Synchronization, Communication, and Scheduling in Parallel Systems

Review Questions

Instructions: You are not required to turn in answers to these review questions, but you should do them on your own after you read this section's papers to test your knowledge. Use Piazza or the TAs if you have questions or get stuck on any of these.

```
Synchronization Algorithms (MCS paper)
--------------------------------------

1.  There is a bug in MCS barrier algorithm on Page 19 of the paper.
    First 10 folks to (independently) report the bug and the fix correctly
    (no credit if the bug reported OR the fix reported is incorrect)
    correctly to the TA will get 1% credit towards the course total
    of 100%. (Send him an email with the subject "CS 6210 MCS Bug".)

2.  The MCS paper claims that the MCS algorithm may be the best for
    large-scale cache coherent multiprocessors.  Argue why this may
    not be necessarily true. [Hint: See the discussion of barrier
    algorithm implementation and resulte in pages 8-10 of the
    "scalability study of KSR-1" paper which is a suggested reading
    for project 2.

3.  MCS paper shows results that MCS barrier does better than tournament
    on a bus-based shared memory machine like the Sequent Symmetry.
    Explain why this is so.

4.  Anderson lock algorithm uses arrays and MCS algorithm uses linked list,
    but both are variants of maintaining a FIFO queue of lock requests.

    (a) Comment on the space requirements of Anderson algorithm versus MCS
        algorithm.
    (b) Which of the two variants will work better on a shared
        memory machine that does not support hardware cache coherence?  Why?


LRPC paper
----------

1. Come up with the data structures and pseudo code that the kernel has
   to maintain for LRPC.

2. Assume a server exports a function:
       int foo(int a[30]; float b[20]);
   The server allows a max of 10 simultaneous calls per client.

   a)  Sketch the details of an
          import call from a client with respect to foo assuming the
          call succeeds
   b)  Sketch the details of an actual call from the client to the
       procedure foo clearly detailing what interaction goes through the
       kernel and what interactions go directly happen between the
```

client and the server.

3. LRPC on a multiprocessor suggests having multiple domains pre-loaded into different processors of an SMP to service client calls. Why was this a good choice for the platform (Firefly multiprocessor) that they targeted for their implementation.

4. Tornado paper argues that the LRPC approach (server domains on multiple processors waiting for work) would be prohibitive for today's multiprocessors. Justify their argument quantitatively assuming a modern SMP (larger caches, larger disparity between cache and memory access times) with an invalidation-based cache coherence protocol. Present your own arguments why this may still be a good decision in modern multiprocessors.
   [Hint: Come up with pathological examples where the LRPC results in performance win and performance loss assuming an SMP with an invalidation-based cache coherence protocol.]

Scheduling
---

1. Assume a multiprocessor with 4 processors; 10 tasks ready to run; Using the notation
   A{superfix task}{suffix processor}
   to denote affinity for a particular task to a particular processor,
   (a) come up with an execution history for tasks on each processor that would result in a different schedule under each of FCFS, FP, LP, MI, LMI (with only 2 associations per task), and LMR schedules.
   (b) rank the scheduled you come up with in terms of increasing variance for response times of the schedules.

2. Discuss in general the issues in multiprocessor scheduling such as load imbalance, cache effects, response time, and throughput. Also consider multiprogrammed workload versus multithreaded workload on how these play into scheduling decisions. Note these issues are beyond this specific paper and would require some reading of the background material as well as drawing from other papers in this section.

3. Looking at the throughput results (Figure 6), you will notice that MI does better than FP for light load (48 tasks) while it does worse than FP for heavy load (128 tasks) with increasing reload time. Explain.

4. Come up with data structures and pseudo-code for each of FCFS, FP LP, MI, LMI, and LMR schedulers.

5. We see a recurring theme in several papers that we studied, namely, greediness does not pay. Explain how this is borne out in
   (a) the design of synchronization algorithms
   (b) multiprocessor scheduling
   (c) the design of network protocols such as Ethernet

CMT
---

1. What is hardware multithreading setting out to do and how does that impact
multiprocessor scheduling decisions in a CMT (Multithreaded Chip Multiprocessor)?

2. Discuss the solution proposed by Fedorova et al. for addressing the above scheduling problem in a CMT.

Tornado
---

1. Starting with Figure 3, first discuss for each object in the path of the memory manager which clustering strategy (a single shared object, partial replication, full replication, partitioned reps) is most appropriate.

2. Building on Q1, develop the high level data structures and pseudo-code for each of the clustered objects that implement a memory manager. Explicitly state the level of contention for the reps in your design (i.e. is it across the entire system, a cluster or processors, threads running on the same processor, etc.)? Based on this discussion argue the scalability of your implementation.

3. (a) As As an OS designer for an SMP, present key objectives you have to pay attention to.

   (b) Mention properties of a clustered object that help meet the above objectives.

   (c) We want to design a processor scheduler for an SMP. Demonstrate a concrete use case of clustered objects for designing the scheduler.

Cellular Disco
---
1. A user process running in a VM on top of cellular Disco makes a blocking fread() system call.
Discuss the sequence of actions until the process is resumed.

2. Discuss the approach to fault containment in Cellular Disco.

Corey
---
1. Discuss the difference between the region concept in Tornado and address range concept in Corey.

2. Discuss the motivation for each of the three abstractions: address range, Shares, and Kernel cores from the point of view of many-core architectures.