the list
cutfeat	the chosen feature
cutval	the threshold value to be compared
lim1	split index candidate for meansplit
lim2	split index candidate for meansplit

8.110.1.8 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

result	priority queue to store results
vec	vector to be searched
node	current node to be traversed
mindist	current minimum distance obtained
checkCount	count of checks on multiple trees
maxCheck	max check on multiple trees
epsError	error to be compared with worst distance
heap	heap structure to store branches
checked	visited bitmap

TODO:not sure + IP

8.110.1.9 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

v	values of variance

Returns

the index of randomly chosen highest variance

8.110.1.10 setConfig()

```
bool CANDY::KdTree::setConfig (
             INTELLI::ConfigMapPtr cfg )  [override], [virtual]
```

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::FlannComponent.

8.110.1.11 setParams()

```
bool CANDY::KdTree::setParams (
             FlannParam param )  [override], [virtual]
```

set the params from auto-tuning

Parameters

param	best param

Returns

true if success

Reimplemented from CANDY::FlannComponent.

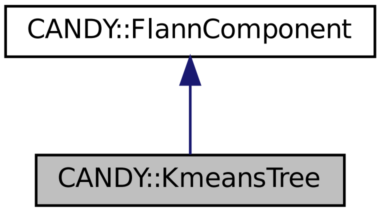
The documentation for this class was generated from the following files:

- include/CANDY/FlannIndex/KdTree.h
- src/CANDY/FlannIndex/KdTree.cpp

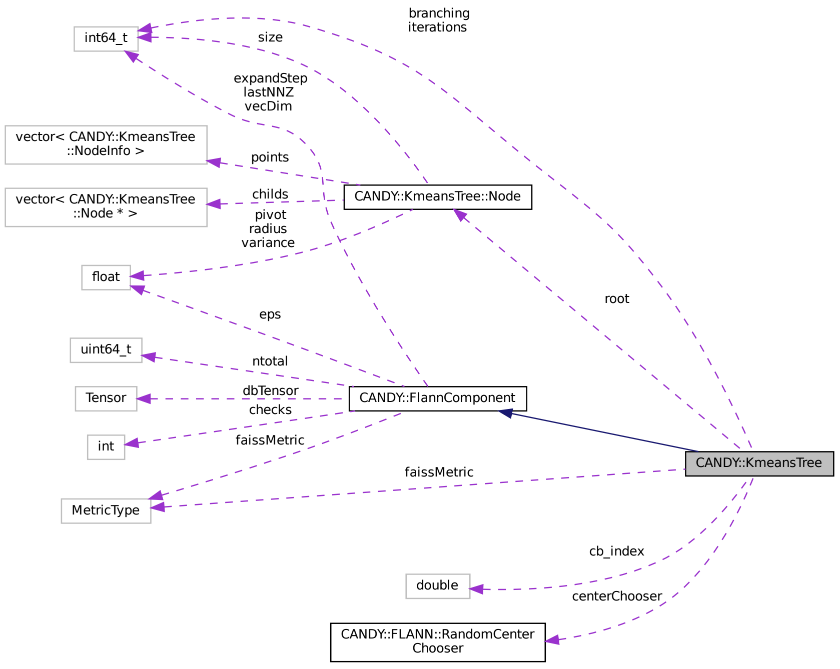
8.111 CANDY::KmeansTree Class Reference

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

8.111.1 Detailed Description

The structure representing hierarchical k-means tree used in FLANN.

8.111.2 Member Function Documentation

8.111.2.1 addPoints()

```
void CANDY::KmeansTree::addPoints (
            torch::Tensor & t ) [override], [virtual]
```

add data into the tree either by reconstruction or appending

Parameters

t	new data

Reimplemented from CANDY::FlannComponent.

8.111.2.2 addPointToTree()

```
void CANDY::KmeansTree::addPointToTree (
            NodePtr node,
            int64_t idx,
            float dist )
```

add dbTensor[idx] to tree with root as node

Parameters

node	typically a tree root
idx	index in dbTensor
dist	

8.111.2.3 computeClustering()

```
void CANDY::KmeansTree::computeClustering (
            NodePtr node,
            int64_t * indices,
            int64_t indices_length,
            int64_t branching )
```

#brief compute the cluster iteratively

Parameters

node	the node where the cluster starts
indices	indexes to be involved
indices_length	length of indexes to be involved
branching	number of branching in tree

8.111.2.4 computeNodeStat()

```
void CANDY::KmeansTree::computeNodeStat (
            NodePtr node,
            std::vector< int64_t > & indices )
```

compute the radius, variance and mean for this cluster

Parameters

node	the node representing the cluster
indices	the indexes within the cluster

8.111.2.5 explore()

```
int64_t CANDY::KmeansTree::explore (
            NodePtr node,
            float * q,
            FLANN::Heap< BranchSt > * heap )
```

explore from the node for the closest center

Parameters

node	node to be explored
q	query vector
heap	heap set

Returns

the index of center

8.111.2.6 findNN()

```
void CANDY::KmeansTree::findNN (
            NodePtr node,
            FLANN::ResultSet & result,
            float * vec,
            int & check,
            int maxCheck,
            FLANN::Heap< BranchSt > * heap )
```

practice KNN search

Parameters

node	starting node
result	result set
vec	query vector
check	current check time
maxCheck	max check times
heap	heap set

8.111.2.7 getNeighbors()

```
void CANDY::KmeansTree::getNeighbors (
            FLANN::ResultSet & result,
            float * vec,
            int maxCheck )
```

called by knnSearch, to search the vec within the true

Parameters

result	result set
vec	vector to be searched
maxCheck	max times to check

8.111.2.8 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

q	query data to be searched
idx	result vectors indices
distances	result vectors' distances with query
aknn	number of approximate neighbors

Returns

number of results obtained

Reimplemented from CANDY::FlannComponent.

8.111.2.9 setConfig()

```
bool CANDY::KmeansTree::setConfig (
            INTELLI::ConfigMapPtr cfg )  [override], [virtual]
```

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::FlannComponent.

8.111.2.10 setParams()

```
bool CANDY::KmeansTree::setParams (
            FlannParam param )  [override], [virtual]
```

set the params from auto-tuning

Parameters

param	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

8.112 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)

8.112.1 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)

8.112.2 Member Function Documentation

8.112.2.1 getAvgCpu()

```
double INTELLI::MemoryTracker::getAvgCpu ( )  [inline]
```

To return the average Cpu utilization rate during the sampling.

Returns

the fractional

8.112.2.2 getAvgMem()

```
size_t INTELLI::MemoryTracker::getAvgMem ( )  [inline]
```

To return the average memory usage during the sampling.

Returns

size_t the memory usage in KB

8.112.2.3 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

8.112.2.4 getMaxCpu()

```
double INTELLI::MemoryTracker::getMaxCpu ( )  [inline]
```

To return the max Cpu utilization rate during the sampling.

Returns

the fractional

8.112.2.5 getMaxMem()

```
size_t INTELLI::MemoryTracker::getMaxMem ( )  [inline]
```

To return the max memory usage during the sampling.

Returns

size_t the memory usage in KB

8.112.2.6 start()

```
void INTELLI::MemoryTracker::start (
            uint64_t sec,
            uint64_t usec = 0 )  [inline]
```

To start memory usage tracing.

Parameters

sec	the second of sampling
usec	the micro-second of sampling

Note

call after setPerfList

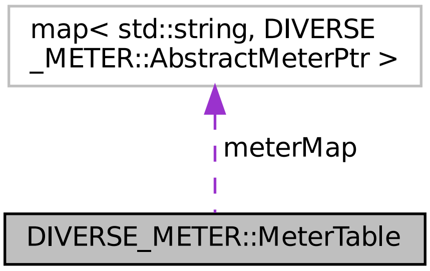
The documentation for this class was generated from the following files:

- include/Utils/MemTracker.h
- src/Utils/MemTracker.cpp

8.113 DIVERSE_METER::MeterTable Class Reference

The table class to index all meters.

Collaboration diagram for DIVERSE_METER::MeterTable:


map< std::string, DIVERSE _METER::AbstractMeterPtr >
meterMap
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**

8.113.1 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

8.113.2 Constructor & Destructor Documentation

8.113.2.1 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

8.113.3 Member Function Documentation

8.113.3.1 findMeter()

```
DIVERSE_METER::AbstractMeterPtr DIVERSE_METER::MeterTable::findMeter (
            std::string name )  [inline]
```

find a meter in the table according to its name

Parameters

name	The nameTag of loader

Returns

The Meter, nullptr if not found

8.113.3.2 registerNewMeter()

```
void DIVERSE_METER::MeterTable::registerNewMeter (
            DIVERSE_METER::AbstractMeterPtr dnew,
            std::string tag )  [inline]
```

To register a new meter.

Parameters

onew	The new operator
tag	THe name tag

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

- include/Utils/Meters/MeterTable.h
- src/Utils/Meters/MeterTable.cpp

8.114 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.

8.114.1 Detailed Description

The all-in-one class for the Micro dataset.

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

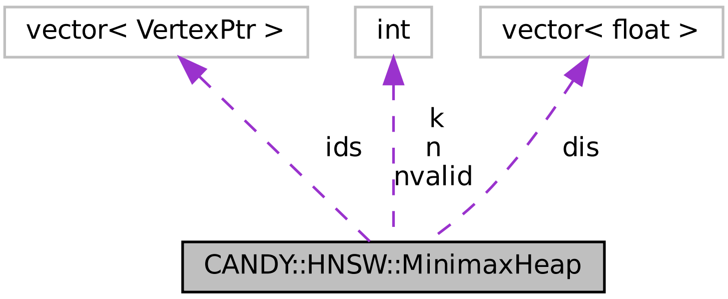
- include/Utils/MicroDataSet.hpp

8.115 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**

8.115.1 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

8.116 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

8.116.1 Member Function Documentation

8.116.1.1 hash()

```
torch::Tensor CANDY::MLPBucketIdxModel::hash (
            torch::Tensor input )  [virtual]
```

the forward hashing function

Parameters

input	The input tensor

Returns

the output tensor for encoding

8.116.1.2 init()

```
void CANDY::MLPBucketIdxModel::init (
            int64_t inputDim,
            int64_t idxMax,
            INTELLI::ConfigMapPtr extraConfig )  [virtual]
```

init the model class

Parameters

inputDim	the dimension of model ending input
outputDim	the dimension of model ending output
extraConfig	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

8.116.1.3 trainModel()

```
void CANDY::MLPBucketIdxModel::trainModel (
            torch::Tensor & x1,
            torch::Tensor & x2,
            torch::Tensor & labels )  [virtual]
```

the training function

Parameters

x1	an 2D tensor sized [n∗d]
x2	an 2D tensor sized [n∗d]
labels	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

8.117 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

8.117.1 Member Function Documentation

8.117.1.1 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

x1	an 2D tensor sized [n∗d]
x2	an 2D tensor sized [n∗d]
labels	an 1D integer tensor sized n, indicating whether x1[i] is similar to x2[i]
epochs	the number of epoches
lr	the learning rate

8.117.1.2 hash()

```
torch::Tensor CANDY::MLPHashingModel::hash (
            torch::Tensor input )  [virtual]
```

the forward hashing function

Parameters

input	The input tensor

Returns

the output tensor for encoding

8.117.1.3 init()

```
void CANDY::MLPHashingModel::init (
            int64_t inputDim,
            int64_t outputDim,
            INTELLI::ConfigMapPtr extraConfig )  [virtual]
```

init the model class

Parameters

inputDim	the dimension of model ending input
outputDim	the dimension of model ending output
extraConfig	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 in regulating variance used in training, Double, default 0
- MLTrainEpochs the number of epochs in training, I64, default 10

8.117.1.4 trainModel()

```
void CANDY::MLPHashingModel::trainModel (
             torch::Tensor & x1,
             torch::Tensor & x2,
             torch::Tensor & labels )  [virtual]
```

the training function

Parameters

x1	an 2D tensor sized [n∗d]
x2	an 2D tensor sized [n∗d]
labels	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/MLPHashingModel.h
- src/CANDY/HashingModels/MLPHashingModel.cpp

8.118 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.

8.118.1 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

T	The return type of the futures.

8.118.2 Constructor & Destructor Documentation

8.118.2.1 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

*num_↩ futures_*	The desired number of futures to store.

8.118.3 Member Function Documentation

8.118.3.1 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.

8.118.3.2 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	The index of the desired future.

Returns

The future.

8.118.3.3 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

future	The future to append.

8.118.3.4 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

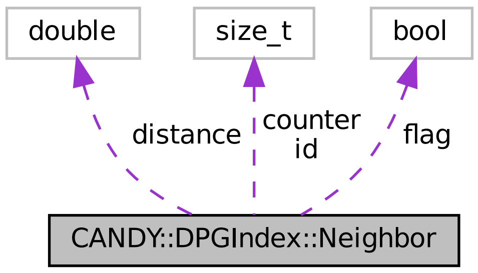
The number of futures.

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

- include/Utils/BS_thread_pool.hpp

8.119 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

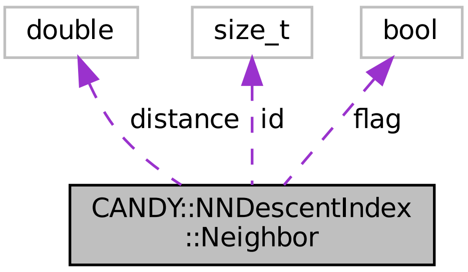
- size_t **id**
- double **distance**
- bool **flag**
- size_t **counter**

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

- include/CANDY/DPGIndex.h

8.120 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

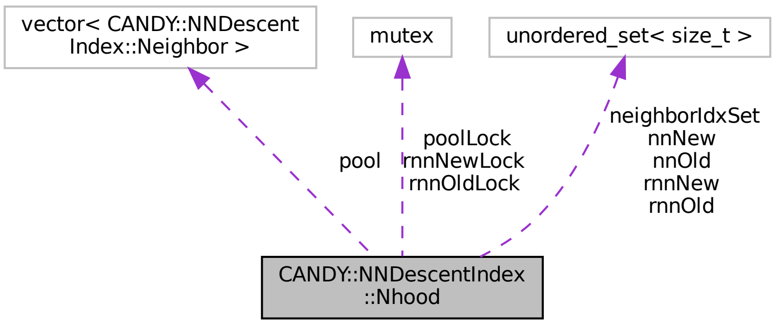
- size_t **id**
- double **distance**
- bool **flag**

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

- include/CANDY/NNDescentIndex.h

8.121 CANDY::NNDescentIndex::Nhood Struct Reference

Collaboration diagram for CANDY::NNDescentIndex::Nhood:



Public Member Functions

- **Nhood** (const Nhood &other)

Public Attributes

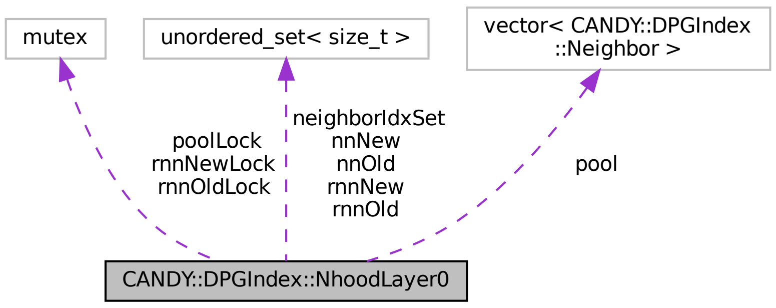
- std::mutex **poolLock**
- std::vector< Neighbor > **pool**
- std::unordered_set< size_t > **neighborIdxSet**
- std::unordered_set< size_t > **nnOld**
- std::unordered_set< size_t > **nnNew**
- std::mutex **rnnOldLock**
- std::unordered_set< size_t > **rnnOld**
- std::mutex **rnnNewLock**
- std::unordered_set< size_t > **rnnNew**

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

- include/CANDY/NNDescentIndex.h

8.122 CANDY::DPGIndex::NhoodLayer0 Struct Reference

Collaboration diagram for CANDY::DPGIndex::NhoodLayer0:



Public Member Functions

- **NhoodLayer0** (const NhoodLayer0 &other)

Public Attributes

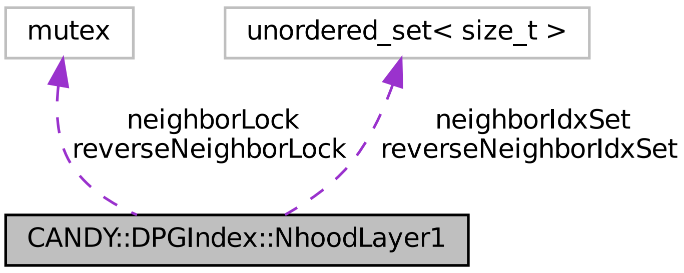
- std::mutex **poolLock**
- std::vector< Neighbor > **pool**
- std::unordered_set< size_t > **neighborIdxSet**
- std::unordered_set< size_t > **nnOld**
- std::unordered_set< size_t > **nnNew**
- std::mutex **rnnOldLock**
- std::unordered_set< size_t > **rnnOld**
- std::mutex **rnnNewLock**
- std::unordered_set< size_t > **rnnNew**

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

- include/CANDY/DPGIndex.h

8.123 CANDY::DPGIndex::NhoodLayer1 Struct Reference

Collaboration diagram for CANDY::DPGIndex::NhoodLayer1:



Public Member Functions

- **NhoodLayer1** (const NhoodLayer1 &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:

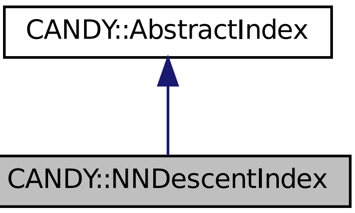
- include/CANDY/DPGIndex.h

8.124 CANDY::NNDescentIndex Class Reference

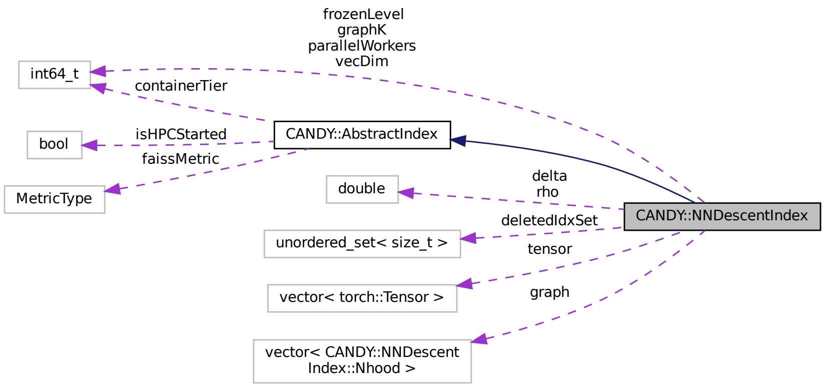
An index whose core algorithm is only used for offline construction, but based on its main data structure we have implemented online update operations that need to be optimized.

```
#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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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< Nhood > **graph**
- std::vector< torch::Tensor > **tensor**
- std::unordered_set< size_t > **deletedIdxSet**

Additional Inherited Members

8.124.1 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, I64, default 1 (set this to less than 0 will use max hardware_concurrency);
- vecDim, the dimension of vectors, default 768, I64
- graphK, the neighbors of every node in internal data struct, default 20, I64
- 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 @warnning Make sure you are using 2D tensors!

8.124.2 Member Function Documentation

8.124.2.1 deleteTensor()

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

delete a tensor

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 from CANDY::AbstractIndex.

8.124.2.2 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.

8.124.2.3 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.124.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.124.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.124.2.6 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

Reimplemented from CANDY::AbstractIndex.

8.124.2.7 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.124.2.8 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised
w	the revised value

Returns

bool whether the revising is successful

Note

only support to delete and insert, no straightforward revision

Reimplemented from CANDY::AbstractIndex.

8.124.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.124.2.10 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.124.2.11 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

Reimplemented from CANDY::AbstractIndex.

8.124.2.12 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

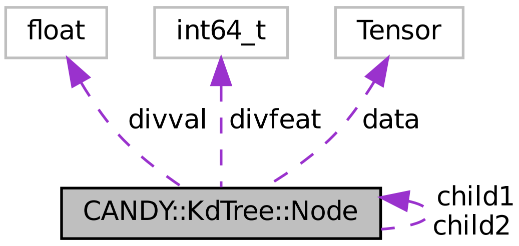
Reimplemented from CANDY::AbstractIndex.

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

- include/CANDY/NNDescentIndex.h
- src/CANDY/NNDescentIndex.cpp

8.125 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

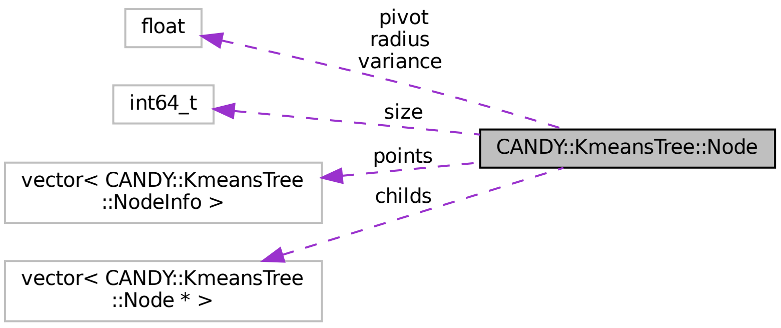
  Node data.
- Node ∗ **child1**
- Node ∗ **child2**

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

- include/CANDY/FlannIndex/KdTree.h

8.126 CANDY::KmeansTree::Node Struct Reference

Collaboration diagram for CANDY::KmeansTree::Node:


float
pivot
radius
variance
int64_t
size
CANDY::KmeansTree::Node
points
vector< CANDY::KmeansTree ::NodeInfo >
childs
vector< 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:

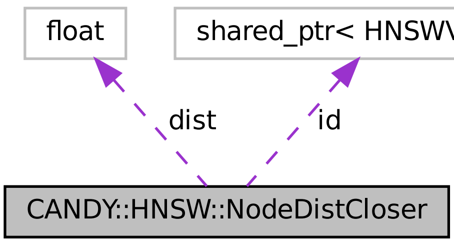
- include/CANDY/FlannIndex/Kmeans.h

8.127 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**

8.127.1 Detailed Description

sort pairs from nearest to farthest by distance

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

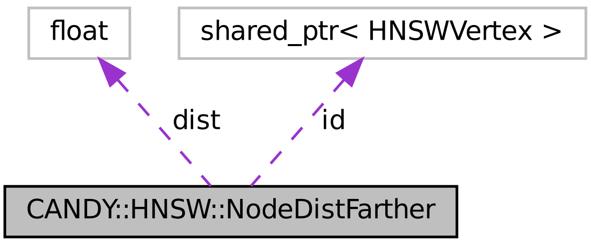
- include/CANDY/HNSWNaive/HNSW.h

8.128 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**

8.128.1 Detailed Description

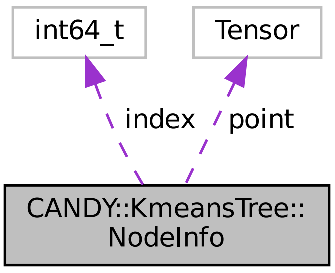
sort pairs from farthest to nearest

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

- include/CANDY/HNSWNaive/HNSW.h

8.129 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:

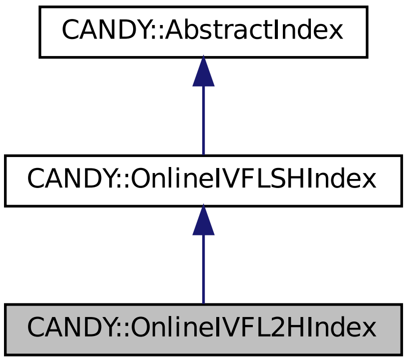
- include/CANDY/FlannIndex/Kmeans.h

8.130 CANDY::OnlineIVFL2HIndex Class Reference

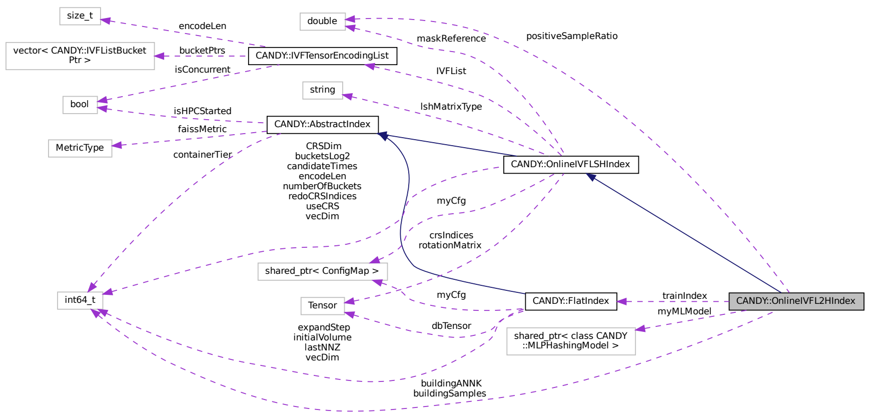
A L2H (learning 2 hash) indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The L2H function is using ML to approximate spectral hashing principles (NIPS 2008)

```
#include <CANDY/OnlineIVFL2HIndex.h>
```

Inheritance diagram for CANDY::OnlineIVFL2HIndex:



Collaboration diagram for CANDY::OnlineIVFL2HIndex:



Public Member Functions

- virtual bool loadInitialTensor (torch::Tensor &t)

  load the initial tensors of a data base, use this BEFORE insertTensor
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specific config related to one index
- virtual bool loadInitialTensorAndQueryDistribution (torch::Tensor &t, torch::Tensor &query)

  load the initial tensors and query distributions of a data base, use this BEFORE insertTensor

Protected Member Functions

- virtual torch::Tensor **randomProjection** (torch::Tensor &a)
- void trainModelWithData (torch::Tensor &t)

  self-supervised learning on data, including automatic labeling

Protected Attributes

- MLPHashingModelPtr **myMLModel** = nullptr
- int64_t **buildingSamples** = -1
- int64_t **buildingANNK** = 10
- FlatIndex **trainIndex**
- double **positiveSampleRatio** = 0.1

Additional Inherited Members

8.130.1 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 titer 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)

8.130.2 Member Function Documentation

8.130.2.1 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.130.2.2 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

t	the data tensor
query	the example query tensor

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.130.2.3 setConfig()

```
bool CANDY::OnlineIVFL2HIndex::setConfig (
            INTELLI::ConfigMapPtr cfg )  [virtual]
```

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

generate the rotation matrix for random projection

Reimplemented from CANDY::OnlineIVFLSHIndex.

8.130.2.4 trainModelWithData()

```
void CANDY::OnlineIVFL2HIndex::trainModelWithData (
            torch::Tensor & t )  [protected]
```

self-supervised learning on data, including automatic labeling

Parameters

t	the input tensor

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

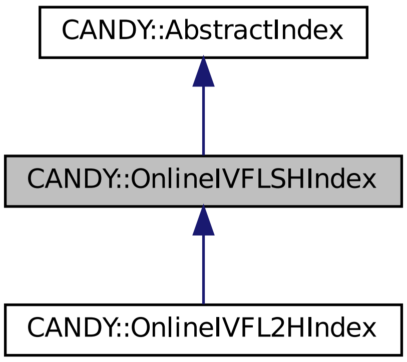
- include/CANDY/OnlineIVFL2HIndex.h
- src/CANDY/OnlineIVFL2HIndex.cpp

8.131 CANDY::OnlineIVFLSHIndex Class Reference

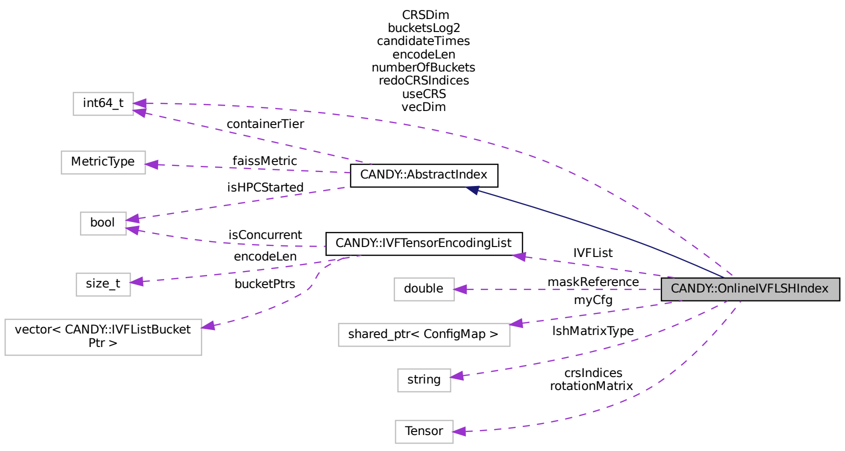
A LSH indexing, using 2-tier IVF List to manage buckets. The base tier is hamming encoding, implemented under list, the top tier is sampled summarization of hamming encoding, implemented under vector (faster access, harder to change, but less representative). The LSH function is the vanilla random projection (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 inited 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

8.131.1 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, I64
- candidateTimes, the times of k to determine minimum candidates, default 1 ,I64
- numberOfBuckets, the number of first titer buckets, default 1, I64, suggest 2^n
- encodeLen, the length of LSH encoding, in bytes, default 1, I64
- 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), I64
  - further trade off of accuracy v.s. efficiency
- CRSDim, the dimension which are not pruned by crs, 1/10 of vecDim, I64
- redoCRSIndices, whether or not re-generate the indices of CRS, 0 (No), I64

8.131.2 Member Function Documentation

8.131.2.1 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

A	the left side matrix
B	the right side matrix
idx	the indices of sampling
_crsDim	the dimension of preserved dimensions

8.131.2.2 deleteRowsInline()

```
bool CANDY::OnlineIVFLSHIndex::deleteRowsInline (
             torch::Tensor & t )  [protected]
```

the inline function of deleting rows

Parameters

t	the tensor, multiple rows

Returns

bool whether the deleting is successful

8.131.2.3 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.131.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.131.2.5 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.131.2.6 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.131.2.7 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	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:

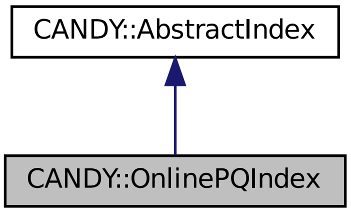
- include/CANDY/OnlineIVFLSHIndex.h
- src/CANDY/OnlineIVFLSHIndex.cpp

8.132 CANDY::OnlinePQIndex Class Reference

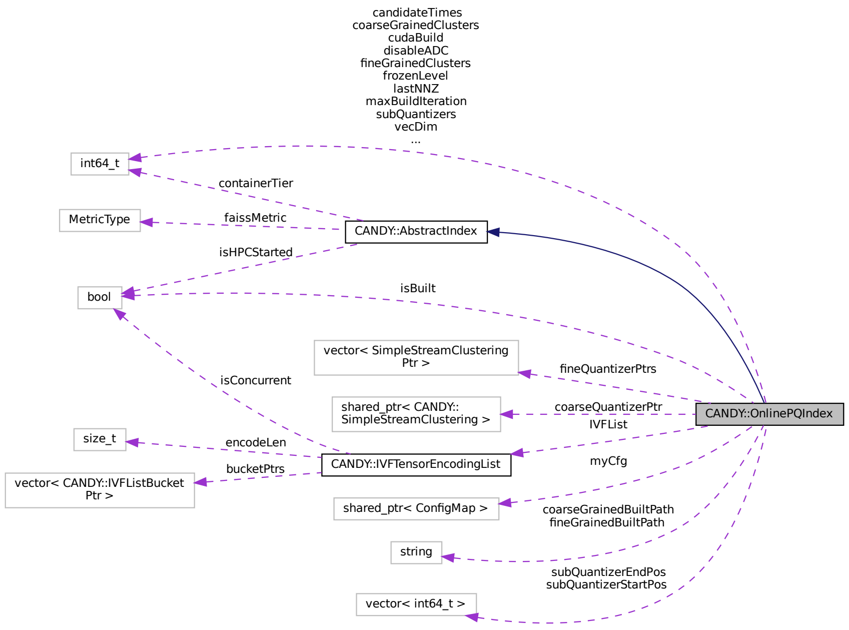
The class of online PQ approach, using IVF-style coarse-grained + fine-grained quantizers.

```
#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 inited 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

8.132.1 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, I64
- coarseGrainedClusters,the number of coarse-grained clusters, default 4096, I64
- fineGrainedClusters,the number of fine-grained clusters in each sub quantizer, default 256, 1∼256 I64
- subQuantizers, the number of sub quantizers, default 8, I64
- 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, I64
- maxBuildIteration, the maxium iterations of buildoing, default 1000, I64
- candidateTimes, the times of k to determine minimum candidates, default 1 ,I64
- disableADC, set this to 1 will disable ADC or residential computing and go back to IVFPQ, default 0 (means IVFADC mode), I64

8.132.2 Member Function Documentation

8.132.2.1 deleteRowsInline()

```
bool CANDY::OnlinePQIndex::deleteRowsInline (
            torch::Tensor & t )  [protected]
```

the inline function of deleting rows

Parameters

t	the tensor, multiple rows

Returns

bool whether the deleting is successful

8.132.2.2 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.132.2.3 fineGrainedEncode()

```
std::vector< std::vector< uint8_t > > CANDY::OnlinePQIndex::fineGrainedEncode (
             torch::Tensor & residential )  [protected]
```

1. get the output of each subquantizer

8.132.2.4 insertTensor()

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

insert a tensor

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.132.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.132.2.6 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

t	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.

8.132.2.7 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.132.2.8 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.132.2.9 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

create cluster instances

Reimplemented from CANDY::AbstractIndex.

8.132.2.10 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	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:

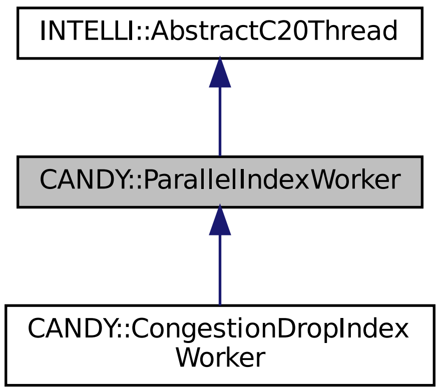
- include/CANDY/OnlinePQIndex.h
- src/CANDY/OnlinePQIndex.cpp

8.133 CANDY::ParallelIndexWorker Class Reference

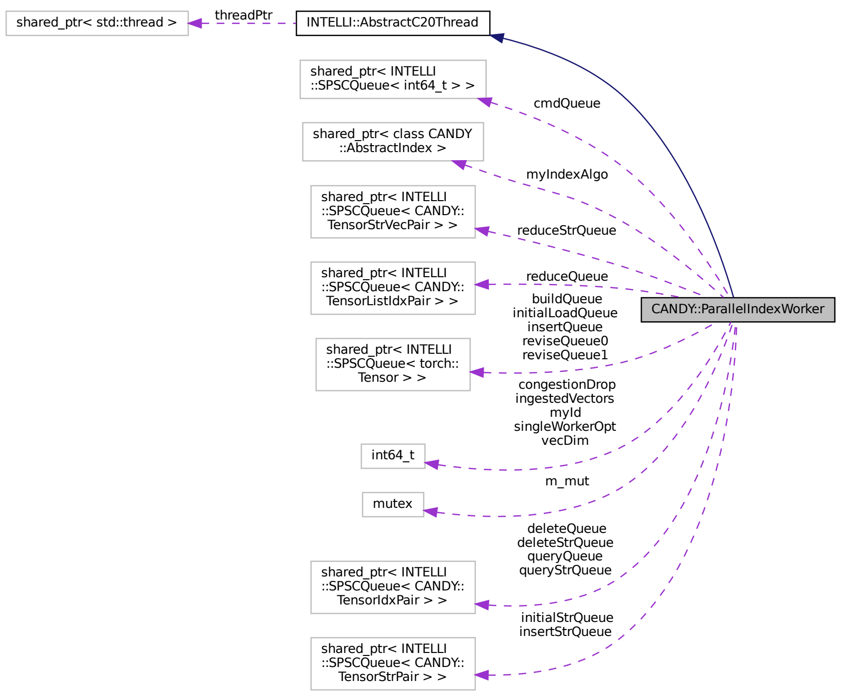
A worker class of parallel index thread.

```
#include <CANDY/ParallelPartitionIndex/ParallelIndexWorker.h>
```

Inheritance diagram for CANDY::ParallelIndexWorker:


INTELLI::AbstractC20Thread
CANDY::ParallelIndexWorker
CANDY::CongestionDropIndex Worker

Collaboration diagram for CANDY::ParallelIndexWorker:


shared_ptr< std::thread >
threadPtr
INTELLI::AbstractC20Thread
shared_ptr< INTELLI ::SPSCQueue< int64_t > >
cmdQueue
shared_ptr< class CANDY ::AbstractIndex >
myIndexAlgo
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorStrVecPair > >
reduceStrQueue
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorListIdxPair > >
reduceQueue
CANDY::ParallelIndexWorker
buildQueue
initialLoadQueue
insertQueue
reviseQueue0
reviseQueue1
shared_ptr< INTELLI ::SPSCQueue< torch:: Tensor > >
congestionDrop
ingestedVectors
myId
singleWorkerOpt
vecDim
int64_t
m_mut
mutex
deleteQueue
deleteStrQueue
queryQueue
queryStrQueue
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorIdxPair > >
initialStrQueue
insertStrQueue
shared_ptr< INTELLI ::SPSCQueue< CANDY:: TensorStrPair > >

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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

    set the index-specfic 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

8.133.1 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, I64, default 10
- vecDim the dimension of vectors, I674, default 768
- congestionDrop, whether or not drop the data when congestion occurs, I64, default 0

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

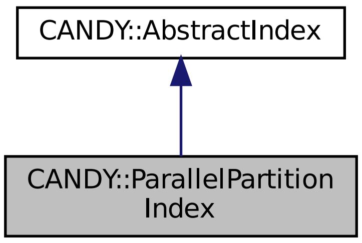
- include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h
- src/CANDY/ParallelPartitionIndex/ParallelIndexWorker.cpp

8.134 CANDY::ParallelPartitionIndex Class Reference

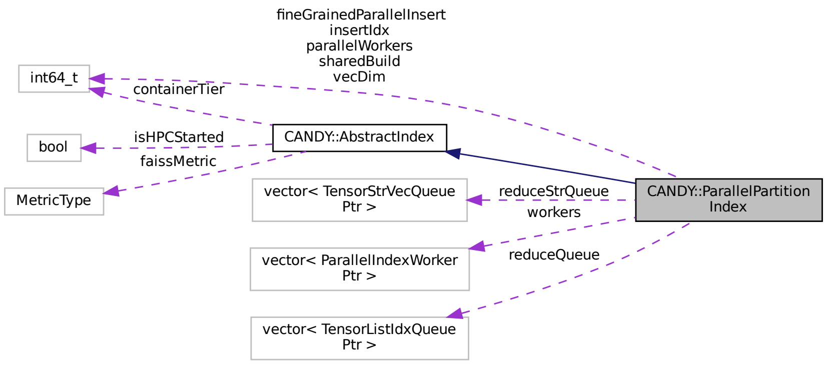
A basic parallel index, works under generic data partition, allow configurable index of threads, following round-robin insert and map-reduce query, have an optional congestion-and-drop feature.

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

Inheritance diagram for CANDY::ParallelPartitionIndex:



Collaboration diagram for CANDY::ParallelPartitionIndex:


fineGrainedParallelInsert
insertIdx
parallelWorkers
sharedBuild
vecDim
int64_t
containerTier
bool
isHPCStarted
CANDY::AbstractIndex
faissMetric
MetricType
vector< TensorStrVecQueue Ptr >
reduceStrQueue
workers
CANDY::ParallelPartition Index
vector< ParallelIndexWorker Ptr >
reduceQueue
vector< TensorListIdxQueue Ptr >

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 inited status
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  set the index-specfic 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**

8.134.1 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, I64, default 10
- parallelWorkers The number of paraller workers, I64, default 1 (set this to less than 0 will use max hardware_concurrency);
- vecDim, the dimension of vectors, default 768, I64
- fineGrainedParallelInsert, whether or not conduct the insert in an extremely fine-grained way, i.e., per-row, I64, default 0
- sharedBuild whether let all sharding using shared build, 1, I64
- congestionDrop, whether or not drop the data when congestion occurs, I64, default 0 @warnning Make sure you are using 2D tensors!

8.134.2 Member Function Documentation

8.134.2.1 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

t	the tensor, some index need to be single row
k	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.

8.134.2.2 deleteTensor()

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

delete a tensor

Parameters

t	the tensor, some index needs to be single row
k	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.

8.134.2.3 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.

8.134.2.4 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.134.2.5 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

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

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.6 insertTensor()

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

insert a tensor

Parameters

t	the tensor, some index need to be single row

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.7 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

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

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.8 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.9 offlineBuild()

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

offline build phase

Parameters

t	the tensor for offline build

Returns

whether the building is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.10 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.134.2.11 reviseTensor()

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

revise a tensor

Parameters

t	the tensor to be revised
w	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.

8.134.2.12 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.134.2.13 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

t	the tensor, allow multiple rows
k	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.

8.134.2.14 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

t	the tensor, allow multiple rows
k	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.

8.134.2.15 setConfig()

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

set the index-specfic config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.16 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	the level of frozen, 0 means freeze any online update in internal state

Returns

whether the setting is successful

Reimplemented from CANDY::AbstractIndex.

8.134.2.17 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.

8.134.2.18 waitPendingOperations()

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

a busy waitting 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;

Reimplemented from CANDY::AbstractIndex.

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

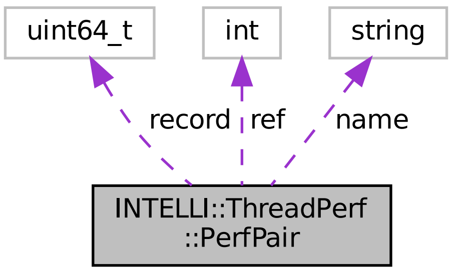
- include/CANDY/ParallelPartitionIndex.h
- src/CANDY/ParallelPartitionIndex.cpp

8.135 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**

8.135.1 Detailed Description

a record pair of perf events

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

- include/Utils/ThreadPerf.hpp

8.136 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:

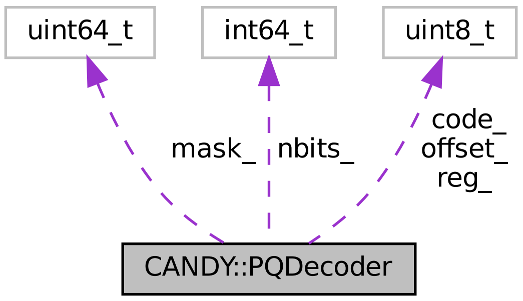
- include/Utils/ThreadPerf.hpp

8.137 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_**

8.137.1 Detailed Description

class for decoding from codes, approximated assignment of centroids, to centroids indices

8.137.2 Member Function Documentation

8.137.2.1 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:

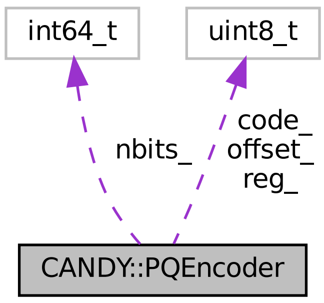
- include/CANDY/PQIndex.h

8.138 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_**

8.138.1 Detailed Description

class for encoding input vectors to codes, standing for approximated assignment of centroids

8.138.2 Member Function Documentation

8.138.2.1 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:

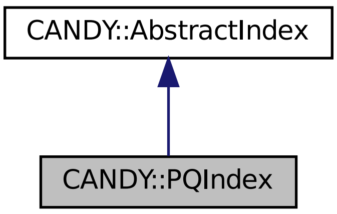
- include/CANDY/PQIndex.h

8.139 CANDY::PQIndex Class Reference

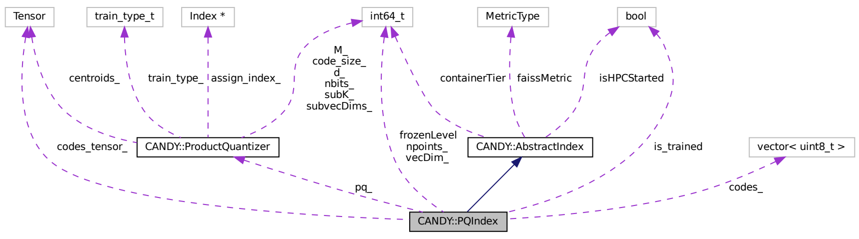
class for indexing vectors using product quantizations, this is a raw implementation without hierachical

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

Inheritance diagram for CANDY::PQIndex:



Collaboration diagram for CANDY::PQIndex:



Public Member Functions

- virtual void reset ()

    reset this index to inited 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

8.139.1 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, I64
- subQuantizers, the number of sub quantizers, default 8, I64
- nBits, the number of bits in each sub quantizer, default 8, I64

8.139.2 Member Function Documentation

8.139.2.1 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

nx	number of input x vectors
x	input vectors as tensors

8.139.2.2 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

t	the tensor, recommend single row
k	the number of nearest neighbors

Returns

bool whether the deleting is successful

Reimplemented from CANDY::AbstractIndex.

8.139.2.3 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

idx	the index, follow faiss's style, allow the KNN index of multiple queries
k	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.

8.139.2.4 insertTensor()

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

insert a tensor. In PQIndex setting it requires to re-train on new data

Parameters

t	the tensor, accept multiple rows

Returns

bool whether the insertion is successful

Reimplemented from CANDY::AbstractIndex.

8.139.2.5 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

t	the tensor, some index need to be single row

Returns

bool whether the loading is successful

Reimplemented from CANDY::AbstractIndex.

8.139.2.6 rawData()

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

return the rawData of tensor

Returns

The raw data stored in tensor

Reimplemented from CANDY::AbstractIndex.

8.139.2.7 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

t	the tensor to be revised, recommend single row
w	the revised value

Returns

bool whether the revising is successful

Reimplemented from CANDY::AbstractIndex.

8.139.2.8 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.139.2.9 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

t	the tensor, allow multiple rows
k	the returned neighbors

Returns

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

Reimplemented from CANDY::AbstractIndex.

8.139.2.10 setConfig()

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

set the index-specific config related to one index

Parameters

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

8.139.2.11 setFrozenLevel()

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

set the frozen level of online updating internal state

Parameters

frozenLv	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.

8.139.2.12 train()

```
void CANDY::PQIndex::train (
             int64_t nx,
             torch::Tensor x )  [protected]
```

train the PQIndex upon input vectors. Should be called after add()

Parameters

nx	number of input x vectors
x	input vectors as tensors

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

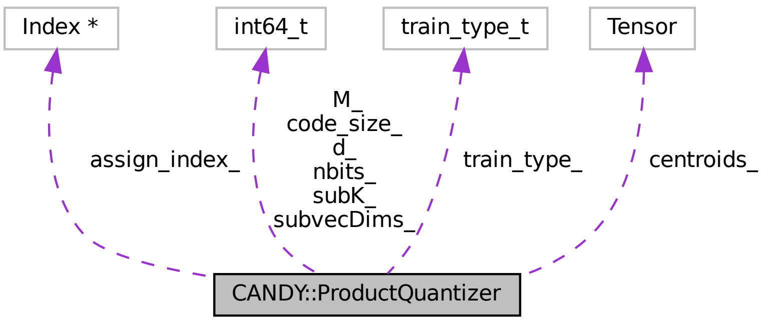
- include/CANDY/PQIndex.h
- src/CANDY/PQIndex.cpp

8.140 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_)

8.140.1 Detailed Description

class for basic product quantization operations on input of tensors

8.140.2 Member Function Documentation

8.140.2.1 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

nx	number of input vectors
x	input vectors to be added

8.140.2.2 compute_code()

```
void CANDY::ProductQuantizer::compute_code (
            const float * x,
            uint8_t * code ) const
```

compute a single vector to codes

Parameters

x	vectors to be encoded
codes	destination codes

8.140.2.3 compute_codes()

```
void CANDY::ProductQuantizer::compute_codes (
            const float * x,
            uint8_t * codes,
            int64_t n ) const
```

compute vectors to codes

Parameters

x	input vectors
codes	store computation results from x, pointer target is a torch::Tensor size of (nx, code_size_);
nx	number of input vectors
start	pointers denoting where it starts

8.140.2.4 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

x	
dis_table	

Returns

distance table tensor size of M∗subK

8.140.2.5 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

x	tensor of nx ∗ d
dis_table	output table sizeof nx∗M∗subK
nx	number of vectors

8.140.2.6 decode()

```
void CANDY::ProductQuantizer::decode (
             const torch::Tensor code,
             torch::Tensor * x ) const
```

decode from codes

Parameters

code	codes to be decoded
x	destination vectors

8.140.2.7 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

x	vectors to be searched
nx	number of input vectors
codes	codes to be consulted during search
ncodes	number of codes
res	result heap
init_finalize_heap	whether at the end of searching each vector to reorder the heap

8.140.2.8 setCentroidsFrom() [1/2]

```
void CANDY::ProductQuantizer::setCentroidsFrom (
            Clustering cls ) const
```

Set centroids from trained clustering.

Parameters

cls	Clustering

8.140.2.9 setCentroidsFrom() [2/2]

```
void CANDY::ProductQuantizer::setCentroidsFrom (
            Clustering cls,
            int64_t M ) const
```

Set centroids[M] for the Mth subquantizer from trained clustering.

Parameters

cls	Clustering
M	subquantizer identifier

8.140.2.10 train()

```
void CANDY::ProductQuantizer::train (
            int64_t n,
            torch::Tensor t )
```

train PQ with given tensor

Parameters

n	number of inputs
t	input vectors as tensor

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

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

8.141 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 &centers_length)

8.141.1 Detailed Description

The class used in hierarchical kmeans tree to choose center.

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

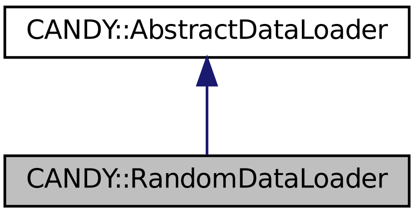
- include/CANDY/FlannIndex/FlannUtils.h

8.142 CANDY::RandomDataLoader Class Reference

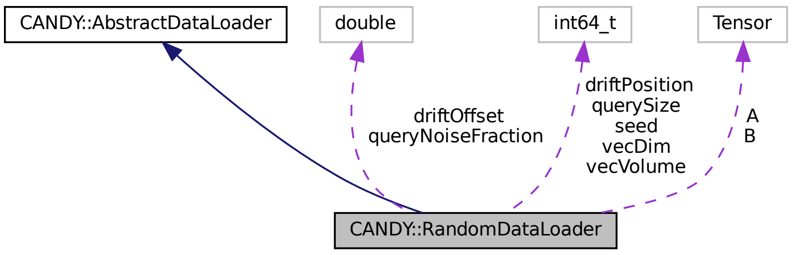
The class of ranom data loader,.

```
#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**

8.142.1 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, I64
- vecVolume, the volume of vectors, default 1000, I64
- 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 random seed, default 7758258, I64

: default name tags "random": RandomDataLoader

8.142.2 Member Function Documentation

8.142.2.1 getData()

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

get the data tensor

Returns

the generated data tensor

Reimplemented from CANDY::AbstractDataLoader.

8.142.2.2 getQuery()

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

get the query tensor

Returns

the generated query tensor

Reimplemented from CANDY::AbstractDataLoader.

8.142.2.3 setConfig()

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

Set the GLOBAL config map related to this loader.

Parameters

cfg	The config map

Returns

bool whether the config is successfully set

Note

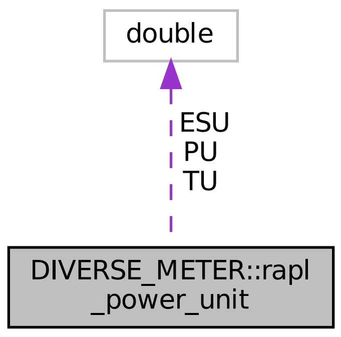
Reimplemented from CANDY::AbstractDataLoader.

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

- include/DataLoader/RandomDataLoader.h
- src/DataLoader/RandomDataLoader.cpp

8.143 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:

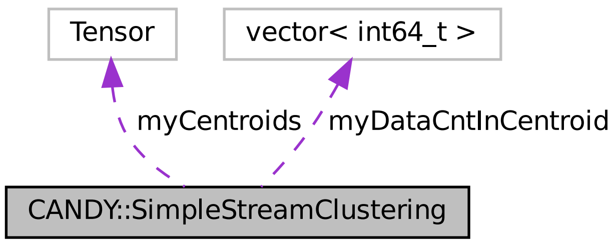
- include/CANDY/FlannIndex/FlannUtils.h

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 distance←Func=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::euclideanDis

  classify a single row of tensor
- std::vector< int64_t > classifyMultiRow (torch::Tensor &rowTensor, DistanceFunction_t distance←Func=SimpleStreamClustering::euclideanDistance)

  classify multi rows of tensor

- bool addSingleRow (torch::Tensor &rowTensor, int64_t frozenLevel=0, UpdateFunction_t insert←↩ Func=SimpleStreamClustering::euclideanInsert, DistanceFunction_t distanceFunc=SimpleStreamClustering::euclideanDistanc

  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 distance←↩ Func=SimpleStreamClustering::euclideanDistance)

  add a single row of tensor, with specifying its cluster index
- bool deleteSingleRow (torch::Tensor &rowTensor, int64_t frozenLevel=0, UpdateFunction_t delete←↩ Func=SimpleStreamClustering::euclideanDelete, DistanceFunction_t distanceFunc=SimpleStreamClustering::euclideanDistan

  delete a single row of tensor
- bool deleteSingleRowWithIdx (torch::Tensor &rowTensor, int64_t clusterIdx, int64_t frozenLevel=0, UpdateFunction_t deleteFunc=SimpleStreamClustering::euclideanInsert, DistanceFunction_t distance←↩ Func=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

rowTensor	the 1∗D row tensor
insertFunc	the insert function
frozenLevel	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

rowTensor	the 1∗D row tensor
clusterIdx	the cluster index
insertFunc	the insert function
frozenLevel	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

trainSet	the trainset, N∗D
k	the number of centroids
maxIterations	the max iteratiosn for setting up cluesters
distanceFunc	the distance function
usingCuda	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

rowsTensor	the N∗D row tensor
distanceFunc	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

rowTensor	the 1∗D row tensor
distanceFunc	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

rowTensor	the 1∗D row tensor
deleteFunc	the delete function
frozenLevel	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

rowTensor	the 1∗D row tensor
clusterIdx	the cluster index
deleteFunc	the delete function
frozenLevel	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

a	the data tensor
b	the centroids
c	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

a	the data tensor
b	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

a	the data tensor
b	the centroids
c	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

fname	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

externCentroid	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

fname	the name of file

Returns

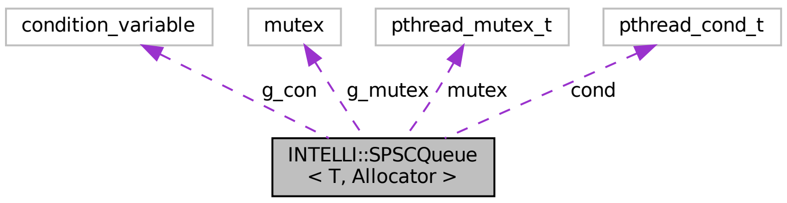
whether the saving is successful

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:

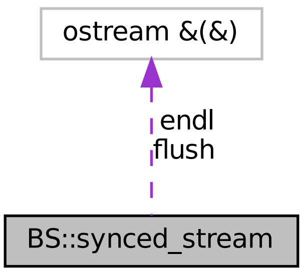
- include/Utils/SPSCQueue.hpp

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

*out_↩ stream_*	The output stream to print to. The default value is std::cout.

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

T	The types of the items

Parameters

items	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

T	The types of the items

Parameters

items	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:

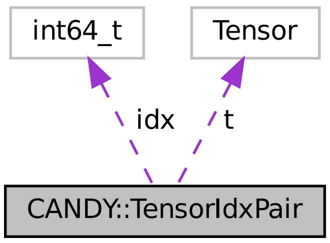
- include/Utils/BS_thread_pool.hpp

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

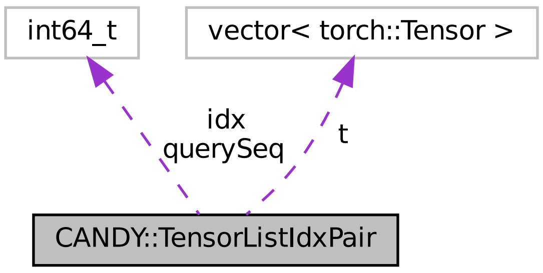
The class to define a tensor along with some idx.

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

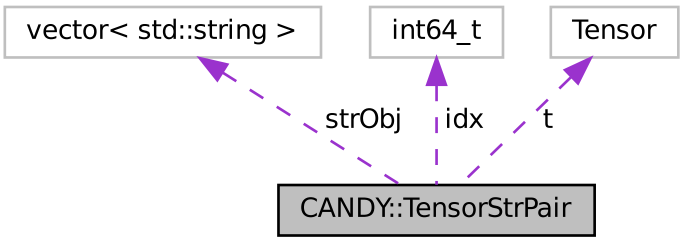
- std::vector< torch::Tensor > **t**
- int64_t **idx**
- int64_t **querySeq**

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

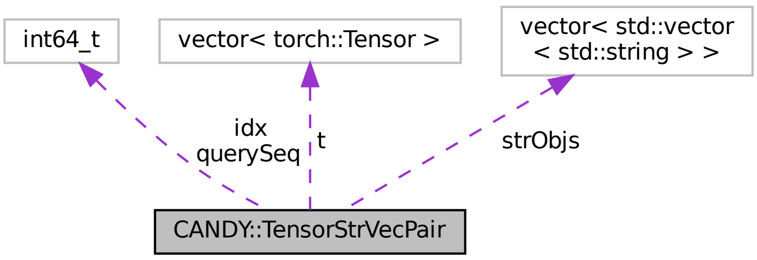
- torch::Tensor **t**
- int64_t **idx**
- std::vector< std::string > **strObj**

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 > > **strObjs**

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

*thread_↩ count_*	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

F	The type of the function to loop through.
T	The type of the loop indices. Should be a signed or unsigned integer.
R	The return value of the loop function F (can be void).

Parameters

index_after_last	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.
loop	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)".

Parameters

num_blocks	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

F	The type of the function to loop through.
T1	The type of the first index in the loop. Should be a signed or unsigned integer.
T2	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.
T	The common type of T1 and T2.
R	The return value of the loop function F (can be void).

Parameters

first_index	The first index in the loop.
index_after_last	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.
loop	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)".
num_blocks	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

F	The type of the function to loop through.
T	The type of the loop indices. Should be a signed or unsigned integer.

Parameters

index_after_last	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.
loop	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)".
num_blocks	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

F	The type of the function to loop through.
T1	The type of the first index in the loop. Should be a signed or unsigned integer.
T2	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.
T	The common type of T1 and T2.

Parameters

first_index	The first index in the loop.
index_after_last	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.
loop	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)".
num_blocks	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

F	The type of the function.
A	The types of the arguments.

Parameters

task	The function to push.
args	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

*thread_↩ count_*	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

F	The type of the function.
A	The types of the zero or more arguments to pass to the function.
R	The return type of the function (can be void).

Parameters

task	The function to submit.
args	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:

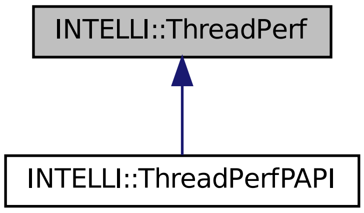
- include/Utils/BS_thread_pool.hpp

8.153 INTELLI::ThreadPerf Class Reference

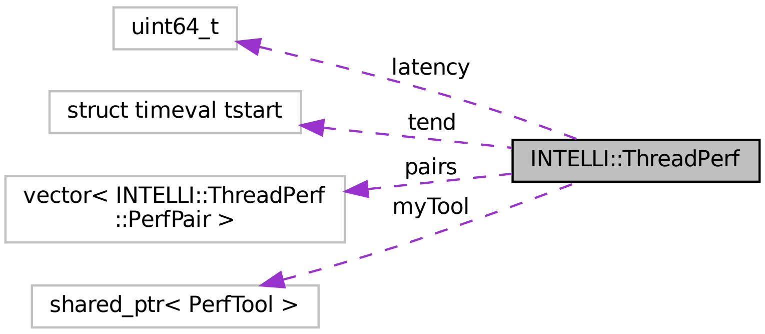
The top entity to provide perf traces, please use this class only UNLESS you know what you are doing.

```
#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 getResultById (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.
- 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

cpu	>=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

idx	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

idx	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

cfg	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:

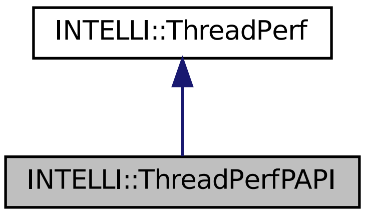
- include/Utils/ThreadPerf.hpp

8.154 INTELLI::ThreadPerfPAPI Class Reference

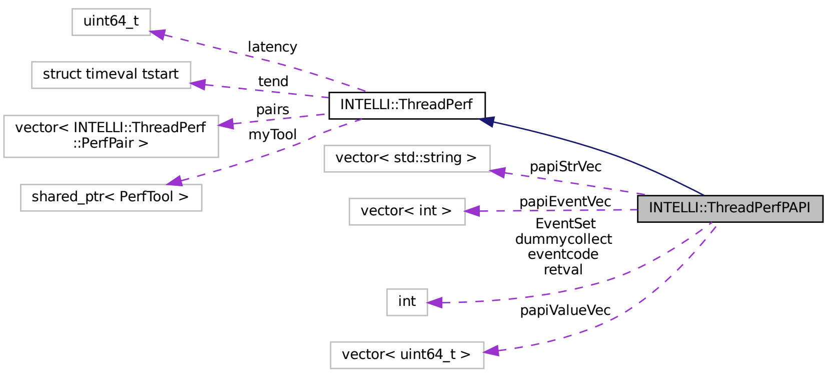
The top entity to provide perf traces by using PAPI lib.

```
#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 getResultById (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

cpu	>=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

displayTag	the tag to be displayed in your results
code	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

displayTag	the tag to be displayed in your results
papiTag	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

idx	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

idx	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

cfg	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

id	the core you plan to bind, -1 means let os decide

Returns

cpuId, 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

groundTruth	The ground truth
prob	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

fraction	the percentile in 0∼1
myTs	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

fname	the name of output file
myTs	the time stamp vector
skipZero	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

folderName	the name of folder
tensors	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

A	the list of tensors
folderName	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.

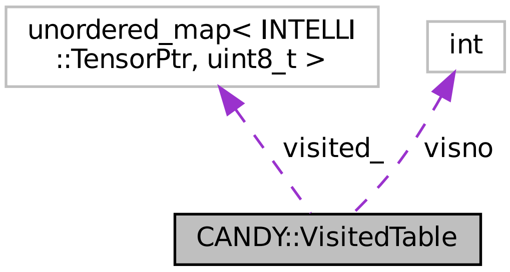
The documentation for this class was generated from the following file:

- 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

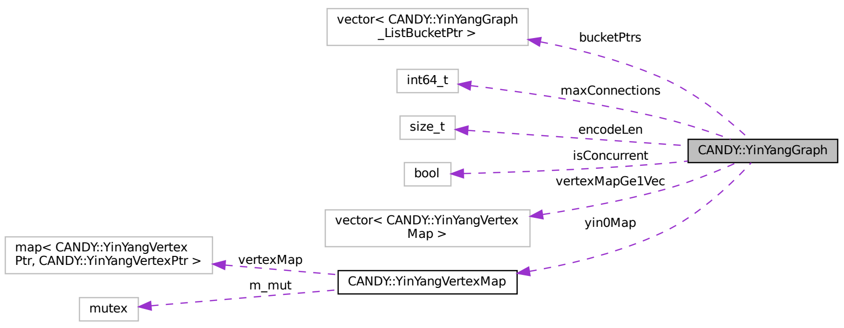
The documentation for this class was generated from the following file:

- 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 is↩ Concurrent=false)

    insert a tensor with its encode
- bool deleteTensorWithEncode (torch::Tensor &t, std::vector< uint8_t > &encode, uint64_t bktIdx, bool is↩ Concurrent=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 assit in graph navigation
- This is a hiearchical 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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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

bkts	the number of buckets
_encodeLen	the length of tensors' encoding
_maxCon	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

t	the tensor
encode	the corresponding encode
bktIdx	the index number of bucket
isConcurrent	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)

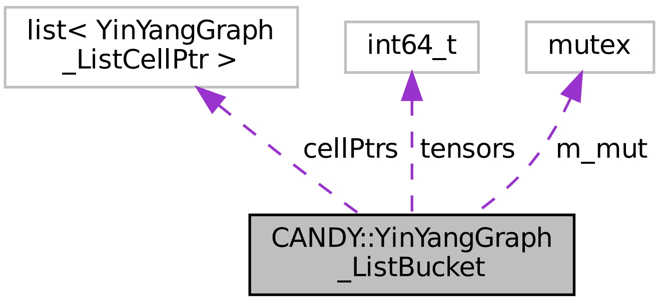
The documentation for this class was generated from the following file:

- include/CANDY/YinYangGraphIndex/YinYangGraph.h

8.164 CANDY::YinYangGraph_ListBucket Class Reference

a bucket of multiple YinYangGraph_ListCell

Collaboration diagram for CANDY::YinYangGraph_ListBucket:


list< YinYangGraph _ListCellPtr >
int64_t
mutex
cellPtrs
tensors
m_mut
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 is↩ Concurrent=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

t	the tensor
isConcurrent	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

t	the tensor
encode	the corresponding encode
isConcurrent	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 > & vertexMapGe1Vec,
            bool isConcurrent = false )
```

insert a tensor with its encode

Parameters

t	the tensor
maxNeighborCnt	
encode	the corresponding encode
yin0Map	the map of yin vertex at level 0
vertexMapGe1Vec	the vector of vertexMap in all level greater or equal to 1
isConcurrent	whether this process is concurrently executed

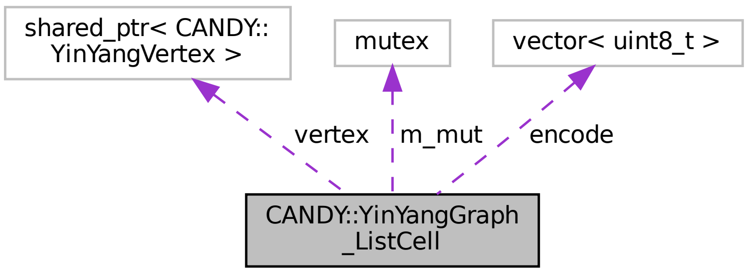
The documentation for this class was generated from the following files:

- 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

t	the tensor @returen 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

t	the tensor
maxNeighborCnt	the maximum count of neighbors
yin0Map	the map of yin vertex at level 0
vertexMapGe1Vec	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:

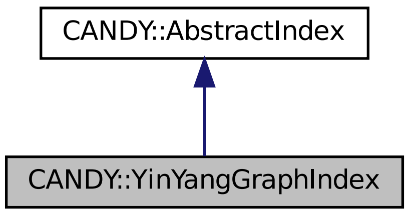
- include/CANDY/YinYangGraphIndex/YinYangGraph.h
- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.166 CANDY::YinYangGraphIndex Class Reference

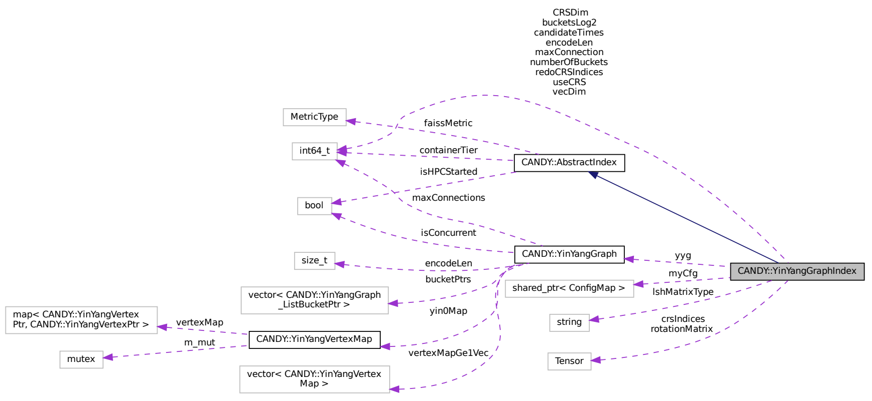
The class of indexing using a yinyang graph, first use LSH to roughly locate the range of a tensor, then search it in the linked yinyanggraph.

```
#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 **lshMatrixType** = "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, I64
- maxConnection, the max number of connections in the yinyang graph (for yang vertex of data), default 256, I64
- candidateTimes, the times of k to determine minimum candidates, default 1 ,I64
- numberOfBuckets, the number of first titer buckets, default 4096, I64, suggest 2^n
- encodeLen, the length of LSH encoding, in bytes, default 1, I64
- 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), I64
  - further trade off of accuracy v.s. efficiency
- CRSDim, the dimension which are not pruned by crs, 1/10 of vecDim, I64
- redoCRSIndices, whether or not re-generate the indices of CRS, 0 (No), I64

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

A	the left side matrix
B	the right side matrix
idx	the indices of sampling
_crsDim	the dimension of preserved dimensions

8.166.2.2 insertTensor()

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

insert a tensor

Parameters

t	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

t	the tensor, allow multiple rows
k	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

cfg	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:

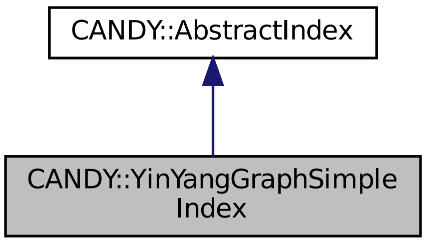
- include/CANDY/YinYangGraphIndex.h
- src/CANDY/YinYangGraphIndex.cpp

8.167 CANDY::YinYangGraphSimpleIndex Class Reference

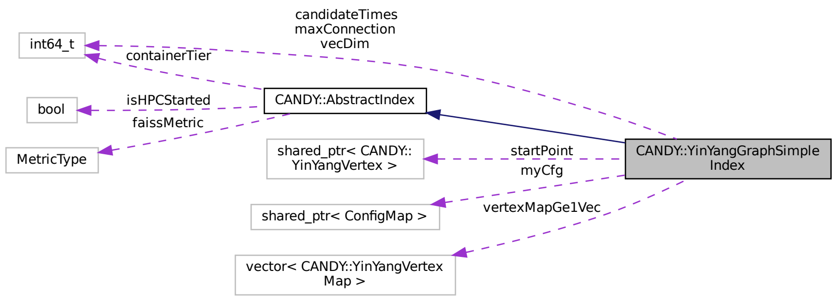
The class of indexing using a simpe yinyang graph,there is no LSH search is only within the linked yinyanggraph.

`#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, I64
- maxConnection, the max number of connections in the yinyang graph (for yang vertex of data), default 256, I64
- candidateTimes, the times of k to determine minimum candidates, default 1 ,I64
- 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

t	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

t	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

t	the tensor, allow multiple rows
k	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

cfg	the config of this class

Returns

bool whether the configuration is successful

Reimplemented from CANDY::AbstractIndex.

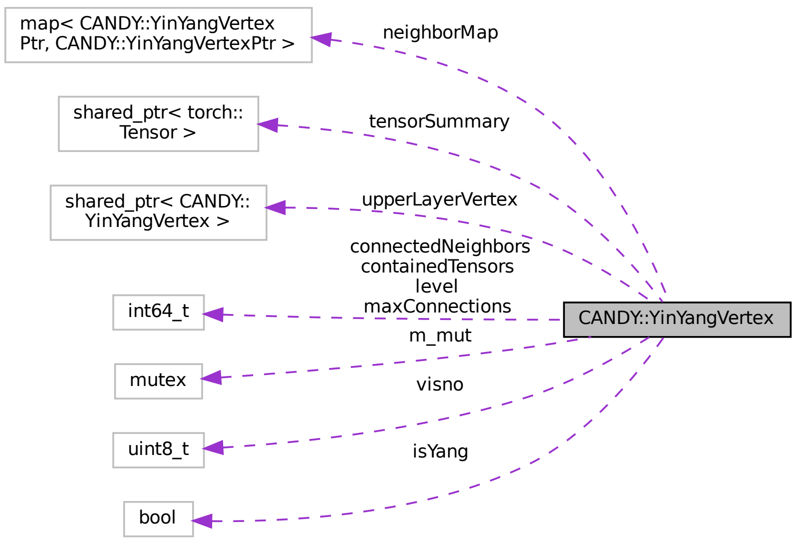
The documentation for this class was generated from the following files:

- 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 _is↩ Yang)

  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 setUperLayer (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

ts	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

ts	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

src	the source tensor to be used as reference
entryPoint	the entryPoint to start gready search
k	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

src	the source tensor
entryPoint	the entryPoint to start gready search
k	the number
ignoreYin	whether or not ignore YinVertex
forceTheSameLevel	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

src	the source vertex to be used as reference
entryPoint	the entryPoint to start gready search
k	the number
ignoreYin	whether or not ignore YinVertex
forceTheSameLevel	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

src	the source tensor to be used as reference
entryPoint	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

src	the source vertex to be used as reference
entryPoint	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

ts	the tensor linked to this vertex
_level	the level of this one
maxNumOfNeighbor	the maximum number of neighbors
_containedTensors	the number of contained tensors
_isYang	whether this is a yang vertex

8.168.2.9 setUperLayer()

```
void CANDY::YinYangVertex::setUperLayer (
            YinYangVertexPtr upv )  [inline]
```

@breif to set the upper layer vertex of this one

Parameters

upv	the upper layer vertex

8.168.2.10 toString()

```
std::string CANDY::YinYangVertex::toString (
            bool shortInfo = true )
```

@breif convert this vertex into string

Parameters

shortInfo	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

a	the new vertex
b	some existing vertex
vertexMapGe1Vec	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

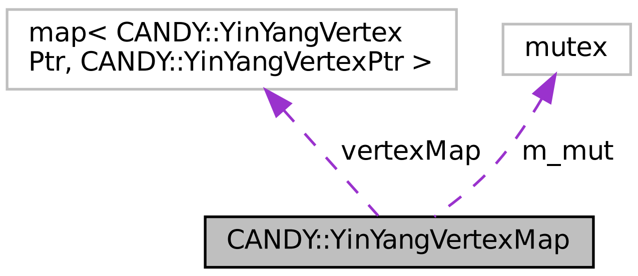
1. shrink connection of b

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

kv	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

kv	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 existis in the map.

Parameters

key	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

src	the source vertex to be used as reference
vmap,the	vertex map
k,the	number of nearest vertex to be found
df	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

src	the source vertex to be used as reference
vmap,the	vertex map
df,the	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

src	the source vertex to be used as reference

Returns

the nearest vertex

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

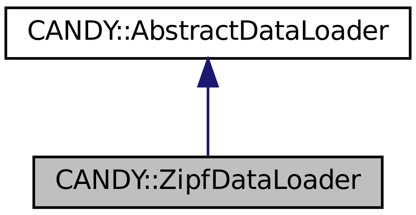
- include/CANDY/YinYangGraphIndex/YinYangGraph.h
- src/CANDY/YinYangGraphIndex/YinYangGraph.cpp

8.170 CANDY::ZipfDataLoader Class Reference

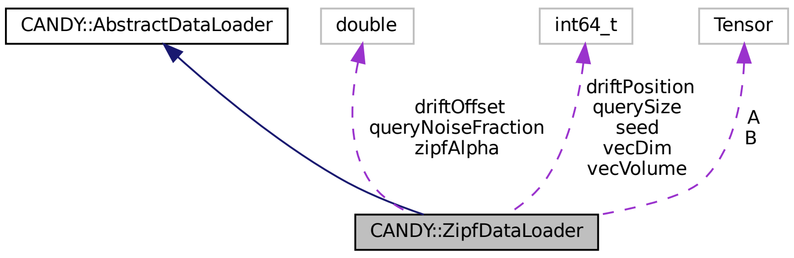
The class to load zipf data.

```
#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

cfg	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

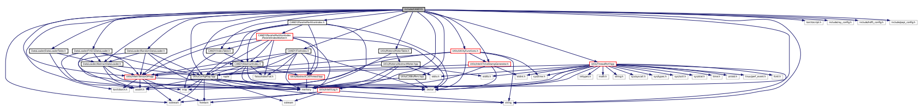
Chapter 9

File Documentation

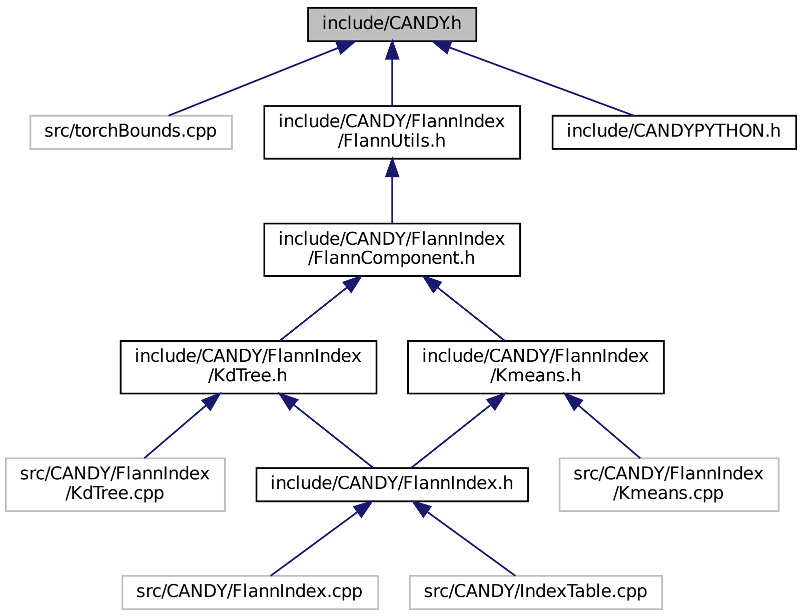
9.1 include/CANDY.h File Reference

```
#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:



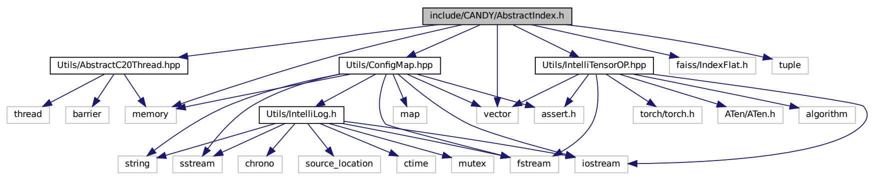
This graph shows which files directly or indirectly include this file:


include/CANDY.h
src/torchBounds.cpp
include/CANDY/FlannIndex /FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex /FlannComponent.h
include/CANDY/FlannIndex /KdTree.h
include/CANDY/FlannIndex /Kmeans.h
src/CANDY/FlannIndex /KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex /Kmeans.cpp
src/CANDY/FlannIndex.cpp
src/CANDY/IndexTable.cpp

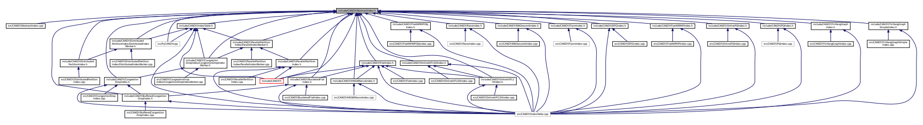
9.2 include/CANDY/AbstractIndex.h File Reference

```
#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

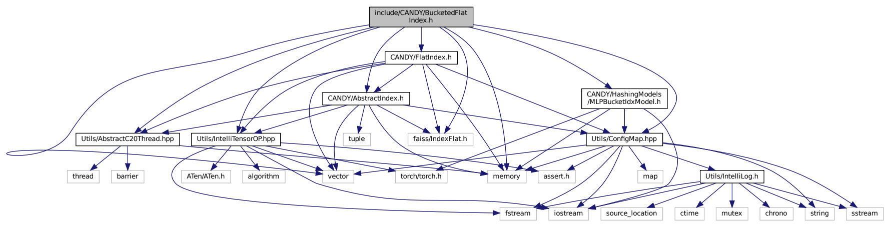
- typedef std::shared_ptr< class CANDY::AbstractIndex > CANDY::AbstractIndexPtr

  The class to describe a shared pointer to AbstractIndex.

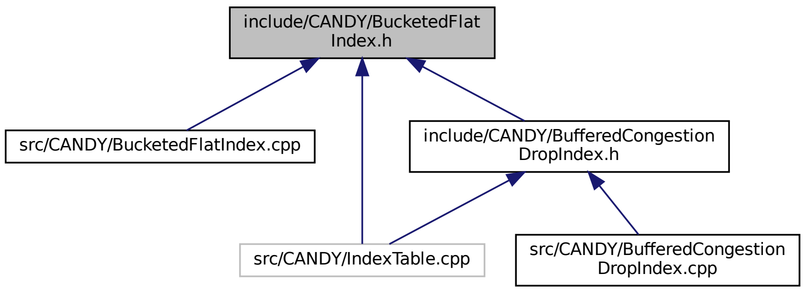
9.3 include/CANDY/BucketedFlatIndex.h File Reference

```
#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:


include/CANDY/BucketedFlat Index.h
src/CANDY/BucketedFlatIndex.cpp
include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/IndexTable.cpp
src/CANDY/BufferedCongestion DropIndex.cpp

Classes

- class CANDY::BucketedFlatIndex

  The class of splitting similar vectors into fixed number of buckets, each bucket is managed by FlatIndex.

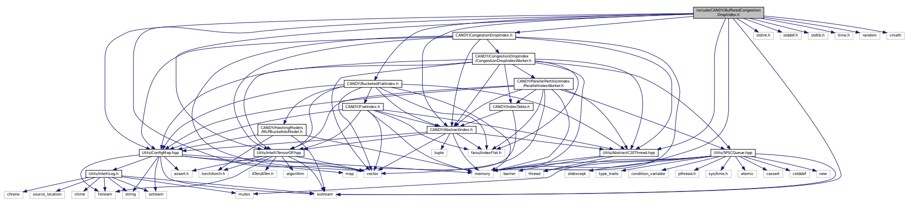
- #define newBucketedFlatIndex std::make_shared<CANDY::BucketedFlatIndex>

  (Macro) To creat a new BucketedFlatIndex shared pointer.
- typedef std::shared_ptr< class CANDY::BucketedFlatIndex > CANDY::BucketedFlatIndexPtr

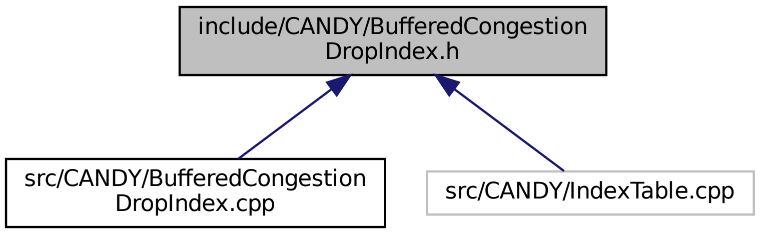
  The class to describe a shared pointer to BucketedFlatIndex.

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:


include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/BufferedCongestion DropIndex.cpp
src/CANDY/IndexTable.cpp

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>

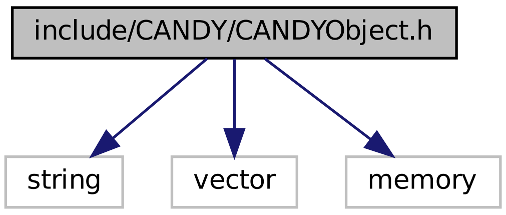
  (Macro) To creat a new BufferedCongestionDropIndex shared pointer.
- typedef std::shared_ptr< class CANDY::BufferedCongestionDropIndex > CANDY::BufferedCongestionDropIndexPtr

  The class to describe a shared pointer to BufferedCongestionDropIndex.

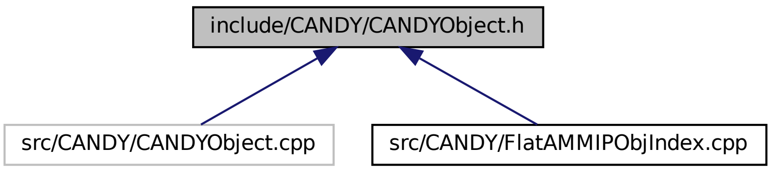
9.5 include/CANDY/CANDYObject.h File Reference

```
#include <string>
#include <vector>
#include <memory>
```

Include dependency graph for CANDYObject.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/CANDYObject.h
src/CANDY/CANDYObject.cpp
src/CANDY/FlatAMMIPObjIndex.cpp

Classes

- class CANDY::CANDYObject

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

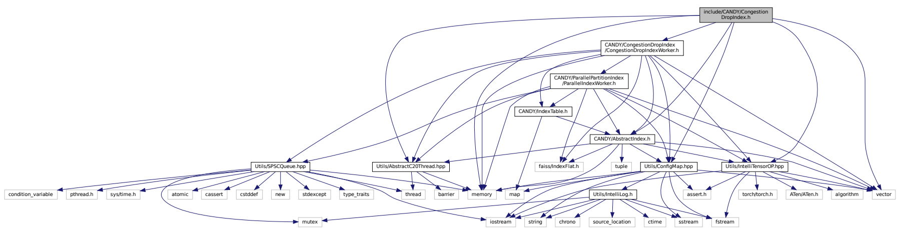
- #define **newCANDYObject** std::make_shared<CANDY::CANDYObject>
- typedef std::shared_ptr< class CANDY::CANDYObject > CANDY::CANDYObjectPtr

  The class to describe a shared pointer to CANDYObject.

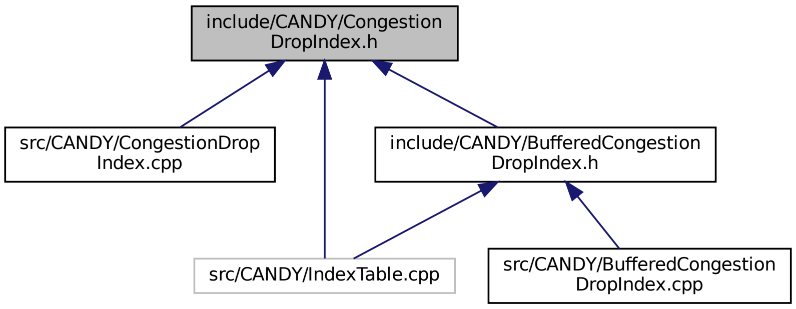
9.6 include/CANDY/CongestionDropIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/AbstractIndex.h>
#include <CANDY/CongestionDropIndex/CongestionDropIndexWorker.h>
```

Include dependency graph for CongestionDropIndex.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/Congestion DropIndex.h
src/CANDY/CongestionDrop Index.cpp
include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/IndexTable.cpp
src/CANDY/BufferedCongestion DropIndex.cpp

Classes

- 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.

- #define newCongestionDropIndex std::make_shared<CANDY::CongestionDropIndex>

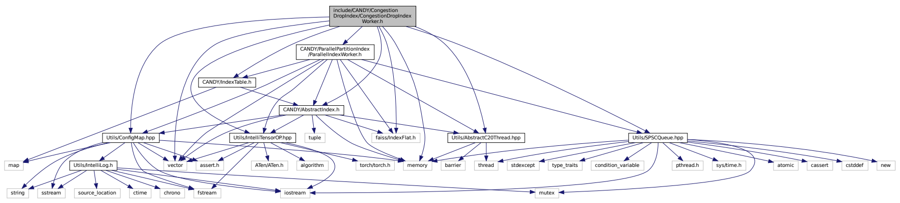
  (Macro) To creat a new CongestionDropIndex shared pointer.
- typedef std::shared_ptr< class CANDY::CongestionDropIndex > CANDY::CongestionDropIndexPtr

  The class to describe a shared pointer to CongestionDropIndex.

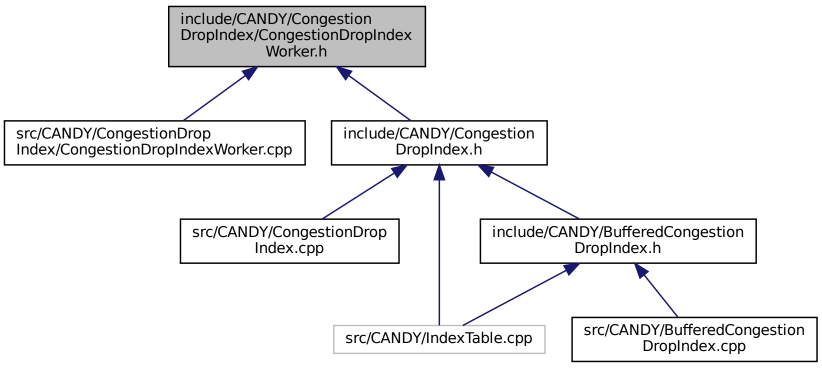
9.7 include/CANDY/CongestionDropIndex/CongestionDropIndex↩ Worker.h File Reference

```
#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:


include/CANDY/Congestion DropIndex/CongestionDropIndex Worker.h
src/CANDY/CongestionDrop Index/CongestionDropIndexWorker.cpp
include/CANDY/Congestion DropIndex.h
src/CANDY/CongestionDrop Index.cpp
include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/IndexTable.cpp
src/CANDY/BufferedCongestion DropIndex.cpp

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.

Typedefs

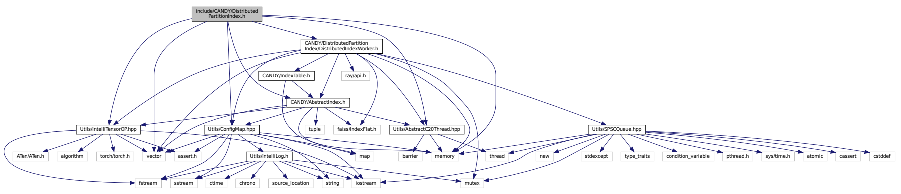
- typedef std::shared_ptr< class CANDY::CongestionDropIndexWorker > CANDY::CongestionDropIndexWorkerPtr

  The class to describe a shared pointer to CongestionDropIndexWorker.

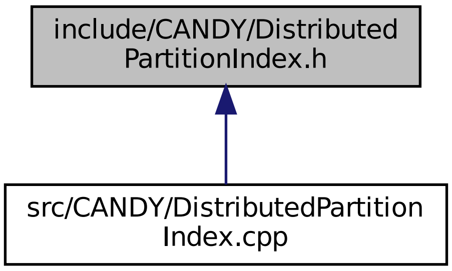
9.8 include/CANDY/DistributedPartitionIndex.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/AbstractIndex.h>
#include <CANDY/DistributedPartitionIndex/DistributedIndexWorker.h>
```

Include dependency graph for DistributedPartitionIndex.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/Distributed PartitionIndex.h
src/CANDY/DistributedPartition Index.cpp

Classes

- 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.

- #define newDistributedPartitionIndex std::make_shared<CANDY::DistributedPartitionIndex>

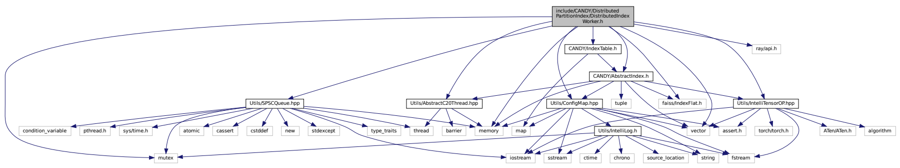
  (Macro) To creat a new DistributedPartitionIndex shared pointer.
- typedef std::shared_ptr< class CANDY::DistributedPartitionIndex > CANDY::DistributedPartitionIndexPtr

  The class to describe a shared pointer to DistributedPartitionIndex.

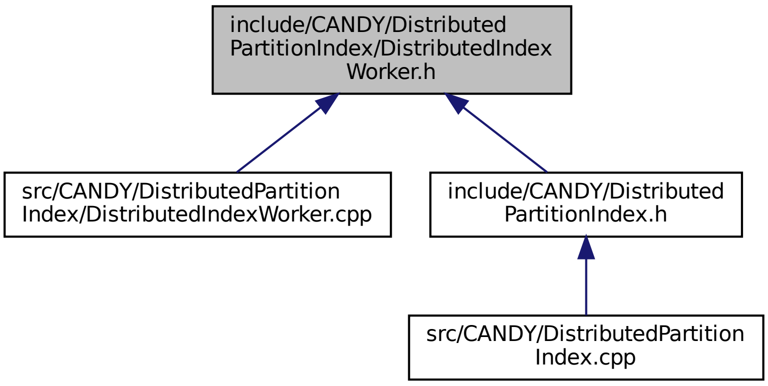
9.9 include/CANDY/DistributedPartitionIndex/DistributedIndexWorker.h File Reference

```
#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:


include/CANDY/Distributed PartitionIndex/DistributedIndex Worker.h
src/CANDY/DistributedPartition Index/DistributedIndexWorker.cpp
include/CANDY/Distributed PartitionIndex.h
src/CANDY/DistributedPartition Index.cpp

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>

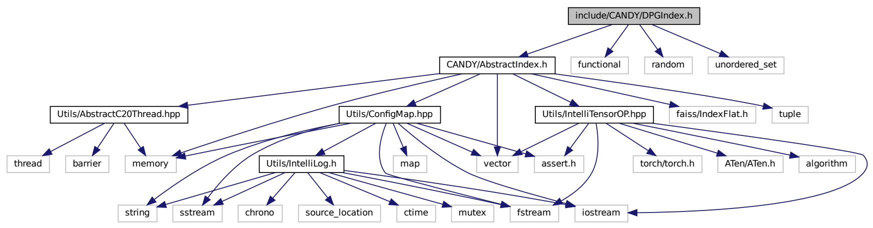
  (Macro) To creat a new DistributedIndexWorker shared pointer.
- typedef std::shared_ptr< class CANDY::DistributedIndexWorker > CANDY::DistributedIndexWorkerPtr

  The class to describe a shared pointer to DistributedIndexWorker.

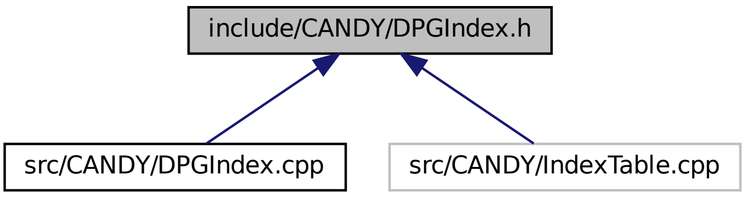
9.10 include/CANDY/DPGIndex.h File Reference

```
#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>

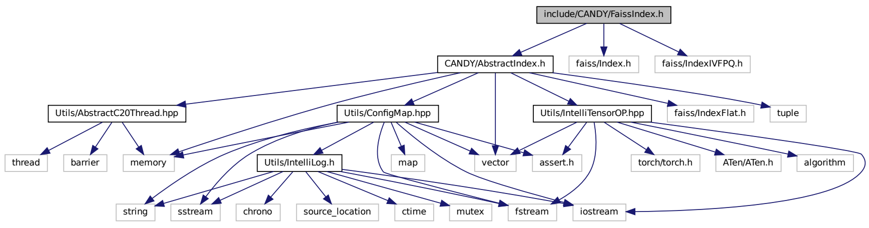
  (Macro) To creat a new DPGIndex shared pointer.
- typedef std::shared_ptr< class CANDY::DPGIndex > CANDY::DPGIndexPtr

  The class to describe a shared pointer to DPGIndex.

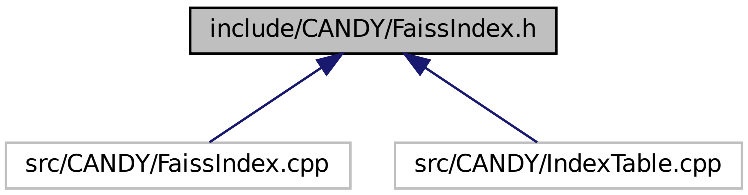
9.11 include/CANDY/FaissIndex.h File Reference

```
#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

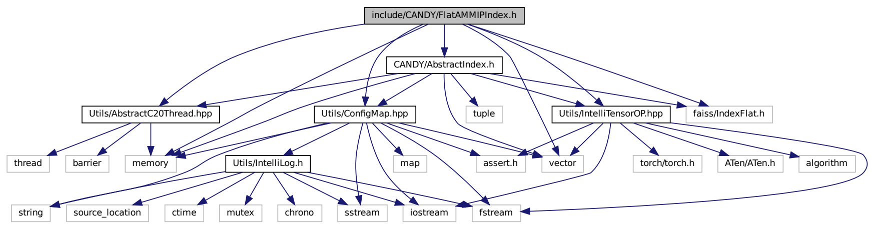
  The class to describe a shared pointer to FaissIndexPtr.

9.12 include/CANDY/FlatAMMIPIndex.h File Reference

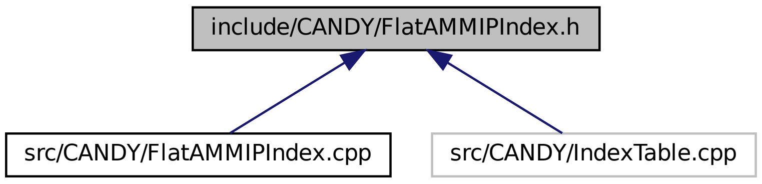
```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
```

```
#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

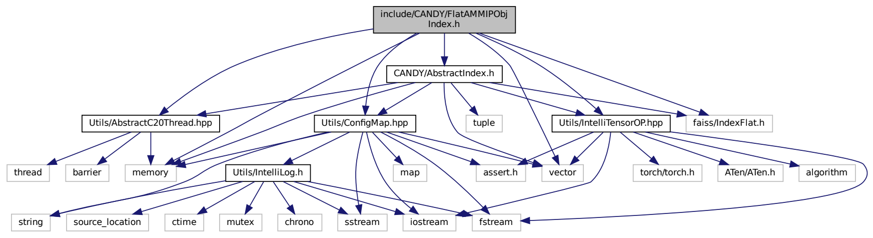
  The class to describe a shared pointer to FlatAMMIPIndex.

9.13 include/CANDY/FlatAMMIPObjIndex.h File Reference

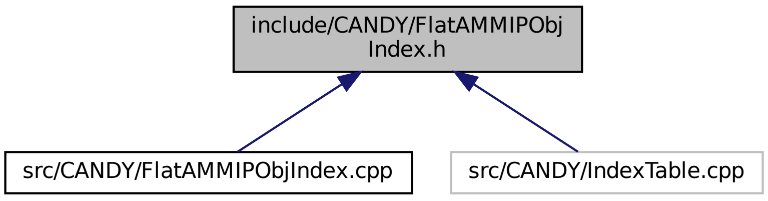
```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
```

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

Include dependency graph for FlatAMMIPObjIndex.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/FlatAMMIPObj Index.h
src/CANDY/FlatAMMIPObjIndex.cpp
src/CANDY/IndexTable.cpp

Classes

- class CANDY::FlatAMMIPObjIndex

  Similar to FlatAMMIPIndex, but additionally has object storage (currently only string)

- #define newFlatAMMIPObjIndex std::make_shared<CANDY::FlatAMMIPObjIndex>

  (Macro) To creat a new FlatAMMIPObjIndex shared pointer.
- typedef std::shared_ptr< class CANDY::FlatAMMIPObjIndex > CANDY::FlatAMMIPObjIndexPtr

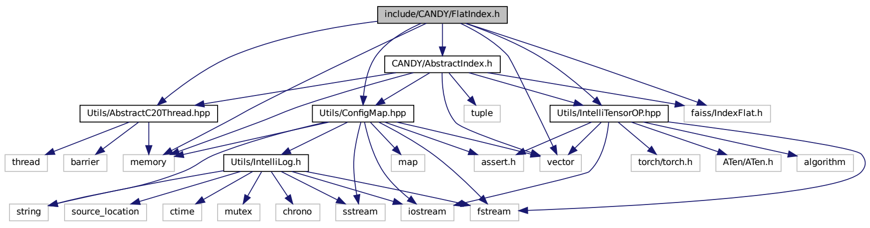
  The class to describe a shared pointer to FlatAMMIPObjIndex.

9.14 include/CANDY/FlatIndex.h File Reference

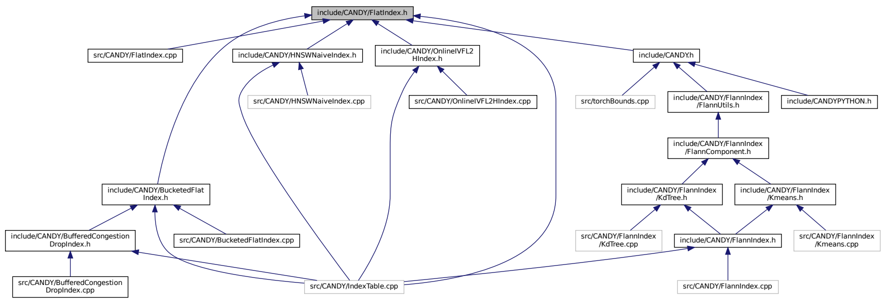
```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <faiss/IndexFlat.h>
```

`#include <CANDY/AbstractIndex.h>`

Include dependency graph for FlatIndex.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/FlatIndex.h
src/CANDY/FlatIndex.cpp
include/CANDY/HNSWNaiveIndex.h
include/CANDY/OnlineIVFL2HIndex.h
include/CANDY.h
src/CANDY/HNSWNaiveIndex.cpp
src/CANDY/OnlineIVFL2HIndex.cpp
src/torchBounds.cpp
include/CANDY/FlannIndex/FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex/FlannComponent.h
include/CANDY/BucketedFlatIndex.h
include/CANDY/FlannIndex/KdTree.h
include/CANDY/FlannIndex/Kmeans.h
include/CANDY/BufferedCongestionDropIndex.h
src/CANDY/BucketedFlatIndex.cpp
src/CANDY/FlannIndex/KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex/Kmeans.cpp
src/CANDY/BufferedCongestionDropIndex.cpp
src/CANDY/IndexTable.cpp
src/CANDY/FlannIndex.cpp

Classes

- class CANDY::FlatIndex

  The class of a flat index approach, using brutal force management.

- #define newFlatIndex std::make_shared<CANDY::FlatIndex>

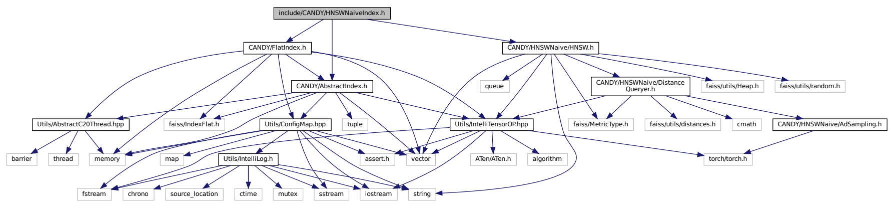
  (Macro) To creat a new FlatIndex shared pointer.
- typedef std::shared_ptr< class CANDY::FlatIndex > CANDY::FlatIndexPtr

  The class to describe a shared pointer to FlatIndex.

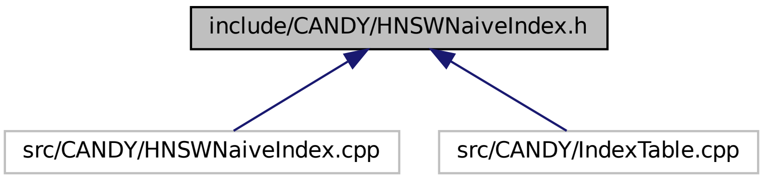
9.15 include/CANDY/HNSWNaiveIndex.h File Reference

```
#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.

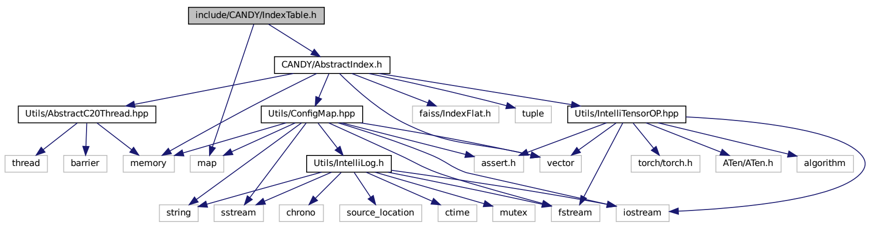
Macros

- #define **newHNSWNaiveIndex** std::make_shared<CANDY::HNSWNaiveIndex>
- #define **newNSWIndex** std::make_shared<CANDY::HNSWNaiveIndex>

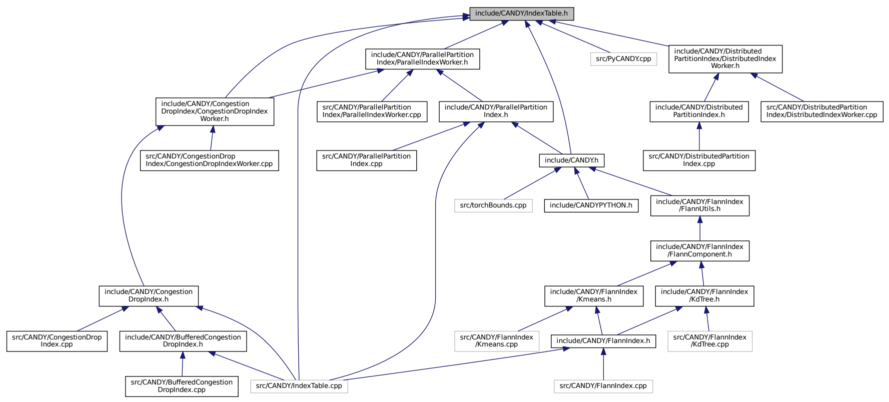
9.16 include/CANDY/IndexTable.h File Reference

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

Include dependency graph for IndexTable.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/IndexTable.h
include/CANDY/ParallelPartition Index/ParallelIndexWorker.h
src/PyCANDY.cpp
include/CANDY/Distributed PartitionIndex/DistributedIndex Worker.h
include/CANDY/Congestion DropIndex/CongestionDropIndex Worker.h
src/CANDY/ParallelPartition Index/ParallelIndexWorker.cpp
include/CANDY/ParallelPartition Index.h
include/CANDY/Distributed PartitionIndex.h
src/CANDY/DistributedPartition Index/DistributedIndexWorker.cpp
src/CANDY/CongestionDrop Index/CongestionDropIndexWorker.cpp
src/CANDY/ParallelPartition Index.cpp
include/CANDY.h
src/CANDY/DistributedPartition Index.cpp
src/torchBounds.cpp
include/CANDYPYTHON.h
include/CANDY/FlannIndex /FlannUtils.h
include/CANDY/FlannIndex /FlannComponent.h
include/CANDY/Congestion DropIndex.h
include/CANDY/FlannIndex /Kmeans.h
include/CANDY/FlannIndex /KdTree.h
src/CANDY/CongestionDrop Index.cpp
include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/FlannIndex /Kmeans.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex /KdTree.cpp
src/CANDY/BufferedCongestion DropIndex.cpp
src/CANDY/IndexTable.cpp
src/CANDY/FlannIndex.cpp

Classes

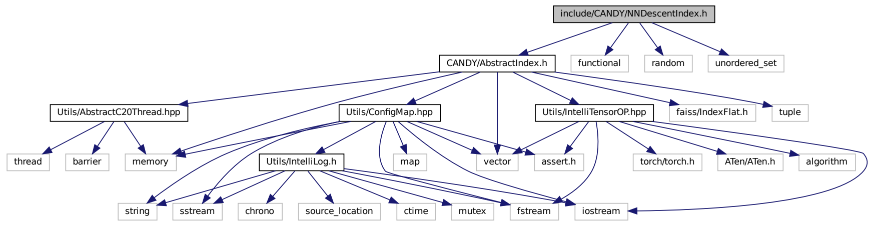
- class CANDY::IndexTable

  The table to index index algos.

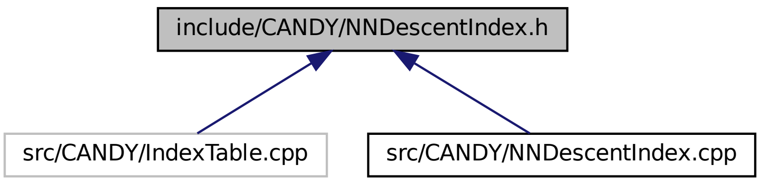
9.17 include/CANDY/NNDescentIndex.h File Reference

```
#include <CANDY/AbstractIndex.h>
#include <functional>
#include <random>
#include <unordered_set>
```

Include dependency graph for NNDescentIndex.h:



This graph shows which files directly or indirectly include this file:



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::Nhood

- #define newNNDescentIndex std::make_shared<CANDY::NNDescentIndex>

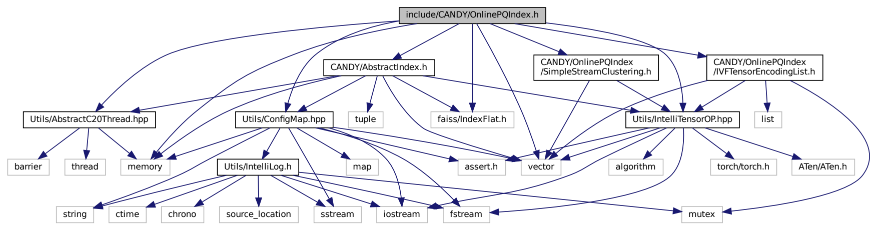
  (Macro) To creat a new NNDescentIndex shared pointer.
- typedef std::shared_ptr< class CANDY::NNDescentIndex > CANDY::NNDescentIndexPtr

  The class to describe a shared pointer to NNDescentIndex.

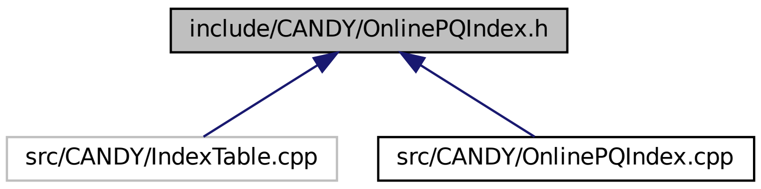
9.18 include/CANDY/OnlinePQIndex.h File Reference

```
#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.

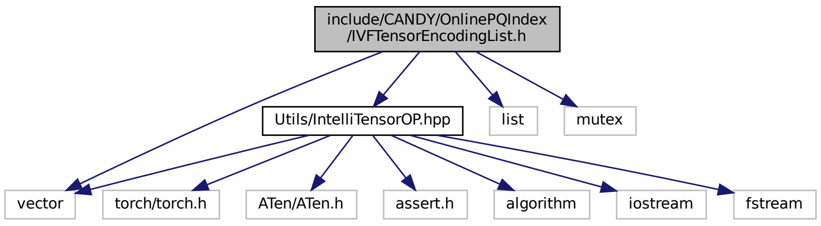
- #define newOnlinePQIndex std::make_shared<CANDY::OnlinePQIndex>

  (Macro) To creat a new OnlinePQIndex shared pointer.
- typedef std::shared_ptr< class CANDY::OnlinePQIndex > CANDY::OnlinePQIndexPtr

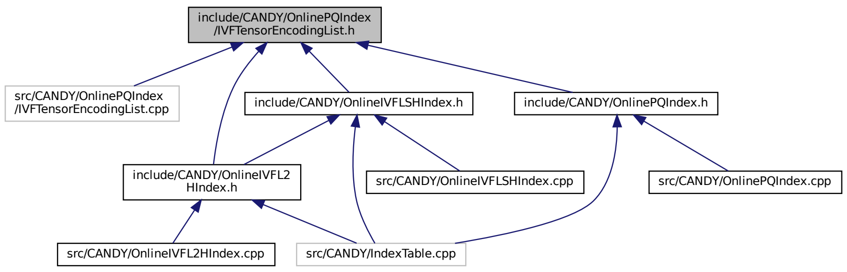
  The class to describe a shared pointer to OnlinePQIndex.

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:


include/CANDY/OnlinePQIndex
/IVFTensorEncodingList.h
src/CANDY/OnlinePQIndex
/IVFTensorEncodingList.cpp
include/CANDY/OnlineIVFLSHIndex.h
include/CANDY/OnlinePQIndex.h
include/CANDY/OnlineIVFL2
HIndex.h
src/CANDY/OnlineIVFLSHIndex.cpp
src/CANDY/OnlinePQIndex.cpp
src/CANDY/OnlineIVFL2HIndex.cpp
src/CANDY/IndexTable.cpp

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

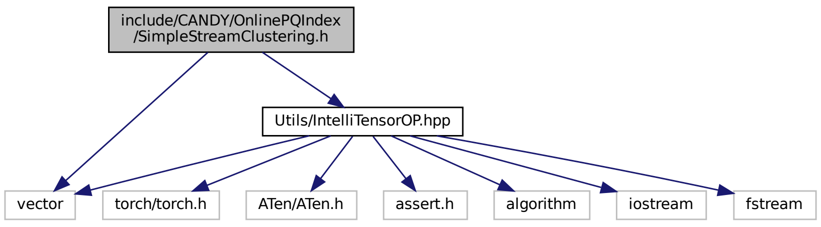
  The class to describe a shared pointer to IVFListCell.
- typedef std::shared_ptr< CANDY::IVFListBucket > CANDY::IVFListBucketPtr

  The class to describe a shared pointer to IVFListBucket.

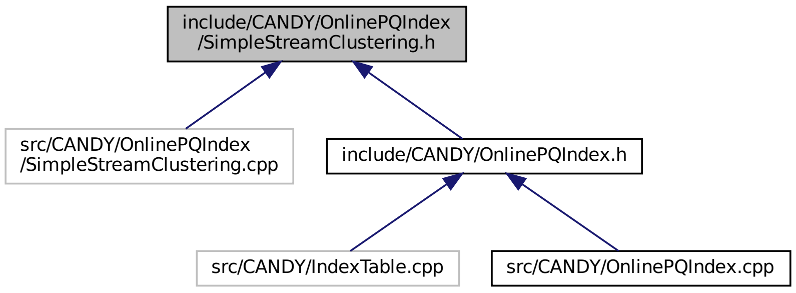
9.20 include/CANDY/OnlinePQIndex/SimpleStreamClustering.h File Reference

```
#include <Utils/IntelliTensorOP.hpp>
#include <vector>
```

Include dependency graph for SimpleStreamClustering.h:


include/CANDY/OnlinePQIndex
/SimpleStreamClustering.h
Utils/IntelliTensorOP.hpp
vector
torch/torch.h
ATen/ATen.h
assert.h
algorithm
iostream
fstream

This graph shows which files directly or indirectly include this file:


include/CANDY/OnlinePQIndex
/SimpleStreamClustering.h
src/CANDY/OnlinePQIndex
/SimpleStreamClustering.cpp
include/CANDY/OnlinePQIndex.h
src/CANDY/IndexTable.cpp
src/CANDY/OnlinePQIndex.cpp

Classes

- class CANDY::SimpleStreamClustering

  a simple class for stream clustering, following online PQ style and using simple linear equations

Typedefs

- using **CANDY::DistanceFunction_t** = torch::Tensor($*$)(const torch::Tensor &, const torch::Tensor &)
- using **CANDY::UpdateFunction_t** = void($*$)(const torch::Tensor $*$, torch::Tensor $*$, const int64_t)

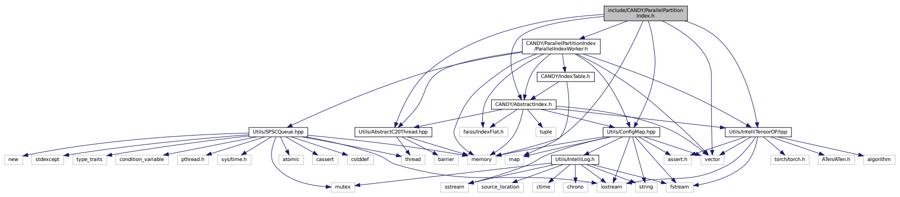
- #define newSimpleStreamClustering make_shared$<$CANDY::SimpleStreamClustering$>$

  (Macro) To creat a new SimpleStreamClustering under shared pointer.
- typedef std::shared_ptr$<$ CANDY::SimpleStreamClustering $>$ CANDY::SimpleStreamClusteringPtr

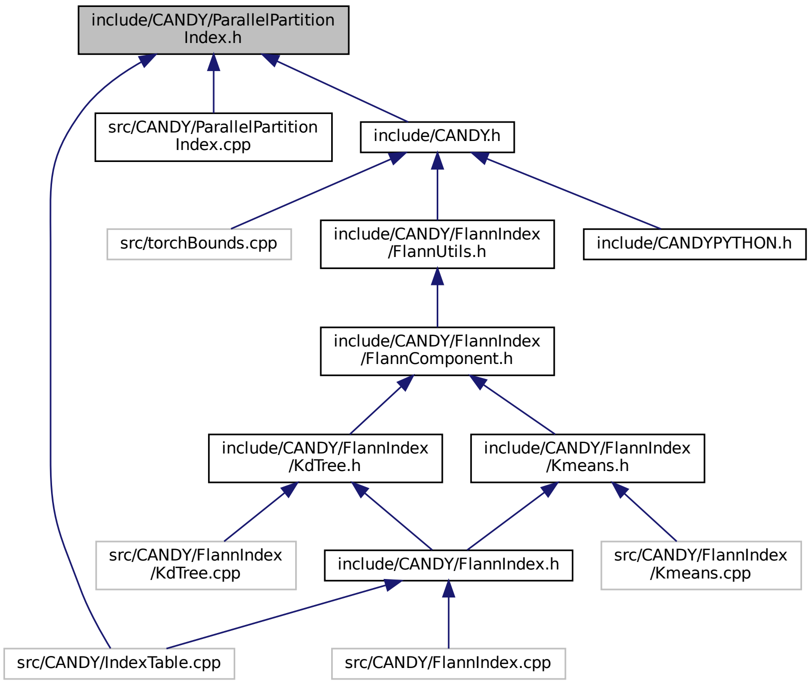
  The class to describe a shared pointer to SimpleStreamClustering.

9.21 include/CANDY/ParallelPartitionIndex.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/ParallelPartitionIndex/ParallelIndexWorker.h>
```
Include dependency graph for ParallelPartitionIndex.h:



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

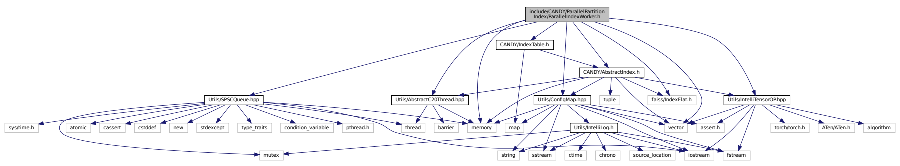
  The class to describe a shared pointer to ParallelPartitionIndex.

9.22 include/CANDY/ParallelPartitionIndex/ParallelIndexWorker.h File Reference

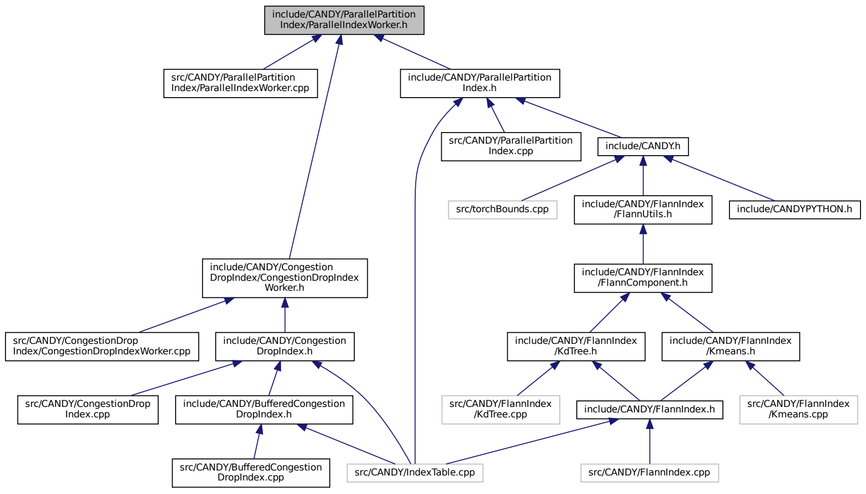
```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <memory>
#include <vector>
#include <Utils/IntelliTensorOP.hpp>
#include <CANDY/IndexTable.h>
#include <Utils/SPSCQueue.hpp>
#include <CANDY/AbstractIndex.h>
```

```
#include <faiss/IndexFlat.h>
```

Include dependency graph for ParallelIndexWorker.h:



This graph shows which files directly or indirectly include this file:


include/CANDY/ParallelPartition Index/ParallelIndexWorker.h
src/CANDY/ParallelPartition Index/ParallelIndexWorker.cpp
include/CANDY/ParallelPartition Index.h
src/CANDY/ParallelPartition Index.cpp
include/CANDY.h
src/torchBounds.cpp
include/CANDY/FlannIndex /FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/Congestion DropIndex/CongestionDropIndex Worker.h
include/CANDY/FlannIndex /FlannComponent.h
src/CANDY/CongestionDrop Index/CongestionDropIndexWorker.cpp
include/CANDY/Congestion DropIndex.h
include/CANDY/FlannIndex /KdTree.h
include/CANDY/FlannIndex /Kmeans.h
src/CANDY/CongestionDrop Index.cpp
include/CANDY/BufferedCongestion DropIndex.h
src/CANDY/FlannIndex /KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex /Kmeans.cpp
src/CANDY/BufferedCongestion DropIndex.cpp
src/CANDY/IndexTable.cpp
src/CANDY/FlannIndex.cpp

Classes

- class CANDY::TensorIdxPair

  The class to define a tensor along with some idx.
- class CANDY::TensorListIdxPair
- class CANDY::TensorStrPair
- class CANDY::TensorStrVecPair
- class CANDY::ParallelIndexWorker

  A worker class of parallel index thread.

Macros

- #define newParallelIndexWorker std::make_shared<CANDY::ParallelIndexWorker>

  (Macro) To creat a new ParallelIndexWorker shared pointer.

Typedefs

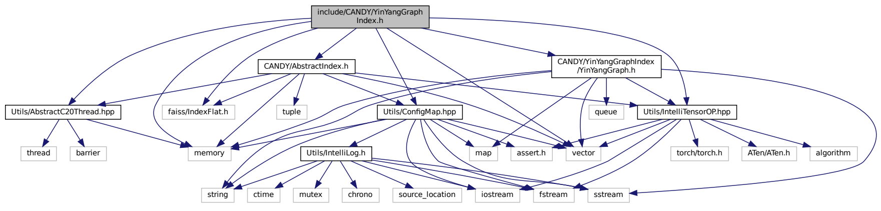
- typedef std::shared_ptr< INTELLI::SPSCQueue< torch::Tensor > > **CANDY::TensorQueuePtr**
- typedef std::shared_ptr< INTELLI::SPSCQueue< CANDY::TensorIdxPair > > **CANDY::TensorIdx↩ QueuePtr**
- typedef std::shared_ptr< INTELLI::SPSCQueue< CANDY::TensorListIdxPair > > **CANDY::TensorList↩ IdxQueuePtr**
- typedef std::shared_ptr< INTELLI::SPSCQueue< int64_t > > **CANDY::CmdQueuePtr**
- typedef std::shared_ptr< INTELLI::SPSCQueue< CANDY::TensorStrPair > > **CANDY::TensorStrQueue↩ Ptr**
- typedef std::shared_ptr< INTELLI::SPSCQueue< CANDY::TensorStrVecPair > > **CANDY::TensorStr↩ VecQueuePtr**
- typedef std::shared_ptr< class CANDY::ParallelIndexWorker > CANDY::ParallelIndexWorkerPtr

  The class to describe a shared pointer to ParallelIndexWorker.

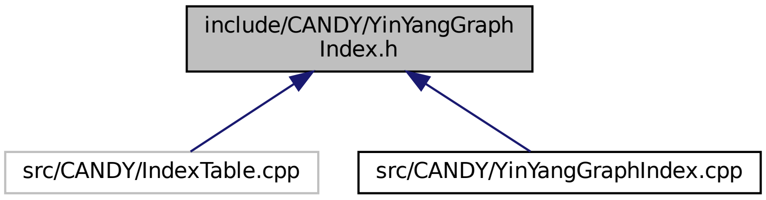
9.23 include/CANDY/YinYangGraphIndex.h File Reference

```
#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:


include/CANDY/YinYangGraph Index.h
src/CANDY/IndexTable.cpp
src/CANDY/YinYangGraphIndex.cpp

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>

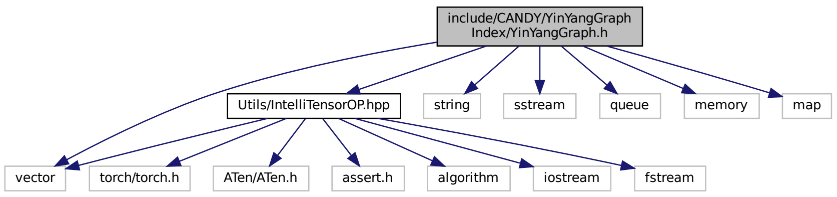
  (Macro) To creat a new YinYangGraphIndex shared pointer.
- typedef std::shared_ptr< class CANDY::YinYangGraphIndex > CANDY::YinYangGraphIndexPtr

  The class to describe a shared pointer to YinYangGraphIndex.

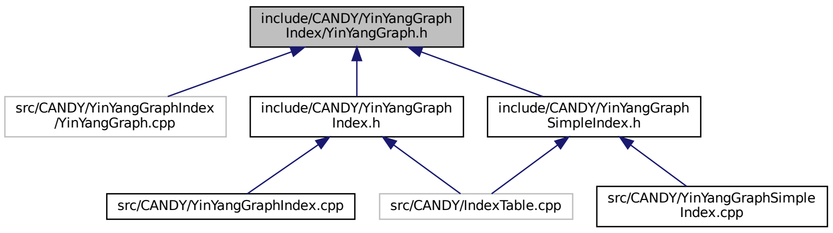
9.24 include/CANDY/YinYangGraphIndex/YinYangGraph.h File Reference

```
#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:


include/CANDY/YinYangGraph Index/YinYangGraph.h
src/CANDY/YinYangGraphIndex /YinYangGraph.cpp
include/CANDY/YinYangGraph Index.h
include/CANDY/YinYangGraph SimpleIndex.h
src/CANDY/YinYangGraphIndex.cpp
src/CANDY/IndexTable.cpp
src/CANDY/YinYangGraphSimple Index.cpp

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

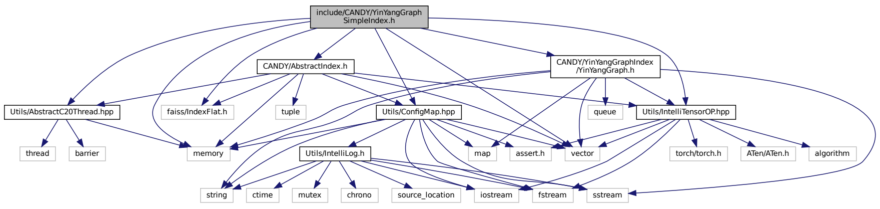
  The class to describe a shared pointer to YinYangGraph_ListCell.
- typedef std::shared_ptr< CANDY::YinYangGraph_ListBucket > CANDY::YinYangGraph_ListBucketPtr

  The class to describe a shared pointer to YinYangGraph_ListBucket.

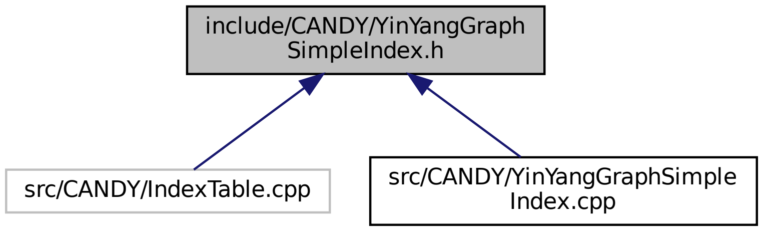
9.25 include/CANDY/YinYangGraphSimpleIndex.h File Reference

```
#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:


include/CANDY/YinYangGraph SimpleIndex.h
src/CANDY/IndexTable.cpp
src/CANDY/YinYangGraphSimple Index.cpp

Classes

- class CANDY::YinYangGraphSimpleIndex

  The class of indexing using a simpe yinyang graph,there is no LSH search is only within the linked yinyanggraph.

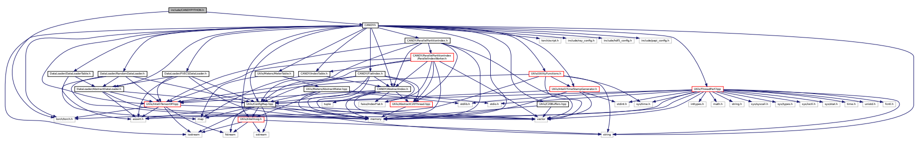
- #define newYinYangGraphSimpleIndex std::make_shared<CANDY::YinYangGraphSimpleIndex>

  (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.

9.26 include/CANDYPYTHON.h File Reference

```
#include <CANDY.h>
#include <torch/torch.h>
```
Include dependency graph for CANDYPYTHON.h:



Classes

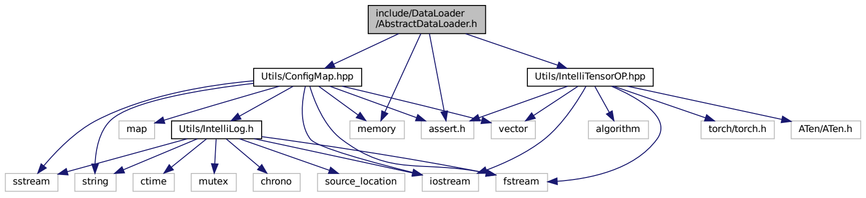
- class CANDY::Candy_Python

  The python bounding functions.

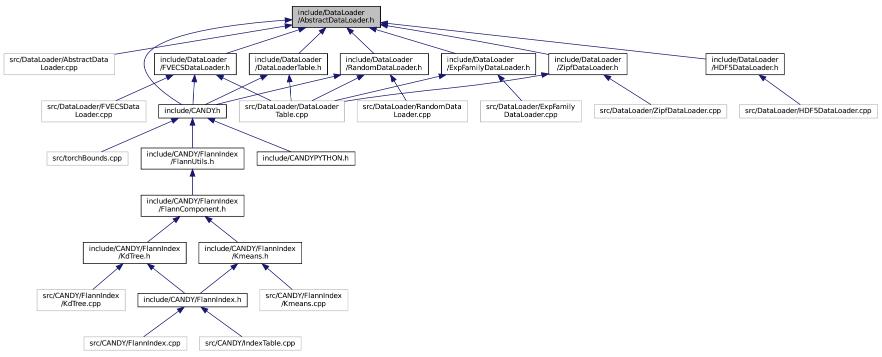
9.27 include/DataLoader/AbstractDataLoader.h File Reference

```
#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:


include/DataLoader/AbstractDataLoader.h
src/DataLoader/AbstractDataLoader.cpp
include/DataLoader/FVECSDataLoader.h
include/DataLoader/DataLoaderTable.h
include/DataLoader/RandomDataLoader.h
include/DataLoader/ExpFamilyDataLoader.h
include/DataLoader/ZipfDataLoader.h
include/DataLoader/HDF5DataLoader.h
src/DataLoader/FVECSDataLoader.cpp
include/CANDY.h
src/DataLoader/DataLoaderTable.cpp
src/DataLoader/RandomDataLoader.cpp
src/DataLoader/ExpFamilyDataLoader.cpp
src/DataLoader/ZipfDataLoader.cpp
src/DataLoader/HDF5DataLoader.cpp
src/torchBounds.cpp
include/CANDY/FlannIndex/FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex/FlannComponent.h
include/CANDY/FlannIndex/KdTree.h
include/CANDY/FlannIndex/Kmeans.h
src/CANDY/FlannIndex/KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex/Kmeans.cpp
src/CANDY/FlannIndex.cpp
src/CANDY/IndexTable.cpp

Classes

- class CANDY::AbstractDataLoader

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

- #define newAbstractDataLoader std::make_shared<CANDY::AbstractDataLoader>

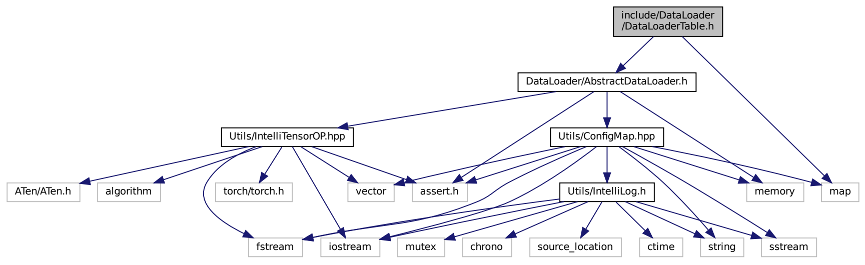
    (Macro) To creat a new AbstractDataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::AbstractDataLoader > CANDY::AbstractDataLoaderPtr

    The class to describe a shared pointer to AbstractDataLoader.

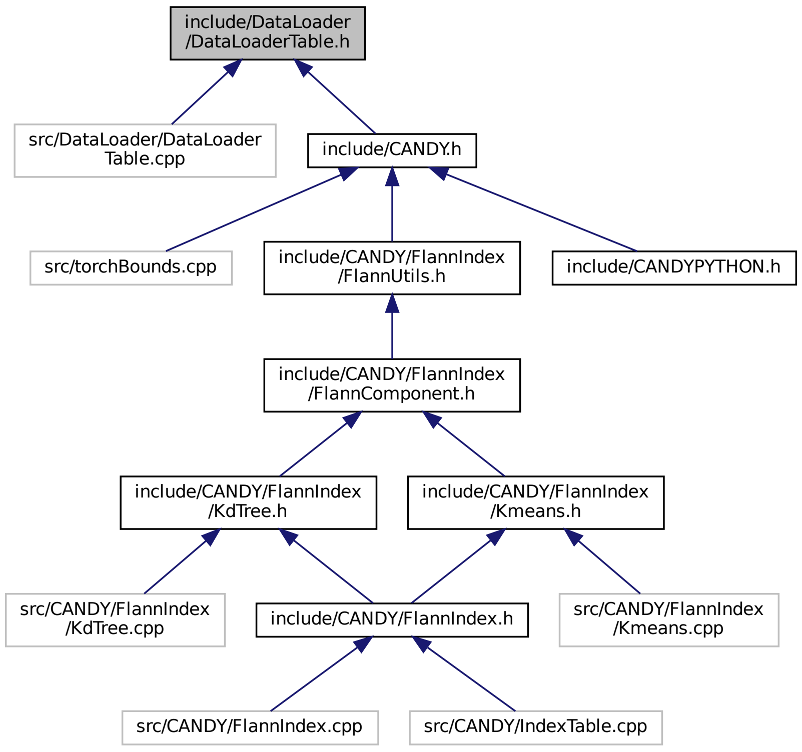
9.28 include/DataLoader/DataLoaderTable.h File Reference

```
#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

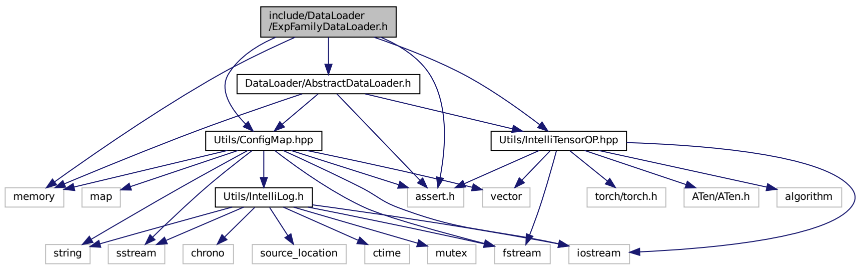
- #define newDataLoaderTable std::make_shared<CANDY::DataLoaderTable>

    (Macro) To creat a new DataLoaderTable under shared pointer.

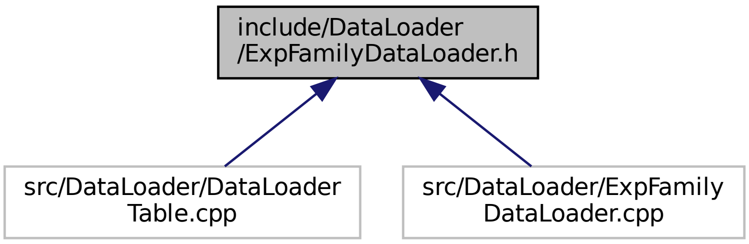
9.29 include/DataLoader/ExpFamilyDataLoader.h File Reference

```
#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>

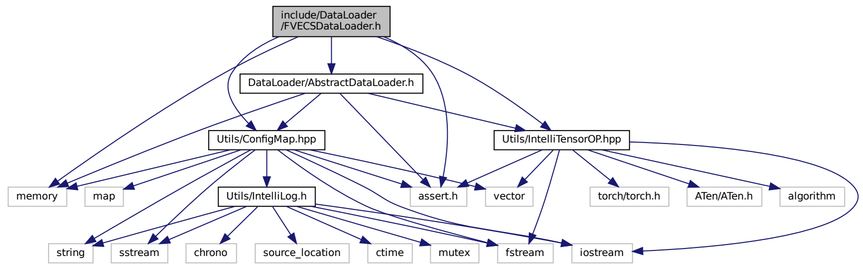
  (Macro) To creat a new ExpFamilyDataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::ExpFamilyDataLoader > CANDY::ExpFamilyDataLoaderPtr

  The class to describe a shared pointer to ExpFamilyDataLoader.

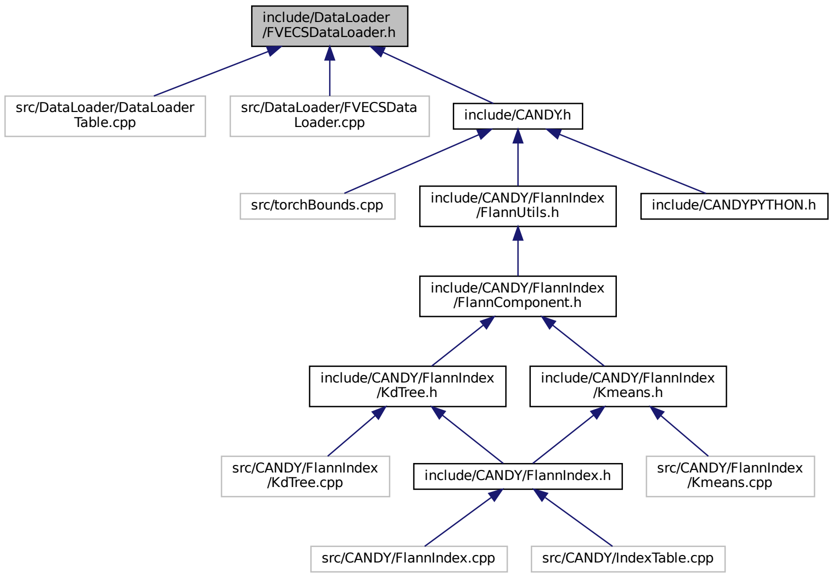
9.30 include/DataLoader/FVECSDataLoader.h File Reference

```
#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>

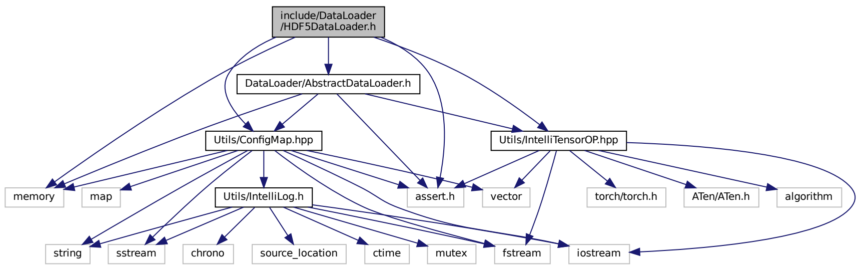
  (Macro) To creat a new FVECSDataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::FVECSDataLoader > CANDY::FVECSDataLoaderPtr

  The class to describe a shared pointer to FVECSDataLoader.

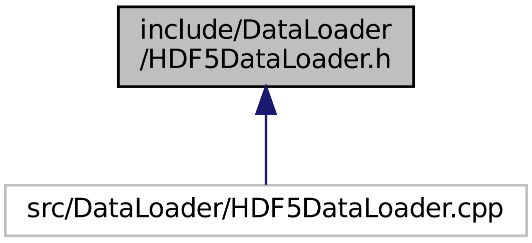
9.31 include/DataLoader/HDF5DataLoader.h File Reference

```
#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:


include/DataLoader/HDF5DataLoader.h
src/DataLoader/HDF5DataLoader.cpp

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>

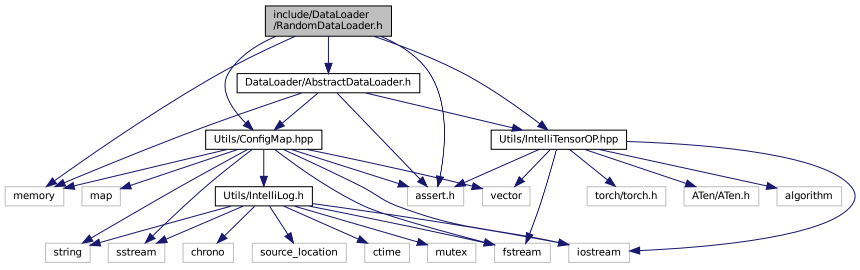
  (Macro) To creat a new HDF5DataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::HDF5DataLoader > CANDY::HDF5DataLoaderPtr

  The class to describe a shared pointer to HDF5DataLoader.

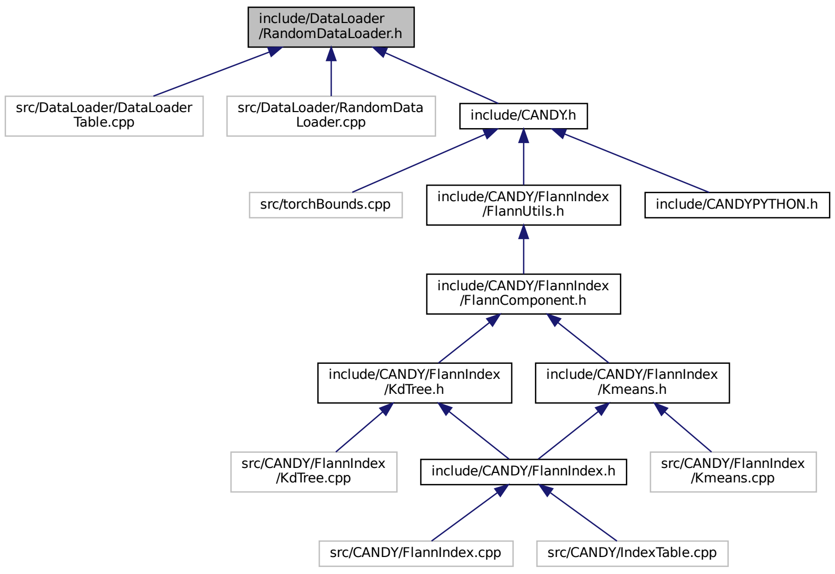
9.32 include/DataLoader/RandomDataLoader.h File Reference

```
#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 ranom data loader,.

- #define newRandomDataLoader std::make_shared<CANDY::RandomDataLoader>

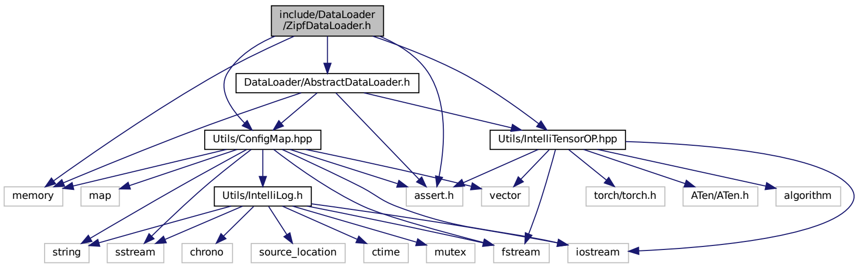
  (Macro) To creat a new RandomDataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::RandomDataLoader > CANDY::RandomDataLoaderPtr

  The class to describe a shared pointer to RandomDataLoader.

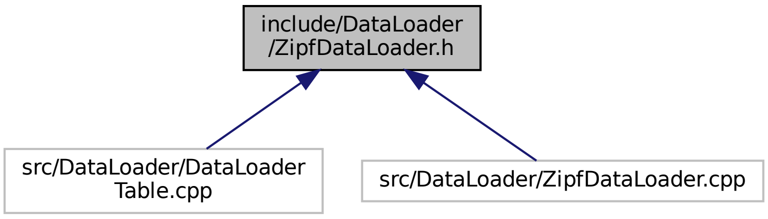
9.33 include/DataLoader/ZipfDataLoader.h File Reference

```
#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:


include/DataLoader/ZipfDataLoader.h
src/DataLoader/DataLoaderTable.cpp
src/DataLoader/ZipfDataLoader.cpp

Classes

- class CANDY::ZipfDataLoader

  The class to load zipf data.

- #define newZipfDataLoader std::make_shared<CANDY::ZipfDataLoader>

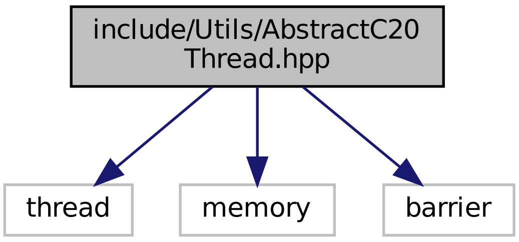
  (Macro) To creat a new ZipfDataLoader under shared pointer.
- typedef std::shared_ptr< class CANDY::ZipfDataLoader > CANDY::ZipfDataLoaderPtr

  The class to describe a shared pointer to ZipfDataLoader.

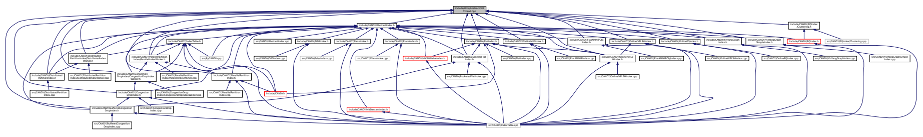
9.34 include/Utils/AbstractC20Thread.hpp File Reference

```
#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

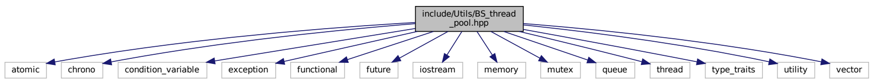
  The class to describe a shared pointer to AbstractC20Thread.
- typedef std::shared_ptr< std::barrier<> > **INTELLI::BarrierPtr**

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:synced←_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::synced_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:synced↩
_stream, and BS::timer.

Author

Barak Shoshany (`baraksh@gmail.com`) (`http://baraksh.com`)

Version

3.3.0

Date

2022-08-03

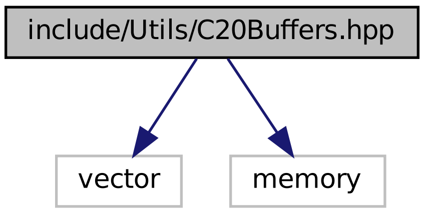
Copyright

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)

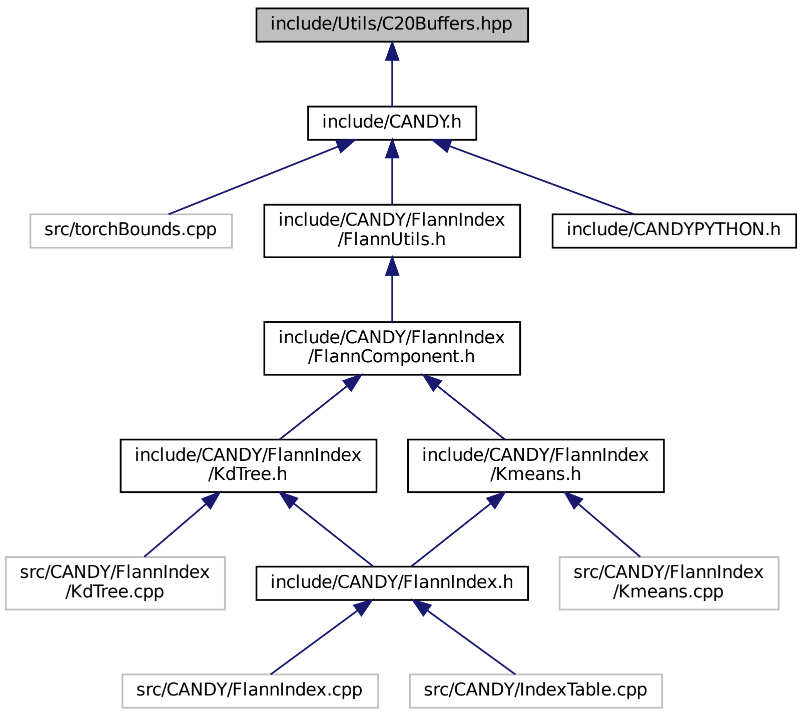
9.36 include/Utils/C20Buffers.hpp File Reference

```
#include <vector>
#include <memory>
```

Include dependency graph for C20Buffers.hpp:



This graph shows which files directly or indirectly include this file:


include/Utils/C20Buffers.hpp
include/CANDY.h
src/torchBounds.cpp
include/CANDY/FlannIndex/FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex/FlannComponent.h
include/CANDY/FlannIndex/KdTree.h
include/CANDY/FlannIndex/Kmeans.h
src/CANDY/FlannIndex/KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex/Kmeans.cpp
src/CANDY/FlannIndex.cpp
src/CANDY/IndexTable.cpp

Classes

- class INTELLI::C20Buffer< dataType >

Macros

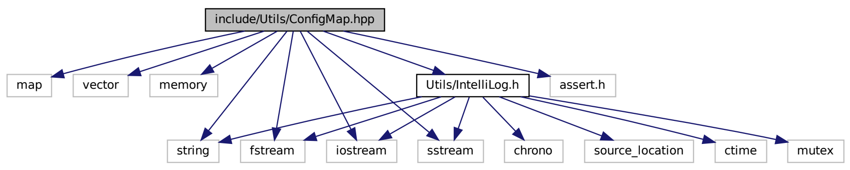
- #define **_UTILS_C20BUFFERS_HPP_**
- #define **ADB_memcpy**(dst, src, size) memcpy(dst, src, size)

9.37 include/Utils/ConfigMap.hpp File Reference

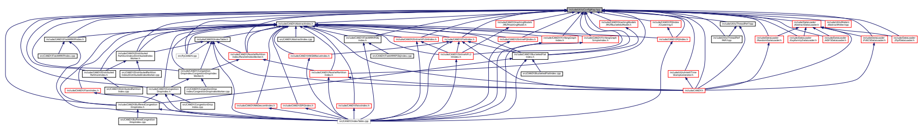
```
#include <map>
#include <vector>
#include <memory>
#include <string>
#include <fstream>
#include <iostream>
#include <sstream>
#include <assert.h>
```

```
#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

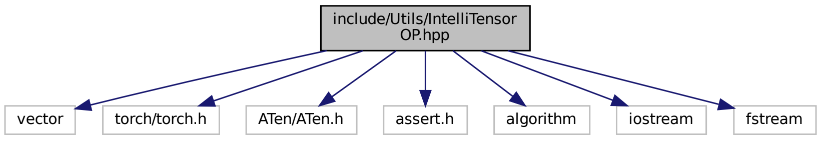
- typedef std::shared_ptr< ConfigMap > INTELLI::ConfigMapPtr

  The class to describe a shared pointer to ConfigMap.

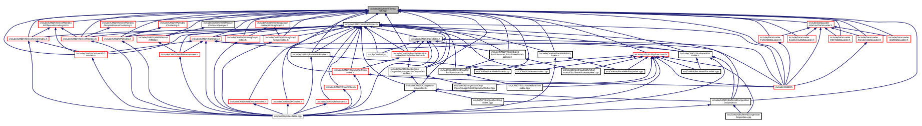
9.38 include/Utils/IntelliTensorOP.hpp File Reference

```
#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.

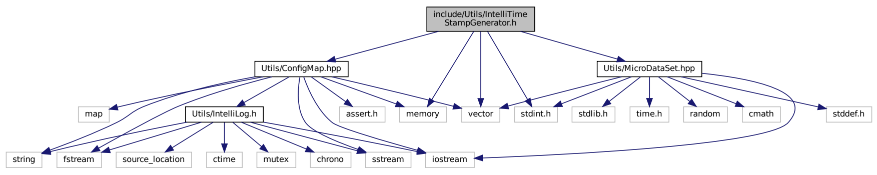
Typedefs

- typedef std::shared_ptr< torch::Tensor > INTELLI::TensorPtr

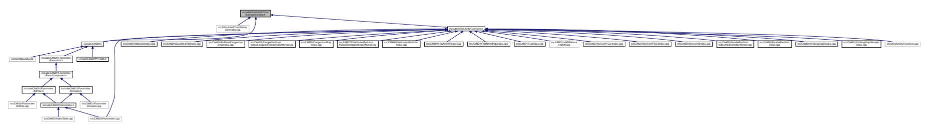
    The class to describe a shared pointer to torch::Tensor.

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.

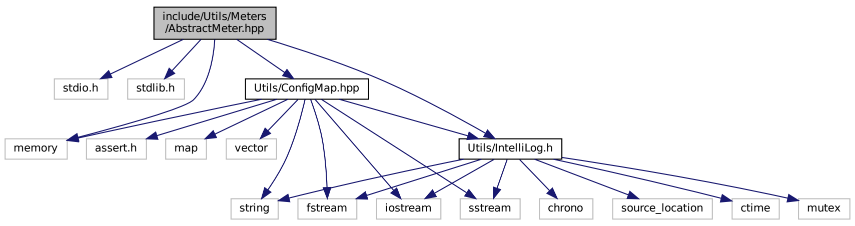
Typedefs

- typedef std::shared_ptr< INTELLI::IntelliTimeStamp > INTELLI::IntelliTimeStampPtr

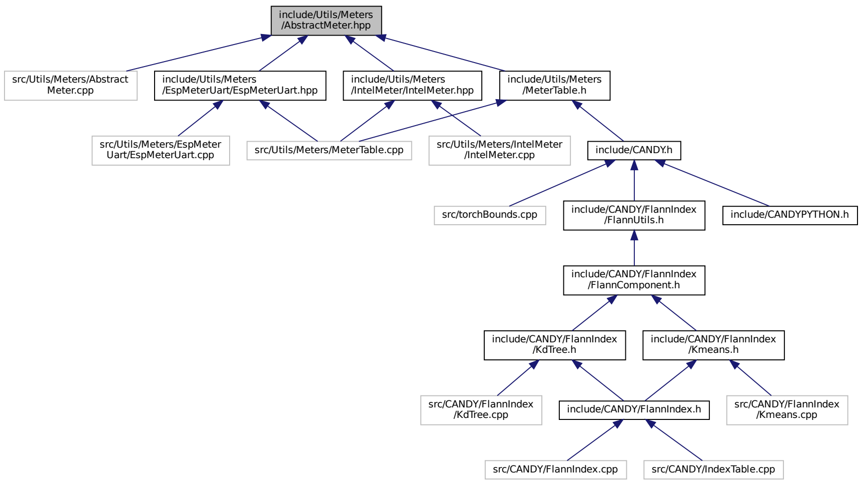
    The class to describe a shared pointer to IntelliTimeStamp.

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:


include/Utils/Meters/AbstractMeter.hpp
src/Utils/Meters/AbstractMeter.cpp
include/Utils/Meters/EspMeterUart/EspMeterUart.hpp
include/Utils/Meters/IntelMeter/IntelMeter.hpp
include/Utils/Meters/MeterTable.h
src/Utils/Meters/EspMeterUart/EspMeterUart.cpp
src/Utils/Meters/MeterTable.cpp
src/Utils/Meters/IntelMeter/IntelMeter.cpp
include/CANDY.h
src/torchBounds.cpp
include/CANDY/FlannIndex/FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex/FlannComponent.h
include/CANDY/FlannIndex/KdTree.h
include/CANDY/FlannIndex/Kmeans.h
src/CANDY/FlannIndex/KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex/Kmeans.cpp
src/CANDY/FlannIndex.cpp
src/CANDY/IndexTable.cpp

Classes

- class DIVERSE_METER::AbstractMeter

    The abstract class for all meters.

Macros

- #define **METER_ERROR**(n) INTELLI_ERROR(n)

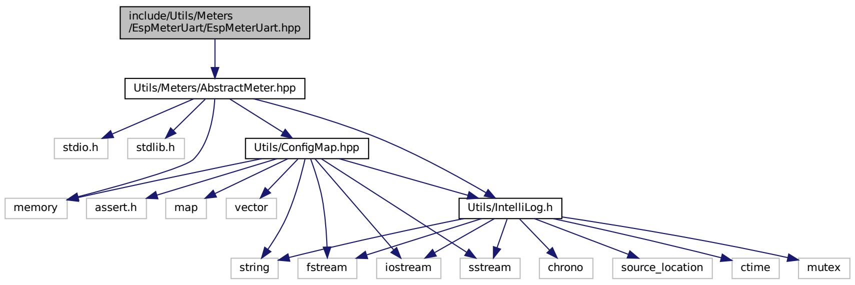
Typedefs

- typedef std::shared_ptr< DIVERSE_METER::AbstractMeter > **DIVERSE_METER::AbstractMeterPtr**

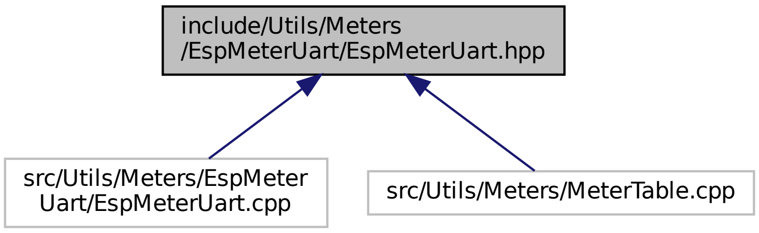
9.41 include/Utils/Meters/EspMeterUart/EspMeterUart.hpp File Reference

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

Include dependency graph for EspMeterUart.hpp:



This graph shows which files directly or indirectly include this file:


include/Utils/Meters
/EspMeterUart/EspMeterUart.hpp
src/Utils/Meters/EspMeter
Uart/EspMeterUart.cpp
src/Utils/Meters/MeterTable.cpp

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

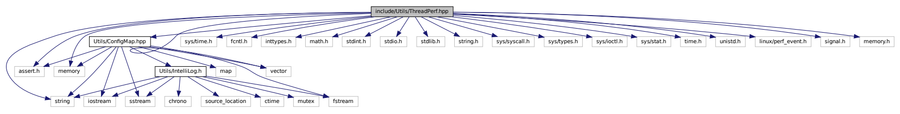
- typedef std::shared_ptr< DIVERSE_METER::EspMeterUart > **DIVERSE_METER::EspMeterUartPtr**

9.42 include/Utils/ThreadPerf.hpp File Reference

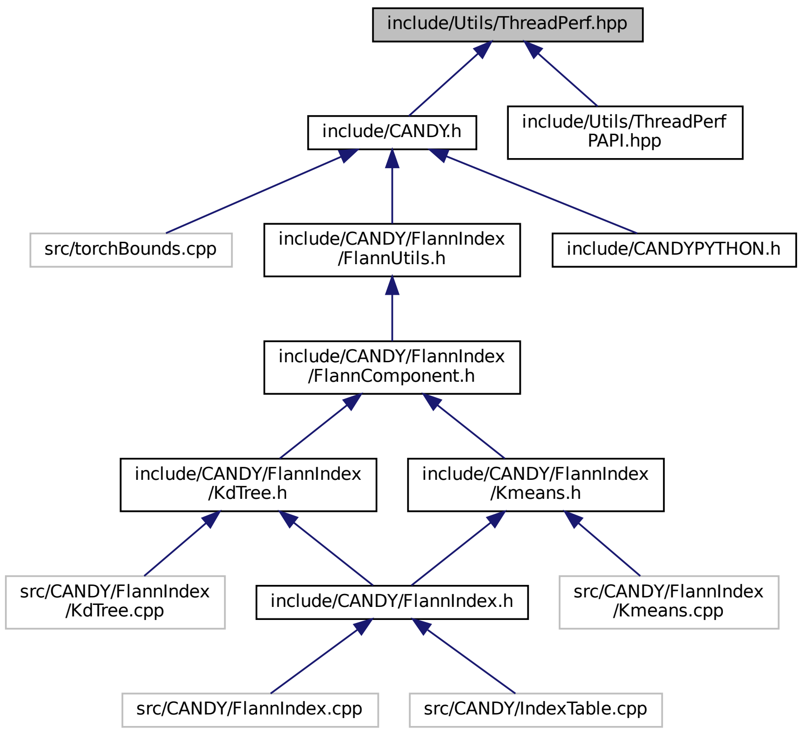
```
#include <string>
#include <sys/time.h>
#include <assert.h>
#include <fcntl.h>
#include <inttypes.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <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:


include/Utils/ThreadPerf.hpp
include/CANDY.h
include/Utils/ThreadPerf PAPI.hpp
src/torchBounds.cpp
include/CANDY/FlannIndex /FlannUtils.h
include/CANDYPYTHON.h
include/CANDY/FlannIndex /FlannComponent.h
include/CANDY/FlannIndex /KdTree.h
include/CANDY/FlannIndex /Kmeans.h
src/CANDY/FlannIndex /KdTree.cpp
include/CANDY/FlannIndex.h
src/CANDY/FlannIndex /Kmeans.cpp
src/CANDY/FlannIndex.cpp
src/CANDY/IndexTable.cpp

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

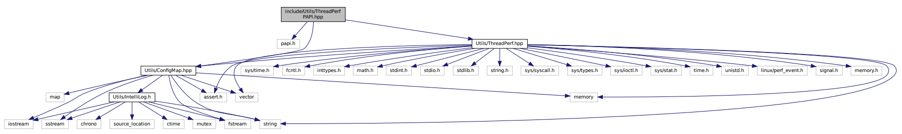
- enum INTELLI::perfTrace {
COUNT_SW_CPU_CLOCK = 0 , **COUNT_SW_TASK_CLOCK** = 1 , **COUNT_SW_CONTEXT_SWITCHES** = 2 , **COUNT_SW_CPU_MIGRATIONS** = 3 ,
COUNT_SW_PAGE_FAULTS = 4 , **COUNT_SW_PAGE_FAULTS_MIN** = 5 , **COUNT_SW_PAGE_FAULTS↩_MAJ** = 6 , **COUNT_HW_CPU_CYCLES** = 7 ,
COUNT_HW_INSTRUCTIONS = 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.

Typedefs

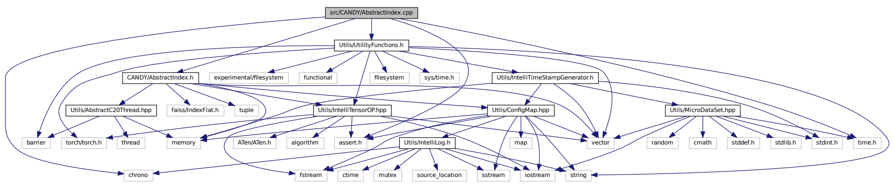
- typedef std::shared_ptr< INTELLI::ThreadPerfPAPI > INTELLI::ThreadPerfPAPIPtr

  The class to describe a shared pointer to ThreadPerfPAPI.

9.44 src/CANDY/AbstractIndex.cpp File Reference

```
#include <CANDY/AbstractIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

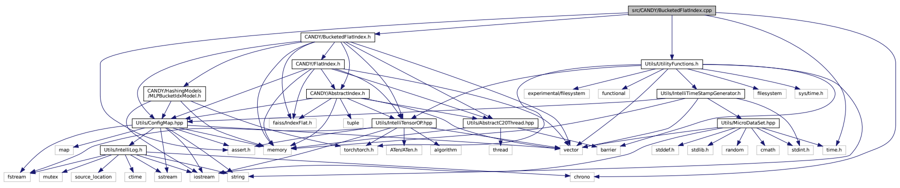
Include dependency graph for AbstractIndex.cpp:



9.45 src/CANDY/BucketedFlatIndex.cpp File Reference

```
#include "CANDY/BucketedFlatIndex.h"
#include "Utils/UtilityFunctions.h"
#include <time.h>
#include <chrono>
#include <assert.h>
```

Include dependency graph for BucketedFlatIndex.cpp:



Macros

- #define BF_NEXT_POW_2(V)
- #define **HASH**(X, MASK, SKIP) (((X) & MASK) >> SKIP)

9.45.1 Macro Definition Documentation

9.45.1.1 BF_NEXT_POW_2

```
#define BF_NEXT_POW_2(
                V )
```

Value:

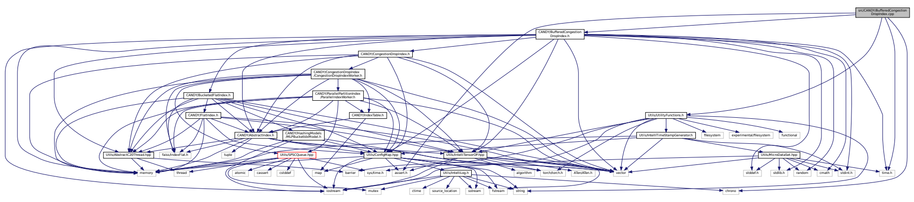
```
    do {                                              \
        V--;                                          \
        V |= V » 1;                                  \
        V |= V » 2;                                  \
        V |= V » 4;                                  \
        V |= V » 8;                                  \
        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>
```

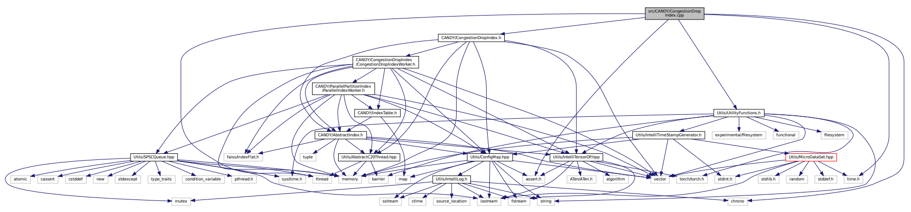
Include dependency graph for BufferedCongestionDropIndex.cpp:



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>
```

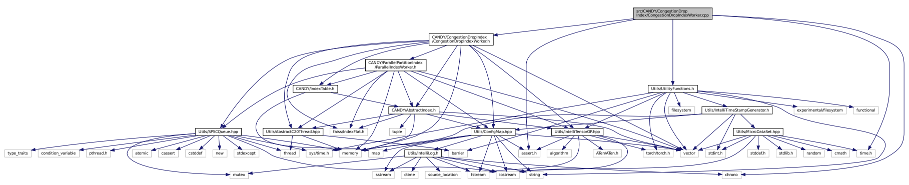
Include dependency graph for CongestionDropIndex.cpp:



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>
```

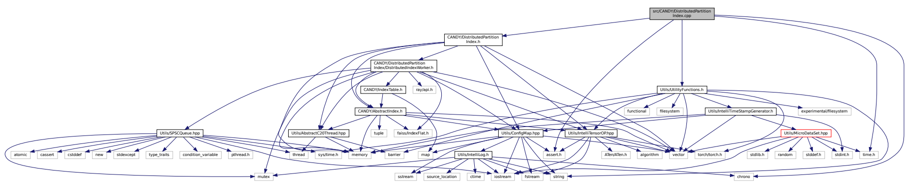
Include dependency graph for CongestionDropIndexWorker.cpp:



9.49 src/CANDY/DistributedPartitionIndex.cpp File Reference

```
#include <CANDY/DistributedPartitionIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

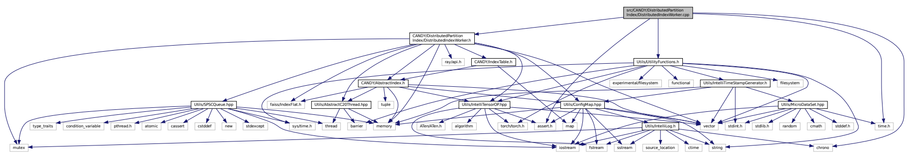
Include dependency graph for DistributedPartitionIndex.cpp:



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:



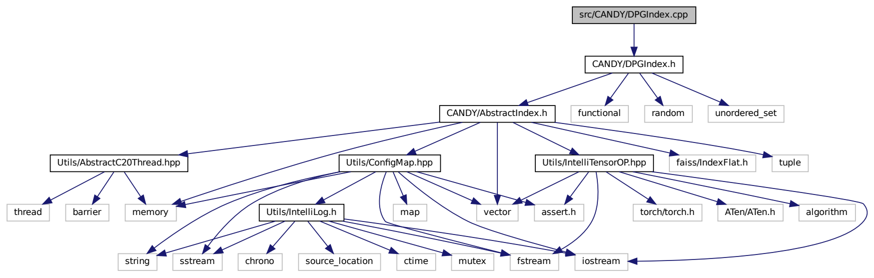
Functions

- **RAY_REMOTE** (CANDY::DIW_RayWrapper::FactoryCreate, &CANDY::DIW_RayWrapper::setConfig, &CANDY::DIW_RayWrapper::insertTensor, &CANDY::DIW_RayWrapper::deleteTensor, &CANDY::DIW_RayWrapper::searchTe &CANDY::DIW_RayWrapper::reset, &CANDY::DIW_RayWrapper::startHPC, &CANDY::DIW_RayWrapper::endHPC, &CANDY::DIW_RayWrapper::setFrozenLevel, &CANDY::DIW_RayWrapper::offlineBuild, &CANDY::DIW_RayWrapper::loadIniti &CANDY::DIW_RayWrapper::waitPendingOperations)

9.51 src/CANDY/DPGIndex.cpp File Reference

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

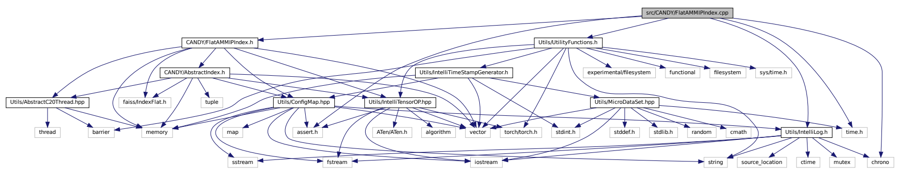
Include dependency graph for DPGIndex.cpp:



9.52 src/CANDY/FlatAMMIPIndex.cpp File Reference

```
#include <CANDY/FlatAMMIPIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
#include <Utils/IntelliLog.h>
```

Include dependency graph for FlatAMMIPIndex.cpp:

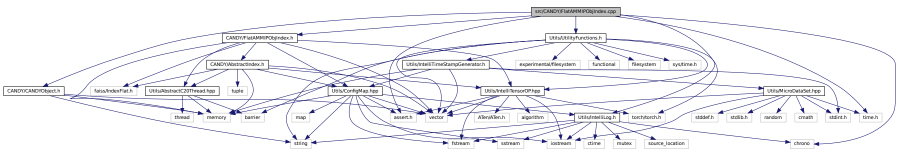


9.53 src/CANDY/FlatAMMIPObjIndex.cpp File Reference

```
#include <CANDY/FlatAMMIPObjIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
```

```
#include <assert.h>
#include <Utils/IntelliLog.h>
#include <CANDY/CANDYObject.h>
```

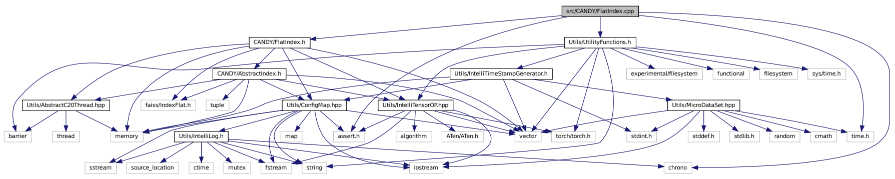
Include dependency graph for FlatAMMIPObjIndex.cpp:



9.54 src/CANDY/FlatIndex.cpp File Reference

```
#include <CANDY/FlatIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

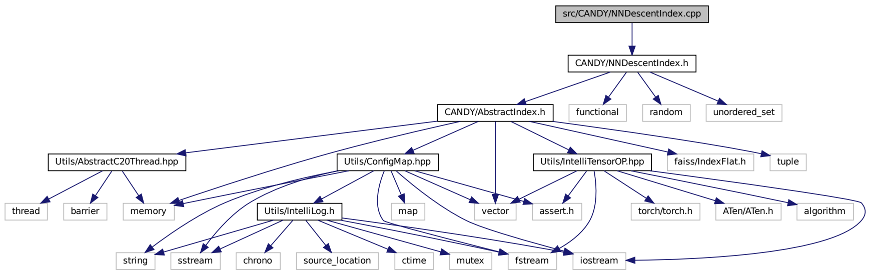
Include dependency graph for FlatIndex.cpp:



9.55 src/CANDY/NNDescentIndex.cpp File Reference

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

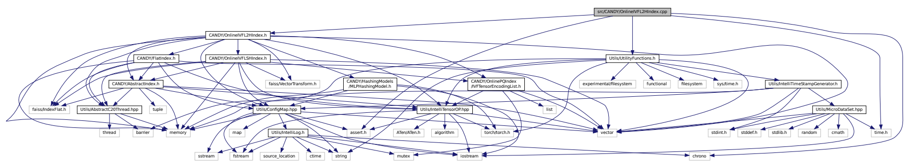
Include dependency graph for NNDescentIndex.cpp:



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)
```

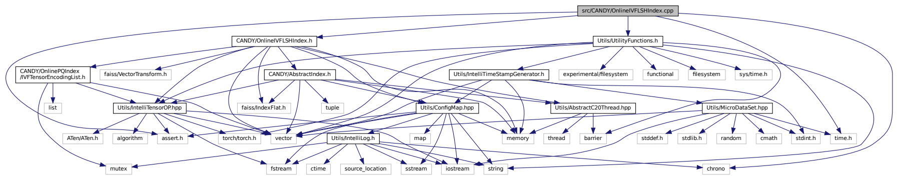
compute the next number, greater than or equal to 32-bit unsigned v. taken from "bit twiddling hacks": http://graphics.stanford.edu/~seander/bithacks.html

9.57 src/CANDY/OnlineIVFLSHIndex.cpp File Reference

```
#include <CANDY/OnlineIVFLSHIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
```

```
#include <assert.h>
```

Include dependency graph for OnlineIVFLSHIndex.cpp:



Macros

- #define ONLINEIVF_NEXT_POW_2(V)
- #define **HASH**(X, MASK, SKIP) (((X) & MASK) >> SKIP)

9.57.1 Macro Definition Documentation

9.57.1.1 ONLINEIVF_NEXT_POW_2

```
#define ONLINEIVF_NEXT_POW_2(
                V )
```

Value:

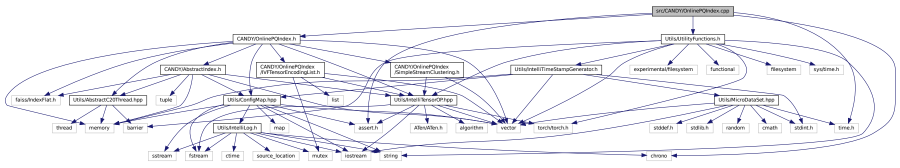
```
    do {                                            \
        V--;                                        \
        V |= V » 1;                                 \
        V |= V » 2;                                 \
        V |= V » 4;                                 \
        V |= V » 8;                                 \
        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

9.58 src/CANDY/OnlinePQIndex.cpp File Reference

```
#include <CANDY/OnlinePQIndex.h>
#include <Utils/UtilityFunctions.h>
#include <time.h>
#include <chrono>
#include <assert.h>
```

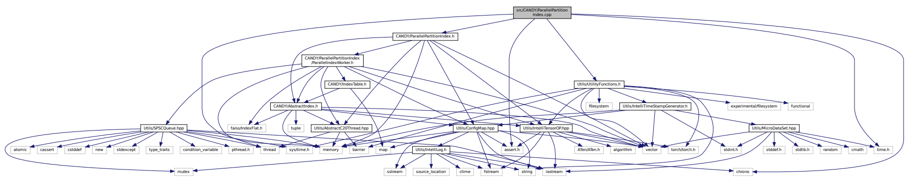
Include dependency graph for OnlinePQIndex.cpp:



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>
```

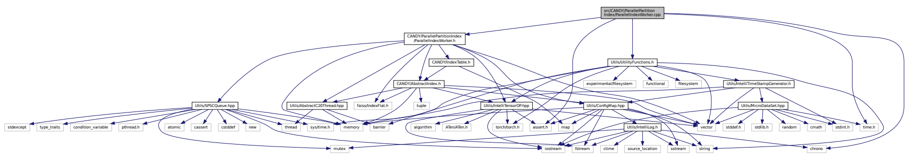
Include dependency graph for ParallelPartitionIndex.cpp:



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>
```

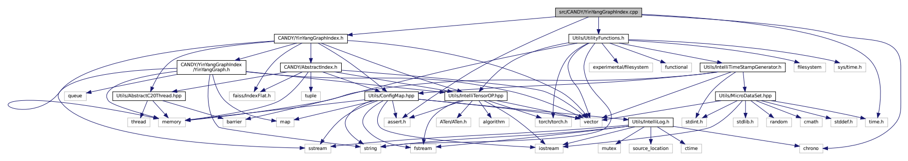
Include dependency graph for ParallelIndexWorker.cpp:



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:

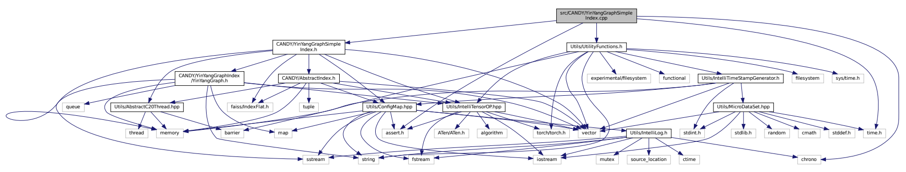
```
    do {                                        \
        V--;                                    \
        V |= V » 1;                            \
        V |= V » 2;                            \
        V |= V » 4;                            \
        V |= V » 8;                            \
        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