How To Use

Alex Tritthart
Matr.-Nr. 296001

January 16, 2015

Contents

This document describes the installation and usage of the *communityEvolution* Library. The first chapter handles prerequisites to use the library and describes the installation steps. In the second chapter basic knowledge to get used to the software and library gets provided. The next chapter sets its focus on the library and the last chapter handles common problems and solutions.

1 Prerequisites and Installation

This chapter handles preparation steps.

1.1 Hardware

1.1.1 GPU

The library works only with graphics cards with a compute capability greater than 2.0. To check whether a video card fulfills this, lookup your card in `https://developer.nvidia.com/cuda-gpus`.

In Addition keep your video card driver up to date.

1.1.2 Storage

It is recommended also to use a solid-state-drive as in some cases it is necessary to temporary store data on a harddrive.

1.2 Software

1.2.1 Visual Studio

Dealing with the library I recommend to use Visual Studio. Preferably the 2013 edition. The reason is that the tests and examples are designed to work with Visual Studio. If you want to use another software keep in mind to change settings accordingly. In the following The installation steps are provided for Visual Studio.

Additional information can be found at `http://www.visualstudio.com/de-de/downloads/download-visual-studio-vs.aspx`.

1. Visual Studio projects are bundled in files appended with *.sln*.
2. Solution Explorer contains the overview about your open projects.
3. Project menu: Right click a project to performe project specific actions, like build, run, settings.
4. Build/Compile a project: Open *Solution Explorer*, Right click a project and choose *Build*.

5. Run a project: Choose *Debug* from the menu and pick *Start new instance*. A zero typically indicates a successful run.
6. Project settings: From the menu choose *Properties*

1.2.2 CUDA

To work with the library CUDA has to be installed. The version 6.5 was used.

1. Download CUDA from `https://developer.nvidia.com/cuda-downloads`.
2. Follow instructions at `http://docs.nvidia.com/cuda/cuda-getting-started-guide-fo`
3. Test

   To test the installed software go to `...\ProgramData\NvidiaCorporation\CUDASamples\v6.5\bin\win32\Release` and execute *deviceQuery* and *bandwidthTest*. Both should pass.
4. Compile Test

   Start Samples_vs2013.sln with Visual Studio and compile and run a project (e.g. matrixMul). The project should exit with zero.
5. Thrust Test

   Compile and run *radixSortThrust*. The project should exit with zero.

   For questions to thrust, its current version and additional examples lookup `https://code.google.com/p/thrust/wiki/QuickStartGuide`.

1.2.3 Dirent

To lookup and manage files the dirent library is used.

1. Follow the instructions of `http://www.softagalleria.net/dirent.php`.
2. Major steps: download current dirent package (library uses v 1.20.1), extract the content, copy the header file *dirent.h* to `...\ProgramFiles(x86)\MicrosoftVisualStudio`
   `0\VC\Include`.

1.2.4 cppUnit

1. Follow the instructions of `http://www.comp.nus.edu.sg/~cs3215/tools/cppunitAll.html`
2. Major steps: download current cppunit package (used 1.12.1), extract its content, open `...\cppunit-1.12.1\src\CppUnitLibraries.sln`, build *cppunit* (lib gets produced), copy cppunit-1.12.1 somewhere to remember (e.g. `C:\`)
3. In projects change properties to use cppUnit like the following:
   a) C++ → General → add `C:\cppunit-1.12.1\include` to Additional Include Directories
   b) Linker → Input → add `C:\cppunit-1.12.1\lib\cppunit.lib` to Additional Depencdencies

1.2.5 Test setup

Tests cppUnit and Linking.

Open project *generalTesting.sln*. Build and run *cppUnitTest*. Should show no errors.

2 Basics

2.1 Creation

2.1.1 Create Project in Visual Studio

1. Go to: File → New → Project...
2. Pick *Win32 Console Application*, choose name and location, click OK.
3. Wizard popups. If you want to create a DLL check DLL at its second page, otherwise click finish.
4. Solution Explorer now shows the new project.
5. Especially for big projects it is common to keep header files in one location, e.g. a folder named *header*, and source files also in one place, e.g. *src* or *source*.

2.1.2 Change a project using CUDA

1. Choose a *Console Application*
2. Set Build to *CUDA*
   a) *Right click* your project
   b) Go to → *Build Dependencies* → *Build Customizations*
   c) Activate *CUDA x.0*
3. Add the CUDA library
   a) Open project *Properties*
   b) Switch to → *Configuration Properties* → *Linker* → *Input*
   c) Add *cudart.lib* to *Additional Dependencies*
4. Relocatable
   a) Open project *Properties*
   b) Switch to → *Configuration Properties* → *CUDA C/C++* → *Common*
   c) Set *Generate Relocatable Device Code* to *Yes*
5. Warning C4996
   a) Open project *Properties*
   b) Switch to → *Configuration Properties* → *C/C++* → *Preprocessor*
   c) Add *_CRT_SECURE_NO_DEPRECATE* to *Preprocessor Definitions*
6. Add the main file

   a) Create a file ending in *.cu* (as all CUDA source files have to end with)
   b) Open project *Properties*
   c) Switch to → *General*
   d) Set *Item Type* to *CUDA C/C++*

7. Add headers
   a) Add *#include <cuda.h>*
   b) Add *#include <cuda-runtime>*
8. Check Folder
   a) Open project *Properties*
   b) Switch to → *Configuration Properties* → *C/C++* → *General*
   c) *Additional Include Directories* should contain *$(CudaToolkitIncludeDir)*
9. (Optional) Change Code Generation (Should already be correct)
   a) Open project *Properties*
   b) Switch to → *Configuration Properties* → *CUDA C/C++* → *Device*
   c) Set *Code Generation* to *compute-20,sm-20* (support for your device may differ)
10. (Optional) Additional (Was needed for CUDA 6.0)
    a) Open project *Properties*
    b) Switch to → *Configuration Properties* → *CUDA C/C++* → *Command Line*
    c) Add *-arch sm-20* to *Additional Options*

2.1.3 Create DLL Library

1. Create Application. When Wizard popups, check DLL at its second page.
2. Make it use CUDA if needed
3. For each header insert the code from 2.1, name it according to your file. For each function add the API phrase at front as in line 7.
4. Also make sure to store all headerfiles that should be accessed from the outside in one place. Name the folder e.g. *include*.

```
1 #ifdef CHECKSETTINGSDLL_EXPORTS
2 #define CHECKSETTINGSDLL_API __declspec(dllexport)
3 #else
4 #define CHECKSETTINGSDLL_API __declspec(dllimport)
5 #endif
6
7 CHECKSETTINGSDLL_API myFunction();
```

Listing 2.1: Header Code

2.1.4 Create cppUnit Test Application

1. Create Console Application
2. Open project *Properties*
3. Switch to → *Configuration Properties* → *C/C++* → *General*
4. In: C++ → General → add `C:\cppunit-1.12.1\include` to Additional Include Directories
5. In: Linker → Input → add `C:\cppunit-1.12.1\lib\cppunit.lib` to Additional Depencdencies

2.2 Link to library

To make use of any library it is necessary to add its include folder and its library.

1. Add the include directory to: *Properties* → *C++* → *General* → *Additional Include Directories*. (e.g. `..\communityEvolutionDll\include`).
2. Add the library to: *Properties* → *Linker* → *Input* → *Additional Dependencies.* (e.g. `..\Debug\communityEvolutionDll.lib`).

2.3 Test

To test your configurations I recommend to use the *check_settings* files from the *generalTesting* project. Copy them into your project, build and start it.

3 The Program

3.1 Library

Setting up a project to use the library just follow the instructions of section 2.2 to link to the library. After this step, all exported methods are available. To get an overview about the different functions have a look at the following table.

Table 3.1: Category - Algorithms

Name	Idea	File
algorithm_clique	Analyses pairs and retrieves communities by first creating cliques and melting them afterwards	algorithm_clique.h
algorithm_event_extraction	Analyses communities concerning their behavior from snapshot to snapshot	algorithm_event_extraction.h
algorithm_propinquity	Analyses pairs and retrieves communities by iteratively calculating their propinquity values and adding, removing edges. Communities get detected in the end via a breath-first search	algorithm_propqinuity.h
setStorageCounter	Supports propinquity algorithm	algorithm_propqinuity.h
getStorageCounter	Supports propinquity algorithm	algorithm_propqinuity.h
compress_files	Supports propinquity algorithm	algorithm_propqinuity.h
get_specific_pairs	Supports propinquity algorithm	algorithm_propqinuity.h
couple_increment	Supports propinquity algorithm	algorithm_propqinuity.h
calculate_propinquity	Supports propinquity algorithm	algorithm_propqinuity.h
update_propinquity	Supports propinquity algorithm	algorithm_propqinuity.h
bfs	Runs a breath-first search	algorithm_propqinuity.h
host_algorithm_clique	Analyses pairs and retrieves communities by first creating cliques and melting them afterwards	host_algorithm_clique.h
generate_cliques	Creates cliques for the clique algorithm	host_algorithm_clique.h
generate_communities	Creates communities from cliques	host_algorithm_clique.h

Table 3.2: Category - Device

Name	Idea	File
get_number_of_diff_elements	Determines the number of different elements in a snapshot of pairs	device_analytic.h
get_degree_mirror	Determines the degree of each pair in a snapshot. Directed version	device_analytic.h
get_degree	Determines the degree of each pair in a snapshot. Undirected version	device_analytic.h
get_count	Determines number of occurences of pairs in a vector of pairs	device_analytic.h
get_firsts	Determines position of first edge for each node	device_analytic.h
get_lasts	Determines position of last edge for each node	device_analytic.h
get_nodes	Retrieves all nodes of a snapshot	device_analytic.h
get_max_combinations	Calculates the maximal numbers of possible combinations	device_analytic.h
get_max_combination	Calculates the maximal number of possible combinations	device_analytic.h
get_max_combinations_scanned	Calculates the maximal numbers of possible combinations and returns a scanned vector accordingly	device_analytic.h
get_intersection	Intersects the neighbours for each pair of edgeendings	device_analytic.h
get_modularity	Calculates modularity for a snapshot	device_analytic.h
translate_snapshot_to_vector	Translates a snapshot into a one dimensional vector	device_convert.h
translate_snapshot_to_matrix	Translates a snapshot into a matrix	device_convert.h
translate_scom_to_vector	Creates vector according to the sizes in scom	device_convert.h
combine_values	Gets a vector of pairs with a vector of values and summarizes the values for identical pairs	device_pair.h
combine_pairs	Gets a vector of pairs with a vector of values and just combines identical pairs	device_pair.h
pairsToNodes	Converts vector of pairs into vector of nodes	device_pair.h
pairsToUniqueNodes	Converts vector of pairs into unique vector of nodes	device_pair.h
mirror_pairs	Mirrors a vector of pairs. Converts it from undirected to directed	device_pair.h
mirror_pairs_inplace	Mirrors a vector of pairs. Converts it from undirected to directed. Overrides Input	device_pair.h
generate_pairs_deep	Takes directed, sorted vector of pairs and generates combinations. Second version	device_pair_construct.h
generate_pairs	Takes directed, sorted vector of pairs and generates combinations	device_pair_construct.h
generate_pairs_limit	Takes directed, sorted vector of pairs and generates combinations until a limit is reached	device_pair_construct.h
generate_unique_pairs	Generates combinations and uniques them	device_pair_construct.h
get_pairs	Used to get combinations from a starting to an end	device_pair_construct.h
from_device_store	Stores vectors and pairs from device	device_storage_serilization.h
to_device_load	Loads vectors and pairs to device	device_storage_serilization.h
from_host_store	Stores vectors and pairs from host	device_storage_serilization.h
to_host_load	Loads vectors and pairs to host	device_storage_serilization.h

Table 3.3: Category - General

Name	Idea	File
g_binary_search	Searchs for the first occurence of a value or pair in an array. Can be used on host and device	general_search.h

Table 3.4: Category - Headers

Name	Idea	File
device_headers.h	Headers used in the library for device work	device_headers.h
general_defines.h	Definitions used in the library	general_defines.h
host_headers.h	Headers used in the library for host work	host_headers.h

Table 3.5: Category - Host

Name	Idea	File
Device on Host	All device functions are also available to compute only on CPU	host__*.h
find__file	Finds a file in local storage	host__storage__human.h
get__files	Retrieves all files of a specific folder or all	host__storage__human.h
display__files	Displays all files of a specific folder or all	host__storage__human.h
get__file	Retrieves a specific file	host__storage__human.h
store__compress	Store numbers compressed on disk	host__storage__serilization.h
load__compress	Loads numbers compressed from disk	host__storage__serilization.h
store	Stores data on disk, numbers or pairs	host__storage__serilization.h
load	Loads data from disk, numbers or pairs	host__storage__serilization.h

Table 3.6: Category - Data

Name	Idea	File
Source	Constructor, gets an unique ID	host__*.h
set__source	sets name, datatype and reads data in internal storage. Returns zero if fails	data__source.h
convert__source	Reads data in internal storage	data__source.h
getter	Retrieve of different values, like pairs, communities, ID	data__source.h
store__in__file	Store data in readable or binary format on disk	data__source.h
display	Displays either pairs or communities	data__source.h

3.2 Interface

The interface is a project that helps the user to deal with the library. At first it is possible to load and store data. Second it provides access to the implemented algorithms. The basic structure is shown below. Only two levels are provided since the lower level will probably change more often and do not literally support the comprehension.

1. Manage Data
    1. Set Defaults
    2. Load Data
    3. Store Data
    4. Display Files
2. Algorithms
    1. Analyse snapshots via GPU, event extraction
    2. Extract snapshots via Zhang Propinquity GPU
3. Display Data
    1. Display Pairs
    2. Display Snaps
4. Help
5. Exit

4 Problems and Solutions