cs378: Concurrency: Lab 1 Writeup Template

You
Department of Computer Science
UT Austin

August 22, 2018

1 Step 1: Unsynchronized Workers

1.1 Scalability of an Unsynchronized Counter

Using the Unix time utility, time and graph the runtime of your program with a maxcounter of 1000000 as a function of the number of workers from 1 (sequential) to twice the number of processor cores on your machine....

- Provide a graph of scalability similar to Figure **??**.
- List at least two reasons you can think of why performance gets worse as parallelism increases.
- If performance does not get monotonically worse list at least one reason why this might be the case.

1.2 Counting lost updates

Now change your program so that each worker counts the number of times it actually increments the shared counter variable. You can do this with a global array of per-worker counters indexed by say, worker id. This will allow you to measure lost updates as well as load imbalance. The ratio of lost updates to correct updates should simply be the total number of updates summed across all workers, divided by the maxcounter value. Without synchronization, this should be greater than 1. Create a graph of the lost update ratio as a function of worker thread count...

- Provide a graph of lost updates similar to Figure **??**.
- Does the number of lost updates surprise you or match your expectations?
- Does the trend match the scalability you observe, or differ?
- Why or why not?

1.3 Load Imbalance

...Each thread will differ from expectation differently so graph the average difference from the expected value for each worker...

- Provide a graph of load imbalance similar to Figure **??**.
- Does load imbalance follow a similar pattern to your scalability?

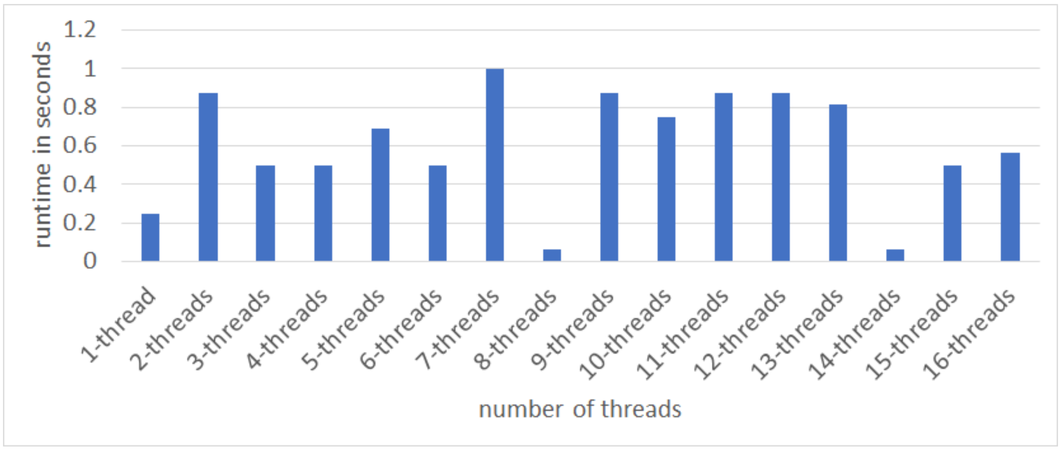

Figure 1: Step 1 scalability sample graph. *NOTE: the data in these graphs are random. If your data show different trends that's GOOD.*

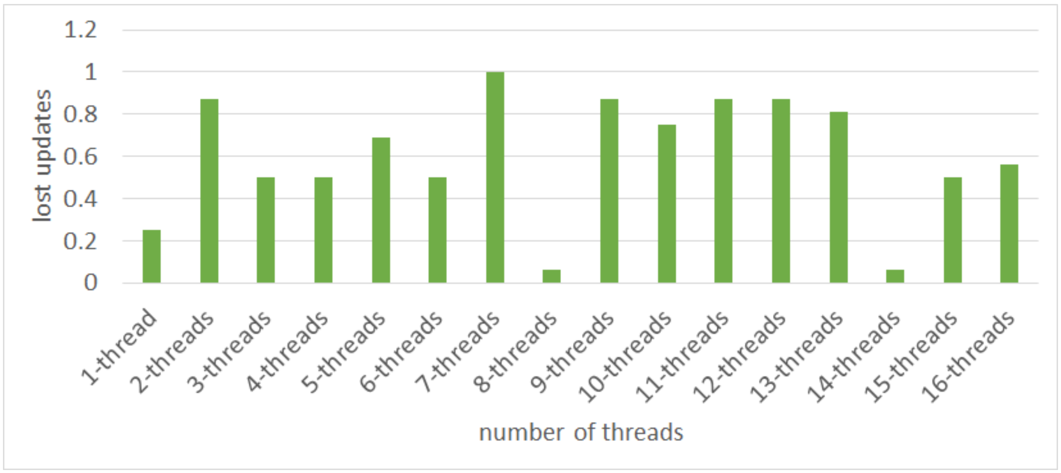

Figure 2: Step 1 lost-updates sample graph. *NOTE: the data in these graphs are random. If your data show different trends that's GOOD.*

2 Step 2: Synchronization

In this sample, the graphs for mutex, spinlocks, and atomics are combined. You can follow this pattern, or provide separate graphs for each.

- Provide a graph of scalability similar to Figure **??**.
- Provide a graph of load imbalance similar to Figure **??**.
- Do the run-time and load balance as a function of the number of threads change significantly for mutexes from the unsynchronized case? How about absolute performance? Does the lock make things slower or faster or is it a wash?
- Do the run-time and load balance as a function of the number of threads change significantly from the unsynchronized case? How about absolute performance? Does the spinlock make things slower or faster or is it "a wash"? How does it affect load imbalance, and why?

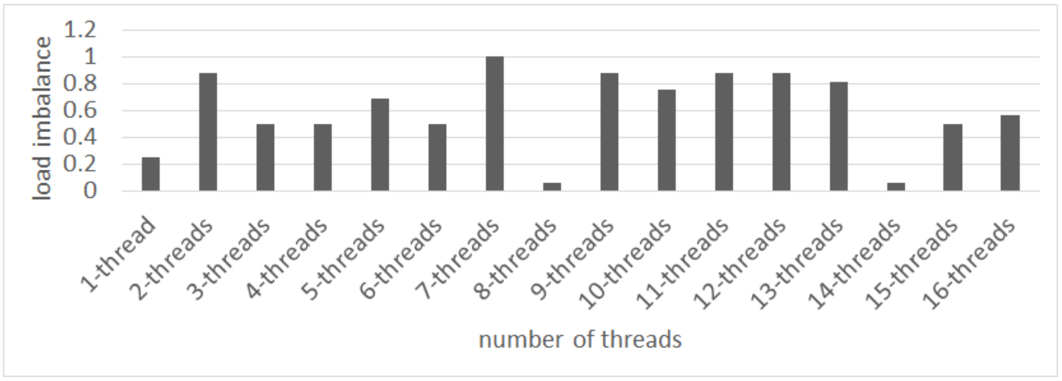

Figure 3: Step 1 load-imbalance sample graph. *NOTE: the data in these graphs are random. If your data show different trends that's GOOD.*

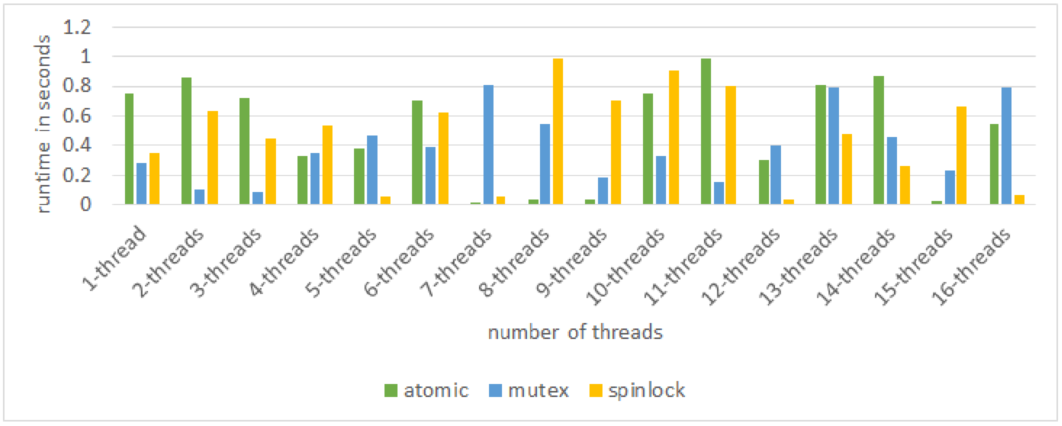

Figure 4: Step 2 scaling sample graph. NOTE: DATA IN SAMPLES IS RANDOM.

- Do scalability, absolute performance, or load imbalance change? Why/why not?

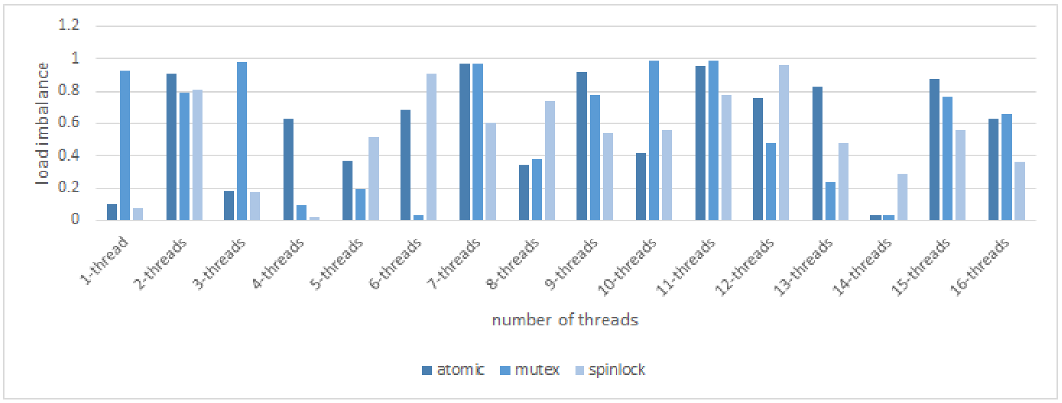

Figure 5: Step 2 imbalance sample graph. NOTE: RANDOM DATA!

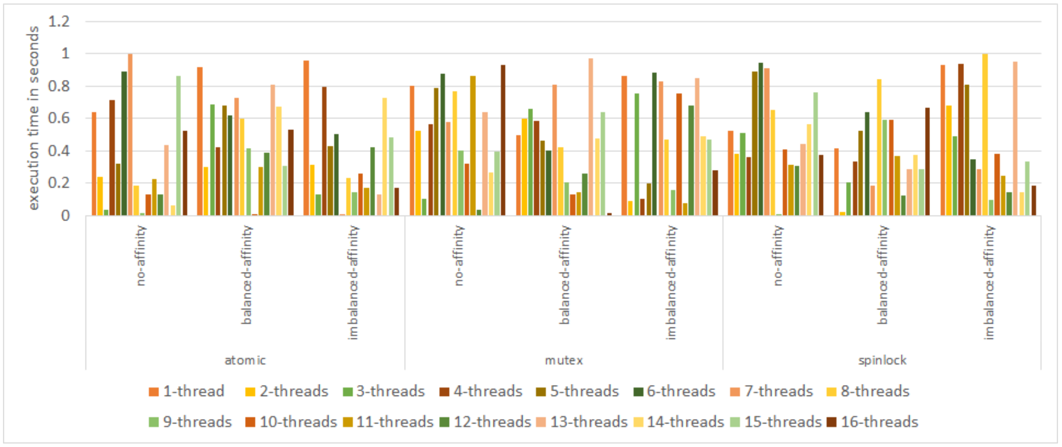

Figure 6: Step 3 scaling sample graph. NOTE: data in samples are random.

3 Step 3: Load balancing with core affinity

In this sample, the graphs for affinity, load-balanced and single-core pinned with mutex, spinlocks, and atomics are combined. You can follow this pattern, or provide separate graphs for each.

- Provide a graph of scalability similar to Figure **??**. *Recall that you are not required to do every locking primitive. It is fine to measure just one, although we hope you find it interesting to investigate them all.*
- Provide a graph of load imbalance similar to Figure **??**. *Recall that you are not required to do every locking primitive. It is fine to measure just one, although we hope you find it interesting to investigate them all.*
- What changes when you evenly distribute workers to cores and why?
- What changes when you put all workers on one core and why?

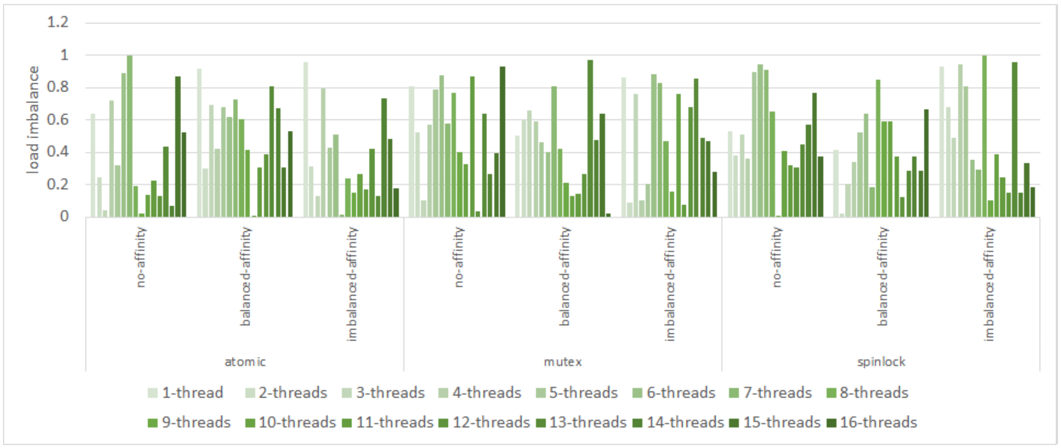

Figure 7: Step 3 load imbalance sample graph. NOTE: data in samples are random.

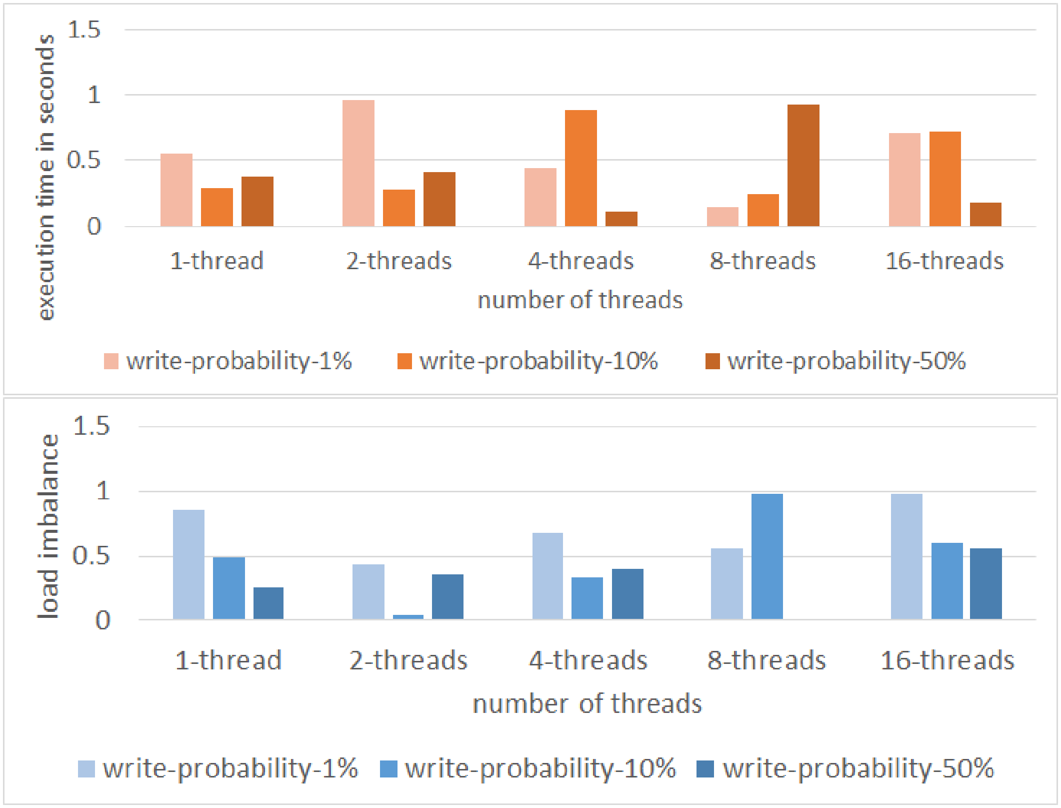

Figure 8: Step 4 scaling and load imbalance sample graphs for spinlocks. *NOTE: SAMPLES USE RANDOM DATA*

4 Step 4

- Provide graphs of scaling and load imbalance for spinlocks as a function of thread count and write probability similar to Figure **??**.
- Provide graphs of scaling and load imbalance for atomics as a function of thread count and write

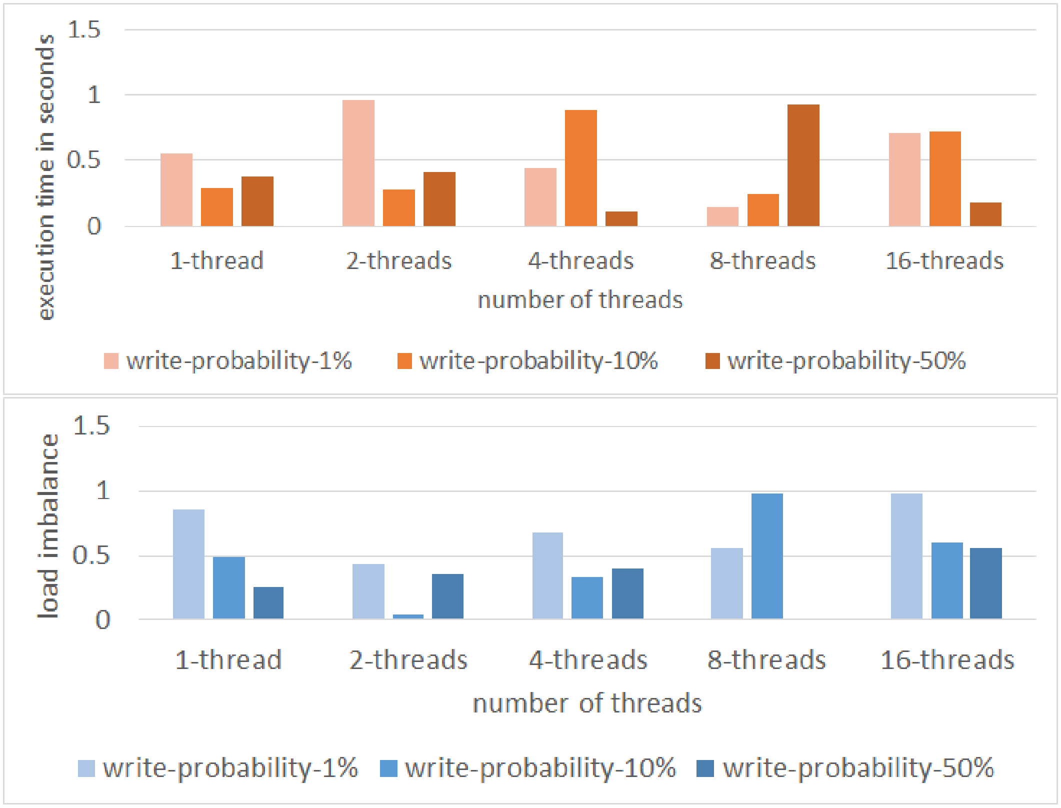

Figure 9: Step 4 scaling and load imbalance sample graphs for atomics. *NOTE: SAMPLES USE RANDOM DATA*

probability similar to Figure **??**.

- How do scalability and load imbalance change with read-write ratio using spinlocks? Do the data match your expectations? Please speculate on any reasons for divergence; you are not required to investigate them (yet), but you should feel free to do so if you're curious.
- How do scalability and load imbalance change with read-write ratio using atomics? Do the data match your expectations? Please speculate on any reasons for divergence; you are not required to investigate them (yet), but you should feel free to do so if you're curious.

5 Step 5

- Provide graphs of scaling and load imbalance for reader/writer locks as a function of thread count and write probability similar to Figure **??**.
- How do scalability and load imbalance change with read-write ratio using spinlocks? Do the data match your expectations? Please speculate on any reasons for divergence; you are not required to investigate them (yet), but you should feel free to do so if you're curious.

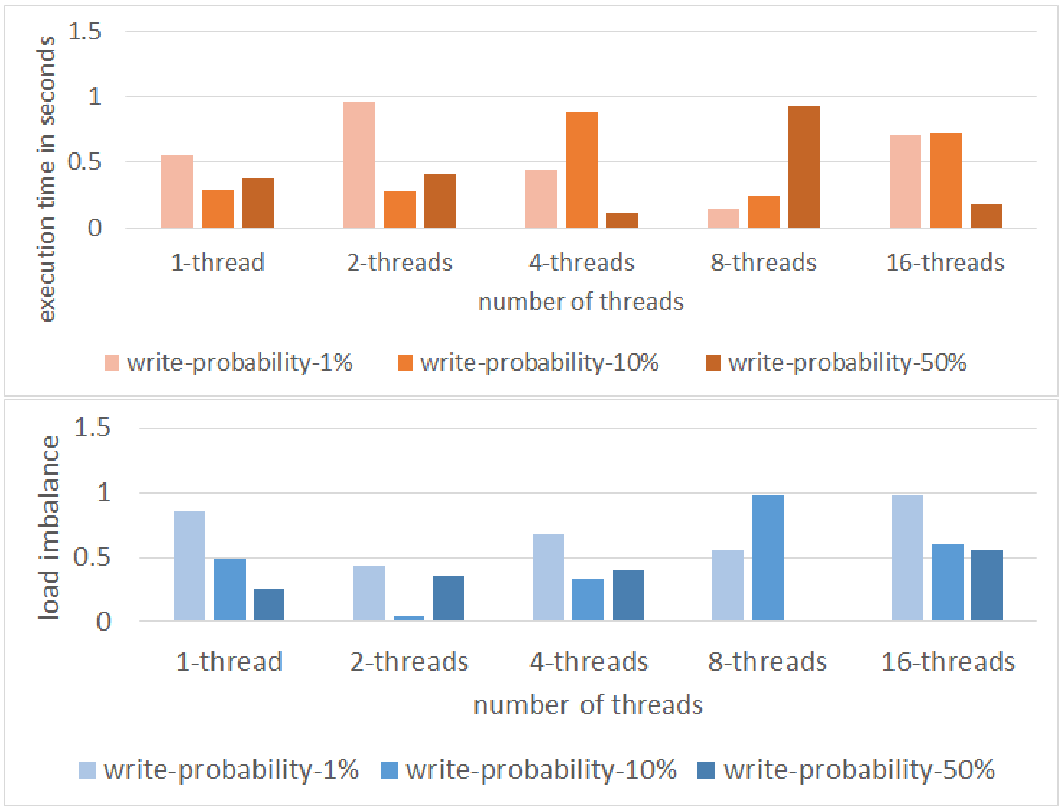

Figure 10: Step 5 scaling and load imbalance sample graphs for reader-writer locks. *NOTE: SAMPLES USE RANDOM DATA*