libfaster API Documentation

Development Version

Generated by Doxygen 1.8.12

Contents

Chapter 1

API Introduction

Faster defines the `faster` namespace which contains all framework classes and definitions.

The context class is the class that manages dataset resources and task execution.

- faster::fastContext class

The user can create, using the context class several types of distributted datasets:

- faster::fdd - a dataset of a single type.
- faster::indexedFdd dataset - a indexed dataset containing a key and a value.
- faster::groupedFdd dataset class - a group of indexed datasets.

Step by step

In order to run code using faster you need:

1. Create a context object (faster::fastContext)
2. Register user functions and variables (faster::fastContext::registerFunction)
3. Start worker processes (faster::fastContext::startWorkers)
4. Create a dataset from file or memory (faster::fdd::fdd() or faster::indexedFdd::indexedFdd())
5. Apply your functions to the dataset (faster::fdd::map() or faster::fdd::reduce() etc.)
6. Write the dataset to disk or collect its content (faster::fddCore::writeToFile(), faster::fdd::collect())

Examples

- Examples Full working examples

Chapter 2

Operator Groups

Operators can be divided by behaviour and variants, but also, there are special operator reserved for grouped datasets.

There are four main operator behaviour:

- `Update`
- `Map`
- `FlatMap`
- `Reduce`

Also, there are two variants:

- `Bulk`
- `ByKey`

Also, when two or more datasets are grouped together, some functions listed before can be used:

- `Grouped`

Chapter 3

Examples

Faster has full working examples at src/examples directory.

Some toy examples:

- fexample-int.cpp - A example applying map and reduce to a faster::fdd <int> created from memory
- fexample-int-file.cpp - A example applying map and reduce to a faster::fdd <int> created from file
- fexample-int-vector.cpp - A example applying map and reduce to a faster::fdd <vector<int>> created from memory
- fexample-indexed.cpp - A example applying map and reduce to a faster::indexedFdd <int,int> created from memory

Some algorithm implementations using Faster:

- pagerank.cpp - A pagerank implementation without using bulk functions
- pagerank-bulk2.cpp - A pagerank implementation without using bulk functions

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Namespace Index

5.1 Namespace List

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

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

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

Chapter 7

Class Index

7.1 Class List

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

Chapter 8

Module Documentation

8.1 Update Operators

8.1.1 Description

Run a iterative update operaton.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *void F(T&)* that will be used on each dataset entry

Returns

A pointer to a new dataset

Functions

- indexedFdd< K, T > * faster::indexedFdd< K, T >::update (updateIFunctionP< K, T > funcP)
  updates the content of a indexedFDD

Typedefs

- template<typename K , typename T >
  using **faster::updateIFunctionP** = void(*)(K &inKey, T &input)

8.2 Map Operators

8.2.1 Description

Run a **n to n** map operaton.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *U F(T&)* that will be used on each dataset entry

Returns

A pointer to a new dataset

Functions

- template<typename U >
  fdd< U > ∗ faster::fdd< T >::map (mapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ faster::fdd< T >::map (PmapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::map (ImapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::map (IPmapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::map (ImapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::map (IPmapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::map (mapIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::map (PmapIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>

Typedefs

- template<typename T , typename U >
  using **faster::mapFunctionP** = U(∗)(T &input)
- template<typename T , typename L , typename U >
  using **faster::lmapFunctionP** = std::pair< L, U >(∗)(T &input)
- template<typename T , typename U >
  using **faster::PmapFunctionP** = std::pair< U, size_t >(∗)(T &input)
- template<typename T , typename L , typename U >
  using **faster::lPmapFunctionP** = std::tuple< L, U, size_t >(∗)(T &input)
- template<typename T , typename U >
  using **faster::mapPFunctionP** = U(∗)(T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::lmapPFunctionP** = std::pair< L, U >(∗)(T ∗input, size_t size)
- template<typename T , typename U >
  using **faster::PmapPFunctionP** = std::pair< U, size_t >(∗)(T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::lPmapPFunctionP** = std::tuple< L, U, size_t >(∗)(T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::lmapIFunctionP** = std::pair< L, U >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename U >
  using **faster::mapIFunctionP** = U(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **faster::lPmapIFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename U >
  using **faster::PmapIFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **faster::lmapIPFunctionP** = std::pair< L, U >(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::mapIPFunctionP** = U(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::lPmapIPFunctionP** = std::tuple< L, U, size_t >(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PmapIPFunctionP** = std::pair< U, size_t >(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename To >
  using **faster::mapByKeyG2FunctionP** = To(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename To >
  using **faster::mapByKeyG3FunctionP** = To(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename Ko , typename To >
  using **faster::lmapByKeyG2FunctionP** = std::pair< Ko, To >(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename Ko , typename To >
  using **faster::lmapByKeyG3FunctionP** = std::pair< Ko, To >(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)

8.3 FlatMap Operators

8.3.1 Description

Run a **n to m** flatMap operation.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *deque<T> F(T,T)* that will be used on each dataset entry

Returns

A pointer to a new dataset

Functions

- template<typename U >
  fdd< U > ∗ faster::fdd< T >::flatMap (flatMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ faster::fdd< T >::flatMap (PflatMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::flatMap (IflatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::flatMap (IPflatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::flatMap (IflatMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::flatMap (IPflatMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::flatMap (flatMapIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::flatMap (PflatMapIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>
- std::pair< K, T > faster::indexedFdd< K, T >::reduce (IreduceIFunctionP< K, T > funcP)

  summarizes a fdd<K,T> into a single value of type T

Typedefs

- template<typename T , typename U >
  using **faster::flatMapFunctionP** = std::deque< U >(∗)(T &input)
- template<typename T , typename L , typename U >
  using **faster::IflatMapFunctionP** = std::deque< std::pair< L, U >>(∗)(T &input)
- template<typename T , typename U >
  using **faster::PflatMapFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T &input)
- template<typename T , typename L , typename U >
  using **faster::IPflatMapFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T &input)
- template<typename T , typename U >
  using **faster::flatMapPFunctionP** = std::deque< U >(∗)(T ∗&input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IflatMapPFunctionP** = std::deque< std::pair< L, U >>(∗)(T ∗&input, size_t size)
- template<typename T , typename U >
  using **faster::PflatMapPFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IPflatMapPFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IflatMapIFunctionP** = std::deque< std::pair< L, U >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename U >
  using **faster::flatMapIFunctionP** = std::deque< U >(∗)(K inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **faster::IPflatMapIFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename U >
  using **faster::PflatMapIFunctionP** = std::deque< std::pair< U, size_t >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **faster::IflatMapIPFunctionP** = std::deque< std::pair< L, U >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::flatMapIPFunctionP** = std::deque< U >(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IPflatMapIPFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PflatMapIPFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T ∗&input, size_t size)

8.4 Reduce Operators

8.4.1 Description

Run a **n to 1** reduce.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
funcP	- A function pointer of a user function *T F(T,T)* that will be used to summarize values

Returns

summarized value of type T

Functions

- T faster::fdd< T >::reduce (reduceFunctionP< T > funcP)

  summarizes a fdd<T> into a single value of type T

Typedefs

- template<typename T >
  using **faster::reduceFunctionP** = T(∗)(T &a, T &b)
- template<typename T >
  using **faster::PreducePFunctionP** = std::pair< T ∗, size_t >(∗)(T ∗a, size_t sizeA, T ∗b, size_t sizeB)
- template<typename K , typename T >
  using **faster::IreduceIFunctionP** = std::pair< K, T >(∗)(const K &keyA, T &a, const K &keyB, T &b)
- template<typename K , typename T >
  using **faster::IPreduceIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K keyA, T ∗a, size_t sizeA, K keyB, T ∗b, size_t sizeB)

8.5 Bulk Operator Variants

8.5.1 Description

A variant of original operators that receive multiple entries of a dataset at the same time.

Bulk operators use user functions that can access multiple entries of the local dataset at the same time *U F(T∗, size_t)*.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *U F(T&)* that will be used on each dataset entry

Returns

A pointer to a new dataset

Functions

- template<typename U >
  fdd< U > ∗ faster::fdd< T >::bulkMap (bulkMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ faster::fdd< T >::bulkMap (PbulkMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::bulkMap (IbulkMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::bulkMap (IPbulkMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ faster::fdd< T >::bulkFlatMap (bulkFlatMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ faster::fdd< T >::bulkFlatMap (PbulkFlatMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::bulkFlatMap (IbulkFlatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::fdd< T >::bulkFlatMap (IPbulkFlatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- T faster::fdd< T >::bulkReduce (bulkReduceFunctionP< T > funcP)

summarizes a fdd< T> into a single value of type T using a bulk function T F(T,T)

- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::bulkMap (IbulkMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::bulkMap (IPbulkMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::bulkMap (bulkMapIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::bulkMap (PbulkMapIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::bulkFlatMap (IbulkFlatMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ faster::indexedFdd< K, T >::bulkFlatMap (IPbulkFlatMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::bulkFlatMap (bulkFlatMapIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ faster::indexedFdd< K, T >::bulkFlatMap (PbulkFlatMapIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>
- std::pair< K, T > faster::indexedFdd< K, T >::bulkReduce (IbulkReduceIFunctionP< K, T > funcP)

  summarizes a fdd<K,T> into a single value of type T using a bulk function pair<K,T> F(K, T, K, T)

Typedefs

- template<typename T , typename U >
  using **faster::bulkMapFunctionP** = void(∗)(U ∗output, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IbulkMapFunctionP** = void(∗)(L ∗outKey, U ∗output, T ∗input, size_t size)
- template<typename T , typename U >
  using **faster::PbulkMapFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IPbulkMapFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, T ∗input, size↩
  _t size)
- template<typename T , typename U >
  using **faster::bulkFlatMapFunctionP** = void(∗)(U ∗&output, size_t &outputSize, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IbulkFlatMapFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, T ∗input, size↩
  _t size)
- template<typename T , typename U >
  using **faster::PbulkFlatMapFunctionP** = void(∗)(U ∗&output, size_t ∗&outputDataSizes, size_t &outputSize, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IPbulkFlatMapFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗&outputDataSizes, size↩
  _t &outputSize, T ∗input, size_t size)
- template<typename T >
  using **faster::bulkReduceFunctionP** = T(∗)(T ∗input, size_t size)

- template<typename T , typename U >
  using **faster::bulkMapPFunctionP** = void(∗)(U ∗output, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IbulkMapPFunctionP** = void(∗)(L ∗outKey, U ∗output, T ∗∗input, size_t ∗inputDataSizes, size←
  _t size)
- template<typename T , typename U >
  using **faster::PbulkMapPFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, T ∗∗input, size_t ∗input←
  DataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IPbulkMapPFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename U >
  using **faster::bulkFlatMapPFunctionP** = void(∗)(U ∗&output, size_t &outputSize, T ∗∗input, size_t ∗input←
  DataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IbulkFlatMapPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename U >
  using **faster::PbulkFlatMapPFunctionP** = void(∗)(U ∗&output, size_t ∗outputDataSizes, size_t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **faster::IPbulkFlatMapPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗outputDataSizes, size←
  _t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T >
  using **faster::PbulkReducePFunctionP** = std::pair< T ∗, size_t >(∗)(T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IbulkMapIFunctionP** = void(∗)(L ∗outKey, U ∗output, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::bulkMapIFunctionP** = void(∗)(U ∗output, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IPbulkMapIFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PbulkMapIFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗input, size←
  _t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IbulkFlatMapIFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::bulkFlatMapIFunctionP** = void(∗)(U ∗&output, size_t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IPbulkFlatMapIFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗&outputDataSizes, size←
  _t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PbulkFlatMapIFunctionP** = void(∗)(U ∗&output, size_t ∗&outputDataSizes, size_t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T >
  using **faster::IbulkReduceIFunctionP** = std::pair< K, T >(∗)(K ∗key, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::IbulkMapIPFunctionP** = void(∗)(L ∗outKey, U ∗output, K ∗inKey, T ∗∗input, size_t ∗inputData←
  Sizes, size_t size)
- template<typename K , typename T , typename U >
  using **faster::bulkMapIPFunctionP** = void(∗)(U ∗output, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)

- template<typename K , typename T , typename L , typename U >
  using **faster::lPbulkMapIPFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PbulkMapIPFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::lbulkFlatMapIPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **faster::bulkFlatMapIPFunctionP** = void(∗)(U ∗&output, size_t &outputSize, K ∗inKey, T ∗∗input, size↩_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **faster::lPbulkFlatMapIPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗outputDataSizes, size↩_t &outputSize, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **faster::PbulkFlatMapIPFunctionP** = void(∗)(U ∗&output, size_t ∗outputDataSizes, size_t &outputSize, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T >
  using **faster::IPbulkReduceIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K ∗key, T ∗∗input, size_t ∗input↩DataSizes, size_t size)
- template<typename K >
  using **faster::bulkUpdateG2FunctionP** = void(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K >
  using **faster::bulkUpdateG3FunctionP** = void(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)
- template<typename K , typename To >
  using **faster::bulkFlatMapG2FunctionP** = std::deque< To >(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K , typename To >
  using **faster::bulkFlatMapG3FunctionP** = std::deque< To >(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)
- template<typename K , typename Ko , typename To >
  using **faster::IbulkFlatMapG2FunctionP** = std::deque< std::pair< Ko, To >>(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K , typename Ko , typename To >
  using **faster::IbulkFlatMapG3FunctionP** = std::deque< std::pair< Ko, To >>(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)

8.6 ByKey Operator Variants

8.6.1 Description

A variant of original operators that groups entries by key to be processed.

ByKey operators use user functions that can access multiple entries of the same corresponding key *U F(K, vector<void*>, size_t)*.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *U F(K, vector<void*>, size_t)* that will be used on each dataset entry

Returns

A pointer to a new dataset

Functions

- template<typename To >
  fdd< To > * **faster::groupedFdd< K >::mapByKey** (mapByKeyG3FunctionP< K, To > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * faster::indexedFdd< K, T >::mapByKey (ImapByKeyIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > * faster::indexedFdd< K, T >::mapByKey (IPmapByKeyIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>*
- template<typename L , typename U >
  fdd< U > * faster::indexedFdd< K, T >::mapByKey (mapByKeyIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > * faster::indexedFdd< K, T >::mapByKey (PmapByKeyIFunctionP< K, T, U > funcP)

  *creates a fdd<U *>*

Typedefs

- template<typename K , typename T >
  using **faster::updateByKeyIFunctionP** = void(*)(K &inKey, std::vector< T * > &input)
- template<typename K , typename T , typename L , typename U >
  using **faster::ImapByKeyIFunctionP** = std::pair< L, U >(*)(const K &inKey, std::vector< T * > &input)
- template<typename K , typename T , typename U >
  using **faster::mapByKeyIFunctionP** = U(*)(const K &inKey, std::vector< T * > &input)

- template<typename K , typename T , typename L , typename U >
  using **faster::IPmapByKeyIFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename U >
  using **faster::PmapByKeyIFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T >
  using **faster::IreduceByKeyIFunctionP** = std::pair< K, T >(∗)(const K &keyA, T ∗a, size_t sizeA, const K &keyB, T ∗b, size_t sizeB)
- template<typename K , typename T , typename L , typename U >
  using **faster::ImapByKeyIPFunctionP** = std::pair< L, U >(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename U >
  using **faster::mapByKeyIPFunctionP** = U(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename L , typename U >
  using **faster::IPmapByKeyIPFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, std::vector< std↩ ::pair< T ∗, size_t >>)
- template<typename K , typename T , typename U >
  using **faster::PmapByKeyIPFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T >
  using **faster::IPreduceByKeyIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K keyA, T ∗∗a, size_t ∗data↩ SizesA, size_t sizeA, K keyB, T ∗∗b, size_t ∗dataSizesB, size_t sizeB)
- template<typename K >
  using **faster::updateByKeyG2FunctionP** = void(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K >
  using **faster::updateByKeyG3FunctionP** = void(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename To >
  using **faster::flatMapByKeyG2FunctionP** = std::deque< To >(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename To >
  using **faster::flatMapByKeyG3FunctionP** = std::deque< To >(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename Ko , typename To >
  using **faster::IflatMapByKeyG2FunctionP** = std::deque< std::pair< Ko, To >>(∗)(const K &key, std↩ ::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename Ko , typename To >
  using **faster::IflatMapByKeyG3FunctionP** = std::deque< std::pair< Ko, To >>(∗)(const K &key, std↩ ::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)

8.7 Memory Model

8.7.1 Description

Automatic memory deallocation.

In order to allow for operator chains like this:

```
...
int result = someFdd -> map(&myMap) -> flatMap(&myFlatMap) -> reduce(&myReduce);
...
```

a automatic memory deallocation model was adopted. If a user apply some operators to a dataset, its distributed memory will be deallocated. In order to use a dataset more than once, the user needs to protect his dataset with the cache() function and discard its content once it is done with the *discard()* function.

Returns

pointer to self

Functions

- void faster::fddCore< T >::discard ()

  deallocates previusly cached fdd
- fdd< T > ∗ faster::fdd< T >::cache ()

  Prevents automatic memory deallocation from hapenning.
- groupedFdd< K > ∗ faster::groupedFdd< K >::cache ()

  Prevents automatic memory deallocation from hapenning.
- void faster::groupedFdd< K >::discard ()

  deallocates previously cached fdd
- void faster::iFddCore< K, T >::discard ()

  deallocates previusly cached FDD
- indexedFdd< K, T > ∗ faster::indexedFdd< K, T >::cache ()

  Prevents automatic memory deallocation from hapenning.

8.7.2 Function Documentation

8.7.2.1 cache() [1/3]

```
template<typename K >
groupedFdd<K>* faster::groupedFdd< K >::cache ( )  [inline]
```

Prevents automatic memory deallocation from hapenning.

Returns

pointer to the cached dataset (self)

8.7.2.2 cache() [2/3]

```
template<class T >
fdd<T>* faster::fdd< T >::cache ( )  [inline]
```

Prevents automatic memory deallocation from hapenning.

Returns

pointer to the cached dataset (self)

8.7.2.3 cache() [3/3]

```
template<typename K , typename T >
indexedFdd<K,T>* faster::indexedFdd< K, T >::cache ( )  [inline]
```

Prevents automatic memory deallocation from hapenning.

Returns

pointer to the cached dataset (self)

8.8 Grouped Datasets Operators

8.8.1 Description

Once the user run a indexedFdd::cogroup a grouped dataset will be created.

The grouped dataset created is a lightweight object that wrapps existing datasets in order to offer more complex operations.

Parameters

K	- Key type of the created dataset
T	- Value type of the source dataset
L	- Key type of the created dataset
U	- Value type of the created dataset
funcP	- A function pointer of a user function *U F(K, vector<void*>, size_t)* that will be used on each dataset entry

Returns

A pointer to a dataset group

Functions

- groupedFdd< K > * **faster::groupedFdd< K >::updateByKey** (updateByKeyG2FunctionP< K > funcP)
- groupedFdd< K > * **faster::groupedFdd< K >::updateByKey** (updateByKeyG3FunctionP< K > funcP)
- groupedFdd< K > * **faster::groupedFdd< K >::bulkUpdate** (bulkUpdateG2FunctionP< K > funcP)
- groupedFdd< K > * **faster::groupedFdd< K >::bulkUpdate** (bulkUpdateG3FunctionP< K > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **faster::groupedFdd< K >::mapByKey** (ImapByKeyG2FunctionP< K, Ko, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **faster::groupedFdd< K >::mapByKey** (ImapByKeyG3FunctionP< K, Ko, To > funcP)
- template<typename To >
  fdd< To > * **faster::groupedFdd< K >::mapByKey** (mapByKeyG2FunctionP< K, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **faster::groupedFdd< K >::flatMapByKey** (IflatMapByKeyG2FunctionP< K, Ko, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **faster::groupedFdd< K >::flatMapByKey** (IflatMapByKeyG3FunctionP< K, Ko, To > funcP)
- template<typename To >
  fdd< To > * **faster::groupedFdd< K >::flatMapByKey** (flatMapByKeyG2FunctionP< K, To > funcP)
- template<typename To >
  fdd< To > * **faster::groupedFdd< K >::flatMapByKey** (flatMapByKeyG3FunctionP< K, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **faster::groupedFdd< K >::bulkFlatMap** (IbulkFlatMapG2FunctionP< K, Ko, To > funcP)

- template< typename Ko , typename To >
  indexedFdd< Ko, To > ∗ **faster::groupedFdd< K >::bulkFlatMap** (IbulkFlatMapG3FunctionP< K, Ko, To > funcP)
- template< typename To >
  fdd< To > ∗ **faster::groupedFdd< K >::bulkFlatMap** (bulkFlatMapG2FunctionP< K, To > funcP)
- template< typename To >
  fdd< To > ∗ **faster::groupedFdd< K >::bulkFlatMap** (bulkFlatMapG3FunctionP< K, To > funcP)

8.9 Shuffle Related Operations

8.9.1 Description

dataset entry exchange between machines.

The groupByKey() and cogroup() operations performa shuffle of information between machines in the cluster. The group locally in each machine every element of a dataset that has the same key. Shufle operations are usually associated with network operations because in order to group elements by key in the cluster, all machines have to send data that does not belong to it to the propper owner.

Note that when a dataset is grouped by key, the key location data is saved to be reused. That way, when calling cogroup multiple times, execution time is saved.

```
...
auto g1 = data.cogroup(data2); <--- this will take longer
auto g2 = data.cogroup(data3); <--- now it will take less time
...
```

Returns

pointer to self

Functions

- template<typename U >
  groupedFdd< K > ∗ faster::iFddCore< K, T >::cogroup (iFddCore< K, U > ∗fdd1)

  Groupes two datasets twogether according with the keys of the first dataset.
- template<typename U , typename V >
  groupedFdd< K > ∗ faster::iFddCore< K, T >::cogroup (iFddCore< K, U > ∗fdd1, iFddCore< K, V > ∗fdd2)

  Groupes tree datasets together according with the keys of the first dataset.
- indexedFdd< K, T > ∗ faster::iFddCore< K, T >::groupByKey ()

  Groups distributed dataset by key.

8.9.2 Function Documentation

8.9.2.1 cogroup() [1/2]

```
template<typename K, typename T>
template<typename U >
groupedFdd<K>* faster::iFddCore< K, T >::cogroup (
             iFddCore< K, U > * fdd1 )  [inline]
```

Groupes two datasets twogether according with the keys of the first dataset.

Template Parameters

U	- Value type of the second dataset

Parameters

fdd1	- second dataset

Returns

pointer to a dataset group

8.9.2.2 cogroup() [2/2]

```
template<typename K, typename T>
template<typename U , typename V >
groupedFdd<K>* faster::iFddCore< K, T >::cogroup (
             iFddCore< K, U > * fdd1,
             iFddCore< K, V > * fdd2 )  [inline]
```

Groupes tree datasets together according with the keys of the first dataset.

Template Parameters

U	- Value type of the second dataset
V	- Value type of the third dataset

Parameters

fdd1	- second dataset
fdd2	- third dataset

Returns

8.9.2.3 groupByKey()

```
template<typename K , typename T >
indexedFdd< K, T > * faster::iFddCore< K, T >::groupByKey ( )
```

Groups distributed dataset by key.

Returns

pointer to itself

Chapter 9

Namespace Documentation

9.1 faster Namespace Reference

9.1.1 Description

libfaster main namespace

Typedefs

- typedef unsigned int fddType

  Dataset type.
- typedef unsigned int fddOpType

  Dataset operation type.
- template< typename T , typename U >
  using **mapFunctionP** = U(∗)(T &input)
- template< typename T , typename L , typename U >
  using **ImapFunctionP** = std::pair< L, U >(∗)(T &input)
- template< typename T , typename U >
  using **PmapFunctionP** = std::pair< U, size_t >(∗)(T &input)
- template< typename T , typename L , typename U >
  using **IPmapFunctionP** = std::tuple< L, U, size_t >(∗)(T &input)
- template< typename T , typename U >
  using **bulkMapFunctionP** = void(∗)(U ∗output, T ∗input, size_t size)
- template< typename T , typename L , typename U >
  using **IbulkMapFunctionP** = void(∗)(L ∗outKey, U ∗output, T ∗input, size_t size)
- template< typename T , typename U >
  using **PbulkMapFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, T ∗input, size_t size)
- template< typename T , typename L , typename U >
  using **IPbulkMapFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, T ∗input, size_t size)
- template< typename T , typename U >
  using **flatMapFunctionP** = std::deque< U >(∗)(T &input)
- template< typename T , typename L , typename U >
  using **IflatMapFunctionP** = std::deque< std::pair< L, U >>(∗)(T &input)
- template< typename T , typename U >
  using **PflatMapFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T &input)
- template< typename T , typename L , typename U >
  using **IPflatMapFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T &input)

- template<typename T , typename U >
  using **bulkFlatMapFunctionP** = void(∗)(U ∗&output, size_t &outputSize, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **IbulkFlatMapFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, T ∗input, size_t size)
- template<typename T , typename U >
  using **PbulkFlatMapFunctionP** = void(∗)(U ∗&output, size_t ∗&outputDataSizes, size_t &outputSize, T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **IPbulkFlatMapFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗&outputDataSizes, size_↩ t &outputSize, T ∗input, size_t size)
- template<typename T >
  using **reduceFunctionP** = T(∗)(T &a, T &b)
- template<typename T >
  using **bulkReduceFunctionP** = T(∗)(T ∗input, size_t size)
- template<typename T , typename U >
  using **mapPFunctionP** = U(∗)(T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **ImapPFunctionP** = std::pair< L, U >(∗)(T ∗input, size_t size)
- template<typename T , typename U >
  using **PmapPFunctionP** = std::pair< U, size_t >(∗)(T ∗input, size_t size)
- template<typename T , typename L , typename U >
  using **IPmapPFunctionP** = std::tuple< L, U, size_t >(∗)(T ∗input, size_t size)
- template<typename T , typename U >
  using **bulkMapPFunctionP** = void(∗)(U ∗output, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **IbulkMapPFunctionP** = void(∗)(L ∗outKey, U ∗output, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename U >
  using **PbulkMapPFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, T ∗∗input, size_t ∗inputData↩ Sizes, size_t size)
- template<typename T , typename L , typename U >
  using **IPbulkMapPFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename U >
  using **flatMapPFunctionP** = std::deque< U >(∗)(T ∗&input, size_t size)
- template<typename T , typename L , typename U >
  using **IflatMapPFunctionP** = std::deque< std::pair< L, U >>(∗)(T ∗&input, size_t size)
- template<typename T , typename U >
  using **PflatMapPFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename T , typename L , typename U >
  using **IPflatMapPFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename T , typename U >
  using **bulkFlatMapPFunctionP** = void(∗)(U ∗&output, size_t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **IbulkFlatMapPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename U >
  using **PbulkFlatMapPFunctionP** = void(∗)(U ∗&output, size_t ∗outputDataSizes, size_t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T , typename L , typename U >
  using **IPbulkFlatMapPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗outputDataSizes, size_↩ t &outputSize, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename T >
  using **PreducePFunctionP** = std::pair< T ∗, size_t >(∗)(T ∗a, size_t sizeA, T ∗b, size_t sizeB)
- template<typename T >
  using **PbulkReducePFunctionP** = std::pair< T ∗, size_t >(∗)(T ∗∗input, size_t ∗inputDataSizes, size_t size)

- template<typename K , typename T >
  using **updateIFunctionP** = void(∗)(K &inKey, T &input)
- template<typename K , typename T >
  using **updateByKeyIFunctionP** = void(∗)(K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename L , typename U >
  using **ImapIFunctionP** = std::pair< L, U >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename U >
  using **mapIFunctionP** = U(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **IPmapIFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename U >
  using **PmapIFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **ImapByKeyIFunctionP** = std::pair< L, U >(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename U >
  using **mapByKeyIFunctionP** = U(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename L , typename U >
  using **IPmapByKeyIFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename U >
  using **PmapByKeyIFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, std::vector< T ∗ > &input)
- template<typename K , typename T , typename L , typename U >
  using **IbulkMapIFunctionP** = void(∗)(L ∗outKey, U ∗output, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **bulkMapIFunctionP** = void(∗)(U ∗output, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPbulkMapIFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **PbulkMapIFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IflatMapIFunctionP** = std::deque< std::pair< L, U >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename U >
  using **flatMapIFunctionP** = std::deque< U >(∗)(K inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **IPflatMapIFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename U >
  using **PflatMapIFunctionP** = std::deque< std::pair< U, size_t >>(∗)(K inKey, T &input)
- template<typename K , typename T , typename L , typename U >
  using **IbulkFlatMapIFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **bulkFlatMapIFunctionP** = void(∗)(U ∗&output, size_t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPbulkFlatMapIFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗&outputDataSizes, size_↩
  t &outputSize, K ∗inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **PbulkFlatMapIFunctionP** = void(∗)(U ∗&output, size_t ∗&outputDataSizes, size_t &outputSize, K ∗in↩
  Key, T ∗input, size_t size)
- template<typename K , typename T >
  using **IreduceIFunctionP** = std::pair< K, T >(∗)(const K &keyA, T &a, const K &keyB, T &b)
- template<typename K , typename T >
  using **IreduceByKeyIFunctionP** = std::pair< K, T >(∗)(const K &keyA, T ∗a, size_t sizeA, const K &keyB, T ∗b, size_t sizeB)
- template<typename K , typename T >
  using **IbulkReduceIFunctionP** = std::pair< K, T >(∗)(K ∗key, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **ImapIPFunctionP** = std::pair< L, U >(∗)(K inKey, T ∗input, size_t size)

- template<typename K , typename T , typename U >
  using **mapIPFunctionP** = U(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPmapIPFunctionP** = std::tuple< L, U, size_t >(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename U >
  using **PmapIPFunctionP** = std::pair< U, size_t >(∗)(K inKey, T ∗input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **ImapByKeyIPFunctionP** = std::pair< L, U >(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename U >
  using **mapByKeyIPFunctionP** = U(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename L , typename U >
  using **IPmapByKeyIPFunctionP** = std::tuple< L, U, size_t >(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename U >
  using **PmapByKeyIPFunctionP** = std::pair< U, size_t >(∗)(const K &inKey, std::vector< std::pair< T ∗, size_t >>)
- template<typename K , typename T , typename L , typename U >
  using **IbulkMapIPFunctionP** = void(∗)(L ∗outKey, U ∗output, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **bulkMapIPFunctionP** = void(∗)(U ∗output, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPbulkMapIPFunctionP** = void(∗)(L ∗outKey, U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **PbulkMapIPFunctionP** = void(∗)(U ∗output, size_t ∗outputDataSizes, K ∗inKey, T ∗∗input, size_←
  t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IflatMapIPFunctionP** = std::deque< std::pair< L, U >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename U >
  using **flatMapIPFunctionP** = std::deque< U >(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPflatMapIPFunctionP** = std::deque< std::tuple< L, U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename U >
  using **PflatMapIPFunctionP** = std::deque< std::pair< U, size_t >>(∗)(T ∗&input, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IbulkFlatMapIPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t &outputSize, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **bulkFlatMapIPFunctionP** = void(∗)(U ∗&output, size_t &outputSize, K ∗inKey, T ∗∗input, size_←
  t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename L , typename U >
  using **IPbulkFlatMapIPFunctionP** = void(∗)(L ∗&outKey, U ∗&output, size_t ∗outputDataSizes, size_←
  t &outputSize, K ∗inKey, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T , typename U >
  using **PbulkFlatMapIPFunctionP** = void(∗)(U ∗&output, size_t ∗outputDataSizes, size_t &outputSize, K ∗in←
  Key, T ∗∗input, size_t ∗inputDataSizes, size_t size)
- template<typename K , typename T >
  using **IPreduceIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K keyA, T ∗a, size_t sizeA, K keyB, T ∗b, size_t sizeB)
- template<typename K , typename T >
  using **IPreduceByKeyIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K keyA, T ∗∗a, size_t ∗dataSizesA, size_t sizeA, K keyB, T ∗∗b, size_t ∗dataSizesB, size_t sizeB)
- template<typename K , typename T >
  using **IPbulkReduceIPFunctionP** = std::tuple< K, T ∗, size_t >(∗)(K ∗key, T ∗∗input, size_t ∗inputDataSizes, size_t size)

- template<typename K >
  using **updateByKeyG2FunctionP** = void(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K >
  using **updateByKeyG3FunctionP** = void(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K >
  using **bulkUpdateG2FunctionP** = void(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K >
  using **bulkUpdateG3FunctionP** = void(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)
- template<typename K , typename To >
  using **mapByKeyG2FunctionP** = To(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename To >
  using **mapByKeyG3FunctionP** = To(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename Ko , typename To >
  using **ImapByKeyG2FunctionP** = std::pair< Ko, To >(∗)(const K &key, std::vector< void ∗ > &a, std↩ ::vector< void ∗ > &b)
- template<typename K , typename Ko , typename To >
  using **ImapByKeyG3FunctionP** = std::pair< Ko, To >(∗)(const K &key, std::vector< void ∗ > &a, std↩ ::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename To >
  using **flatMapByKeyG2FunctionP** = std::deque< To >(∗)(const K &key, std::vector< void ∗ > &a, std↩ ::vector< void ∗ > &b)
- template<typename K , typename To >
  using **flatMapByKeyG3FunctionP** = std::deque< To >(∗)(const K &key, std::vector< void ∗ > &a, std↩ ::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename Ko , typename To >
  using **IflatMapByKeyG2FunctionP** = std::deque< std::pair< Ko, To >>(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b)
- template<typename K , typename Ko , typename To >
  using **IflatMapByKeyG3FunctionP** = std::deque< std::pair< Ko, To >>(∗)(const K &key, std::vector< void ∗ > &a, std::vector< void ∗ > &b, std::vector< void ∗ > &c)
- template<typename K , typename To >
  using **bulkFlatMapG2FunctionP** = std::deque< To >(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K , typename To >
  using **bulkFlatMapG3FunctionP** = std::deque< To >(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)
- template<typename K , typename Ko , typename To >
  using **IbulkFlatMapG2FunctionP** = std::deque< std::pair< Ko, To >>(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb)
- template<typename K , typename Ko , typename To >
  using **IbulkFlatMapG3FunctionP** = std::deque< std::pair< Ko, To >>(∗)(K ∗keyA, void ∗a, size_t na, K ∗keyB, void ∗b, size_t nb, K ∗keyC, void ∗c, size_t nc)

Partition function definitions

- template<typename T >
  using **onlineFullPartFuncP** = int(∗)(T &input)
- template<typename K , typename T >
  using **IonlineFullPartFuncP** = int(∗)(K &key, T &input)

Enumerations

- enum **dFuncName** : char {
NewWorkerDL = 0x01, **NewWorkerSDL** = 0x02, **DiscardWorkerDL** = 0x03, **GetTypeDL** = 0x04,
GetKeyTypeDL = 0x05, **SetDataDL** = 0x06, **SetDataRawDL** = 0x07, **GetLineSizesDL** = 0x08,
GetFddItemDL = 0x09, **GetKeysDL** = 0x0a, **GetDataDL** = 0x0b, **GetSizeDL** = 0x0c,
ItemSizeDL = 0x0d, **BaseSizeDL** = 0x0e, **SetSizeDL** = 0x0f, **DeleteItemDL** = 0x10,
ShrinkDL = 0x11, **InsertDL** = 0x12, **InsertListDL** = 0x13, **PreapplyDL** = 0x14,
CollectDL = 0x15, **GroupByKeyDL** = 0x16, **CountByKeyDL** = 0x17, **ExchangeDataByKeyDL** = 0x18,
GetKeyLocationDL = 0x19, **GetUKeysDL** = 0x1a, **SetUKeysDL** = 0x1b, **GetKeyMapDL** = 0x1c,
SetKeyMapDL = 0x1d, **WriteToFileDL** = 0x1e }
- enum **commMode** { **Local**, **Mesos** }
- enum **msgTag** : int {
MSG_TASK, **MSG_CREATEFDD**, **MSG_CREATEIFDD**, **MSG_CREATEGFDD**,
MSG_DISCARDFDD, **MSG_FDDSETDATAID**, **MSG_FDDSETDATA**, **MSG_FDDSET2DDATAID**,
MSG_FDDSET2DDATASIZES, **MSG_FDDSET2DDATA**, **MSG_READFDDFILE**, **MSG_WRITEFDDFILE**,
MSG_FILENAME, **MSG_COLLECT**, **MSG_FDDDATAID**, **MSG_FDDDATA**,
MSG_TASKRESULT, **MSG_FDDINFO**, **MSG_FDDSETIDATAID**, **MSG_FDDSETIDATA**,
MSG_FDDSETIKEYS, **MSG_FDDSET2DIDATAID**, **MSG_FDDSET2DIDATASIZES**, **MSG_FDDSET2DID↩ATA**,
MSG_FDDSET2DIKEYS, **MSG_KEYOWNERSHIPSUGEST**, **MSG_MYKEYOWNERSHIP**, **MSG_MYKEY↩COUNT**,
MSG_IFDDDATAID, **MSG_IFDDDATAKEYS**, **MSG_IFDDDATA**, **MSG_COLLECTDATA**,
MSG_KEYMAP, **MSG_DISTKEYMAP**, **MSG_GROUPBYKEYDATA**, **MSG_FINISH** }
- enum **fileMode** : int { **R** = O_RDONLY, **W** = O_WRONLY, **CR** = O_RDONLY | O_CREAT, **CW** = O_WRONLY | O_CREAT }

Functions

- procstat **getProcStat** ()
- fddType **decodeType** (size_t typeCode)
- const std::string **decodeOptype** (fddOpType op)
- const std::string **decodeOptypeAb** (fddOpType op)
- template<typename T >
double **mean** (std::vector< T > v)
- template<typename T >
double **max** (std::vector< T > v)
- template<typename T >
double **sum** (std::vector< T > v)
- template<typename T >
double **stdDev** (std::vector< T > v, double mean)
- workerFddBase ∗ **newWorkerSDL** (unsigned long int id, fddType type, size_t size)
- void **discardWorkerDL** (workerFddBase ∗fdd)
- fddType **getTypeDL** (workerFddBase ∗fdd)
- fddType **getKeyTypeDL** (workerFddBase ∗fdd)
- void **setDataDL** (workerFddBase ∗fdd, void ∗keys, void ∗data, size_t ∗lineSizes, size_t size)
- void **setDataRawDL** (workerFddBase ∗fdd, void ∗keys, void ∗data, size_t ∗lineSizes, size_t size)
- size_t ∗ **getLineSizesDL** (workerFddBase ∗fdd)
- void ∗ **getFddItemDL** (workerFddBase ∗fdd, size_t address)
- void ∗ **getKeysDL** (workerFddBase ∗fdd)
- void ∗ **getDataDL** (workerFddBase ∗fdd)
- size_t **getSizeDL** (workerFddBase ∗fdd)
- size_t **itemSizeDL** (workerFddBase ∗fdd)
- size_t **baseSizeDL** (workerFddBase ∗fdd)
- void **setSizeDL** (workerFddBase ∗fdd, size_t s)

- void **deleteItemDL** (workerFddBase ∗fdd, void ∗item)
- void **shrinkDL** (workerFddBase ∗fdd)
- void **insertDL** (workerFddBase ∗fdd, void ∗k, void ∗v, size_t s)
- void **insertListDL** (workerFddBase ∗fdd, void ∗v)
- void **preapplyDL** (workerFddBase ∗fdd, unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm)
- void **collectDL** (workerFddBase ∗fdd, fastComm ∗comm)
- void **exchangeDataByKeyDL** (workerFddBase ∗fdd, fastComm ∗comm)
- void ∗ **getKeyLocationsDL** (workerFddBase ∗fdd)
- void ∗ **getUKeysDL** (workerFddBase ∗fdd)
- void **setUKeysDL** (workerFddBase ∗fdd, void ∗uk)
- void ∗ **getKeyMapDL** (workerFddBase ∗fdd)
- void **setKeyMapDL** (workerFddBase ∗fdd, void ∗km)
- void **writeToFileDL** (workerFddBase ∗fdd, void ∗path, size_t procId, void ∗sufix)

Variables

- const int **BUFFER_INITIAL_SIZE** = 512∗1024

Classes

- class _workerFdd
- class _workerFdd< T ∗ >
- class _workerIFdd
- class _workerIFdd< K, T ∗ >
- class fastComm
- class fastCommBuffer
- class fastContext

    Framework context class.
- class fastScheduler
- class fastSettings

    Context Configuration Class.
- class fastTask
- class fdd

    Fast Distributted Dataset(FDD) is like a cluster distributted Array. This class is the user side implementation.
- class fdd< T ∗ >
- class fddBase
- class fddCore

    core class that implements simple operations.
- class fddStorage
- class fddStorage< T ∗ >
- class fddStorageBase
- class fddStorageCore
- class groupedFdd
- class hasher
- class hasher< double >
- class hasher< float >
- class hasher< std::string >
- class hdfsEngine
- class hdfsFile
- class iFddCore
- class indexedFdd

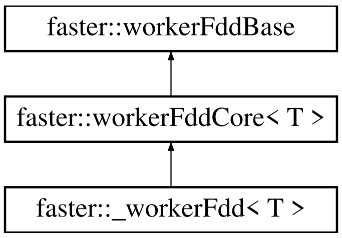
- class indexedFdd< K, T * >
- class indexedFddStorage
- class indexedFddStorage< K, T * >
- class indexedFddStorageCore
- class procstat
- class worker
- class workerFdd
- class workerFddBase
- class workerFddCore
- class workerFddGroup
- class workerIFdd
- class workerIFddCore

Chapter 10

Class Documentation

10.1 faster::_workerFdd< T > Class Template Reference

Inheritance diagram for faster::_workerFdd< T >:



10.1.1 Description

template<class T>
class faster::_workerFdd< T >

Public Member Functions

- **_workerFdd** (unsigned int ident, fddType t)
- **_workerFdd** (unsigned int ident, fddType t, size_t size)
- void **setData** (T ∗data, size_t size)
- void **setData** (void ∗d UNUSED, size_t size UNUSED)
- void **setData** (void ∗d UNUSED, size_t ∗lineSizes UNUSED, size_t size UNUSED)
- void **setData** (void ∗k UNUSED, void ∗d UNUSED, size_t ∗lineSizes UNUSED, size_t size UNUSED)
- void **setDataRaw** (void ∗data, size_t size) override
- void **setDataRaw** (void ∗data UNUSED, size_t ∗listSizes UNUSED, size_t size UNUSED) override
- size_t ∗ **getLineSizes** ()
- void **insert** (void ∗k, void ∗in, size_t s)
- void **insertl** (void ∗in)
- void **insert** (T &in)
- void **insert** (T ∗in UNUSED, size_t s UNUSED)
- void **insert** (std::deque< T > &in)
- void **insert** (std::deque< std::pair< T ∗, size_t >> &in UNUSED)

- void **apply** (void ∗func, fddOpType op, workerFddBase ∗dest, fastCommBuffer &buffer)
- void **collect** (fastComm ∗comm) override
- template<typename U >
  void **map** (workerFddBase ∗dest, mapPFunctionP< T, U > mapFunc)
- template<typename U >
  void **map** (workerFddBase ∗dest, PmapPFunctionP< T, U > mapFunc)
- template<typename L , typename U >
  void **map** (workerFddBase ∗dest, ImapPFunctionP< T, L, U > mapFunc)
- template<typename L , typename U >
  void **map** (workerFddBase ∗dest, IPmapPFunctionP< T, L, U > mapFunc)
- template<typename U >
  void **bulkMap** (workerFddBase ∗dest, bulkMapPFunctionP< T, U > bulkMapFunc)
- template<typename U >
  void **bulkMap** (workerFddBase ∗dest, PbulkMapPFunctionP< T, U > bulkMapFunc)
- template<typename L , typename U >
  void **bulkMap** (workerFddBase ∗dest, IbulkMapPFunctionP< T, L, U > bulkMapFunc)
- template<typename L , typename U >
  void **bulkMap** (workerFddBase ∗dest, IPbulkMapPFunctionP< T, L, U > bulkMapFunc)
- template<typename U >
  void **flatMap** (workerFddBase ∗dest, flatMapPFunctionP< T, U > flatMapFunc)
- template<typename U >
  void **flatMap** (workerFddBase ∗dest, PflatMapPFunctionP< T, U > flatMapFunc)
- template<typename L , typename U >
  void **flatMap** (workerFddBase ∗dest, IflatMapPFunctionP< T, L, U > flatMapFunc)
- template<typename L , typename U >
  void **flatMap** (workerFddBase ∗dest, IPflatMapPFunctionP< T, L, U > flatMapFunc)
- template<typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, bulkFlatMapPFunctionP< T, U > bulkFlatMapFunc)
- template<typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, PbulkFlatMapPFunctionP< T, U > bulkFlatMapFunc)
- template<typename L , typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, IbulkFlatMapPFunctionP< T, L, U > bulkFlatMapFunc)
- template<typename L , typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, IPbulkFlatMapPFunctionP< T, L, U > bulkFlatMapFunc)

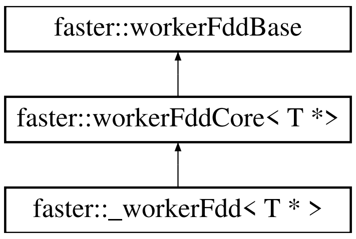
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/_workerFdd.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerPFdd.cpp

10.2 faster::_workerFdd< T ∗ > Class Template Reference

Inheritance diagram for faster::_workerFdd< T ∗ >:



10.2.1 Description

template< class T >
class faster::_workerFdd< T * >

Public Member Functions

- **_workerFdd** (unsigned int ident, fddType t)
- **_workerFdd** (unsigned int ident, fddType t, size_t size)
- void **setData** (T **data, size_t *lineSizes, size_t size)
- void **setData** (void *d UNUSED, size_t size UNUSED)
- void **setData** (void *data UNUSED, size_t *lineSizes UNUSED, size_t size UNUSED)
- void **setData** (void *k UNUSED, void *d UNUSED, size_t *lineSizes UNUSED, size_t size UNUSED)
- void **setDataRaw** (void *data UNUSED, size_t size UNUSED) override
- void **setDataRaw** (void *data, size_t *lineSizes, size_t size) override
- size_t * **getLineSizes** ()
- void **insert** (void *k, void *in, size_t s)
- void **insertl** (void *in)
- void **insert** (T &in)
- void **insert** (T *&in, size_t s)
- void **insert** (std::deque< T > &in)
- void **insert** (std::deque< std::pair< T *, size_t > > &in)
- void **apply** (void *func, fddOpType op, workerFddBase *dest, fastCommBuffer &buffer)
- void **collect** (fastComm *comm) override

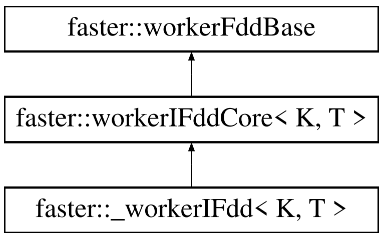
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h

10.3 faster::_workerIFdd< K, T > Class Template Reference

Inheritance diagram for faster::_workerIFdd< K, T >:



10.3.1 Description

template<class K, class T>
class faster::_workerIFdd< K, T >

Public Member Functions

- **_workerIFdd** (unsigned int ident, fddType kt, fddType t)
- **_workerIFdd** (unsigned int ident, fddType kt, fddType t, size_t size)
- void **setData** (K ∗keys, T ∗data, size_t size)
- void **setData** (void ∗keys, void ∗data, size_t size)
- void **setData** (void ∗keys, void ∗data, size_t ∗lineSizes UNUSED, size_t size)
- void **setDataRaw** (void ∗keys, void ∗data, size_t size) override
- void **setDataRaw** (void ∗keys UNUSED, void ∗data UNUSED, size_t ∗lineSizes UNUSED, size_t size UN↩ USED) override
- size_t ∗ **getLineSizes** ()
- void **insert** (void ∗k, void ∗in, size_t s)
- void **insertl** (void ∗in)
- void **insert** (K &key, T &in)
- void **insert** (std::deque< std::pair< K, T > > &in)
- void **apply** (void ∗func, fddOpType op, workerFddBase ∗dest, fastCommBuffer &buffer)
- void **collect** (fastComm ∗comm) override
- template<typename L , typename U >
  void **map** (workerFddBase ∗dest, ImapIPFunctionP< K, T, L, U > mapFunc)
- template<typename L , typename U >
  void **map** (workerFddBase ∗dest, IPmapIPFunctionP< K, T, L, U > mapFunc)
- template<typename U >
  void **map** (workerFddBase ∗dest, mapIPFunctionP< K, T, U > mapFunc)
- template<typename U >
  void **map** (workerFddBase ∗dest, PmapIPFunctionP< K, T, U > mapFunc)
- template<typename L , typename U >
  void **bulkMap** (workerFddBase ∗dest, IbulkMapIPFunctionP< K, T, L, U > bulkMapFunc)
- template<typename L , typename U >
  void **bulkMap** (workerFddBase ∗dest, IPbulkMapIPFunctionP< K, T, L, U > bulkMapFunc)
- template<typename U >
  void **bulkMap** (workerFddBase ∗dest, bulkMapIPFunctionP< K, T, U > bulkMapFunc)
- template<typename U >
  void **bulkMap** (workerFddBase ∗dest, PbulkMapIPFunctionP< K, T, U > bulkMapFunc)
- template<typename L , typename U >
  void **flatMap** (workerFddBase ∗dest, IflatMapIPFunctionP< K, T, L, U > flatMapFunc)
- template<typename L , typename U >
  void **flatMap** (workerFddBase ∗dest, IPflatMapIPFunctionP< K, T, L, U > flatMapFunc)
- template<typename U >
  void **flatMap** (workerFddBase ∗dest, flatMapIPFunctionP< K, T, U > flatMapFunc)
- template<typename U >
  void **flatMap** (workerFddBase ∗dest, PflatMapIPFunctionP< K, T, U > flatMapFunc)
- template<typename L , typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, IbulkFlatMapIPFunctionP< K, T, L, U > bulkFlatMapFunc)
- template<typename L , typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, IPbulkFlatMapIPFunctionP< K, T, L, U > bulkFlatMapFunc)
- template<typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, bulkFlatMapIPFunctionP< K, T, U > bulkFlatMapFunc)
- template<typename U >
  void **bulkFlatMap** (workerFddBase ∗dest, PbulkFlatMapIPFunctionP< K, T, U > bulkFlatMapFunc)

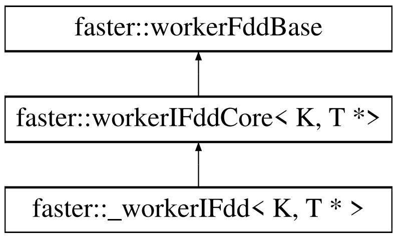
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerIFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerIFdd.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerIFddDependent.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerIPFdd.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerIPFddDependent.cpp

10.4 faster::_workerIFdd< K, T * > Class Template Reference

Inheritance diagram for faster::_workerIFdd< K, T * >:


faster::workerFddBase
faster::workerIFddCore< K, T *>
faster::_workerIFdd< K, T * >

10.4.1 Description

template<class K, class T>
class faster::_workerIFdd< K, T * >

Public Member Functions

- **_workerIFdd** (unsigned int ident, fddType kt, fddType t)
- **_workerIFdd** (unsigned int ident, fddType kt, fddType t, size_t size)
- void **setData** (K *keys, T **data, size_t *lineSizes, size_t size)
- void **setData** (void *keys UNUSED, void *data UNUSED, size_t size UNUSED)
- void **setData** (void *keys, void *data, size_t *lineSizes, size_t size)
- void **setDataRaw** (void *keys UNUSED, void *data UNUSED, size_t size UNUSED) override
- void **setDataRaw** (void *keys, void *data, size_t *lineSizes, size_t size) override
- size_t * **getLineSizes** ()
- void **insert** (void *k, void *in, size_t s)
- void **insertl** (void *in)
- void **insert** (K &key, T *&in, size_t s)
- void **insert** (std::deque< std::tuple< K, T *, size_t > > &in)
- void **apply** (void *func, fddOpType op, workerFddBase *dest, fastCommBuffer &buffer)
- void **collect** (fastComm *comm) override

Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerIFdd.h

10.5 faster::fastComm Class Reference

10.5.1 Description

Public Member Functions

- **fastComm** (int &argc, char ∗∗argv)
- int **getProcId** ()
- int **getNumProcs** ()
- fastCommBuffer & **getResultBuffer** ()
- fastCommBuffer ∗ **getSendBuffers** ()
- bool **isDriver** ()
- void **probeMsgs** (int &tag, int &src)
- void **waitForReq** (int numReqs)
- void **joinAll** ()
- void **joinSlaves** ()
- template<typename T >
  size_t **getSize** (T ∗data UNUSED, size_t ∗ds UNUSED, size_t s)
- template<typename T >
  size_t **getSize** (std::vector< T > ∗data, size_t ∗ds UNUSED, size_t s)
- template<typename T >
  size_t **getSize** (T ∗∗data UNUSED, size_t ∗ds, size_t s)
- size_t **getSize** (std::string ∗data, size_t ∗ds UNUSED, size_t s)
- void **sendTask** (fastTask &task)
- void **recvTask** (fastTask &task)
- void **sendTaskResult** ()
- void ∗ **recvTaskResult** (unsigned long int &tid, unsigned long int &sid, size_t &size, size_t &time, procstat &stat)
- void **sendCreateFDD** (unsigned long int id, fddType type, size_t size, int dest)
- void **recvCreateFDD** (unsigned long int &id, fddType &type, size_t &size)
- void **sendCreateIFDD** (unsigned long int id, fddType kType, fddType tType, size_t size, int dest)
- void **recvCreateIFDD** (unsigned long int &id, fddType &kType, fddType &tType, size_t &size)
- void **sendCreateFDDGroup** (unsigned long int id, fddType keyType, std::vector< unsigned long int > &members)
- void **recvCreateFDDGroup** (unsigned long int &id, fddType &keyType, std::vector< unsigned long int > &members)
- void **sendDiscardFDD** (unsigned long int id)
- void **recvDiscardFDD** (unsigned long int &id)
- template<typename T >
  void **sendFDDSetData** (unsigned long int id, int dest, T ∗data, size_t size)
- template<typename T >
  void **sendFDDSetData** (unsigned long int id, int dest, T ∗∗data, size_t ∗lineSizes, size_t size)
- template<typename K , typename T >
  void **sendFDDSetIData** (unsigned long int id, int dest, K ∗keys, T ∗data, size_t size)
- template<typename K , typename T >
  void **sendFDDSetIData** (unsigned long int id, int dest, K ∗keys, T ∗∗data, size_t ∗lineSizes, size_t size)
- void **recvFDDSetData** (unsigned long int &id, void ∗&data, size_t &size)
- void **recvFDDSetData** (unsigned long int &id, void ∗&data, size_t ∗&lineSizes, size_t &size)
- template<typename K , typename T >
  void **recvFDDSetIData** (unsigned long int &id, K ∗&keys, T ∗&data, size_t &size)
- template<typename K , typename T >
  void **recvFDDSetIData** (unsigned long int &id, K ∗&keys, T ∗&data, size_t ∗&lineSizes, size_t &size)
- template<typename T >
  void **sendFDDData** (unsigned long int id, int dest, T ∗data, size_t size)

- template<typename K , typename T >
  void **sendIFDDData** (unsigned long int id, int dest, K ∗keys, T ∗data, size_t size)
- void **recvFDDData** (unsigned long int &id, void ∗data, size_t &size)
- void **recvIFDDData** (unsigned long int &id, void ∗keys, void ∗data, size_t &size)
- template<typename T >
  void **sendFDDDataCollect** (unsigned long int id, T ∗data, size_t size)
- template<typename T >
  void **sendFDDDataCollect** (unsigned long int id, T ∗∗data, size_t ∗dataSizes, size_t size)
- template<typename K , typename T >
  void **sendFDDDataCollect** (unsigned long int id, K ∗keys, T ∗data, size_t size)
- template<typename K , typename T >
  void **sendFDDDataCollect** (unsigned long int id, K ∗keys, T ∗∗data, size_t ∗dataSizes, size_t size)
- template<typename T >
  void **decodeCollect** (T &item)
- template<typename T >
  void **decodeCollect** (std::pair< T ∗, size_t > &item)
- template<typename K , typename T >
  void **decodeCollect** (std::pair< K, T > &item)
- template<typename K , typename T >
  void **decodeCollect** (std::tuple< K, T ∗, size_t > &item)
- template<typename T >
  void **recvFDDDataCollect** (std::vector< T > &ret)
- void **sendReadFDDFile** (unsigned long int id, std::string filename, size_t size, size_t offset, int dest)
- void **recvReadFDDFile** (unsigned long int &id, std::string &filename, size_t &size, size_t &offset)
- void **sendWriteFDDFile** (unsigned long int id, std::string &path, std::string &sufix)
- void **recvWriteFDDFile** (unsigned long int &id, std::string &path, std::string &sufix)
- void **sendFDDInfo** (size_t size)
- void **recvFDDInfo** (size_t &size, int &src)
- void **sendFileName** (std::string path)
- void **recvFileName** (std::string &filename)
- void **sendCollect** (unsigned long int id)
- void **recvCollect** (unsigned long int &id)
- void **sendFinish** ()
- void **recvFinish** ()
- void **bcastBuffer** (int src, int i)
- template<typename K >
  void **sendKeyMap** (unsigned long tid, std::unordered_map< K, int > &keyMap)
- template<typename K >
  void **recvKeyMap** (unsigned long tid, std::unordered_map< K, int > &keyMap)
- template<typename K >
  void **distributeKeyMap** (std::unordered_map< K, int > &localKeyMap, std::unordered_map< K, int > &keyMap)
- template<typename K >
  void **sendCogroupData** (unsigned long tid, std::unordered_map< K, int > &keyMap, std::vector< bool > &flags)
- template<typename K >
  void **recvCogroupData** (unsigned long tid, std::unordered_map< K, int > &keyMap, std::vector< bool > &flags)
- bool **isSendBufferFree** (int i)
- void **sendGroupByKeyData** (int i)
- void ∗ **recvGroupByKeyData** (int &size)
- template<typename T >
  void **sendDataUltraPlus** (int dest, T ∗data, size_t ∗lineSizes UNUSED, size_t size, int tag, fastCommBuffer &b UNUSED, MPI_Request ∗request)
- template<typename T >
  void **sendDataUltraPlus** (int dest, std::vector< T > ∗data, size_t ∗lineSizes UNUSED, size_t size, int tag, fastCommBuffer &b UNUSED, MPI_Request ∗request)

Public Attributes

- const size_t **maxMsgSize** = 15000

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastComm.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fastComm.cpp

10.6 faster::fastCommBuffer Class Reference

10.6.1 Description

Public Member Functions

- **fastCommBuffer** (size_t s)
- void **setBuffer** (void ∗buffer, size_t s)
- void **reset** ()
- char ∗ **data** ()
- char ∗ **pos** ()
- char ∗ **pos** (size_t pos)
- size_t **size** ()
- size_t **free** ()
- void **advance** (size_t pos)
- void **grow** (size_t s)
- void **print** ()
- template<typename T >
  void **write** (T &v, size_t s)
- template<typename T >
  void **writePos** (const T &v, size_t s, size_t pos)
- template<typename T >
  void **writePos** (const T &v, size_t pos)
- template<typename T >
  void **writeSafe** (T ∗v, size_t s)
- template<typename T >
  void **write** (T ∗v, size_t s)
- template<typename T >
  void **write** (T v)
- void **write** (std::string i)
- void **write** (std::vector< std::string > v)
- template<typename T >
  void **write** (std::vector< T > v)
- template<typename K , typename T >
  void **write** (std::pair< K, T > p)
- template<typename K , typename T >
  void **write** (std::tuple< K, T, size_t > t)
- void **write** (procstat &s)
- void **writePos** (procstat &s, size_t pos)
- void **read** (procstat &s)
- void **advance** (procstat &s)
- template<typename T >
  void **read** (T &v, size_t s)

- template<typename T >
  void **read** (T ∗v, size_t s)
- template<typename T >
  void **read** (T &v)
- template<typename T >
  void **readVec** (std::vector< T > &v, size_t s)
- void **read** (std::vector< std::string > &v)
- void **readString** (std::string &v, size_t s)
- template<typename T >
  void **read** (std::vector< T > &v)
- void **read** (std::string &s)
- template<typename K , typename T >
  void **read** (std::pair< K, T > &p)
- template<typename K , typename T >
  void **read** (std::tuple< K, T, size_t > &t)
- template<typename T >
  fastCommBuffer & **operator**<< (T v)
- template<typename T >
  fastCommBuffer & **operator**>> (T &v)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastCommBuffer.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fastCommBuffer.cpp

10.7 faster::fastContext Class Reference

```
#include <fastContext.h>
```

10.7.1 Description

Framework context class.

The context manages communication, scheduler and start Workers. A context is needed to create datasets!

Public Member Functions

- fastContext (int argc=0, char ∗∗argv=NULL)

  fastContext default constructor
- fastContext (const fastSettings &s, int argc, char ∗∗argv)

  fastContext constructor with custom settings
- ∼fastContext ()

  fastContext destructor
- void startWorkers ()

  Start worker machines computation.
- bool isDriver ()

  Checks for the driver process.
- int numProcs ()

  Return the number of processes running.

- void calibrate ()

  Performs a microbenchmark to do dynamic load balancing (UNUSED)

Function and global variables registration

- void registerFunction (void ∗funcP)

  Register a user custom function in the context.
- void registerFunction (void ∗funcP, const std::string name)

  Register a user custom function in the context.
- template< class T >
  void registerGlobal (T ∗varP)

  Gegisters a primitive global varible to be used inside used defined functions in distributted environment.
- template< class T >
  void registerGlobal (T ∗∗varP, size_t s)

  Gegisters a global array to be used inside used defined functions in distributted environment.
- template< class T >
  void registerGlobal (std::vector< T > ∗varP)

  Gegisters a global Vector to be used inside used defined functions in distributted environment.

Online file reading and parsing

- template< typename T >
  fdd< T > ∗ onlineFullPartRead (std::string path, onlineFullPartFuncP< T > funcP)

  Reads a file with online parsing and partition (NOT IMPLEMENTED)
- template< typename K , typename T >
  indexedFdd< K, T > ∗ **onlineFullPartRead** (std::string path, IonlineFullPartFuncP< K, T > funcP)
- template< typename K , typename T >
  indexedFdd< K, T > ∗ **onlinePartRead** (std::string path, IonlineFullPartFuncP< K, T > funcP)
- template< typename T >
  fdd< T > ∗ onlineRead (std::string path, onlineFullPartFuncP< T > funcP)

  Reads a file with online parsing and mapping (?)
- template< typename K , typename T >
  indexedFdd< K, T > ∗ **onlineRead** (std::string path, IonlineFullPartFuncP< K, T > funcP)

Task execution profiling

- void printInfo ()

  Prints task execution information for all tasks executed by the user.
- void printHeader ()

  Prints a header for task execution information.
- void updateInfo ()

  Prints information from tesk ran since last faster::fastContext::updateInfo() called.

10.7.2 Constructors and Destructors

10.7.2.1 fastContext()

```
faster::fastContext::fastContext (
            int argc = 0,
            char ** argv = NULL )
```

fastContext default constructor

Parameters

argc	- number of arguments from main
argv	- arguments from main

10.7.3 Member Function Documentation

10.7.3.1 isDriver()

```
bool faster::fastContext::isDriver ( )
```

Checks for the driver process.

Returns

- true if the process is the driver process

10.7.3.2 numProcs()

```
int faster::fastContext::numProcs ( )  [inline]
```

Return the number of processes running.

Returns

number of active processes

10.7.3.3 onlineFullPartRead()

```
template<typename T >
fdd<T>* faster::fastContext::onlineFullPartRead (
            std::string path,
            onlineFullPartFuncP< T > funcP )
```

Reads a file with online parsing and partition (NOT IMPLEMENTED)

Template Parameters

T	- Dataset type

Parameters

path	- Input file path
funcP	- partition function pointer of types ::faster::onlineFullPartFuncP or ::faster::IonlineFullPartFuncP

Returns

- a dataset of ::faster::fdd<t> type and faster::indexedFdd<K,T>

10.7.3.4 onlineRead()

```
template<typename T >
fdd<T>* faster::fastContext::onlineRead (
             std::string path,
             onlineFullPartFuncP< T > funcP )
```

Reads a file with online parsing and mapping (?)

Template Parameters

K	- Dataset key type
T	- Dataset type

Parameters

path	- File path
funcP	- (?)

Returns

10.7.3.5 printHeader()

```
void faster::fastContext::printHeader ( )
```

Prints a header for task execution information.

To be used with faster::fastContext::updateInfo()

10.7.3.6 registerFunction() [1/2]

```
void faster::fastContext::registerFunction (
             void * funcP )
```

Register a user custom function in the context.

Registering a user custom functions is necessary in order to pass it as parametes to FDD functions like **map** and **reduce**.

Parameters

funcP	- Function pointer to a user defined function.

10.7.3.7 registerFunction() [2/2]

```
void faster::fastContext::registerFunction (
            void * funcP,
            const std::string name )
```

Register a user custom function in the context.

Registering a user custom functions is necessary in order to pass it as parametes to FDD functions like **map** and **reduce**.

Parameters

funcP	- Function pointer to a user defined function.
name	- Custom name to registered funciton.

10.7.3.8 registerGlobal() [1/3]

```
template<class T >
void faster::fastContext::registerGlobal (
            T * varP )
```

Gegisters a primitive global varible to be used inside used defined functions in distributted environment.

Template Parameters

T	- Type of the global variable to be registered

Parameters

varP	- Global variable to be registered

10.7.3.9 registerGlobal() [2/3]

```
template<class T >
void faster::fastContext::registerGlobal (
            T ** varP,
            size_t s )
```

Gegisters a global array to be used inside used defined functions in distributted environment.

Template Parameters

T	- Type of the global array to be registered

Parameters

varP	- Global array to be registered

Parameters

s	- Size of the array

10.7.3.10 registerGlobal() [3/3]

```
template<class T >
void faster::fastContext::registerGlobal (
             std::vector< T > * varP )
```

Gegisters a global Vector to be used inside used defined functions in distributted environment.

Template Parameters

T	- Type of the global vector to be registered

Parameters

varP	- Global vector to be registered

10.7.3.11 startWorkers()

```
void faster::fastContext::startWorkers ( )
```

Start worker machines computation.

When this function is called, the driver processes and works processes diverge from execution. While the Driver process starts to execute user code, the worker processes start to waiting for tasks. Then workers should exit short after this function is called.

10.7.3.12 updateInfo()

```
void faster::fastContext::updateInfo ( )
```

Prints information from tesk ran since last faster::fastContext::updateInfo() called.

To be used with faster::fastContext::printHeader()

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastContext.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fastContext.cpp

10.8 faster::fastScheduler Class Reference

10.8.1 Description

Public Member Functions

- **fastScheduler** (unsigned int numProcs, std::vector< std::string > ∗funcName)
- fastTask ∗ **enqueueTask** (fddOpType opT, unsigned long int idSrc, unsigned long int idRes, int funcId, size_t size, std::vector< std::tuple< void ∗, size_t, int > > &globalTable)
- fastTask ∗ **enqueueTask** (fddOpType opT, unsigned long int id, size_t size, std::vector< std::tuple< void ∗, size_t, int > > &globalTable)
- void **taskProgress** (unsigned long int id, unsigned long int pid, size_t time, procstat &stat)
- void **taskFinished** (unsigned long int id, size_t time)
- void **setCalibration** (std::vector< size_t > time)
- void **printProcstats** (fastTask ∗task)
- void **printTaskInfo** ()
- void **printTaskInfo** (size_t task)
- void **printHeader** ()
- void **updateTaskInfo** ()
- bool **dataMigrationNeeded** ()
- std::vector< std::deque< std::pair< int, long int > > > **getDataMigrationInfo** ()
- std::vector< size_t > **getAllocation** (size_t size)
- void **setAllocation** (std::vector< size_t > &alloc, size_t size)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastScheduler.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fastScheduler.cpp

10.9 faster::fastSettings Class Reference

```
#include <fastContext.h>
```

10.9.1 Description

Context Configuration Class.

Throught the fastSetting Class, the programmer can change default framework settings. like ...

Public Member Functions

- fastSettings ()

  fastSetting default constructor
- fastSettings (const fastSettings &s UNUSED)

  fastSetting dummy constructor
- void allowDataBalancing ()

  Enables dynamic load balancing.

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastContext.h

10.10 faster::fastTask Class Reference

10.10.1 Description

Public Attributes

- unsigned long int **id**
- unsigned long int **srcFDD**
- unsigned long int **destFDD**
- fddOpType **operationType**
- int **functionId**
- size_t **size**
- void ∗ **result**
- size_t **resultSize**
- size_t **workersFinished**
- std::vector< size_t > **times**
- size_t **duration**
- std::shared_ptr< std::vector< double > > **allocation**
- std::vector< procstat > **procstats**
- std::vector< std::tuple< void ∗, size_t, int > > **globals**

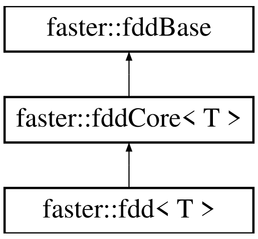
The documentation for this class was generated from the following file:

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastTask.h

10.11 faster::fdd< T > Class Template Reference

```
#include <fdd.h>
```

Inheritance diagram for faster::fdd< T >:



10.11.1 Description

template<class T>
class faster::fdd< T >

Fast Distributted Dataset(FDD) is like a cluster distributted Array. This class is the user side implementation.

Template Parameters

T	- The type of the dataset entries

Public Member Functions

- fdd (fastContext &c)

  Create a empty fdd.
- fdd (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)

  Create a empty fdd with a pre allocated size.
- fdd (fastContext &c, size_t s)

  Create a empty fdd with a pre allocated size.
- fdd (fastContext &c, T ∗data, size_t size)

  Create a fdd from a array in memory.
- fdd (fastContext &c, std::vector< T > &dataV)

  Create a fdd from a vector in memory.
- fdd (fastContext &c, const char ∗fileName)

  Create a fdd from a file.
- void assign (std::vector< T > &data)

  Assign a fdd content from a vector.
- void assign (T ∗data, size_t size)

  Assign a fdd content from a array.
- ∼fdd ()

  Class Destructor. WARNING: It will deallocate ditributted memory.
- std::vector< T > collect ()

  Brings the distributted data from a FDD to the driver memory.
- fdd< T > ∗ cache ()

  Prevents automatic memory deallocation from hapenning.
- template<typename U >
  fdd< U > ∗ map (mapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ map (PmapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ map (ImapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ map (IPmapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ bulkMap (bulkMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ bulkMap (PbulkMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkMap (IbulkMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>

- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkMap (IPbulkMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ flatMap (flatMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ flatMap (PflatMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ flatMap (IflatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ flatMap (IPflatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ bulkFlatMap (bulkFlatMapFunctionP< T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ bulkFlatMap (PbulkFlatMapFunctionP< T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkFlatMap (IbulkFlatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkFlatMap (IPbulkFlatMapFunctionP< T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- T reduce (reduceFunctionP< T > funcP)

  summarizes a fdd<T> into a single value of type T
- T bulkReduce (bulkReduceFunctionP< T > funcP)

  summarizes a fdd<T> into a single value of type T using a bulk function T F(T,T)

Additional Inherited Members

10.11.2 Member Function Documentation

10.11.2.1 collect()

```
template<class T >
std::vector<T> faster::fdd< T >::collect ( )  [inline]
```

Brings the distributted data from a FDD to the driver memory.

Returns

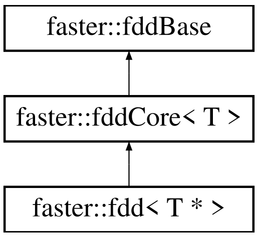
a vector with the content of the FDD

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastContext.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/fdd.h

10.12 faster::fdd< T * > Class Template Reference

Inheritance diagram for faster::fdd< T * >:



10.12.1 Description

template<class T>
class faster::fdd< T * >

Public Member Functions

- **fdd** (fastContext &c)
- **fdd** (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)
- **fdd** (fastContext &c, size_t s)
- **fdd** (fastContext &c, T *data[ ], size_t dataSizes[ ], size_t size)
- template<typename U >
  fdd< U > * **map** (mapPFunctionP< T, U > funcP)
- template<typename U >
  fdd< U > * **map** (PmapPFunctionP< T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **map** (ImapPFunctionP< T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **map** (IPmapPFunctionP< T, L, U > funcP)
- template<typename U >
  fdd< U > * **bulkMap** (bulkMapPFunctionP< T, U > funcP)
- template<typename U >
  fdd< U > * **bulkMap** (PbulkMapPFunctionP< T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **bulkMap** (IbulkMapPFunctionP< T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **bulkMap** (IPbulkMapPFunctionP< T, L, U > funcP)
- template<typename U >
  fdd< U > * **flatMap** (flatMapPFunctionP< T, U > funcP)
- template<typename U >
  fdd< U > * **flatMap** (PflatMapPFunctionP< T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **flatMap** (IflatMapPFunctionP< T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **flatMap** (IPflatMapPFunctionP< T, L, U > funcP)
- template<typename U >
  fdd< U > * **bulkFlatMap** (bulkFlatMapPFunctionP< T, U > funcP)
- template<typename U >
  fdd< U > * **bulkFlatMap** (PbulkFlatMapPFunctionP< T, U > funcP)

- template<typename L , typename U >
  indexedFdd< L, U > ∗ **bulkFlatMap** (IbulkFlatMapPFunctionP< T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > ∗ **bulkFlatMap** (IPbulkFlatMapPFunctionP< T, L, U > funcP)
- std::vector< T > **reduce** (PreducePFunctionP< T > funcP)
- std::vector< T > **bulkReduce** (PbulkReducePFunctionP< T > funcP)
- std::vector< std::pair< T ∗, size_t > > **collect** ()
- fdd< T ∗ > ∗ **cache** ()

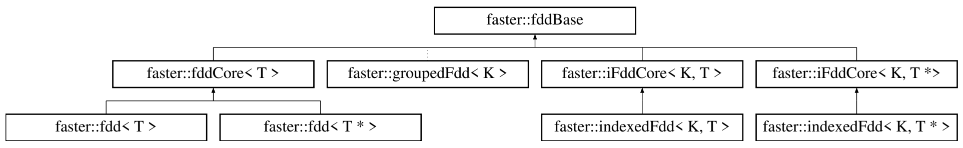
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fdd.h

10.13 faster::fddBase Class Reference

Inheritance diagram for faster::fddBase:



10.13.1 Description

Public Member Functions

- void **setSize** (size_t &s)
- size_t getSize ()
  Returns the size of the dataset.
- int getId ()
  Returns the identification number of the dataset.
- const std::vector< size_t > & getAlloc ()
  Returns the allocation identification number of the dataset.
- fddType **tType** ()
- fddType **kType** ()
- bool isCached ()
  Returns true if the dataset is cached.
- virtual void **discard** ()=0
- virtual bool **isGroupedByKey** ()=0
- virtual void **setGroupedByKey** (bool gbk)=0

Protected Attributes

- fddType **_kType**
- fddType **_tType**
- unsigned long int **id**
- unsigned long int **totalBlocks**
- unsigned long int **size**
- std::vector< size_t > **dataAlloc**
- bool **cached**

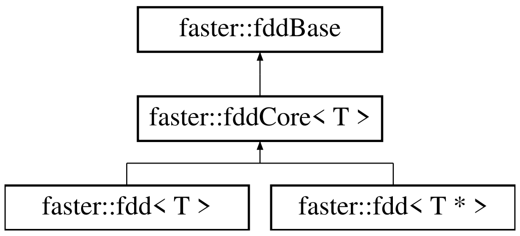
The documentation for this class was generated from the following file:

- /home/mtcs/pesquisa/faster/faster.git/src/include/fddBase.h

10.14 faster::fddCore< T > Class Template Reference

```
#include <fdd.h>
```

Inheritance diagram for faster::fddCore< T >:



10.14.1 Description

template<typename T>
class faster::fddCore< T >

core class that implements simple operations.

Template Parameters

T	- The type of the dataset entries

Public Member Functions

- void discard ()

  deallocates previusly cached fdd
- void writeToFile (std::string &path, std::string &sufix)

  Writes FDD content to file.
- void ∗ getKeyMap ()

(UNUSED)
- void setKeyMap (void ∗keyMap UNUSED)

  (UNUSED)
- bool isGroupedByKey ()

  (UNUSED)
- void setGroupedByKey (bool gbk UNUSED)

  (UNUSED)

Protected Member Functions

- **fddCore** (fastContext &c)
- **fddCore** (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)
- fddBase ∗ **_map** (void ∗funcP, fddOpType op, fddBase ∗newFdd)
- template<typename L , typename U >
  indexedFdd< L, U > ∗ **mapI** (void ∗funcP, fddOpType op)
- template<typename U >
  fdd< U > ∗ **map** (void ∗funcP, fddOpType op)

Protected Attributes

- fastContext ∗ **context**

10.14.2 Member Function Documentation

10.14.2.1 writeToFile()

```
template<typename T >
void faster::fddCore< T >::writeToFile (
            std::string & path,
            std::string & sufix )
```

Writes FDD content to file.

Every process will write its own file with a rank number between the prefix and the suffix.

Parameters

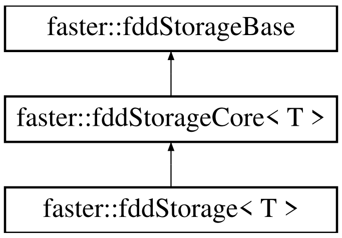
path	- Prefix of the file path to be written
sufix	- Sufix of the file path to be written

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fdd.h

10.15 faster::fddStorage< T > Class Template Reference

Inheritance diagram for faster::fddStorage< T >:



10.15.1 Description

template<class T>
class faster::fddStorage< T >

Public Member Functions

- **fddStorage** (size_t s)
- **fddStorage** (T ∗data, size_t s)
- void **setData** (T ∗data, size_t s)
- void **setDataRaw** (void ∗data, size_t s)
- void **setSize** (size_t s) override
- void **insert** (T &item)
- void **grow** (size_t toSize)
- void **shrink** ()

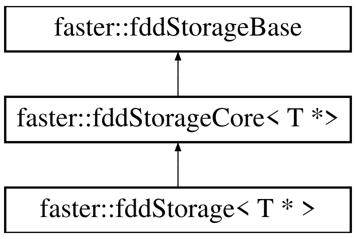
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/fddStorage.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fddStorage.cpp

10.16 faster::fddStorage< T ∗ > Class Template Reference

Inheritance diagram for faster::fddStorage< T ∗ >:



10.16.1 Description

template$<$class T$>$
class faster::fddStorage$<$ T $*$ $>$

Public Member Functions

- **fddStorage** (size_t s)
- **fddStorage** (T $**$data, size_t $*$lineSizes, size_t s)
- void **setData** (T $**$data, size_t $*$lineSizes, size_t s)
- void **setDataRaw** (void $*$data, size_t $*$lineSizes, size_t s)
- void **setSize** (size_t s) override
- void **insert** (T $*$&item, size_t s)
- size_t $*$ **getLineSizes** ()
- void **grow** (size_t toSize)
- void **shrink** ()

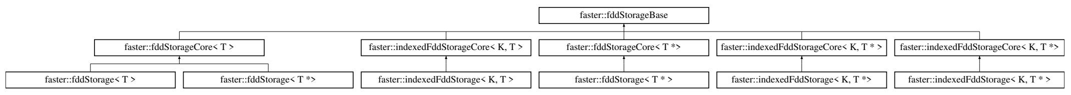
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fddStorage.h

10.17 faster::fddStorageBase Class Reference

Inheritance diagram for faster::fddStorageBase:


faster::fddStorageBase
faster::fddStorageCore< T >
faster::indexedFddStorageCore< K, T >
faster::fddStorageCore< T *>
faster::indexedFddStorageCore< K, T * >
faster::indexedFddStorageCore< K, T *>
faster::fddStorage< T >
faster::fddStorage< T *>
faster::indexedFddStorage< K, T >
faster::fddStorage< T * >
faster::indexedFddStorage< K, T *>
faster::indexedFddStorage< K, T * >

10.17.1 Description

Public Member Functions

- virtual void **grow** (size_t toSize)=0
- size_t **getSize** ()
- virtual void **setSize** (size_t s UNUSED)

Protected Attributes

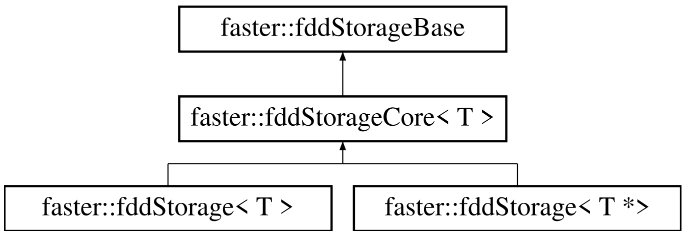
- size_t **size**
- size_t **allocSize**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fddStorageBase.h

10.18 faster::fddStorageCore< T > Class Template Reference

Inheritance diagram for faster::fddStorageCore< T >:



10.18.1 Description

template<class T>
class faster::fddStorageCore< T >

Public Member Functions

- **fddStorageCore** (size_t s)
- T * **getData** ()
- void **setSize** (size_t s UNUSED)
- T & **operator[ ]** (size_t ref)

Protected Attributes

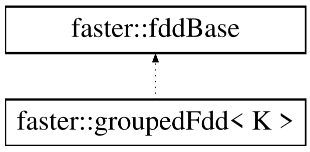
- T * **localData**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fddStorage.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/fddStorage.cpp

10.19 faster::groupedFdd< K > Class Template Reference

Inheritance diagram for faster::groupedFdd< K >:



10.19.1 Description

template<typename K>
class faster::groupedFdd< K >

Public Member Functions

- template<typename T , typename U >
  groupedFdd (fastContext *c, iFddCore< K, T > *fdd0, iFddCore< K, U > *fdd1, system_clock::time_point &start)

  Creates a indexedFdd group with two members.
- template<typename T , typename U , typename V >
  groupedFdd (fastContext *c, iFddCore< K, T > *fdd0, iFddCore< K, U > *fdd1, iFddCore< K, V > *fdd2, system_clock::time_point &start)

  Creates a indexedFdd group with two members.
- groupedFdd< K > * cache ()

  Prevents automatic memory deallocation from hapenning.
- void discard ()

  deallocates previously cached fdd
- bool **isGroupedByKey** ()
- void **setGroupedByKey** (bool gbk UNUSED)
- groupedFdd< K > * **updateByKey** (updateByKeyG2FunctionP< K > funcP)
- groupedFdd< K > * **updateByKey** (updateByKeyG3FunctionP< K > funcP)
- groupedFdd< K > * **bulkUpdate** (bulkUpdateG2FunctionP< K > funcP)
- groupedFdd< K > * **bulkUpdate** (bulkUpdateG3FunctionP< K > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **mapByKey** (ImapByKeyG2FunctionP< K, Ko, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **mapByKey** (ImapByKeyG3FunctionP< K, Ko, To > funcP)
- template<typename To >
  fdd< To > * **mapByKey** (mapByKeyG2FunctionP< K, To > funcP)
- template<typename To >
  fdd< To > * **mapByKey** (mapByKeyG3FunctionP< K, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **flatMapByKey** (IflatMapByKeyG2FunctionP< K, Ko, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **flatMapByKey** (IflatMapByKeyG3FunctionP< K, Ko, To > funcP)
- template<typename To >
  fdd< To > * **flatMapByKey** (flatMapByKeyG2FunctionP< K, To > funcP)
- template<typename To >
  fdd< To > * **flatMapByKey** (flatMapByKeyG3FunctionP< K, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **bulkFlatMap** (IbulkFlatMapG2FunctionP< K, Ko, To > funcP)
- template<typename Ko , typename To >
  indexedFdd< Ko, To > * **bulkFlatMap** (IbulkFlatMapG3FunctionP< K, Ko, To > funcP)
- template<typename To >
  fdd< To > * **bulkFlatMap** (bulkFlatMapG2FunctionP< K, To > funcP)
- template<typename To >
  fdd< To > * **bulkFlatMap** (bulkFlatMapG3FunctionP< K, To > funcP)

10.19.2 Constructors and Destructors

10.19.2.1 groupedFdd() [1/2]

```
template<typename K >
template<typename T , typename U >
faster::groupedFdd< K >::groupedFdd (
            fastContext * c,
            iFddCore< K, T > * fdd0,
            iFddCore< K, U > * fdd1,
            system_clock::time_point & start )  [inline]
```

Creates a indexedFdd group with two members.

Template Parameters

T	- value type of the first dataset
U	- value type of the second dataset

Parameters

c	- the context
fdd0	- first dataset
fdd1	- second dataset
start	- start timestamp

10.19.2.2 groupedFdd() [2/2]

```
template<typename K >
template<typename T , typename U , typename V >
faster::groupedFdd< K >::groupedFdd (
            fastContext * c,
            iFddCore< K, T > * fdd0,
            iFddCore< K, U > * fdd1,
            iFddCore< K, V > * fdd2,
            system_clock::time_point & start )  [inline]
```

Creates a indexedFdd group with two members.

Template Parameters

T	- value type of the first dataset
U	- value type of the second dataset
V	- value type of the third dataset

Parameters

c	- the context
fdd0	- first dataset
fdd1	- second dataset

Parameters

fdd2	- third dataset
start	- start timestamp

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/groupedFdd.h

10.20 faster::hasher< K > Class Template Reference

10.20.1 Description

template<typename K>
class faster::hasher< K >

Public Member Functions

- **hasher** (int spectrum)
- int **get** (K key)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hasher.h

10.21 faster::hasher< double > Class Template Reference

10.21.1 Description

template<>
class faster::hasher< double >

Public Member Functions

- **hasher** (int spectrum)
- int **get** (double key)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hasher.h

10.22 faster::hasher< float > Class Template Reference

10.22.1 Description

template<>
class faster::hasher< float >

Public Member Functions

- **hasher** (int spectrum)
- int **get** (float key)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hasher.h

10.23 faster::hasher< std::string > Class Template Reference

10.23.1 Description

template<>
class faster::hasher< std::string >

Public Member Functions

- **hasher** (int spectrum)
- int **get** (std::string key)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hasher.h

10.24 faster::hdfsEngine Class Reference

10.24.1 Description

Public Member Functions

- bool **isReady** ()
- bool **isConnected** ()
- faster::hdfsFile **open** (std::string path, fileMode mode)
- void **close** (faster::hdfsFile &f)
- void **del** (std::string path)
- bool **exists** (std::string path)

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hdfsEngine.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/hdfsEngine.cpp

10.25 faster::hdfsFile Class Reference

10.25.1 Description

Public Member Functions

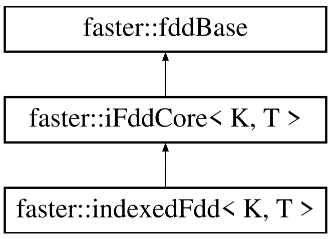
- **hdfsFile** (void ∗fs, std::string &path, fileMode mode)
- void **close** ()
- size_t **read** (char ∗v, size_t n)
- size_t **write** (char ∗v, size_t n)
- size_t **seek** (size_t offset)
- size_t **readLine** (char ∗v, size_t n, char sep)
- std::vector< std::deque< int > > **getBlocksLocations** ()
- void **del** ()

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/hdfsEngine.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/hdfsEngine.cpp

10.26 faster::iFddCore< K, T > Class Template Reference

Inheritance diagram for faster::iFddCore< K, T >:



10.26.1 Description

template<typename K, typename T>
class faster::iFddCore< K, T >

Public Member Functions

- template<typename U >
  groupedFdd< K > ∗ cogroup (iFddCore< K, U > ∗fdd1)

  Groupes two datasets twogether according with the keys of the first dataset.
- template<typename U , typename V >
  groupedFdd< K > ∗ cogroup (iFddCore< K, U > ∗fdd1, iFddCore< K, V > ∗fdd2)

  Groupes tree datasets together according with the keys of the first dataset.
- std::unordered_map< K, size_t > countByKey ()

  Count how many unique key there is in the dataset.

- indexedFdd< K, T > ∗ groupByKey ()

    Groups distributed dataset by key.
- void discard ()

    deallocates previously cached FDD
- void writeToFile (std::string path, std::string sufix)

    Writes FDD content to file.
- bool isGroupedByKey ()

    Determines if a dataset is grouped by key.
- void setGroupedByKey (bool gbk)

    (UNUSED)
- void setGroupedByMap (bool gbm)

    (UNUSED)

Protected Member Functions

- **iFddCore** (fastContext &c)
- **iFddCore** (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)
- std::unordered_map< K, std::tuple< size_t, int, size_t > > ∗ **calculateKeyCount** (std::vector< std::pair< void ∗, size_t > > &result)
- std::unordered_map< K, int > **calculateKeyMap** (std::unordered_map< K, std::tuple< size_t, int, size_t >> &count)
- void **update** (void ∗funcP, fddOpType op)
- fddBase ∗ **_map** (void ∗funcP, fddOpType op, fddBase ∗newFdd, system_clock::time_point &start)
- template<typename U >
  fdd< U > ∗ **map** (void ∗funcP, fddOpType op)
- template<typename L , typename U >
  indexedFdd< L, U > ∗ **mapI** (void ∗funcP, fddOpType op)
- indexedFdd< K, T > ∗ **groupByKeyMapped** ()
- indexedFdd< K, T > ∗ **groupByKeyHashed** ()

Protected Attributes

- bool **groupedByKey**
- bool **groupedByMap**
- fastContext ∗ **context**

10.26.2 Member Function Documentation

10.26.2.1 countByKey()

```
template<typename K , typename T >
std::unordered_map< K, size_t > faster::iFddCore< K, T >::countByKey ( )
```

Count how many unique key there is in the dataset.

Returns

a unordered_map (hash) of the key count.

10.26.2.2 isGroupedByKey()

```
template<typename K, typename T>
bool faster::iFddCore< K, T >::isGroupedByKey ( )  [inline], [virtual]
```

Determines if a dataset is grouped by key.

Returns

true is it has been groupe by key

Implements faster::fddBase.

10.26.2.3 writeToFile()

```
template<typename K , typename T >
void faster::iFddCore< K, T >::writeToFile (
            std::string path,
            std::string sufix )
```

Writes FDD content to file.

Every process will write its own file with a rank number between the prefix and the suffix.

Parameters

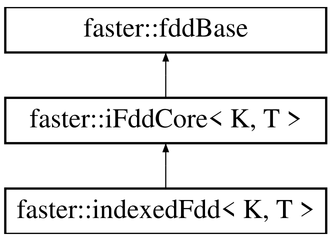
path	- Prefix of the file path to be written
sufix	- Sufix of the file path to be written

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/groupedFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFdd.h

10.27 faster::indexedFdd< K, T > Class Template Reference

Inheritance diagram for faster::indexedFdd< K, T >:



10.27.1 Description

template<typename K, typename T>
class faster::indexedFdd< K, T >

Public Member Functions

- indexedFdd (fastContext &c)

  Create a empty indexedFdd.
- indexedFdd (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)

  Create a empty indexedFdd with a pre allocated size.
- indexedFdd (fastContext &c, size_t s)

  Create a empty indexedFdd with a pre allocated size.
- indexedFdd (fastContext &c, K ∗keys, T ∗data, size_t size)

  Create a indexedFdd from a array in memory.
- ∼indexedFdd ()

  Class Destructor. WARNING: It will deallocate distributed memory.
- std::vector< std::pair< K, T > > collect ()

  Brings the distributted data from a indexedFDD to the driver memory.
- indexedFdd< K, T > ∗ cache ()

  Prevents automatic memory deallocation from hapenning.
- indexedFdd< K, T > ∗ update (updateIFunctionP< K, T > funcP)

  updates the content of a indexedFDD
- template<typename L , typename U >
  indexedFdd< L, U > ∗ map (ImapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ map (IPmapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename U >
  fdd< U > ∗ map (mapIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename U >
  fdd< U > ∗ map (PmapIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ mapByKey (ImapByKeyIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ mapByKey (IPmapByKeyIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ mapByKey (mapByKeyIFunctionP< K, T, U > funcP)

  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ mapByKey (PmapByKeyIFunctionP< K, T, U > funcP)

  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkMap (IbulkMapIFunctionP< K, T, L, U > funcP)

  creates a indexedFdd<L,U>

- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkMap (IPbulkMapIFunctionP< K, T, L, U > funcP)
  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ bulkMap (bulkMapIFunctionP< K, T, U > funcP)
  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ bulkMap (PbulkMapIFunctionP< K, T, U > funcP)
  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ flatMap (IflatMapIFunctionP< K, T, L, U > funcP)
  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ flatMap (IPflatMapIFunctionP< K, T, L, U > funcP)
  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ flatMap (flatMapIFunctionP< K, T, U > funcP)
  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ flatMap (PflatMapIFunctionP< K, T, U > funcP)
  creates a fdd<U ∗>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkFlatMap (IbulkFlatMapIFunctionP< K, T, L, U > funcP)
  creates a indexedFdd<L,U>
- template<typename L , typename U >
  indexedFdd< L, U > ∗ bulkFlatMap (IPbulkFlatMapIFunctionP< K, T, L, U > funcP)
  creates a indexedFdd<L,U∗>
- template<typename L , typename U >
  fdd< U > ∗ bulkFlatMap (bulkFlatMapIFunctionP< K, T, U > funcP)
  creates a fdd<U>
- template<typename L , typename U >
  fdd< U > ∗ bulkFlatMap (PbulkFlatMapIFunctionP< K, T, U > funcP)
  creates a fdd<U ∗>
- std::pair< K, T > reduce (IreduceIFunctionP< K, T > funcP)
  summarizes a fdd<K,T> into a single value of type T
- std::pair< K, T > bulkReduce (IbulkReduceIFunctionP< K, T > funcP)
  summarizes a fdd<K,T> into a single value of type T using a bulk function pair<K,T> F(K, T, K, T)

Additional Inherited Members

10.27.2 Member Function Documentation

10.27.2.1 collect()

```
template<typename K , typename T >
std::vector<std::pair<K,T> > faster::indexedFdd< K, T >::collect ( )  [inline]
```

Brings the distributted data from a indexedFDD to the driver memory.

Returns

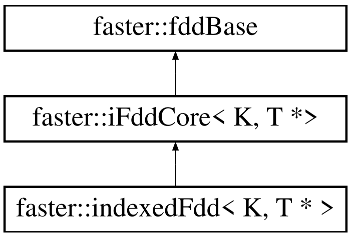
a vector with the content of the indexedFDD

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/fastContext.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFdd.h

10.28 faster::indexedFdd< K, T * > Class Template Reference

Inheritance diagram for faster::indexedFdd< K, T * >:



10.28.1 Description

template<typename K, typename T>
class faster::indexedFdd< K, T * >

Public Member Functions

- **indexedFdd** (fastContext &c)
- **indexedFdd** (fastContext &c, size_t s, const std::vector< size_t > &dataAlloc)
- **indexedFdd** (fastContext &c, size_t s)
- **indexedFdd** (fastContext &c, K *keys, T **data, size_t *dataSizes, size_t size)
- template<typename L , typename U >
  indexedFdd< L, U > * **map** (ImapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **map** (IPmapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  fdd< U > * **map** (mapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  fdd< U > * **map** (PmapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **mapByKey** (ImapByKeyIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **mapByKey** (IPmapByKeyIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  fdd< U > * **mapByKey** (mapByKeyIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  fdd< U > * **mapByKey** (PmapByKeyIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **bulkMap** (IbulkMapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **bulkMap** (IPbulkMapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  fdd< U > * **bulkMap** (bulkMapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  fdd< U > * **bulkMap** (PbulkMapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **flatMap** (IflatMapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > * **flatMap** (IPflatMapIPFunctionP< K, T, L, U > funcP)

- template<typename L , typename U >
  fdd< U > ∗ **flatMap** (flatMapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  fdd< U > ∗ **flatMap** (PflatMapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > ∗ **bulkFlatMap** (IbulkFlatMapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  indexedFdd< L, U > ∗ **bulkFlatMap** (IPbulkFlatMapIPFunctionP< K, T, L, U > funcP)
- template<typename L , typename U >
  fdd< U > ∗ **bulkFlatMap** (bulkFlatMapIPFunctionP< K, T, U > funcP)
- template<typename L , typename U >
  fdd< U > ∗ **bulkFlatMap** (PbulkFlatMapIPFunctionP< K, T, U > funcP)
- std::vector< std::pair< K, T > > **reduce** (IPreduceIPFunctionP< K, T > funcP)
- std::vector< std::pair< K, T > > **bulkReduce** (IPbulkReduceIPFunctionP< K, T > funcP)
- std::vector< std::tuple< K, T ∗, size_t > > **collect** ()
- indexedFdd< K, T ∗ > ∗ **cache** ()

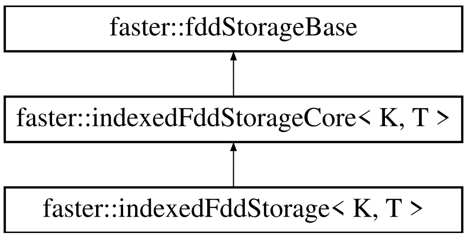
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFdd.h

10.29 faster::indexedFddStorage< K, T > Class Template Reference

Inheritance diagram for faster::indexedFddStorage< K, T >:


faster::fddStorageBase
faster::indexedFddStorageCore< K, T >
faster::indexedFddStorage< K, T >

10.29.1 Description

template<class K, class T>
class faster::indexedFddStorage< K, T >

Public Member Functions

- **indexedFddStorage** (size_t s)
- **indexedFddStorage** (K ∗keys, T ∗data, size_t s)
- void **setData** (K ∗keys, T ∗data, size_t s)
- void **setDataRaw** (void ∗keys, void ∗data, size_t s)
- void **setSize** (size_t s) override
- void **insert** (K key, T &item)
- void **insertRaw** (void ∗d, size_t s)
- void **grow** (size_t toSize)
- void **shrink** ()

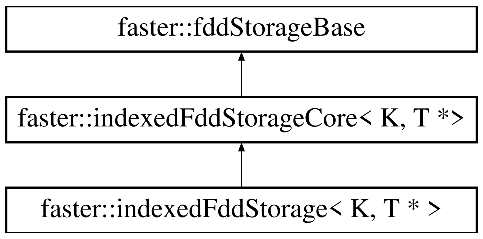
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerIFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFddStorage.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/indexedFddStorage.cpp

10.30 faster::indexedFddStorage$<$ K, T $*$ $>$ Class Template Reference

Inheritance diagram for faster::indexedFddStorage$<$ K, T $*$ $>$:


faster::fddStorageBase
faster::indexedFddStorageCore< K, T *>
faster::indexedFddStorage< K, T * >

10.30.1 Description

template$<$class K, class T$>$
class faster::indexedFddStorage$<$ K, T $*$ $>$

Public Member Functions

- **indexedFddStorage** (size_t s)
- **indexedFddStorage** (K $*$keys, T $**$data, size_t $*$lineSizes, size_t s)
- void **setData** (K $*$keys, T $**$data, size_t $*$lineSizes, size_t s)
- void **setDataRaw** (void $*$keys, void $*$data, size_t $*$lineSizes, size_t s)
- void **setSize** (size_t s) override
- void **insert** (K key, T $*$&item, size_t s)
- void **insertRaw** (void $*$d, size_t s)
- size_t $*$ **getLineSizes** ()
- void **grow** (size_t toSize)
- void **shrink** ()

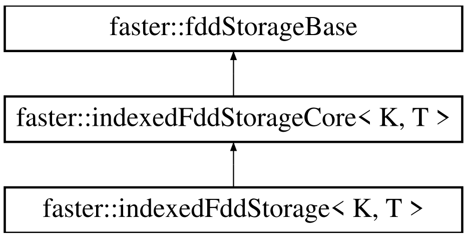
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFddStorage.h

10.31 faster::indexedFddStorageCore< K, T > Class Template Reference

Inheritance diagram for faster::indexedFddStorageCore< K, T >:


faster::fddStorageBase
faster::indexedFddStorageCore< K, T >
faster::indexedFddStorage< K, T >

10.31.1 Description

template< class K, class T >
class faster::indexedFddStorageCore< K, T >

Public Member Functions

- **indexedFddStorageCore** (size_t s)
- T ∗ **getData** ()
- K ∗ **getKeys** ()
- void **setSize** (size_t s UNUSED)
- T & **operator[ ]** (size_t ref)
- void **sortByKey** ()

Protected Attributes

- T ∗ **localData**
- K ∗ **localKeys**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/indexedFddStorage.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/indexedFddStorage.cpp

10.32 faster::procstat Class Reference

10.32.1 Description

Public Attributes

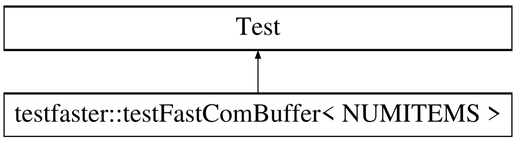
- double **ram**
- long unsigned **utime**
- long unsigned **stime**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/misc.h

10.33 testfaster::testFastComBuffer< NUMITEMS > Class Template Reference

Inheritance diagram for testfaster::testFastComBuffer< NUMITEMS >:



10.33.1 Description

template<int NUMITEMS = 10∗1000>
class testfaster::testFastComBuffer< NUMITEMS >

Public Member Functions

- template<typename T >
  void **comp** (T &a, T &b)
- template<typename T >
  void **comp** (std::pair< T, T > &a, std::pair< T, T > &b)
- template<typename T >
  void **comp** (std::tuple< T, T, T, T > &a, std::tuple< T, T, T, T > &b)
- template<typename T >
  void **comp** (std::vector< T > &a, std::vector< T > &b)
- void **comp** (std::vector< std::string > &a, std::vector< std::string > &b)
- template<typename T >
  void **testWrite** (T &val, const char ∗result, int size)

Protected Member Functions

- virtual void **SetUp** ()
- virtual void **TearDown** ()

Protected Attributes

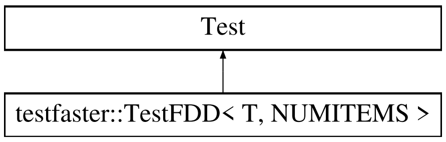
- faster::fastCommBuffer **buff**

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

- /home/mtcs/pesquisa/faster/faster.git/src/tests/gtest-fastCommBuffer.cpp

10.34 testfaster::TestFDD< T, NUMITEMS > Class Template Reference

Inheritance diagram for testfaster::TestFDD< T, NUMITEMS >:



10.34.1 Description

template< typename T, int NUMITEMS = 10∗1000>
class testfaster::TestFDD< T, NUMITEMS >

Protected Member Functions

- virtual void **SetUp** ()
- virtual void **TearDown** ()

Protected Attributes

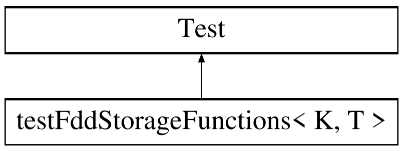
- fastContext **fc**
- vector< T > **localData**
- fdd< T > ∗ **data** = NULL

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

- /home/mtcs/pesquisa/faster/faster.git/src/tests/gtest-fdd.cpp

10.35 testFddStorageFunctions< K, T > Class Template Reference

Inheritance diagram for testFddStorageFunctions< K, T >:



10.35.1 Description

template< typename K, typename T>
class testFddStorageFunctions< K, T >

Protected Member Functions

- virtual void **SetUp** ()
- virtual void **TearDown** ()

Protected Attributes

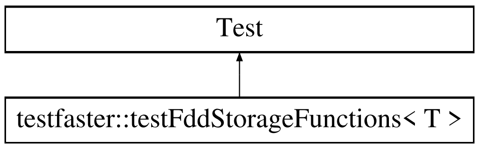
- faster::indexedFddStorage< K, T > **storage**
- std::vector< T > **rawKeys**
- std::vector< T > **rawData**

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

- /home/mtcs/pesquisa/faster/faster.git/src/tests/gtest-indexedFddStorage.cpp

10.36 testfaster::testFddStorageFunctions< T > Class Template Reference

Inheritance diagram for testfaster::testFddStorageFunctions< T >:



10.36.1 Description

template<typename T>
class testfaster::testFddStorageFunctions< T >

Protected Member Functions

- virtual void **SetUp** ()
- virtual void **TearDown** ()

Protected Attributes

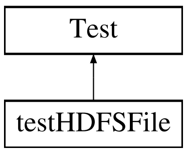
- faster::fddStorage< T > **storage**
- std::vector< T > **rawData**

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

- /home/mtcs/pesquisa/faster/faster.git/src/tests/gtest-fddStorage.cpp

10.37 testHDFSFile Class Reference

Inheritance diagram for testHDFSFile:



10.37.1 Description

Public Attributes

- faster::hdfsEngine **fs**

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

- /home/mtcs/pesquisa/faster/faster.git/src/tests/gtest-hdfsEngine.cpp

10.38 faster::worker Class Reference

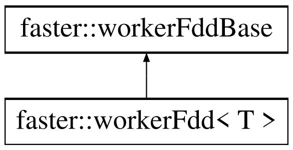
10.38.1 Description

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/worker.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/worker.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerCreate.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerICreate.cpp
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerRun.cpp

10.39 faster::workerFdd< T > Class Template Reference

Inheritance diagram for faster::workerFdd< T >:



10.39.1 Description

template<class T>
class faster::workerFdd< T >

Public Member Functions

- **workerFdd** (fddType t)
- **workerFdd** (fddType kt, fddType t)
- **workerFdd** (unsigned long int ident, fddType t)
- **workerFdd** (unsigned long int ident, fddType t, size_t size)
- **workerFdd** (unsigned long int ident, fddType kt, fddType t)
- **workerFdd** (unsigned long int ident, fddType kt, fddType t, size_t size)
- fddType **getType** ()
- fddType **getKeyType** ()
- void ∗ **getItem** (size_t address)
- void ∗ **getKeys** ()
- void ∗ **getData** ()
- size_t **getSize** ()
- size_t **itemSize** ()
- size_t **baseSize** ()
- void **setSize** (size_t s)
- void **deleteItem** (void ∗item)
- void **shrink** ()
- void **setData** (void ∗d, size_t size)
- void **setData** (void ∗d, size_t ∗lineSizes, size_t size)

- void **setData** (void ∗k, void ∗d, size_t size)
- void **setData** (void ∗k, void ∗d, size_t ∗lineSizes, size_t size)
- void **setDataRaw** (void ∗data, size_t size) override
- void **setDataRaw** (void ∗data, size_t ∗lineSizes, size_t size)
- void **setDataRaw** (void ∗k, void ∗d, size_t s)
- void **setDataRaw** (void ∗k, void ∗d, size_t ∗l, size_t s)
- size_t ∗ **getLineSizes** ()
- void **insert** (void ∗k, void ∗in, size_t s)
- void **insertl** (void ∗in)
- void **apply** (void ∗func UNUSED, fddOpType op UNUSED, workerFddBase ∗dest UNUSED, fastCommBuffer &comm UNUSED)
- void **preapply** (unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm) override
- void **collect** (fastComm ∗comm) override
- void **groupByKey** (fastComm ∗comm)
- void **countByKey** (fastComm ∗comm)
- void **exchangeDataByKey** (fastComm ∗comm)
- std::vector< std::vector< void ∗ > > ∗ **getKeyLocations** ()
- void ∗ **getUKeys** ()
- void **setUKeys** (void ∗uk)
- void ∗ **getKeyMap** ()
- void **setKeyMap** (void ∗km)
- void **writeToFile** (void ∗path, size_t procId, void ∗sufix)

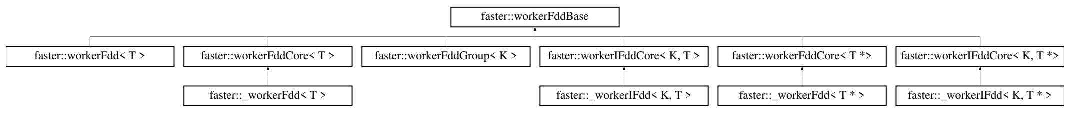
Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/include/workerFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerFddWrapper.cpp

10.40 faster::workerFddBase Class Reference

Inheritance diagram for faster::workerFddBase:


faster::workerFddBase
faster::workerFdd< T >
faster::workerFddCore< T >
faster::workerFddGroup< K >
faster::workerIFddCore< K, T >
faster::workerFddCore< T *>
faster::workerIFddCore< K, T *>
faster::_workerFdd< T >
faster::_workerIFdd< K, T >
faster::_workerFdd< T * >
faster::_workerIFdd< K, T * >

10.40.1 Description

Public Member Functions

- **workerFddBase** (unsigned int ident, fddType t)
- virtual fddType **getType** ()=0
- virtual fddType **getKeyType** ()=0
- virtual void **setData** (void ∗, size_t)=0
- virtual void **setData** (void ∗, size_t ∗, size_t)=0
- virtual void **setData** (void ∗, void ∗, size_t)=0
- virtual void **setData** (void ∗, void ∗, size_t ∗, size_t)=0
- virtual void **setDataRaw** (void ∗, size_t)=0
- virtual void **setDataRaw** (void ∗, size_t ∗, size_t)=0
- virtual void **setDataRaw** (void ∗, void ∗, size_t)=0
- virtual void **setDataRaw** (void ∗, void ∗, size_t ∗, size_t)=0
- virtual void ∗ **getItem** (size_t)=0
- virtual void ∗ **getKeys** ()=0
- virtual void ∗ **getData** ()=0
- virtual size_t **getSize** ()=0
- virtual size_t ∗ **getLineSizes** ()=0
- virtual void **setSize** (size_t s)=0
- virtual size_t **itemSize** ()=0
- virtual size_t **baseSize** ()=0
- virtual void **deleteItem** (void ∗item)=0
- virtual void **shrink** ()=0
- virtual void **insertl** (void ∗v)=0
- virtual void **insert** (void ∗k, void ∗v, size_t s)=0
- virtual void **preapply** (unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm)=0
- virtual void **apply** (void ∗func, fddOpType op, workerFddBase ∗dest, fastCommBuffer &buffer)=0
- virtual void **collect** (fastComm ∗comm)=0
- virtual void **exchangeDataByKey** (fastComm ∗comm)=0
- virtual std::vector< std::vector< void ∗ > > ∗ **getKeyLocations** ()=0
- virtual void ∗ **getUKeys** ()=0
- virtual void **setUKeys** (void ∗uk)=0
- virtual void ∗ **getKeyMap** ()=0
- virtual void **setKeyMap** (void ∗km)=0
- virtual void **writeToFile** (void ∗path, size_t procId, void ∗sufix)=0

Protected Attributes

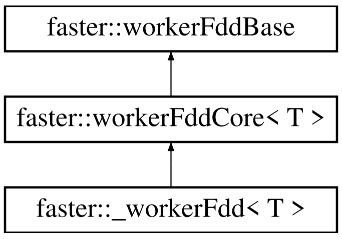
- unsigned long int **id**
- fddType **type**
- fddType **keyType**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/workerFddBase.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerFddBase.cpp

10.41 faster::workerFddCore< T > Class Template Reference

Inheritance diagram for faster::workerFddCore< T >:



10.41.1 Description

template<class T>
class faster::workerFddCore< T >

Public Member Functions

- **workerFddCore** (unsigned int ident, fddType t)
- **workerFddCore** (unsigned int ident, fddType t, size_t size)
- void **setData** (void ∗k UNUSED, void ∗d UNUSED, size_t size UNUSED)
- void **setDataRaw** (void ∗keys UNUSED, void ∗data UNUSED, size_t size UNUSED) override
- void **setDataRaw** (void ∗keys UNUSED, void ∗data UNUSED, size_t ∗lineSizes UNUSED, size_t size UN←USED) override
- fddType **getType** () override
- fddType **getKeyType** () override
- T & **operator[ ]** (size_t address)
- void ∗ **getItem** (size_t address)
- void ∗ **getKeys** () override
- void ∗ **getData** () override
- size_t **getSize** () override
- size_t **itemSize** () override
- size_t **baseSize** () override
- void **setSize** (size_t s)
- void **deleteItem** (void ∗item) override
- void **shrink** ()
- void **writeToFile** (void ∗path, size_t procId, void ∗sufix)
- void **preapply** (unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm)

Protected Member Functions

- void **exchangeDataByKey** (fastComm ∗comm UNUSED)
- void ∗ **getUKeys** ()
- void **setUKeys** (void ∗uk UNUSED)
- void ∗ **getKeyMap** ()
- void **setKeyMap** (void ∗km UNUSED)
- std::vector< std::vector< void ∗ > > ∗ **getKeyLocations** ()

Protected Attributes

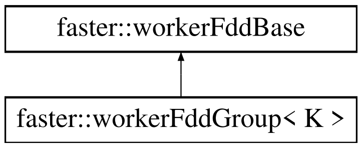
- fddStorage< T > ∗ **localData**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerFddCore.cpp

10.42 faster::workerFddGroup< K > Class Template Reference

Inheritance diagram for faster::workerFddGroup< K >:



10.42.1 Description

template<typename K>
class faster::workerFddGroup< K >

Public Member Functions

- **workerFddGroup** (unsigned long int id, fddType keyT, std::vector< workerFddBase ∗> &members)
- fddType **getType** ()
- fddType **getKeyType** ()
- void **setData** (void ∗d UNUSED, size_t s UNUSED)
- void **setData** (void ∗d UNUSED, size_t ∗ds UNUSED, size_t s UNUSED)
- void **setData** (void ∗k UNUSED, void ∗d UNUSED, size_t s UNUSED)
- void **setData** (void ∗k UNUSED, void ∗d UNUSED, size_t ∗ds UNUSED, size_t s UNUSED)
- void **setDataRaw** (void ∗d UNUSED, size_t s UNUSED)
- void **setDataRaw** (void ∗d UNUSED, size_t ∗ds UNUSED, size_t s UNUSED)
- void **setDataRaw** (void ∗k UNUSED, void ∗d UNUSED, size_t s UNUSED)
- void **setDataRaw** (void ∗k UNUSED, void ∗d UNUSED, size_t ∗ds UNUSED, size_t s UNUSED)
- void ∗ **getItem** (size_t UNUSED p)
- void ∗ **getKeys** ()
- void ∗ **getData** ()
- size_t **getSize** ()
- size_t ∗ **getLineSizes** ()
- void **setSize** (size_t s UNUSED)
- size_t **itemSize** ()
- size_t **baseSize** ()
- void **deleteItem** (void ∗item UNUSED)
- void **shrink** ()
- void **insertl** (void ∗v UNUSED)
- void **insert** (void ∗k UNUSED, void ∗v UNUSED, size_t s UNUSED)
- void **apply** (void ∗func, fddOpType op, workerFddBase ∗dest, fastCommBuffer &buffer)
- void **preapply** (unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm)
- void **collect** (fastComm ∗comm UNUSED)
- void ∗ **getUKeys** ()
- void **setUKeys** (void ∗uk)
- void ∗ **getKeyMap** ()
- void **setKeyMap** (void ∗km)
- void **writeToFile** (void ∗path UNUSED, size_t procId UNUSED, void ∗sufix UNUSED)

Additional Inherited Members

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/workerFddGroup.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerFddGroup.cpp

10.43 faster::workerIFdd< K, T > Class Template Reference

10.43.1 Description

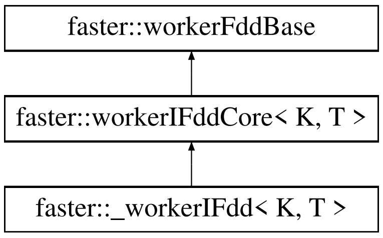
template<class K, class T>
class faster::workerIFdd< K, T >

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerFdd.h

10.44 faster::workerIFddCore< K, T > Class Template Reference

Inheritance diagram for faster::workerIFddCore< K, T >:



10.44.1 Description

template<typename K, typename T>
class faster::workerIFddCore< K, T >

Public Member Functions

- **workerIFddCore** (unsigned int ident, fddType kt, fddType t)
- **workerIFddCore** (unsigned int ident, fddType kt, fddType t, size_t size)
- fddType **getType** () override
- fddType **getKeyType** () override
- void **setData** (void ∗data UNUSED, size_t size UNUSED)
- void **setData** (void ∗data UNUSED, size_t ∗ls UNUSED, size_t size UNUSED)
- void **setDataRaw** (void ∗data UNUSED, size_t size UNUSED) override
- void **setDataRaw** (void ∗data UNUSED, size_t ∗lineSizes UNUSED, size_t size UNUSED) override
- T & **operator[ ]** (size_t address)

- void ∗ **getItem** (size_t address)
- void ∗ **getData** () override
- void ∗ **getKeys** ()
- size_t **getSize** () override
- size_t **itemSize** () override
- size_t **baseSize** () override
- void **setSize** (size_t s)
- void **deleteItem** (void ∗item) override
- void **shrink** ()
- std::vector< std::vector< T ∗ > > **findKeyInterval** (K ∗keys, T ∗data, size_t fddSize)
- void **preapply** (unsigned long int id, void ∗func, fddOpType op, workerFddBase ∗dest, fastComm ∗comm)
- bool **onlineReadStage3** (std::deque< std::vector< std::pair< K, T >>> &q2, omp_lock_t &q2lock)
- bool **onlinePartReadStage3** (std::unordered_map< K, int > &localKeyMap, fastComm ∗comm, void ∗funcP, std::deque< std::vector< std::pair< K, T >>> &q2, omp_lock_t &q2lock)
- void **onlineFullPartRead** (fastComm ∗comm, void ∗funcP)
- void **onlinePartRead** (fastComm ∗comm, void ∗funcP)
- void **onlineRead** (fastComm ∗comm)
- void **groupByKey** (fastComm ∗comm)
- void **groupByKeyHashed** (fastComm ∗comm)
- void **countByKey** (fastComm ∗comm)
- void **exchangeDataByKey** (fastComm ∗comm)
- bool **exchangeDataByKeyHashed** (fastComm ∗comm)
- void **exchangeDataByKeyMapped** (fastComm ∗comm)
- std::vector< std::vector< void ∗ > > ∗ **getKeyLocations** ()
- void ∗ **getUKeys** ()
- void **setUKeys** (void ∗uk)
- void ∗ **getKeyMap** ()
- void **setKeyMap** (void ∗km)
- void **writeToFile** (void ∗path, size_t procId, void ∗sufix)

Protected Member Functions

- K ∗ **distributeOwnership** (fastComm ∗comm, K ∗uKeys, size_t cSize)
- void **sendPartKeyCount** (fastComm ∗comm)
- std::unordered_map< K, size_t > **recvPartKeyMaxCount** (fastComm ∗comm, std::unordered_map< K, std::pair< size_t, std::deque< int >> > &keyPPMaxCount)
- std::unordered_map< K, size_t > **recvPartKeyCount** (fastComm ∗comm)
- std::unordered_map< K, size_t > **distributedMaxKeyCount** (fastComm ∗comm, std::unordered_map< K, std::pair< size_t, std::deque< int >> > &keyPPMaxCount)
- bool **EDBKsendDataAsync** (fastComm ∗comm, int owner, K &key, T &data, std::vector< size_t > &data↩ Size)
- bool **sendPending** (fastComm ∗comm, std::vector< std::deque< std::pair< K, T > > > &pendingSend, std::vector< size_t > &dataSize)
- void **flushDataSend** (fastComm ∗comm, std::vector< size_t > &dataSize)
- bool **EDBKSendData** (fastComm ∗comm, std::vector< size_t > &dataSize)
- bool **EDBKSendDataHashed** (fastComm ∗comm, size_t &pos, std::vector< bool > &deleted, std::vector< size_t > &dataSize, std::deque< std::pair< K, T > > &recvData, std::vector< std::deque< std::pair< K, T > > > &pendingSend, bool &dirty)
- bool **EDBKRecvData** (fastComm ∗comm, size_t &pos, size_t &posLimit, std::vector< bool > &deleted, std↩ ::deque< std::pair< K, T > > &recvData, int &peersFinised, bool &dirty)
- void **EDBKFinishDataInsert** (std::vector< bool > &deleted, std::deque< std::pair< K, T > > &recvData, size_t &pos)
- void **EDBKShrinkData** (std::vector< bool > &deleted, size_t &pos)
- void **findMyKeys** (int numProcs, int Id)
- void **findMyKeysByHash** (int numProcs)

Protected Attributes

- indexedFddStorage< K, T > ∗ **localData**
- std::shared_ptr< std::vector< K > > **uKeys**
- std::shared_ptr< std::unordered_map< K, int > > **keyMap**
- std::vector< std::vector< void ∗ > > **keyLocations**
- bool **groupedByKey**
- bool **groupedByHash**

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

- /home/mtcs/pesquisa/faster/faster.git/src/include/_workerIFdd.h
- /home/mtcs/pesquisa/faster/faster.git/src/libfaster/workerIFddCore.cpp

Index