James Scherick Due 8 December 2022

Midterm 3

Professor: Maurice Herlihy

CS1760: Multiprocessor Synchronization

Please solve *exactly* 3 out of the 4 questions (if you solve more, please specify which 3 you would like us to check).

This is a closed book exam. Please do not consult the textbook, other humans, or the internet.

Good luck!

Problem 1. Let $P(x) = \sum_{i=0}^{d} p_i x^i$ and $Q(x) = \sum_{i=0}^{d} q_i x^i$ be polynomials of degree d, where d is a power of 2. We can write

$$P(x) = P_0(x) + (P_1(x) \cdot x^{d/2})$$
$$Q(x) = Q_0(x) + (Q_1(x) \cdot x^{d/2})$$

where $P_0(x), P_1(x), Q_0(x)$, and $Q_1(x)$ are polynomials of degree $d/2$.

The Polynomial class provides constant-time put() and get() methods to access coefficients as well as a constant-time split () method that splits a d-degree polynomial $P(x)$ into the two $(d/2)$-degree polynomials $P_0(x)$ and $P_1(x)$ as defined above.

Your task is to analyze the work and critical path lengths of parallel addition and multiplication algorithms decomposed as follows.

The *sum* of $P(x)$ and $Q(x)$ is decomposed as:

$$P(x) + Q(x) = (P_0(x) + Q_0(x)) + (P_1(x) + Q_1(x)) \cdot x^{d/2}.$$

The *product* of $P(x)$ and $Q(x)$ is decomposed as:

$$P(x) \cdot Q(x) = (P_0(x) \cdot Q_0(x)) + (P_0(x) \cdot Q_1(x) + P_1(x) \cdot Q_0(x)) \cdot x^{d/2} + (P_1(x) \cdot Q_1(x)) \cdot x^d$$

Give recurrences for:

1. Polynomial addition's *work*.
2. Polynomial addition's *critical path length*.
3. Polynomial multiplication's *work*.
4. Polynomial multiplication's *critical path length*.

You do not need to solve these recurrences, just state them.

Problem 2. In the following question you are required to implement mutual exclusion from specific types of concurrent objects. you may *not* use any additional read-write registers. Your solutions should be simple, and you may describe them using simple English sentences or pseudocode. Include a brief explanation why your proposals are correct.

a) An atomic FIFO (first-in, first-out) QUEUE supports linearizable, wait-free ENQ(x) and DEQ() methods for any number of threads.

   Implement a deadlock-free mutual exclusion algorithm from atomic FIFO QUEUE objects.

b) An atomic FIFO PEEKABLEQUEUE additionally supports a linearizable, wait-free PEEK() method which returns the next value to be dequeued without actually removing it.

   Implement a deadlock-free mutual exclusion algorithm that provides first-come-first-served fairness using atomic FIFO PEEKABLEQUEUE objects.

Problem 3. In multiple choice questions more than one of the answers may be correct.

1. (4 points) In which case are spin-locks clearly a better alternative than monitor locks:

   ☐ short critical sections.

   ☐ long critical sections.

   ☐ critical sections with shared variables.

   ☐ critical sections without shared variables.

2. (2 points) There is a wait-free linearizable implementation of a queue object for 2 dequeuer threads using read-write registers.

   ☐ True.

   ☐ False.

   Explain in one sentence.

3. (4 points) There is a wait-free linearizable implementation of a queue object for 1 enqueuer thread and 1 dequeuer thread using read and write registers.

   ☐ True.

   ☐ False.

   Explain in one sentence.

4. (4 points) The ABA problem can arise from:

   ☐ Failing to validate optimistic data structures

   ☐ Failing to signal a blocking thread when there is work to be done

   ☐ Allowing Multiple threads to acqure locks in different orders

   ☐ Reusing memory blocks

5. (4 points) The difference between *linearizability* and *sequential consistency* is:

   ☐ Linearizability imposes a real-time order.

   ☐ Sequential consistency imposes a real-time order.

   ☐ Sequential consistency applies to multiple objects.

   ☐ Linearizability is composable.

6. (2 points) Reentrant locks allow multiple threads to concurrently enter the critical sections.

   ☐ True.

   ☐ False.

7. (4 points) Which of the following techniques is a way to reduce contention:

   ☐ Lock striping.

- ☐ Exponential backoff.
- ☐ False sharing.
- ☐ Elimination array.

8. (4 points) One can use an atomic markable reference at the top of the lock-free stack implementation to overcome the ABA problem.

   - ☐ True.
   - ☐ False.

9. (4 points) Which of the following is a way to overcome the lost wake-up problem:

   - ☐ Signal all the threads.
   - ☐ Hold the locks while signaling.
   - ☐ Always signal waiting threads.
   - ☐ Do not hold the locks while signaling.

10. (4 points) Explain in one sentence why in optimistic synchronization after acquiring locks threads perform validation.

Problem 4. You may assume that the actual running time of a parallel program on a dedicated P-processor machine is

$$T_P = T_1/P + T_\infty.$$

Your research group has produced two chess programs, a simple one and an optimized one. The simple one has $T_1 = 2048$ seconds and $T_\infty = 1$ second. When you run it on your 32-processor machine, sure enough, the running time is 65 steps. Your students then produce an "optimized" version with $T'_1 = 1024$ seconds and $T'_\infty = 8$ seconds. When you run it on your 32-processor machine, the running time is 40 steps, as predicted by our formula.

Which program will scale better to a 512-processor machine?