CS4961: Parallel Programming Final Take-Home Quiz
December 7, 2012
Due December 13, 2012

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 13. Use the CADE handin program, as follows:

handin cs4230 final <file>

or drop off in my office no later than December 13.

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 7, 2012

I. Short answer (20 points) [Fill in the blanks]

a. In CUDA, we can improve bandwidth to global memory by _____.
We can improve bandwidth to shared memory by _____.

b. List three programming model language features and briefly list how these are similar/different in OpenMP, MPI and CUDA.

c. In MPI, give an example of how to overlap communication with computation.

d. In MPI, a reduction (MPI_Reduce) is a form of _____ communication.

II. General Knowledge (10 points)

What are some types of algorithms or features of algorithms that lead to load imbalance, and what are ways in which load imbalance can be addressed in a parallel program?

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. (3.9 on page 141 of textbook) Write an MPI program (in pseudo code) for the histogram program from Section 2.7.1 of the textbook. The main loop of the computation looks like this:

```
for (i=0; i<data_count; i++) {
    bin = Find_bin(data[i], bin_maxes, bin_count, min_meas);
    bin_counts[bin]++;
}
```

where bin_maxes is an array of the maximum value of the elements in each bin, bin_counts is an array of the number of elements in each bin, and Find_bin returns in which bin a data element belongs. Your MPI code needs to compute final values for bin_maxes and bin_counts for each bin. Make sure your MPI constructs are fleshed out, while the rest can be pseudo-code.

b. (i) The following 6x6 matrix is represented in a dense form. Convert it to an ELL representationß (see lecture notes), which capitalizes on the fact that there is a known upper bound on the number of nonzeros per row.

$$A = \begin{matrix} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 1 & -1 & 2 \\ 0 & 5 & 0 & -4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{matrix}$$

(ii) As you used in your programming assignment, the CSR implementation of sparse matrix-vector multiplication looks like the following.

```
for (i=0; i<nr; i++) {
   for (j = ptr[i]; j<ptr[i+1]; j++) {
     t[i] = t[i] + data[j] * b[indices[j]];
   }
 }
```

What would the code look like for ELL?

(iii) Compare the complexity of the code in each case and the size of the resulting data structures, for CSR and ELL. Specifically, how does the additional structure of ELL improve the code, and at what cost in storage? When would ELL be preferable and when would CSR be preferable?

c. (Problem 6.13 on page 344 of textbook) Using Figure 6.6 as a guide (see below), sketch the communications that would be needed in an obvious MPI implementation of the reduced n-body solver if there were three processes, six particles, and the solver used a cyclic distribution of the particles.

d. For the following sequential computation of matrix-matrix multiplication, your goal is to map it to a CUDA implementation. Assume that you are parallelizing the outer two loops, and each thread computes just one element corresponding to c[i][j]. NxN is unknown, but evenly divisible by 16.

```
for (i=0; i<N; i++) {
   for (j=0; j<N; j++) {
      for (k=0; k<N; k++) {
            c[i][j] += a[i][k]*b[k][j];
      }
   }
}
```

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 MM_GPU. You will need to come up with the appropriate indexing functions in CUDA that results in this distribution.

```
dim3 dimGrid(N/16,N/8);
dim3 dimBlock(16,8);
MM_GPU<<<dimGrid,dimBlock>>>(A,B,C,N);
```

e. Suppose you were parallelizing the following image correlation to a CUDA implementation. (DO NOT ACTUALLY WRITE THE CUDA CODE BECAUSE IT IS A BIT COMPLEX.) List the steps of parallelizing the computation, paying attention to loop bounds, dependences and opportunities for data reuse in the memory hierarchy. If you miss some of the steps, you can still get partial credit for the ones you get right.

It is an NxN image, where N is unknown, but is evenly divisible by 8. The templates are 8x8.

```
#define TEMPLATE_NROWS   8
#define TEMPLATE_NCOLS     8
Correlation(float *th, float *image, int N) {
for (k=0; k<N-TEMPLATE_NROWS+1;k++) {
  for (l=0; l<N-TEMPLATE_NCOLS+1; l++) {
     for (i=1; i<TEMPLATE_NROWS-1; i++) {
       for (j=1; j<TEMPLATE_NCOLS-1; j++) {
            if (mask[i][j] != 0) {
             th[k][l] += image[i+k][j+l};
  }
 }
}
```

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.