Date:_____

Quiz for Chapter 7 Multicores, Multiprocessors, and Clusters

Not all questions are of equal difficulty. Please review the entire quiz first and then budget your time carefully.

Name:_____

Course:_____

1. [5 points] Applying the send/receive programming model as outlined in the example in Section 7.4, describe a parallel message passing program that uses 10 processor-memory nodes to multiply a 50x50 matrix by a scalar value, c.

Name:_____

2. [10 points] Consider the following GPU that consists of 8 multiprocessors clocked at 1.5 GHz, each of which contains 8 multithreaded single-precision floating-point units and integer processing units. It has a memory system that consists of 8 partitions of 1GHz Graphics DDR3DRAM, each 8 bytes wide and with 256 MB of capacity. Making reasonable assumptions (state them), and a naive matrix multiplication algorithm, compute how much time the computation C = A * B would take. A, B, and C are n * n matrices and n is determined by the amount of memory the system has.

Name:_____

3. [5 points] Besides network bandwidth and bisection bandwidth, two other properties sometimes used to describe network typologies are the diameter and the nodal degree. The diameter of a network is defined as the longest minimal path possible, examining all pairs of nodes. The nodal degree is the number of links connecting to each node. If the bandwidth of each link in a network is B, find the diameter, nodal degree, network bandwidth, and bisection bandwidth for the 2D grid and n-cube tree shown in Figure 7.9.

Name:_____

4. [5 points] Vector architecture exploits the data-level parallelism to achieve significant speedup. For programmers, it is usually be make the problem/data bigger. For instance, programmers ten years ago might want to model a map with a 1000 x 1000 single-precision floating-point array, but may now want to do this with a 5000 x 5000 double-precision floating-point array. Obviously, there is abundant data-level parallelism to explore. Give some reasons why computer architecture do not intend to create a super-big vector machine (in terms of the number and the length of vector registers) to take advantage of this opportunity?

Name:_____

5. [5 points] Why should there be stride-access for vector load instruction? Give an example when this feature is especially useful.

Name:_____

6. [10 points] A two-part question.

(a) What are the advantages and disadvantages of fine-grained multithreading, coarse-grained multithreading, and simultaneous multithreading?

Name:_____

(b) Given the following instruction sequence of three threads, how many clock cycles will fine-grained multithreading, coarse-grained multithreading use respectively? Annotations are the same as in Figure 7.5

Thread 1:

```
[]  []  []
[]  []  []  []
[]  []
[]
[]  []  []
(stall)
(stall)
[]  []
(stall)
[]  []  []
```

Thread 2:

```
[]  []
[]  []  []
(stall)
[]  []  []  []
[]
[]
[]  []
(stall)
[]
```

Thread 3:

```
[]
[]  []
[]  []  []
(stall)
[]  []
[]
[]  []  []  []
[]  []  []
[]
```

Name:_____

7. [10 points] Consider a multi-core processor with 64 cores where first shared cache is at level L3 (L1 and L2 are private to each core). Suppose an application needs to compute the sum of all the nodes in a perfectly balanced binary tree T (A tree where every node has either two or zero children and all the leaves are at the same depth/level). Assuming that an add operation takes 10 units of time, the total sum can be computed sequentially, in time 10*n units, ignoring the time to load and traverse links in the tree (assume they are factored in the add). Here n is the number of nodes in T. One way to compute the sum in a parallel manner is to have two arrays of length 64: (a) an input array having the roots of 64 leaf subtrees (b) an output array that holds the partial sums computed for the 64 leaf subtrees for each core. Both the arrays are indexed by the id (0 to 63) of the core that is responsible for that entry. Furthermore, assume the following:

- The input arrays are already filled in with the roots of the leaf subtrees and are in the L1 caches of each core.
- Core 0 is responsible for handling the internal nodes that are not a part of any of the 64 subtrees. It is also responsible for computing the final sum once the partial sums are filled in.
- Every time a partial sum is filled in the output array by a core, another core can fill in its partial sum only after a minimum delay of 50 units (due to cache invalidations).

In order to achieve a speedup of greater than 2 (over sequential code), what is the minimum number of nodes that should be present in the tree?

Name:_____

8. [10 points] Consider a multi-core processor with heterogeneous cores: A, B, C and D where core B runs twice as fast as A, core C runs three times as fast as A and cores C and A run at the same speed (ie have the same processor frequency, micro architecture etc). Suppose an application needs to compute the square of each element in an array of 256 elements. Consider the following two divisions of labor:

(a)

Core A	32 elements
Core B	128 elements
Core C	64 elements
Core D	32 elements

(b)

Core A	48 elements
Core B	128 elements
Core C	80 elements
Core D	Unused

Compute (1) the total execution time taken in the two cases and (2) cumulative processor utilization (Amount of total time the processors are not idle divided by the total execution time). For case (b), if you do not consider Core D in cumulative processor utilization (assuming we have another application to run on Core D), how would it change? Ignore cache effects by assuming that a perfect prefetcher is in operation.

Name:_____

9. [10 points] Consider a system with two multiprocessors with the following configurations:

(a) Machine 1, a NUMA machine with two processors, each with local memory of 512 MB with local memory access latency of 20 cycles per word and remote memory access latency of 60 cycles per word.

(b) Machine 2, a UMA machine with two processors, with a shared memory of 1GB with access latency of 40 cycles per word.

Suppose an application has two threads running on the two processors, each of them need to access an entire array of 4096 words, is it possible to partition this array on the local memories of the NUMA machine so that the application runs faster on it rather than the UMA machine? If so, specify the partitioning. If not, by how many more cycles should the UMA memory latency be worsened for a partitioning on the NUMA machine to enable a faster run than the UMA machine? Assume that the memory operations dominate the execution time.

Name:_____

10. [10 points] Consider the following code that adds two matrices A and B and stores the result in a matrix C:

```
for (i= 0 to 15) {
       for (j= 0 to 63) {
              C[i][j] = A[i][j] + B[i][j];
       }
}
```

If we had a quad-core multiprocessor, where the elements of the matrices A, B, C are stored in row major order, which one of the following two parallelizations is better and why ? What about when they are stored in column major order ?

(a) For each P_k in {0, 1, 2, 3}:

```
for (i= 0 to 15) {
       for (j= P_k*15  + P_k to (P_k+1)*15 + P_k)
      {
            // Inner Loop Parallelization
              C[i][j] = A[i][j] + B[i][j];
       }
}
```

(b) For each P_k in {0, 1, 2, 3}:

```
for (i= P_k*3  + P_k to (P_k+1)*3 + P_k) {
       // Outer Loop Parallelization
       for (j= 0 to 63) {
              C[i][j] = A[i][j] + B[i][j];
       }
}
```

Name:_____

11. [10 points] How would you rewrite the following sequential code so that it can be run as two parallel threads on a dual-core processor ? Try to balance the loads as much as possible between the two threads:

```
int A[80], B[80], C[80], D[80];
for (i = 0 to 40)
{
       A[i] = B[i] * D[2*i];
       C[i] = C[i] + B[2*i];
       D[i] = 2*B[2*i];
       A[i+40] = C[2*i] + B[i];
}
```

Name:_____

12. [10 points] Suppose we have a dual core chip multiprocessor with two level cache hierarchy: Both the cores have their own private first level cache (L1) while they share their second level cache (L2). The first level cache on both the cores is 2-way set associative with cache line size of 2K bytes, and access latency of 30ns per word, while the shared cache is direct mapped with cache line size of 4K bytes and access latency of 80ns per word. Consider a process with two threads running on these cores as follows (assume the size of an integer to be 4 bytes which is same as the word size):

Thread 1:

```
int A[1024];
for (i=0; i < 1024; i++)
{
A[i]  = A[i] + 1;
}
```

Thread 2:

```
int B[1024];
for (i=0; i< 1024; i++)
{
B[i] = B[i] + 1;
}
```

Initially assume that both the arrays A and B are in main memory, whose access latency is 200ns per word. Assume that an int is word sized. Furthermore, assume that A and B when mapped to L2 start at address 0 of a cache line. Assume a write back policy for both L1 and L2 caches.

(a) If the main memory blocks having arrays A and B map to different L2 cache lines, how much time would it take the process to complete its execution in the worst case? (Assuming this is the only process running on the machine.)

(b) If the main memory blocks having arrays A and B map to the same L2 cache line, how much time would it take the process to complete its execution in the worst case? (Assuming this is the only process running on the machine.)

In the worst case, thread 1 could access A[0], thread 2 could access B[0], then thread 1 could access A[1] followed by B[1] access by thread 2 and so on. Every time A[I] or B[i] is accessed, it evicts the other array from L2 cache and so a subsequent access to the other array has to again cause a main memory access.