PENN STATE UNIVERSITY
Department of Economics

Econ 597D Sec 001 Computational Economics — Gallant
Project Suggestion 10 — Fall 2015
Due Dec 8, 2015

The pair `matrixvecmult.cpp, matrixvecmult.cl` of OpenCL C++ code and OpenCL kernel code at `http://www.aronaldg.org/webfiles/arg/compecon/src/opencl/` illustrates five implementations of $c = Ba$, where B is stored in row major form and a is a vector. The pair `vecmatmult.cpp, vecmatmult.cl` is the translation of the fastest of these five, MatrixVectorMul5, to $c = aB$ where B is stored in column major form. The pair `matmatmult.cpp, matmatmult.cl` is analogous to the slowest of these five, MatrixVectorMul1, for $C = AB$ where all three matrices are stored in column major form.

Use the ideas from the kernels MatrixVectorMul3, MatrixVectorMul4, or MatrixVectorMul5 to improve the speed of `matmatmult.cpp, matmatmult.cl`.

For this project, the matrices A, B, and C must be in column major form on the host. Any transposition of matrices must be done on the device, not the host.

Turn in a description of the ideas that you tried and at least one implementation that gets the correct answer and improves on `matmatmult.cpp, matmatmult.cl` together with a timing run for both your code and `matmatmult.cpp, matmatmult.cl` on the same device.

The use of barriers and atomic operations is tricky. You should read Section 7.4 of Scarpino, Matthew, (2011) *OpenCL in Action*, Manning Publications, Shelter Island, NY, before starting this project.

You might consider using Scarpino's approach in Chapter 12 of computing $W = A'$ with one kernel and then computing $C = W'B$ with another. That way the code in MatrixVectorMul3, MatrixVectorMul4, or MatrixVectorMul5 would be much easier to translate and only a one dimensional `NDRange` would be necessary. If you follow this approach, I doubt that you can beat `matmatmult.cpp, matmatmult.cl` if you loop over the columns of W within one kernel. You probably can beat `matmatmult.cpp, matmatmult.cl` by simultaneously launching a kernel for each column of W.

You can work this project if you have an Apple with Mac OS X 10.6 (Snow Leopard) or later. As we saw in class, timing differences are greater on a Linux machine that has a relatively powerful GPU board installed such as an Nvidia Telsa C1060. The project will be more interesting on such a machine because there is a bigger payoff to code that accesses memory efficiently.