Data Mining - Android - OpenCL

Generated by Doxygen 1.8.14

Contents

Chapter 1

AGPUDM

Introduction

This project allows to address the GPU on Android devices using the OpenCL framework. In contrast to other projects the OpenCL library of the device is loaded only at runtime. There is no need to include it during the build process.

Two data mining algorithms (DBSCAN and Kmeans) have been implemented with two different programming languages and different programming paradigms (single-threaded, multi-threaded, task/data parallelism (GPU)).

Docs

There is a `html` and `pdf` doxygen documentation available for this project. As of 9/15/2021 only the C part (except dbscan_c.c) has been included in this documentation. For the Java part, only the automatically generated doxygen documentation is available. Additional documentation will be available soon.

Installation

Clone this repository (either with "git clone" or downloading and extracting the zip-file). Import the project to AndroidStudio. Attach your Android device and enable developper options (see manual of your device). Build and run this project on your device. It should run out of the box.

Setup

This app has only one activity. The data mining jobs are executed in a deferred manner in the background. If the devices is rebooted, the calculations will resume automatically. When the user launches the app, the main activity tries to connect to the background jobs and tries to read out the status information. If there are no background jobs, the user can submit a new background job. If there are already background jobs, the status is displayed and the user can cancel the jobs.

The app tries to find the OpenCL library on the device automatically. If an OpenCL library is found and can be loaded, some information is displayed. If not, the user can enter the path to the OpenCL library on the device manually and try to load it manually.

How to set parameters

The parameters for the data mining jobs have to be set in the file app/src/main/res/values/values.xml. The following attributes can be set:

- **mode** (string):
  - "dynamic" multiple test are made with built in values. The attributes 'clusterno', 'clustersize' and 'features' must be set correctly but will not be used.
  - "fixed" tests are made with the attributes specified in this file
- **passes** (integer) Number of passes that should be made
- **threads** (integer) Number of threads that should be used for multithreaded implementations. Zero means the maximum number of available cores.
- **export** (boolean) "true" export results, "false" do not export results
- **append** (boolean) "true" append to old results if available, "false" delete old results
- **resultfilename** (string) Name of the csv-file in which to store the results. **DO NOT PREPEND A PATH!** The correct path will be prepended automatically. In the version for larger screen sizes the full path and name are shown in the info box.
- **log** (boolean) "true" log information, "false" do not log information (there is just one log-level)
- **logfilename** (string) Name of the text-file in which to store the results. **DO NOT PREPEND A PATH!** The correct path will be prepended automatically. In the version for larger screen sizes the full path and name are shown in the info box.
- **kmeanseps** Maximum cluster center displacement. If the sum of the absolute values of the cluster displacements drops below this threshold, the algorithm terminates.
- **dbscaneps** Search radius for DBSCAN (0=sqrt(features))
- **dbscanneigh** Minimum number of neighbours within the serach radius (0=10∗features)
- **clusterno** Number of clusters to generate randomly. For Kmeans also the number of clusters to search for.
- **clustersize** Size of the clusters (equal size for all clusters).
- **features** Number of features for each data item.

In a future version an additional activity, that will allow to set the attributes on the device during runtime, will be added to this project.

Results

The results are stored in csv-format on the device. The path of the app is used (should be something like sdcard/↩ Android/data/com.example.dmocl/files). Log information is also stored in this path. This path is set automatically - do not prepend the path in the values.xml file.

The result file has 21 columns separated by ';'. The first four contain the parameters of the test (cores;size;cluster;features). The next five show the wall clock time for different implementations of the kmeans algorithm (Java, C, C+GPU, multithreaded Java, multithreaded C). The following five columns hold the wall clock times for the DBSCAN algorithm (again Java, C, C+GPU, multithreaded Java, multithreaded C). The next columns contain the exclusive time for the GPU and the multithreaded implementations. 'Exclusive time' means the time needed for the execution of the entire algorithm minus the time needed for the setup of the GPU or the threads. Three values are collected by Kmeans and another three by DBSCAN. The last column is zero if the output of all the implementations of the DBSCAN algorithm was EXACTLY equal. For Kmeans a comparison is not possible because each implementation selects the cluster centers randomly at the beginning.

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

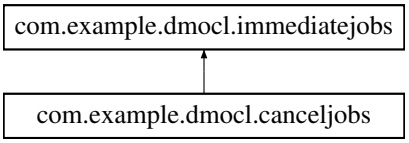
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 com.example.dmocl.canceljobs Class Reference

Inheritance diagram for com.example.dmocl.canceljobs:



Public Member Functions

- **canceljobs** (Handler resultHandler, Executor executor, Context context, TextView jobinfo, Button startbutton)
- void **startcanceljobs** (final RepositoryCallback< jobschedresponse > callback)

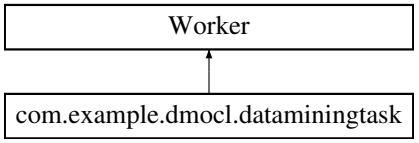
Additional Inherited Members

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/canceljobs.java

5.2 com.example.dmocl.dataminingtask Class Reference

Inheritance diagram for com.example.dmocl.dataminingtask:



Public Member Functions

- **dataminingtask** (@NonNull Context context, @NonNull WorkerParameters params)
- void **onStopped** ()
- Result **doWork** ()

Static Public Member Functions

- static final String **compileprogressoutput** (String fn, int z, int meth, int cores, int clusterno, short[ ] b, double wct)

Static Public Attributes

- static final String [ ] **prependnames** = {"Java", "C", "C+GPU", "Java+Threads", "C+Threads"}

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/dataminingtask.java

5.3 com.example.dmocl.dbscan Class Reference

Static Public Member Functions

- static void **dbscanabort** ()
- static void **dbscanresume** ()
- static native short **dbscan_c** (short[ ] b, float[ ] data, float eps, int kk, int features)
- static native short **dbscan_c_gpu** (short[ ] b, float[ ] data, float eps, int kk, int features, long[ ] e)
- static native short **dbscan_c_phtreads** (short[ ] b, float[ ] data, float eps, int kk, int features, int cores, long[ ] e)
- static short **dbscan_st** (short[ ] b, float[ ] data, float eps, int kk, int features)
- static short **dbscan_threads** (short[ ] b, float[ ] data, float eps, int kk, int features, int cores, long[ ] ej) throws InterruptedException

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/dbscan.java

5.4 dbscan_pt Struct Reference

Public Attributes

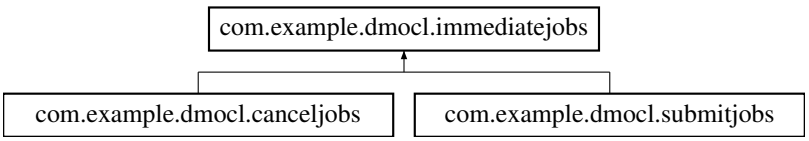
- unsigned short int **status**
- int **num**
- unsigned short ∗ **b**
- float ∗ **data**
- int **blen**
- float **eps**
- int **kk**
- int **features**
- int **start**
- int **len**
- pthread_t **thread1**
- pthread_t **thread2**
- sem_t **sem1**
- sem_t **semret1**
- sem_t **sem2**
- sem_t **semret2**
- pthread_mutex_t **MUTEX_var1**
- volatile int **cmpto1**
- volatile int **itemcounter1**
- pthread_mutex_t **MUTEX_var2**
- volatile int **cmpto2**
- volatile int **itemcounter2**
- pthread_mutex_t **MUTEX_fertig1**
- volatile unsigned char **fertig1**
- pthread_mutex_t **MUTEX_fertig2**
- volatile unsigned char **fertig2**

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/dbscan_c.c

5.5 com.example.dmocl.immediatejobs Class Reference

Inheritance diagram for com.example.dmocl.immediatejobs:



Classes

- class **jobschedresponse**

Protected Member Functions

- void **notifyResult** (final Result< canceljobs.jobschedresponse > result, final RepositoryCallback< canceljobs.jobschedresponse > callback, final Handler resultHandler)

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/immediatejobs.java

5.6 com.example.dmocl.kmeans Class Reference

Static Public Member Functions

- static void **kmabort** ()
- static void **kmresume** ()
- static native short **kmeans_c** (short[ ] b, float[ ] data, float eps, int cluno, int features)
- static native short **kmeans_c_gpu** (short[ ] b, float[ ] data, float eps, int cluno, int features, long[ ] e)
- static native short **kmeans_c_phtreads** (short[ ] b, float[ ] data, float eps, int cluno, int features, int cores, long[ ] e)
- static short **kmeans_st** (short[ ] b, float[ ] data, float eps, int cluno, int features)
- static short **kmeans_threads** (short[ ] b, float[ ] data, float eps, int cluno, int features, int cores, long[ ] ej) throws InterruptedException

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/kmeans.java

5.7 kmeans_pt Struct Reference

Parameters for the kmeans thread.

Public Attributes

- unsigned short int status

  (const) status (needed only for setup and destruction)
- int num

  (const) number of thread
- unsigned short ∗ b

  (out) number of closest cluster center
- float ∗ data

  (const) input data
- volatile float ∗ clucent

  (in) cluster centers
- int blen

  (const) number of data items
- int cluno

(const) number of clusters

- int features

  (const) number of features per data item

- int start

  (const) first data item

- int len

  (const) last data item

- volatile unsigned char fertig

  (in) 1=quit loop

- pthread_t thread

  (const) reference to the thread

- sem_t sem

  (const) semaphore used for start

- sem_t semret

  (const) semaphore for notification that results are ready

5.7.1 Detailed Description

Parameters for the kmeans thread.

This struct holds the parameters for each kmeans thread. (in) means parameters that are NOT changed by the thread but may be changed by the method that submits the job. (const) means that the value is never changed after setup. (out) attributes are changed by the thread. The submitting method must not write access any (in) or (out) field while the calculations are running. The submitting method may read access all (in) while the calculations are running.

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/kmeans_c.c

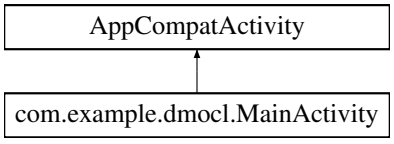
5.8 com.example.dmocl.LinkToFile Class Reference

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/LinkToFile.java

5.9 com.example.dmocl.MainActivity Class Reference

Inheritance diagram for com.example.dmocl.MainActivity:



Classes

- class **WorkManagerNoInformationException**

Protected Member Functions

- synchronized boolean **tryLoadGPU** (String gpupath)
- void **onResume** ()
- void **onCreate** (Bundle savedInstanceState)
- void **onDestroy** ()

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/MainActivity.java

5.10 com.example.dmocl.oclwrap.oclinforet Class Reference

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/oclwrap.java

5.11 com.example.dmocl.oclwrap Class Reference

Classes

- class oclinforet

Public Member Functions

- native int loadOpenCL (String oclLibpath)
- native void unloadOpenCL ()
- native int AndrCLGetPlatformCnt ()
- native int **AndrCLGetDeviceCnt** (int platf)
- native oclinforet **AndrCLgetDeviceName** (int platf, int dev)

Static Public Member Functions

- static native int **isCLang** ()
- static native int **getCLmaj** ()
- static native int **getCLmin** ()
- static native int **getCLpatch** ()
- static native int getArchitecture ()
- static oclwrap getOclWrapper ()

5.11.1 Detailed Description

A singleton class for the OpenCL wrapper.

Each class (or activity) that needs the use OpenCL can get a reference to the OpenCL runtime invoking the method *getOclWrapper*

The OpenCL runtime implemented has two abstraction layers: The first abstraction layer is the link between Java (Android) and a C-Wrapper. The communication between Java and C is done using JNI. This wrapper has to be written by the user. It can expose the entire set of OpenCL calls or implement an own API that exposes the functionality of more complex tasks that are carried out in C/OpenCL.

The second abstraction layer is the link between C and OpenCL. It is a thin wrapper that loads dynamically at runtime the local OpenCL library (libOpenCL.so) and exposes its functionality to the user. This layer must not be modified by the user. The user has to call once (before the very first call to an OpenCL function the function *loadOpenCL* that loads the libOpenCL.so library on the device. This library does not have to be present at compile time. After the last call to an OpenCL function, the function *unloadOpenCL* should be called. If the user wishes, on Android onStop-Event the OpenCL library can be unloaded (saving some memory). If one tries to call an OpenCL function WITHOUT having called 'loadOpenCL' a runtime error will occur. The best place to call 'loadOpenCL' would be Androids 'onCreate' method.

All methods of this class are fully thread-safe if the underlying JNI methods are thread-safe. The OpenCL library cannot be unloaded while there is still some calculation in progress. If one tries to call an OpenCL function AFTER the OpenCL library has been unloaded, the library will be reloaded automatically. The Java- and JNI part do not have to matter about synchronization issues. Synchronization is provided by the OpenCL wrapper library.

5.11.2 Member Function Documentation

5.11.2.1 AndrCLGetPlatformCnt()

```
native int com.example.dmocl.oclwrap.AndrCLGetPlatformCnt ( )
```

Returns the number of platforms available.

Remarks

If only number of platforms is relevant, this method is much faster than *AndrCLGetPlatformIDs*

Returns

OCLANDROID_ERROR or number of platforms found fully threadsafe

5.11.2.2 getArchitecture()

```
static native int com.example.dmocl.oclwrap.getArchitecture ( )  [static]
```

Returns the type of architecture.

Returns

The the architecture used (0=arm-v7, 1=arm-v8, 2=x86, 3=x86_64, -1=unknown) fully

5.11.2.3 getOclWrapper()

```
static oclwrap com.example.dmocl.oclwrap.getOclWrapper ( )  [inline], [static]
```

Returns a reference to the singleton.

Returns

The (single) instance of the OpenCL wrapper. Fully threadsafe.

5.11.2.4 loadOpenCL()

```
native int com.example.dmocl.oclwrap.loadOpenCL (
            String ocllibpath )
```

Loads the OpenCL library on the device. The library does not have to be present at compile time. Must be called once before any other call to an OpenCL function. Subsequent calls to this method have no effect (even if the library is currently not loaded). The path to the library can be set only at the call to this function and is immutable afterwards (you have to restart the app to change the path).

Parameters

ocllibpath	The path and name of the OpenCL-library on the device (e.g. "/system/vendor/lib/libOpenCL.so" for Mali graphics cards.

Returns

-1 = if the library could not be loaded, 1 = this method has already been called, 0 else; if the return value is greater or equal to zero, the library can be used. If the return value is negative, most probably the path to the OpenCL library on the device was wrong or there is no OpenCL shared library on the device. fully thread safe as long as underlying JNI function is thread-safe (synchronization is provided in library function).

5.11.2.5 unloadOpenCL()

```
native void com.example.dmocl.oclwrap.unloadOpenCL ( )
```

Unloads the OpenCL library. This method can be called also if Android's 'onStop' event occurs. Fully thread-safe as long as the underlying JNI implementation is thread save. The underlying library function provides synchronization methods.

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/oclwrap.java

5.12 rwlockwp Struct Reference

A struct thats holds all necessary components for the lock.

```
#include <rwlock_wp.h>
```

Public Attributes

- pthread_mutex_t g
  A mutex for the reader/writer lock.
- pthread_cond_t c
  A condition variable for the reader/writer lock.
- int num_writers_waiting
  Number of writers waiting.
- int num_reader_active
  Number of readers active.
- int writer_active
  Number of writers active.

5.12.1 Detailed Description

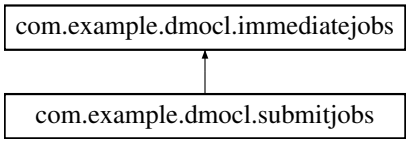
A struct thats holds all necessary components for the lock.

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/rwlock_wp.h

5.13 com.example.dmocl.submitjobs Class Reference

Inheritance diagram for com.example.dmocl.submitjobs:



Public Member Functions

- **submitjobs** (Handler resultHandler, Executor executor, Context context, TextView jobinfo, Button startbutton, String GPUpath, boolean doexport, boolean dolog, boolean appendresults, String logfn, String fn, String mode, int clusterno, int passes, int clusi, int features, String kmeanseps, String dbscaneps, int dbscanneigh, boolean GPUfound, int cores)
- void **startcalculations** ()
- void **startsubmitjobs** (final RepositoryCallback< jobschedresponse > callback)

Additional Inherited Members

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

- /home/robert/AndroidStudioProjects/DMGPU/app/src/main/java/com/example/dmocl/submitjobs.java

Chapter 6

File Documentation

6.1 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/AndroidOpenCL.h File Reference

Load/Unload method prototypes.

Functions

- int loadOpenCL (const char ∗c)

  Loads the OpenCL library of the Android device dynamically.
- void unloadOpenCL (void)

  Unloads the OpenCL library.

6.1.1 Detailed Description

Load/Unload method prototypes.

This headerfile contains two method definitions for loading and unloading the OpenCL shared library..

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Date

11.9.2021

6.1.2 Function Documentation

6.1.2.1 loadOpenCL()

```
int loadOpenCL (
            const char * c )
```

Loads the OpenCL library of the Android device dynamically.

Loads the OpenCL library dynamically. This function **MUST** be called exactly once before any other call to an OpenCL function. The function stores the path of the library. If the library has already been loaded, a call to this method will have no effect. Any call to an OpenCL function without prior call to this method will result in an error.

Parameters

c	(in) Pointer to the path and name of the OpenCL-library on the device (can be reused after the call)

Returns

OK: 0, library has already been loaded: -1, unable to load library: -2

Multithreading:

fully threadsafe

6.1.2.2 unloadOpenCL()

```
void unloadOpenCL (
            void  )
```

Unloads the OpenCL library.

This function unloads the library.

Multithreading:

fully threadsafe

6.2 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/dbscan_c.h File Reference

Header file for the C/C+GPU implementations of the DBSCAN algorithm.

```
#include <jni.h>
```

Functions

- JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint kk, jint features)
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c_1gpu (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint kk, jint features, jlongArray e)
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c_1phtreads (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint kk, jint features, jint cores, jlongArray e)

6.2.1 Detailed Description

Header file for the C/C+GPU implementations of the DBSCAN algorithm.

This header file contains three method prototypes, that allow to perform single- or multithreaded CPU or GPU based DBSCAN cluster searches.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Warning

This file is machine generated

Date

11.9.2021

6.2.2 Function Documentation

6.2.2.1 Java_com_example_dmocl_dbscan_dbscan_1c()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint kk,
            jint features )
```

Performs a DBSCAN cluster search on the input data with one thread.

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
kk	number of neighbours
features	number of features per data item contained in the data array

Returns

number of clusters found (can be zero if only noise points have been detected) or - if negative - an error code

Multithreading:

fully threadsafe

6.2.2.2 Java_com_example_dmocl_dbscan_dbscan_1c_1gpu()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c_1gpu (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint kk,
            jint features,
            jlongArray e )
```

Performs a DBSCAN cluster search on the GPU.

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
kk	number of neighbours
features	number of features per data item contained in the data array
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

number of clusters found (can be zero if only noise points have been detected) or - if negative - an error code

Multithreading:

fully threadsafe

6.2.2.3 Java_com_example_dmocl_dbscan_dbscan_1c_1phtreads()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_dbscan_dbscan_1c_1phtreads (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint kk,
            jint features,
            jint cores,
            jlongArray e )
```

Performs a DBSCAN cluster search on the input data with multiple threads.

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
kk	number of neighbours
features	number of features per data item contained in the data array
cores	number of cores that should be used
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

number of clusters found (can be zero if only noise points have been detected) or - if negative - an error code

Multithreading:

fully threadsafe

6.3 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/kmeans_c.h File Reference

Header file for the C/C+GPU implementations of the Kmeans algorithm.

```
#include <jni.h>
```

Functions

- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint kk, jint features)
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1gpu (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint cluno, jint features, jlongArray e)
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint cluno, jint features, jint cores, jlongArray e)
- JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmabort_1c (JNIEnv ∗env, jclass clazz)
- JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmresume_1c (JNIEnv ∗env, jclass clazz)

6.3.1 Detailed Description

Header file for the C/C+GPU implementations of the Kmeans algorithm.

This header file contains three method prototypes, that allow to perform single- or multithreaded CPU or GPU based Kmeans cluster searches.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Warning

This file is machine generated

Date

11.9.2021

6.3.2 Function Documentation

6.3.2.1 Java_com_example_dmocl_kmeans_kmabort_1c()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmabort_1c (
            JNIEnv * env,
            jclass clazz )
```

Signals all running Kmeans algorithms to abort immediately. Any new Kmeans cluster search will be aborted imediately.

Warning

This function acts on a 'global' scale: All callers that use this library will not be any more able to make calls to the library functions of this library.

Parameters

env	JNI environment variable
clazz	JNI class variable

Multithreading:

fully threadsafe

6.3.2.2 Java_com_example_dmocl_kmeans_kmeans_1c()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint kk,
            jint features )
```

Performs a Kmeans cluster search on the CPU (one thread).

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
kk	number of clusters to search for
features	number of features per data item contained in the data array

Returns

0 = no error, $<$0 = error number

Multithreading:

fully threadsafe

6.3.2.3 Java_com_example_dmocl_kmeans_kmeans_1c_1gpu()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1gpu (
            JNIEnv * env,
            jclass jc,
```

```
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint cluno,
            jint features,
            jlongArray e )
```

Performs a Kmeans cluster search on the GPU.

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
cluno	number of clusters to search for
features	number of features per data item contained in the data array
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.3.2.4 Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint cluno,
            jint features,
            jint cores,
            jlongArray e )
```

Performs a Kmeans cluster search on the CPU (multiple threads).

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers
rf	(in) Array of data points
eps	search radius
cluno	numbers of clusters that should be found
features	number of features per data item contained in the data array
cores	number of cores that should be used
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.3.2.5 Java_com_example_dmocl_kmeans_kmresume_1c()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmresume_1c (
            JNIEnv * env,
            jclass clazz )
```

Allows to make new Kmeans cluster searches.

Warning

This function acts on a 'global' scale. It reverts the effect of Java_com_example_dmocl_kmeans_kmabort_1c.

Parameters

env	JNI environment variable
clazz	JNI class variable

Multithreading:

fully threadsafe

6.4 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/oclwrapper.h File Reference

Defines the default target OpenCL version.

Macros

- #define UNKNOWN -1
  unknown architecture
- #define ARM32 0
  arm-v7
- #define ARM64 1
  arm-v8
- #define INTEL32 2
  intel x86
- #define INTEL64 3
  intel x86_64
- #define CL_TARGET_OPENCL_VERSION 120
  the default target OpenCL version

Functions

- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_isCLang (JNIEnv ∗env, jclass clazz)

  Checks if CLANG has been used for compilation.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmaj (JNIEnv ∗env, jclass clazz)

  Returns the CLANG major version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmin (JNIEnv ∗env, jclass clazz)

  Returns the CLANG minor version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLpatch (JNIEnv ∗env, jclass clazz)

  Returns the CLANG patch version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_loadOpenCL (JNIEnv ∗env, jobject thiz, jstring s)

  Java wrapper function to load the native OpenCL library.
- JNIEXPORT void JNICALL Java_com_example_dmocl_oclwrap_unloadOpenCL (JNIEnv ∗env, jobject thiz)

  Java wrapper function to unload the native OpenCL library.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt (JNIEnv ∗env, jobject thiz)

  Returns the number of OpenCL platforms.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt (JNIEnv ∗env, jobject thiz, jint i)

  Returns the number of OpenCL devices for a given platform.
- JNIEXPORT jobject JNICALL Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName (JNIEnv ∗env, jobject thiz, jint platf, jint dev)

  Returns some info for a given OpenCL device (and platform number)
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getArchitecture (JNIEnv ∗env, jclass clazz)

  Returns the CPU architecture.

6.4.1 Detailed Description

Defines the default target OpenCL version.

Defines the default target OpenCL version

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Warning

This header file must be included **BEFORE** any OpenCL header file

Author

Robert Fritze

Date

11.9.2021

6.4.2 Function Documentation

6.4.2.1 Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt (
            JNIEnv * env,
            jobject thiz,
            jint i )
```

Returns the number of OpenCL devices for a given platform.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
i	platform number

Returns

>=0 number of devices, <0 error occurred

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.4.2.2 Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName()

```
JNIEXPORT jobject JNICALL Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName (
            JNIEnv * env,
            jobject thiz,
            jint platf,
            jint dev )
```

Returns some info for a given OpenCL device (and platform number)

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
platf	platform number
dev	device number

Returns

an instance of the class *oclinforet* with the information filled in

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.4.2.3 Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt (
            JNIEnv * env,
            jobject thiz )
```

Returns the number of OpenCL platforms.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class

Returns

>=0 number of platfroms, <0 error occurred

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.4.2.4 Java_com_example_dmocl_oclwrap_getArchitecture()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getArchitecture (
            JNIEnv * env,
            jclass clazz )
```

Returns the CPU architecture.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

One of the constants ARM32, ARM64, INTEL32, INTEL64, UNKNOWN

Multithreading:

fully threadsafe

6.4.2.5 Java_com_example_dmocl_oclwrap_getCLmaj()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmaj (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG major version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.4.2.6 Java_com_example_dmocl_oclwrap_getCLmin()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmin (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG minor version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.4.2.7 Java_com_example_dmocl_oclwrap_getCLpatch()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLpatch (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG patch version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.4.2.8 Java_com_example_dmocl_oclwrap_isCLang()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_isCLang (
            JNIEnv * env,
            jclass clazz )
```

Checks if CLANG has been used for compilation.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

1 compiled by CLANG, 0 compiled with other compiler

Multithreading:

fully threadsafe

6.4.2.9 Java_com_example_dmocl_oclwrap_loadOpenCL()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_loadOpenCL (
            JNIEnv * env,
            jobject thiz,
            jstring s )
```

Java wrapper function to load the native OpenCL library.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
s	path and name of the OpenCL library on the device

Returns

OK: 0, library has already been loaded: -1, unable to load library: -2

Multithreading:

fully threadsafe

6.4.2.10 Java_com_example_dmocl_oclwrap_unloadOpenCL()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_oclwrap_unloadOpenCL (
            JNIEnv * env,
            jobject thiz )
```

Java wrapper function to unload the native OpenCL library.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class

Multithreading:

fully threadsafe

6.5 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/include/rwlock_wp.h File Reference

Header file for a writer preferred reader/writer lock.

```
#include <pthread.h>
```

Classes

- struct rwlockwp

  A struct thats holds all necessary components for the lock.

Macros

- #define RWLOCK_STATIC_INITIALIZER { PTHREAD_MUTEX_INITIALIZER, PTHREAD_COND_INITIA↩ LIZER, 0, 0, 0 }

  A static initializer that can be used by assignment.

Functions

- void rwlockwp_reader_acquire (volatile struct rwlockwp *)

  Acquires the reader lock.
- void rwlockwp_reader_release (volatile struct rwlockwp *)

  Releases the reader lock.
- void rwlockwp_writer_acquire (volatile struct rwlockwp *)

  Acquires the writer lock.
- void rwlockwp_writer_release (volatile struct rwlockwp *)

  Releases the writer lock.

6.5.1 Detailed Description

Header file for a writer preferred reader/writer lock.

Defines the struct needed for a writer preferred reader/writer lock. Read- and Writer locks can be acquired and released. A static initializer for the lock is provided.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Date

11.9.2021

6.5.2 Function Documentation

6.5.2.1 rwlockwp_reader_acquire()

```
void rwlockwp_reader_acquire (
            volatile struct rwlockwp *  )
```

Acquires the reader lock.

Acquires the reader lock. Multiple readers can acquire the lock at the same time. If a writer has acquired the writer lock, all new readers are blocked until the writer has finished.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.5.2.2 rwlockwp_reader_release()

```
void rwlockwp_reader_release (
            volatile struct rwlockwp *  )
```

Releases the reader lock.

Releases the reader lock. If no more other readers are holding a reader lock and a writer is waiting, the writer will get exclusive access.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.5.2.3 rwlockwp_writer_acquire()

```
void rwlockwp_writer_acquire (
            volatile struct rwlockwp *  )
```

Acquires the writer lock.

Acquires the writer lock. All new readers have to queue up. The writer is blocked until all reader that already hold a reader lock have finished.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.5.2.4 rwlockwp_writer_release()

```
void rwlockwp_writer_release (
            volatile struct rwlockwp *  )
```

Releases the writer lock.

Releases the writer lock. All waiting readers will wake up.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.6 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/kmeans_c.c File Reference

Source file for the C/C+GPU implementations of the Kmeans algorithm.

```
#include <jni.h>
#include "kmeans_c.h"
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "oclwrapper.h"
#include <CL/opencl.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdint.h>
#include <time.h>
#include <string.h>
#include <rwlock_wp.h>
```

Classes

- struct kmeans_pt

  Parameters for the kmeans thread.

Macros

- #define CL_USE_DEPRECATED_OPENCL_1_2_APIS

  use older OpenCL APIS
- #define GPUTIMING

  Define if detailed timing for the GPU should be made.
- #define MAXCYCLES 100000

  maximum numbers of cycles for kmeans (to avoid endless cycling)

Functions

- int rand_lim (int limit)

  Random number generator.
- short kmeans (unsigned short ∗b, const float ∗data, const int blen, const float eps, const int cluno, const int features)

  Kmeans cluster search.
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint kk, jint features)
- short kmeans_gpu (cl_ushort ∗b, const cl_float ∗data, const int blen, const float eps, const int cluno, const int features, cl_command_queue commands, cl_program program, cl_device_id device, cl_kernel kernel_↩ testdistance, cl_mem data_g, cl_mem b_g, cl_mem clucent_g)

  kmeans cluster search on the GPU
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1gpu (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint cluno, jint features, jlongArray e)
- void ∗ kmthread (void ∗arg)

  thread for calculating kmeans in parallel
- short kmeans_pthreads (unsigned short ∗b, const float ∗data, float ∗clucent, const int blen, const int cluno, const int features, const int cores, struct kmeans_pt ∗kmthreads, const float eps)

  Perform multithreaded Kmeans cluster search.
- JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads (JNIEnv ∗env, jclass jc, jshortArray b, jfloatArray rf, jfloat eps, jint cluno, jint features, jint cores, jlongArray e)
- JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmabort_1c (JNIEnv ∗env, jclass clazz)
- JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmresume_1c (JNIEnv ∗env, jclass clazz)

Variables

- volatile struct rwlockwp abortcalckm = RWLOCK_STATIC_INITIALIZER

  A reader writer lock for premature abort.
- volatile int doabort = 0

  1=abort, access with abortcalckm
- const char ∗ clsource

  Kmeans OpenCL kernel.

6.6.1 Detailed Description

Source file for the C/C+GPU implementations of the Kmeans algorithm.

This header file contains three method prototypes, that allow to perform single- or multithreaded CPU or GPU based Kmeans cluster searches.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Date

11.9.2021

6.6.2 Function Documentation

6.6.2.1 Java_com_example_dmocl_kmeans_kmabort_1c()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmabort_1c (
            JNIEnv * env,
            jclass clazz )
```

Signals all running Kmeans algorithms to abort immediately. Any new Kmeans cluster search will be aborted imediately.

Warning

This function acts on a 'global' scale: All callers that use this library will not be any more able to make calls to the library functions of this library.

Parameters

env	JNI environment variable
clazz	JNI class variable

Multithreading:

fully threadsafe

6.6.2.2 Java_com_example_dmocl_kmeans_kmeans_1c()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint kk,
            jint features )
```

Performs a Kmeans cluster search on the CPU (one thread).

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
kk	number of clusters to search for
features	number of features per data item contained in the data array

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.6.2.3 Java_com_example_dmocl_kmeans_kmeans_1c_1gpu()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1gpu (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
            jfloatArray rf,
            jfloat eps,
            jint cluno,
            jint features,
            jlongArray e )
```

Performs a Kmeans cluster search on the GPU.

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers (0=noise point)
rf	(in) Array of data points
eps	search radius
cluno	number of clusters to search for
features	number of features per data item contained in the data array
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.6.2.4 Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads()

```
JNIEXPORT jshort JNICALL Java_com_example_dmocl_kmeans_kmeans_1c_1phtreads (
            JNIEnv * env,
            jclass jc,
            jshortArray b,
```

```
            jfloatArray rf,
            jfloat eps,
            jint cluno,
            jint features,
            jint cores,
            jlongArray e )
```

Performs a Kmeans cluster search on the CPU (multiple threads).

Parameters

env	JNI environment variable
jc	JNI class variable
b	(out) Array of cluster numbers
rf	(in) Array of data points
eps	search radius
cluno	numbers of clusters that should be found
features	number of features per data item contained in the data array
cores	number of cores that should be used
e	(out) Array of exactly one long value, contains the exclusive time needed (in ns)

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.6.2.5 Java_com_example_dmocl_kmeans_kmresume_1c()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_kmeans_kmresume_1c (
            JNIEnv * env,
            jclass clazz )
```

Allows to make new Kmeans cluster searches.

Warning

This function acts on a 'global' scale. It reverts the effect of Java_com_example_dmocl_kmeans_kmabort_1c.

Parameters

env	JNI environment variable
clazz	JNI class variable

Multithreading:

fully threadsafe

6.6.2.6 kmeans()

```
short kmeans (
             unsigned short * b,
             const float * data,
             const int blen,
             const float eps,
             const int cluno,
             const int features )
```

Kmeans cluster search.

Performs a Kmeans cluster search on the CPU (one thread).

Parameters

b	(out) Array of cluster numbers
data	(in) Array of data points
blen	number of data items in data
eps	maximum cluster center displacement
cluno	number of clusters to search for
features	number of features per data item

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.6.2.7 kmeans_gpu()

```
short kmeans_gpu (
             cl_ushort * b,
             const cl_float * data,
             const int blen,
             const float eps,
             const int cluno,
             const int features,
             cl_command_queue commands,
             cl_program program,
             cl_device_id device,
             cl_kernel kernel_testdistance,
             cl_mem data_g,
             cl_mem b_g,
             cl_mem clucent_g )
```

kmeans cluster search on the GPU

Performs a Kmeans cluster search on the GPU

Parameters

b	(out) Array of cluster numbers
data	(in) Array of data points
blen	number of data items in data
eps	maximum cluster center displacement
cluno	number of clusters to search for
features	number of features per data item
commands	the OpenCL command queue
program	the OpenCL program
device	the OpenCL device
kernel_testdistance	the OpenCL kernel
data_g	OpenCL data buffer
b_g	OpenCL cluster number buffer
clucent_g	OpenCL cluster center buffer

Returns

0 = no error, <0 = error number

Multithreading:

fully threadsafe

6.6.2.8 kmeans_pthreads()

```
short kmeans_pthreads (
            unsigned short * b,
            const float * data,
            float * clucent,
            const int blen,
            const int cluno,
            const int features,
            const int cores,
            struct kmeans_pt * kmthreads,
            const float eps )
```

Perform multithreaded Kmeans cluster search.

Performs a multithreaded Kmeans cluster search on the CPU

Parameters

b	(out) Array of cluster numbers
data	(in) Array of data points
clucent	(out) Array of cluster centers
blen	number of data items in data
cluno	number of clusters to search for
features	number of features per data item
cores	number of threads to be used (CPU can be oversubscribed)
kmthreads	pointer to the threads (array must contain 'cores' elements)
eps	maximum cluster center displacement

Returns

0=algorithm finished correctly, <0 error occurred

6.6.2.9 kmthread()

```
void* kmthread (
            void * arg )
```

thread for calculating kmeans in parallel

One or more threads perform a kmeans search in parallel. The thread calculates the distances to the cluster centers and saves the number of the cluster center with the smallest distance. Two semaphores are used. The first is acquired by this thread and released by the method that submits the calculations. The second semaphore is acquired by the method that submits the job and released by this thread

Parameters

arg	(in) A pointer to the struct with the parameters

Returns

NULL

6.6.2.10 rand_lim()

```
int rand_lim (
            int limit )
```

Random number generator.

Generates uniformly distributed random numbers [0,limit]

Parameters

limit	the maximum random number desired

Returns

a random number form a uniform distribution over [0,limit]

Multithreading:

fully threadsafe

6.6.3 Variable Documentation

6.6.3.1 clsource

```
const char* clsource
```

Kmeans OpenCL kernel.

__kernel void testdistance

(
global const float* data,
global unsigned short* b,
constant const float* clucent,
const int features,
const int cluno
)

calculates for each data item the eucledean distance to all cluster centers and saves the number of the cluster center with the smallest distance.

parameter	in/out	description
global const float data*	in	input data
global unsigned short b*	out	cluster number for each data item
constant const float clucent*	in	cluster centers
features	in	the number of features per data item
cluno	in	the number of clusters to search for

6.7 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/oclwrapper.c File Reference

Helper functions for OpenCL devices to be called directly form JAVA.

```
#include <jni.h>
#include "oclwrapper.h"
#include "AndroidOpenCL.h"
#include "CL/cl.h"
#include "CL/cl_platform.h"
#include <string.h>
#include <stdlib.h>
```

Macros

- #define TARGETARCH UNKNOWN

  unknown target

Functions

- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_isCLang (JNIEnv ∗env, jclass clazz)
  Checks if CLANG has been used for compilation.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmaj (JNIEnv ∗env, jclass clazz)
  Returns the CLANG major version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmin (JNIEnv ∗env, jclass clazz)
  Returns the CLANG minor version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLpatch (JNIEnv ∗env, jclass clazz)
  Returns the CLANG patch version number.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_loadOpenCL (JNIEnv ∗env, jobject thiz, jstring s)
  Java wrapper function to load the native OpenCL library.
- JNIEXPORT void JNICALL Java_com_example_dmocl_oclwrap_unloadOpenCL (JNIEnv ∗env, jobject thiz)
  Java wrapper function to unload the native OpenCL library.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt (JNIEnv ∗env, jobject thiz)
  Returns the number of OpenCL platforms.
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt (JNIEnv ∗env, jobject thiz, jint i)
  Returns the number of OpenCL devices for a given platform.
- JNIEXPORT jobject JNICALL Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName (JNIEnv ∗env, jobject thiz, jint platf, jint dev)
  Returns some info for a given OpenCL device (and platform number)
- JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getArchitecture (JNIEnv ∗env, jclass clazz)
  Returns the CPU architecture.

Variables

- const char isclang = 0
  0=CLANG has not been used

6.7.1 Detailed Description

Helper functions for OpenCL devices to be called directly form JAVA.

This source file contains some helper functions that allow to read out system information and some OpenCL device information directly without the need of C.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Date

11.9.2021

6.7.2 Function Documentation

6.7.2.1 Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetDeviceCnt (
            JNIEnv * env,
            jobject thiz,
            jint i )
```

Returns the number of OpenCL devices for a given platform.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
i	platform number

Returns

>=0 number of devices, <0 error occurred

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.7.2.2 Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName()

```
JNIEXPORT jobject JNICALL Java_com_example_dmocl_oclwrap_AndrCLgetDeviceName (
            JNIEnv * env,
            jobject thiz,
            jint platf,
            jint dev )
```

Returns some info for a given OpenCL device (and platform number)

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
platf	platform number
dev	device number

Returns

an instance of the class *oclinforet* with the information filled in

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.7.2.3 Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_AndrCLGetPlatformCnt (
            JNIEnv * env,
            jobject thiz )
```

Returns the number of OpenCL platforms.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class

Returns

>=0 number of platfroms, <0 error occurred

Multithreading:

fully threadsafe (if native OpenCL function is threadsafe)

6.7.2.4 Java_com_example_dmocl_oclwrap_getArchitecture()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getArchitecture (
            JNIEnv * env,
            jclass clazz )
```

Returns the CPU architecture.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

One of the constants ARM32, ARM64, INTEL32, INTEL64, UNKNOWN

Multithreading:

fully threadsafe

6.7.2.5 Java_com_example_dmocl_oclwrap_getCLmaj()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmaj (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG major version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.7.2.6 Java_com_example_dmocl_oclwrap_getCLmin()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLmin (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG minor version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.7.2.7 Java_com_example_dmocl_oclwrap_getCLpatch()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_getCLpatch (
            JNIEnv * env,
            jclass clazz )
```

Returns the CLANG patch version number.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

version number or -1 if not compiled with CLANG

Multithreading:

fully threadsafe

6.7.2.8 Java_com_example_dmocl_oclwrap_isCLang()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_isCLang (
            JNIEnv * env,
            jclass clazz )
```

Checks if CLANG has been used for compilation.

Parameters

env	pointer to JNI environment
clazz	reference to JNI class

Returns

1 compiled by CLANG, 0 compiled with other compiler

Multithreading:

fully threadsafe

6.7.2.9 Java_com_example_dmocl_oclwrap_loadOpenCL()

```
JNIEXPORT jint JNICALL Java_com_example_dmocl_oclwrap_loadOpenCL (
            JNIEnv * env,
            jobject thiz,
            jstring s )
```

Java wrapper function to load the native OpenCL library.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class
s	path and name of the OpenCL library on the device

Returns

OK: 0, library has already been loaded: -1, unable to load library: -2

Multithreading:

fully threadsafe

6.7.2.10 Java_com_example_dmocl_oclwrap_unloadOpenCL()

```
JNIEXPORT void JNICALL Java_com_example_dmocl_oclwrap_unloadOpenCL (
            JNIEnv * env,
            jobject thiz )
```

Java wrapper function to unload the native OpenCL library.

Parameters

env	pointer to JNI environment
thiz	reference to JNI class

Multithreading:

fully threadsafe

6.8 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/OpenCL.c File Reference

A this OpenCL wrapper for the libOpenCL.so shared library on the Android device.

```
#include "AndroidOpenCL.h"
#include <CL/opencl.h>
#include <CL/cl_icd.h>
#include <dlfcn.h>
#include <string.h>
#include <pthread.h>
```

Macros

- #define CL_TARGET_OPENCL_VERSION 120

  rescue definition of the OpenCL version
- #define SAVECHECKER(a)

  Macro that checks if prerequists for calling a native OpenCL function are met.
- #define WRAPPERCLFUNCT(a, b, c)

  Macro that simplifys the definition of the wrapper methods.

Functions

- CL_API_ENTRY cl_int CL_API_CALL **clGetPlatformIDs** (cl_uint num_entries, cl_platform_id ∗platforms, cl_uint ∗num_platforms) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetPlatformInfo** (cl_platform_id platform, cl_platform_info param_name, size_t param_value_size, void ∗param_value, size_t ∗param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetDeviceIDs** (cl_platform_id platform, cl_device_type device_type, cl_uint num_entries, cl_device_id ∗devices, cl_uint ∗num_devices) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetDeviceInfo** (cl_device_id device, cl_device_info param_name, size_t param_value_size, void ∗param_value, size_t ∗param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_context CL_API_CALL **clCreateContext** (const cl_context_properties ∗properties, cl_uint num_devices, const cl_device_id ∗devices, void(CL_CALLBACK ∗pfn_notify)(const char ∗errinfo, const void ∗private_info, size_t cb, void ∗user_data), void ∗user_data, cl_int ∗errcode_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_context CL_API_CALL **clCreateContextFromType** (const cl_context_properties ∗properties, cl_device_type device_type, void(CL_CALLBACK ∗pfn_notify)(const char ∗errinfo, const void ∗private_info, size_t cb, void ∗user_data), void ∗user_data, cl_int ∗errcode_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clRetainContext** (cl_context context) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clReleaseContext** (cl_context context) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetContextInfo** (cl_context context, cl_context_info param_name, size_t param_value_size, void ∗param_value, size_t ∗param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clRetainCommandQueue** (cl_command_queue command_queue) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clReleaseCommandQueue** (cl_command_queue command_queue) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetCommandQueueInfo** (cl_command_queue command_queue, cl_command_queue_info param_name, size_t param_value_size, void ∗param_value, size_t ∗param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_mem CL_API_CALL **clCreateBuffer** (cl_context context, cl_mem_flags flags, size_t size, void ∗host_ptr, cl_int ∗errcode_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clRetainMemObject** (cl_mem memobj) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clReleaseMemObject** (cl_mem memobj) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetSupportedImageFormats** (cl_context context, cl_mem_flags flags, cl_mem_object_type image_type, cl_uint num_entries, cl_image_format ∗image_formats, cl_uint ∗num_image_formats) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clGetMemObjectInfo** (cl_mem memobj, cl_mem_info param_name, size_t param_value_size, void ∗param_value, size_t ∗param_value_size_ret) CL_API_SUFFIX__VERSION_1_0

- CL_API_ENTRY cl_int CL_API_CALL **clGetImageInfo** (cl_mem image, cl_image_info param_name, size_t param_value_size, void *param_value, size_t *param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clReleaseProgram** (cl_program program) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clReleaseKernel** (cl_kernel kernel) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clEnqueueReadBuffer** (cl_command_queue command_queue, cl_mem buffer, cl_bool blocking_read, size_t offset, size_t size, void *ptr, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clEnqueueNDRangeKernel** (cl_command_queue command_queue, cl_kernel kernel, cl_uint work_dim, const size_t *global_work_offset, const size_t *global_work_size, const size_t *local_work_size, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clSetKernelArg** (cl_kernel kernel, cl_uint arg_index, size_t arg_size, const void *arg_value) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_kernel CL_API_CALL **clCreateKernel** (cl_program program, const char *kernel_name, cl_int *errcode_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_program CL_API_CALL **clCreateProgramWithSource** (cl_context context, cl_uint count, const char **strings, const size_t *lengths, cl_int *errcode_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY CL_EXT_PREFIX__VERSION_1_2_DEPRECATED cl_command_queue CL_API_CALL **clCreateCommandQueue** (cl_context context, cl_device_id device, cl_command_queue_properties properties, cl_int *errcode_ret) CL_EXT_SUFFIX__VERSION_1_2_DEPRECATED
- CL_API_ENTRY cl_int CL_API_CALL **clGetProgramBuildInfo** (cl_program program, cl_device_id device, cl_program_build_info param_name, size_t param_value_size, void *param_value, size_t *param_value_size_ret) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clBuildProgram** (cl_program program, cl_uint num_devices, const cl_device_id *device_list, const char *options, void(CL_CALLBACK *pfn_notify)(cl_program program, void *user_data), void *user_data) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clEnqueueWriteBuffer** (cl_command_queue command_queue, cl_mem buffer, cl_bool blocking_write, size_t offset, size_t size, const void *ptr, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event) CL_API_SUFFIX__VERSION_1_0
- CL_API_ENTRY cl_int CL_API_CALL **clFinish** (cl_command_queue command_queue) CL_API_SUFFIX__VERSION_1_0
- int loadOpenCL (const char *p)

  Loads the OpenCL library of the Android device dynamically.
- void unloadOpenCL ()

  Unloads the OpenCL library.

Variables

- void * dlcall = NULL

  Holds the reference to the loaded native library libOpenCL.so
- pthread_mutex_t dllock = PTHREAD_MUTEX_INITIALIZER

  A mutex for the mechanism that loads the library.
- pthread_rwlock_t lock = PTHREAD_RWLOCK_INITIALIZER

  A R/W lock for the mutual exclusion of the library load/unload mechanism.
- const cl_icd_dispatch CL_WRAP_CALL_ZERO

  A constant with all pointers set to zero.
- cl_icd_dispatch cl_wrap_call

  holds the function pointers to the native OpenCL library

6.8.1 Detailed Description

A this OpenCL wrapper for the libOpenCL.so shared library on the Android device.

This library acts as a small wrapper (glue) for the native OpenCL library on an Android device. The library on the device usually is not present at compile time and if it would, it would be added to the apk file at compile time. The app would run ONLY on this type of device (e.g., a Mali GPU) but could not be ported to other Android devices. Therefore this wrapper loads the OpenCL library and all necessary symbols at runtime. This library supports OpenCL 3.0. If your device does not support this version, either rebuild this library with the version appropriate for your device or simply do not call methods not supported on your device. If you do so, a runtime error will occur as the necessary symbol will not be found in the library.

Copyright

Copyright Robert Fritze 2021

Version

1.0

Author

Robert Fritze

Date

11.9.2021

License:

This program is released under the MIT License.

6.8.2 Macro Definition Documentation

6.8.2.1 SAVECHECKER

```
#define SAVECHECKER(
              a )
```

Value:

```
pthread_mutex_lock( &dllock );            \
                                          \
      if (dlcall == NULL){                \
        weiter = 1;                       \
      }                                   \
                                          \
      if ((weiter == 0) && (cl_wrap_call.a==NULL)) {   \
        cl_wrap_call.a = (cl_api_##a) dlsym(dlcall, #a ); \
                                          \
        if (cl_wrap_call.a == NULL) {     \
          weiter = 1;                     \
        }                                 \
      }                                   \
                                          \
      pthread_mutex_unlock( &dllock );    \
```

Macro that checks if prerequists for calling a native OpenCL function are met.

This macro checks if the native library has been loaded and if the necessary symbol for the method call has been resolved. If the library has been loaded, but the symbol has not yet been resolved, the symbol will be resolved. Sets the "weiter" variable accordingly. Uses a lock to get exclusive access.

Parameters

a	the method name of the native library

Warning

The lock **lock** must have been acquired before

Multithreading:

fully threadsafe

6.8.2.2 WRAPPERCLFUNCT

```
#define WRAPPERCLFUNCT(
              a,
              b,
              c )
```

Value:

```
\
    pthread_rwlock_rdlock( &lock );             \
                                                \
    int weiter = 0;                             \
                                                \
    SAVECHECKER( a )                            \
                                                \
    if (weiter == 0) {                          \
      ret = cl_wrap_call.a b;                   \
    }                                           \
    else {                                      \
      ret = c;                                  \
    }                                           \
                                                \
    pthread_rwlock_unlock( &lock );             \
                                                \
    return( ret );                              \
```

Macro that simplifys the definition of the wrapper methods.

This macro checks the native library is ready and calls the method of the native library.

Parameters

a	OpenCL method name
b	a list with all parameters of the native function
c	the error code that should be returned if the library function can not be called

Multithreading:

fully threadsafe

6.8.3 Function Documentation

6.8.3.1 loadOpenCL()

```
int loadOpenCL (
            const char * c )
```

Loads the OpenCL library of the Android device dynamically.

Loads the OpenCL library dynamically. This function **MUST** be called exactly once before any other call to an OpenCL function. The function stores the path of the library. If the library has already been loaded, a call to this method will have no effect. Any call to an OpenCL function without prior call to this method will result in an error.

Parameters

c	(in) Pointer to the path and name of the OpenCL-library on the device (can be reused after the call)

Returns

OK: 0, library has already been loaded: -1, unable to load library: -2

Multithreading:

fully threadsafe

6.8.3.2 unloadOpenCL()

```
void unloadOpenCL (
            void  )
```

Unloads the OpenCL library.

This function unloads the library.

Multithreading:

fully threadsafe

6.8.4 Variable Documentation

6.8.4.1 cl_wrap_call

```
cl_icd_dispatch cl_wrap_call
```

holds the function pointers to the native OpenCL library

This variable is a struct that holds the function pointers to the native functions of the native OpenCL-libraray. The function symbols are resolved only just before they are actually needed. As long as they are not needed, they hold the value NULL. This avoids a long initialization overhead once the library is loaded.

Warning

Access only with the lock *lock* (read-only access) and the lock *dllock* (write access)

6.8.4.2 dlcall

```
void* dlcall = NULL
```

Holds the reference to the loaded native library *libOpenCL.so*

Warning

Use **dllock** for read+write access

6.8.4.3 dllock

```
pthread_mutex_t dllock = PTHREAD_MUTEX_INITIALIZER
```

A mutex for the mechanism that loads the library.

This mutex guarantees exclusive access to the attributes *dlcall* and *cl_wrap_call*

Warning

For cascade lock use together with lock, acqurire first *lock*

6.8.4.4 lock

```
pthread_rwlock_t lock = PTHREAD_RWLOCK_INITIALIZER
```

A R/W lock for the mutual exclusion of the library load/unload mechanism.

This lock provides the mechanism to exclude to OpenCL library unload mechanism while the library is beeing loaded or some OpenCL function is executed. Arbitrary many methods can acquire the reader lock (including loadOpenCL) but only *unloadOpenCL* acquires the writer lock (resulting in an exclusive access to the entire library).

Warning

For cascade lock use together with dllock, acqurire first **lock**

6.9 /home/robert/AndroidStudioProjects/DMGPU/app/src/C/source/rwlock_wp.c File Reference

A writer preferred reader/writer lock.

```
#include "rwlock_wp.h"
```

Functions

- void rwlockwp_reader_acquire (volatile struct rwlockwp *rwl)
  Acquires the reader lock.
- void rwlockwp_reader_release (volatile struct rwlockwp *rwl)
  Releases the reader lock.
- void rwlockwp_writer_acquire (volatile struct rwlockwp *rwl)
  Acquires the writer lock.
- void rwlockwp_writer_release (volatile struct rwlockwp *rwl)
  Releases the writer lock.

6.9.1 Detailed Description

A writer preferred reader/writer lock.

This file implements a writer preferred reader/writer lock. The lock is reentrant for the readers and exclusive for the writers. Once a writer is waiting all readers that have acquired a reader lock are allowed to finish but new readers have to queue up until the writer has finished.

Copyright

Copyright Robert Fritze 2021

License:

MIT

Version

1.0

Author

Robert Fritze

Date

11.9.2021

6.9.2 Function Documentation

6.9.2.1 rwlockwp_reader_acquire()

```
void rwlockwp_reader_acquire (
            volatile struct rwlockwp *  )
```

Acquires the reader lock.

Acquires the reader lock. Multiple readers can acquire the lock at the same time. If a writer has acquired the writer lock, all new readers are blocked until the writer has finished.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.9.2.2 rwlockwp_reader_release()

```
void rwlockwp_reader_release (
            volatile struct rwlockwp *  )
```

Releases the reader lock.

Releases the reader lock. If no more other readers are holding a reader lock and a writer is waiting, the writer will get exclusive access.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.9.2.3 rwlockwp_writer_acquire()

```
void rwlockwp_writer_acquire (
            volatile struct rwlockwp *  )
```

Acquires the writer lock.

Acquires the writer lock. All new readers have to queue up. The writer is blocked until all reader that already hold a reader lock have finished.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

6.9.2.4 rwlockwp_writer_release()

```
void rwlockwp_writer_release (
            volatile struct rwlockwp *  )
```

Releases the writer lock.

Releases the writer lock. All waiting readers will wake up.

Parameters

rwlockwp	Pointer to the reader/writer lock

Multithreading:

fully threadsafe

Index