MATLAB Interface

for ViennaCL 1.0.2

Institute for Microelectronics
Gußhausstraße 27-29 / E360
A-1040 Vienna, Austria

Main Contributors:

Florian Rudolf
Karl Rupp
Josef Weinbub

Current Maintainers:

Florian Rudolf
Karl Rupp
Josef Weinbub

Institute for Microelectronics
Vienna University of Technology
Gußhausstraße 27-29 / E360
A-1040 Vienna, Austria/Europe

Phone	+43-1-58801-36001
FAX	+43-1-58801-36099
Web	http://www.iue.tuwien.ac.at

Contents

Introduction

The MATLAB interface of ViennaCL provides simple access to the computational resources of GPUs using ViennaCL under MATLAB. An interface for the following iterative solvers (confer for example to the book of Y. Saad [1]) is provided:

- Conjugate Gradient (CG)
- Stabilized BiConjugate Gradient (BiCGStab)
- Generalized Minimum Residual (GMRES)

In the current version, all solvers are provided without preconditioner. Since MATLAB relies on double precision arithmetics, only a few GPUs can be used, see Tab. 1.

Double precision arithmetic on GPUs is only possible if it is provided by the GPU. There is no double precision emulation in ViennaCL.

Compute Device	ViennaCL-Matlab
Nvidia Geforce 86XX GT/GSO	no
Nvidia Geforce 88XX GTX/GTS	no
Nvidia Geforce 96XX GT/GSO	no
Nvidia Geforce 98XX GTX/GTS	no
Nvidia GT 230	no
Nvidia GT(S) 240	no
Nvidia GTS 250	no
Nvidia GTX 260	yes
Nvidia GTX 275	yes
Nvidia GTX 280	yes
Nvidia GTX 285	yes
Nvidia GTX 465	yes
Nvidia GTX 470	yes
Nvidia GTX 480	yes
Nvidia Quadro FX 46XX	no
Nvidia Quadro FX 48XX	yes
Nvidia Quadro FX 56XX	no
Nvidia Quadro FX 58XX	yes
Nvidia Tesla 870	no
Nvidia Tesla C10XX	yes
Nvidia Tesla C20XX	yes
ATI Radeon HD 45XX	no
ATI Radeon HD 46XX	no
ATI Radeon HD 47XX	no
ATI Radeon HD 48XX	maybe
ATI Radeon HD 54XX	no
ATI Radeon HD 55XX	no
ATI Radeon HD 56XX	no
ATI Radeon HD 57XX	no
ATI Radeon HD 58XX	maybe
ATI Radeon HD 59XX	maybe
ATI FireStream V92XX	maybe
ATI FirePro V78XX	maybe
ATI FirePro V87XX	maybe
ATI FirePro V88XX	maybe

Table 1: Supported GPUs for the MATLAB interface of `ViennaCL`. At the release of the MATLAB interface for `ViennaCL 1.0.2`, GPUs from AMD/ATI do not comply to `OpenCL` standard for double precision extensions. Once the driver of these GPUs complies to the double precision extension standard of `OpenCL`, they can be used with the MATLAB interface for `ViennaCL` immediately.

Chapter 1

Installation and Usage

This chapter shows how the MATLAB interface for ViennaCL is compiled and how it can be used. The necessary steps are outlined for several different platforms, but we could not check every possible combination of hardware, operating system and compiler. If you experience any trouble, please write to the maining list at

viennacl-support@lists.sourceforge.net

1.1 Dependencies

- A MATLAB version with MEX-interface (eg. R2009a)
- A recent C++ compiler (GCC 4.2.x and higher as well as the Visual C++ compiler in Visual Studio 2008 are known to work)
- OpenCL [2, 3] for accessing compute devices (GPUs); see Section 1.2 for details.

1.2 Get the OpenCL Library

The development of OpenCL applications based on graphics cards requires a suitable driver and a corresponding library, e.g. libOpenCL.so under Unix based systems. This section describes how this library can be acquired.

Note, that for Mac OS X systems there is no need to install an OpenCL capable driver and the corresponding library. The OpenCL library is already present if a suitable graphics card is present. Using ViennaCL on Mac OS X is discussed in Section 1.3.2.

1.2.1 NVIDIA cards

NVIDIA provides the OpenCL library with the driver. Therefore, if a NVIDIA driver is present on the system, the library is too. However, not all of the released drivers contain

the OpenCL library. A driver which is known to support OpenCL, and hence providing the required library, is 195.36.24.

1.2.2 ATI cards

As of the release of ViennaCL, ATI cards lack the ability of full double support [4]. Since the MATLAB interface for ViennaCL requires double precision support, it cannot be used unless full standard-compliant double precision support is made available from AMD/ATI.

1.3 Building the MATLAB Interface

In the following a generic description is given, then some OS-specific details are explained.

The first step is to configure MATLAB. Type

```
mex -setup
```

and choose a suitable C++ compiler.

Make sure that the selected compiler supports C++, not just C.

Then change into the base directory of the MATLAB interface for ViennaCL. If the OpenCL include and library files are installed system-wide, the commands

```
mex viennacl_cg.cpp       -I. -lOpenCL
mex viennacl_bicgstab.cpp -I. -lOpenCL
mex viennacl_gmres.cpp    -I. -lOpenCL
```

build the three solvers.

On 64-bit systems, you may have to append the -largeArrayDims option, otherwise you might get a runtime error when calling the solvers.

1.3.1 Linux

If you are using a new version of GCC(4.3.x and above), you may get linker errors when calling any of the solvers. In that case, install version 4.2 of GCC and change the compiler call in

```
$HOME/.matlab/MATLABVERSION/mexopts.sh
```

to e.g. g++-4.2, where MATLABVERSION refers to you MATLAB version, e.g. R2010a.

On Ubuntu, you can directly install GCCin version 4.2.x from the repository. The executeable is called g++-4.2.

1.3.2 Mac OS X

The tools mentioned in Section 1.1 are available on macintosh platforms too. For the GCC compiler the Xcode [5] package has to be installed. To install CMake and Boost external portation tools have to be used, for example, Fink [6], DarwinPorts [7] or MacPorts [8]. Such portation tools provide the aforementioned packages, CMake and Boost, for macintosh platforms.

For Mac OS X, the following linker flag has to be added to the compilation call.

```
-framework OpenCL
```

This is best done in the mexopts.sh file usually located at

$HOME/.matlab/MATLABVERSION/mexopts.sh .

Typically, this is achieved by adding

```
-framework OpenCL
```

to the C++ compiler flags CXXFLAGS for your architecture (mind that the configuration for both 32 bit and 64 bit systems is located in the mexopts file).

1.3.3 Windows

Since the include and library files for OpenCL are usually not available system-wide, you have to specify their location manually. Assuming that the NVidia CUDA SDK located at C:\CUDA\ is used, type

```
mex viennacl_cg.cpp       -I. -IC:\CUDA\include -LC:\CUDA\lib -lOpenCL
mex viennacl_bicgstab.cpp -I. -IC:\CUDA\include -LC:\CUDA\lib -lOpenCL
mex viennacl_gmres.cpp    -I. -IC:\CUDA\include -LC:\CUDA\lib -lOpenCL
```

1.4 Usage

Simply call the ViennaCL solver interface as for the built-in functions provided by MATLAB:

```
result = viennacl_cg(A, rhs);
result = viennacl_bicgstab(A, rhs);
result = viennacl_gmres(A, rhs);
```

There are a few things to note about the performance of the solvers provided by ViennaCL:

- At the very first invocation of a ViennaCL, the OpenCL compute kernels are compiled, which may take a few seconds. Subsequent calls of any ViennaCL solvers do not have this overhead.
- Since MATLAB stores sparse matrices column-wise, but ViennaCL requires a row-wise storage, the non-symmetric system matrices for BiCGStab and GMRES have to be rearranged in system memory.

- On 32-bit systems, matrix indices are stored as `signed integers`, whereas `ViennaCL` requires `unsigned integers`. Thus, the data structures holding the matrix indices have to be converted, which is also a runtime penalty. At present, indices are also converted on 64-bit systems, even if this may not be required.
- All data has to be transferred to the GPU before the solver can start.
- The solution vector needs to be copied from the GPU back to the main memory, which also constitutes a runtime penalty.

Therefore, the use of the `ViennaCL` solvers in `MATLAB` does not pay off for small systems with only a few unknowns (say, less than 10.000) and very well conditioned systems which need only a few solver iterations to converge. The general rule of thumb is that at least twenty to forty iterations are required to have a significant benefit using `ViennaCL`, cf. Chap. 2.

Chapter 2

Benchmark Results

We have compared the performance of the conjugate gradient solver provided via the MATLAB interface of ViennaCL with the built-in functions of MATLAB. The code used for the benchmarks can be found in the files test_cg.m.

CPU	Intel Core i7 960
GPU	NVidia Geforce GTX 470
RAM	6 GB
OS	Windows 7 Ultimate, 32 bit
Nvidia driver version:	197.75
ViennaCL version	1.0.2

Compute kernels are not fully optimized yet, results are likely to improve considerably in future releases of ViennaCL

The results in Fig. 2.1 show that there is a certain overhead related to starting the compute kernels in OpenCL. However, for large systems, this overhead becomes negligible and the performance benefit can readily be seen.

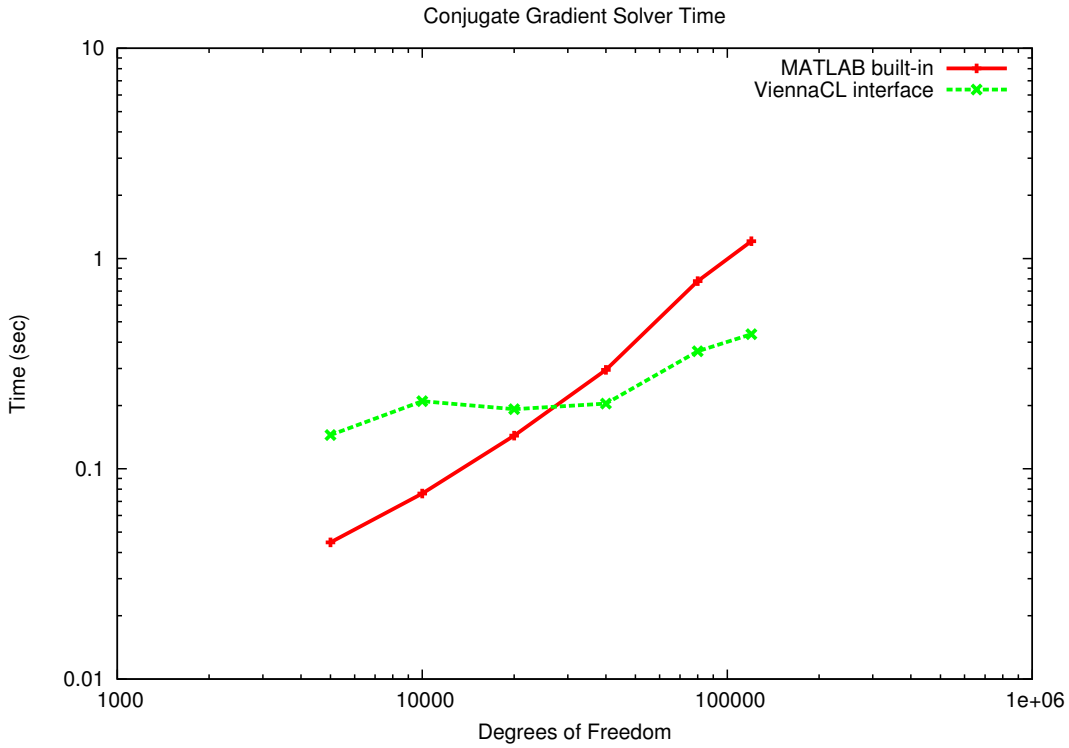

Figure 2.1: Execution time for ten conjugate gradient solver runs (27 iterations each) for different problem sizes.

Change Logs

Version 1.0.x

Version 1.0.2

First release of the MATLAB interface for `ViennaCL`.

License

Bibliography

[1] Y. Saad, *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, April 2003.

[2] "Khronos OpenCL." [Online]. Available: http://www.khronos.org/opencl/

[3] "Nvidia OpenCL." [Online]. Available: http://www.nvidia.com/object/cuda_opencl_new.html

[4] "ATI Knowledge Base - Double Support." [Online]. Available: http://developer.amd.com/support/KnowledgeBase/Lists/KnowledgeBase/DispForm.aspx?ID=88

[5] "Xcode Developer Tools." [Online]. Available: http://developer.apple.com/technologies/tools/xcode.html

[6] "Fink." [Online]. Available: http://www.finkproject.org/

[7] "DarwinPorts." [Online]. Available: http://darwinports.com/

[8] "MacPorts." [Online]. Available: http://www.macports.org/