Simon Yin – Simon.Yin@uts.edu.au

23/04/15

Setting up CUDA and Mex to compile GPU models in SABL_2015a – Windows 7

Table of Contents

1 Revision History

20150420 – Introduce a new document.

2 Quickstart

Assuming Matlab 2015a (or Matlab 2014b) 64-bit Windows version has already been installed then:

Step 1 of 4: Download MS Visual Studio 2013 Community Edition.

Install it and use defaults for all configuration settings.

Step 2 of 4: Download CUDA Toolkit v7.0 (install it with the “Local Installer”)

Step 3 of 4: Start Matlab; and in the Command Window, issue the command:

```
mex -setup C
```

Confirm that it returns: “*MEX configured to use 'Microsoft Visual C++ 2013 Professional (C)' for C language*”

Otherwise click the link: Microsoft Visual C++ 2013 Professional (C)

```
mex -setup C++
```

Confirm that it returns: “*MEX configured to use 'Microsoft Visual C++ 2013 Professional' for C++ language*”

Otherwise click the link: Microsoft Visual C++ 2013 Professional

Step 4 of 4: Change to a SABL project folder (eg. projects/toyMVN). In file **p_monitor** set `E.gpu=false`, and then from the Command Window issue the command:

```
runexample
```

The SABL MVN model will then run (on CPU only). After it finishes, note the wall-clock time taken by the model.

Next switch to GPU operation: in file **p_monitor** set `E.gpu=true;` and then from the Command Window issue the command:

```
addpath([pwd,'\..\..\models\MVN'])

compile_cuda

runexample
```

3 Background

This document is targeted at SABL 'modeller' level. Familiarity with Nvidia GPGPU hardware and C/C++ programming concepts is assumed. Here we describe the software requirements and installation steps (for the **Windows 7 (64-bit)** platform) that are recommended for compiling and running the CUDA codes that are included in **SABL_2015a.** These codes use the **CUDA Runtime API** and also rely on **cuBLAS** and **thrust** for library routines.

Matlab 2015a (64-bit) is assumed to be the version of Matlab. Matlab 2015a's own Distributed Computing Toolbox is built on CUDA Toolkit v6.5. Here we describe the installation of **CUDA Toolkit v7.0** (backward compatible to v6.5) and the interfacing of the `nvcc` compiler with Matlab's `mex` compiler. Furthermore, the underlying C++ compiler, linker, and profiling tools require the **MS Visual Studio 2013 Community Edition** - this document will describe how this should be installed.

3.1 Other Matlab versions

- **Matlab 2014b (64-bit)** – tested and works ok with the same steps described here.

3.2 Other MS Visual Studio versions

- None tested. See 3.3 Other CUDA Toolkit versions for more information.

3.3 Other CUDA Toolkit versions

- Documentation from Nvidia indicates that several other combinations of MS Visual Studio + CUDA Toolkit are supported:
  - CUDA Toolkit v7.0 + Visual Studio 2012 with Visual C++ 11.0 (not tested)
  - CUDA Toolkit v7.0 + Visual Studio 2010 with Visual C++ 10.0 (tested and works ok. Modify cufiles/mex_CUDA_win64.xml to set `COMPFLAGS: --cl-version 2010`)

4 Order of installation

Ensure that Visual Studio and Matlab are correctly installed first. Then proceed to install the CUDA Toolkit. Finally, install SABL and verify using an example project that CUDA code is compiling and running correctly.

5 Installing CUDA Toolkit v7.0

Nvidia's starting guide is the basis of the information provided here.

6 Styling conventions

Source code will appear in monospace, e.g.

```
__global__ void sabl_mvn()
```

Command line statements are prefixed by $ and in italics, e.g.

```
$ mex -v -largeArrayDims myfile.cu
```

Optional values are in square brackets [], where the mutually exclusive options are listed separated by a token '|' , e.g.

```
$ ./mytest.out [--someoption1 | --someoption2]
```

Required values are in angled brackets <> e.g.

```
$ cp <source filename> <destination filename>
```

7 Glossary

(GP)GPU

(General Purpose) Graphics Processing Unit

SABL

Successive Approximation Bayesian Learning