CS4961: Parallel Programming Final Take-Home Quiz
December 7, 2011
Due December 15, 2011

Instructions:
This is a take-home open-book, open-note exam. The format is that of a long homework. The exam is due by 11:59PM, Thursday, December 15. Use the CADE handin program, as follows:

`handin cs4961 final <file>`
or drop off in my office no later than December 15.

The goal of the exam is to reinforce your understanding of issues we have studied in class.
CS4961: Parallel Programming
Final Take-Home Quiz
December 8, 2011

I. Definitions (10 points)
Provide a very brief definition of the following terms:

a. MPI_Isend
b. Scatter (in MPI)
c. Divergent branches
d. Sparse matrix
e. One-sided communication

II. Short answer (20 points) [Fill in the blanks]
a. In CUDA, we can synchronize a collection of threads using a ________________
   construct. This type of synchronization, where a thread waits to proceed until all
   threads have reached this point, is called a ________________.

b. List two reasons why it is useful for a parallel programming model to have a
   “reduction” construct in the language.

  c. What was the key construct in MPI used in the tree search algorithm of Chapter 6
     to determine the new best cost, and why was it used?

  d. In MPI, ________________ communication involves transmitting a message between
     a pair of processes, and ________________ communication involves all the processes
     in a communicator.

III. Problem Solving (50 points)
In this set of five questions, you will be asked to provide code solutions to solve
particular problems. This portion of the exam may take too much time if you write
out the solution in detail. I will accept responses that sketch the solution, without
necessarily writing out the code or worrying about correct syntax. Just be sure you
have conveyed the intent and issues you are addressing in your solution.
a. Take a look at the conjugate gradient code CG from the NAS Parallel Benchmarks. The serial cg-seq.f and MPI code cg-mpi.f are found off the class website adjacent to the link for this exam. They are in Fortran, but you should be able to figure it out even if Fortran is unfamiliar to you. The main computation can be found in the function conj_grad in both versions. (i) Identify the loop nest that performs a sparse matrix-vector multiply, and provide its code below;

(ii) There are eight occurrences in the MPI version of this function of the same sequence of MPI calls. Explain what this sequence is doing.

b. (i) The following 5x5 matrix is represented in a dense form. Convert it to a CSR representation (compressed sparse row) format.

\[
A = \begin{bmatrix}
2 & 0 & 0 & 1 & 0 \\
0 & 3 & 0 & 1 & -1 \\
0 & 5 & 0 & -4 & 0 \\
2 & 0 & 6 & 0 & 0 \\
0 & 0 & 0 & -2 & 1 \\
\end{bmatrix}
\]

(ii) Provide a dense matrix representation of the following undirected graph.

(iii) Now provide a sparse representation of the graph, either as a sparse matrix as in (i), or an adjacency list.
c. (Problem 6.25 on page 349 of textbook) Pointers cannot be sent by MPI programs since an address that is valid on the sending process may cause a segmentation violation the receiving process, or perhaps worse, refer to memory that’s being used by the receiving process (because the two processes have their own separate address space). There are a couple of alternatives that can be used to address this problem:

(i) The object that uses pointers can be packed into contiguous memory by the sender and unpacked by the receiver.
(ii) The sender and the receiver can build MPI derived datatypes that map the memory used by the sender and valid memory on the receiver.

Sketch two Send_linked_list functions and two matching Recv_linked_list functions using (i) and (ii) above. Note that for (ii) you may need to send two messages: the first will tell the receiver how many nodes are in the linked list, and the second will send the actual list.

d. For the following sequential computation, your goal is to map it to a CUDA implementation. This computation is a 2-dimensional Jacobi relaxation on an NxN grid, where N is unknown; for simplicity, you can assume N is evenly divisible by 32.

```c
Jacobi(float *a, float *b, int N) {
    for (i=1; i<N-1; i++) {
        for (j=1; j<N-1; j++) {
            a[i][j]=0.8*(b[i-1][j]+b[i+1][j] + b[i][j-1] + b[i][j+1]);
        }
    }
}
```

Assume that you are parallelizing the outer two loops, and each thread computes just one element. Your goal is to parallelize it such that you use a cyclic distribution in both the x-dimension and y-dimension of threads. Write the CUDA code for `Jacobi_GPU`. You will need to come up with the appropriate indexing functions in CUDA that results in this distribution.

```c
dim3 dimGrid(N/32,N/32);
dim3 dimBlock(32,32);
Jacobi_GPU<<<dimGrid,dimBlock>>>(A,B,N);
```

e. For the Jacobi computation above, suppose N was very large, and you were optimizing it for locality in cache and registers. Describe the set of optimizations you would apply to this code.
III. Essay Question (20 points)
Please write a brief essay to answer the following question: What are the five most important lessons you learned in this course from your programming assignments or your final project? In addition to listing the lessons, please write a sentence expanding on each.