Reducing Contention Through Priority Updates

Julian Shun (CMU) Guy Blelloch (CMU) Jeremy Fineman (Georgetown) Phillip Gibbons (Intel, Pittsburgh)

Motivation

- Memory contention can be a serious performance bottleneck in concurrent programs on shared memory machines
- But shared memory accesses can be very useful in parallel algorithms, concurrent data structures, thread communication protocols, all of which are important in cloud-based applications

Sharing vs. Contention

- **Sharing:** Concurrent access to shared memory resource by multiple cores
- **Contention:** Performance penalty associated with sharing

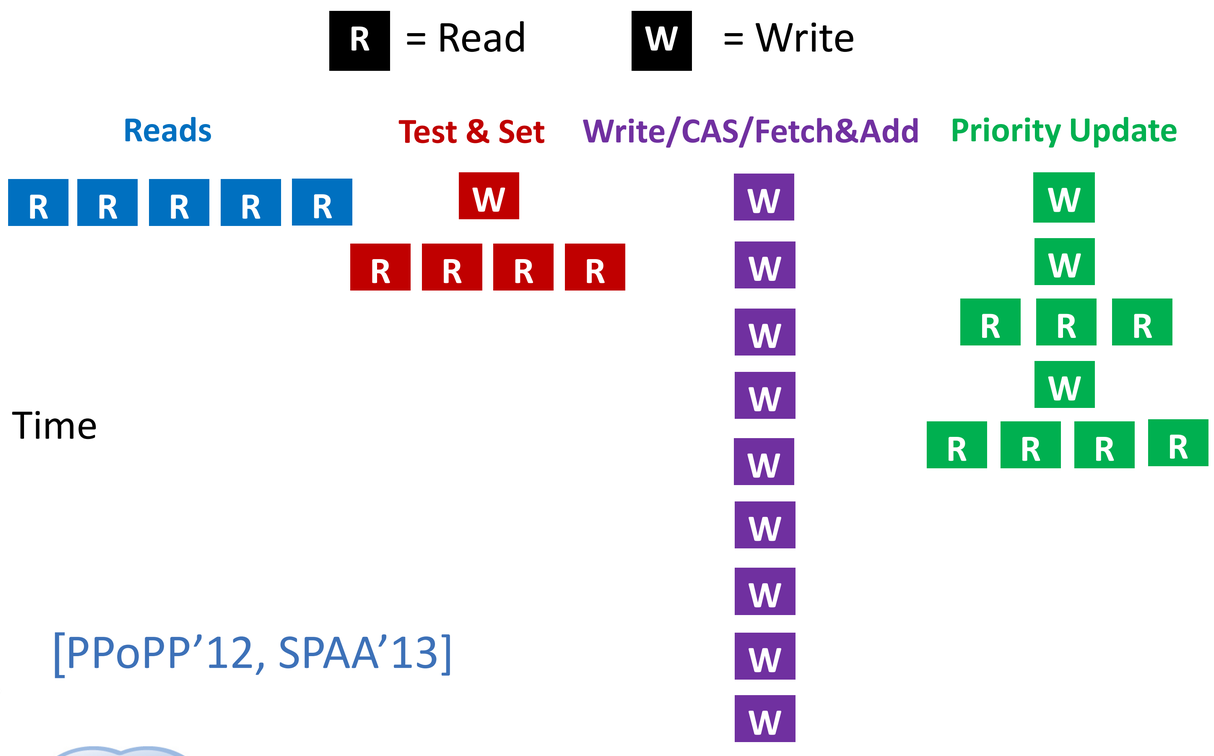
Priority Update

- Takes as arguments a memory location *addr*, a new value *newval* and a comparison function $>_p$
- Atomically compares the value at *addr* with *newval* and stores *newval* at *addr* if and only if it has a higher priority according to $>_p$
- Implemented with compare-and-swap (CAS)
- Pseudocode:

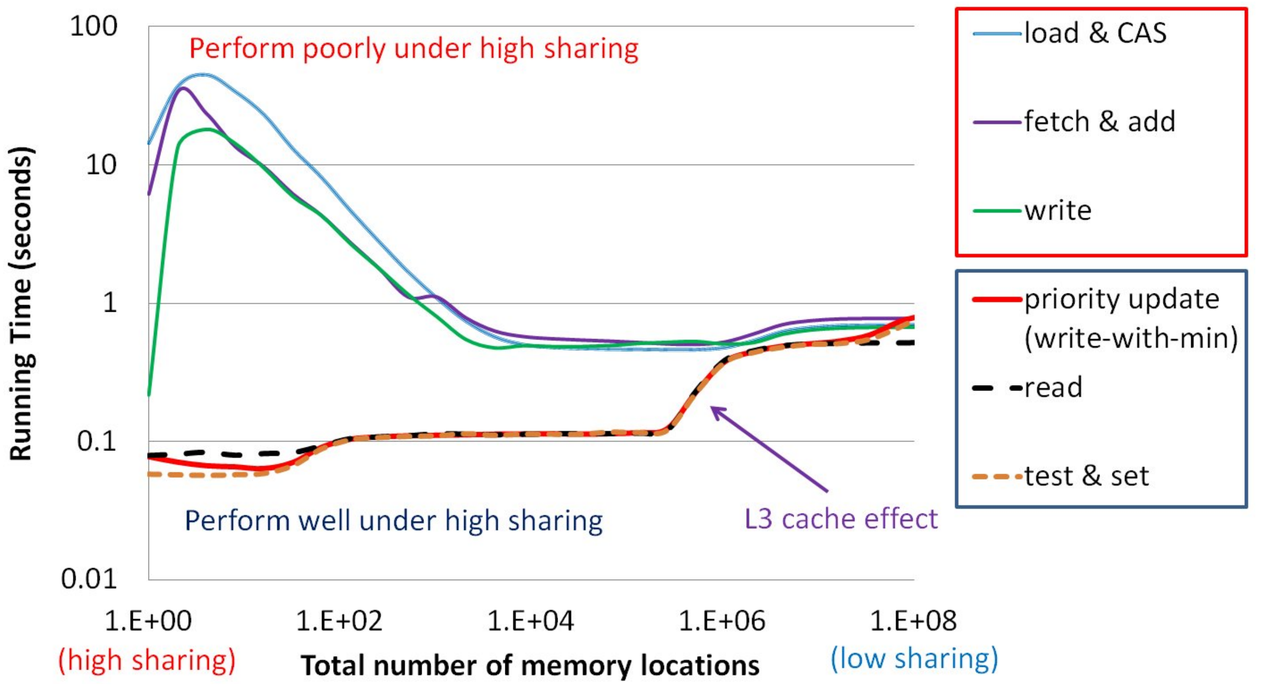
```
procedure PriorityUpdate(addr, newval, >p)
  oldval = *addr
  while(newval >p oldval) do
    if compare-and-swap(addr, oldval, newval) then
      return
    else
      oldval = *addr
```

- Test & Set: special case of Priority Update over values {0,1}

Concurrent Operations in Action



Low Contention Even Under High Sharing

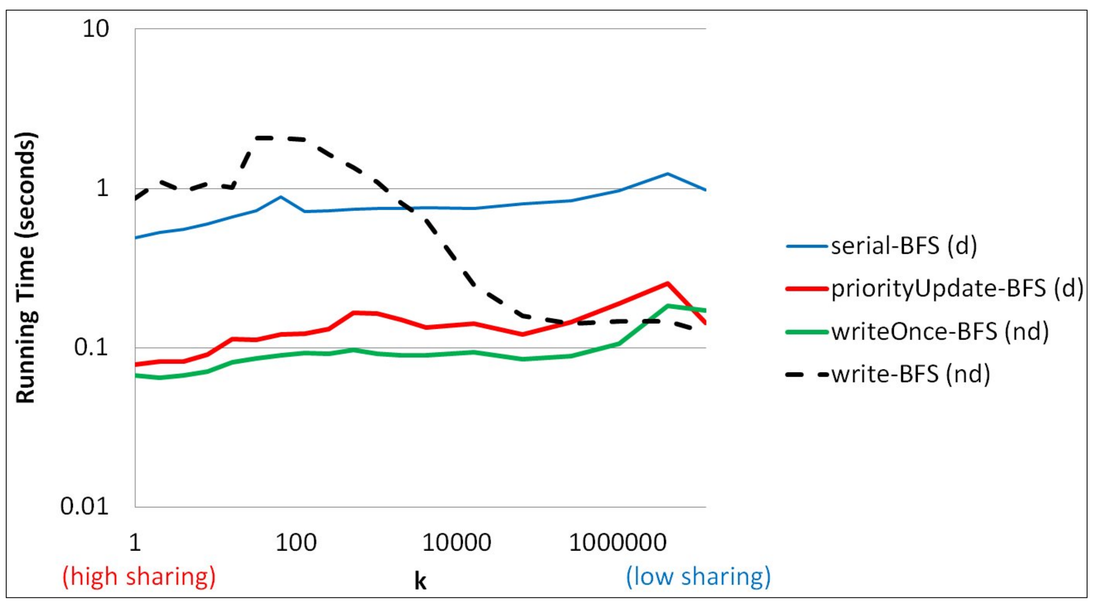


- Perform 5 runs of 10^8 concurrent operations on varying number of locations (i.e., varying degrees of sharing)
- Under high sharing, write-with-min (priority update) outperforms plain write, fetch & add/xadd and CAS by orders of magnitude!
- Write-with-min performs just as well as test and test-and-set, while having much broader applicability

Perform well under high sharing	Perform poorly under high sharing
Write-with-min (Priority Update)	Compare and swap (CAS)
Test-and-set	Fetch & add / xadd
Read	Plain write

Applications of Priority Update

- Deterministic algorithms: using priorities to resolve conflicts (breadth-first search, maximal matching, Delaunay triangulation)
- Correctness of algorithms: using write-with-min in shortest paths, connected components, minimum spanning forest
- Progress/termination: using priority updates to guarantee a "winner" in each iteration
- Case study: Breadth-first search using priority updates, using plain writes (write-BFS), write-once, sequential (serial-BFS)



Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley · University of Washington