15-418 Final Project
Cachesim

Sam Flattery (sflatter)
Brian Wei (bwei1)

What We Did

- NUMA cache simulator using distributed directory based cache coherence
- Used Intel's pin to generate traces
- Compared block types
    - MSI
    - MESI
    - MOESI
- Compared cache perf of different locks
    - Test-and-set
    - Test-and-test-and-set
    - Ticketlock
    - Arraylock (aligned and unaligned)

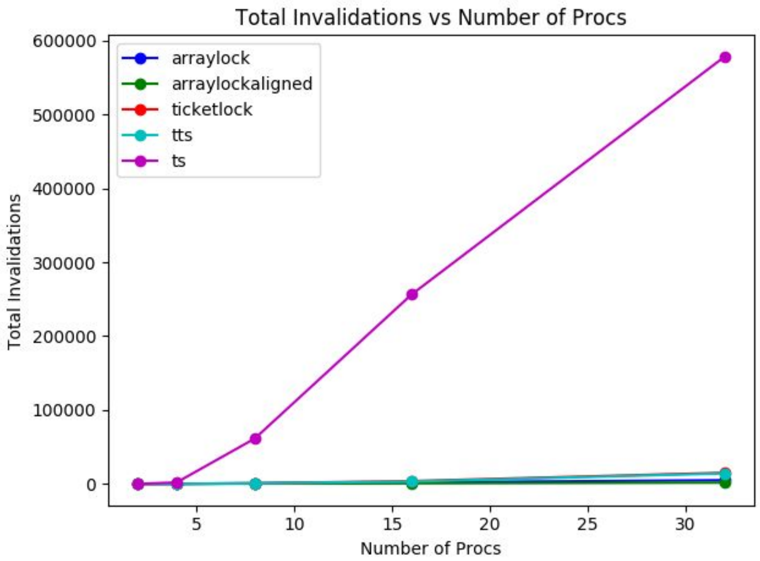
```
[CPU] [R/W] [Address] [NUMA Node]
0 R 0x7ffef17ecea8 0
1 W 0x7ffef17ecea0 1
1 W 0x7ffef17ece98 1
0 W 0x7ffef17ece90 0
2 W 0x7ffef17ece88 0
2 W 0x7ffef17ece80 0
3 R 0x7ffef17ece78 0
4 W 0x7fad12ad1e00 0
```

Stats / Latencies

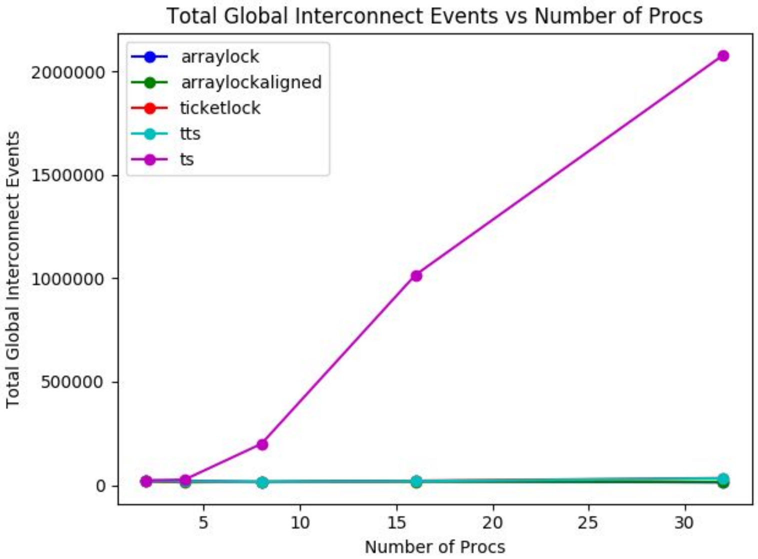
```
    ** Aggregate Stats Without Processor 0 ***


Total Reads/Writes:      8362032


Caches
------
Total Hits:              8359837
Total Misses:            2195
Total Flushes:           560
Total Evictions:         0
Total Dirty Evictions:   0
Total Invalidations:     740


Interconnects
-------------
Total Local Interconnect Events:    64993
Total Global Interconnect Events:   17732
```

```
Memory
------
Total Memory Reads:      14820
Total Memory Writes:     560
Total Memory Accesses:   15380


Latencies
---------
Cache Access Latency:               8ms
Memory Read Latency:                1ms
Memory Write Latency:               56us
Memory Access Latency:              1ms
Local Interconnect Latency:         64us
Global Interconnect Latency:        35us
```

Test-And-Set



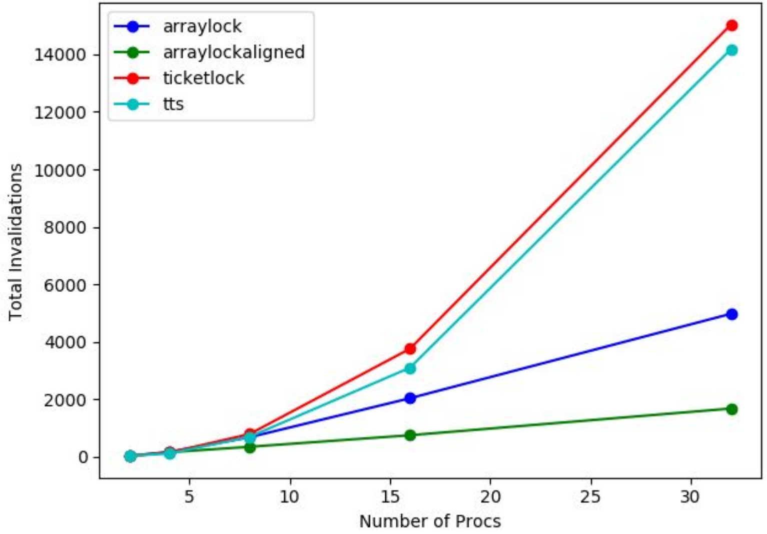
Total Invalidations vs Number of Procs

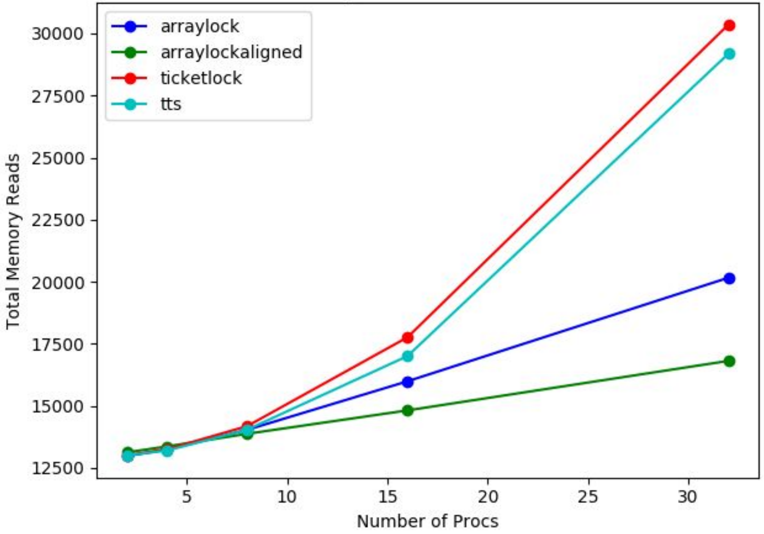


Total Global Interconnect Events vs Number of Procs

- Much worse performance than other locks due to one invalidation per lock acquisition attempt

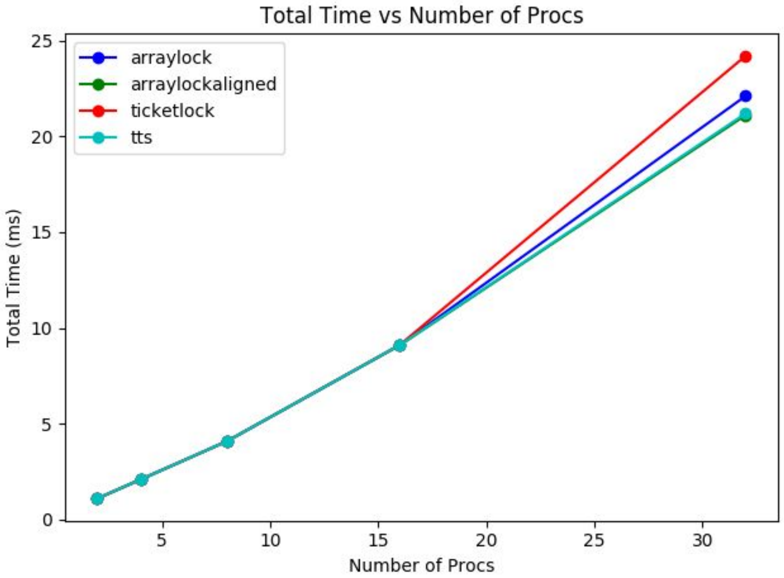
Invalidations

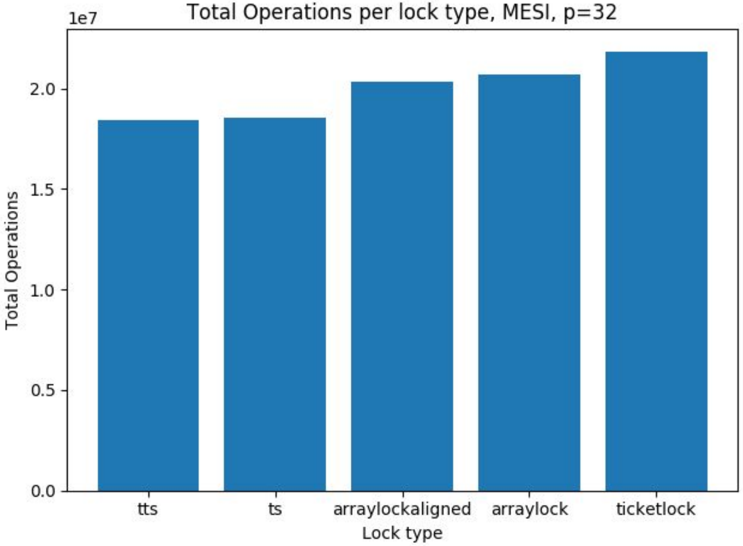

Total Invalidations vs Number of Procs


Total Memory Reads vs Number of Procs

- The aligned arraylock had by far the best cache performance due to having O(1) invalidations per lock release, which meant far fewer memory reads
- The ticketlock and test-and-test-and-set had O(P) invalidations per lock release

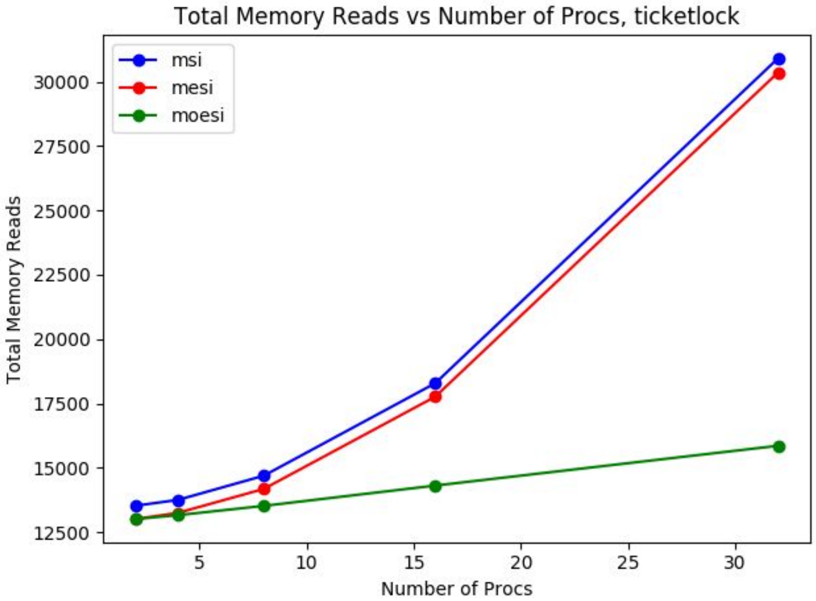
Total Time Estimates

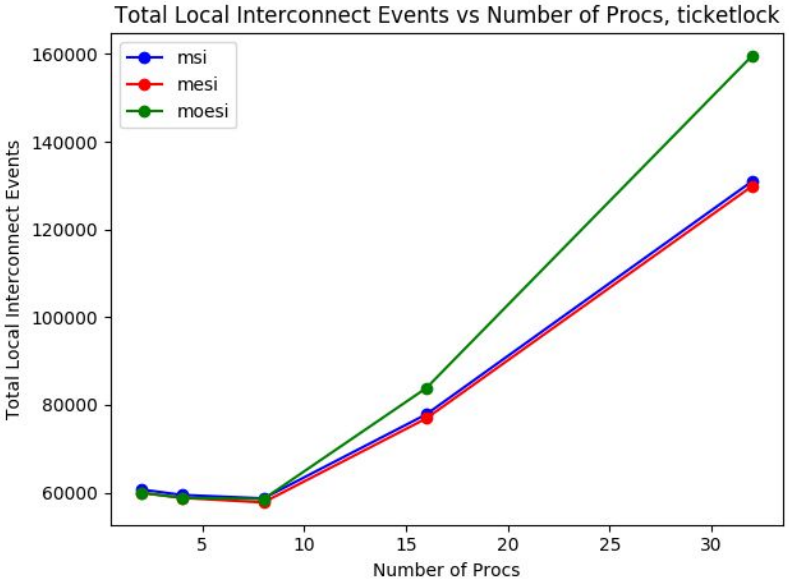




- Even though arraylock had far better cache performance, it has higher acquisition/release cost
- TTS lock performed well because it was cheap, even with relatively poor cache performance

MOESI vs MESI vs MSI





- MOESI gives up to 50% reduction in memory reads at expense of more interconnect traffic
- MESI gives small reduction in memory reads due to one fewer BusRd necessary