CIS 631: Parallel Processing

Prof. Allen D. Malony

Midterm – May 21, 2007

NAME: ____________________________

<table>
<thead>
<tr>
<th>Section</th>
<th>Total Points</th>
<th>Points Scored</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Parallel Architecture</td>
<td>5, 5, 5, 5, 5, 5</td>
<td>/ 30</td>
</tr>
<tr>
<td>2. Parallel Algorithms</td>
<td>5, 5, 5, 5, 5, 20, 20</td>
<td>/ 70</td>
</tr>
<tr>
<td>3. Parallel Programming</td>
<td>5, 5, 5, 5, 20, 20</td>
<td>/ 70</td>
</tr>
<tr>
<td>4. Performance Analysis</td>
<td>5, 5, 5, 5, 20, 20</td>
<td>/ 70</td>
</tr>
<tr>
<td>Total</td>
<td>240</td>
<td>/ 240</td>
</tr>
</tbody>
</table>

Comments:

1. Half of the total available points are from short answer questions. Make sure you try these.
1 Parallel Architecture (30)

1.1 Short Answer (5,5,5,5,5,5)

a) Name three advantages of a shared memory parallel system over a distributed memory parallel system. Name three advantages of a distributed memory parallel system over a shared memory parallel system.

b) What does it mean that a parallel machine architecture is a UMA, NUMA, and CC-NUMA architecture? From an architecture evolution point of view, why was each type of architecture developed?

c) Why do we have multi-core processors? Are not single-core processors good enough? Why is multi-core a disruptive technology from the point of view of parallel computing?
d) Can a message passing parallel system be built from a CC-NUMA parallel machine? Explain how you would do it.

e) For large-scale parallel systems, the interconnection network is key. Would you agree? Explain.

f) What prospects do you see for heterogeneous parallel systems? What are the challenges?
2 Parallel Algorithms (70)

2.1 Short Answer (5,5,5,5,5,5)

a) List three important considerations for good parallel algorithm design.

b) What is a “hybrid” parallel algorithm? What is an “adaptive” parallel algorithm?

c) Does it ever make sense to increase overhead in order to get a better parallel algorithm? Why or why not?
d) Why is divide-and-conquer an organize-by-task pattern? Does it naturally leads to a data decomposition?

e) When is a pipeline pattern, as an algorithm structure, best suited and what are its advantages?

f) How is Google able to respond to search queries so quickly?  
(Hint: see MapReduce, http://en.wikipedia.org/wiki/MapReduce).
2.2 Matrix Transpose (20)

Consider a $N \times N$ matrix that is block decomposed across $P$ processors. Assume $N \times N$ is a multiple of $P$ and, thus, each processor is assigned $\frac{N}{\sqrt{P}} \times \frac{N}{\sqrt{P}}$ elements. (Assume $P$ has an integer square root.) For some numerical applications, it is necessary to transpose a matrix for certain calculations. Given a matrix $A$, its transpose, $B = A^T$, is defined on an element by element basis as $b[j][i] = a[i][j]$, for $1 \leq i, j \leq N$.

Sketch out a parallel algorithm to transpose a matrix $A$ that is block decomposed on $P$ processors on a distributed memory machine. Your solution should be such that if processor $p$ was assigned $a[i][j]$ before the transpose, $b[i][j]$ will also be assigned on $p$. 
2.3 Component Labeling (20)

The component labeling problem in image processing is defined as follows. We are given a two-dimensional array of pixels valued 0 or 1. The 1-pixels must be labeled in such a way that two pixels have the same label if and only if they are in the same connected component. Two 1-pixels are in the same connected component if there is a path of contiguous 1-pixels linking them together. Two pixels are contiguous if they are adjacent vertically or horizontally. Sketch out a parallel algorithm to solve the component labeling problem assuming that you have a task for each pixel. (Hint: First, assign each 1-pixel a unique label.).
3 Parallel Programming (70)

3.1 Short Answer (5,5,5,5,5,5)

a) Why are Intel and AMD now so concerned with parallel programming? Is not programming for multi-core processors the same as programming for shared-memory multiprocessor systems of the last 5 years?

b) What advantages might there be in programming a parallel machine in a “bulk synchronous” parallel fashion?

c) Why are MPI and OpenMP important parallel programming technologies? Discuss the advantages and disadvantages of the two styles.
d) In shared memory parallel programming, does it ever make sense to create more threads than the number of available processors to use in a computation? Explain why or why not.


e) Contrast the master/worker and fork/join program structuring patterns. Are not they really the same?


f) What are the benefits of blocking message communication? Why not use non-blocking message communication all the time?
3.2 OpenMP and Pthreads (20)

OpenMP compilers work by interpreting the OpenMP directives in the program and generating a new multi-threaded program. This program interfaces with the OpenMP runtime library during parallel execution to implement the OpenMP directives used. You can think of an OpenMP compiler functioning as a source-to-source translator, rewriting the sequential program (plus OpenMP directives) to create threads of execution and control their operation. Certainly, compilers can perform sophisticated transformations on the program to make it run more efficiently, but the translation of basic OpenMP constructs is relatively straightforward.

Pretend you are an OpenMP compiler. Figure 1 below shows a skeleton of an OpenMP program in the top box and a partial translation of the program in the lower box. Fill in the white boxes with the appropriate thread code to implement the desired OpenMP functionality. Please show real code (to the extent possible) and not pseudo code.

```c
int a, b;
main() {
    // serial segment
    #pragma omp parallel num_threads (8) private (a) shared (b)
    {
        // parallel segment
    }
    // rest of serial segment
}

int a, b;
main() {
    // serial segment
    for (i=0; i<8; i++)
        // parallel segment
}
    // rest of serial segment
}
```

Figure 1: OpenMP Translation to Multi-threaded Program.
3.3 Matrix Transpose Revisited (20)

Write a MPI program for the matrix transpose problem in Section §2.2.
4 Performance Analysis (70)

4.1 Short Answer (5,5,5,5,5,5)

a) In the context of parallel computing, “Amdahls’ Law” is regarded as applying to computations of “fixed” size. Define Amdahl’s Law respect to this concept.

b) What is meant by the following terms:

- *hot spot*

- *bottleneck*

- *critical path*

c) Why is the ratio of computation time versus communication time important for performance in distributed memory message passing systems? Is this metric useful when you can have overlapped computation and communication?
d) Is it ever reasonable to calculate speedup for $P$ processors with respect to the execution time for $P = 2$ processors? Explain.

e) Given two algorithms, $A$ and $B$, whose parallel speedups are $S_{p}^{A}$ and $S_{p}^{B}$, respectively, under what conditions can you claim $A$ is a “faster” algorithm than $B$?

f) For a given problem size, why does the efficiency go down as the number of processing elements increase? Is this always true?
4.2 Speedup, Efficiency, and Isoefficiency (20)

Consider the problem of computing the dot product of two vectors, A and B, each of length n. The dot product is defined as:

\[
A \bullet B = \sum_{i=1}^{n} a_i \cdot b_i , 1 \leq i \leq n
\]  

(1)

a) Describe how you would parallelize this problem.

b) Assume that multiplying two numbers takes 5 units of time, adding two numbers takes 2 units of time, and communicating one number between two processing elements takes 10 units of time. What is the parallel runtime, speedup, and efficiency of your parallel algorithm when run on p processing elements? You can assume n and p are a power of 2. If you can, write your answer in terms of computation time and communication time components.
c) Calculate the speedup and efficiency assuming that the problem for $p = 1$ is that of computing the dot product for two vectors of length 256. Use $p = 1, 4, 16, 64, \text{ and } 256$, and assume the same time costs as in b).

<table>
<thead>
<tr>
<th>$p$</th>
<th>Speedup</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

d) When executing on 64 processors, how large would $n$ have to be to achieve the same efficiency as achieved on 4 processors for $n = 256$?

e) Scaled speedup is defined as the speedup obtained when the problem size is increased linearly with the number of processing elements. That is, if $W$ is chosen as a base problem size for a single processing element, then

$$Scaled \ speedup = \frac{pW}{T_p(pW, p)}$$

For the dot product problem, calculate the scaled speedups, assuming that the base problem for $p = 1$ is that of computing the dot product for two vectors of length 256. Use $p = 1, 4, 16, 64, \text{ and } 256$, and assume the same time costs as before.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Scaled Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
</tr>
</tbody>
</table>
4.3 DAG Model of Parallel Computation (20)

Parallel algorithms can often be represented by dependency graphs. Four such dependency graphs are shown in Figure 2. If a program can be broken into several tasks, then each node of the graph represents one task. The directed edges of the graph represent the dependencies between the tasks or the order in which they must be performed to yield correct results. A node of the dependency graph can be scheduled for execution as soon as the tasks at all the nodes that have incoming edges to that node have finished execution. For example, in Figure 2 (b), the nodes on the second level from the root can begin execution only after the task at the root is finished.

Any deadlock-free dependency graph must be a directed acyclic graph (DAG) (i.e., devoid of cycles). All the nodes that are scheduled for execution can be worked on in parallel provided enough processing elements are available. If $N$ is the number of nodes in a graph, and $n$ is an integer, then $N = 2^n - 1$ for graphs (a) and (b), $N = n^2$ for graph (c), and $N = n(n + 1)/2$ for graph (d). (Graphs (a) and (b) are drawn for $n = 4$, and graphs (c) and (d) are drawn for $n = 8$.)

Assuming that each task takes one unit of time and that interprocessor communication time is 0, for the algorithms represented by each of these graphs:

1. Compute degree of concurrency.

2. Compute the maximum possible speedup if an unlimited number of processing elements is available.

3. Compute the values of speedup, efficiency, and the overhead function if the number of processing elements is (i) the same as the degree of concurrency and (ii) equal to half of the degree of concurrency.
Figure 2: Dependency Graphs