ECEC-622: Introduction to Parallel Computer Architecture
CUDA Lab Assignment 3

Prof. Naga Kandasamy, ECE Department, Drexel University

May 23, 2013

The assignment is due on June 5, 2013. You may work on this assignment in groups of up to two.

The Cholesky Decomposition. This problem asks you to develop the Cholesky decomposition method on the GPU. A brief description of Cholesky decomposition follows.

Consider the problem of solving a system of linear equations of the form $Ax = b$ where A is a $n \times n$ coefficient matrix, and x and b are $n \times 1$ vectors. Here, A and b are known and we seek to determine the solution vector x. If the matrix A is symmetric and positive definite, that is $x^{\mathrm{T}}Ax > 0$ for all non-zero vectors x[1], then Cholesky decomposition is an efficient method of computing an upper triangular matrix U with positive diagonal elements such that $U^{\mathrm{T}}U = A$. So, to solve $Ax = b$, one solves first $U^{\mathrm{T}}y = b$ for y and then $Ux = y$ for x.

Consider a simple example of how to obtain the Cholesky decomposition of a 4×4 symmetric matrix A. We would like to obtain a corresponding 4×4 upper triangular matrix U such that $A = U^T U$. So,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} u_{11} & 0 & 0 & 0 \\ u_{12} & u_{22} & 0 & 0 \\ u_{13} & u_{23} & u_{33} & 0 \\ u_{14} & u_{24} & u_{34} & u_{44} \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

$$= \begin{pmatrix} u_{11}^2 & u_{11}u_{12} & u_{11}u_{13} & u_{11}u_{14} \\ u_{11}u_{12} & u_{12}^2 + u_{22}^2 & u_{12}u_{13} + u_{22}u_{23} & u_{12}u_{14} + u_{22}u_{24} \\ u_{11}u_{13} & u_{12}u_{13} + u_{22}u_{23} & u_{13}^2 + u_{23}^2 + u_{33}^2 & u_{13}u_{14} + u_{23}u_{24} + u_{33}u_{34} \\ u_{11}u_{14} & u_{12}u_{14} + u_{22}u_{24} & u_{13}u_{14} + u_{23}u_{24} + u_{33}u_{34} & u_{14}^2 + u_{24}^2 + u_{34}^2 + u_{44}^2 \end{pmatrix}$$

Equating the left and the right-hand sides, we obtain

$$u_{11} = \sqrt{a_{11}}, u_{12} = \frac{a_{12}}{u_{11}}, u_{13} = \frac{a_{13}}{u_{11}}, u_{14} = \frac{a_{14}}{u_{11}}$$

$$u_{22} = \sqrt{a_{22} - u_{12}^2}, u_{23} = \frac{a_{23} - u_{12}u_{13}}{u_{22}}, u_{24} = \frac{a_{24} - u_{12}u_{14}}{u_{22}}$$

$$u_{33} = \sqrt{a_{33} - u_{13}^2 - u_{23}^2}, u_{34} = \frac{a_{34} - u_{13}u_{14} - u_{23}u_{24}}{u_{33}}$$

$$u_{44} = \sqrt{a_{44} - u_{14}^2 - u_{24}^2 - u_{34}^2}$$

Examining the above equations, u_{12} can be computed easily since the key variable on the right-hand side, u_{11} is already known in the previous step. Similarly, u_{22} can be easily computed since u_{12} is known a previous step, and so on. So, the elements of the matrix U can be determined knowing just the elements of A. A serial implementation of a row-oriented Cholesky decomposition algorithm is shown below.
Unlike Gaussian elimination, Cholesky decomposition does not require pivoting since the diagonal element $A[k, k] > 0$ if the matrix A is positive definite. So, our implementation is numerically stable.

[1] A matrix is positive definite if and only if the determinant of each of the principal sub-matrices is positive.

```
1: procedure CHOLESKY(A)
2: int i, j, k;
3: for k := 0 to n − 1 do
4:    A[k, k] := √A[k, k];      /* Obtain the square root of the diagonal element. */
5:    for j := k + 1 to n − 1 do
6:       A[k, j] := A[k, j]/A[k, k];      /* The division step. */
7:    end for
8:    for i := k + 1 to n − 1 do
9:       for j := i to n − 1 do
10:         A[i, j] := A[i, j] - A[k, i] × A[k, j];      /* The elimination step. */
11:      end for
12:   end for
13: end for
```

This problem asks you to identify the parallelism available within the above `CHOLESKY` algorithm and develop a parallel formulation for the GPU. Before tackling this problem, I suggest that you work out the above algorithm on a small matrix to identify potential sources of parallelism. The program given to you accepts no arguments. The CPU computes the reference solution which is compared with the result provided by the GPU. If the solutions match within a certain tolerance, the application will print out "Test PASSED" to the screen before exiting. If you encounter precision issues related to the algorithm, please use the double-precision units on the GPU. Edit the `chol_on_device()` function in `chol.cu` and the `chol_kernel()` function in `chol_kernel.cu` to complete the functionality on the GPU. You may develop additional GPU kernels as necessary. Use various matrix sizes to test and benchmark your code.

E-mail me all of the files needed to run your code as a single zip file called `lab_3.zip`.

This question will be graded on the following parameters:

- **(10 points)** At the very least, you must use GPU global memory to get your code working correctly.
- **(15 points)** Optimize the performance of your code, making efficient, and in my opinion, clever use of shared, texture, and constant memory, by appropriately sizing the thread granularity, unrolling loops, etc.
- **(5 points)** Provide a two/three page report describing how you designed your kernel(s) (use code or pseudocode if that helps the discussion) and the amount of speedup obtained over the serial version. Ignore the CPU/GPU communication overhead.