CUDADClusterer

0.1

Generated by Doxygen 1.8.9.1

Wed Apr 29 2015 14:53:52

Contents

Chapter 1

CUDADClusterer

A fast implementation of the density cluster algorithm in CUDA of biological datasets.

CUDADClusterer uses the .xtc Groomacs file extension.

Requirements

- CMake 3.2.1
- GCC 4.9.2
- CUDA 7.0
- Boost 1.57

OBS: These requirements were tested and proved to work. Feel free to test older versions of the requirements

TODO

- [x] .xtc file parser [IMP]
- [ ] CUDA clusterer kernel
- [ ] Dimentionality reduction
- [ ] Data Visualizer

labels:

- [DON] - Done
- [IMP] - To be improoved
- [BUG] - Buggy and experimental

Credits

Author: Tiago LOBATO GIMENES: ∗tlgimenes.com∗

Contributors: Maybe you !

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 adjacency_graph Class Reference

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

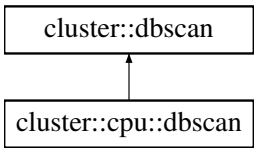
- knn/adjacency_graph.hpp

5.2 cluster::cpu::dbscan Class Reference

Class for applying DBSCAN clustering algorithm to data.

`#include <dbscan_cpu.hpp>`

Inheritance diagram for cluster::cpu::dbscan:



Public Member Functions

- dbscan (std::shared_ptr< const std::vector< float >> data, const float eps, const int min_pts, const int dim, metric::cpu::metric_f metric=metric::cpu::euclidean)

  Constructor of the clusterer.

Protected Attributes

- tree::cpu::vp_tree _tree

  VP-tree for the knn search.

5.2.1 Detailed Description

Class for applying DBSCAN clustering algorithm to data.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 cluster::cpu::dbscan::dbscan (std::shared_ptr< const std::vector< float >> *data*, const float *eps*, const int *min_pts*, const int *dim*, metric::cpu::metric_f *metric* = `metric::cpu::euclidean`) [inline]

Constructor of the clusterer.

Parameters

data	Data array
eps	Epsilon distance parameter of the DBSCAN algorithm
min_pts	Minimal number of points for the DBSCAN algorithm
dim	Dimention of the data
metric	Metric function to use for the DBSCAN algorithm

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

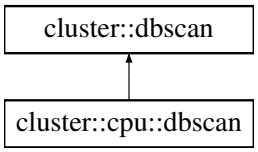
- clusterer/dbscan_cpu.hpp

5.3 cluster::dbscan Class Reference

Base class for applying DBSCAN clustering algorithm to data.

`#include <dbscan.hpp>`

Inheritance diagram for cluster::dbscan:



Public Member Functions

- dbscan (std::shared_ptr< const std::vector< float >> data, const float eps, const int min_pts, const int dim)

  Constructor of the clusterer.
- float & eps ()

  Set epsilon value for the DBSCAN algorithm.
- int & min_pts ()

  Set minimun points for the DBSCAN algorithm.
- const float & eps () const

  Get epsilon value of the DBSCAN algorithm.
- const int & min_pts () const

  Get minimun points of the DBSCAN algorithm.

Protected Attributes

- float _eps
- int _min_pts
- int _dim
- std::shared_ptr< const std::vector< float > > _data

5.3.1 Detailed Description

Base class for applying DBSCAN clustering algorithm to data.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 cluster::dbscan::dbscan (std::shared_ptr< const std::vector< float >> *data*, const float *eps*, const int *min_pts*, const int *dim*) [inline]

Constructor of the clusterer.

Parameters

data	Data array
eps	Epsilon distance parameter of the DBSCAN algorithm
min_pts	Minimal number of points for the DBSCAN algorithm
dim	Dimention of the data

5.3.3 Member Data Documentation

5.3.3.1 std::shared_ptr<const std::vector<float> > cluster::dbscan::_data [protected]

Vector containing data

5.3.3.2 int cluster::dbscan::_dim [protected]

Data's dimention

5.3.3.3 float cluster::dbscan::_eps [protected]

Epsilon parameters

5.3.3.4 int cluster::dbscan::_min_pts [protected]

Minimal number of points Parameters

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

- clusterer/dbscan.hpp

5.4 key_value< K, V > Class Template Reference

Key-Value class implementation.

`#include <types.hpp>`

Public Member Functions

- key_value (K k, V v)
  Constructor.
- key_value ()
  Constructs new element with default key and value.
- const V & val () const
  Get value.
- const K & key () const
  Get key.
- V & val ()
  Set value.
- K & key ()
  Set key.
- const key_value & operator= (const key_value &other)
  Copies key and value from one object to this.
- bool operator< (const key_value &other) const
  The comparision is made using the value and not the key.

5.4.1 Detailed Description

template<typename K, typename V>class key_value< K, V >

Key-Value class implementation.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 template<typename K , typename V > key_value< K, V >::key_value (K *k*, V *v*) [inline]

Constructor.

Parameters

k	Key
v	Value

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

- utils/types.hpp

5.5 console::modifier Class Reference

`#include <color.hpp>`

Public Member Functions

- modifier (enum ccode code)
  color code
- enum ccode & code () const
  Get the color code.

Friends

- std::ostream & operator<< (std::ostream &os, const modifier &mod)
  operator for using this class in the standard c++ output

5.5.1 Detailed Description

This class should be used in the following way: modifier red(FG_RED); std::cout << red << "this text is red" << std::endl;

5.5.2 Constructor & Destructor Documentation

5.5.2.1 console::modifier::modifier (enum ccode *code*) [inline]

color code

Constructor for the color modifier

Parameters

code	color code for the modifier

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

- utils/color.hpp

5.6 console::parser Class Reference

Class for parsing the Command Line Interface.

`#include <parser.hpp>`

Static Public Member Functions

- static void parse (int argc, const char **argv)
  Parses the command line interface.
- static void add_argument (const std::string &short_form, const std::string &help)
  Adds arguments to be parsed.
- static const std::string get (const std::string &arg, bool required)
  Gets the value of the argument.

Static Protected Member Functions

- static void print_help ()
  Prints the help in CLI.

5.6.1 Detailed Description

Class for parsing the Command Line Interface.

5.6.2 Member Function Documentation

5.6.2.1 void console::parser::add_argument (const std::string & *short_form*, const std::string & *help*) [inline], [static]

Adds arguments to be parsed.

Parameters

short_form	short form of the parameter. Ex: "-t", "-a", etc
help	help for the parameter

5.6.2.2 const std::string console::parser::get (const std::string & *arg*, bool *required*) [inline], [static]

Gets the value of the argument.

Parameters

arg	short form of the parameter. Ex: "-t", "-a", etc
required	true if parameter is required, false otherwise

Returns

string containing the value passed in CLI. If parameter is not required the DEFAULT_STRING will be returned

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

- parser/parser.hpp

5.7 primitive< T, N > Class Template Reference

Implementation of general N-dimentional type.

`#include <types.hpp>`

Public Member Functions

- primitive (T *data)
  Constructor of type.
- primitive (...)
  Constructor that allows intation of each element separatly.
- operator T () const
  Cast operator.
- const T & operator[ ] (int n) const
  Array like operator.
- primitive< T, N > operator+ (const primitive< T, N > &other) const
  Element-wise operator plus.
- primitive< T, N > operator- (const primitive< T, N > &other) const
  Element-wise operator minus.
- primitive< T, N > operator* (const primitive< T, N > &other) const
  Element-wise operator times.
- const bool operator< (const primitive< T, N > &other) const
  Element-wise operator smaller than.
- T & operator[ ] (int n)
  Array like operator.
- primitive< T, N > & operator= (const primitive< T, N > &other)
  Element-wise assignement operator.
- template<>
  primitive (...)
- template<>
  primitive (...)

Protected Attributes

- T _data [N]
  Where data is stored.

5.7.1 Detailed Description

template<typename T, int N>class primitive< T, N >

Implementation of general N-dimentional type.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 template<typename T , int N> primitive< T, N >::primitive (T * *data*) [inline]

Constructor of type.

Parameters

data	Data to be stored in the "variable"

5.7.3 Member Function Documentation

5.7.3.1 template<typename T , int N> const bool primitive< T, N >::operator< (const primitive< T, N > & *other*) const [inline]

Element-wise operator smaller than.

Returns

True if first element of this is smaller than other. If equal continue the search in other items of the data array

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

- utils/types.hpp

5.8 reader_xtc Class Reference

Class for reading trajlist and .xtc files.

`#include <reader_xtc.hpp>`

Static Public Member Functions

- static void read_list (const std::string &home, const std::string &trajlist, std::vector< float > &data, int &n_↩ atoms)

  Reads all trajectories specified in the trajlist file.

Static Protected Member Functions

- static void read_trajfile (const std::string &trajfile, std::vector< float > &data, int &n_atoms, int &n_samples)

  Reads a trajectory file.
- static void get_framefile_list (std::vector< std::string > &framefile_list, const std::string &home, const std↩ ::string &trajlist)

  *finds *.xtc files from trajlist*
- static bool is_ext_supported (const std::string &file_name)

  Checks if the extension of file "file_name" is supported or not by this class.

5.8.1 Detailed Description

Class for reading trajlist and .xtc files.

5.8.2 Member Function Documentation

5.8.2.1 void reader_xtc::get_framefile_list (std::vector< std::string > & *framefile_list,* const std::string & *home,* const std::string & *trajlist*) [inline], [static], [protected]

finds ∗.xtc files from trajlist

Given a trajlist path, this function will recursively try to find the (∗.xtc) files and insert it on the framefile_list. The complete path for the file must be given by home/trajlist

5.8.2.2 bool reader_xtc::is_ext_supported (const std::string & *file_name*) [inline], [static], [protected]

Checks if the extension of file "file_name" is supported or not by this class.

Returns

true if file extension is supported, false otherwise

5.8.2.3 void reader_xtc::read_list (const std::string & *home,* const std::string & *trajlist,* std::vector< float > & *data,* int & *n_atoms*) [inline], [static]

Reads all trajectories specified in the trajlist file.

A trajlist file can contains the relative paths to the .xtc files or name of files that contains relative paths to the .xtc files. The absolute path is always done in the following way : path = home/trajlist

5.8.2.4 void reader_xtc::read_trajfile (const std::string & *trajfile,* std::vector< float > & *data,* int & *n_atoms,* int & *n_samples*) [inline], [static], [protected]

Reads a trajectory file.

It can be ∗.xtc or any other file defined in _supported_ext vector and appends the new trajectories in the data vector

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

- utils/reader_xtc.hpp

5.9 tree::vp_node_t Struct Reference

vp-tree node for a linearized tree in an array

`#include <vp_tree.hpp>`

Public Member Functions

- vp_node_t (int k=0, float d=0.0f, int lc=0, int rc=0)

  Creates a new node.

Public Attributes

- int _key
- float _d
- int _lc
- int _rc

5.9.1 Detailed Description

vp-tree node for a linearized tree in an array

5.9.2 Constructor & Destructor Documentation

5.9.2.1 tree::vp_node_t::vp_node_t (int *k* = 0, float *d* = 0.0f, int *lc* = 0, int *rc* = 0) [inline]

Creates a new node.

Parameters

k	index in data vector
d	distance threshold if it's an internal node
lc	index in tree vector of left child
rc	index in tree vector or right child

5.9.3 Member Data Documentation

5.9.3.1 float tree::vp_node_t::_d

distance threshold

5.9.3.2 int tree::vp_node_t::_key

index in _data

5.9.3.3 int tree::vp_node_t::_lc

left child index

5.9.3.4 int tree::vp_node_t::_rc

right child index

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

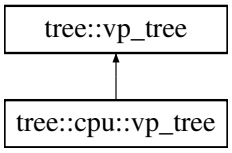
- knn/vp_tree.hpp

5.10 tree::vp_tree Class Reference

Base class for creating vp-tree.

`#include <vp_tree.hpp>`

Inheritance diagram for tree::vp_tree:



Public Member Functions

- vp_tree ()

  Constructs a new empty tree.
- vp_tree (std::shared_ptr< const std::vector< float >> data, int dim)

  Constructs a tree with the new data and dimention.
- vp_tree (const vp_tree &other)

  Constructs new tree based on existing tree.
- void fit (std::shared_ptr< const std::vector< float >> data, int dim)

  Constructs a new tree with the given data and dimention.
- const std::shared_ptr< const std::vector< float > > & data () const

  Get data vector.
- int dim () const

  Get dimention.
- std::shared_ptr< const std::vector< float > > & data ()

  Set data vector.
- int & dim ()

  Set dimention.

Protected Attributes

- std::shared_ptr< const std::vector< float > > _data
- int _dim

5.10.1 Detailed Description

Base class for creating vp-tree.

5.10.2 Member Function Documentation

5.10.2.1 std::shared_ptr< const std::vector< float > > & tree::vp_tree::data ( ) [inline]

Set data vector.

Use this function as your own responsability

5.10.2.2 int& tree::vp_tree::dim ( ) [inline]

Set dimention.

Use this function as your own responsability

5.10.3 Member Data Documentation

5.10.3.1 std::shared_ptr<const std::vector<float> > tree::vp_tree::_data [protected]

data

5.10.3.2 int tree::vp_tree::_dim [protected]

Dimention of data contained in data

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

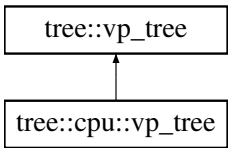
- knn/vp_tree.hpp

5.11 tree::cpu::vp_tree Class Reference

Base class for creating vp-tree.

`#include <vp_tree_cpu.hpp>`

Inheritance diagram for tree::cpu::vp_tree:



Public Member Functions

- vp_tree (std::shared_ptr< const std::vector< float >> data, int dim, metric::cpu::metric_f metric=metric::cpu::euclidean)

  Constructs a new vp-tree.
- void knn (int query, float delta, std::vector< int > &id)

  Performs the knn search and returns all elements within the radius delta of the query.
- void knn (int query, int k, std::vector< int > &id)

  Performs the knn search and returns k elements closest to the query.
- void brute_knn (int query, float delta, std::vector< int > &id)

  Same function as knn but using the brute force algorithm.
- void brute_knn (int query, int k, std::vector< int > &id)

  Same function as knn but using the brute force algorithm.
- int find (int query) const

  Finds the query in _data vector and returns its index in the tree.
- bool belongs (int query) const

  Brute force algorithm to check wheater a query belongs to the tree or not.
- const std::vector< tree::vp_node > & t () const

  Get the tree.

Protected Member Functions

- void dist2 (int p, std::vector< ifloat > &index_set) const

  Evaluates the distance between p and the set index_set setting each float in index_set.

- int select_vp (const std::vector< ifloat > &index_set) const

  Select among elements in index_set the vantage point to split the Tree.
- float split (std::vector< ifloat > &index_set, std::vector< ifloat > &lc, std::vector< ifloat > &rc) const

  Splits the index_set in two sub sets such that lc and rc are aproximately the same size.
- int make_vp_tree (std::vector< ifloat > &index_set)

  Constructs populating the _tree vector a vp_tree corresponding to the data stored in the _data vector and specified in the index_set.
- void print_tree ()

  prints the whole tree
- void print_range (int b, int e)

  prints a range of the tree
- void check_tree ()

Protected Attributes

- std::vector< tree::vp_node > _tree

  Tree structure is stored here.
- metric::cpu::metric_f _metric

  Pointer to the metric function.

5.11.1 Detailed Description

Base class for creating vp-tree.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 tree::cpu::vp_tree::vp_tree (std::shared_ptr< const std::vector< float >> *data*, int *dim*, metric::cpu::metric_f *metric* = `metric::cpu::euclidean`) [inline]

Constructs a new vp-tree.

Parameters

data	Data for creating vp-tree
dim	Dimention of the data
metric	metric function used in the vp-tree

5.11.3 Member Function Documentation

5.11.3.1 void tree::cpu::vp_tree::check_tree ( ) [inline], [protected]

checks the tree for errors

5.11.3.2 void tree::cpu::vp_tree::knn (int *query*, float *delta*, std::vector< int > & *id*) [inline]

Performs the knn search and returns all elements within the radius delta of the query.

Parameters

query	index of query in the data
delta	maximum distance exclusive to search
id	ids of elements in data closer to query than delta

5.11.3.3 int tree::cpu::vp_tree::select_vp (const std::vector< ifloat > & *index_set*) const [inline], [protected]

Select among elements in index_set the vantage point to split the Tree.

Basic implementation still. Don't see the point for a more complicated code

5.11.3.4 float tree::cpu::vp_tree::split (std::vector< ifloat > & *index_set,* std::vector< ifloat > & *lc,* std::vector< ifloat > & *rc*) const [inline], [protected]

Splits the index_set in two sub sets such that lc and rc are aproximately the same size.

Returns

The distance used for splitting the two subsets

5.11.4 Member Data Documentation

5.11.4.1 metric::cpu::metric_f tree::cpu::vp_tree::_metric [protected]

Pointer to the metric function.

The metric function should return the squared distance between two elements in the data

5.11.4.2 std::vector<tree::vp_node> tree::cpu::vp_tree::_tree [protected]

Tree structure is stored here.

Vector containing the vp-tree representation of all the data stored in _data vector of super class

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

- knn/vp_tree_cpu.hpp

Chapter 6

File Documentation

6.1 clusterer/dbscan.hpp File Reference

Definition of the class base for the dbscan clusterer.

```
#include "vp_tree.hpp"
```

Classes

- class cluster::dbscan

  Base class for applying DBSCAN clustering algorithm to data.

6.1.1 Detailed Description

Definition of the class base for the dbscan clusterer.

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-20 17:54

This file contains the implementation of the base class for the dbscan clustering algorithm

6.2 clusterer/dbscan_cpu.hpp File Reference

Definition of the class specialization of dbscan for CPU.

```
#include "dbscan.hpp"
#include "metrics.hpp"
#include "vp_tree_cpu.hpp"
```

Classes

- class cluster::cpu::dbscan

  Class for applying DBSCAN clustering algorithm to data.

6.2.1 Detailed Description

Definition of the class specialization of dbscan for CPU.

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-28 13:53

This file contains the implementation of the class specialization for the dbscan clustering algorithm on CPU

6.3 knn/metrics.hpp File Reference

metric definitions

Typedefs

- using **metric::cpu::metric_f** = float(∗)(int, int, const std::vector< float > &, int)

  Definition of the distance function used for defining the metric space.

Functions

- float **metric::cpu::euclidean** (int a, int b, const std::vector< float > &data, int dim)

  Implementation of the euclidean metric.

6.3.1 Detailed Description

metric definitions

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-22 22:35

This file contains implementation of metric functions

6.4 knn/vp_tree.hpp File Reference

vp_tree base class implementation

```
#include <vector>
#include <memory>
#include "error.hpp"
```

Classes

- struct tree::vp_node_t
  vp-tree node for a linearized tree in an array
- class tree::vp_tree
  Base class for creating vp-tree.

Typedefs

- using **tree::vp_node** = struct vp_node_t
  vp-tree node typedef

Functions

- std::ostream & operator<< (std::ostream &in, const tree::vp_node &n)
  function for printing vp_node struct

6.4.1 Detailed Description

vp_tree base class implementation

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-20 18:09

This file contains base class implementation of vp_tree

6.5 knn/vp_tree_cpu.hpp File Reference

vp_tree cpu class especification

```
#include <algorithm>
#include "vp_tree.hpp"
#include "metrics.hpp"
#include "types.hpp"
#include "time.hpp"
```

Classes

- class tree::cpu::vp_tree
  Base class for creating vp-tree.

Macros

- #define EPSILON 1e-6
  Float comparision threshold.
- #define LEAF -1
- #define ROOT -2
- #define UNDEF -3

6.5.1 Detailed Description

vp_tree cpu class especification

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-20 21:58

This file contains the cpu specialization of vp_tree class

6.5.2 Macro Definition Documentation

6.5.2.1 #define EPSILON 1e-6

Float comparision threshold.

If the modulus of the difference between two floats is smaller than this value the float values are considered to be equal

6.5.2.2 #define LEAF -1

Leaf descriptor

6.5.2.3 #define ROOT -2

Root descriptor

6.5.2.4 #define UNDEF -3

Undefined node descriptor

6.6 parser/parser.hpp File Reference

CLI parser.

```
#include <string>
#include <cstring>
#include <utility>
#include <map>
#include "error.hpp"
```

Classes

- class console::parser

  Class for parsing the Command Line Interface.

Macros

- #define DEFAULT_STRING "0"
  Default string to be returned in case of failure.

6.6.1 Detailed Description

CLI parser.

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-03-30 15:05

This file contains implementation of a simple CLI parser

6.7 utils/color.hpp File Reference

color CLI modifier

```
#include <ostream>
```

Classes

- class console::modifier

Enumerations

- enum **ccode** {
  FG_RED = 31, **FG_GREEN** = 32, **FG_YELLOW** = 33, **FG_BLUE** = 34,
  FG_MAGENTA = 35, **FG_CYAN** = 36, **FG_L_GRAY** = 37, **FG_D_GRAY** = 90,
  FG_DEFAULT = 39, **BG_RED** = 41, **BG_GREEN** = 42, **BG_YELLOW** = 43,
  BG_BLUE = 44, **BG_MAGENTA** = 45, **BG_CYAN** = 46, **BG_L_GRAY** = 47,
  BG_D_GRAY = 100, **BG_DEFAULT** = 49, **SET_BOLD** = 1, **SET_DIM** = 2,
  RESET_BOLD = 21, **RESET_DIM** = 22, **DEFAULT** = 0 }
  Color code for the modifier.

6.7.1 Detailed Description

color CLI modifier

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-27 13:43

This file contains a class implementation that allows to change colors in the terminal. If the terminal does not support colors, some strange character chains will be printed

6.8 utils/error.hpp File Reference

error repporting interface

```
#include <string>
#include <chrono>
#include <iostream>
#include "color.hpp"
```

Macros

- #define DBG_MESSAGE(str) __message(str, console::FG_D_GRAY)

  Message to be displayed in debug only mode.
- #define FATAL_ERROR(str) __error(str, __FILE__, __LINE__)

  Fatal errors.
- #define WARNING_ERROR(str) __warning(str, __FILE__, __LINE__)

  Warning.
- #define ASSERT_FATAL_ERROR(boolean, str) (void)((boolean) || (__error(str, __FILE__, __LINE__),0))

  Asserts with fatal error message.
- #define ASSERT_WARNING_ERROR(boolean, str) (void)((boolean) || (__warning(str, __FILE__, __LINE↩ __),0))

  Asserts with warning message.
- #define CUDA_SAFE(code) ASSERT_FATAL_ERROR(code == cudaSuccess, cudaGetErrorString(cuda↩ GetLastError()))

  This macro searches for errors in cuda functions execution.

6.8.1 Detailed Description

error repporting interface

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-02-02 11:13

This file contains the implementation of a simple error repporting interface

6.8.2 Macro Definition Documentation

6.8.2.1 #define CUDA_SAFE(*code*) ASSERT_FATAL_ERROR(code == cudaSuccess, cudaGetErrorString(cudaGetLastError()))

This macro searches for errors in cuda functions execution.

One should use always this macro in each cuda function call

6.8.2.2 #define FATAL_ERROR(*str*) __error(str, __FILE__, __LINE__)

Fatal errors.

This macro writes the error message in str and exits

6.8.2.3 #define WARNING_ERROR(*str*) __warning(str, __FILE__, __LINE__)

Warning.

This macro writes the warning message in str and continues

6.9 utils/reader_xtc.hpp File Reference

.xtc file reader

```
#include <vector>
#include <string>
#include <sstream>
#include <fstream>
#include <boost/filesystem.hpp>
#include "error.hpp"
#include "xdrfile/xdrfile.h"
#include "xdrfile/xdrfile_xtc.h"
```

Classes

- class reader_xtc

  Class for reading trajlist and .xtc files.

Macros

- #define MAX_FILE_NAME_LENGTH 128

  Maximun file name lenght.

6.9.1 Detailed Description

.xtc file reader

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-03-30 18:49

This file contains the implementation of a class for reading .xtc files with a trajectory list

6.10 utils/time.hpp File Reference

functions and tools for mesuring the execution time of a given code

```
#include <string>
#include <iostream>
#include <vector>
#include "color.hpp"
#include "error.hpp"
```

Macros

- #define CPP99 199711L
- #define TIME_BETWEEN(code) WARNING_ERROR("TIME_BETWEEN needs at least C++11 compliant compiler")

  Measures the execution time between the code given.

6.10.1 Detailed Description

functions and tools for mesuring the execution time of a given code

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-21 18:36

This file contains the implementation of functions for mesuring the execution time between code. It's necessary to have C++11 support

6.10.2 Macro Definition Documentation

6.10.2.1 #define CPP99 199711L

C++99 version number

6.11 utils/types.hpp File Reference

Simple type definitions.

```
#include <cstdarg>
#include "error.hpp"
```

Classes

- class primitive< T, N >

  Implementation of general N-dimentional type.
- class key_value< K, V >

  Key-Value class implementation.

Typedefs

- template<int N>
  using floatn = primitive< float, N >
- template<int N>
  using intn = primitive< int, N >
- using float4 = floatn< 4 >
- using ifloat = key_value< int, float >

6.11.1 Detailed Description

Simple type definitions.

Author

Tiago LOBATO GIMENES (tlgimenes@gmail.com)

Date

2015-04-21 15:10

This file contains the implementation and definition of simple types imeplemented on CUDA and OpenCL

6.11.2 Typedef Documentation

6.11.2.1 using float4 = floatn<4>

4-Dimentional float definition

6.11.2.2 template<int N> using floatn = primitive<float, N>

N-Dimentional float definition

6.11.2.3 using ifloat = key_value<int, float>

Indexed float (ifloat) definition

6.11.2.4 template<int N> using intn = primitive<int, N>

N-Dimentional int definition

Index