CSMC 23010-1 Due May 21, 2019

Homework 3b

Professor: Hank Hoffmann *TA: Will Kong*

Design Review

We will do the design review in class for 3a and 3b at the same time (5/7/2019).

Theory

No theory problems for this assignment because this one has a shorter turnaround time.

Programming

In part A, we built a set of locks. We will now use them to balance the load of our packet distribution system.

To test load balancing we will extend the packet distribution system that we built in assignment 2 to improve the load balancing characteristics. In particular, we will protect the DEQUEUE interface to the Lamport queue in the previous assignment with a lock. The Dispatcher will be the only thread enqueueing data, so a lock is not necessary to protect the ENQUEUE interface; (*i.e.*, locking the DEQUEUE side gives the illusion of single-reader / single-writer semantics).

- SERIALPACKET This application features a single worker which calculates the checksums for each source serially. This version is configurable by:
  - NUMMILLISECONDS (M) the time in milliseconds that the experiment should run.
  - NUMSOURCES (n) the total number of sources (*i.e.* worker threads)
  - MEAN (W) the expected amount of work per packet
  - UNIFORMFLAG = TRUE for Uniform distributed packets, FALSE for Exponentially distributed packets
  - EXPERIMENTNUMBER non-negative integer, seeding the packet generator
- PARALLELPACKET This applications differs from PARALLEL in the previous assignment in that the Worker threads grab the lock that is associated with each queue before calling DEQUEUE. The Worker threads will also be designed to use any of the following strategies, S, for picking a queue:
  - LOCKFREE This is the same behavior as PARALLEL from the previous assignment (*i.e.* 1-to-1 mapping between queue and Worker).
  - HOMEQUEUE This is the same fixed mapping as in LOCKFREE, but with the added requirement that the worker grabs the (albeit uncontended) lock for the associated queue.
  - AWESOME Your design. Use your judgment to design a new strategy that will be faster. You can design a strategy for a specific case (*e.g.*, specific for uniform or exponential) or for a general case. You are responsible for describing your design, testing it, and implementing it. You are also responsible for forming a reasonable hypothesis about why your design is better. You are free to use any technique you wish in the workers. Do not change the dispatcher. Do not drop packets. More details follow.

  This code is configurable by the same parameters as SERIALPACKET, plus:

  - QUEUEDEPTH (D) queue depth (always equal to 8 for this assignment).

– lockType (L) the lock algorithm

– strategy (S) the strategy for picking a queue to work on

Experiment

You should use the basic components for this class (*e.g.* utilities for timing code, a random packet generator, a payload checksum calculator etc.).

Next, you will perform the following set of experiments across various cross-products of these parameters. Each data point is a measurement taken on a dynamic system (a computer...) and is thus subject to noise. As a result, some care should be taken to extract representative data—we would propose running some reasonable number of experiments for each data point and selecting the median value as the representative. For Uniform packets and the Counter tests, something like 5 data points should suffice whereas for Exponentially Distributed packets 11 may be required to get relatively smooth plots—please use your own engineering judgment to decide how many trials you require. You should set the *seed* parameter for PacketSource to the trial number (or some deterministic function thereof) to ensure that you're seeing a reasonable variation in load from each source. Setting the experiment time M to 2000 (*i.e.* 2 seconds) should suffice to warm up the caches for all of the following experiments. In each of the following experiments, we describe a plot that you should produce, analyze and discuss in the writeup.

A note on describing performance: Use quantitative language when describing your observations. For example, strategy x provides a 10% improvement compared to strategy y. Avoid qualitative language of the type, "Strategy x is a little better than strategy y."

A note on measuring performance: All code should be compiled with optimizations on (`-O3`, for example). You should describe what optimizations you are using. Remember, many lock implementations will work without optimizations and fail with them on. Such behavior usually means that some pointer was not appropriately declared to be volatile.

Packet Tests Choose a set of two locks. You should justify your choice. For example, you could choose the two fastest locks you implemented. You could choose the fastest one when threads $\leq$ cores and the fastest when threads $>$ cores. You could choose your best implementation and the pthread mutex. Essentially, choose two locks where you think the difference in the behavior you measured in part A might affect the results of part B. We will call the set of two locks you choose *Locks*. Note that if you have a bug in one of your locks, this choice allows you to stop using that lock and forget about trying to debug it. I believe everyone should at least have a TAS and pthread mutex working, so everyone can fulfill the requirements of this assignment even if 3a went off the rails.

1. **Idle Lock Overhead** Run ParallelPacket with $n = 1$, $S \in \{\text{LockFree}, \text{HomeQueue}\}$, all $L \in \textit{Locks}$ and $W \in \{25, 50, 100, 200, 400, 800\}$ on the uniformly distributed packets. Plot the speedup of ParallelPacket (with $S = \text{HomeQueue}$) throughput relative to ParallelPacket (with $S = \text{LockFree}$) for each L on the Y-axis vs. W on the X-axis. Is this consistent with the insights you drew in the Counter Tests? How does the **Worker Rate** compare with your measurements in assignment 2 at each W; *i.e.*, what is the overhead of the locks in terms of Worker Rate?

2. **Speedup with Uniform Load** Run ParallelPacket and SerialPacket on the uniformly distributed packets with the number of workers $n \in \{1, 2, 3, 7, 13, 27\}$, $W \in \{1000, 2000, 4000, 8000\}$, $S \in \{\text{LockFree}, \text{HomeQueue}\}$ and $L \in \textit{Locks}$. For each W (*i.e.* four graphs), make a plot of speedup (relative to SerialPacket with the same parameters) for each $\langle S, L\rangle$ combination on the Y-axis vs. $n + 1$ (which is the total number of threads) on the X-axis. How does the scalability of *ParallelPacket* change given the use of locks and load balancing? *Note:* $S = \text{LockFree}$ *does not need to be evaluated against all values of* L*—it doesn't use locks!* This answer should, again,

demonstrate understanding. Particularly, you should form a hypothesis about expected behavior before running the experiments. If your hypothesis does not hold up, you may need to reevaluate your hypothesis, rework your code, or change your experiment. In particular, at small values of M results may be distorted by randomness. You may consider increasing both M and the number of trials over which you take the median. Also note that what may appear to be small differences may be meaningful—stick to quantitative language.

3. **Speedup with Exponential Load** Repeat the previous experiment, except with the exponentially distributed packets. How does the scaling change? Why? Again, you should be able to argue that your results make sense.

4. **Speedup with Awesome** Demonstrate the speedup of your design under either uniform or exponential load or for whatever scenario you have designed it to outperform the given strategies. You should describe why your design is better and demonstrate its superior qualities with an experiment. Note that you have a lot of freedom to design here.

Submission

As with previous assignments, a design and test document should be submitted in class. Note that the major area for design in this project is the Awesome scheme that you are coming up with. It will not be possible to write a comprehensive design for this scheme as it should be based on what you learn from other experiments. The best thing to discuss for this scheme is how you will evaluate opportunities for improving on the other approaches. You should discuss correctness and performance testing for all modules in this assignment.

You are responsible for submitting a writeup and all code associated with this project.

You write up should be concise, but it should primarily demonstrate your understanding of the code and experiments and your engagement with the problems. The writeup should include quantitative language that clearly supports your statements. The writeup should stand alone as a document that explains the code you developed, the experiments you ran and the conclusions you drew. The writeup should have a section for each of the experiments described above and an additional section describing your test plan. You should have a test plan for each lock and for each of Last Queue and Awesome strategies.

The grades will be primarily based on the understanding demonstrated in the writeup, so please spend time on them. This means you need to start early and get help if there is a result you don't understand.

You should submit your working directory and all the associated code including test code. It should be easy for old people to run your tests and attempt to recreate your results (differences in machines might make this difficult, but in principle it should be doable).

You are responsible for understanding what is required of you on this assignment. If you are not clear about what to do, then ask clarifying questions.