CMU 15-418/618 Exam 2 Practice Problems

Consistency

Problem 1. (X points):

Consider the following code sequences running on three processors using either sequential consistency (SC) or relaxed consistency (RC). Assume that the lock is initially free, and that all data locations are initially zero. Variables in upper case are shared variables on their own cache lines, and `r1` through `r4` are registers.

```
   P1                      P2
   --                      --
LOCK(L1);                  while X == 0;
X = 1;                     LOCK(L1);
Y = 1;                     r1 = X;
UNLOCK(L1);                r2 = Y;
                           r3 = X;
                           UNLOCK(L1);
```

A. What are the possible outcomes for (`r1`,`r2`,`r3`,`r4`) under the following memory consistency models:

   (a) **Sequential Consistency:**

   (b) **Relaxed Consistency:**

B. Your partner proposes to remove the lock / unlock statements in P1 and P2. Do your answers to the previous question change? Why or why not?

Miscellaneous Questions

Problem 2. (X points):

A. You are working on parallelizing a matrix-vector multiplication, and try creating a result vector for each thread (p). Your code then has each thread sum a subset (n/p) of rows (n) out of each result vector. However, you find that the performance is unchanged as you increase the number of threads. What algorithmic flaw is present in this approach?

B. Recall that the semantics of `cilk_spawn` are that the spawned function is executed in a logically asynchronous thread of control that *may or may not* run in parallel with execution of the caller. Also recall that the *implementation of Cilk work scheduling* always runs the child (spawned) function first.

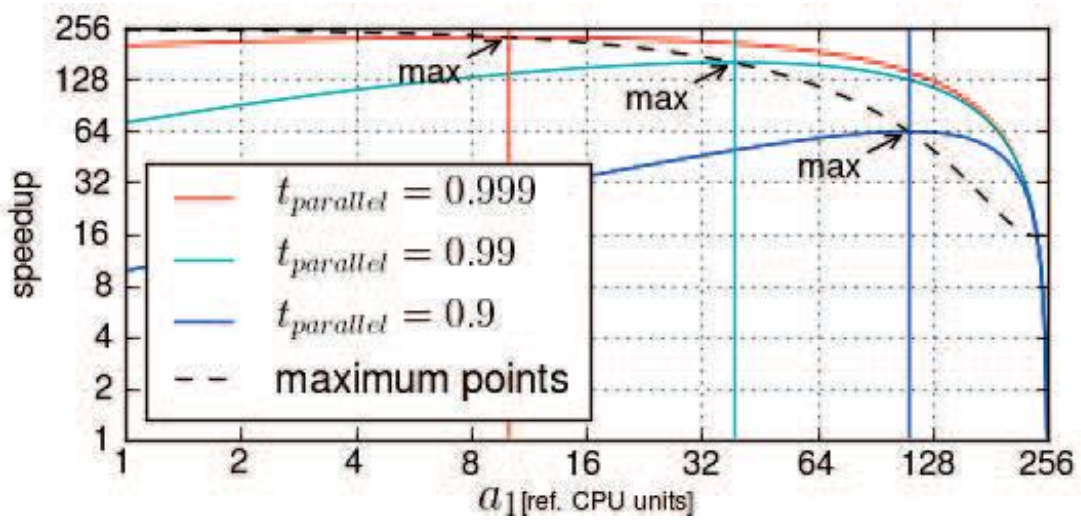
Consider the following two implementations of a parallel for loop in Cilk.

```
/* implementation 1 */
for (int i=0 i<N; i++)
   cilk_spawn myfunction(i);

/* implementation 2 */
void genwork(int start, int end) {

  while (start < end - GRANULARITY) {
    int mid = (end + start) / 2;
    cilk_spawn genwork(start, mid);
    start = mid;
  }

  for (int i=start; i<end; i++)
    myfunction(i);
}

genwork(0, N);  // launch all work
```

Give one reason why implementation 2 might give better performance than implementation 1 on a parallel machine. (Your answer should be precise and refer to how the two programs would be scheduled by the Cilk runtime.)

C. Graph algorithms often have low arthimetic intensity. Desribe a cause and a possible mitigation.

D. The following figure shows the speedup of parallel workloads with asymmetric processors. Along the black dashed line are the maximum speedup for different levels of parallelism. Describe how the maximum speedup points relate between the percentage of core area devoted to the "fat" core and to the percentage of parallelism of the program.



E. In the quake simulation, one of the approaches was LMV, using locks to protect the result vector as each thread *added* its results. At the time, they explored increasing the number of locks, such that each lock would protect smaller chunks of the vector. You are asked to port this code to the Xeon PHI with a vector length of 1 billion. How can you minimize the overhead and contention to sum the results?

F. What is the motivation for a content distribution network (CDN)? Why do Facebook images get served to your browser by a CDN but your main news feed is not?

G. **Domain-Specific Programming on Graphs:** Recall that during our lecture on graph DSLs, we discussed the fact that the size of the graph was a major concern. Assume that 32 bytes of data is associated with each vertex, and that 32 bytes of data is associated with each edge. Which of the following statements is more likely to be true?

A: The total amount of *edge* data is probably much larger than the total amount of vertex data.

B: The total amount of *vertex* data is probably much larger than the total amount of edge data.

Enter your answer here: _____

H. Give justification for your answer above.

Synchronization

Problem 3. (X points):

A. What is the ABA probem that can occur in lock-free data structure implementations? How is that problem corrected?

B. Instead of using atomic compare-and-exchange, some architectures support LL/SC. Would a lock-free data structure using LL/SC be suspectible to the ABA problem? Why or why not?

C. What coherence state(s) are required for executing an atomic test-and-set? Why?

Interconnection Networks

Problem 4. (X points):

A. Imagine you are designing a parallel machine with P processors to execute convolutions in parallel on a 2D array. (Recall each array element is updated to be a weighed combination of all neighboring values). Do you advocate for a mesh or torus interconnect for this system? Why? (Hint: think about performance/cost trade-offs)

B. Consider the difference between store-and-forward and cut-through flow control in a chip interconnect. Although cut-through flow control will typically reduce packet transmission latency, its behavior degenerates to that of the store-and-forward scheme under high contention. Why?