LibAMM

0.1.4

Generated by Doxygen 1.9.1

Chapter 1

Introduction

This project is for benchmarking common (approximate) matrix multiplication algorithms under various archeitectures in as streaming setting

1.1 Benchmark Tips

usage: ./benchmark [configfile]

Note

Require configs in configfile:

- aRow (U64) the rows of tensor A, required by RandomMatrixLoader, SparseMatrixLoader
- aCol (U64) the columns of tensor A, required by RandomMatrixLoader, SparseMatrixLoader
- bCol (U64) the columns of tensor B, required by RandomMatrixLoader, SparseMatrixLoader
- aDensity The density factor of matrix A, Double, 1.0, required by SparseMatrixLoader
- bDensity The density factor of matrix B, Double, 1.0, required by SparseMatrixLoader
- aReduce Reduce some rows of A to be linearly dependent, U64, 0, required by SparseMatrixLoader
- "bReduce Reduce some rows of A to be linearly dependent, U64, 0, required by SparseMatrixLoader
- sketchDimension (U64) the dimension of sketch matrix, default 50
- coreBind (U64) the specific core tor run this benchmark, default 0
- ptFile (String) the path for the $*$.pt to be loaded, default torchscripts/FDAMM.pt
- matrixLoaderTag (String) the nameTag of matrix loader, see MatrixLoaderTable, default is random
- useCPP (U64) force the benchmark to use static and pure cpp implementation instead of pt, default 0
- cppAlgoTag (String) The algorithm tag to index a cpp algorithm, works only under useCPP=1, default "mm", see also CPPAlgoTable
- threads, U64, the number of worker threads, default 2
- osScheduling, U64, whether use default os scheduling instead of my own core bind, default 0
- firstCoreBind, U64, which core will the first thread be bound to, default 0, see also BlockPartitionRunner

Additional tags for energy measurement (please validate usingMeter first) see also Energy Meter packs

- usingMeter (U64) set to 1 if you want to use some energy meter, default diabled
- meterTag (String) the tag of meter, see also MeterTable, default is intelMsr
- staticPower (Double) set this to >0 if you want to manually config the static power of the device

- meterAddress (String) set this to the file system path of the meter, if it is different from the meter's default
- isStreaming (U64) whether or not use streaming, default 0

by default, we only make A matrix streaming, if also want yo streaming B, please also set streamingTwo↩ Matrixes to 1

- streamingTwoMatrixes (U64) whether make B matrix also streaming, default 0, will only affect when isStreaming=1

Warning

For some platforms, the staticPower automatically measured by sleep is not accurate. Please do this mannulally. See also the template config.csv

Note

Additional tags for hardware counters

- usePAPI (U64) whether or not use PAPI if papi exists, default 1
  - if PAPI is used, please refer to class ThreadPerfPAPI, otherwise, see class ThreadPerf
  - under PAPI, here is an unified way to custoimize perf events without recompile
    - create a configmap readable csv for perfList at perfLists folder, the key is tag you want to display, and the value is inline PAPI tag, type should be String
    - set perfUseExternalList (U64) in top configfile to 1
    - set perfListSrc (String) in top configfile to perfLists/<your file name>

1.2 How to extend a new algorithm (pt-based)

- go to the benchmark/torchscripts
- find any .python as an example
- copy and modify it and generate the ∗pt, please make it under hump style of naming
- the system will then support it by using the name of your pt.

1.3 How to extend a new algorithm (pure static c++ based)

- go to the src/CPPAlgos and include/CPPAlgos
- copy the example class, such as CRSCPPAlgo, rename it, and implement your own amm function
  - copy the cpp and h
  - rename the cpp and h
  - automatically conduct the IDE-full-replace over the CRSCPPAlgo 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 tour function with a tag to src/CPPAlgos/CPPAlgoTable.cpp
- edit the CMakelist.txt at src/CPPAlgos to include your new algo and recompile
- remember to add a test bench, you can refer to CRSTest.cpp at test/SystemTest for example
- if your algorithms have specific parameters like beta, please additionally do the following in the XXXCPPAlgo class:
  - copy the virtual void setConfig(INTELLI::ConfigMapPtr cfg) claim to your $*$.h
  - implement that function in your cpp

1.4 How to add a single point test

- copy your config file to test/scripts, and your pt file to test/torchscripts
- follow and copy the SketchTest.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

Chapter 2

Todo List

Member INTELLI::UtilityFunctions::bind2Core (int id)

unsure about hyper-thread

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 matrix loaders

Classes

- class AMMBench::AbstractMatrixLoader

  The abstract class of matrix loader, parent for all loaders.
- class AMMBench::BetaMatrixLoader

  The Beta class of matrix loader.
- class AMMBench::BinomialMatrixLoader

  The Binomial class of matrix loader.
- class AMMBench::CCAMatrixLoader

  For CCA downstream task.
- class AMMBench::ExponentialMatrixLoader

  The Exponential class of matrix loader.
- class AMMBench::GaussianMatrixLoader

  The Gaussian class of matrix loader.
- class AMMBench::MatrixLoaderTable

  The table class to index all matrix loaders.
- class AMMBench::MediaMillMatrixLoader

  Load MediaMill 2005-2006 data (https://rdrr.io/github/fcharte/mldr.datasets/man/mediamill.↩ html)
- class AMMBench::MNISTMatrixLoader

  The MNIST class of matrix loader https://www.kaggle.com/datasets/hojjatk/mnist-dataset.
- class AMMBench::MtxMatrixLoader

  The matrix loader to load matrixes stored in matrix market mtx format.
- class AMMBench::PoissonMatrixLoader

  The Poisson class of matrix loader.
- class AMMBench::RandomMatrixLoader

  The Random class of matrix loader.
- class AMMBench::SIFTMatrixLoader

  The SIFT class of matrix loader http://corpus-texmex.irisa.fr/.
- class AMMBench::SparseMatrixLoader

  The matrix loader to generate adjustable sparse matrix with adjust rank reduction.
- class AMMBench::ZeroMaskedMatrixLoader

The zero masked class of matrix loader, given generate a n∗m matrix, where only the left-top n1∗m2 contents are not zero.

- class AMMBench::ZipfMatrixLoader

  The Zipf class of matrix loader.

- torch::Tensor AMMBench::loadMatrixFromMatrixMarket (const string &filename)

  the stan-alone function to load a matrix from matrix market mitx file

7.1.1 Detailed Description

7.1.1.1 MatrixLoader

This folder contains the loader to matrixes under different generation rules

We define the generation classes of matrixes. here

7.1.2 Function Documentation

7.1.2.1 loadMatrixFromMatrixMarket()

```
torch::Tensor AMMBench::loadMatrixFromMatrixMarket (
            const string & filename )
```

the stan-alone function to load a matrix from matrix market mitx file

Parameters

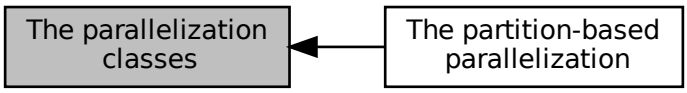
filename	the name of the mtx file

Returns

the loaded tensor

7.2 The parallelization classes

Collaboration diagram for The parallelization classes:



Modules

- The partition-based parallelization

7.2.1 Detailed Description

7.2.1.1 Parallelization

This folder contains the parallelization approaches

We define the parallelization classes of AMM. here

7.3 The streaming classes

Classes

- class AMMBench::BlockPartitionStreamer

  The class to run streaming amm under block partition scheme, let rows of A coming in a streaming manner, all of which are partitioned with BlockPartitionRunner.
- class AMMBench::SingleThreadStreamer

  The class to run streaming amm under single thread, let each row of A coming in a streaming manner.
- class AMMBench::AMMTimeStamp

  The class to define timestamp in streaming.
- class AMMBench::TimeStamper

  The basic class to generate time stamps.

7.3.1 Detailed Description

7.3.1.1 STREAMING

This folder contains the STREAMING approaches

We define the streaming classes of AMM. here

7.4 The c++ amm algorithms

Classes

- class AMMBench::AbstractCPPAlgo
  The abstract class of c++ algos.
- class AMMBench::BCRSCPPAlgo
  The Bernoulli column row sampling (BCRS) class of c++ algos.
- class AMMBench::BetaCoOFDCPPAlgo
  The Beta Co-Occurring FD AMM class of c++ algos.
- class AMMBench::CLMMCPPAlgo
  The MM class of c++ algos using opencl.
- class AMMBench::CoOccurringFDCPPAlgo
  The Co-Occurring FD AMM class of c++ algos.
- class AMMBench::CountSketchCPPAlgo
  The counter sketch class of c++ algos.
- class AMMBench::CPPAlgoTable
  The table to index cpp algos.
- class AMMBench::CRSCPPAlgo
  The column row sampling (CRS) class of c++ algos.
- class AMMBench::CRSV2CPPAlgo
  The column row sampling (CRS) class of c++ algos, a second implementation.
- class AMMBench::EWSCPPAlgo
  The Element Wise Sampling (EWS) class of c++ algos.
- class AMMBench::FastJLTCPPAlgo
  The tug of war class of c++ algoS.
- class AMMBench::INT8CPPAlgo
  The INT8 MM class of c++ algos.
- class AMMBench::ProductQuantizationHash
  The Product Quantization AMM class of c++ algos, using hash function to find matching prototypes.
- class AMMBench::ProductQuantizationRaw
  The Product Quantization AMM class of c++ algos, using Euclidean distance.
- class AMMBench::RIPCPPAlgo
  New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property.
- class AMMBench::SMPPCACPPAlgo
  sketch scaled JL class of c++ algos
- class AMMBench::TugOfWarCPPAlgo
  The tug of war class of c++ algoS.
- class AMMBench::VectorQuantization
  The Vector Quantization AMM class of c++ algos.

- #define newProductQuantizationHashAlgo std::make_shared<AMMBench::ProductQuantizationHash>
  (Macro) To creat a new ProductQuantizationHashAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::ProductQuantizationHash > **AMMBench::ProductQuantizationHashPtr**

- #define newProductQuantizationRawAlgo std::make_shared<AMMBench::ProductQuantizationRaw>
  (Macro) To creat a new ProductQuantizationRawAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::ProductQuantizationRaw > **AMMBench::ProductQuantizationRawPtr**

- #define newVectorQuantizationAlgo std::make_shared<AMMBench::VectorQuantization>
  (Macro) To creat a new VectorQuantizationAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::VectorQuantization > **AMMBench::VectorQuantizationPtr**

7.4.1 Detailed Description

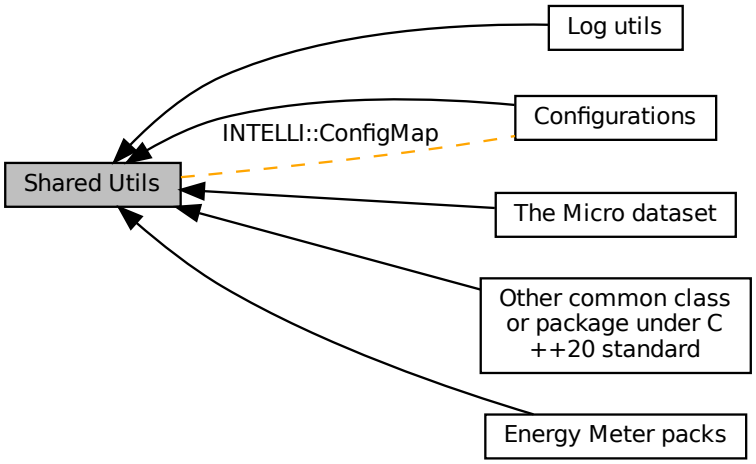
7.4.1.1 c++ algorithms

This folder contains the agorithms implemented under pure c++

We define the c++ algorithm classes of AMM. here

7.5 Shared Utils

Collaboration diagram for Shared Utils:



Modules

- Other common class or package under C++20 standard
- Configurations
- Log utils
- Energy Meter packs
- The Micro dataset

Classes

- class INTELLI::ConfigMap

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

Functions

- void **INTELLI::ConfigMap::spilt** (const std::string s, const std::string &c, vector< std::string > &v)
- void INTELLI::ConfigMap::edit (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 (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 (std::string key, double value)

  Edit the config map. If not exit the config, will create new, or will overwrite.
- void INTELLI::ConfigMap::edit (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 (std::string key)

  To detect whether the key exists and related to a U64.
- bool INTELLI::ConfigMap::existI64 (std::string key)

  To detect whether the key exists and related to a I64.
- bool INTELLI::ConfigMap::existDouble (std::string key)

  To detect whether the key exists and related to a double.
- bool INTELLI::ConfigMap::existString (std::string key)

  To detect whether the key exists and related to a std::string.
- bool INTELLI::ConfigMap::exist (std::string key)

  To detect whether the key exists.
- uint64_t INTELLI::ConfigMap::getU64 (std::string key)

  To get a U64 value by key.
- int64_t INTELLI::ConfigMap::getI64 (std::string key)

  To get a I64 value by key.
- double INTELLI::ConfigMap::getDouble (std::string key)

  To get a double value by key.
- std::string INTELLI::ConfigMap::getString (std::string key)

  To get a std::string value by key.
- std::string INTELLI::ConfigMap::toString (std::string separator="\t", std::string newLine="\n")

  convert the whole map to std::string and retuen
- void INTELLI::ConfigMap::cloneInto (ConfigMap &dest)

  clone this config into destination
- bool INTELLI::ConfigMap::toFile (std::string fname, std::string separator=",", std::string newLine="\n")

  convert the whole map to file
- bool INTELLI::ConfigMap::fromFile (std::string fname, std::string separator=",", std::string newLine="\n")

  update the whole map from file
- 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.
- uint64_t INTELLI::ConfigMap::tryU64 (const string &key, uint64_t defaultValue=0, bool showWarning=false)

  Try to get an U64 from config map, if not exist, use default value instead.
- double INTELLI::ConfigMap::tryDouble (const string &key, double defaultValue=0, bool showWarning=false)

  Try to get a double from config map, if not exist, use default value instead.
- string INTELLI::ConfigMap::tryString (const string &key, const string &defaultValue="", bool showWarning=false)

  Try to get an String from config map, if not exist, use default value instead.
- std::map< std::string, std::string > INTELLI::ConfigMap::getStrMap ()

  return the map of string
- void INTELLI::ConfigMap::addPrefixToKeys (std::string prefix)

  Add prefix to the front of keys, it is useful in downstream task where we need to generate metric config file for each components in the downstream task e.g. instructions -> ${prefix}Instructions.

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**

7.5.1 Detailed Description

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

7.5.2 Function Documentation

7.5.2.1 addPrefixToKeys()

```
void INTELLI::ConfigMap::addPrefixToKeys (
            std::string prefix )  [inline]
```

Add prefix to the front of keys, it is useful in downstream task where we need to generate metric config file for each components in the downstream task e.g. instructions -> ${prefix}Instructions.

Parameters

prefix	The prefix you want to add to the front of keys

Returns

void

7.5.2.2 cloneInto()

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

clone this config into destination

Parameters

dest	The clone destination

7.5.2.3 edit() [1/4]

```
void INTELLI::ConfigMap::edit (
            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

7.5.2.4 edit() [2/4]

```
void INTELLI::ConfigMap::edit (
            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

7.5.2.5 edit() [3/4]

```
void INTELLI::ConfigMap::edit (
            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

7.5.2.6 edit() [4/4]

```
void INTELLI::ConfigMap::edit (
```

```
            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

7.5.2.7 exist()

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

To detect whether the key exists.

Parameters

key	

Returns

bool for the result

7.5.2.8 existDouble()

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

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

Parameters

key	

Returns

bool for the result

7.5.2.9 existI64()

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

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

Parameters

key	

Returns

bool for the result

7.5.2.10 existString()

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

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

Parameters

key	

Returns

bool for the result

7.5.2.11 existU64()

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

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

Parameters

key	

Returns

bool for the result

7.5.2.12 fromFile()

```
bool INTELLI::ConfigMap::fromFile (
            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

7.5.2.13 getDouble()

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

To get a double value by key.

Parameters

key	

Returns

value

Warning

the key must exist!!

7.5.2.14 getI64()

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

To get a I64 value by key.

Parameters

key	

Returns

value

Warning

the key must exist!!

7.5.2.15 getString()

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

To get a std::string value by key.

Parameters

key	

Returns

value

Warning

the key must exist!!

7.5.2.16 getStrMap()

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

return the map of string

Returns

the strMap variable

7.5.2.17 getU64()

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

To get a U64 value by key.

Parameters

key	

Returns

value

Warning

the key must exist!!

7.5.2.18 toFile()

```
bool INTELLI::ConfigMap::toFile (
            std::string fname,
            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

7.5.2.19 toString()

```
std::string INTELLI::ConfigMap::toString (
            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

7.5.2.20 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

7.5.2.21 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

7.5.2.22 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

7.5.2.23 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

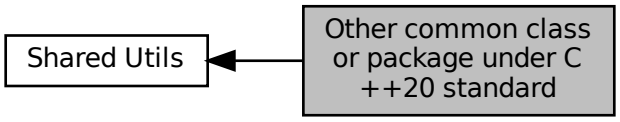
key	The key
defaultValue	The default
showWarning	Whether show warning logs if not found

Returns

The returned value

7.6 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::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

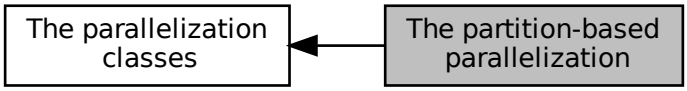
  The class to describe a shared pointer to ThreadPerfPAPI.

7.6.1 Detailed Description

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

7.7 The partition-based parallelization

Collaboration diagram for The partition-based parallelization:



Classes

- class AMMBench::BlockPartitionWorker

  The basic partition worker.
- class AMMBench::BlockPartitionRunner

  The top entity to control all workers, see also BlockPartitionWorker. This one works under a simple row partition parallelization.

- #define newBlockPartitionWorker std::make_shared<AMMBench::BlockPartitionWorker>

  (Macro) To creat a new BlockPartitionWorker under shared pointer.
- typedef std::shared_ptr< AMMBench::BlockPartitionWorker > **AMMBench::BlockPartitionWorkerPtr**

7.7.1 Detailed Description

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

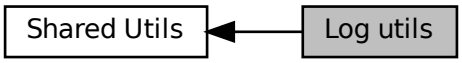
  The class to describe a shared pointer to ConfigMap.

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]
```

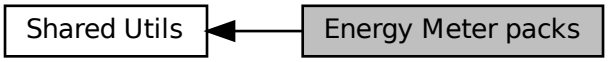
set up the logging file by its name

Parameters

fname	the name of file

7.10 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.

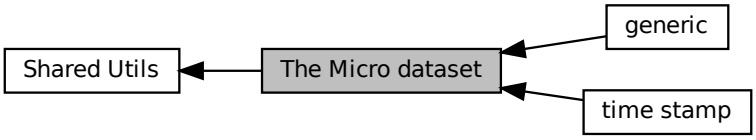
- typedef std::shared_ptr< DIVERSE_METER::AbstractMeter > **DIVERSE_METER::AbstractMeterPtr**

7.10.1 Detailed Description

This package is used for energy meter

7.11 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

7.11.1 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},
 }
```

7.11.2 Function Documentation

7.11.2.1 MicroDataSet()

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

construction with seed

Parameters

seed	The seed for random generator

7.11.2.2 setSeed()

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

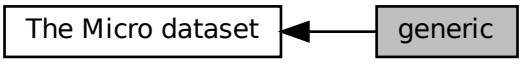
construction with seed

Parameters

seed	The seed for random generator

7.12 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.

7.12.1 Detailed Description

The functions for general generation of Micro

7.12.2 Function Documentation

7.12.2.1 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

7.12.2.2 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

7.12.2.3 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

7.12.2.4 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

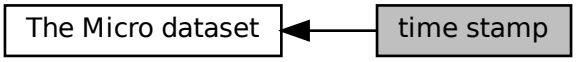
Compute scaling factor such that

sum (lut[i], i=1..alphabet_size) = 1.0

Generate the lookup table

7.13 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.

7.13.1 Detailed Description

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

7.13.2 Function Documentation

7.13.2.1 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

7.13.2.2 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

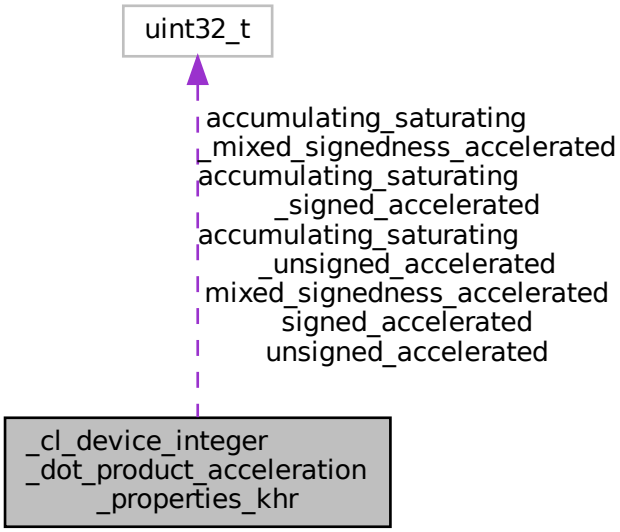
genZipfInt

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

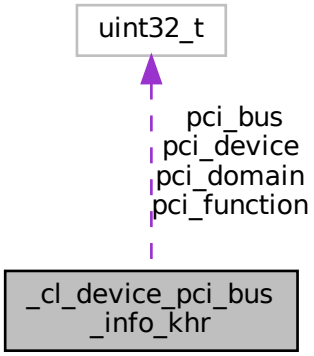
- cl_bool **signed_accelerated**
- cl_bool **unsigned_accelerated**
- cl_bool **mixed_signedness_accelerated**
- cl_bool **accumulating_saturating_signed_accelerated**
- cl_bool **accumulating_saturating_unsigned_accelerated**
- cl_bool **accumulating_saturating_mixed_signedness_accelerated**

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

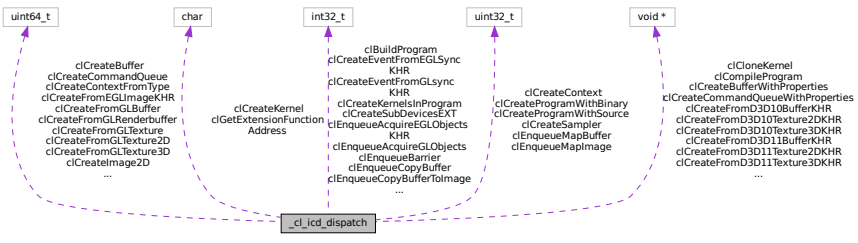
- cl_uint **pci_domain**
- cl_uint **pci_bus**
- cl_uint **pci_device**
- cl_uint **pci_function**

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**

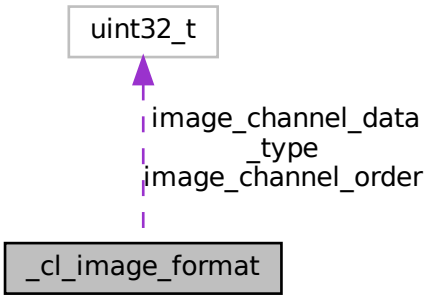
- cl_api_clCreateFromGLTexture **clCreateFromGLTexture**
- cl_api_clGetDeviceIDsFromD3D11KHR **clGetDeviceIDsFromD3D11KHR**
- cl_api_clCreateFromD3D11BufferKHR **clCreateFromD3D11BufferKHR**
- cl_api_clCreateFromD3D11Texture2DKHR **clCreateFromD3D11Texture2DKHR**
- cl_api_clCreateFromD3D11Texture3DKHR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

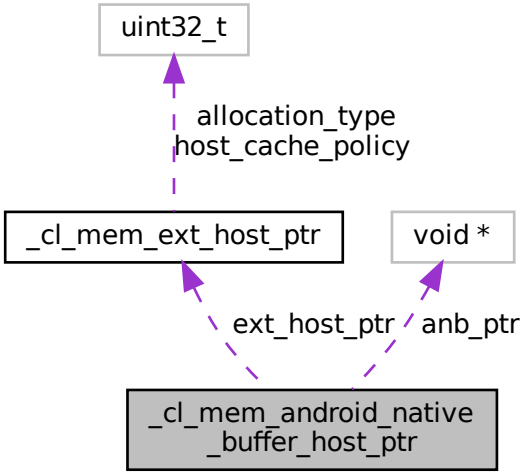
- cl_channel_order **image_channel_order**
- cl_channel_type **image_channel_data_type**

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

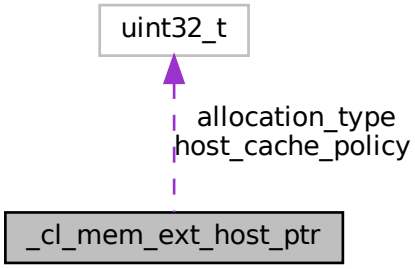
- cl_mem_ext_host_ptr **ext_host_ptr**
- void * **anb_ptr**

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

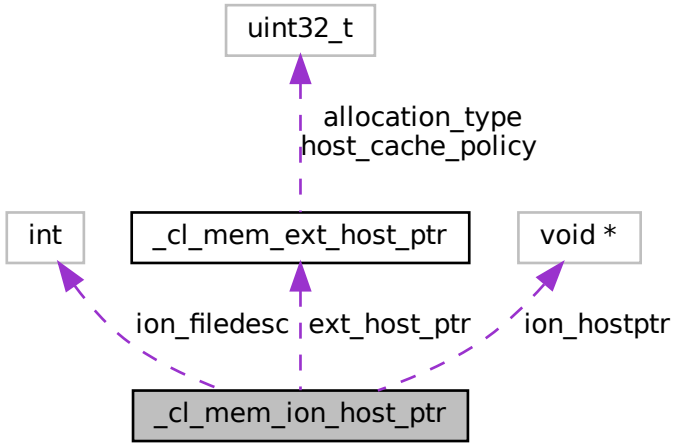
- cl_uint **allocation_type**
- cl_uint **host_cache_policy**

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

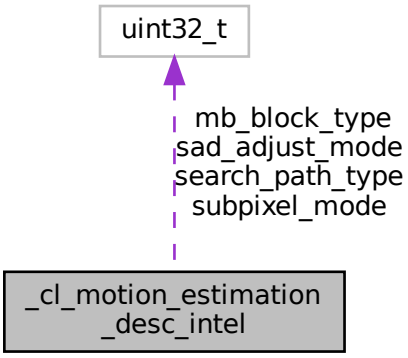
- cl_mem_ext_host_ptr **ext_host_ptr**
- int **ion_filedesc**
- void * **ion_hostptr**

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

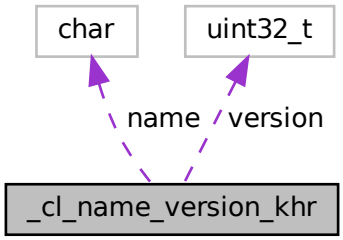
- cl_uint **mb_block_type**
- cl_uint **subpixel_mode**
- cl_uint **sad_adjust_mode**
- cl_uint **search_path_type**

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

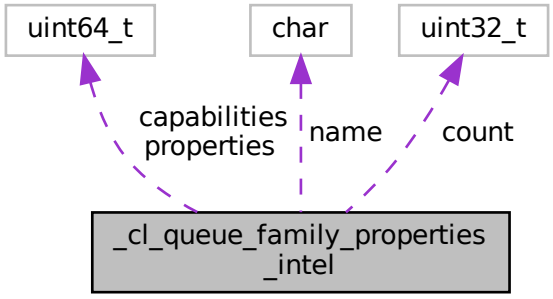
- cl_version_khr **version**
- char **name** [CL_NAME_VERSION_MAX_NAME_SIZE_KHR]

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:

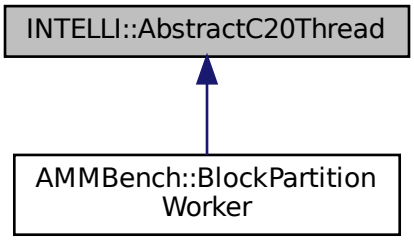
- include/CL/cl_ext.h

8.11 INTELLI::AbstractC20Thread Class Reference

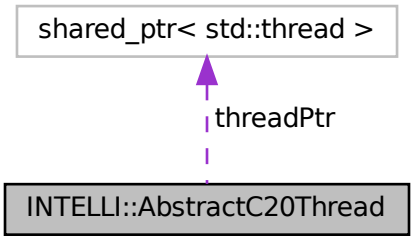
The base class and abstraction of C++20 thread, and it can be derived into other threads.

`#include <Utils/AbstractC20Thread.hpp>`

Inheritance diagram for INTELLI::AbstractC20Thread:



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 AMMBench::BlockPartitionWorker.

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

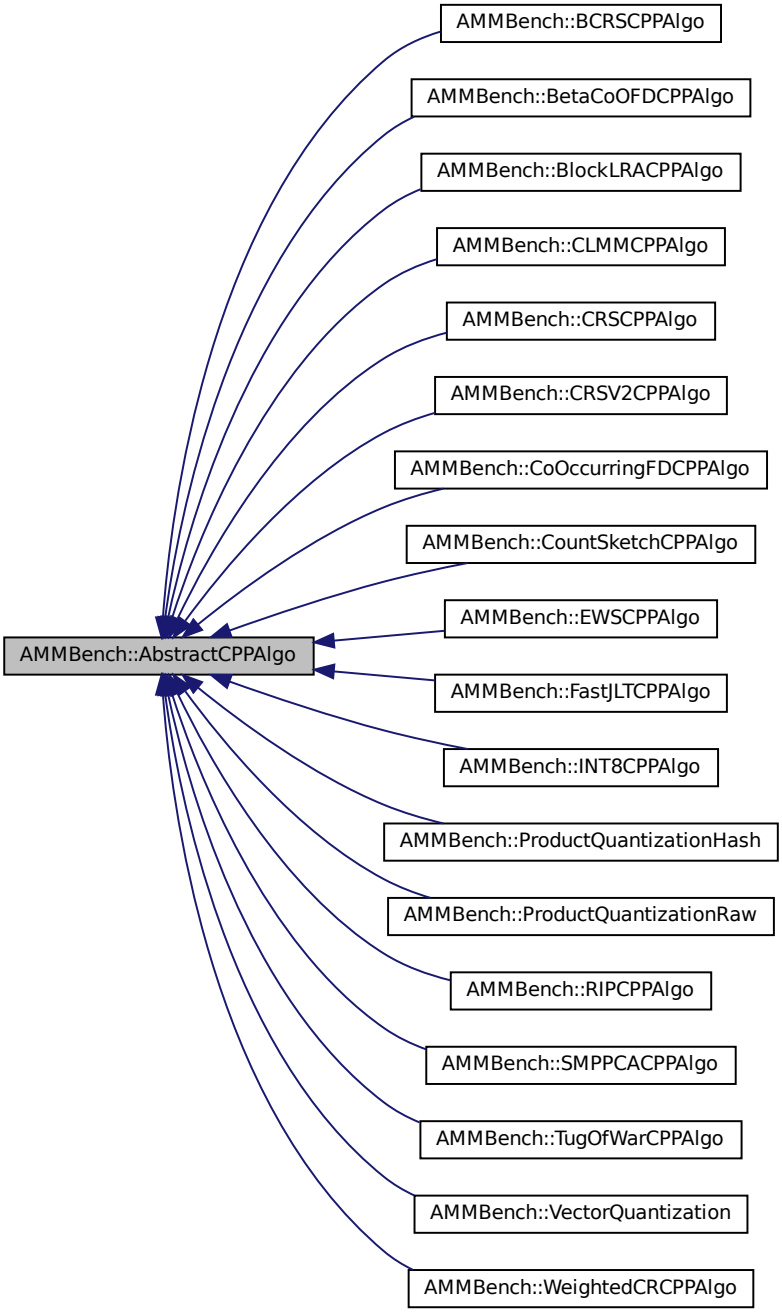
- include/Utils/AbstractC20Thread.hpp

8.12 AMMBench::AbstractCPPAlgo Class Reference

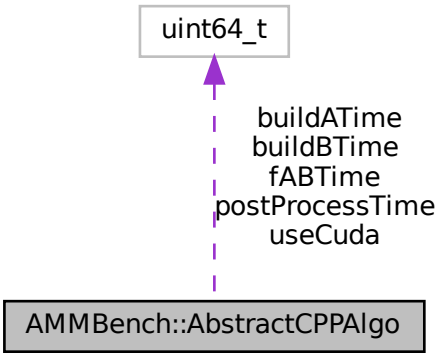
The abstract class of c++ algos.

```
#include <CPPAlgos/AbstractCPPAlgo.h>
```

Inheritance diagram for AMMBench::AbstractCPPAlgo:



Collaboration diagram for AMMBench::AbstractCPPAlgo:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm
- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes
- virtual INTELLI::ConfigMapPtr getBreakDown ()

  to get the breakdown of this algorithm, returned as a config map

Protected Attributes

- uint64_t buildATime = 0

  the default time break dowm variables
- uint64_t **buildBTime** = 0
- uint64_t **fABTime** = 0
- uint64_t **postProcessTime** = 0
- uint64_t **useCuda** = 0

8.12.1 Detailed Description

The abstract class of c++ algos.

++

8.12.2 Member Function Documentation

8.12.2.1 amm()

```
torch::Tensor AMMBench::AbstractCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented in AMMBench::WeightedCRCPPAlgo, AMMBench::VectorQuantization, AMMBench::TugOfWarCPPAlgo, AMMBench::SMPPCACPPAlgo, AMMBench::RIPCPPAlgo, AMMBench::ProductQuantizationRaw, AMMBench::ProductQuantizationH AMMBench::INT8CPPAlgo, AMMBench::FastJLTCPPAlgo, AMMBench::EWSCPPAlgo, AMMBench::CRSV2CPPAlgo, AMMBench::CRSCPPAlgo, AMMBench::CountSketchCPPAlgo, AMMBench::CoOccurringFDCPPAlgo, AMMBench::CLMMCPPAlgo, AMMBench::BetaCoOFDCPPAlgo, AMMBench::BCRSCPPAlgo, and AMMBench::BlockLRACPPAlgo.

8.12.2.2 getBreakDown()

```
INTELLI::ConfigMapPtr AMMBench::AbstractCPPAlgo::getBreakDown ( )  [virtual]
```

to get the breakdown of this algorithm, returned as a config map

Returns

the key-value table breakdown in ConfigMapPtr;

8.12.3 Member Data Documentation

8.12.3.1 buildATime

```
uint64_t AMMBench::AbstractCPPAlgo::buildATime = 0  [protected]
```

the default time break dowm variables

Note

By default, we decompose each AMM as

- buildA, to translate A matrix
- buildB, to translate B matrix
- fABTime, to conduct mm or table look-up over the reduced A,B
- postProcessTime, if f(A,B) is not the finall result, measure the time spend for post process
- useCuda, whether or not use cuda to conduct computation, default 0

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

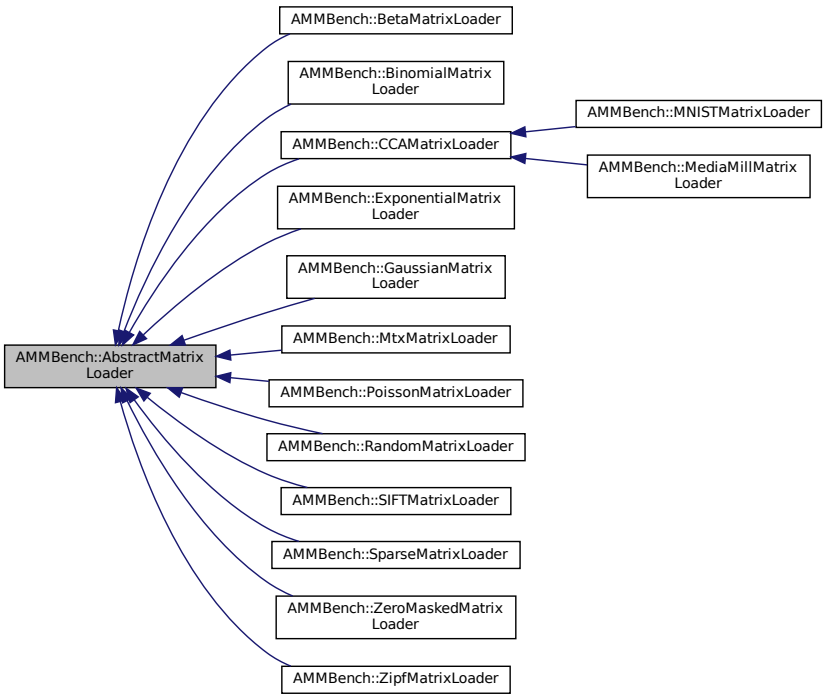
- include/CPPAlgos/AbstractCPPAlgo.h
- src/CPPAlgos/AbstractCPPAlgo.cpp

8.13 AMMBench::AbstractMatrixLoader Class Reference

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

`#include <MatrixLoader/AbstractMatrixLoader.h>`

Inheritance diagram for AMMBench::AbstractMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

8.13.1 Detailed Description

The abstract class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

8.13.2 Member Function Documentation

8.13.2.1 getA()

```
torch::Tensor AMMBench::AbstractMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented in AMMBench::ZipfMatrixLoader, AMMBench::ZeroMaskedMatrixLoader, AMMBench::SparseMatrixLoader, AMMBench::SIFTMatrixLoader, AMMBench::RandomMatrixLoader, AMMBench::PoissonMatrixLoader, AMMBench::MtxMatrixLoade AMMBench::MNISTMatrixLoader, AMMBench::MediaMillMatrixLoader, AMMBench::GaussianMatrixLoader, AMMBench::ExponentialMatrixLoader, AMMBench::CCAMatrixLoader, AMMBench::BinomialMatrixLoader, and AMMBench::BetaMatrixLoader.

8.13.2.2 getB()

```
torch::Tensor AMMBench::AbstractMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented in AMMBench::ZipfMatrixLoader, AMMBench::ZeroMaskedMatrixLoader, AMMBench::SparseMatrixLoader, AMMBench::SIFTMatrixLoader, AMMBench::RandomMatrixLoader, AMMBench::PoissonMatrixLoader, AMMBench::MtxMatrixLoade AMMBench::MNISTMatrixLoader, AMMBench::MediaMillMatrixLoader, AMMBench::GaussianMatrixLoader, AMMBench::ExponentialMatrixLoader, AMMBench::CCAMatrixLoader, AMMBench::BinomialMatrixLoader, and AMMBench::BetaMatrixLoader.

8.13.2.3 setConfig()

```
bool AMMBench::AbstractMatrixLoader::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 AMMBench::ZipfMatrixLoader, AMMBench::ZeroMaskedMatrixLoader, AMMBench::SparseMatrixLoader, AMMBench::SIFTMatrixLoader, AMMBench::RandomMatrixLoader, AMMBench::PoissonMatrixLoader, AMMBench::MtxMatrixLoade AMMBench::MNISTMatrixLoader, AMMBench::MediaMillMatrixLoader, AMMBench::GaussianMatrixLoader, AMMBench::ExponentialMatrixLoader, AMMBench::CCAMatrixLoader, AMMBench::BinomialMatrixLoader, and AMMBench::BetaMatrixLoader.

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

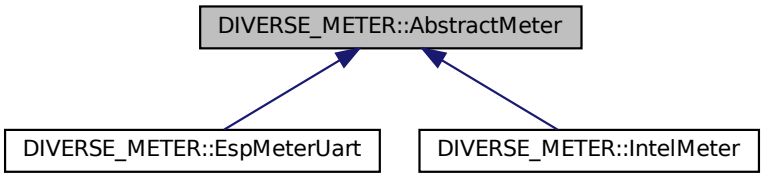
- include/MatrixLoader/AbstractMatrixLoader.h
- src/MatrixLoader/AbstractMatrixLoader.cpp

8.14 DIVERSE_METER::AbstractMeter Class Reference

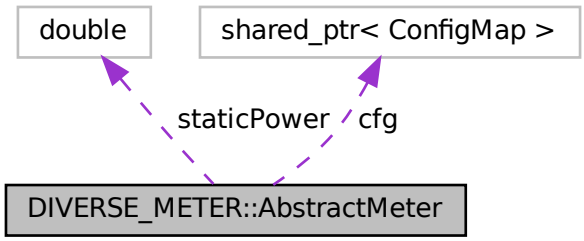
The abstract class for all meters.

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

Inheritance diagram for DIVERSE_METER::AbstractMeter:



Collaboration diagram for DIVERSE_METER::AbstractMeter:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr _cfg)
  to set the configmap
- void setStaticPower (double _sp)
  to manually set the static power
- void testStaticPower (uint64_t sleepingSecond)
  to test the static power of a system by sleeping
- virtual void startMeter ()
  to start the meter into some measuring tasks
- virtual void stopMeter ()
  to stop the meter into some measuring tasks
- virtual double getE ()
  to get the energy in J, including static energy consumption of system
- virtual double getPeak ()

to get the peak power in W, including static power of system

- virtual bool **isValid** ()
- double getStaticPower ()

  to return the tested static power return the staticPower
- double getStaicEnergyConsumption (uint64_t runningUs)

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

Protected Attributes

- double staticPower = 0

  static power of a system in W
- INTELLI::ConfigMapPtr **cfg** = nullptr

8.14.1 Detailed Description

The abstract class for all meters.

Note

default behaviors:

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

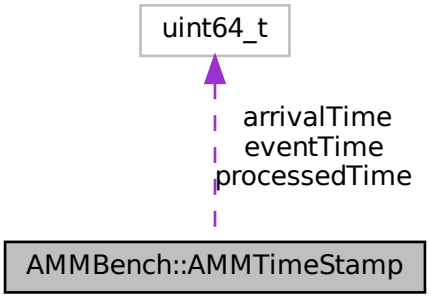
- include/Utils/Meters/AbstractMeter.hpp
- src/Utils/Meters/AbstractMeter.cpp

8.15 AMMBench::AMMTimeStamp Class Reference

The class to define timestamp in streaming.

`#include <Streaming/TimeStamper.h>`

Collaboration diagram for AMMBench::AMMTimeStamp:



Public Member Functions

- **AMMTimeStamp** (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

8.15.1 Detailed Description

The class to define timestamp in streaming.

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

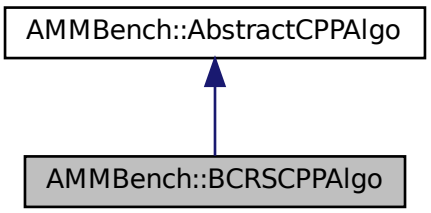
- include/Streaming/TimeStamper.h

8.16 AMMBench::BCRSCPPAlgo Class Reference

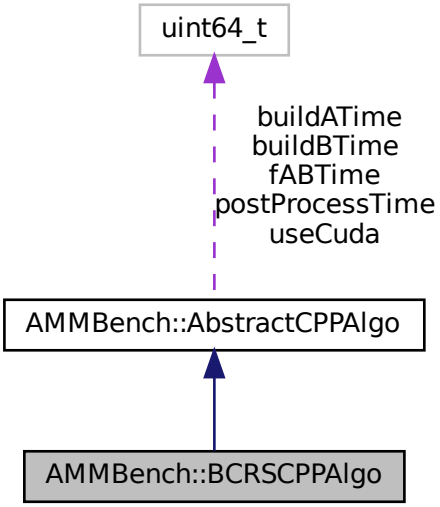
The Bernoulli column row sampling (BCRS) class of c++ algos.

`#include <CPPAlgos/BCRSCPPAlgo.h>`

Inheritance diagram for AMMBench::BCRSCPPAlgo:



Collaboration diagram for AMMBench::BCRSCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.16.1 Detailed Description

The Bernoulli column row sampling (BCRS) class of c++ algos.

++

8.16.2 Member Function Documentation

8.16.2.1 amm()

```
torch::Tensor AMMBench::BCRSCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

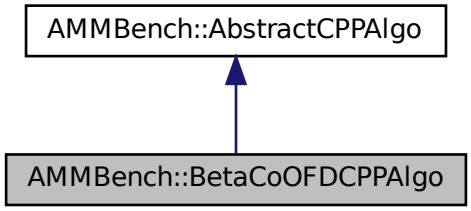
- include/CPPAlgos/BCRSCPPAlgo.h
- src/CPPAlgos/BCRSCPPAlgo.cpp

8.17 AMMBench::BetaCoOFDCPPAlgo Class Reference

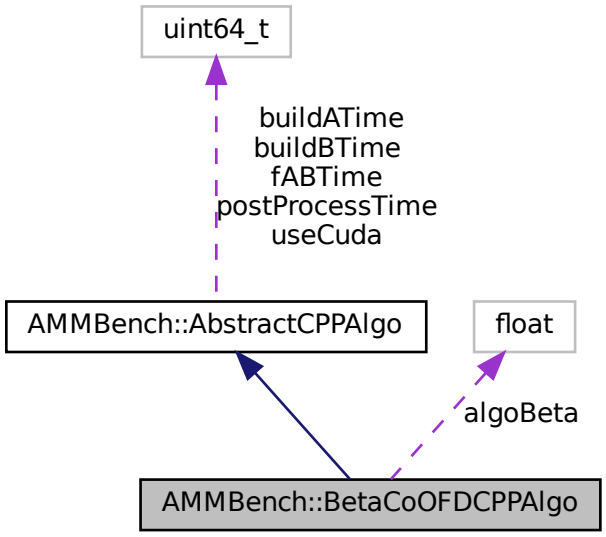
The Beta Co-Occurring FD AMM class of c++ algos.

```
#include <CPPAlgos/BetaCoOFDCPPAlgo.h>
```

Inheritance diagram for AMMBench::BetaCoOFDCPPAlgo:



Collaboration diagram for AMMBench::BetaCoOFDCPPAlgo:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr cfg)
  set the alo-specfic config related to one algorithm
- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Protected Attributes

- float **algoBeta** = 1.0

8.17.1 Detailed Description

The Beta Co-Occurring FD AMM class of c++ algos.

++

Note

parameters

- algoBeta Double, the beta parameters in this algo, default 1.0

8.17.2 Member Function Documentation

8.17.2.1 amm()

```
torch::Tensor AMMBench::BetaCoOFDCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

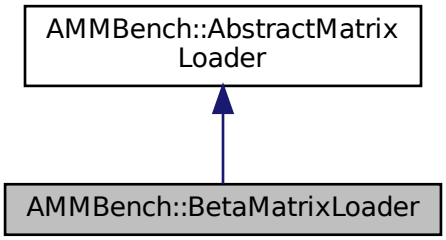
- include/CPPAlgos/BetaCoOFDCPPAlgo.h
- src/CPPAlgos/BetaCoOFDCPPAlgo.cpp

8.18 AMMBench::BetaMatrixLoader Class Reference

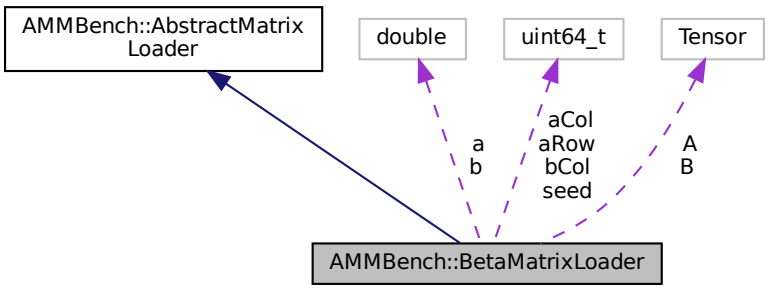
The Beta class of matrix loader.

`#include <MatrixLoader/BetaMatrixLoader.h>`

Inheritance diagram for AMMBench::BetaMatrixLoader:



Collaboration diagram for AMMBench::BetaMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- double **a**
- double **b**

8.18.1 Detailed Description

The Beta class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514
- "a" parameters of beta distribution, Double, 2.0
- "b" parameters of beta distribution, Double, 2.0

: default name tags "random": BetaMatrixLoader

8.18.2 Member Function Documentation

8.18.2.1 getA()

```
torch::Tensor AMMBench::BetaMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.18.2.2 getB()

```
torch::Tensor AMMBench::BetaMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.18.2.3 paraseConfig()

```
void AMMBench::BetaMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.18.2.4 setConfig()

```
bool AMMBench::BetaMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

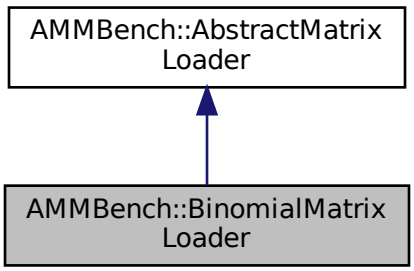
- include/MatrixLoader/BetaMatrixLoader.h
- src/MatrixLoader/BetaMatrixLoader.cpp

8.19 AMMBench::BinomialMatrixLoader Class Reference

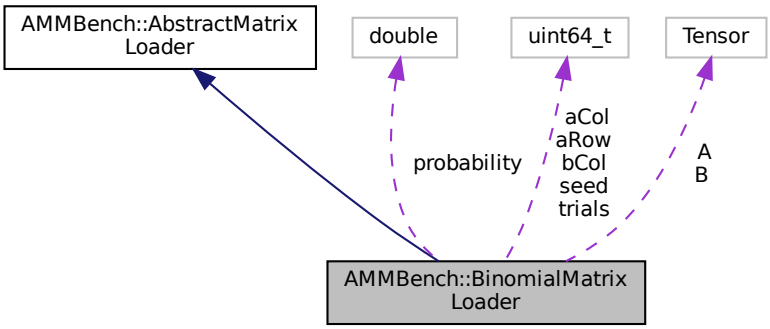
The Binomial class of matrix loader.

`#include <MatrixLoader/BinomialMatrixLoader.h>`

Inheritance diagram for AMMBench::BinomialMatrixLoader:



Collaboration diagram for AMMBench::BinomialMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- uint64_t **trials**
- double **probability**

8.19.1 Detailed Description

The Binomial class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514
- "trials" parameters of binomial distribution, U64, 10
- "probability" parameters of binomial distribution, Double, 0.5

: default name tags "random": BinomialMatrixLoader

8.19.2 Member Function Documentation

8.19.2.1 getA()

```
torch::Tensor AMMBench::BinomialMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.19.2.2 getB()

```
torch::Tensor AMMBench::BinomialMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.19.2.3 paraseConfig()

```
void AMMBench::BinomialMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.19.2.4 setConfig()

```
bool AMMBench::BinomialMatrixLoader::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

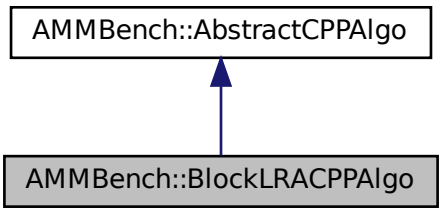
Reimplemented from AMMBench::AbstractMatrixLoader.

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

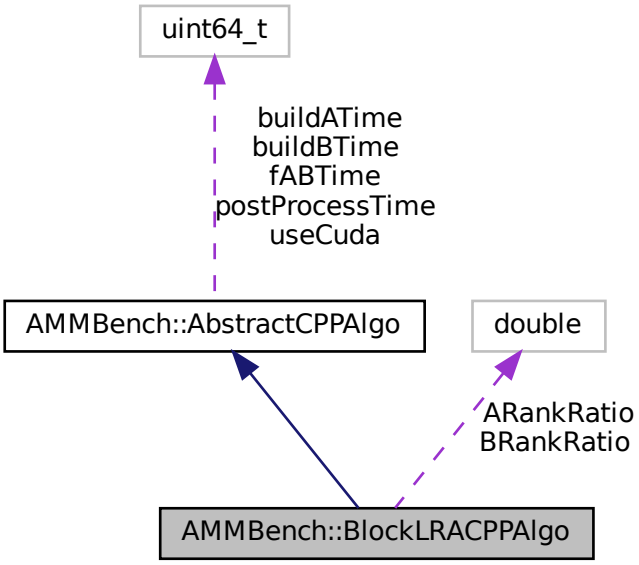
- include/MatrixLoader/BinomialMatrixLoader.h
- src/MatrixLoader/BinomialMatrixLoader.cpp

8.20 AMMBench::BlockLRACPPAlgo Class Reference

Inheritance diagram for AMMBench::BlockLRACPPAlgo:



Collaboration diagram for AMMBench::BlockLRACPPAlgo:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm
- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t blockSize)

  Implementation of paper [IEEE-HPCS 2017] Accelerating Matrix Multiplication in Deep Learning by Using Low-Rank Approximation https://ieeexplore.ieee.org/abstract/document/8035076.

Protected Attributes

- double **ARankRatio** = 0.5
- double **BRankRatio** = 0.5

8.20.1 Member Function Documentation

8.20.1.1 amm()

```
torch::Tensor AMMBench::BlockLRACPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t blockSize )  [virtual]
```

Implementation of paper [IEEE-HPCS 2017] Accelerating Matrix Multiplication in Deep Learning by Using Low-Rank Approximation https://ieeexplore.ieee.org/abstract/document/8035076.

Parameters

A	the A matrix
B	the B matrix
blockSize	the size of block to do SVD

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

8.20.1.2 setConfig()

```
void AMMBench::BlockLRACPPAlgo::setConfig (
            INTELLI::ConfigMapPtr cfg )  [virtual]
```

set the alo-specfic config related to one algorithm

Parameters

ARankRatio	LRA rank ratio over A complete SVD rank
BRankRatio	LRA rank ratio over B complete SVD rank

Reimplemented from AMMBench::AbstractCPPAlgo.

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

- include/CPPAlgos/BlockLRACPPAlgo.h
- src/CPPAlgos/BlockLRACPPAlgo.cpp

8.21 BlockLRACPPIgo Class Reference

The block SVD LRA class of c++ algos.

`#include <CPPAlgos/BlockLRACPPAlgo.h>`

8.21.1 Detailed Description

The block SVD LRA class of c++ algos.

++

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

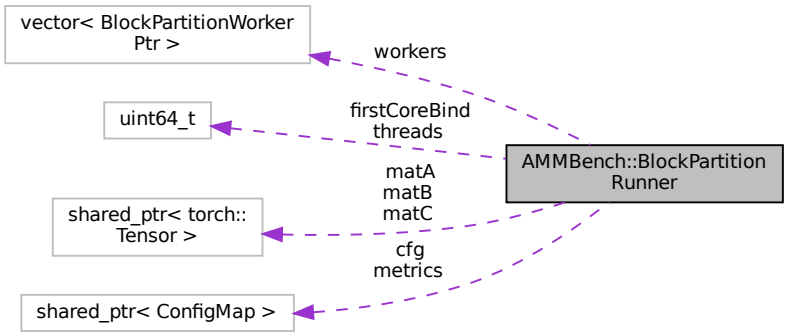
- include/CPPAlgos/BlockLRACPPAlgo.h

8.22 AMMBench::BlockPartitionRunner Class Reference

The top entity to control all workers, see also BlockPartitionWorker. This one works under a simple row partition parallelization.

`#include <Parallelization/BlockPartitionRunner.h>`

Collaboration diagram for AMMBench::BlockPartitionRunner:



Public Member Functions

- void setConfig (INTELLI::ConfigMapPtr _cfg)

  set the config map
- void createABC (torch::Tensor A, torch::Tensor B)

  create the A,B,C matrix and pass it to all workers
- torch::Tensor parallelForward ()

  run a parallel forward of A,B, and return C
- torch::Tensor runAMM (torch::Tensor A, torch::Tensor B)

  conducte the multithread AMM and return
- uint64_t getElapsedTime ()

  get the elapsed time of multithread running
- void appendThreadInfo (INTELLI::ConfigMapPtr ru)

  append the running information of each thread to the result csv
- void calculateMetrics ()

  calculate metrics including the pef result for all threads used in the runner, and elapsed time, throughput..
- INTELLI::ConfigMapPtr getMetrics ()

  get metrics
- virtual INTELLI::ConfigMapPtr getBreakDown ()

  to export the algorithm breakdown

Protected Attributes

- INTELLI::ConfigMapPtr **cfg**
- uint64_t **threads** = 0
- TensorPtr matA = nullptr

  Input matrix A.
- TensorPtr matB = nullptr

  Input matrix B.
- TensorPtr matC = nullptr

  OUTput matrix C.
- std::vector< BlockPartitionWorkerPtr > **workers**
- uint64_t firstCoreBind = 0

  special bind of first core, if need
- INTELLI::ConfigMapPtr **metrics** = newConfigMap()

8.22.1 Detailed Description

The top entity to control all workers, see also BlockPartitionWorker. This one works under a simple row partition parallelization.

Note

parameters

- threads, U64, the number of worker threads, default 2
- osScheduling, U64, whether use default os scheduling instead of my own core bind, default 0
- firstCoreBind, U64, which core will the first thread be bound to, default 0

default behaviors

- create
- call setConfig
- call runAMM and return result
- call getElapsedTime
- call getMetrics

8.22.2 Member Function Documentation

8.22.2.1 appendThreadInfo()

```
void AMMBench::BlockPartitionRunner::appendThreadInfo (
            INTELLI::ConfigMapPtr ru )
```

append the running information of each thread to the result csv

Parameters

ru	The result csv to be appended

8.22.2.2 createABC()

```
void AMMBench::BlockPartitionRunner::createABC (
            torch::Tensor A,
            torch::Tensor B )
```

create the A,B,C matrix and pass it to all workers

Parameters

A	The A matrix
B	The B matrix @warnning call after setConfig

8.22.2.3 getBreakDown()

```
INTELLI::ConfigMapPtr AMMBench::BlockPartitionRunner::getBreakDown ( )  [virtual]
```

to export the algorithm breakdown

Note

only valid for c++ algo

Returns

the key-value table breakdown in ConfigMapPtr;

8.22.2.4 getElapsedTime()

```
uint64_t AMMBench::BlockPartitionRunner::getElapsedTime ( )
```

get the elapsed time of multithread running

Returns

the elapsed time

Note

Exclude the overhead of cleaning thread states such as loaded module

8.22.2.5 getMetrics()

```
INTELLI::ConfigMapPtr AMMBench::BlockPartitionRunner::getMetrics ( )
```

get metrics

Returns

metrics ConfigMapPtr

8.22.2.6 parallelForward()

```
torch::Tensor AMMBench::BlockPartitionRunner::parallelForward ( )
```

run a parallel forward of A,B, and return C

Returns

C=matA∗matB @warnning call after createABC

8.22.2.7 runAMM()

```
torch::Tensor AMMBench::BlockPartitionRunner::runAMM (
            torch::Tensor A,
            torch::Tensor B )
```

conducte the multithread AMM and return

Parameters

A	The A matrix
B	The B matrix

Returns

The AMM(A,B) @warnning call after setConfig

8.22.2.8 setConfig()

```
void AMMBench::BlockPartitionRunner::setConfig (
            INTELLI::ConfigMapPtr _cfg )
```

set the config map

Parameters

_cfg	

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

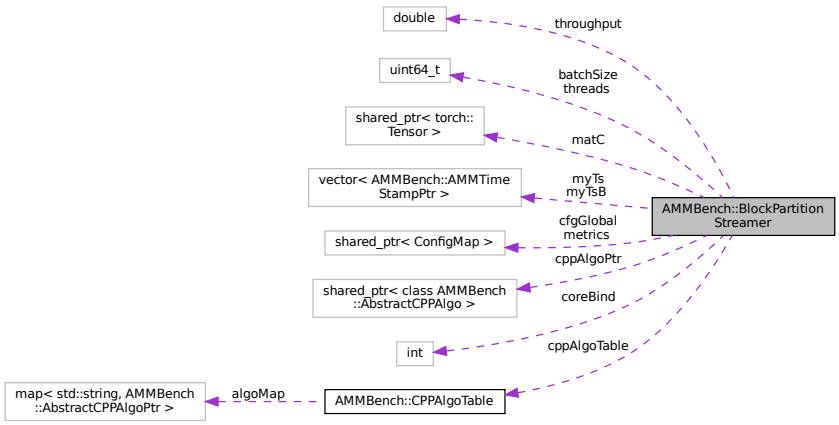
- include/Parallelization/BlockPartitionRunner.h
- src/Parallelization/BlockPartitionRunner.cpp

8.23 AMMBench::BlockPartitionStreamer Class Reference

The class to run streaming amm under block partition scheme, let rows of A coming in a streaming manner, all of which are partitioned with BlockPartitionRunner.

`#include <Streaming/BlockPartitionStreamer.h>`

Collaboration diagram for AMMBench::BlockPartitionStreamer:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this TimerStamper.
- virtual torch::Tensor streamingAmm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize=1)

  To run a streaming Amm, assuming the rows of A coming in a streaming manner and B is fixed.
- virtual torch::Tensor streamingAmm2S (torch::Tensor A, torch::Tensor B, uint64_t sketchSize=1)

  To run a streaming Amm, assuming the rows of A coming in a streaming manner and the cols of B coming in a streaming manner.
- double getThroughput ()

  to get the throughput of last streaming process, the unit is rows/second
- double getLatencyPercentage (double fraction)

  to get the latency within some fraction, such as 0.95
- INTELLI::ConfigMapPtr getMetrics ()

  get metrics (including the pef result for all threads used in the runner, and elapsed time, throughput..)

Public Attributes

- std::vector< AMMBench::AMMTimeStampPtr > myTs

  the timestamps to trace the streaming process
- std::vector< AMMBench::AMMTimeStampPtr > myTsB

  the additional timestamps to trace the streaming process, if B is also stream

Protected Attributes

- INTELLI::ConfigMapPtr **cfgGlobal**
- AMMBench::CPPAlgoTable **cppAlgoTable**
- uint64_t **batchSize** = 1
- AMMBench::AbstractCPPAlgoPtr **cppAlgoPtr** = nullptr
- AMMBench::TensorPtr **matC** = nullptr
- double **throughput** = 0.0
- uint64_t **threads** = 1
- int **coreBind**
- INTELLI::ConfigMapPtr **metrics** = newConfigMap()

8.23.1 Detailed Description

The class to run streaming amm under block partition scheme, let rows of A coming in a streaming manner, all of which are partitioned with BlockPartitionRunner.

Note

Default behavior

- create
- call setConfig, this will also determine how to generate time stamp and config will be passed to TimeStamper
- run streaming amm:
  - call streamingAmm, if only A matrix will be streamed
  - call streamingAmm2S, if both A and B will be streamed
- call getThroughput, and getLatencyPercentage to get the streaming performance

8.23.2 Member Function Documentation

8.23.2.1 getLatencyPercentage()

```
double AMMBench::BlockPartitionStreamer::getLatencyPercentage (
            double fraction )
```

to get the latency within some fraction, such as 0.95

Parameters

fraction	the 0∼1 fraction

Returns

the latency in us

8.23.2.2 getMetrics()

```
INTELLI::ConfigMapPtr AMMBench::BlockPartitionStreamer::getMetrics ( )  [inline]
```

get metrics (including the pef result for all threads used in the runner, and elapsed time, throughput..)

Returns

metrics ConfigMapPtr

8.23.2.3 getThroughput()

```
double AMMBench::BlockPartitionStreamer::getThroughput ( )  [inline]
```

to get the throughput of last streaming process, the unit is rows/second

Returns

the throughput

8.23.2.4 setConfig()

```
bool AMMBench::BlockPartitionStreamer::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

1.set the algo

1. set the batch size

8.23.2.5 streamingAmm()

```
torch::Tensor AMMBench::BlockPartitionStreamer::streamingAmm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize = 1 )  [virtual]
```

To run a streaming Amm, assuming the rows of A coming in a streaming manner and B is fixed.

Parameters

A	The A matrix
B	The B matrix

Returns

bool whether the config is successfully set

update the indexes

8.23.2.6 streamingAmm2S()

```
torch::Tensor AMMBench::BlockPartitionStreamer::streamingAmm2S (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize = 1 )  [virtual]
```

To run a streaming Amm, assuming the rows of A coming in a streaming manner and the cols of B coming in a streaming manner.

Parameters

A	The A matrix
B	The B matrix

Returns

bool whether the config is successfully set

now, the whole batch has arrived, compute

do the incomingA∗newArrivedB part

do the oldArrivedA∗incomingB part

update the indexes

The latency calculation is different from one stream case here, as older A will still be probed by newer B

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

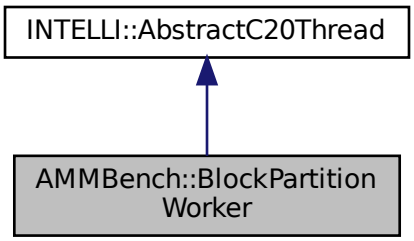
- include/Streaming/BlockPartitionStreamer.h
- src/Streaming/BlockPartitionStreamer.cpp

8.24 AMMBench::BlockPartitionWorker Class Reference

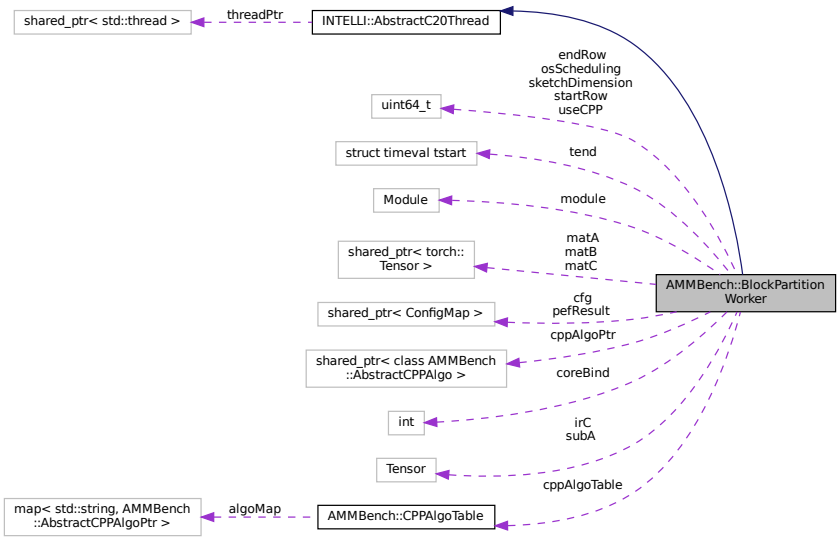
The basic partition worker.

`#include <Parallelization/BlockPartitionRunner.h>`

Inheritance diagram for AMMBench::BlockPartitionWorker:



Collaboration diagram for AMMBench::BlockPartitionWorker:



Public Member Functions

- void setConfig (INTELLI::ConfigMapPtr _cfg)

  set the config map
- void setABC (TensorPtr A, TensorPtr B, TensorPtr C)

  set the pointer to A,B,C matrix
- void setWorkParameters (uint64_t aStart, uint64_t aEnd, int mycore)

  set work parmeters
- void **setCoreBInd** (int cno)
- uint64_t **getElapsedTime** ()
- INTELLI::ConfigMapPtr **getPefResult** ()
- virtual INTELLI::ConfigMapPtr getBreakDown ()

  to export the algorithm breakdown

Public Attributes

- torch::Tensor **irC**
- torch::Tensor **subA**
- uint64_t **startRow** = 0
- uint64_t **endRow** = 0

Protected Member Functions

- virtual void inlineMain ()

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

Protected Attributes

- AMMBench::CPPAlgoTable **cppAlgoTable**
- struct timeval tstart **tend**
- uint64_t **useCPP** = 0
- uint64_t **osScheduling** = 0
- AMMBench::AbstractCPPAlgoPtr **cppAlgoPtr** = nullptr
- TensorPtr matA = nullptr

  Input matrix A.
- TensorPtr matB = nullptr

  Input matrix B.
- TensorPtr matC = nullptr

  OUTput matrix C.
- INTELLI::ConfigMapPtr **cfg**
- torch::jit::script::Module **module**
- uint64_t **sketchDimension** = 0
- int **coreBind**
- INTELLI::ConfigMapPtr **pefResult**

8.24.1 Detailed Description

The basic partition worker.

8.24.2 Member Function Documentation

8.24.2.1 getBreakDown()

```
INTELLI::ConfigMapPtr AMMBench::BlockPartitionWorker::getBreakDown ( )  [virtual]
```

to export the algorithm breakdown

Note

only valid for c++ algo

Returns

the key-value table breakdown in ConfigMapPtr;

8.24.2.2 inlineMain()

```
void AMMBench::BlockPartitionWorker::inlineMain ( )  [protected], [virtual]
```

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

Note

Normally re-write this in derived classes

1. bind core and torch setting
2. multiply sub-matrix of A

Reimplemented from INTELLI::AbstractC20Thread.

8.24.2.3 setConfig()

```
void AMMBench::BlockPartitionWorker::setConfig (
            INTELLI::ConfigMapPtr _cfg )
```

set the config map

Parameters

_cfg	

8.24.2.4 setWorkParameters()

```
void AMMBench::BlockPartitionWorker::setWorkParameters (
            uint64_t aStart,
            uint64_t aEnd,
            int mycore )
```

set work parmeters

Parameters

aStart	The start row in A
aEnd	The end row in A
mycore	the core to be binded

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

- include/Parallelization/BlockPartitionRunner.h
- src/Parallelization/BlockPartitionRunner.cpp

8.25 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.

8.25.1 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.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 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.

8.25.3 Member Function Documentation

8.25.3.1 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.

8.25.3.2 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.

8.25.3.3 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.

8.25.3.4 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.

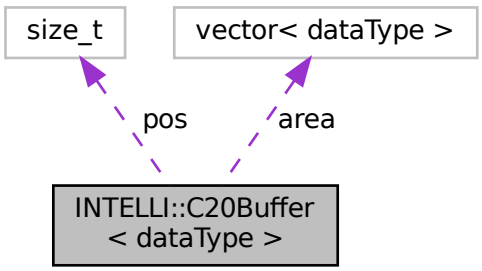
The documentation for this class was generated from the following file:

- include/Utils/BS_thread_pool.hpp

8.26 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

8.26.1 Detailed Description

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

Template Parameters

dataType	The type of your buffering element

8.26.2 Constructor & Destructor Documentation

8.26.2.1 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

8.26.3 Member Function Documentation

8.26.3.1 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

8.26.3.2 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

8.26.3.3 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

8.26.3.4 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

8.26.3.5 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

8.26.3.6 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:

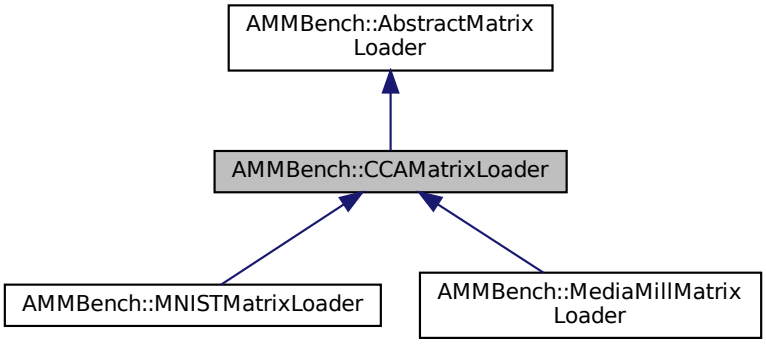
- include/Utils/C20Buffers.hpp

8.27 AMMBench::CCAMatrixLoader Class Reference

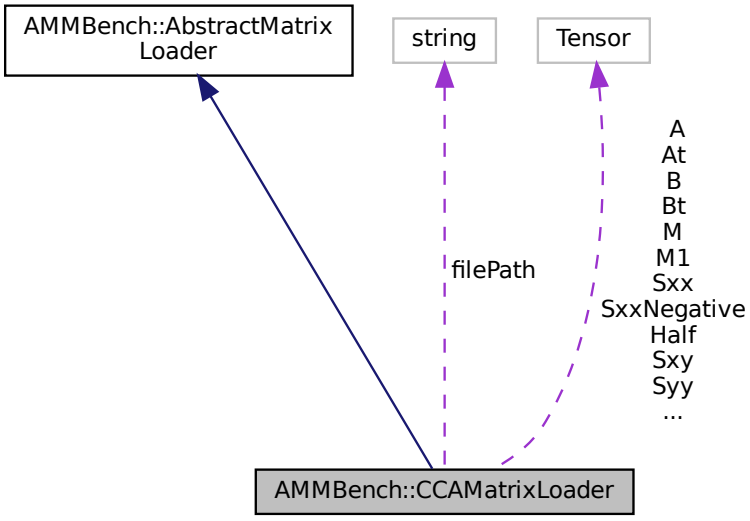
For CCA downstream task.

`#include <MatrixLoader/CCAMatrixLoader.h>`

Inheritance diagram for AMMBench::CCAMatrixLoader:



Collaboration diagram for AMMBench::CCAMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

Set the GLOBAL config map related to this loader.

- virtual void calculate_correlation ()

  Calulate the correlation by mm, and generate tensor Sxx, Sxy, Syy, M, correlation.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix
- virtual torch::Tensor getAt ()

  get the transpose of A matrix
- virtual torch::Tensor getBt ()

  get the transpose of B matrix
- virtual torch::Tensor getSxx ()

  get the Sxx matrix
- virtual torch::Tensor getSyy ()

  get the Sxyymatrix
- virtual torch::Tensor getSxy ()

  get the Sxy matrix
- virtual torch::Tensor getSxxNegativeHalf ()

  get the SxxNegativeHalf matrix
- virtual torch::Tensor getSyyNegativeHalf ()

  get the SyyNegativeHalf matrix
- virtual torch::Tensor getM ()

  M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)
- virtual torch::Tensor getM1 ()

  M1 = mm(SxxNegativeHalf.t(), Sxy)
- virtual torch::Tensor getCorrelation ()

  get the correlation value

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- std::string **filePath**
- torch::Tensor **A**
- torch::Tensor **B**
- torch::Tensor **At**
- torch::Tensor **Bt**
- torch::Tensor **Sxx**
- torch::Tensor **Syy**
- torch::Tensor **Sxy**
- torch::Tensor **SxxNegativeHalf**
- torch::Tensor **SyyNegativeHalf**
- torch::Tensor **M**
- torch::Tensor **M1**
- torch::Tensor **correlation**

8.27.1 Detailed Description

For CCA downstream task.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: does not need config

: default name tags "CCA": CCAMatrixLoader

8.27.2 Member Function Documentation

8.27.2.1 getA()

```
torch::Tensor AMMBench::CCAMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.2 getAt()

```
torch::Tensor AMMBench::CCAMatrixLoader::getAt ( )  [virtual]
```

get the transpose of A matrix

Returns

the A.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.3 getB()

```
torch::Tensor AMMBench::CCAMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.4 getBt()

```
torch::Tensor AMMBench::CCAMatrixLoader::getBt ( )  [virtual]
```

get the transpose of B matrix

Returns

the B.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.5 getCorrelation()

```
torch::Tensor AMMBench::CCAMatrixLoader::getCorrelation ( )  [virtual]
```

get the correlation value

Returns

the generated correlation by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.6 getM()

```
torch::Tensor AMMBench::CCAMatrixLoader::getM ( )  [virtual]
```

M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)

Returns

the generated M matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.7 getM1()

```
torch::Tensor AMMBench::CCAMatrixLoader::getM1 ( )  [virtual]
```

M1 = mm(SxxNegativeHalf.t(), Sxy)

Returns

the generated M1 matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.8 getSxx()

```
torch::Tensor AMMBench::CCAMatrixLoader::getSxx ( )  [virtual]
```

get the Sxx matrix

Returns

the generated Sxx matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.9 getSxxNegativeHalf()

```
torch::Tensor AMMBench::CCAMatrixLoader::getSxxNegativeHalf ( )  [virtual]
```

get the SxxNegativeHalf matrix

Returns

the generated SxxNegativeHalf matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.10 getSxy()

```
torch::Tensor AMMBench::CCAMatrixLoader::getSxy ( )  [virtual]
```

get the Sxy matrix

Returns

the generated Sxy matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.11 getSyy()

```
torch::Tensor AMMBench::CCAMatrixLoader::getSyy ( )  [virtual]
```

get the Sxyymatrix

Returns

the generated Syy matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.12 getSyyNegativeHalf()

```
torch::Tensor AMMBench::CCAMatrixLoader::getSyyNegativeHalf ( )  [virtual]
```

get the SyyNegativeHalf matrix

Returns

the generated SyyNegativeHalf matrix by calling calculate_correlation()

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

8.27.2.13 paraseConfig()

```
void AMMBench::CCAMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.27.2.14 setConfig()

```
bool AMMBench::CCAMatrixLoader::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 AMMBench::AbstractMatrixLoader.

Reimplemented in AMMBench::MNISTMatrixLoader, and AMMBench::MediaMillMatrixLoader.

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

- include/MatrixLoader/CCAMatrixLoader.h
- src/MatrixLoader/CCAMatrixLoader.cpp

8.28 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

8.29 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

8.30 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

8.31 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

8.32 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

8.33 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

8.34 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

8.35 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

8.36 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

8.37 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

8.38 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

8.39 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

8.40 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

8.41 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

8.42 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

8.43 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

8.44 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

8.45 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

8.46 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

8.47 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

8.48 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

8.49 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

8.50 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

8.51 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

8.52 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

8.53 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

8.54 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

8.55 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

8.56 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

8.57 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

8.58 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

8.59 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

8.60 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

8.61 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

8.62 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

8.63 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

8.64 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

8.65 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

8.66 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

8.67 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

8.68 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

8.69 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

8.70 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

8.71 cl_ushort8 Union Reference

Public Member Functions

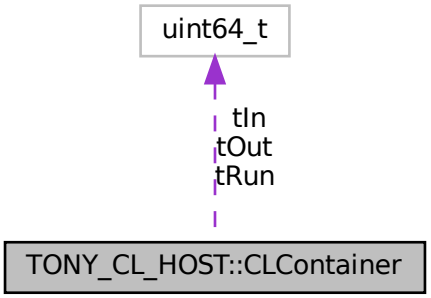
- cl_ushort **CL_ALIGNED** (16) s[8]

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

- include/CL/cl_platform.h

8.72 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:

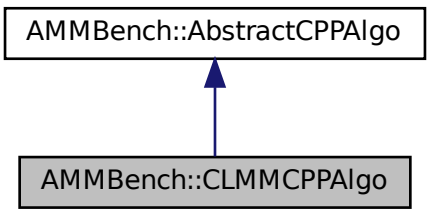
- include/CL/CLContainer.hpp
- src/CLContainer.cpp

8.73 AMMBench::CLMMCPPAlgo Class Reference

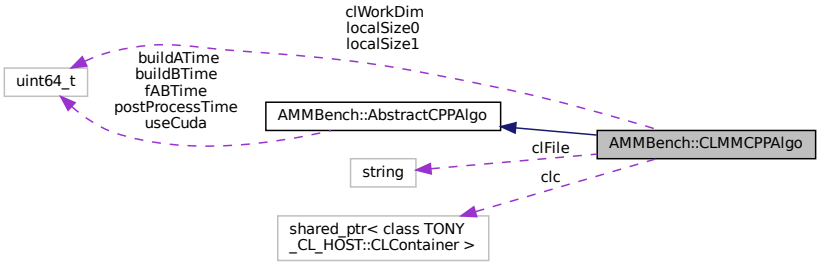
The MM class of c++ algos using opencl.

`#include <CPPAlgos/CLMMCPPAlgo.h>`

Inheritance diagram for AMMBench::CLMMCPPAlgo:



Collaboration diagram for AMMBench::CLMMCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes
- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm

Protected Member Functions

- torch::Tensor clmm (torch::Tensor A, torch::Tensor B)
- torch::Tensor clint8 (torch::Tensor A, torch::Tensor B)

Protected Attributes

- std::string **clFile** = "CL/CLMM"
- uint64_t **clWorkDim** = 2
- TONY_CL_HOST::CLContainerPtr **clc** = nullptr
- uint64_t **localSize0** = 1
- uint64_t **localSize1** = 1

8.73.1 Detailed Description

The MM class of c++ algos using opencl.

++

Note

additionally parameters

- clFile, String, default "CL/CLMM"

8.73.2 Member Function Documentation

8.73.2.1 amm()

```
torch::Tensor AMMBench::CLMMCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

8.73.2.2 clint8()

```
torch::Tensor AMMBench::CLMMCPPAlgo::clint8 (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

build A into std vector

run fAB

fix the time measure related to cl

8.73.2.3 clmm()

```
torch::Tensor AMMBench::CLMMCPPAlgo::clmm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

build A into std vector

fix the time measure related to cl

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

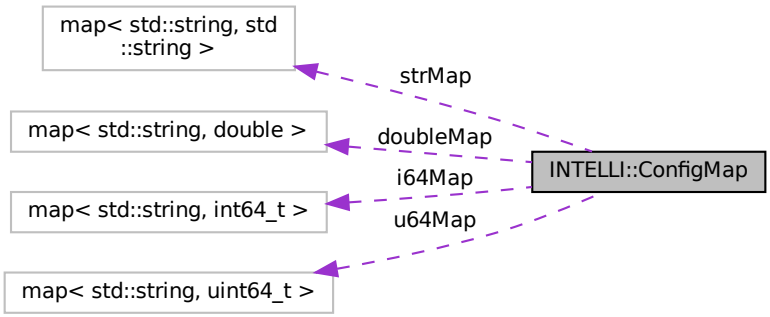
- include/CPPAlgos/CLMMCPPAlgo.h
- src/CPPAlgos/CLMMCPPAlgo.cpp

8.74 INTELLI::ConfigMap Class Reference

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

`#include <Utils/ConfigMap.hpp>`

Collaboration diagram for INTELLI::ConfigMap:



Public Member Functions

- void edit (std::string key, uint64_t value)

  Edit the config map. If not exit the config, will create new, or will overwrite.
- void edit (std::string key, int64_t value)

  Edit the config map. If not exit the config, will create new, or will overwrite.
- void edit (std::string key, double value)

  Edit the config map. If not exit the config, will create new, or will overwrite.
- void edit (std::string key, std::string value)

  Edit the config map. If not exit the config, will create new, or will overwrite.
- bool existU64 (std::string key)

  To detect whether the key exists and related to a U64.
- bool existI64 (std::string key)

  To detect whether the key exists and related to a I64.
- bool existDouble (std::string key)

  To detect whether the key exists and related to a double.
- bool existString (std::string key)

  To detect whether the key exists and related to a std::string.
- bool exist (std::string key)

  To detect whether the key exists.
- uint64_t getU64 (std::string key)

  To get a U64 value by key.
- int64_t getI64 (std::string key)

  To get a I64 value by key.
- double getDouble (std::string key)

  To get a double value by key.
- std::string getString (std::string key)

To get a std::string value by key.

- std::string toString (std::string separator="\t", std::string newLine="\n")

  convert the whole map to std::string and retuen
- void cloneInto (ConfigMap &dest)

  clone this config into destination
- bool toFile (std::string fname, std::string separator=",", std::string newLine="\n")

  convert the whole map to file
- bool fromFile (std::string fname, std::string separator=",", std::string newLine="\n")

  update the whole map from file
- 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.
- 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.
- std::map< std::string, std::string > getStrMap ()

  return the map of string
- void addPrefixToKeys (std::string prefix)

  Add prefix to the front of keys, it is useful in downstream task where we need to generate metric config file for each components in the downstream task e.g. instructions -> ${prefix}Instructions.

Protected Member Functions

- 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**

8.74.1 Detailed Description

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

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

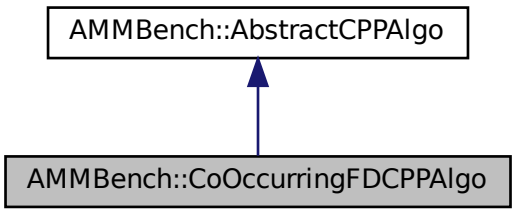
- include/Utils/ConfigMap.hpp

8.75 AMMBench::CoOccurringFDCPPAlgo Class Reference

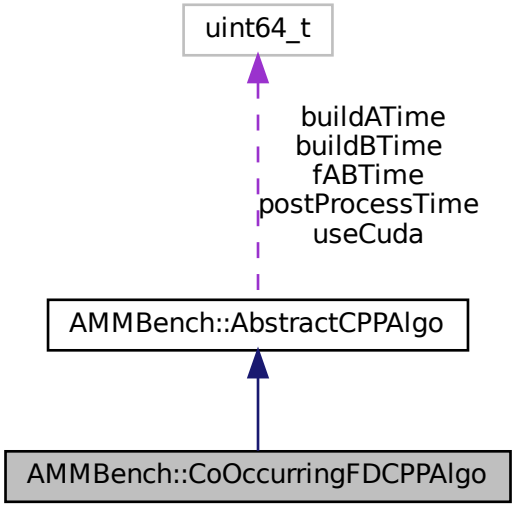
The Co-Occurring FD AMM class of c++ algos.

`#include <CPPAlgos/CoOccurringFDCPPAlgo.h>`

Inheritance diagram for AMMBench::CoOccurringFDCPPAlgo:



Collaboration diagram for AMMBench::CoOccurringFDCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.75.1 Detailed Description

The Co-Occurring FD AMM class of c++ algos.

++

8.75.2 Member Function Documentation

8.75.2.1 amm()

```
torch::Tensor AMMBench::CoOccurringFDCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

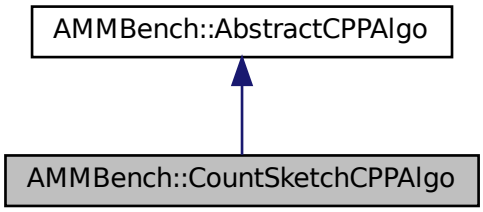
- include/CPPAlgos/CoOccurringFDCPPAlgo.h
- src/CPPAlgos/CoOccurringFDCPPAlgo.cpp

8.76 AMMBench::CountSketchCPPAlgo Class Reference

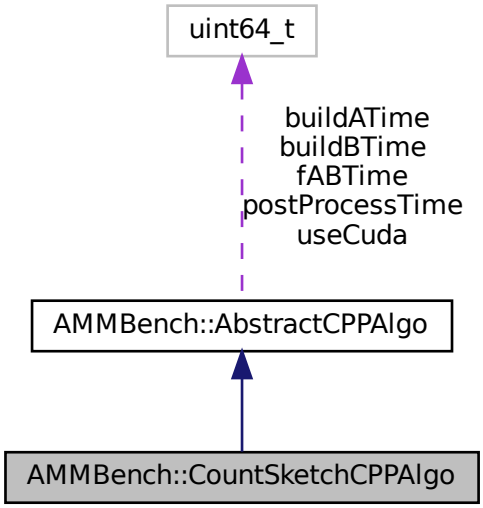
The counter sketch class of c++ algos.

```
#include <CPPAlgos/CountSketchCPPAlgo.h>
```

Inheritance diagram for AMMBench::CountSketchCPPAlgo:



Collaboration diagram for AMMBench::CountSketchCPPAlgo:



Public Member Functions

- torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.76.1 Detailed Description

The counter sketch class of c++ algos.

++

8.76.2 Member Function Documentation

8.76.2.1 amm()

```
torch::Tensor AMMBench::CountSketchCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

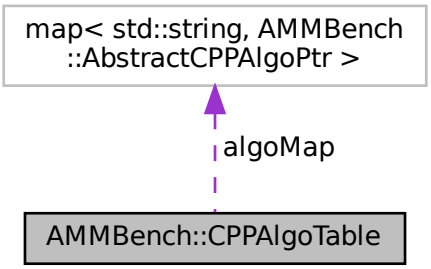
- include/CPPAlgos/CountSketchCPPAlgo.h
- src/CPPAlgos/CountSketchCPPAlgo.cpp

8.77 AMMBench::CPPAlgoTable Class Reference

The table to index cpp algos.

`#include <CPPAlgos/CPPAlgoTable.h>`

Collaboration diagram for AMMBench::CPPAlgoTable:



Public Member Functions

- void registerNewCppAlgo (AMMBench::AbstractCPPAlgoPtr anew, std::string tag)

  To register a new ALGO.
- AMMBench::AbstractCPPAlgoPtr findCppAlgo (std::string name)

  find a dataloader in the table according to its name

Protected Attributes

- std::map< std::string, AMMBench::AbstractCPPAlgoPtr > **algoMap**

8.77.1 Detailed Description

The table to index cpp algos.

++

Note

Default behavior

- create
- (optional) call registerNewCppAlgo for new algo
- find a loader by findCppAlgo using its tag

default tags

- mm AbstractCPPAlgo (default matmul)
- crs CRSCPPAlgo (the column-row-sampling, crs)

8.77.2 Member Function Documentation

8.77.2.1 findCppAlgo()

```
AMMBench::AbstractCPPAlgoPtr AMMBench::CPPAlgoTable::findCppAlgo (
            std::string name )  [inline]
```

find a dataloader in the table according to its name

Parameters

name	The nameTag of loader

Returns

The AbstractCppAlgoPtr, nullptr if not found

8.77.2.2 registerNewCppAlgo()

```
void AMMBench::CPPAlgoTable::registerNewCppAlgo (
            AMMBench::AbstractCPPAlgoPtr anew,
            std::string tag ) [inline]
```

To register a new ALGO.

Parameters

anew	The new algo
tag	THe name tag

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

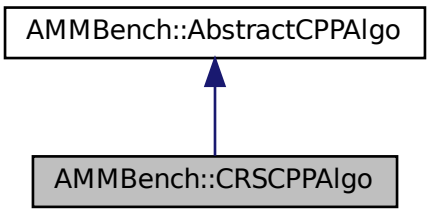
- include/CPPAlgos/CPPAlgoTable.h
- src/CPPAlgos/CPPAlgoTable.cpp

8.78 AMMBench::CRSCPPAlgo Class Reference

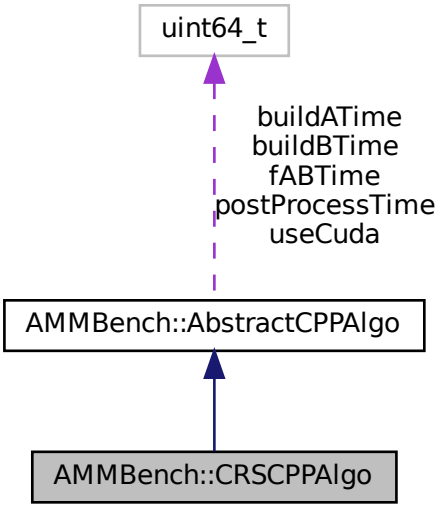
The column row sampling (CRS) class of c++ algos.

`#include <CPPAlgos/CRSCPPAlgo.h>`

Inheritance diagram for AMMBench::CRSCPPAlgo:



Collaboration diagram for AMMBench::CRSCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.78.1 Detailed Description

The column row sampling (CRS) class of c++ algos.

++

8.78.2 Member Function Documentation

8.78.2.1 amm()

```
torch::Tensor AMMBench::CRSCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

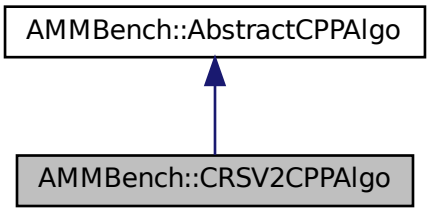
- include/CPPAlgos/CRSCPPAlgo.h
- src/CPPAlgos/CRSCPPAlgo.cpp

8.79 AMMBench::CRSV2CPPAlgo Class Reference

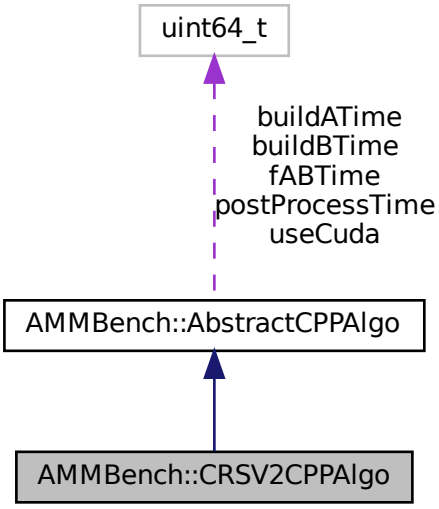
The column row sampling (CRS) class of c++ algos, a second implementation.

`#include <CPPAlgos/CRSV2CPPAlgo.h>`

Inheritance diagram for AMMBench::CRSV2CPPAlgo:



Collaboration diagram for AMMBench::CRSV2CPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.79.1 Detailed Description

The column row sampling (CRS) class of c++ algos, a second implementation.

++

8.79.2 Member Function Documentation

8.79.2.1 amm()

```
torch::Tensor AMMBench::CRSV2CPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

- include/CPPAlgos/CRSV2CPPAlgo.h
- src/CPPAlgos/CRSV2CPPAlgo.cpp

8.80 default_attrs Struct Reference

The low-level perf descriptions passed to OS.

`#include <ThreadPerf.hpp>`

8.80.1 Detailed Description

The low-level perf descriptions passed to OS.

The low-level perf events send to OS call, don't touch me.

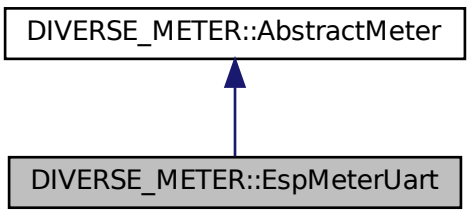
The documentation for this struct was generated from the following file:

- include/Utils/ThreadPerf.hpp

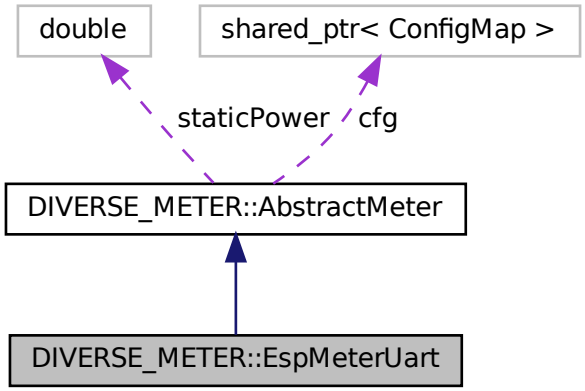
8.81 DIVERSE_METER::EspMeterUart Class Reference

the entity of an esp32s2-based power meter, connected by uart 115200

Inheritance diagram for DIVERSE_METER::EspMeterUart:



Collaboration diagram for DIVERSE_METER::EspMeterUart:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr _cfg)
  to set the configmap
- void startMeter ()
  to start the meter into some measuring tasks
- void stopMeter ()
  to stop the meter into some measuring tasks
- double getE ()
  to get the energy in J, including static energy consumption of system
- double getPeak ()
  to get the peak power in W, including static power of system
- bool **isValid** ()

Additional Inherited Members

8.81.1 Detailed Description

the entity of an esp32s2-based power meter, connected by uart 115200

Note

default behaviors:

- create
- call setConfig() to config this meter
- (optional) call testStaticPower() to test the static power of a device, if you want to exclude it
- call startMeter() to start measurement
- (run your program)

- call stopMeter() to stop measurement
- call getE(), getPeak(), etc to get the measurement resluts

config parameters:

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

tag is "espUart"

8.81.2 Member Function Documentation

8.81.2.1 setConfig()

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

to set the configmap

Parameters

cfg	the config map

Reimplemented from DIVERSE_METER::AbstractMeter.

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

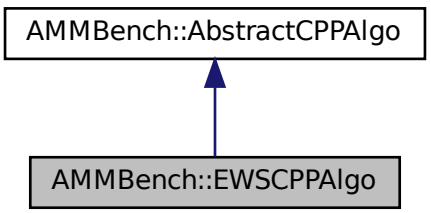
- include/Utils/Meters/EspMeterUart/EspMeterUart.hpp
- src/Utils/Meters/EspMeterUart/EspMeterUart.cpp

8.82 AMMBench::EWSCPPAlgo Class Reference

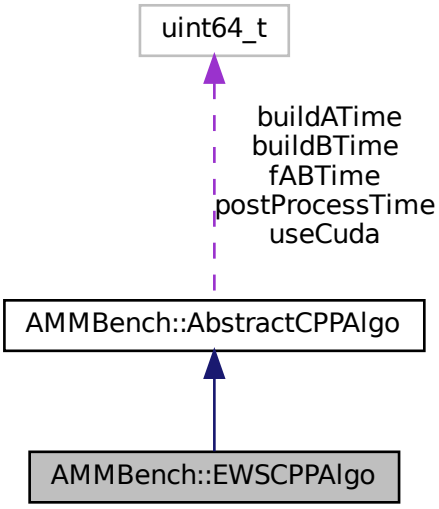
The Element Wise Sampling (EWS) class of c++ algos.

`#include <CPPAlgos/EWSCPPAlgo.h>`

Inheritance diagram for AMMBench::EWSCPPAlgo:



Collaboration diagram for AMMBench::EWSCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.82.1 Detailed Description

The Element Wise Sampling (EWS) class of c++ algos.

++

8.82.2 Member Function Documentation

8.82.2.1 amm()

```
torch::Tensor AMMBench::EWSCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

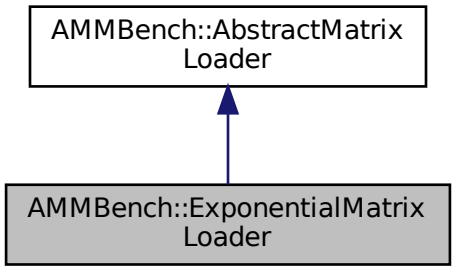
- include/CPPAlgos/EWSCPPAlgo.h
- src/CPPAlgos/EWSCPPAlgo.cpp

8.83 AMMBench::ExponentialMatrixLoader Class Reference

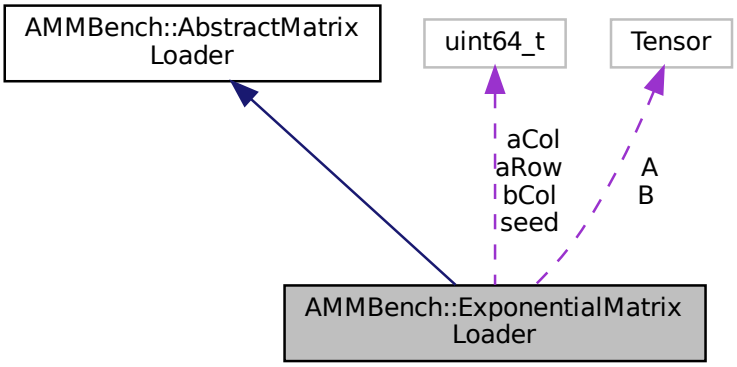
The Exponential class of matrix loader.

`#include <MatrixLoader/ExponentialMatrixLoader.h>`

Inheritance diagram for AMMBench::ExponentialMatrixLoader:



Collaboration diagram for AMMBench::ExponentialMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**

8.83.1 Detailed Description

The Exponential class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514

: default name tags "random": ExponentialMatrixLoader

8.83.2 Member Function Documentation

8.83.2.1 getA()

```
torch::Tensor AMMBench::ExponentialMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.83.2.2 getB()

```
torch::Tensor AMMBench::ExponentialMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.83.2.3 paraseConfig()

```
void AMMBench::ExponentialMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.83.2.4 setConfig()

```
bool AMMBench::ExponentialMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

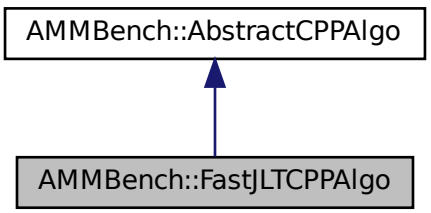
- include/MatrixLoader/ExponentialMatrixLoader.h
- src/MatrixLoader/ExponentialMatrixLoader.cpp

8.84 AMMBench::FastJLTCPPAlgo Class Reference

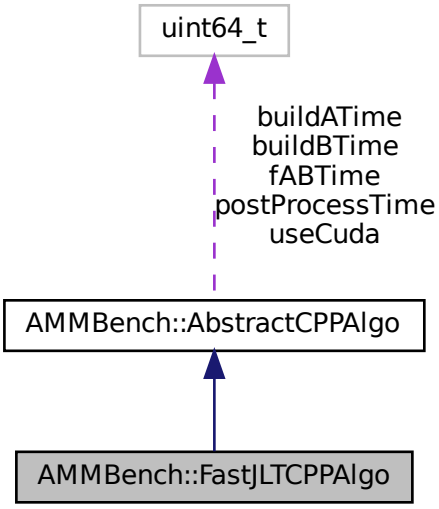
The tug of war class of c++ algoS.

`#include <CPPAlgos/FastJLTCPPAlgo.h>`

Inheritance diagram for AMMBench::FastJLTCPPAlgo:



Collaboration diagram for AMMBench::FastJLTCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.84.1 Detailed Description

The tug of war class of c++ algoS.

++

8.84.2 Member Function Documentation

8.84.2.1 amm()

```
torch::Tensor AMMBench::FastJLTCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

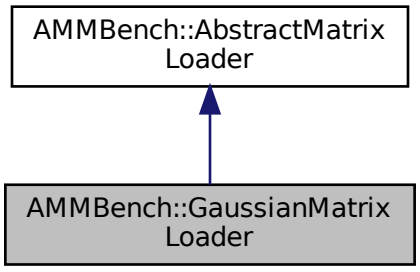
- include/CPPAlgos/FastJLTCPPAlgo.h
- src/CPPAlgos/FastJLTCPPAlgo.cpp

8.85 AMMBench::GaussianMatrixLoader Class Reference

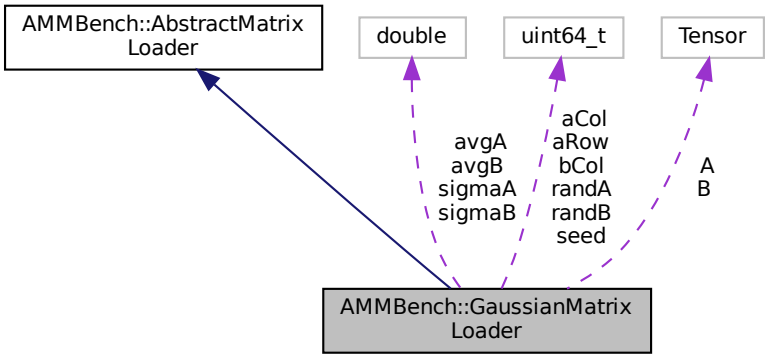
The Gaussian class of matrix loader.

`#include <MatrixLoader/GaussianMatrixLoader.h>`

Inheritance diagram for AMMBench::GaussianMatrixLoader:



Collaboration diagram for AMMBench::GaussianMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- uint64_t **randA**
- uint64_t **randB**
- double **sigmaA**
- double **avgA**
- double **sigmaB**
- double **avgB**

8.85.1 Detailed Description

The Gaussian class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514
- "randA" To generate matrix A under random distribution instead (will disable all guassian-related settings), U64, 0
- "randB" To generate matrix B under random distribution instead (will disable all guassian-related settings), U64, 0
- "sigmaA" The standard divation of A, Double, 1
- "avgA" The average value of A, Double, 0
- "sigmaB" The standard divation of B, Double, 1
- "avgB" The average value of A, Double, 0

: default name tags "random": GaussianMatrixLoader

8.85.2 Member Function Documentation

8.85.2.1 getA()

```
torch::Tensor AMMBench::GaussianMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.85.2.2 getB()

```
torch::Tensor AMMBench::GaussianMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.85.2.3 paraseConfig()

```
void AMMBench::GaussianMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.85.2.4 setConfig()

```
bool AMMBench::GaussianMatrixLoader::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

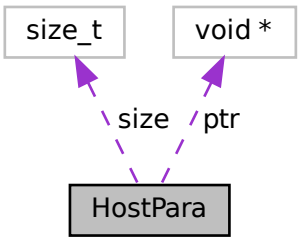
Reimplemented from AMMBench::AbstractMatrixLoader.

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

- include/MatrixLoader/GaussianMatrixLoader.h
- src/MatrixLoader/GaussianMatrixLoader.cpp

8.86 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:

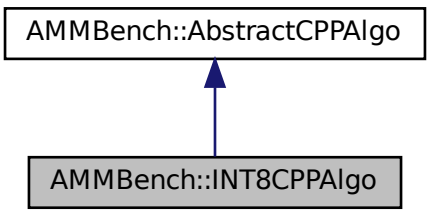
- include/CL/CLContainer.hpp

8.87 AMMBench::INT8CPPAlgo Class Reference

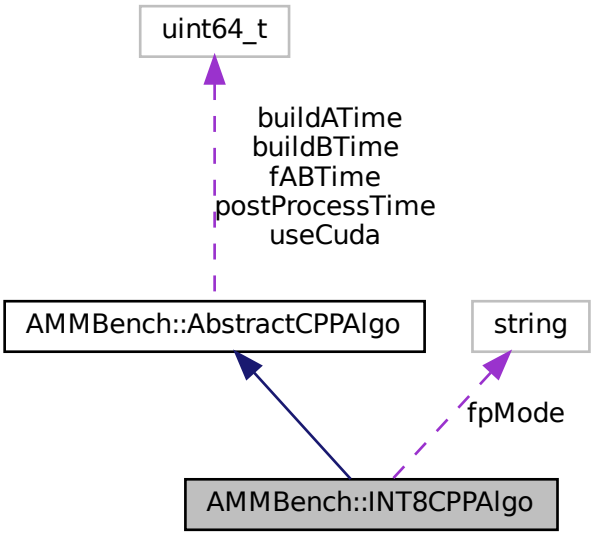
The INT8 MM class of c++ algos.

`#include <CPPAlgos/INT8CPPAlgo.h>`

Inheritance diagram for AMMBench::INT8CPPAlgo:



Collaboration diagram for AMMBench::INT8CPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes
- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm

Protected Member Functions

- torch::Tensor fp32amm (torch::Tensor A, torch::Tensor B)

  the inline amm under nested loop fp32
- torch::Tensor fp64amm (torch::Tensor A, torch::Tensor B)

  the inline amm under nested loop fp64
- torch::Tensor int8amm (torch::Tensor A, torch::Tensor B)

  the inline amm under nested loop int8
- torch::Tensor int4amm (torch::Tensor A, torch::Tensor B)

  the inline amm under nested loop int4
- torch::Tensor int16amm (torch::Tensor A, torch::Tensor B)

  the inline amm under nested loop int16

Protected Attributes

- std::string **fpMode** = "FP32"

8.87.1 Detailed Description

The INT8 MM class of c++ algos.

++

Warning

This function disables all additional optimization by libtorch, as it has different, and not fair SIMD/cache optimization over FP32/INT16/INT8 on cpu, which is hard to compare

Note

additionally parameters

- fpMode, String, default FP32, can also use INT8 or INT16

8.87.2 Member Function Documentation

8.87.2.1 amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

8.87.2.2 fp32amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::fp32amm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

the inline amm under nested loop fp32

Parameters

A	the A matrix
B	the B matrix

Returns

the output c matrix

build A into std vector

8.87.2.3 fp64amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::fp64amm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

the inline amm under nested loop fp64

Parameters

A	the A matrix
B	the B matrix

Returns

the output c matrix

build A into std vector

8.87.2.4 int16amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::int16amm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

the inline amm under nested loop int16

Parameters

A	the A matrix
B	the B matrix

Returns

the output c matrix

build A

build B

run fAB

32/16=2, so we simulate a 2-way SHARED-NOTHING speed up in one loop

post process

8.87.2.5 int4amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::int4amm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

the inline amm under nested loop int4

Parameters

A	the A matrix
B	the B matrix

Returns

the output c matrix

build A

build B

run fAB

32/4=8, so we simulate a 8-way SHARED-NOTHING speed up in one loop

post process

8.87.2.6 int8amm()

```
torch::Tensor AMMBench::INT8CPPAlgo::int8amm (
            torch::Tensor A,
            torch::Tensor B )  [protected]
```

the inline amm under nested loop int8

Parameters

A	the A matrix
B	the B matrix

Returns

the output c matrix

build A

build B

run fAB

32/8=4, so we simulate a 4-way SHARED-NOTHING speed up in one loop

post process

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

- include/CPPAlgos/INT8CPPAlgo.h
- src/CPPAlgos/INT8CPPAlgo.cpp

8.88 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

8.88.1 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

8.89 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

8.89.1 Detailed Description

The protector for concurrent log on a file.

Warning

This class is preserved for internal use only!

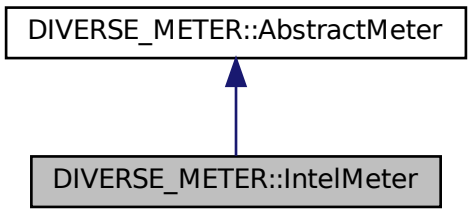
The documentation for this class was generated from the following file:

- include/Utils/IntelliLog.h

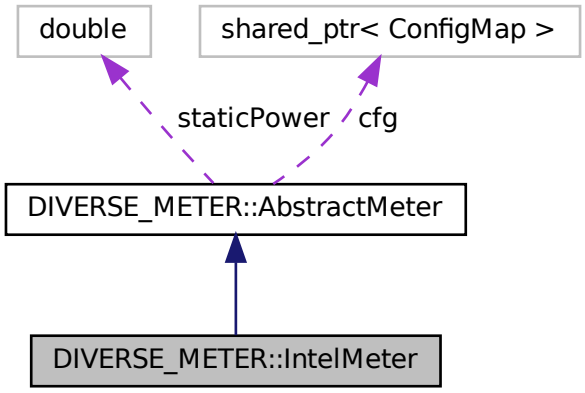
8.90 DIVERSE_METER::IntelMeter Class Reference

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

8.90.1 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"

8.90.2 Member Function Documentation

8.90.2.1 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:

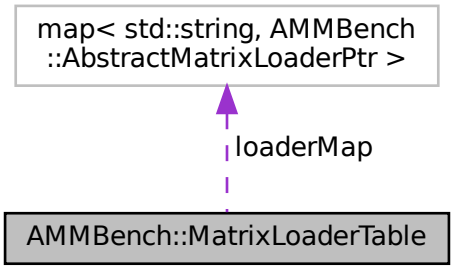
- include/Utils/Meters/IntelMeter/IntelMeter.hpp
- src/Utils/Meters/IntelMeter/IntelMeter.cpp

8.91 AMMBench::MatrixLoaderTable Class Reference

The table class to index all matrix loaders.

`#include <MatrixLoader/MatrixLoaderTable.h>`

Collaboration diagram for AMMBench::MatrixLoaderTable:



Public Types

- typedef std::shared_ptr< class AMMBench::MatrixLoaderTable > MatrixLoaderTablePtr

  The class to describe a shared pointer to MatrixLoaderTable.

Public Member Functions

- MatrixLoaderTable ()

  The constructing function.
- void registerNewDataLoader (AMMBench::AbstractMatrixLoaderPtr dnew, std::string tag)

  To register a new loader.
- AMMBench::AbstractMatrixLoaderPtr findMatrixLoader (std::string name)

  find a dataloader in the table according to its name

Protected Attributes

- std::map< std::string, AMMBench::AbstractMatrixLoaderPtr > **loaderMap**

8.91.1 Detailed Description

The table class to index all matrix loaders.

Note

Default behavior

- create
- (optional) call registerNewDataLoader for new loader
- find a loader by findMatrixLoader using its tag

default tags

- random RandomMatrixLoader
- sparse SparseMatrixLoader

8.91.2 Constructor & Destructor Documentation

8.91.2.1 MatrixLoaderTable()

```
AMMBench::MatrixLoaderTable::MatrixLoaderTable ( )
```

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

8.91.3 Member Function Documentation

8.91.3.1 findMatrixLoader()

```
AMMBench::AbstractMatrixLoaderPtr AMMBench::MatrixLoaderTable::findMatrixLoader (
            std::string name )  [inline]
```

find a dataloader in the table according to its name

Parameters

name	The nameTag of loader

Returns

The MatrixLoader, nullptr if not found

8.91.3.2 registerNewDataLoader()

```
void AMMBench::MatrixLoaderTable::registerNewDataLoader (
            AMMBench::AbstractMatrixLoaderPtr 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:

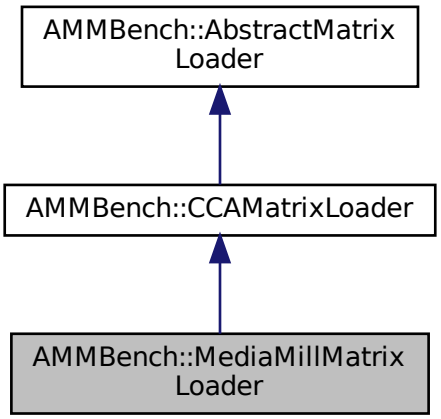
- include/MatrixLoader/MatrixLoaderTable.h
- src/MatrixLoader/MatrixLoaderTable.cpp

8.92 AMMBench::MediaMillMatrixLoader Class Reference

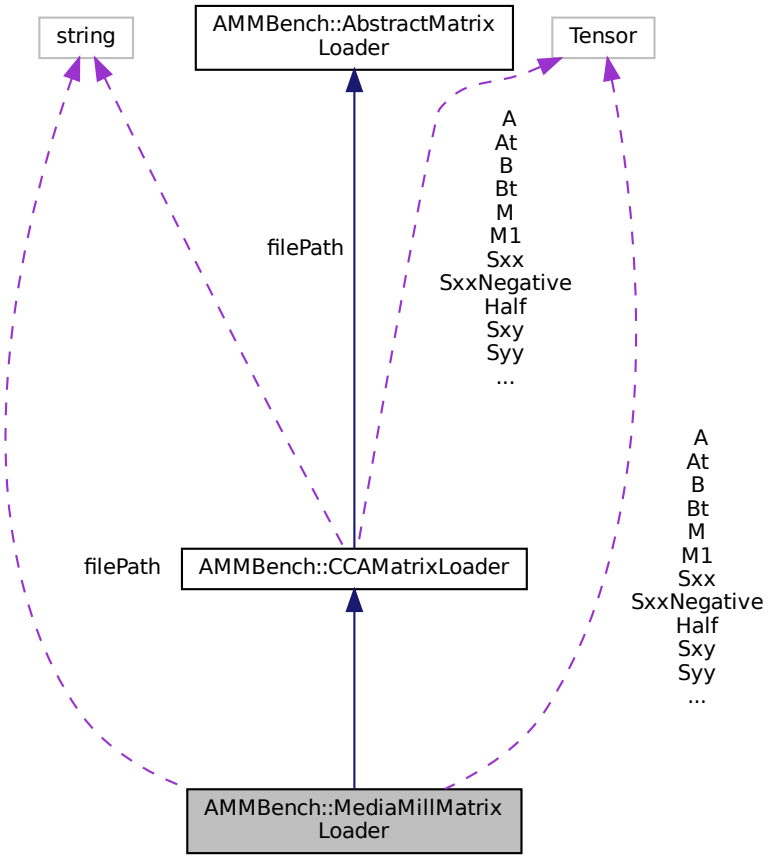
Load MediaMill 2005-2006 data (https://rdrr.io/github/fcharte/mldr.datasets/man/mediamill.↩ html)

`#include <MatrixLoader/MediaMillMatrixLoader.h>`

Inheritance diagram for AMMBench::MediaMillMatrixLoader:



Collaboration diagram for AMMBench::MediaMillMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual void calculate_correlation ()

  Calulate the correlation by mm, and generate tensor Sxx, Sxy, Syy, M, correlation.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix
- virtual torch::Tensor getAt ()

  get the transpose of A matrix
- virtual torch::Tensor getBt ()

  get the transpose of B matrix
- virtual torch::Tensor getSxx ()

  get the Sxx matrix
- virtual torch::Tensor getSyy ()

  get the Sxyymatrix
- virtual torch::Tensor getSxy ()

  get the Sxy matrix
- virtual torch::Tensor getSxxNegativeHalf ()

  get the SxxNegativeHalf matrix
- virtual torch::Tensor getSyyNegativeHalf ()

  get the SyyNegativeHalf matrix
- virtual torch::Tensor getM ()

  M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)
- virtual torch::Tensor getM1 ()

  M1 = mm(SxxNegativeHalf.t(), Sxy)
- virtual torch::Tensor getCorrelation ()

  get the correlation value

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- std::string **filePath** ="datasets/MediaMill/MediaMill.pth"
- torch::Tensor **A**
- torch::Tensor **B**
- torch::Tensor **At**
- torch::Tensor **Bt**
- torch::Tensor **Sxx**
- torch::Tensor **Syy**
- torch::Tensor **Sxy**
- torch::Tensor **SxxNegativeHalf**
- torch::Tensor **SyyNegativeHalf**
- torch::Tensor **M**
- torch::Tensor **M1**
- torch::Tensor **correlation**

8.92.1 Detailed Description

Load MediaMill 2005-2006 data (https://rdrr.io/github/fcharte/mldr.datasets/man/mediamill.↩
html)

2005-2006

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: does not need config

: default name tags "MediaMill": MediaMillMatrixLoader

8.92.2 Member Function Documentation

8.92.2.1 getA()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.2 getAt()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getAt ( )  [virtual]
```

get the transpose of A matrix

Returns

the A.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.3 getB()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.4 getBt()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getBt ( )  [virtual]
```

get the transpose of B matrix

Returns

the B.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.5 getCorrelation()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getCorrelation ( )  [virtual]
```

get the correlation value

Returns

the generated correlation by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.6 getM()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getM ( )  [virtual]
```

M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)

Returns

the generated M matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.7 getM1()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getM1 ( )  [virtual]
```

M1 = mm(SxxNegativeHalf.t(), Sxy)

Returns

the generated M1 matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.8 getSxx()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getSxx ( )  [virtual]
```

get the Sxx matrix

Returns

the generated Sxx matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.9 getSxxNegativeHalf()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getSxxNegativeHalf ( )  [virtual]
```

get the SxxNegativeHalf matrix

Returns

the generated SxxNegativeHalf matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.10 getSxy()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getSxy ( )  [virtual]
```

get the Sxy matrix

Returns

the generated Sxy matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.11 getSyy()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getSyy ( )  [virtual]
```

get the Sxyymatrix

Returns

the generated Syy matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.12 getSyyNegativeHalf()

```
torch::Tensor AMMBench::MediaMillMatrixLoader::getSyyNegativeHalf ( )  [virtual]
```

get the SyyNegativeHalf matrix

Returns

the generated SyyNegativeHalf matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.92.2.13 paraseConfig()

```
void AMMBench::MediaMillMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.92.2.14 setConfig()

```
bool AMMBench::MediaMillMatrixLoader::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 AMMBench::CCAMatrixLoader.

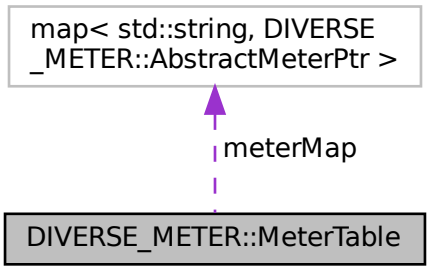
The documentation for this class was generated from the following files:

- include/MatrixLoader/MediaMillMatrixLoader.h
- src/MatrixLoader/MediaMillMatrixLoader.cpp

8.93 DIVERSE_METER::MeterTable Class Reference

The table class to index all meters.

Collaboration diagram for DIVERSE_METER::MeterTable:



Public Types

- typedef std::shared_ptr< class DIVERSE_METER::MeterTable > MeterTablePtr

  The class to describe a shared pointer to MeterTable.

Public Member Functions

- MeterTable ()

  The constructing function.
- void registerNewMeter (DIVERSE_METER::AbstractMeterPtr dnew, std::string tag)

  To register a new meter.
- DIVERSE_METER::AbstractMeterPtr findMeter (std::string name)

  find a meter in the table according to its name

Protected Attributes

- std::map< std::string, DIVERSE_METER::AbstractMeterPtr > **meterMap**

8.93.1 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

8.93.2 Constructor & Destructor Documentation

8.93.2.1 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

8.93.3 Member Function Documentation

8.93.3.1 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

8.93.3.2 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

8.94 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.

8.94.1 Detailed Description

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

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

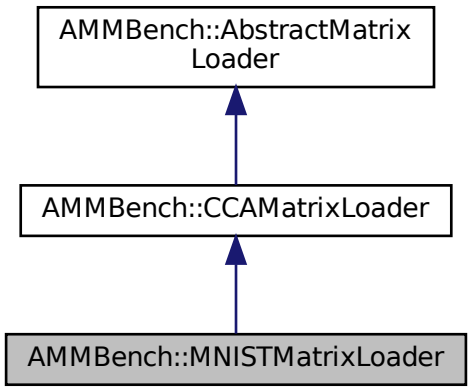
- include/Utils/MicroDataSet.hpp

8.95 AMMBench::MNISTMatrixLoader Class Reference

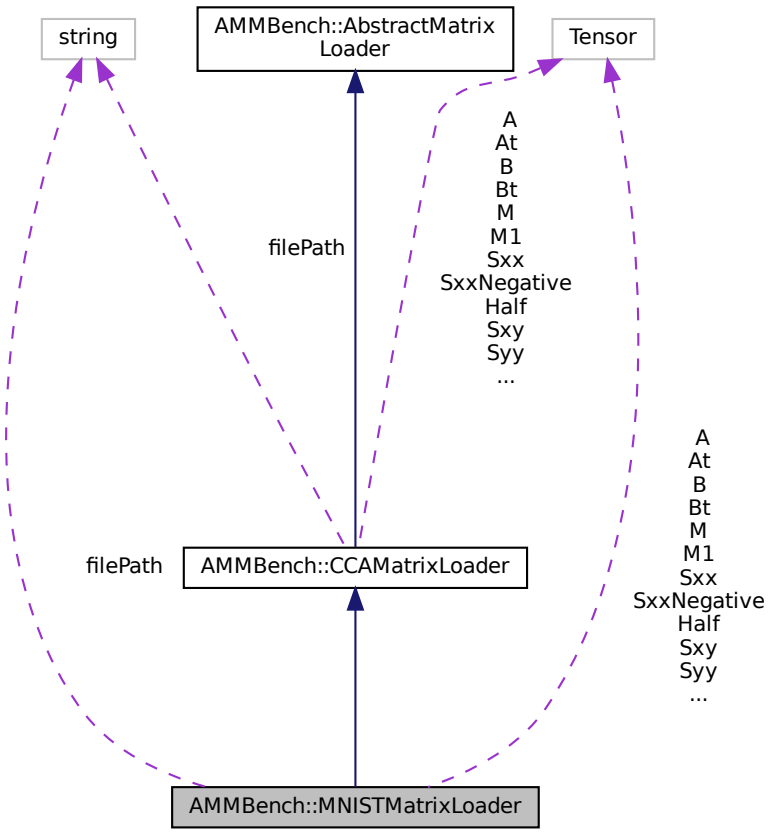
The MNIST class of matrix loader https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

`#include <MatrixLoader/MNISTMatrixLoader.h>`

Inheritance diagram for AMMBench::MNISTMatrixLoader:



Collaboration diagram for AMMBench::MNISTMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual void calculate_correlation ()

  Calulate the correlation by mm, and generate tensor Sxx, Sxy, Syy, M, correlation.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix
- virtual torch::Tensor getAt ()

  get the transpose of A matrix
- virtual torch::Tensor getBt ()

  get the transpose of B matrix
- virtual torch::Tensor getSxx ()

  get the Sxx matrix
- virtual torch::Tensor getSyy ()

  get the Sxyymatrix
- virtual torch::Tensor getSxy ()

get the Sxy matrix

- virtual torch::Tensor getSxxNegativeHalf ()

  get the SxxNegativeHalf matrix
- virtual torch::Tensor getSyyNegativeHalf ()

  get the SyyNegativeHalf matrix
- virtual torch::Tensor getM ()

  M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)
- virtual torch::Tensor getM1 ()

  M1 = mm(SxxNegativeHalf.t(), Sxy)
- virtual torch::Tensor getCorrelation ()

  get the correlation value

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- std::string **filePath** ="datasets/SIFT/MNIST/train-images.idx3-ubyte"
- torch::Tensor **A**
- torch::Tensor **B**
- torch::Tensor **At**
- torch::Tensor **Bt**
- torch::Tensor **Sxx**
- torch::Tensor **Syy**
- torch::Tensor **Sxy**
- torch::Tensor **SxxNegativeHalf**
- torch::Tensor **SyyNegativeHalf**
- torch::Tensor **M**
- torch::Tensor **M1**
- torch::Tensor **correlation**

8.95.1 Detailed Description

The MNIST class of matrix loader https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: does not need config

: default name tags "MNIST": MNISTMatrixLoader

8.95.2 Member Function Documentation

8.95.2.1 getA()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.2 getAt()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getAt ( )  [virtual]
```

get the transpose of A matrix

Returns

the A.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.3 getB()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.4 getBt()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getBt ( )  [virtual]
```

get the transpose of B matrix

Returns

the B.t().contiguous() matrix, which is not a view but has its own memory space

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.5 getCorrelation()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getCorrelation ( )  [virtual]
```

get the correlation value

Returns

the generated correlation by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.6 getM()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getM ( )  [virtual]
```

M = mm(mm(SxxNegativeHalf.t(), Sxy), SyyNegativeHalf)

Returns

the generated M matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.7 getM1()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getM1 ( )  [virtual]
```

M1 = mm(SxxNegativeHalf.t(), Sxy)

Returns

the generated M1 matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.8 getSxx()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getSxx ( )  [virtual]
```

get the Sxx matrix

Returns

the generated Sxx matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.9 getSxxNegativeHalf()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getSxxNegativeHalf ( )  [virtual]
```

get the SxxNegativeHalf matrix

Returns

the generated SxxNegativeHalf matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.10 getSxy()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getSxy ( )  [virtual]
```

get the Sxy matrix

Returns

the generated Sxy matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.11 getSyy()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getSyy ( )  [virtual]
```

get the Sxyymatrix

Returns

the generated Syy matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.12 getSyyNegativeHalf()

```
torch::Tensor AMMBench::MNISTMatrixLoader::getSyyNegativeHalf ( )  [virtual]
```

get the SyyNegativeHalf matrix

Returns

the generated SyyNegativeHalf matrix by calling calculate_correlation()

Reimplemented from AMMBench::CCAMatrixLoader.

8.95.2.13 paraseConfig()

```
void AMMBench::MNISTMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.95.2.14 setConfig()

```
bool AMMBench::MNISTMatrixLoader::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 AMMBench::CCAMatrixLoader.

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

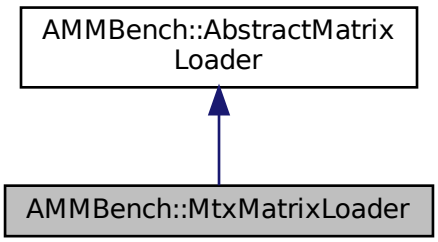
- include/MatrixLoader/MNISTMatrixLoader.h
- src/MatrixLoader/MNISTMatrixLoader.cpp

8.96 AMMBench::MtxMatrixLoader Class Reference

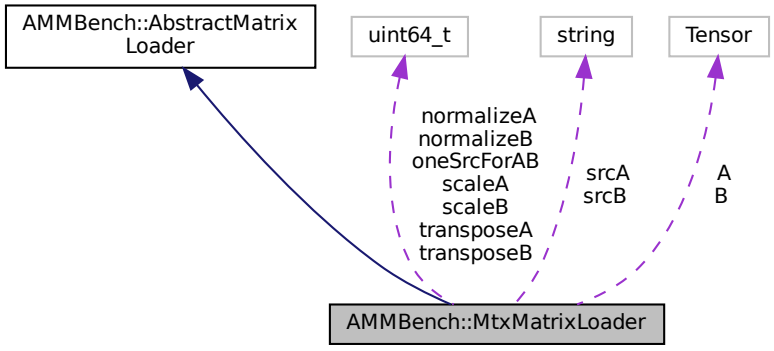
The matrix loader to load matrixes stored in matrix market mtx format.

`#include <MatrixLoader/MtxMatrixLoader.h>`

Inheritance diagram for AMMBench::MtxMatrixLoader:



Collaboration diagram for AMMBench::MtxMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- std::string **srcA**
- std::string **srcB**
- uint64_t **oneSrcForAB**
- uint64_t **transposeA**
- uint64_t **transposeB**
- uint64_t **normalizeA**
- uint64_t **normalizeB**
- uint64_t **scaleA**
- uint64_t **scaleB**

8.96.1 Detailed Description

The matrix loader to load matrixes stored in matrix market mtx format.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "srcA" The file source for A matrix, String, "datasets/ZENIOS/zenios.mtx"
- "oneSrcForAB", U64, whether A and B shares the same source file
- "srcB" The file source for B matrix, String, "datasets/ZENIOS/zenios.mtx"
- "transposeA" Whether or not transpose A matrix, U64, 0
- "transposeB" Whether or not transpose B matrix, U64, 1
- "normalizeA" Whether or not normalize A matrix (Normalization will force the minimum value to be -1) , U64, 0
- "normalizeB" Whether or not transpose B matrix, U64, 0
- "scaleA" Whether or not scale A matrix (scale will force the maximum value to be 1) , U64, 0 -

: do not normalize and scale at the same time

- "scaleB" Whether or not scale B matrix (scale will force the maximum value to be 1) , U64, 0

: default name tags "mtx": MtxMatrixLoader

8.96.2 Member Function Documentation

8.96.2.1 getA()

```
torch::Tensor AMMBench::MtxMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.96.2.2 getB()

```
torch::Tensor AMMBench::MtxMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.96.2.3 paraseConfig()

```
void AMMBench::MtxMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.96.2.4 setConfig()

```
bool AMMBench::MtxMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

- include/MatrixLoader/MtxMatrixLoader.h
- src/MatrixLoader/MtxMatrixLoader.cpp

8.97 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.

8.97.1 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.

8.97.2 Constructor & Destructor Documentation

8.97.2.1 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.

8.97.3 Member Function Documentation

8.97.3.1 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.

8.97.3.2 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.

8.97.3.3 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.

8.97.3.4 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:

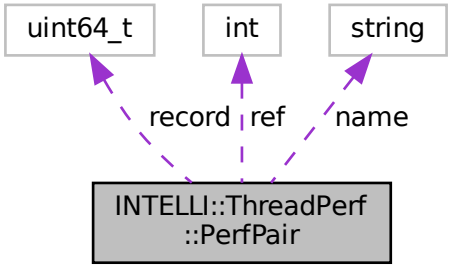
- include/Utils/BS_thread_pool.hpp

8.98 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**

8.98.1 Detailed Description

a record pair of perf events

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

- include/Utils/ThreadPerf.hpp

8.99 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:

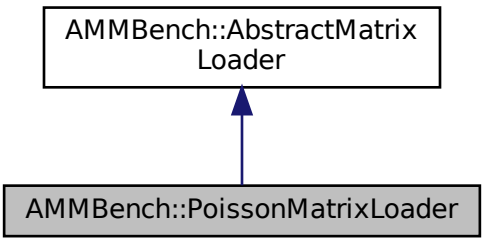
- include/Utils/ThreadPerf.hpp

8.100 AMMBench::PoissonMatrixLoader Class Reference

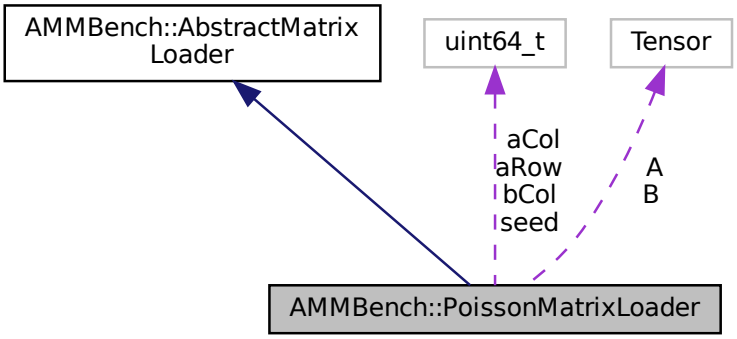
The Poisson class of matrix loader.

`#include <MatrixLoader/PoissonMatrixLoader.h>`

Inheritance diagram for AMMBench::PoissonMatrixLoader:



Collaboration diagram for AMMBench::PoissonMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**

8.100.1 Detailed Description

The Poisson class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514

: default name tags "random": PoissonMatrixLoader

8.100.2 Member Function Documentation

8.100.2.1 getA()

```
torch::Tensor AMMBench::PoissonMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.100.2.2 getB()

```
torch::Tensor AMMBench::PoissonMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.100.2.3 paraseConfig()

```
void AMMBench::PoissonMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.100.2.4 setConfig()

```
bool AMMBench::PoissonMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

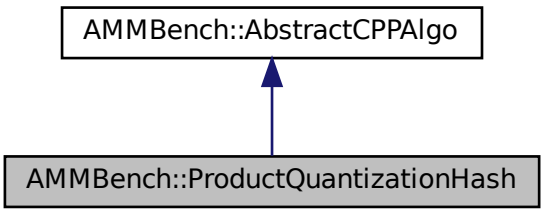
- include/MatrixLoader/PoissonMatrixLoader.h
- src/MatrixLoader/PoissonMatrixLoader.cpp

8.101 AMMBench::ProductQuantizationHash Class Reference

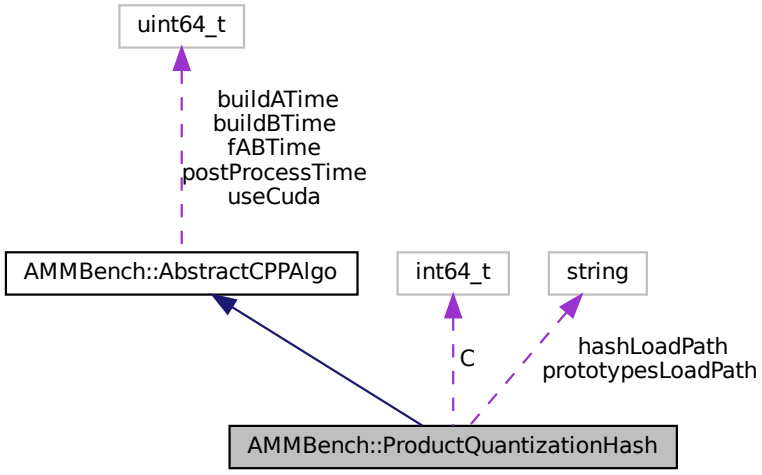
The Product Quantization AMM class of c++ algos, using hash function to find matching prototypes.

`#include <CPPAlgos/ProductQuantizationHash.h>`

Inheritance diagram for AMMBench::ProductQuantizationHash:



Collaboration diagram for AMMBench::ProductQuantizationHash:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm
- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes

Protected Attributes

- string **prototypesLoadPath**
- string **hashLoadPath**
- int64_t **C**

8.101.1 Detailed Description

The Product Quantization AMM class of c++ algos, using hash function to find matching prototypes.

++

8.101.2 Member Function Documentation

8.101.2.1 amm()

```
torch::Tensor AMMBench::ProductQuantizationHash::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

8.101.2.2 setConfig()

```
void AMMBench::ProductQuantizationHash::setConfig (
            INTELLI::ConfigMapPtr cfg )  [virtual]
```

set the alo-specfic config related to one algorithm

Parameters

prototypesLoadPath	where to load prototypes
hashLoadPath	where to load hash

Reimplemented from AMMBench::AbstractCPPAlgo.

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

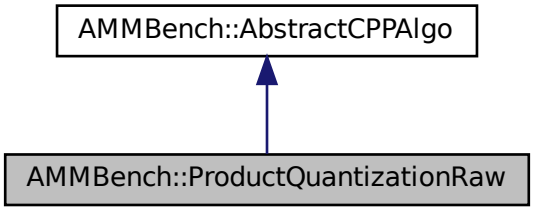
- include/CPPAlgos/ProductQuantizationHash.h
- src/CPPAlgos/ProductQuantizationHash.cpp

8.102 AMMBench::ProductQuantizationRaw Class Reference

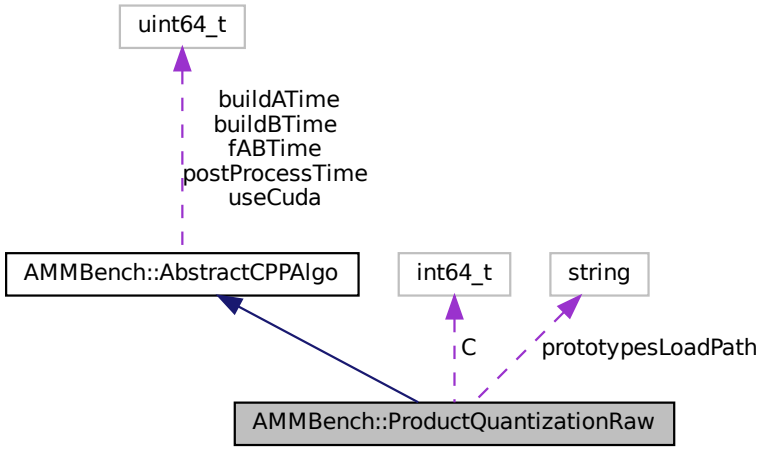
The Product Quantization AMM class of c++ algos, using Euclidean distance.

`#include <CPPAlgos/ProductQuantizationRaw.h>`

Inheritance diagram for AMMBench::ProductQuantizationRaw:



Collaboration diagram for AMMBench::ProductQuantizationRaw:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes
- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm

Protected Attributes

- string **prototypesLoadPath**
- int64_t **C**

8.102.1 Detailed Description

The Product Quantization AMM class of c++ algos, using Euclidean distance.

++

8.102.2 Member Function Documentation

8.102.2.1 amm()

```
torch::Tensor AMMBench::ProductQuantizationRaw::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

8.102.2.2 setConfig()

```
void AMMBench::ProductQuantizationRaw::setConfig (
            INTELLI::ConfigMapPtr cfg )  [virtual]
```

set the alo-specfic config related to one algorithm

Parameters

prototypesLoadPath	where to load prototypes

Reimplemented from AMMBench::AbstractCPPAlgo.

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

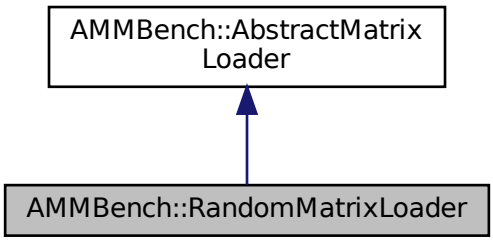
- include/CPPAlgos/ProductQuantizationRaw.h
- src/CPPAlgos/ProductQuantizationRaw.cpp

8.103 AMMBench::RandomMatrixLoader Class Reference

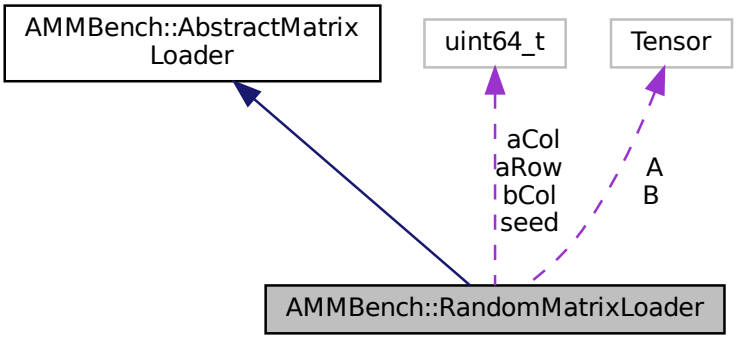
The Random class of matrix loader.

`#include <MatrixLoader/RandomMatrixLoader.h>`

Inheritance diagram for AMMBench::RandomMatrixLoader:



Collaboration diagram for AMMBench::RandomMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**

8.103.1 Detailed Description

The Random class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514

: default name tags "random": RandomMatrixLoader

8.103.2 Member Function Documentation

8.103.2.1 getA()

```
torch::Tensor AMMBench::RandomMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.103.2.2 getB()

```
torch::Tensor AMMBench::RandomMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.103.2.3 paraseConfig()

```
void AMMBench::RandomMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.103.2.4 setConfig()

```
bool AMMBench::RandomMatrixLoader::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

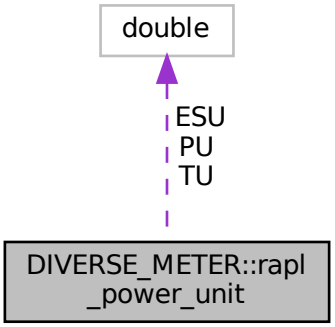
Reimplemented from AMMBench::AbstractMatrixLoader.

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

- include/MatrixLoader/RandomMatrixLoader.h
- src/MatrixLoader/RandomMatrixLoader.cpp

8.104 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:

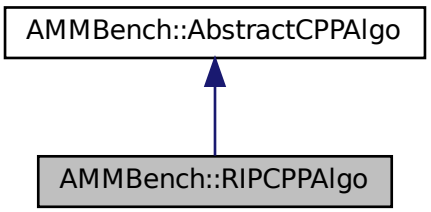
- include/Utils/Meters/IntelMeter/IntelMeter.hpp

8.105 AMMBench::RIPCPPAlgo Class Reference

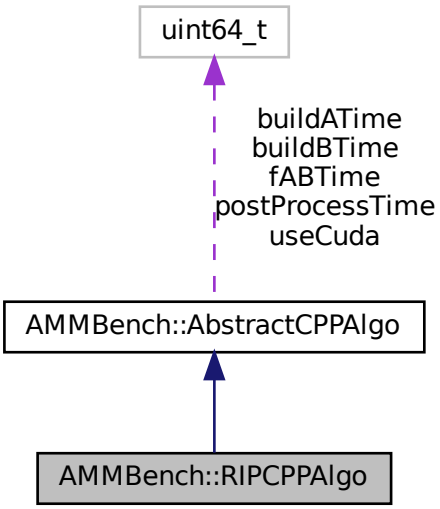
New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property.

`#include <CPPAlgos/RIPCPPAlgo.h>`

Inheritance diagram for AMMBench::RIPCPPAlgo:



Collaboration diagram for AMMBench::RIPCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.105.1 Detailed Description

New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property.

++

8.105.2 Member Function Documentation

8.105.2.1 amm()

```
torch::Tensor AMMBench::RIPCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

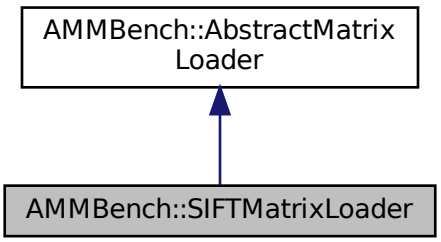
- include/CPPAlgos/RIPCPPAlgo.h
- src/CPPAlgos/RIPCPPAlgo.cpp

8.106 AMMBench::SIFTMatrixLoader Class Reference

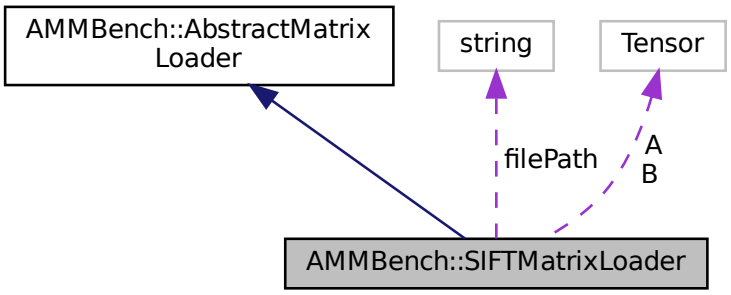
The SIFT class of matrix loader http://corpus-texmex.irisa.fr/.

`#include <MatrixLoader/SIFTMatrixLoader.h>`

Inheritance diagram for AMMBench::SIFTMatrixLoader:



Collaboration diagram for AMMBench::SIFTMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- std::string **filePath** ="datasets/SIFT/siftsmall_base.fvecs"

8.106.1 Detailed Description

The SIFT class of matrix loader http://corpus-texmex.irisa.fr/.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: does not need config

: default name tags "SIFT": SIFTMatrixLoader

8.106.2 Member Function Documentation

8.106.2.1 getA()

```
torch::Tensor AMMBench::SIFTMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.106.2.2 getB()

```
torch::Tensor AMMBench::SIFTMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.106.2.3 paraseConfig()

```
void AMMBench::SIFTMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.106.2.4 setConfig()

```
bool AMMBench::SIFTMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

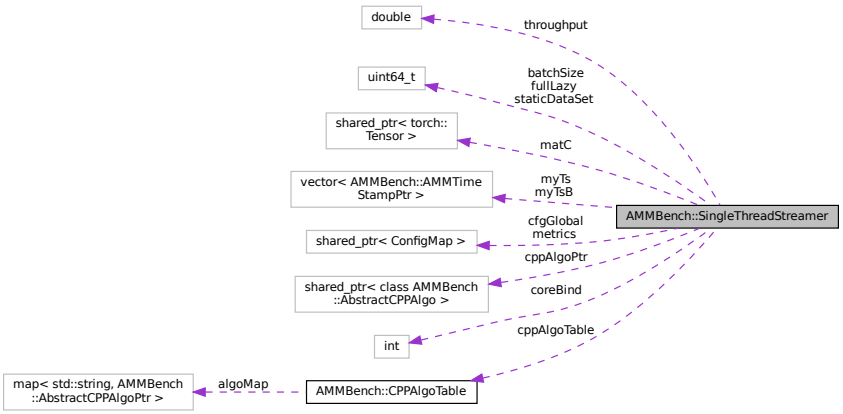
- include/MatrixLoader/SIFTMatrixLoader.h
- src/MatrixLoader/SIFTMatrixLoader.cpp

8.107 AMMBench::SingleThreadStreamer Class Reference

The class to run streaming amm under single thread, let each row of A coming in a streaming manner.

`#include <Streaming/SingleThreadStreamer.h>`

Collaboration diagram for AMMBench::SingleThreadStreamer:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this TimerStamper.
- virtual bool prepareRun (torch::Tensor A, torch::Tensor B)

  create the time stamps and other datastructures for streaming rn
- virtual torch::Tensor streamingAmm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize=1)

  To run a streaming Amm, assuming the rows of A coming in a streaming manner and B is fixed.
- virtual torch::Tensor streamingAmm2S (torch::Tensor A, torch::Tensor B, uint64_t sketchSize=1)

  To run a streaming Amm, assuming the rows of A coming in a streaming manner and the cols of B coming in a streaming manner.
- double getThroughput ()

  to get the throughput of last streaming process, the unit is rows/second
- double getLatencyPercentage (double fraction)

  to get the latency within some fraction, such as 0.95
- INTELLI::ConfigMapPtr getMetrics ()

  get metrics (including the pef result for all threads used in the runner, and elapsed time, throughput..)

Public Attributes

- std::vector< AMMBench::AMMTimeStampPtr > myTs

  the timestamps to trace the streaming process
- std::vector< AMMBench::AMMTimeStampPtr > myTsB

  the additional timestamps to trace the streaming process, if B is also stream

Protected Attributes

- INTELLI::ConfigMapPtr **cfgGlobal**
- AMMBench::CPPAlgoTable **cppAlgoTable**
- uint64_t **batchSize** = 1
- AMMBench::AbstractCPPAlgoPtr **cppAlgoPtr** = nullptr
- AMMBench::TensorPtr **matC** = nullptr
- double **throughput** = 0.0
- int **coreBind**
- INTELLI::ConfigMapPtr **metrics** = newConfigMap()
- uint64_t **fullLazy** = 0
- uint64_t **staticDataSet** =0

8.107.1 Detailed Description

The class to run streaming amm under single thread, let each row of A coming in a streaming manner.

Note

Default behavior

- create
- call setConfig, this will also determine how to generate time stamp and config will be passed to TimeStamper
- run streaming amm:
  - call streamingAmm, if only A matrix will be streamed
  - call streamingAmm2S, if both A and B will be streamed
- call getThroughput, and getLatencyPercentage to get the streaming performance

configs fullLazy U64, 0 whether or not make everything conducted under lazy mode, will force batchsize to the whole rows of A batchSize, U64,1 staticDataSet, U64, 0 , whether or not treat a dataset as static

8.107.2 Member Function Documentation

8.107.2.1 getLatencyPercentage()

```
double AMMBench::SingleThreadStreamer::getLatencyPercentage (
            double fraction )
```

to get the latency within some fraction, such as 0.95

Parameters

fraction	the 0∼1 fraction

Returns

the latency in us

8.107.2.2 getMetrics()

```
INTELLI::ConfigMapPtr AMMBench::SingleThreadStreamer::getMetrics ( )  [inline]
```

get metrics (including the pef result for all threads used in the runner, and elapsed time, throughput..)

Returns

metrics ConfigMapPtr

8.107.2.3 getThroughput()

```
double AMMBench::SingleThreadStreamer::getThroughput ( )  [inline]
```

to get the throughput of last streaming process, the unit is rows/second

Returns

the throughput

8.107.2.4 prepareRun()

```
bool AMMBench::SingleThreadStreamer::prepareRun (
            torch::Tensor A,
            torch::Tensor B )  [virtual]
```

create the time stamps and other datastructures for streaming rn

Parameters

A	

Returns

8.107.2.5 setConfig()

```
bool AMMBench::SingleThreadStreamer::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

1.set the algo

1. set the batch size
2. load other configs

8.107.2.6 streamingAmm()

```
torch::Tensor AMMBench::SingleThreadStreamer::streamingAmm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize = 1 )  [virtual]
```

To run a streaming Amm, assuming the rows of A coming in a streaming manner and B is fixed.

Parameters

A	The A matrix
B	The B matrix

Returns

bool whether the config is successfully set

now, the whole batch has arrived, compute

the new arrived A will be no longer probed, so we can assign the processed time now

update the indexes

8.107.2.7 streamingAmm2S()

```
torch::Tensor AMMBench::SingleThreadStreamer::streamingAmm2S (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize = 1 )  [virtual]
```

To run a streaming Amm, assuming the rows of A coming in a streaming manner and the cols of B coming in a streaming manner.

Parameters

A	The A matrix
B	The B matrix

Returns

bool whether the config is successfully set

now, the whole batch has arrived, compute

do the incomingA∗newArrivedB part

do the oldArrivedA∗incomingB part

update the indexes

The latency calculation is different from one stream case here, as older A will still be probed by newer B

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

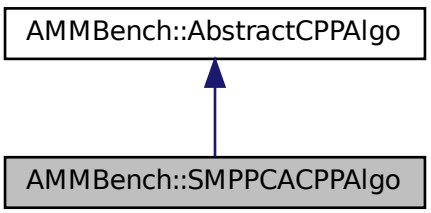
- include/Streaming/SingleThreadStreamer.h
- src/Streaming/SingleThreadStreamer.cpp

8.108 AMMBench::SMPPCACPPAlgo Class Reference

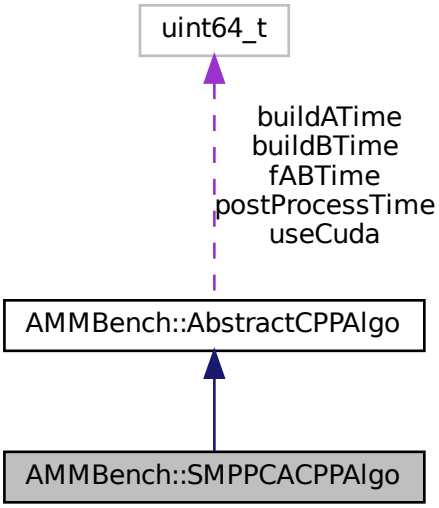
sketch scaled JL class of c++ algos

`#include <CPPAlgos/SMPPCACPPAlgo.h>`

Inheritance diagram for AMMBench::SMPPCACPPAlgo:



Collaboration diagram for AMMBench::SMPPCACPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.108.1 Detailed Description

sketch scaled JL class of c++ algos

++

8.108.2 Member Function Documentation

8.108.2.1 amm()

```
torch::Tensor AMMBench::SMPPCACPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketch

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

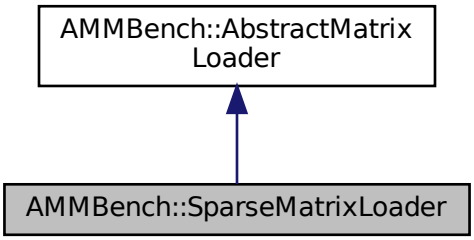
- include/CPPAlgos/SMPPCACPPAlgo.h
- src/CPPAlgos/SMPPCACPPAlgo.cpp

8.109 AMMBench::SparseMatrixLoader Class Reference

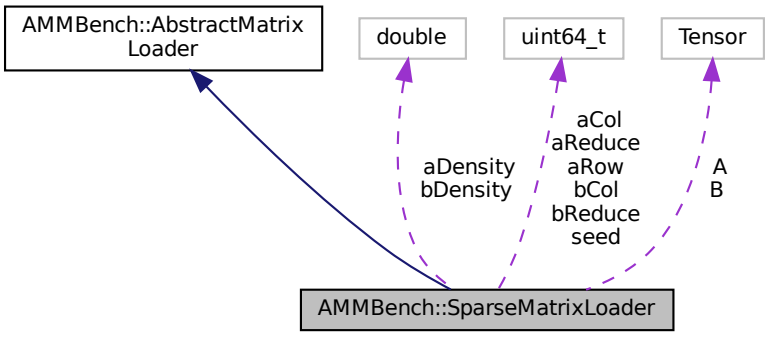
The matrix loader to generate adjustable sparse matrix with adjust rank reduction.

`#include <MatrixLoader/SparseMatrixLoader.h>`

Inheritance diagram for AMMBench::SparseMatrixLoader:



Collaboration diagram for AMMBench::SparseMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- torch::Tensor genSparseMatrix (uint64_t m, uint64_t n, double density, uint64_t reduceRows)

  Inline logic of generate the sparse matrix.
- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- uint64_t **aReduce**
- uint64_t **bReduce**
- double **aDensity**
- double **bDensity**

8.109.1 Detailed Description

The matrix loader to generate adjustable sparse matrix with adjust rank reduction.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline Sparse generator,U64,114514
- "aDensity" The density factor of matrix A, Double, 1.0
- "bDensity" The density factor of matrix B, Double, 1.0
- "aReduce" Reduce some rows of A to be linearly dependent, U64, 0
- "bReduce" Reduce some rows of A to be linearly dependent, U64, 0

: default name tags "sparse": SparseMatrixLoader

8.109.2 Member Function Documentation

8.109.2.1 genSparseMatrix()

```
torch::Tensor AMMBench::SparseMatrixLoader::genSparseMatrix (
            uint64_t m,
            uint64_t n,
            double density,
            uint64_t reduceRows )  [protected]
```

Inline logic of generate the sparse matrix.

Parameters

m	the rows
n	the cols
density	the density in 0∼1
reduceRows	the number of rows to be reduced

1. gen random mat
2. make it sparse according to density
3. reduce rows

8.109.2.2 getA()

```
torch::Tensor AMMBench::SparseMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.109.2.3 getB()

```
torch::Tensor AMMBench::SparseMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.109.2.4 paraseConfig()

```
void AMMBench::SparseMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.109.2.5 setConfig()

```
bool AMMBench::SparseMatrixLoader::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

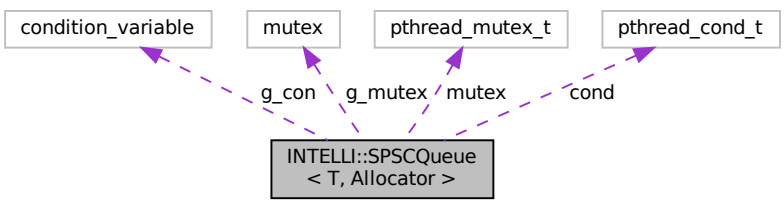
Reimplemented from AMMBench::AbstractMatrixLoader.

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

- include/MatrixLoader/SparseMatrixLoader.h
- src/MatrixLoader/SparseMatrixLoader.cpp

8.110 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

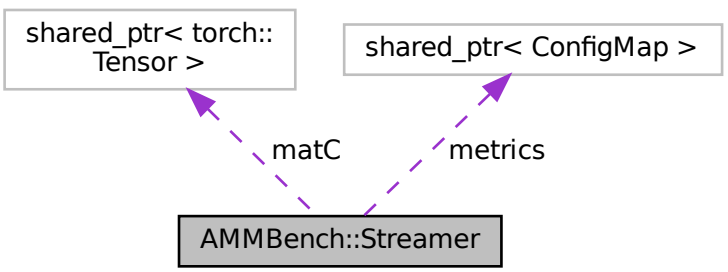
- pthread_cond_t **cond**
- pthread_mutex_t **mutex**
- std::mutex **g_mutex**
- condition_variable **g_con**

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

- include/Utils/SPSCQueue.hpp

8.111 AMMBench::Streamer Class Reference

Collaboration diagram for AMMBench::Streamer:



Public Member Functions

- torch::Tensor **run** (INTELLI::ConfigMapPtr cfg, torch::Tensor A, torch::Tensor B, uint64_t sketchSize=1, string metricPrefix="")
- INTELLI::ConfigMapPtr getMetrics ()

Protected Attributes

- AMMBench::TensorPtr **matC** = nullptr
- INTELLI::ConfigMapPtr **metrics**

8.111.1 Member Function Documentation

8.111.1.1 getMetrics()

```
INTELLI::ConfigMapPtr AMMBench::Streamer::getMetrics ( )  [inline]
```

Returns

all the running metrics as a ConfigMap

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

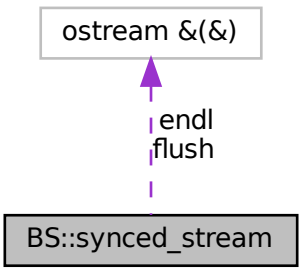
- include/Streaming/Streamer.h
- src/Streaming/Streamer.cpp

8.112 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 '\n' 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.

8.112.1 Detailed Description

A helper class to synchronize printing to an output stream by different threads.

8.112.2 Constructor & Destructor Documentation

8.112.2.1 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.

8.112.3 Member Function Documentation

8.112.3.1 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.

8.112.3.2 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.

8.112.4 Member Data Documentation

8.112.4.1 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.

8.112.4.2 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

8.113 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.

8.113.1 Detailed Description

A fast, lightweight, and easy-to-use C++17 thread pool class.

8.113.2 Constructor & Destructor Documentation

8.113.2.1 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.

8.113.3 Member Function Documentation

8.113.3.1 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.

8.113.3.2 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.

8.113.3.3 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.

8.113.3.4 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.

8.113.3.5 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.

8.113.3.6 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.

8.113.3.7 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.

8.113.3.8 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.

8.113.3.9 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.

8.113.3.10 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.

8.113.3.11 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.

8.113.3.12 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:

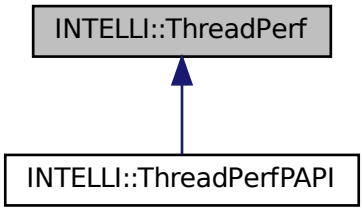
- include/Utils/BS_thread_pool.hpp

8.114 INTELLI::ThreadPerf Class Reference

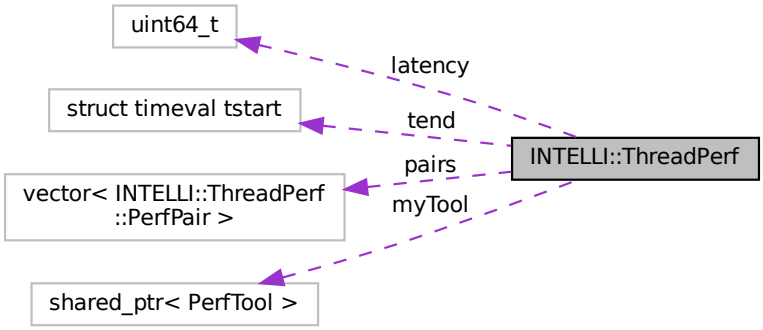
The top entity to provide perf traces, please use this class only UNLESS you know what you are doing.

```
#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**

8.114.1 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

8.114.2 Constructor & Destructor Documentation

8.114.2.1 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

8.114.3 Member Function Documentation

8.114.3.1 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.

8.114.3.2 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.

8.114.3.3 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.

8.114.3.4 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.

8.114.3.5 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:

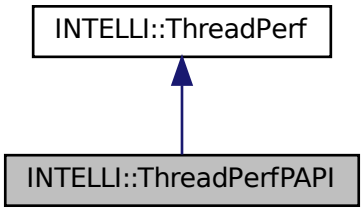
- include/Utils/ThreadPerf.hpp

8.115 INTELLI::ThreadPerfPAPI Class Reference

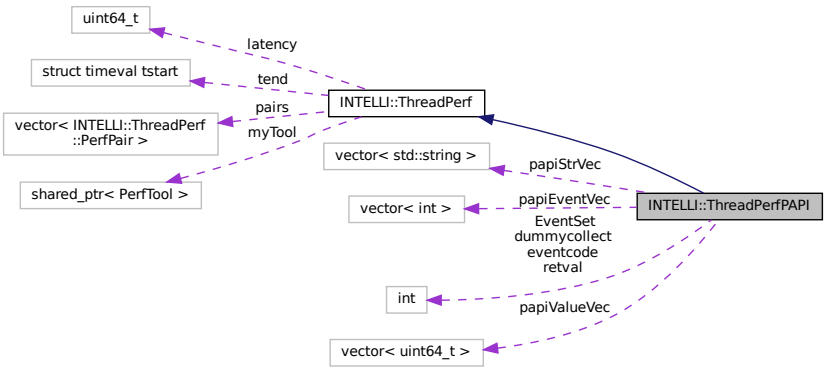
The top entity to provide perf traces by using PAPI lib.

```
#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

8.115.1 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

8.115.2 Constructor & Destructor Documentation

8.115.2.1 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

8.115.3 Member Function Documentation

8.115.3.1 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

8.115.3.2 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

8.115.3.3 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.

8.115.3.4 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.

8.115.3.5 initEventsByCfg()

```
void INTELLI::ThreadPerfPAPI::initEventsByCfg (
            ConfigMapPtr cfg )  [inline], [virtual]
```

init the perf events according to configmap

Parameters

cfg	tyhe configmap

Reimplemented from INTELLI::ThreadPerf.

8.115.3.6 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.

8.115.3.7 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

8.116 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().

8.116.1 Detailed Description

A helper class to measure execution time for benchmarking purposes.

8.116.2 Member Function Documentation

8.116.2.1 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:

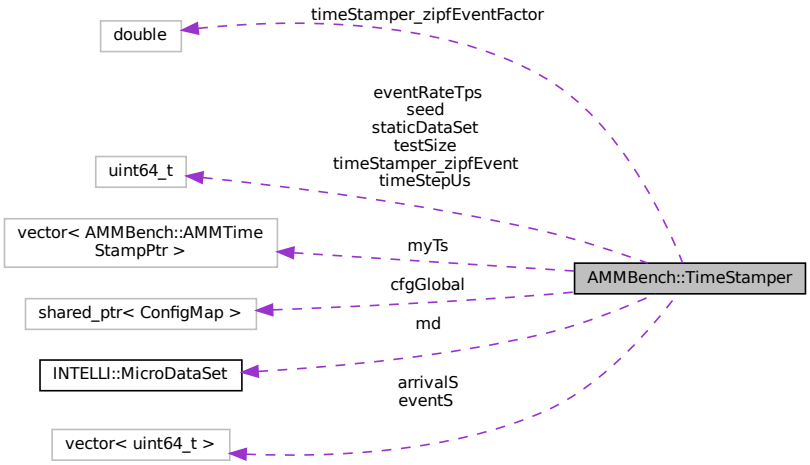
- include/Utils/BS_thread_pool.hpp

8.117 AMMBench::TimeStamper Class Reference

The basic class to generate time stamps.

`#include <Streaming/TimeStamper.h>`

Collaboration diagram for AMMBench::TimeStamper:



Public Member Functions

- void setSeed (uint64_t _seed)

  to set the seed of this timestamer
- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this TimerStamper.
- virtual std::vector< AMMBench::AMMTimeStampPtr > getTimeStamps ()

  get the vector of R tuple

Public Attributes

- std::vector< AMMBench::AMMTimeStampPtr > **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< AMMBench::AMMTimeStampPtr > **constructTimeStamps** (std::vector< uint64_t > eventS, std::vector< uint64_t > arrivalS)

Protected Attributes

- INTELLI::ConfigMapPtr **cfgGlobal**
- INTELLI::MicroDataSet **md**
- uint64_t **timeStamper_zipfEvent** = 0
- double **timeStamper_zipfEventFactor** = 0
- uint64_t **testSize**
- std::vector< uint64_t > **eventS**
- std::vector< uint64_t > **arrivalS**
- uint64_t **eventRateTps** = 0
- uint64_t **timeStepUs** = 40
- uint64_t **seed** = 114514
- uint64_t **staticDataSet** =0

8.117.1 Detailed Description

The basic class to generate time stamps.

Note

require configs:

- eventRateTps U64 The real-world rate of spawn event, in Tuples/s
- streamingTupleCnt U64 The number of "streaming tuples", can be set to the #rows or #cols of a matrix
- timeStamper_zipfEvent, U64, 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, U64, 0 , whether or not treat a dataset as static

Default behavior

- create
- call setSetSeed if you want different seed, default seed is 114514
- call setConfig to generate the timestamp under instructions
- call getTimeStamps to get the timestamp

8.117.2 Member Function Documentation

8.117.2.1 generateArrival()

```
void AMMBench::TimeStamper::generateArrival ( )  [protected]
```

generate the vector of arrival

Note

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

8.117.2.2 getTimeStamps()

```
virtual std::vector<AMMBench::AMMTimeStampPtr> AMMBench::TimeStamper::getTimeStamps ( )  [inline],
[virtual]
```

get the vector of R tuple

Returns

the vector

8.117.2.3 setConfig()

```
bool AMMBench::TimeStamper::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

8.117.2.4 setSeed()

```
void AMMBench::TimeStamper::setSeed (
            uint64_t _seed )  [inline]
```

to set the seed of this timestamer

Parameters

_seed	

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

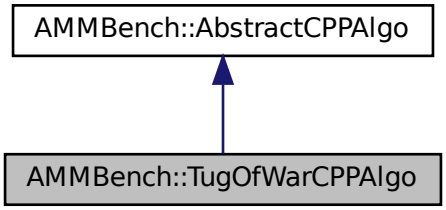
- include/Streaming/TimeStamper.h
- src/Streaming/TimeStamper.cpp

8.118 AMMBench::TugOfWarCPPAlgo Class Reference

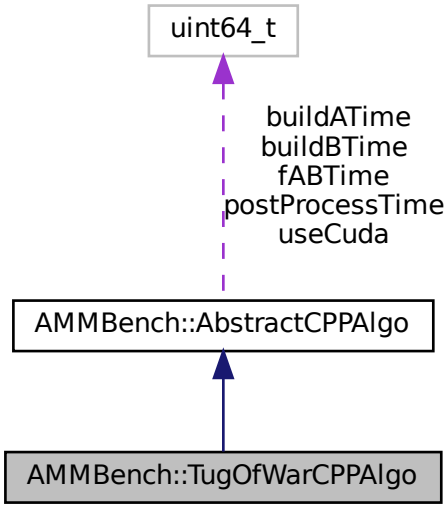
The tug of war class of c++ algoS.

`#include <CPPAlgos/TugOfWarCPPAlgo.h>`

Inheritance diagram for AMMBench::TugOfWarCPPAlgo:



Collaboration diagram for AMMBench::TugOfWarCPPAlgo:



Public Member Functions

- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the algo-specfic config related to one algorithm
- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.118.1 Detailed Description

The tug of war class of c++ algoS.

++

Note

parameters

- algoDelta Double, the delta parameter in this algo, default 0.02

8.118.2 Member Function Documentation

8.118.2.1 amm()

```
torch::Tensor AMMBench::TugOfWarCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

- include/CPPAlgos/TugOfWarCPPAlgo.h
- src/CPPAlgos/TugOfWarCPPAlgo.cpp

8.119 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 **relativeFrobeniusNorm** (torch::Tensor A, torch::Tensor B)
- static double **errorBoundRatio** (torch::Tensor A, torch::Tensor B)

8.119.1 Member Function Documentation

8.119.1.1 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

Todo unsure about hyper-thread

fixed some core bind bugs

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

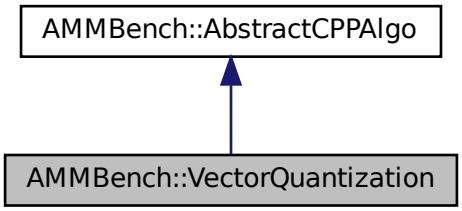
- include/Utils/UtilityFunctions.h
- src/Utils/UtilityFunctions.cpp

8.120 AMMBench::VectorQuantization Class Reference

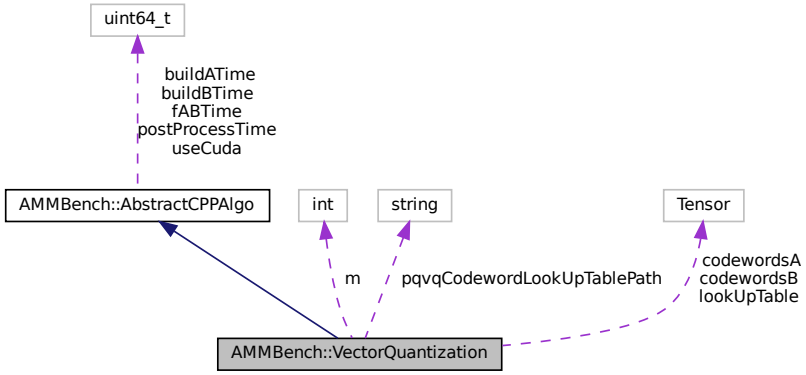
The Vector Quantization AMM class of c++ algos.

```
#include <CPPAlgos/VectorQuantization.h>
```

Inheritance diagram for AMMBench::VectorQuantization:



Collaboration diagram for AMMBench::VectorQuantization:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)

  the virtual function provided for outside callers, rewrite in children classes
- virtual void setConfig (INTELLI::ConfigMapPtr cfg)

  set the alo-specfic config related to one algorithm

Protected Attributes

- string **pqvqCodewordLookUpTablePath**
- int **m**
- torch::Tensor **codewordsA**
- torch::Tensor **codewordsB**
- torch::Tensor **lookUpTable**

8.120.1 Detailed Description

The Vector Quantization AMM class of c++ algos.

++

8.120.2 Member Function Documentation

8.120.2.1 amm()

```
torch::Tensor AMMBench::VectorQuantization::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

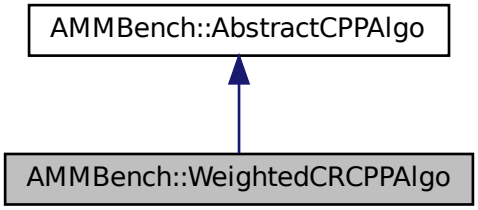
Reimplemented from AMMBench::AbstractCPPAlgo.

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

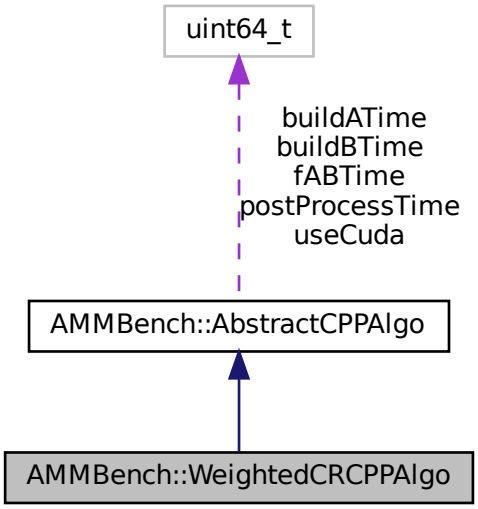
- include/CPPAlgos/VectorQuantization.h
- src/CPPAlgos/VectorQuantization.cpp

8.121 AMMBench::WeightedCRCPPAlgo Class Reference

Inheritance diagram for AMMBench::WeightedCRCPPAlgo:



Collaboration diagram for AMMBench::WeightedCRCPPAlgo:



Public Member Functions

- virtual torch::Tensor amm (torch::Tensor A, torch::Tensor B, uint64_t sketchSize)
  the virtual function provided for outside callers, rewrite in children classes

Additional Inherited Members

8.121.1 Member Function Documentation

8.121.1.1 amm()

```
torch::Tensor AMMBench::WeightedCRCPPAlgo::amm (
            torch::Tensor A,
            torch::Tensor B,
            uint64_t sketchSize )  [virtual]
```

the virtual function provided for outside callers, rewrite in children classes

Parameters

A	the A matrix
B	the B matrix
sketchSize	the size of sketc or sampling

Returns

the output c matrix

Reimplemented from AMMBench::AbstractCPPAlgo.

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

- include/CPPAlgos/WeightedCRCPPAlgo.h
- src/CPPAlgos/WeightedCRCPPAlgo.cpp

8.122 WeightedCRCPPlgo Class Reference

The weighted cloumn row sampling class of c++ algos.

`#include <CPPAlgos/WeightedCRCPPAlgo.h>`

8.122.1 Detailed Description

The weighted cloumn row sampling class of c++ algos.

++

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

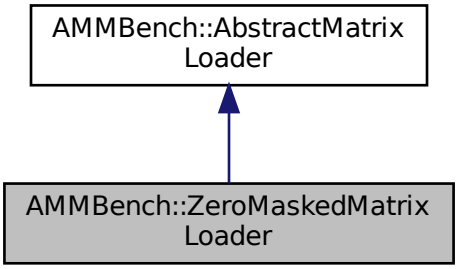
- include/CPPAlgos/WeightedCRCPPAlgo.h

8.123 AMMBench::ZeroMaskedMatrixLoader Class Reference

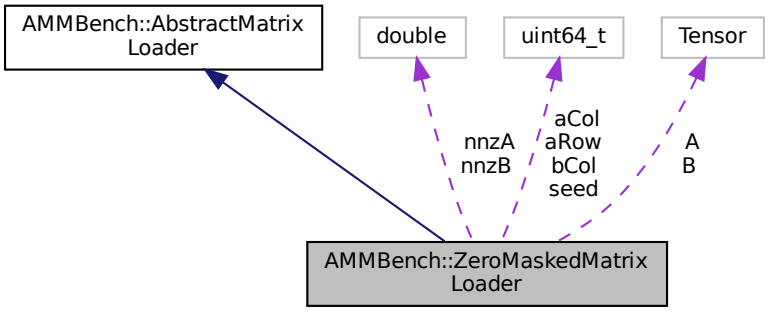
The zero masked class of matrix loader, given generate a n∗m matrix, where only the left-top n1∗m2 contents are not zero.

`#include <MatrixLoader/ZeroMaskedMatrixLoader.h>`

Inheritance diagram for AMMBench::ZeroMaskedMatrixLoader:



Collaboration diagram for AMMBench::ZeroMaskedMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- double **nnzA**
- double **nnzB**

8.123.1 Detailed Description

The zero masked class of matrix loader, given generate a n∗m matrix, where only the left-top n1∗m2 contents are not zero.

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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514
- "nnzA" the ratio of nnz values in matrix A, Double, 1.0
- "nnzB" the ratio of nnz values in matrix B, Double, 1.0

: default name tags "zeroMasked": ZeroMaskedMatrixLoader

8.123.2 Member Function Documentation

8.123.2.1 getA()

```
torch::Tensor AMMBench::ZeroMaskedMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.123.2.2 getB()

```
torch::Tensor AMMBench::ZeroMaskedMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.123.2.3 paraseConfig()

```
void AMMBench::ZeroMaskedMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.123.2.4 setConfig()

```
bool AMMBench::ZeroMaskedMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

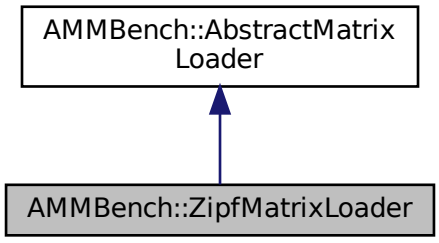
- include/MatrixLoader/ZeroMaskedMatrixLoader.h
- src/MatrixLoader/ZeroMaskedMatrixLoader.cpp

8.124 AMMBench::ZipfMatrixLoader Class Reference

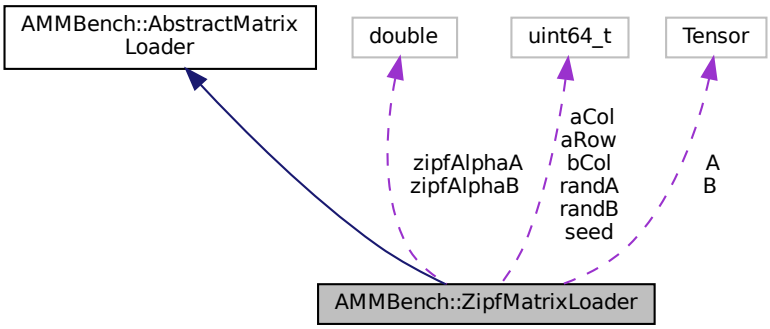
The Zipf class of matrix loader.

`#include <MatrixLoader/ZipfMatrixLoader.h>`

Inheritance diagram for AMMBench::ZipfMatrixLoader:



Collaboration diagram for AMMBench::ZipfMatrixLoader:



Public Member Functions

- virtual bool setConfig (INTELLI::ConfigMapPtr cfg)

  Set the GLOBAL config map related to this loader.
- virtual torch::Tensor getA ()

  get the A matrix
- virtual torch::Tensor getB ()

  get the B matrix

Protected Member Functions

- void paraseConfig (INTELLI::ConfigMapPtr cfg)

  Inline logic of reading a config file.
- void generateAB ()

  inline logic of generating A and B
- torch::Tensor **generateZipfDistribution** (int64_t rows, int64_t cols, double alpha)

Protected Attributes

- torch::Tensor **A**
- torch::Tensor **B**
- uint64_t **aRow**
- uint64_t **aCol**
- uint64_t **bCol**
- uint64_t **seed**
- uint64_t **randA**
- uint64_t **randB**
- double **zipfAlphaA**
- double **zipfAlphaB**

8.124.1 Detailed Description

The Zipf class of matrix 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 getA and getB (assuming we are benchmarking torch.mm(A,B))

: require config parameters and default values

- "aRow" The rows in matrix A, U64, 100
- "aCol" The cols in matrix B, U64, 1000
- "bCol" The rows in matrix B, U64, 500
- "seed" The seed of inline random generator,U64,114514
- "zipfAlphaA" The zipf factor for A, Double, 0-highly skewed value. 1- uniform dist.
- "zipfAlphaB" The zipf factor for B, Double, 0-highly skewed value. 1- uniform dist.
- "randA" whether let A a random matrix. U64 0
- "randB" whether let B a random matrix. U64 0

: default name tags "random": ZipfMatrixLoader

8.124.2 Member Function Documentation

8.124.2.1 getA()

```
torch::Tensor AMMBench::ZipfMatrixLoader::getA ( )  [virtual]
```

get the A matrix

Returns

the generated A matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.124.2.2 getB()

```
torch::Tensor AMMBench::ZipfMatrixLoader::getB ( )  [virtual]
```

get the B matrix

Returns

the generated B matrix

Reimplemented from AMMBench::AbstractMatrixLoader.

8.124.2.3 paraseConfig()

```
void AMMBench::ZipfMatrixLoader::paraseConfig (
            INTELLI::ConfigMapPtr cfg )  [protected]
```

Inline logic of reading a config file.

Parameters

cfg	the config

8.124.2.4 setConfig()

```
bool AMMBench::ZipfMatrixLoader::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 AMMBench::AbstractMatrixLoader.

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

- include/MatrixLoader/ZipfMatrixLoader.h
- src/MatrixLoader/ZipfMatrixLoader.cpp

Chapter 9

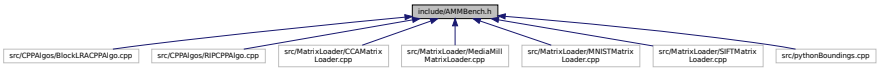
File Documentation

9.1 include/AMMBench.h File Reference

```
#include <torch/torch.h>
#include <iostream>
#include <torch/script.h>
#include <string>
#include <memory>
#include <MatrixLoader/AbstractMatrixLoader.h>
#include <MatrixLoader/RandomMatrixLoader.h>
#include <MatrixLoader/SparseMatrixLoader.h>
#include <MatrixLoader/GaussianMatrixLoader.h>
#include <MatrixLoader/ExponentialMatrixLoader.h>
#include <MatrixLoader/BinomialMatrixLoader.h>
#include <MatrixLoader/PoissonMatrixLoader.h>
#include <MatrixLoader/BetaMatrixLoader.h>
#include <MatrixLoader/SIFTMatrixLoader.h>
#include <MatrixLoader/MNISTMatrixLoader.h>
#include <MatrixLoader/MediaMillMatrixLoader.h>
#include <MatrixLoader/CCAMatrixLoader.h>
#include <MatrixLoader/MatrixLoaderTable.h>
#include <Parallelization/BlockPartitionRunner.h>
#include <Streaming/TimeStamper.h>
#include <Streaming/Streamer.h>
#include <Streaming/SingleThreadStreamer.h>
#include <Streaming/BlockPartitionStreamer.h>
#include <CPPAlgos/AbstractCPPAlgo.h>
#include <CPPAlgos/CPPAlgoTable.h>
#include <CPPAlgos/CRSCPPAlgo.h>
#include <CPPAlgos/CRSV2CPPAlgo.h>
#include <CPPAlgos/CountSketchCPPAlgo.h>
#include <CPPAlgos/BCRSCPPAlgo.h>
#include <CPPAlgos/EWSCPPAlgo.h>
#include <CPPAlgos/CoOccurringFDCPPAlgo.h>
#include <CPPAlgos/BetaCoOFDCPPAlgo.h>
#include <CPPAlgos/ProductQuantizationRaw.h>
#include <CPPAlgos/ProductQuantizationHash.h>
#include <CPPAlgos/VectorQuantization.h>
#include <CPPAlgos/INT8CPPAlgo.h>
#include <CPPAlgos/TugOfWarCPPAlgo.h>
```

```
#include <CPPAlgos/WeightedCRCPPAlgo.h>
#include <CPPAlgos/SMPPCACPPAlgo.h>
#include <CPPAlgos/FastJLTCPPAlgo.h>
#include <CPPAlgos/RIPCPPAlgo.h>
#include <CPPAlgos/BlockLRACPPAlgo.h>
#include <CPPAlgos/CLMMCPPAlgo.h>
#include <Utils/ConfigMap.hpp>
#include <Utils/Meters/MeterTable.h>
#include <Utils/C20Buffers.hpp>
#include <Utils/ThreadPerf.hpp>
#include <Utils/IntelliLog.h>
#include <Utils/UtilityFunctions.h>
#include <Utils/BS_thread_pool.hpp>
```

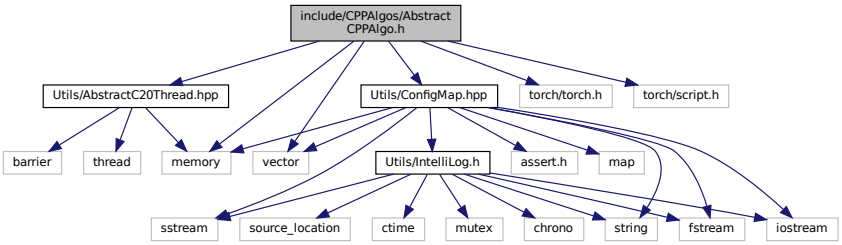
This graph shows which files directly or indirectly include this file:



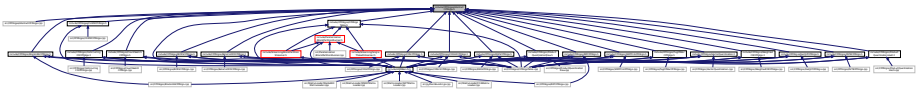
9.2 include/CPPAlgos/AbstractCPPAlgo.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <torch/torch.h>
#include <torch/script.h>
#include <memory>
#include <vector>
```

Include dependency graph for AbstractCPPAlgo.h:



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



Classes

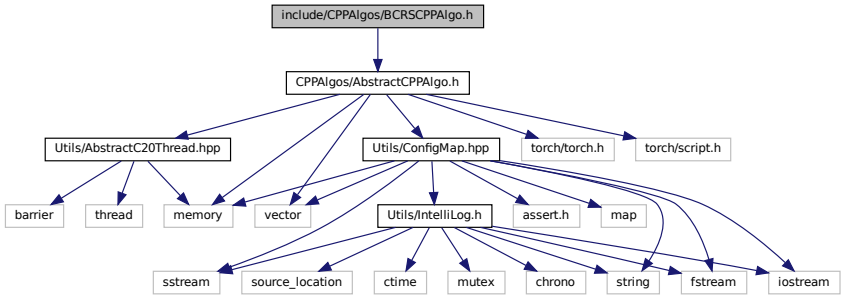
- class AMMBench::AbstractCPPAlgo

  The abstract class of c++ algos.

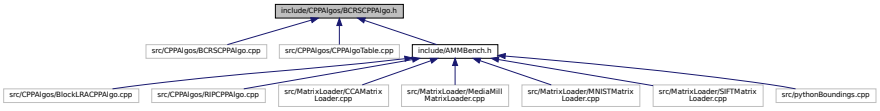
- #define **newAbstractCPPAlgo** std::make_shared<AMMBench::AbstractCPPAlgo>
- typedef std::shared_ptr< class AMMBench::AbstractCPPAlgo > **AMMBench::AbstractCPPAlgoPtr**

9.3 include/CPPAlgos/BCRSCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for BCRSCPPAlgo.h:



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



Classes

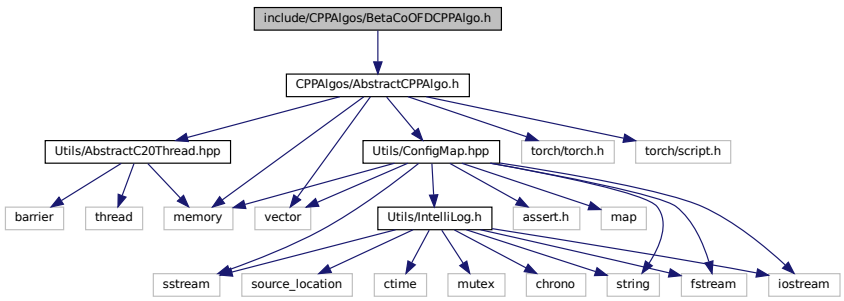
- class AMMBench::BCRSCPPAlgo

  The Bernoulli column row sampling (BCRS) class of c++ algos.

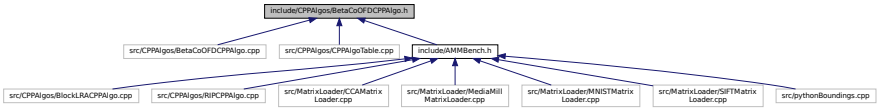
- #define **newBCRSCPPAlgo** std::make_shared<AMMBench::BCRSCPPAlgo>
- typedef std::shared_ptr< class AMMBench::BCRSCPPAlgo > **AMMBench::BCRSCPPAlgoPtr**

9.4 include/CPPAlgos/BetaCoOFDCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for BetaCoOFDCPPAlgo.h:



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



Classes

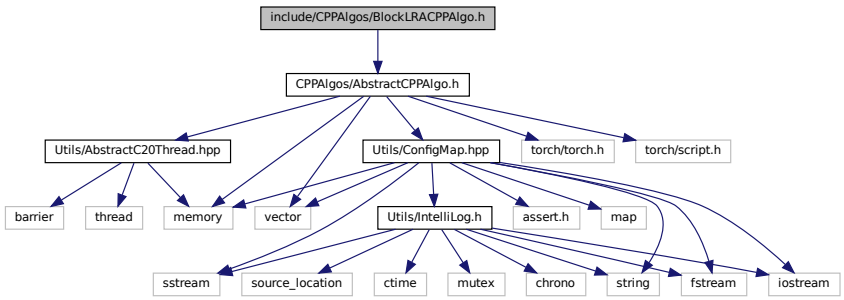
- class AMMBench::BetaCoOFDCPPAlgo

  The Beta Co-Occurring FD AMM class of c++ algos.

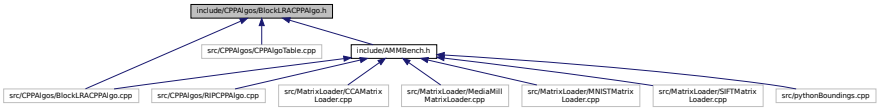
- #define **newBetaCoOFDCPPAlgo** std::make_shared<AMMBench::BetaCoOFDCPPAlgo>
- typedef std::shared_ptr< class AMMBench::BetaCoOFDCPPAlgo > **AMMBench::BetaCoOFDCPPAlgoPtr**

9.5 include/CPPAlgos/BlockLRACPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for BlockLRACPPAlgo.h:



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



Classes

- class AMMBench::BlockLRACPPAlgo

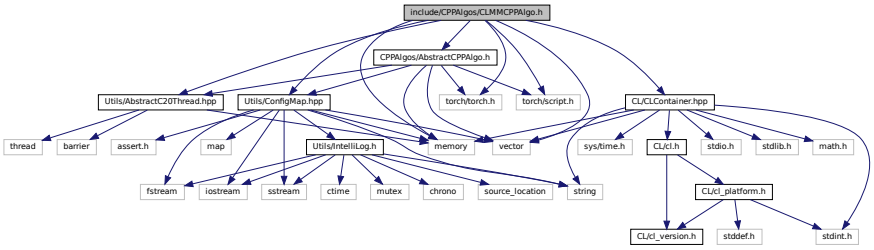
- #define newBlockLRACPPAlgo std::make_shared<AMMBench::BlockLRACPPAlgo>

  (Macro) To creat a new BlockLRACPPAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::BlockLRACPPAlgo > **AMMBench::BlockLRACPPAlgoPtr**

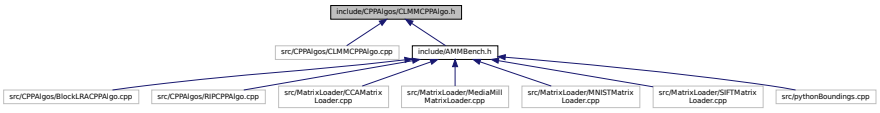
9.6 include/CPPAlgos/CLMMCPPAlgo.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <torch/torch.h>
#include <torch/script.h>
#include <memory>
#include <vector>
#include <CPPAlgos/AbstractCPPAlgo.h>
#include <CL/CLContainer.hpp>
```

Include dependency graph for CLMMCPPAlgo.h:



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



Classes

- class AMMBench::CLMMCPPAlgo

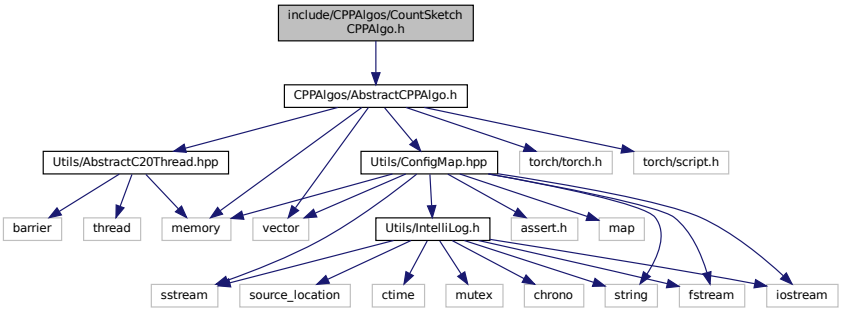
  The MM class of c++ algos using opencl.

- #define **newCLMMCPPAlgo** std::make_shared<AMMBench::CLMMCPPAlgo>
- typedef std::shared_ptr< class AMMBench::CLMMCPPAlgo > **AMMBench::CLMMCPPAlgoPtr**

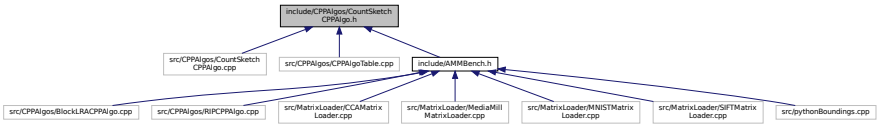
9.7 include/CPPAlgos/CountSketchCPPAlgo.h File Reference

```
#include <CPPAlgos/AbstractCPPAlgo.h>
```

Include dependency graph for CountSketchCPPAlgo.h:



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



Classes

- class AMMBench::CountSketchCPPAlgo

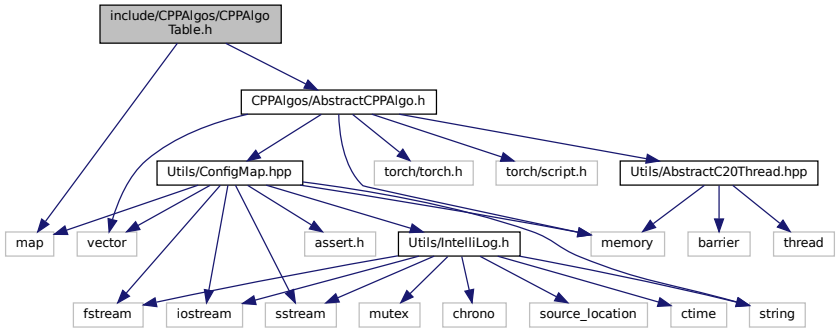
  The counter sketch class of c++ algos.

- #define **newCountSketchCPPAlgo** std::make_shared<AMMBench::CountSketchCPPAlgo>
- typedef std::shared_ptr< class AMMBench::CountSketchCPPAlgo > **AMMBench::CountSketch↩ CPPAlgoPtr**

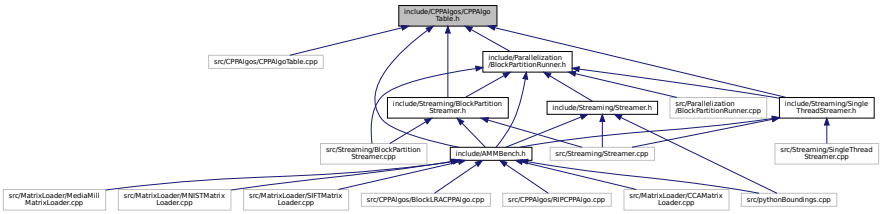
9.8 include/CPPAlgos/CPPAlgoTable.h File Reference

```
#include <map>
#include <CPPAlgos/AbstractCPPAlgo.h>
```

Include dependency graph for CPPAlgoTable.h:



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



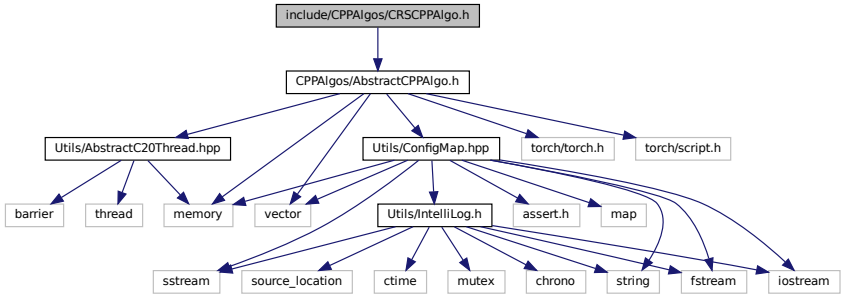
Classes

- class AMMBench::CPPAlgoTable

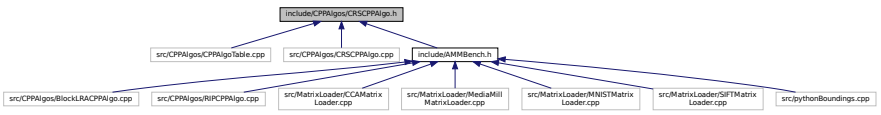
  The table to index cpp algos.

9.9 include/CPPAlgos/CRSCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for CRSCPPAlgo.h:



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



Classes

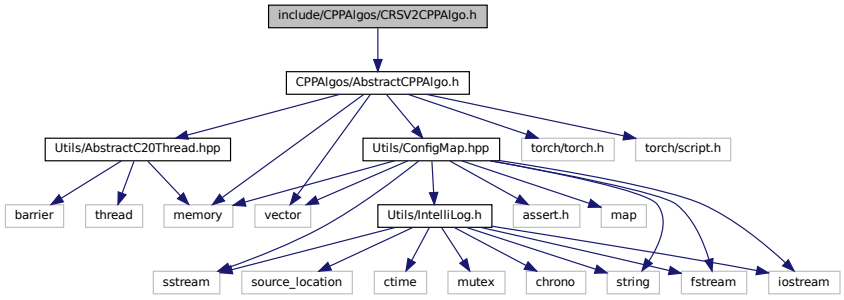
- class AMMBench::CRSCPPAlgo

  The column row sampling (CRS) class of c++ algos.

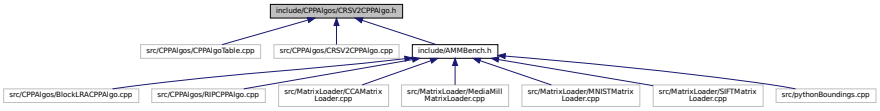
- #define **newCRSCPPAlgo** std::make_shared<AMMBench::CRSCPPAlgo>
- typedef std::shared_ptr< class AMMBench::CRSCPPAlgo > **AMMBench::CRSCPPAlgoPtr**

9.10 include/CPPAlgos/CRSV2CPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for CRSV2CPPAlgo.h:



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



Classes

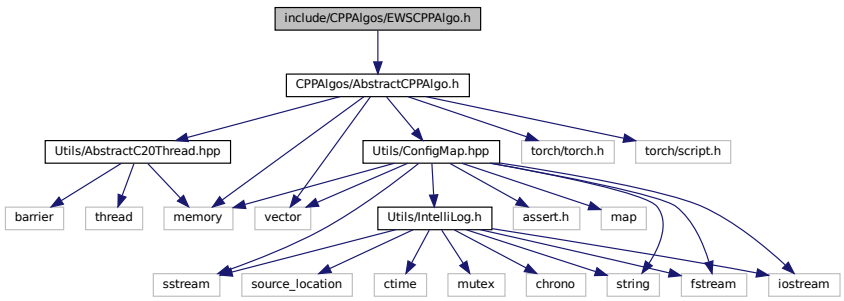
- class AMMBench::CRSV2CPPAlgo

  The column row sampling (CRS) class of c++ algos, a second implementation.

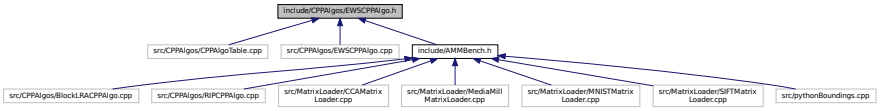
- #define **newCRSV2CPPAlgo** std::make_shared<AMMBench::CRSV2CPPAlgo>
- typedef std::shared_ptr< class AMMBench::CRSV2CPPAlgo > **AMMBench::CRSV2CPPAlgoPtr**

9.11 include/CPPAlgos/EWSCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for EWSCPPAlgo.h:



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



Classes

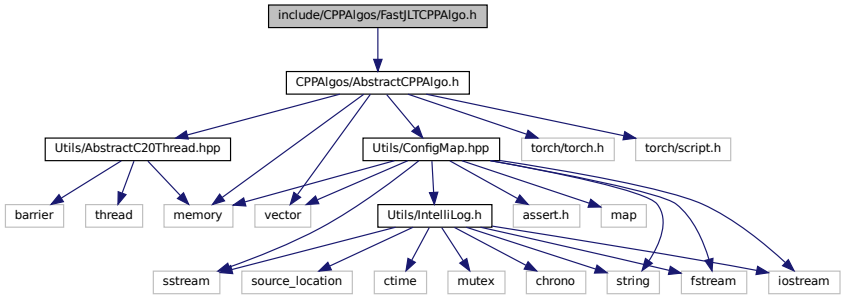
- class AMMBench::EWSCPPAlgo

  The Element Wise Sampling (EWS) class of c++ algos.

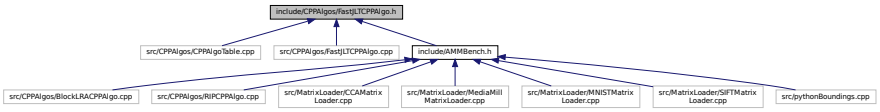
- #define **newEWSCPPAlgo** std::make_shared<AMMBench::EWSCPPAlgo>
- typedef std::shared_ptr< class AMMBench::EWSCPPAlgo > **AMMBench::EWSCPPAlgoPtr**

9.12 include/CPPAlgos/FastJLTCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for FastJLTCPPAlgo.h:



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



Classes

- class AMMBench::FastJLTCPPAlgo

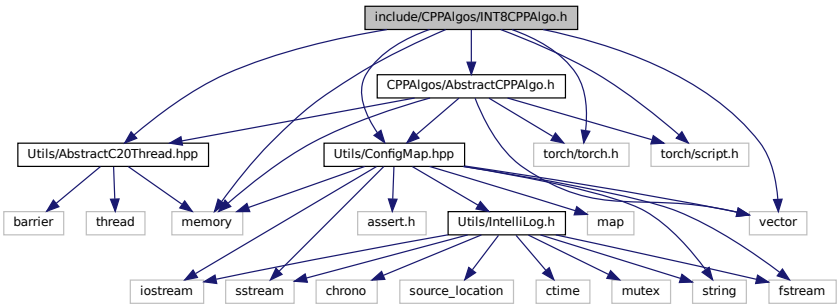
  The tug of war class of c++ algoS.

- #define **newFastJLTCPPAlgo** std::make_shared<AMMBench::FastJLTCPPAlgo>
- typedef std::shared_ptr< class AMMBench::FastJLTCPPAlgo > **AMMBench::FastJLTCPPAlgoPtr**

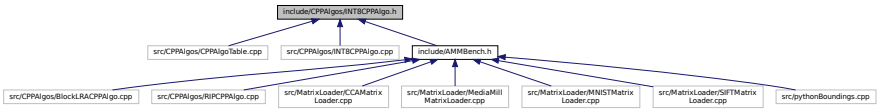
9.13 include/CPPAlgos/INT8CPPAlgo.h File Reference

```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <torch/torch.h>
#include <torch/script.h>
#include <memory>
#include <vector>
#include <CPPAlgos/AbstractCPPAlgo.h>
```

Include dependency graph for INT8CPPAlgo.h:



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



Classes

- class AMMBench::INT8CPPAlgo

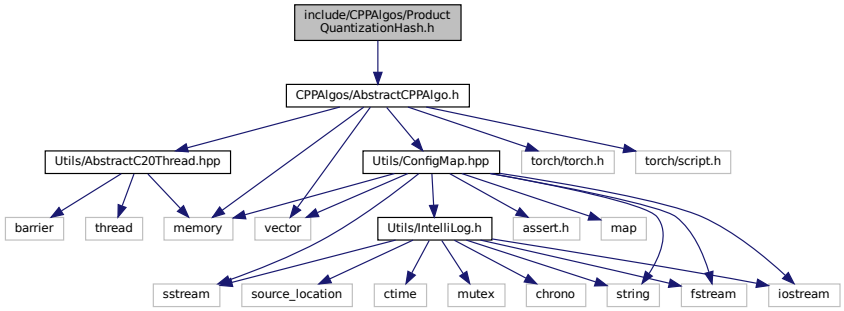
  The INT8 MM class of c++ algos.

- #define **newINT8CPPAlgo** std::make_shared<AMMBench::INT8CPPAlgo>
- typedef std::shared_ptr< class AMMBench::INT8CPPAlgo > **AMMBench::INT8CPPAlgoPtr**

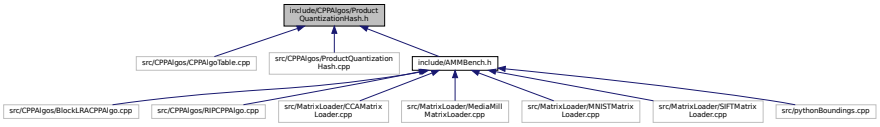
9.14 include/CPPAlgos/ProductQuantizationHash.h File Reference

```
#include <CPPAlgos/AbstractCPPAlgo.h>
```

Include dependency graph for ProductQuantizationHash.h:



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



Classes

- class AMMBench::ProductQuantizationHash

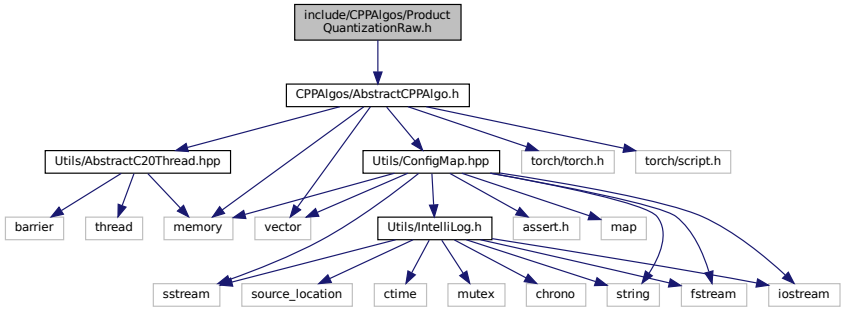
  The Product Quantization AMM class of c++ algos, using hash function to find matching prototypes.

- #define newProductQuantizationHashAlgo std::make_shared<AMMBench::ProductQuantizationHash>

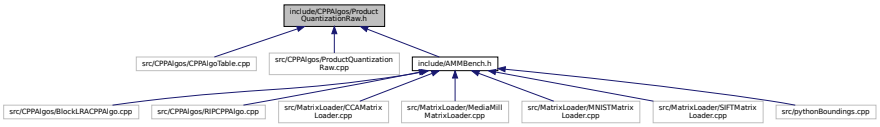
  (Macro) To creat a new ProductQuantizationHashAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::ProductQuantizationHash > **AMMBench::Product↩ QuantizationHashPtr**

9.15 include/CPPAlgos/ProductQuantizationRaw.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for ProductQuantizationRaw.h:



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



Classes

- class AMMBench::ProductQuantizationRaw

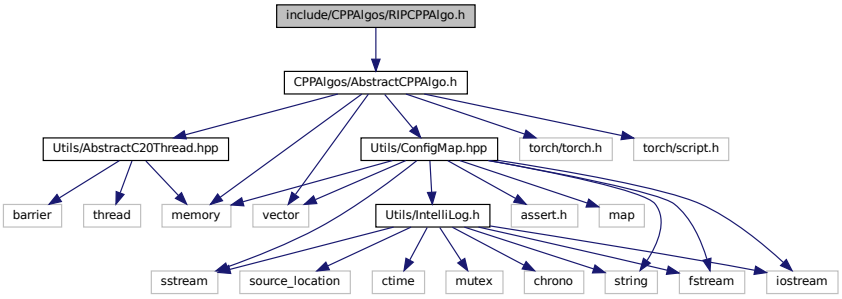
  The Product Quantization AMM class of c++ algos, using Euclidean distance.

- #define newProductQuantizationRawAlgo std::make_shared<AMMBench::ProductQuantizationRaw>

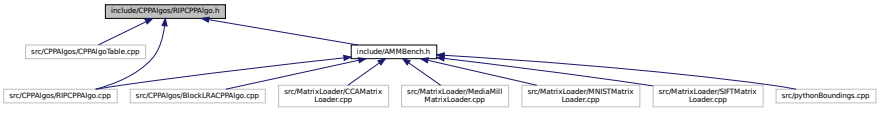
  (Macro) To creat a new ProductQuantizationRawAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::ProductQuantizationRaw > **AMMBench::ProductQuantizationRawPtr**

9.16 include/CPPAlgos/RIPCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for RIPCPPAlgo.h:



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



Classes

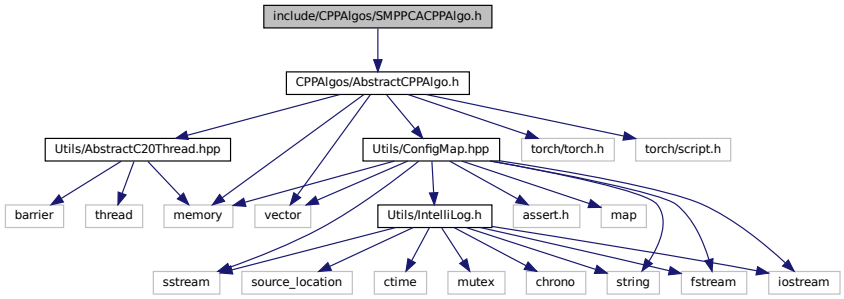
- class AMMBench::RIPCPPAlgo

  New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property.

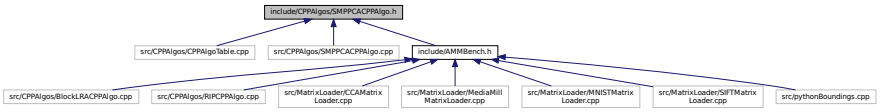
- #define **newRIPCPPAlgo** std::make_shared<AMMBench::RIPCPPAlgo>
- typedef std::shared_ptr< class AMMBench::RIPCPPAlgo > **AMMBench::RIPCPPAlgoPtr**

9.17 include/CPPAlgos/SMPPCACPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for SMPPCACPPAlgo.h:



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



Classes

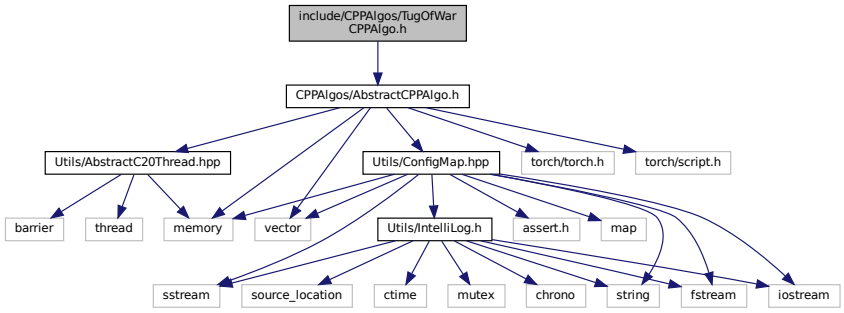
- class AMMBench::SMPPCACPPAlgo

  sketch scaled JL class of c++ algos

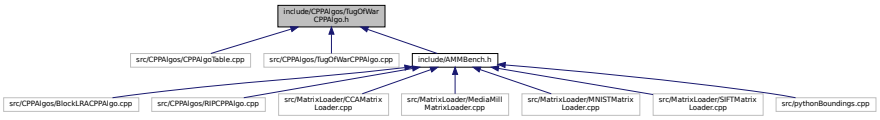
- #define **newSMPPCACPPAlgo** std::make_shared<AMMBench::SMPPCACPPAlgo>
- typedef std::shared_ptr< class AMMBench::SMPPCACPPAlgo > **AMMBench::SMPPCACPPAlgoPtr**

9.18 include/CPPAlgos/TugOfWarCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for TugOfWarCPPAlgo.h:



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



Classes

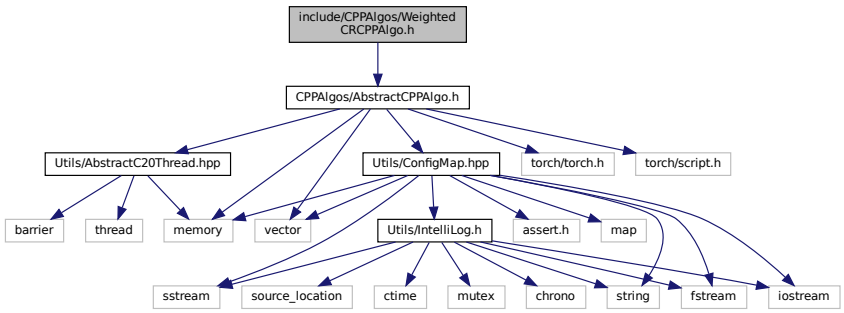
- class AMMBench::TugOfWarCPPAlgo

  The tug of war class of c++ algoS.

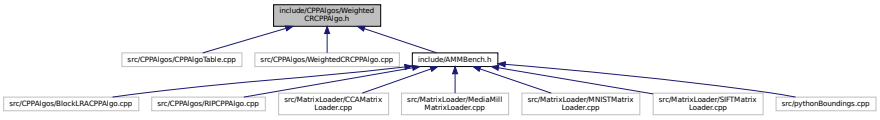
- #define **newTugOfWarCPPAlgo** std::make_shared<AMMBench::TugOfWarCPPAlgo>
- typedef std::shared_ptr< class AMMBench::TugOfWarCPPAlgo > **AMMBench::TugOfWarCPPAlgoPtr**

9.19 include/CPPAlgos/WeightedCRCPPAlgo.h File Reference

`#include <CPPAlgos/AbstractCPPAlgo.h>`
Include dependency graph for WeightedCRCPPAlgo.h:



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



Classes

- class AMMBench::WeightedCRCPPAlgo

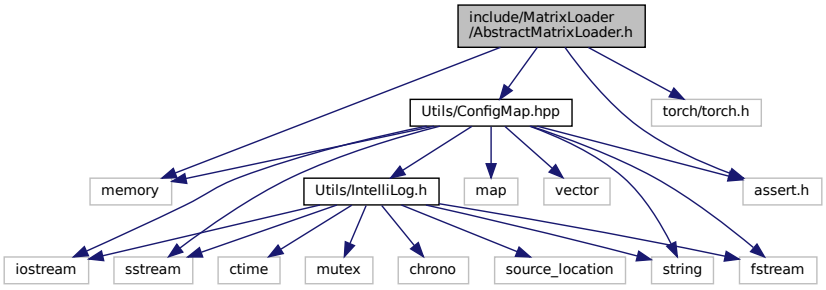
- #define newWeightedCRCPPAlgo std::make_shared<AMMBench::WeightedCRCPPAlgo>

  (Macro) To creat a new WeightedCRCPPAlgounder shared pointer.
- typedef std::shared_ptr< class AMMBench::WeightedCRCPPAlgo > **AMMBench::WeightedCRCPPAlgo↩ Ptr**

9.20 include/MatrixLoader/AbstractMatrixLoader.h File Reference

```
#include <Utils/ConfigMap.hpp>
#include <assert.h>
#include <torch/torch.h>
#include <memory>
```

Include dependency graph for AbstractMatrixLoader.h:



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

Classes

- class AMMBench::AbstractMatrixLoader

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

- #define newAbstractMatrixLoader std::make_shared<AMMBench::AbstractMatrixLoader>

  (Macro) To creat a new AbstractMatrixLoader under shared pointer.
- typedef std::shared_ptr< class AMMBench::AbstractMatrixLoader > AMMBench::AbstractMatrixLoaderPtr

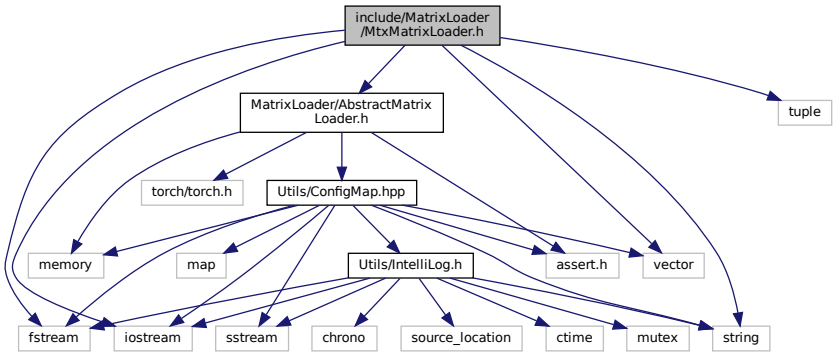
  The class to describe a shared pointer to AbstractMatrixLoader.

9.21 include/MatrixLoader/MtxMatrixLoader.h File Reference

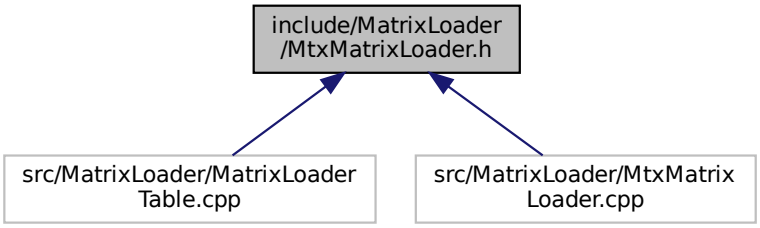
```
#include <MatrixLoader/AbstractMatrixLoader.h>
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
```

```
#include <tuple>
```

Include dependency graph for MtxMatrixLoader.h:



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



Classes

- class AMMBench::MtxMatrixLoader

  The matrix loader to load matrixes stored in matrix market mtx format.

- #define newMtxMatrixLoader std::make_shared<AMMBench::MtxMatrixLoader>

  (Macro) To creat a new MtxMatrixLoader under shared pointer.
- typedef std::shared_ptr< class AMMBench::MtxMatrixLoader > AMMBench::MtxMatrixLoaderPtr

  The class to describe a shared pointer to MtxMatrixLoader.
- torch::Tensor AMMBench::loadMatrixFromMatrixMarket (const string &filename)

  the stan-alone function to load a matrix from matrix market mitx file
- torch::Tensor AMMBench::normalizeIntoPN1 (torch::Tensor a)

  to normalize a tensor into +-1: will be biased by the min value
- torch::Tensor AMMBench::scaleIntoPN1 (torch::Tensor a)

  to scale a tensor into +-1: will NOT change the bias

9.21.1 Function Documentation

9.21.1.1 normalizeIntoPN1()

```
torch::Tensor AMMBench::normalizeIntoPN1 (
            torch::Tensor a )
```

to normalize a tensor into +-1: will be biased by the min value

Parameters

a	the input

Returns

the normalized tensor

9.21.1.2 scaleIntoPN1()

```
torch::Tensor AMMBench::scaleIntoPN1 (
            torch::Tensor a )
```

to scale a tensor into +-1: will NOT change the bias

Parameters

a	the input

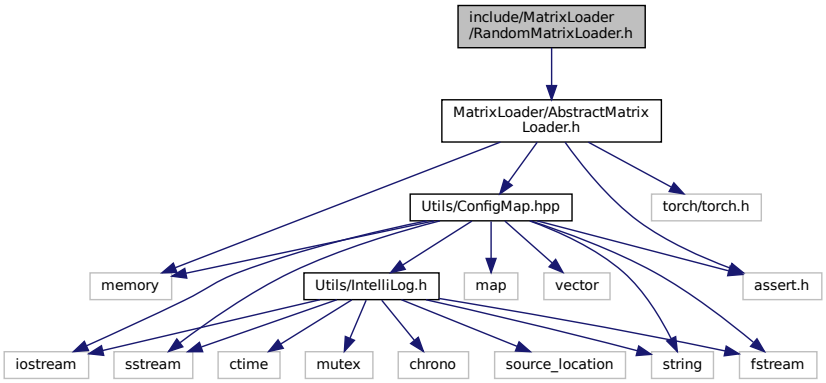
Returns

the normalized tensor

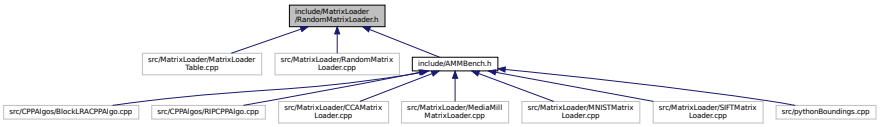
9.22 include/MatrixLoader/RandomMatrixLoader.h File Reference

```
#include <MatrixLoader/AbstractMatrixLoader.h>
```

Include dependency graph for RandomMatrixLoader.h:



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



Classes

- class AMMBench::RandomMatrixLoader

  The Random class of matrix loader.

- #define newRandomMatrixLoader std::make_shared<AMMBench::RandomMatrixLoader>

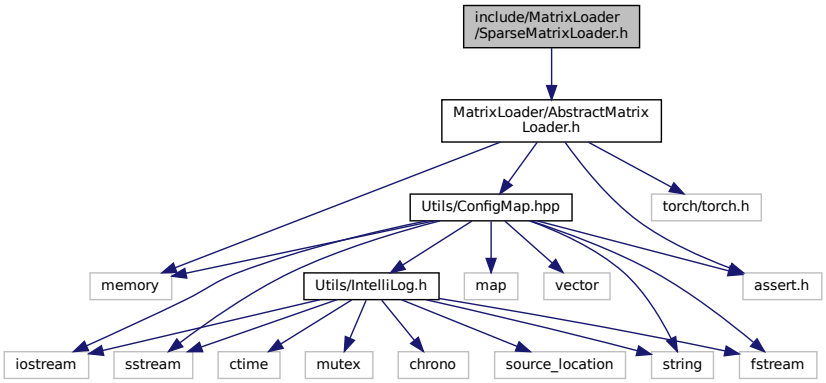
  (Macro) To creat a new RandomMatrixLoader under shared pointer.
- typedef std::shared_ptr< class AMMBench::RandomMatrixLoader > AMMBench::RandomMatrixLoaderPtr

  The class to describe a shared pointer to RandomMatrixLoader.

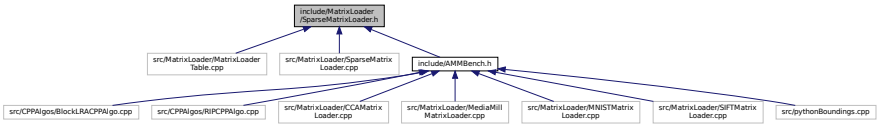
9.23 include/MatrixLoader/SparseMatrixLoader.h File Reference

```
#include <MatrixLoader/AbstractMatrixLoader.h>
```

Include dependency graph for SparseMatrixLoader.h:



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



Classes

- class AMMBench::SparseMatrixLoader

  The matrix loader to generate adjustable sparse matrix with adjust rank reduction.

- #define newSparseMatrixLoader std::make_shared<AMMBench::SparseMatrixLoader>

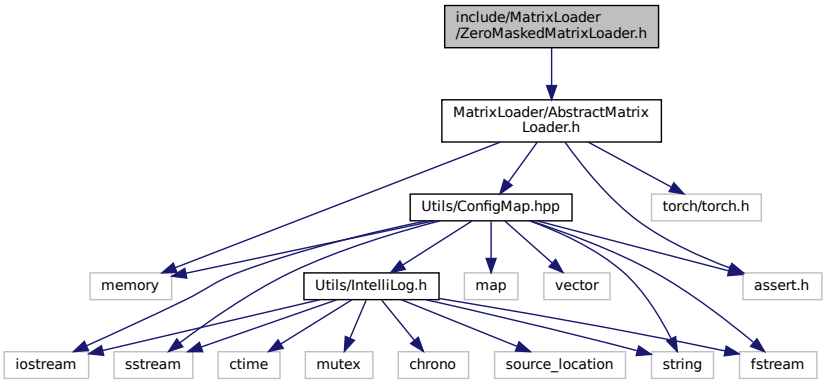
  (Macro) To creat a new SparseMatrixLoader under shared pointer.
- typedef std::shared_ptr< class AMMBench::SparseMatrixLoader > AMMBench::SparseMatrixLoaderPtr

  The class to describe a shared pointer to SparseMatrixLoader.

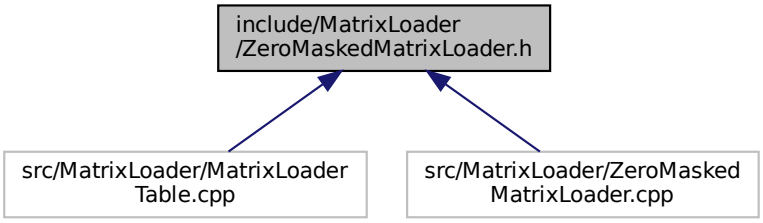
9.24 include/MatrixLoader/ZeroMaskedMatrixLoader.h File Reference

```
#include <MatrixLoader/AbstractMatrixLoader.h>
```

Include dependency graph for ZeroMaskedMatrixLoader.h:



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



Classes

- class AMMBench::ZeroMaskedMatrixLoader

  The zero masked class of matrix loader, given generate a n∗m matrix, where only the left-top n1∗m2 contents are not zero.

- #define newZeroMaskedMatrixLoader std::make_shared<AMMBench::ZeroMaskedMatrixLoader>

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

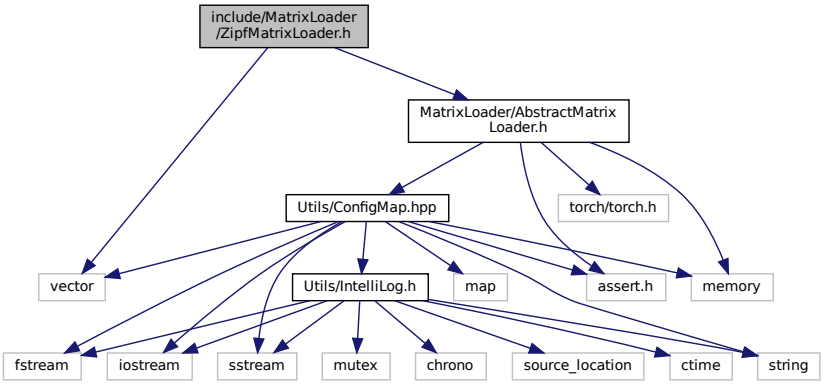
- typedef std::shared_ptr< class AMMBench::ZeroMaskedMatrixLoader > AMMBench::ZeroMaskedMatrixLoaderPtr

  The class to describe a shared pointer to ZeroMaskedMatrixLoader.

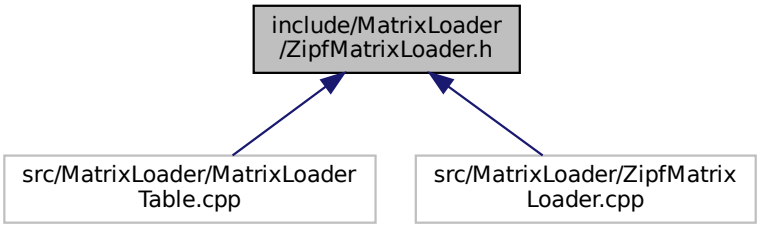
9.25 include/MatrixLoader/ZipfMatrixLoader.h File Reference

```
#include <MatrixLoader/AbstractMatrixLoader.h>
#include <vector>
```

Include dependency graph for ZipfMatrixLoader.h:



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



Classes

- class AMMBench::ZipfMatrixLoader

  The Zipf class of matrix loader.

- #define newZipfMatrixLoader std::make_shared<AMMBench::ZipfMatrixLoader>

  (Macro) To creat a new ZipfMatrixLoader under shared pointer.
- typedef std::shared_ptr< class AMMBench::ZipfMatrixLoader > AMMBench::ZipfMatrixLoaderPtr

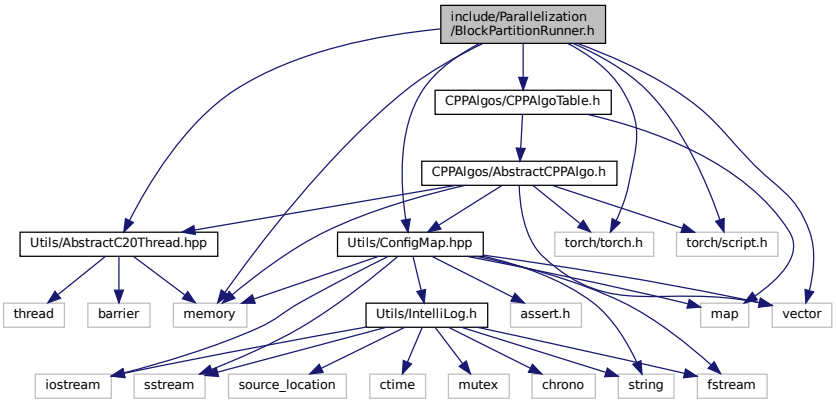
  The class to describe a shared pointer to ZipfMatrixLoader.

9.26 include/Parallelization/BlockPartitionRunner.h File Reference

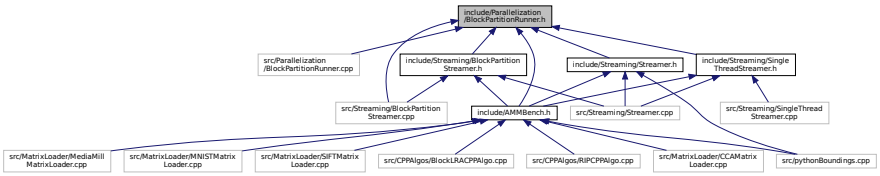
```
#include <Utils/AbstractC20Thread.hpp>
#include <Utils/ConfigMap.hpp>
#include <torch/torch.h>
#include <torch/script.h>
```

```
#include <memory>
#include <vector>
#include <CPPAlgos/CPPAlgoTable.h>
```

Include dependency graph for BlockPartitionRunner.h:



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



Classes

- class AMMBench::BlockPartitionWorker

  The basic partition worker.
- class AMMBench::BlockPartitionRunner

  The top entity to control all workers, see also BlockPartitionWorker. This one works under a simple row partition parallelization.

Macros

- #define **newTensor** make_shared<torch::Tensor>

Typedefs

- typedef std::shared_ptr< torch::Tensor > **AMMBench::TensorPtr**

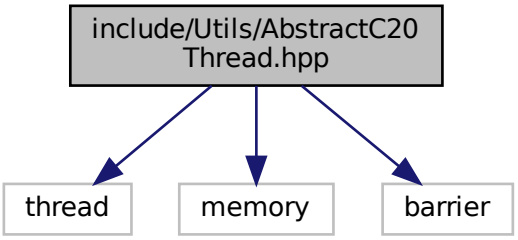
- #define newBlockPartitionWorker std::make_shared<AMMBench::BlockPartitionWorker>

  (Macro) To creat a new BlockPartitionWorker under shared pointer.
- typedef std::shared_ptr< AMMBench::BlockPartitionWorker > **AMMBench::BlockPartitionWorkerPtr**

9.27 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

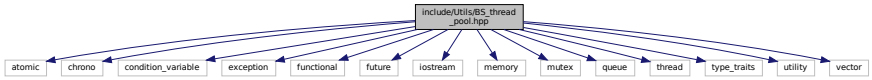
  The class to describe a shared pointer to AbstractC20Thread.
- typedef std::shared_ptr< std::barrier<> > **INTELLI::BarrierPtr**

9.28 include/Utils/BS_thread_pool.hpp File Reference

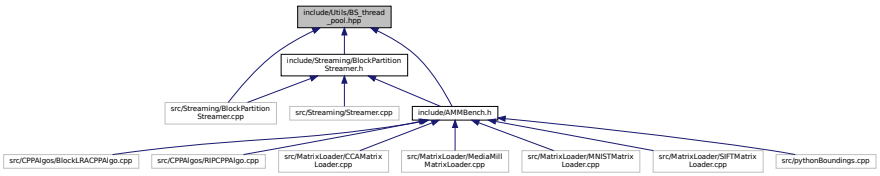
BS::thread_pool: a fast, lightweight, and easy-to-use C++17 thread pool library. This header file contains the entire library, including the main BS::thread_pool class and the helper classes BS::multi_future, BS::blocks, BS:synced↩ _stream, and BS::timer.

```
#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:



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



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**

9.28.1 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

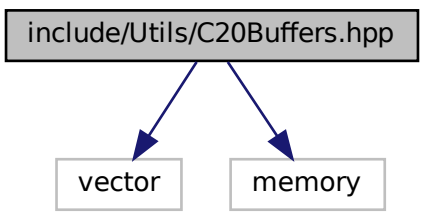
Copyright

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)

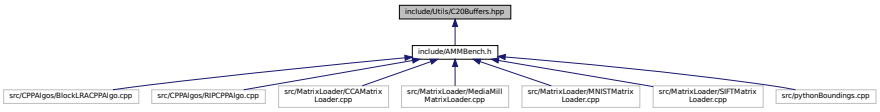
9.29 include/Utils/C20Buffers.hpp File Reference

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

Include dependency graph for C20Buffers.hpp:



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



Classes

- class INTELLI::C20Buffer< dataType >

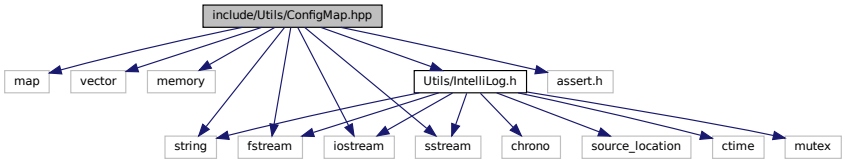
Macros

- #define **_UTILS_C20BUFFERS_HPP_**
- #define **ADB_memcpy**(dst, src, size) memcpy(dst, src, size)

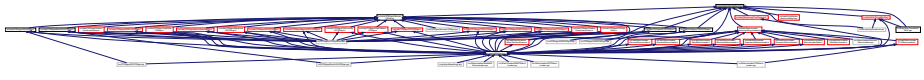
9.30 include/Utils/ConfigMap.hpp File Reference

```
#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

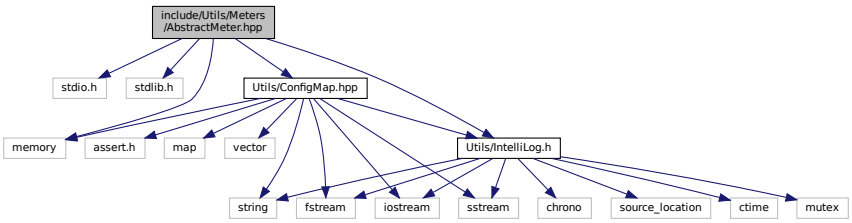
- typedef std::shared_ptr< ConfigMap > INTELLI::ConfigMapPtr

  The class to describe a shared pointer to ConfigMap.

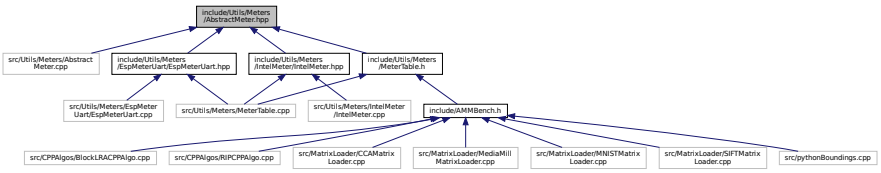
9.31 include/Utils/Meters/AbstractMeter.hpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <Utils/ConfigMap.hpp>
#include <Utils/IntelliLog.h>
#include <memory>
```

Include dependency graph for AbstractMeter.hpp:



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



Classes

- class DIVERSE_METER::AbstractMeter

  The abstract class for all meters.

Macros

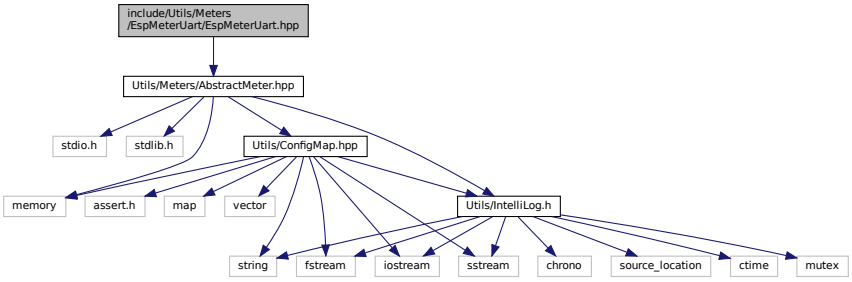
- #define **METER_ERROR**(n) INTELLI_ERROR(n)

Typedefs

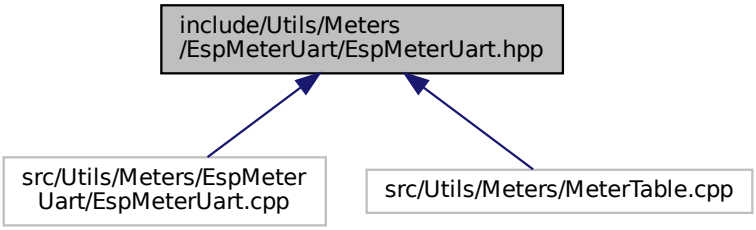
- typedef std::shared_ptr< DIVERSE_METER::AbstractMeter > **DIVERSE_METER::AbstractMeterPtr**

9.32 include/Utils/Meters/EspMeterUart/EspMeterUart.hpp File Reference

`#include <Utils/Meters/AbstractMeter.hpp>`
Include dependency graph for EspMeterUart.hpp:



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



Classes

- class DIVERSE_METER::EspMeterUart

  the entity of an esp32s2-based power meter, connected by uart 115200

Macros

- #define **ADB_INCLUDE_UTILS_EspMeterUartUART_HPP_**
- #define **newEspMeterUart**() std::make_shared<EspMeterUart>();

Typedefs

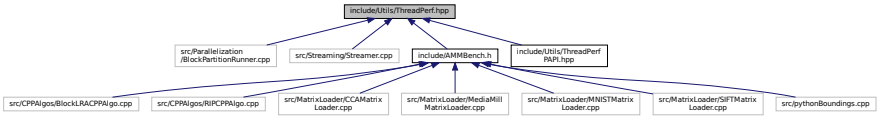
- typedef std::shared_ptr< DIVERSE_METER::EspMeterUart > **DIVERSE_METER::EspMeterUartPtr**

9.33 include/Utils/ThreadPerf.hpp File Reference

```
#include <string>
#include <sys/time.h>
#include <assert.h>
#include <fcntl.h>
#include <inttypes.h>
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <time.h>
#include <unistd.h>
#include <linux/perf_event.h>
#include <signal.h>
#include <memory.h>
#include <memory>
#include <vector>
#include <Utils/ConfigMap.hpp>
```

Include dependency graph for ThreadPerf.hpp:

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



Classes

- class INTELLI::ThreadPerf

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

  a record pair of perf events
- class INTELLI::ThreadPerf::PerfTool

Macros

- #define **PERF_ERROR**(n) printf(n)
- #define **LIBPERF_ARRAY_SIZE**(x) (sizeof(x)/sizeof(x[0]))
- #define newThreadPerf std::make_shared<INTELLI::ThreadPerf>

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

Typedefs

- typedef std::shared_ptr< INTELLI::ThreadPerf > INTELLI::ThreadPerfPtr

  The class to describe a shared pointer to ThreadPerf.

Enumerations

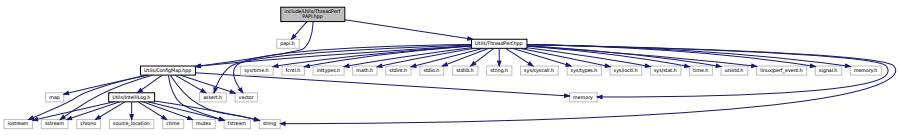
- enum INTELLI::perfTrace {
  COUNT_SW_CPU_CLOCK = 0 , **COUNT_SW_TASK_CLOCK** = 1 , **COUNT_SW_CONTEXT_SWITCHES** = 2 , **COUNT_SW_CPU_MIGRATIONS** = 3 ,
  COUNT_SW_PAGE_FAULTS = 4 , **COUNT_SW_PAGE_FAULTS_MIN** = 5 , **COUNT_SW_PAGE_FAULTS_MAJ** = 6 , **COUNT_HW_CPU_CYCLES** = 7 ,
  COUNT_HW_INSTRUCTIONS = 8 , **COUNT_HW_CACHE_REFERENCES** = 9 , **COUNT_HW_CACHE_MISSES** = 10 , **COUNT_HW_BRANCH_INSTRUCTIONS** = 11 ,
  COUNT_HW_BRANCH_MISSES = 12 , **COUNT_HW_BUS_CYCLES** = 13 , **COUNT_HW_CACHE_L1D_LOADS** = 14 , **COUNT_HW_CACHE_L1D_LOADS_MISSES** = 15 ,
  COUNT_HW_CACHE_L1D_STORES = 16 , **COUNT_HW_CACHE_L1D_STORES_MISSES** = 17 , **COUNT_HW_CACHE_L1D_PREFETCHES** = 18 , **COUNT_HW_CACHE_L1I_LOADS** = 19 ,
  COUNT_HW_CACHE_L1I_LOADS_MISSES = 20 , **COUNT_HW_CACHE_LL_LOADS** = 21 , **COUNT_HW_CACHE_LL_LOADS_MISSES** = 22 , **COUNT_HW_CACHE_LL_STORES** = 23 ,
  COUNT_HW_CACHE_LL_STORES_MISSES = 24 , **COUNT_HW_CACHE_DTLB_LOADS** = 25 , **COUNT_HW_CACHE_DTLB_LOADS_MISSES** = 26 , **COUNT_HW_CACHE_DTLB_STORES** = 27 ,
  COUNT_HW_CACHE_DTLB_STORES_MISSES = 28 , **COUNT_HW_CACHE_ITLB_LOADS** = 29 , **COUNT_HW_CACHE_ITLB_LOADS_MISSES** = 30 , **COUNT_HW_CACHE_BPU_LOADS** = 31 ,
  COUNT_HW_CACHE_BPU_LOADS_MISSES = 32 }

  The low level description of perf events, used inside, don't touch me UNLESS you know what you are doing.

9.34 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.

Index