RCU vs. Locking Performance on Different Types of CPUs

Paul E. McKenney
IBM Linux Technology Center
Paul.McKenney@us.ibm.com

Who is Paul McKenney, Anyway?

- Oregon State University
  - BSCS & BSME 1981, MSCS 1988
- Self-employed contract programer 1981-1985
- Unix® sysadm, packet-radio/Internet research, SRI International, 1986-1990
- TCP performance, SMP/NUMA hacking, co-inventor of RCU, Sequent 1990-1999
- AIX®, Linux, IBM 1999-present
- Oregon Graduate Institute Ph.D. in progress

Overview

- *Why isn't Moore's Law helping my code???*
- Hash-table mini-benchmark
- How can we fix this?
  - Linked-list insertion and removal
- Performance results
  - On x86, IPF/x86, Opteron, and PPC
- Summary and Conclusions

Why Isn't Moore's Law Helping My Code?

- ? Moore's Law provided uneven benefits:
  - – Instruction execution overhead much improved
  - – Pipeline-flush overhead has not improved much
  - – Memory latencies have not improved much
  - – Contention overhead not helped
- ? Moore's Law speeds up instructions
- ? But SMP SW does pipeline flushes, memory accesses, and suffers contention

Operation Costs: How Bad???
4-CPU 700MHz i386 P-III

Operation	Nanoseconds
Instruction	0.7
Clock Cycle	1.4
L2 Cache Hit	12.9
Atomic Increment	58.2
Cmpxchg Atomic Increment	107.3
Atomic Incr. Cache Transfer	113.2
Main Memory	162.4
CPU-Local Lock	163.7
Cmpxchg Blind Cache Transfer	170.4
Cmpxchg Cache Transfer and Invalidate	360.9

But Wait!!!

How bad is this, really???

Don't speculate, run a benchmark!

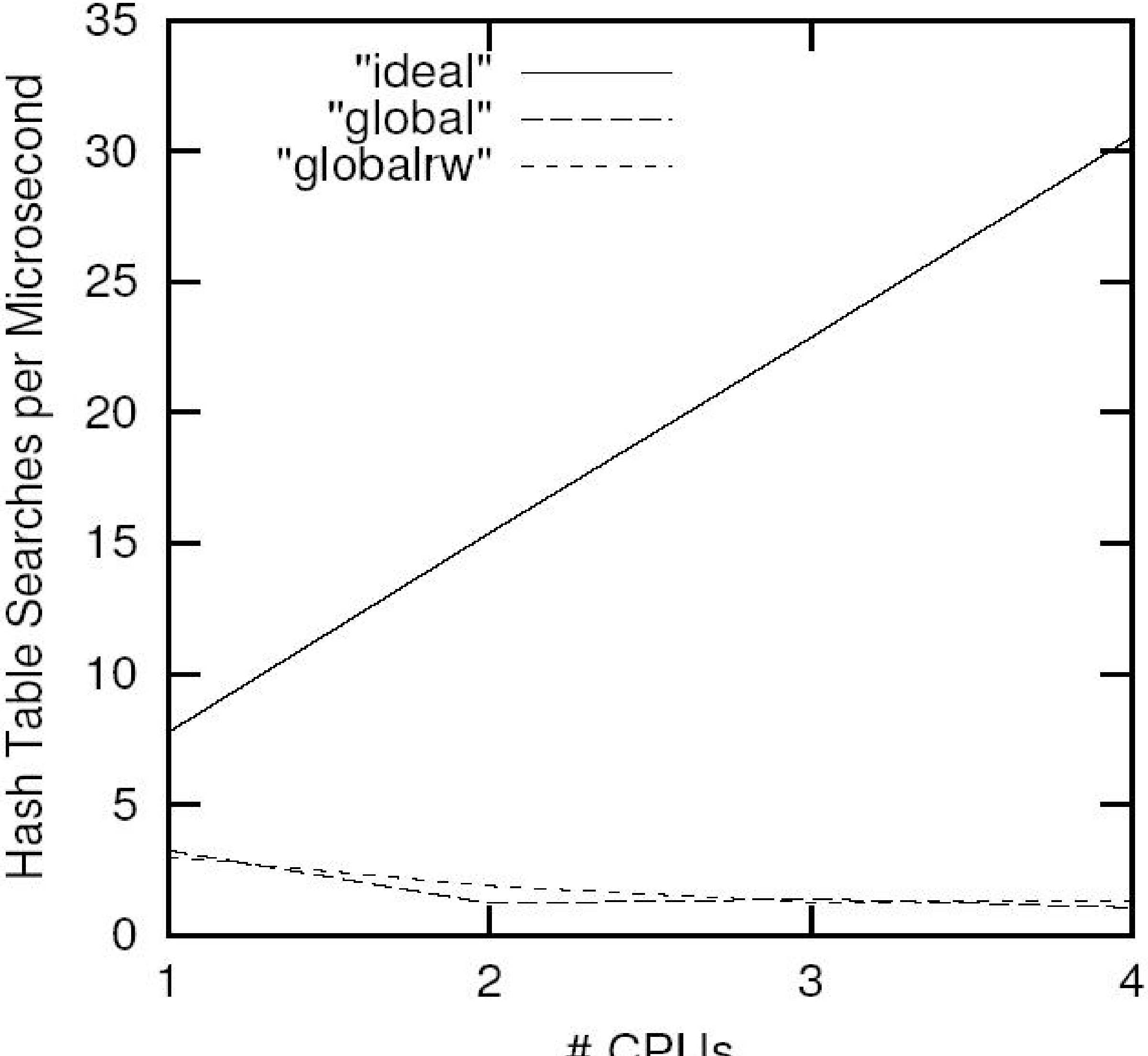
Hash-Table Mini-Benchmark

- Dense array of buckets
- Doubly-linked hash chains
- One element per hash chain
  - You do tune your hash tables, don't you???
- Mix of operations:
  - Search
  - Delete followed by reinsertion: maintain loading
  - Random run lengths selected for specified mix

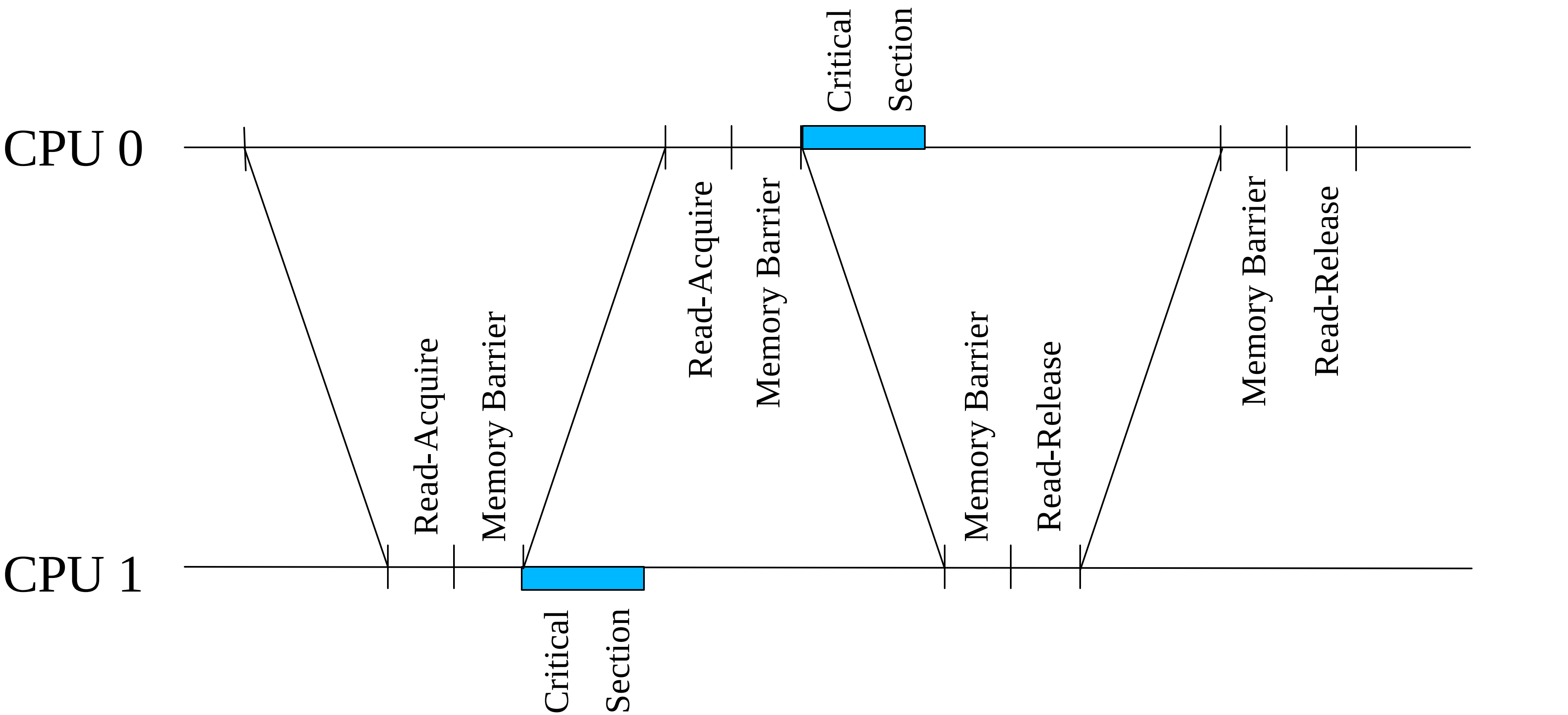
Hash-Table Mini-Benchmark

- Locking Designs Tested:
  - Global spinlock & rwlock
  - Per-bucket spinlock & rwlock
  - brlock
  - RCU
  - “Ideal”: take single-CPU results without locking, and multiply by the number of CPUs
    - Can be achieved in some cases using per-CPU data
    - No whining, no excuses!!!

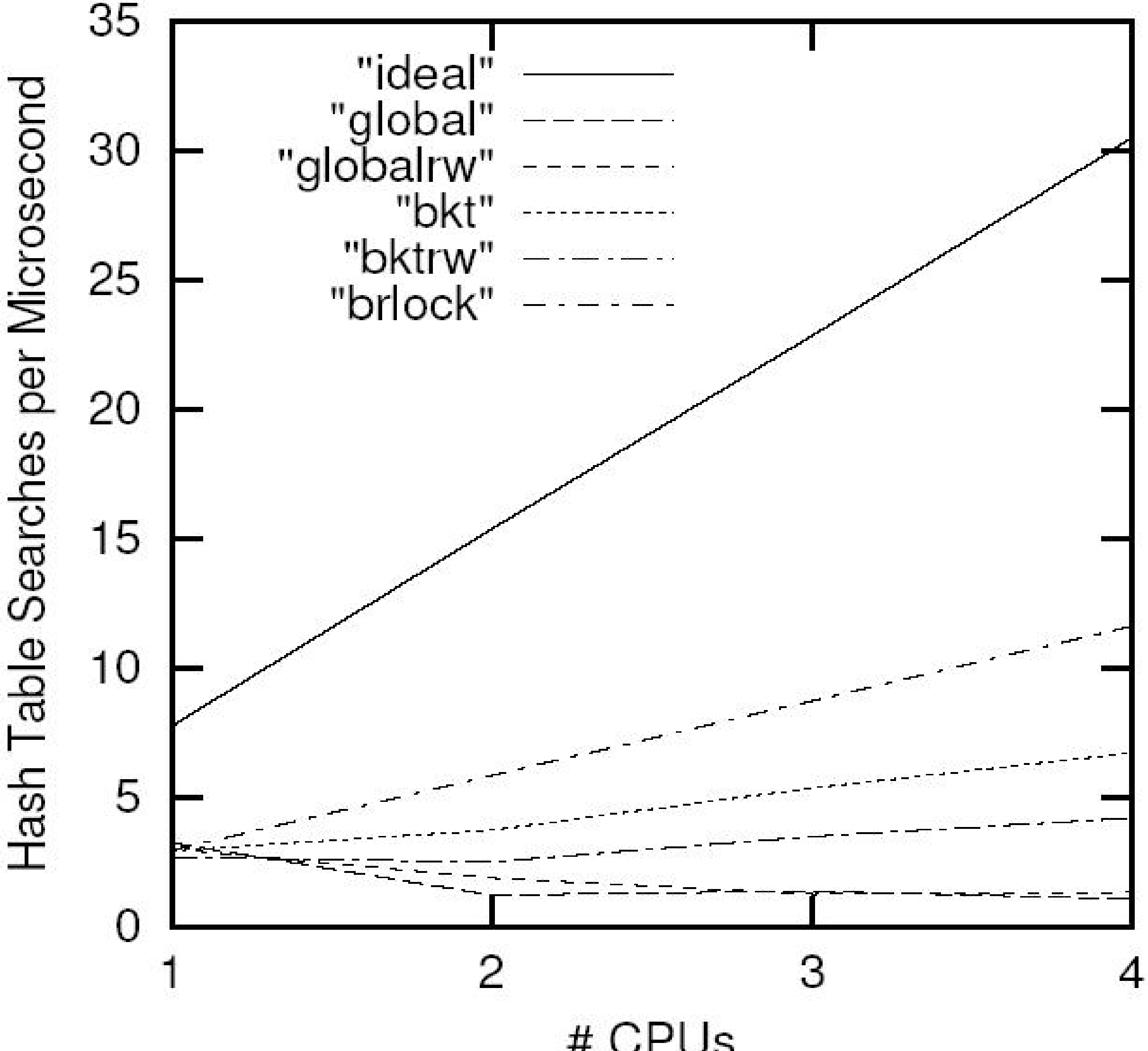
Global Locking



What is With rwlock???



“Scalable” Locking



How Can We Fix This???

- What do we want?
  - Good locking for read-mostly data structures!!!
  - Want to avoid expensive operations for readers
    - No memory latency (cache thrashing)
    - No pipeline flushing (memory barriers)
    - No contention
  - Can accept some additional overhead for writers
    - But must stay within the realm of reason

We Can Do Linked-List Insertion...

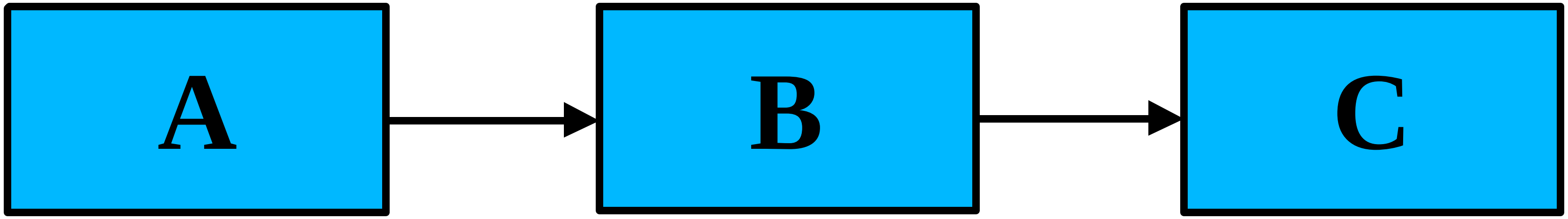
- ? Initialize then insert
  - – Readers will either see it or not
  - – But list will always be properly formatted
- ? Need memory barriers on weakly ordered machines (pretty much all of them)
- ? Taken care of for you by _rcu() list macros:
  - – Use list_add_rcu() to insert into the list
  - – Use list_for_each_entry_rcu() to scan the list

But Sooner Or Later...

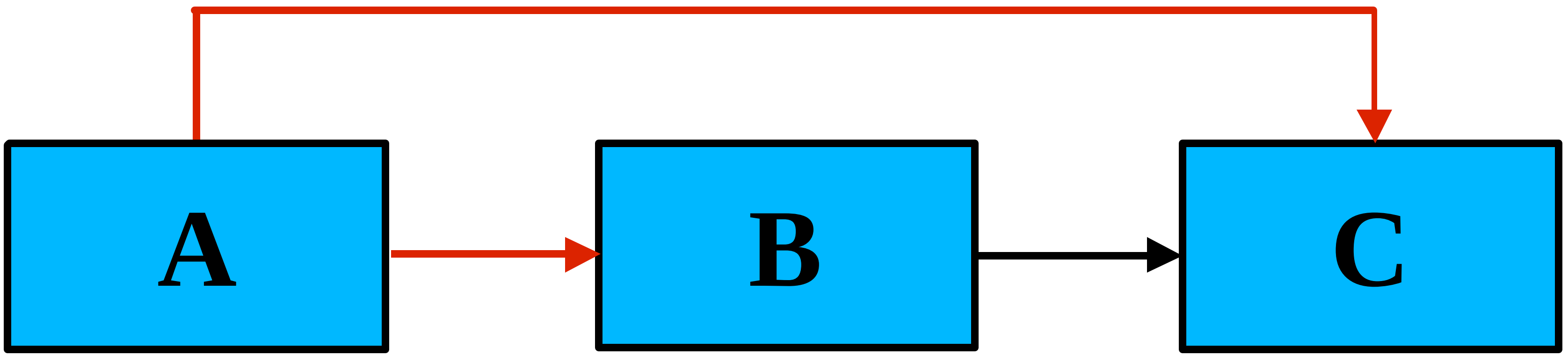
Something will need to be removed from the list

Just hop the pointer over the element to be deleted!!!

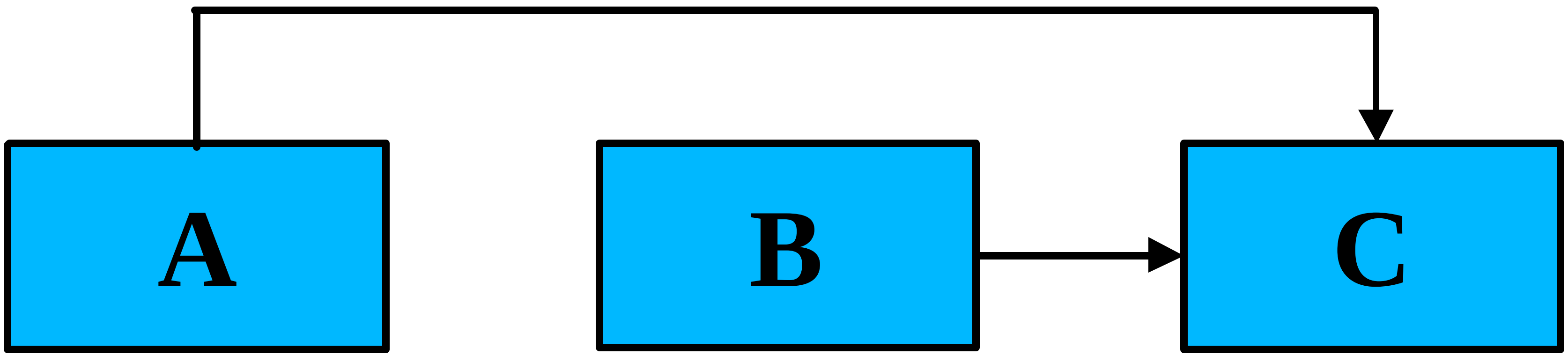
Lock-Free Removal Animation (1)



Lock-Free Removal Animation (2)



Lock-Free Removal Animation (3)



But Sooner Or Later...

It will be necessary to free up elements removed from the list...

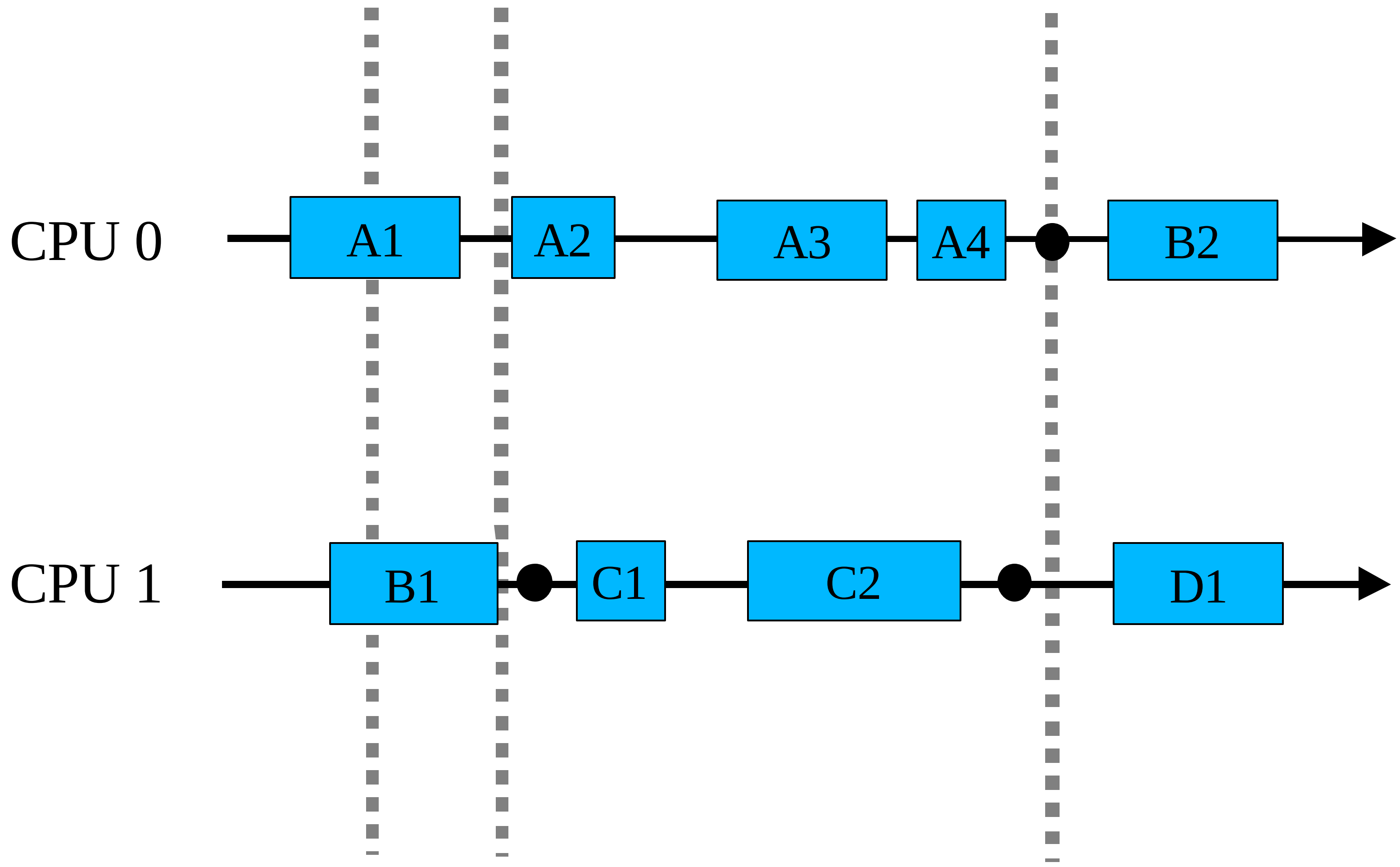
Unless it is OK to wantonly leak memory!!!

But readers might be referencing the removed element for quite some time...

When Are Readers Done?

- Read-side rwlock critical section:
  - Preemption disabled
  - No blocking
  - No return to user-mode execution
  - No page faults or exceptions
  - No holding references from one CS to another!
- If a CPU does a context switch, it is done!
  - All prior read-side critical sections complete
  - With *no* locking operations!!!

Grace Periods



Implemented in 2.6 kernel

It is called “RCU”

(Short for “Read-Copy Update”)

RCU Performance Testing

- Four-CPU 700MHz P-III System
- Four-CPU 1.4GHz IPF System (running x86 code)
- Four-CPU 1.4GHz Opteron System
- Eight-CPU 1.45GHz Power4+ System
  - Only four CPUs were used in these benchmarks

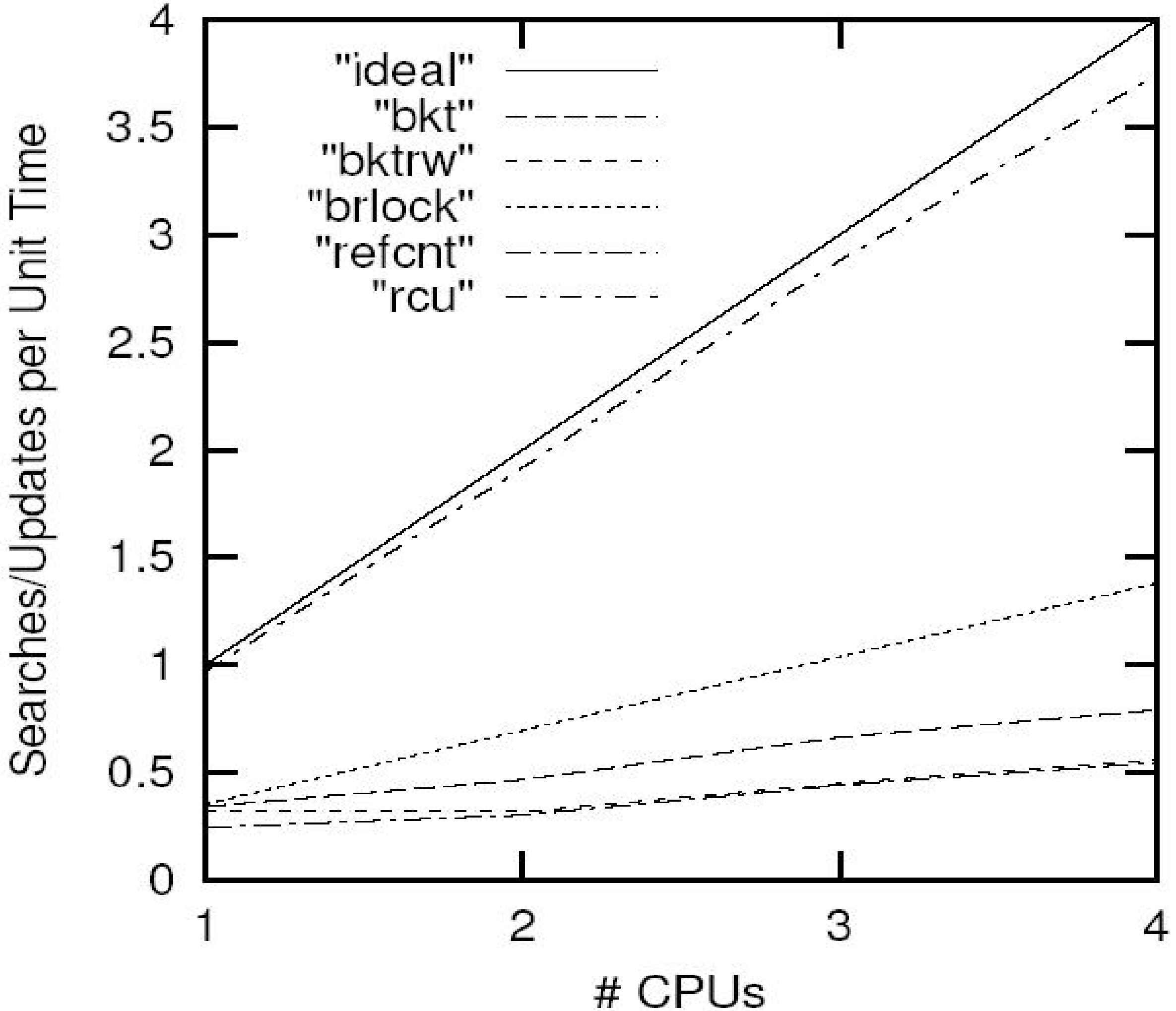
Test Scenarios

- Read-only test
  - For data structures that are almost never modified
    - Routing tables, HW/SW configuration, policies
- Mixed workload
  - Vary fraction of accesses that are updates
  - See how things change as read-intensity varies
  - Expect breakeven point for RCU and locking

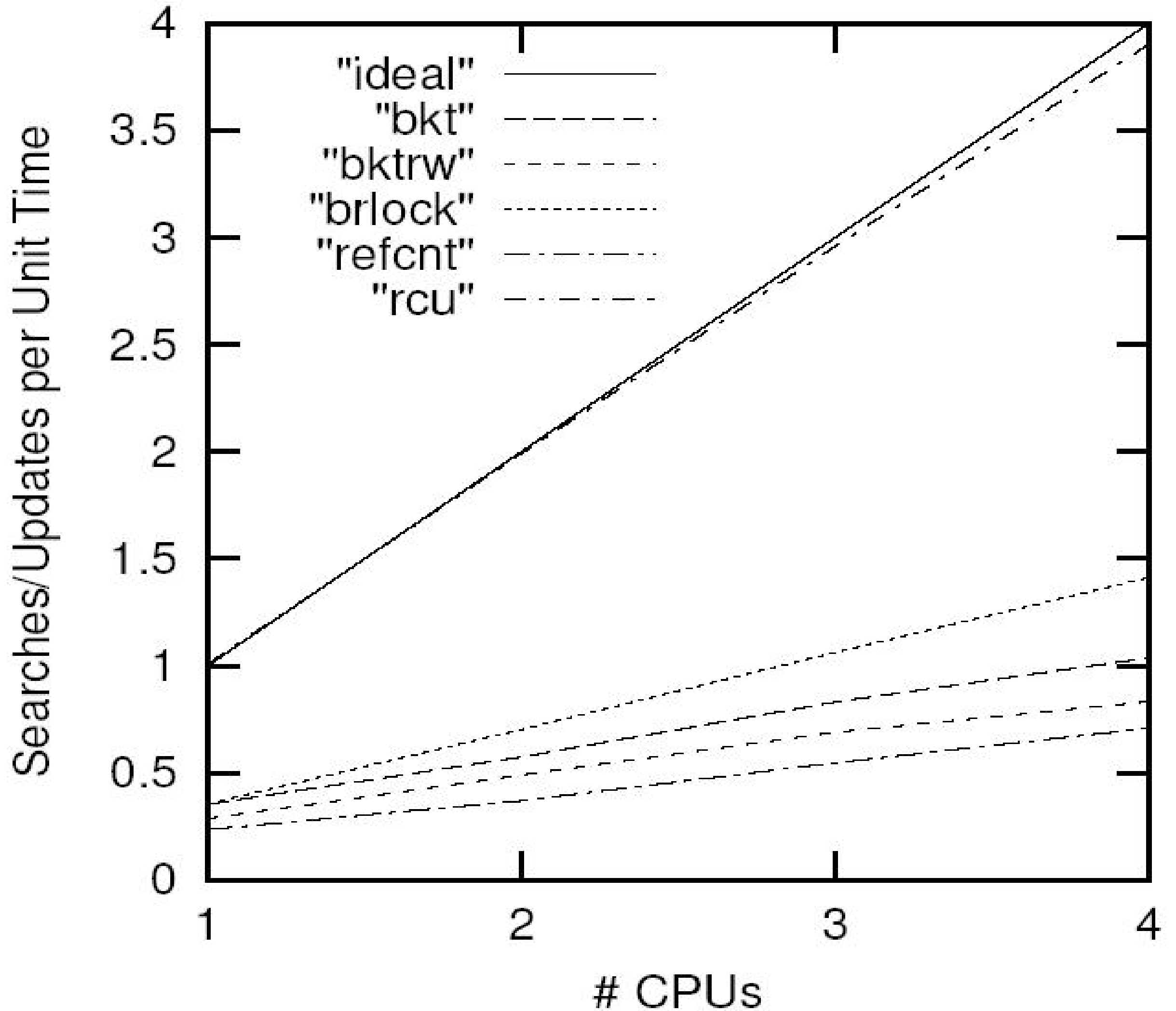
Overview of Results: Read-only

- Global spinlock/rwlock scale negatively
- Per-bucket schemes scale, but poorly
  - 10-20% of ideal at 4 CPUs
  - Less than half of ideal on single CPU
    - But why would you run CONFIG_SMP on one CPU?
- brlock scales better
  - But still less than 40% of ideal
  - And brlock is known to have trouble on writes...

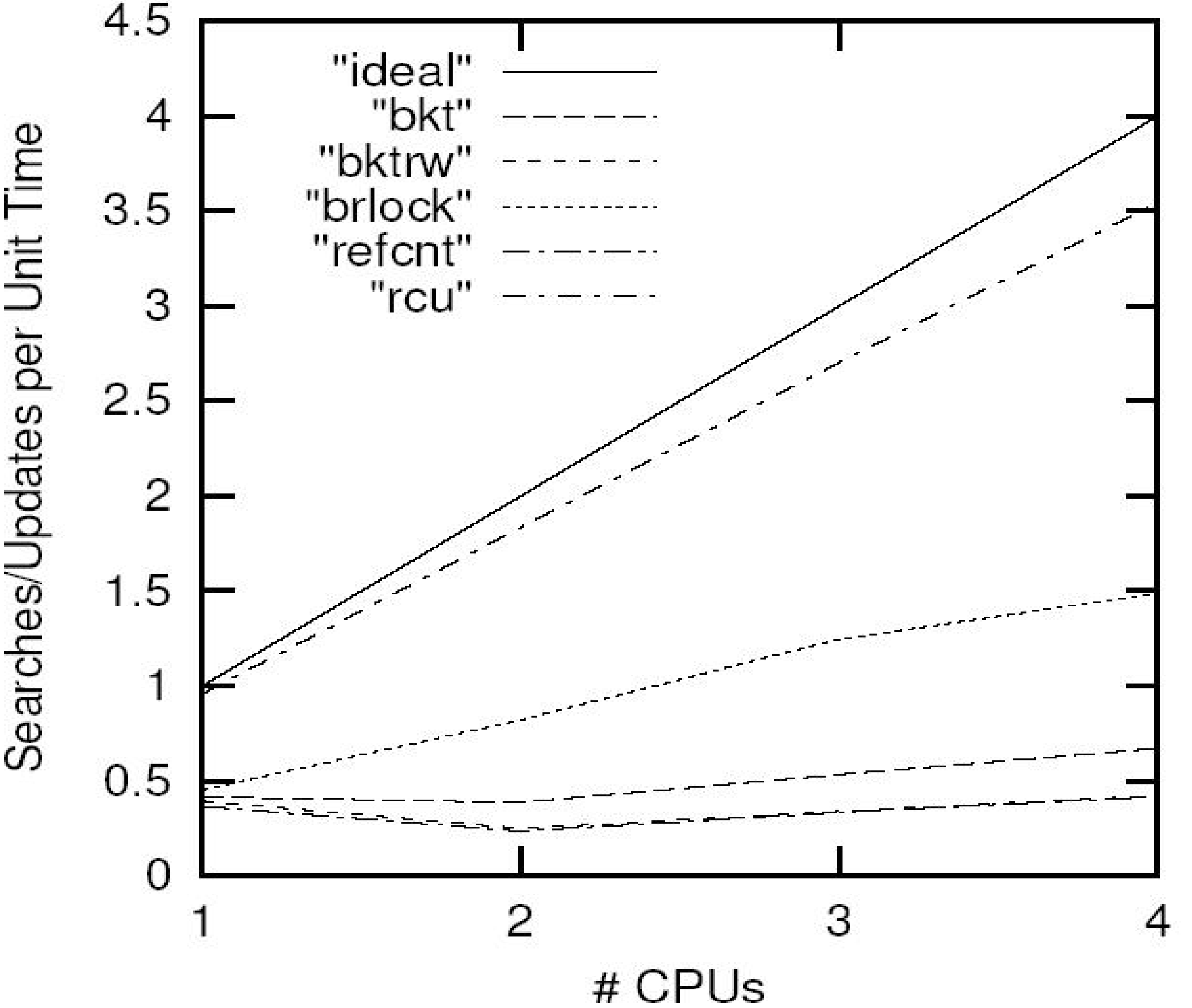
x86 Read-Only Results



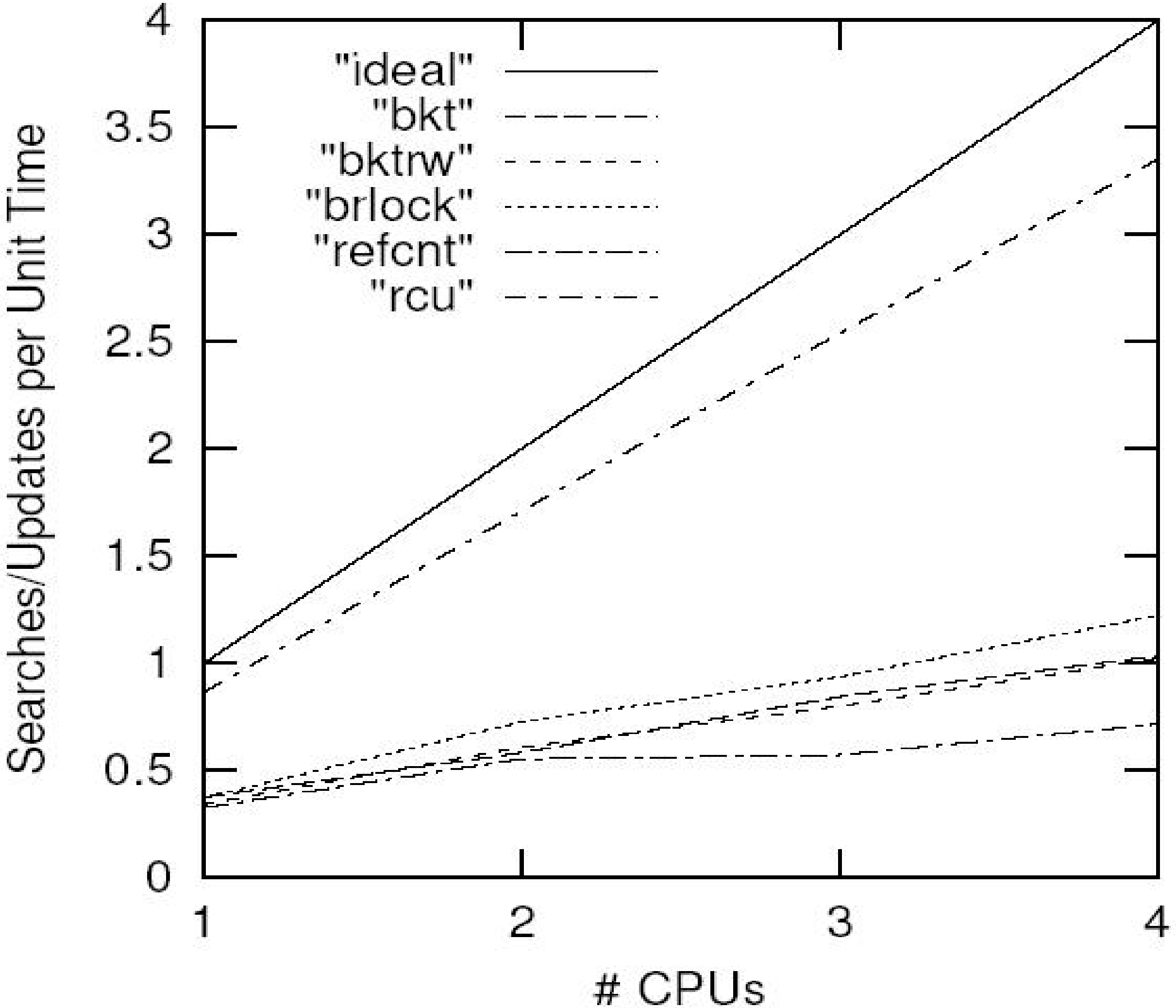
IPF Read-Only Results



Opteron Read-Only Results



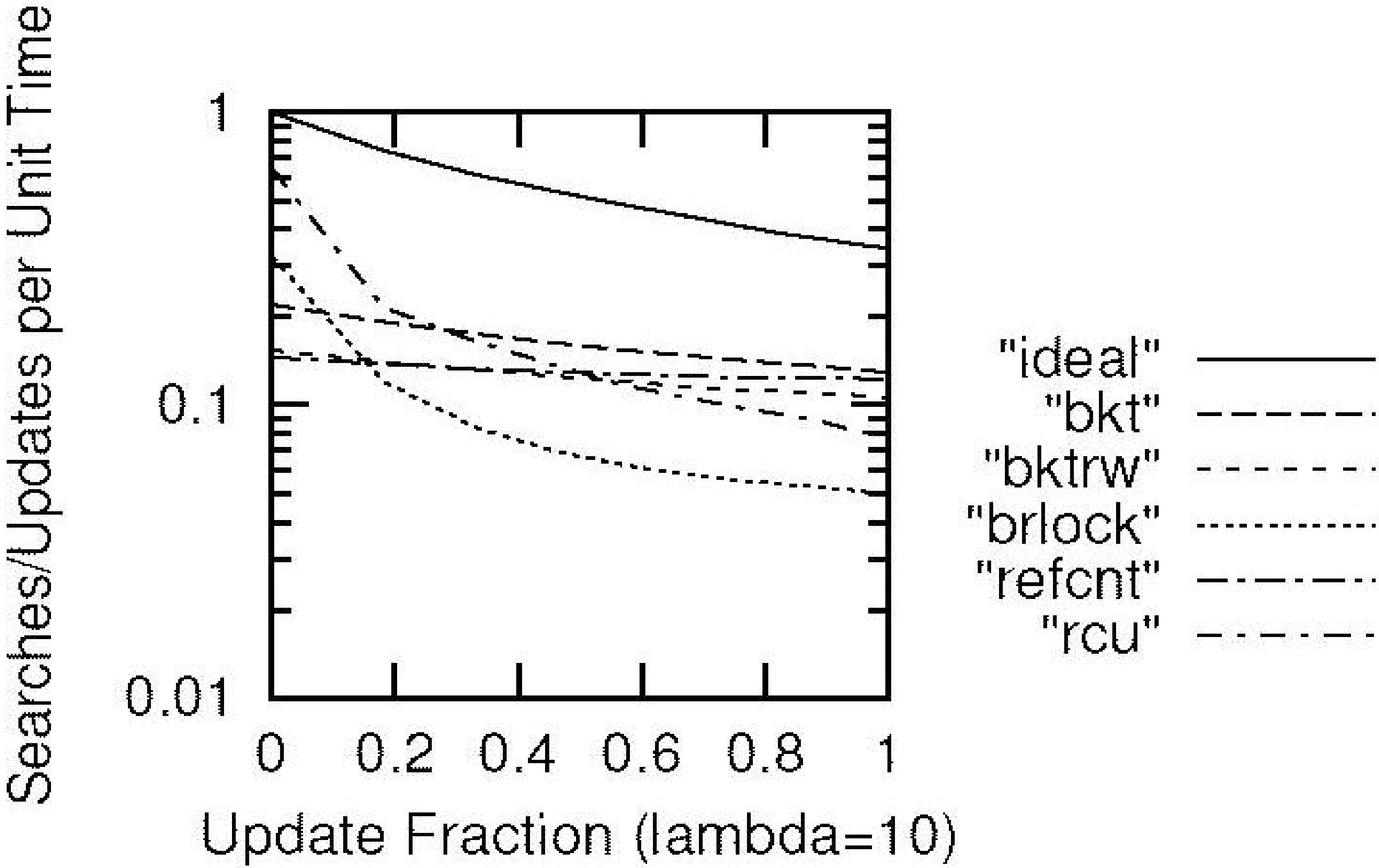
PPC Read-Only Results



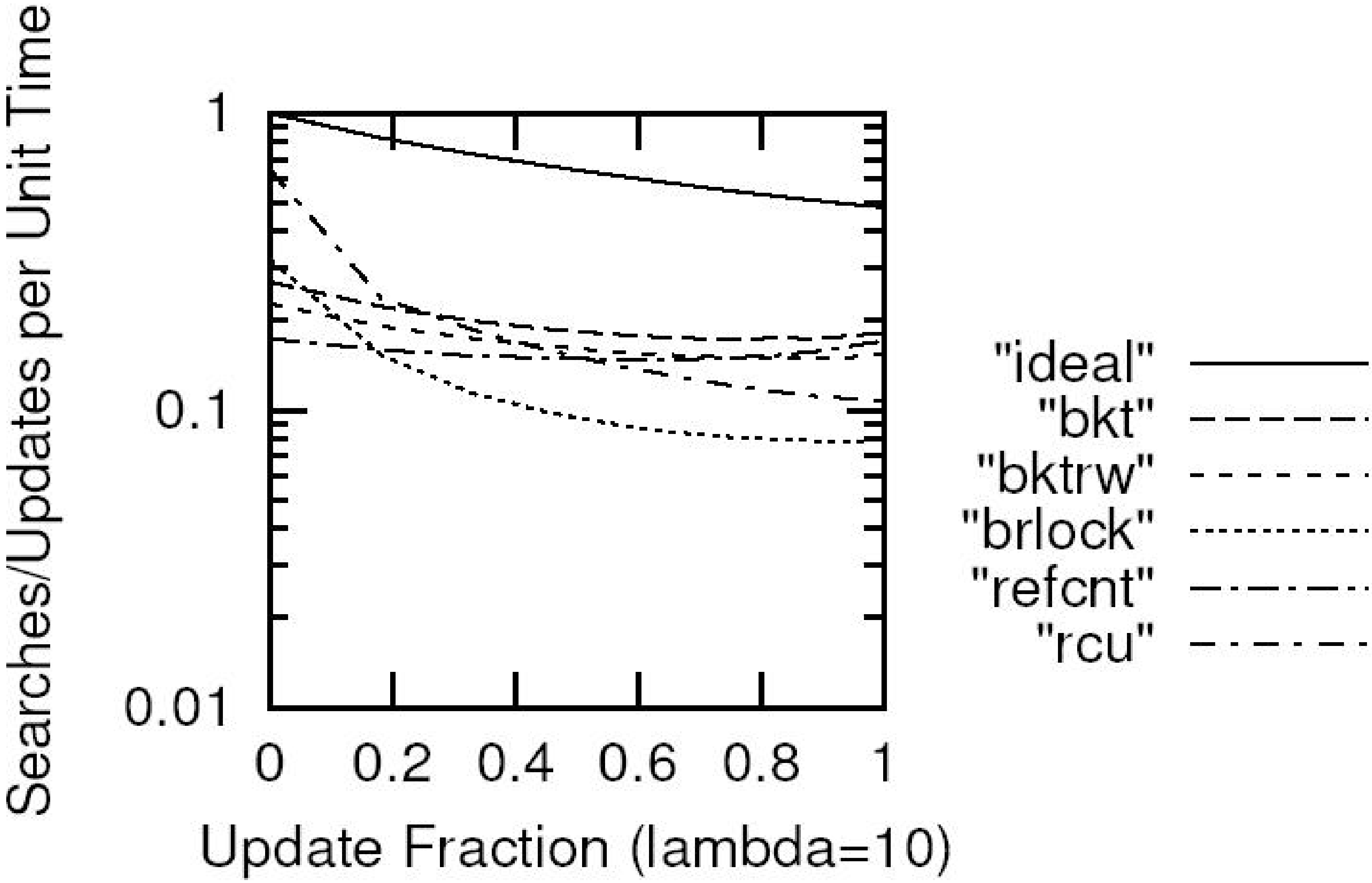
Overview of Results: Mixed Workload

CPU Type	***Crossover***
X86	***0.2-0.5***
IPF/x86	***0.1-0.4***
Opteron	***0.2-0.5***
PPC	***0.3-0.5***

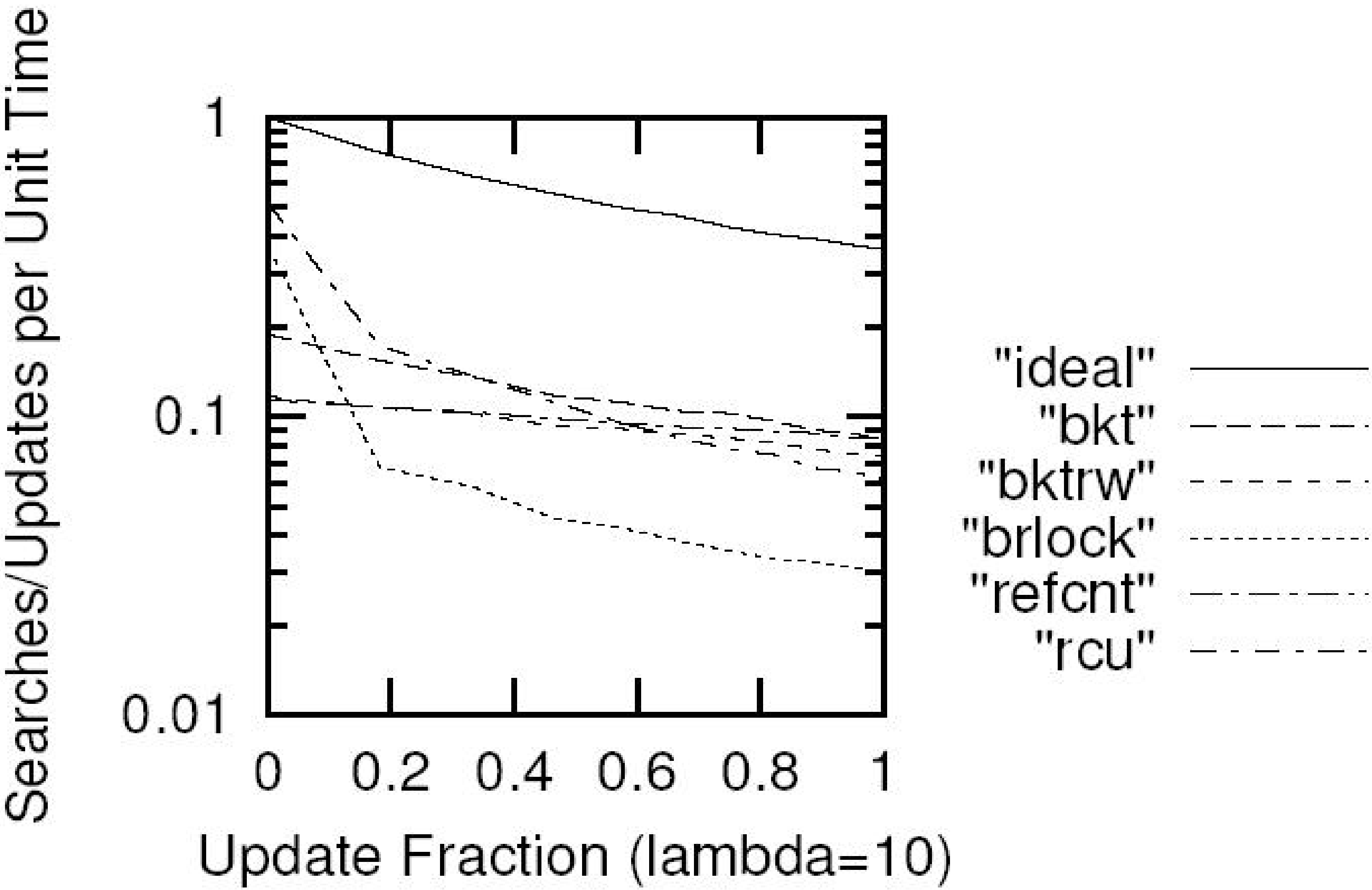
x86 Results for Mixed Workload



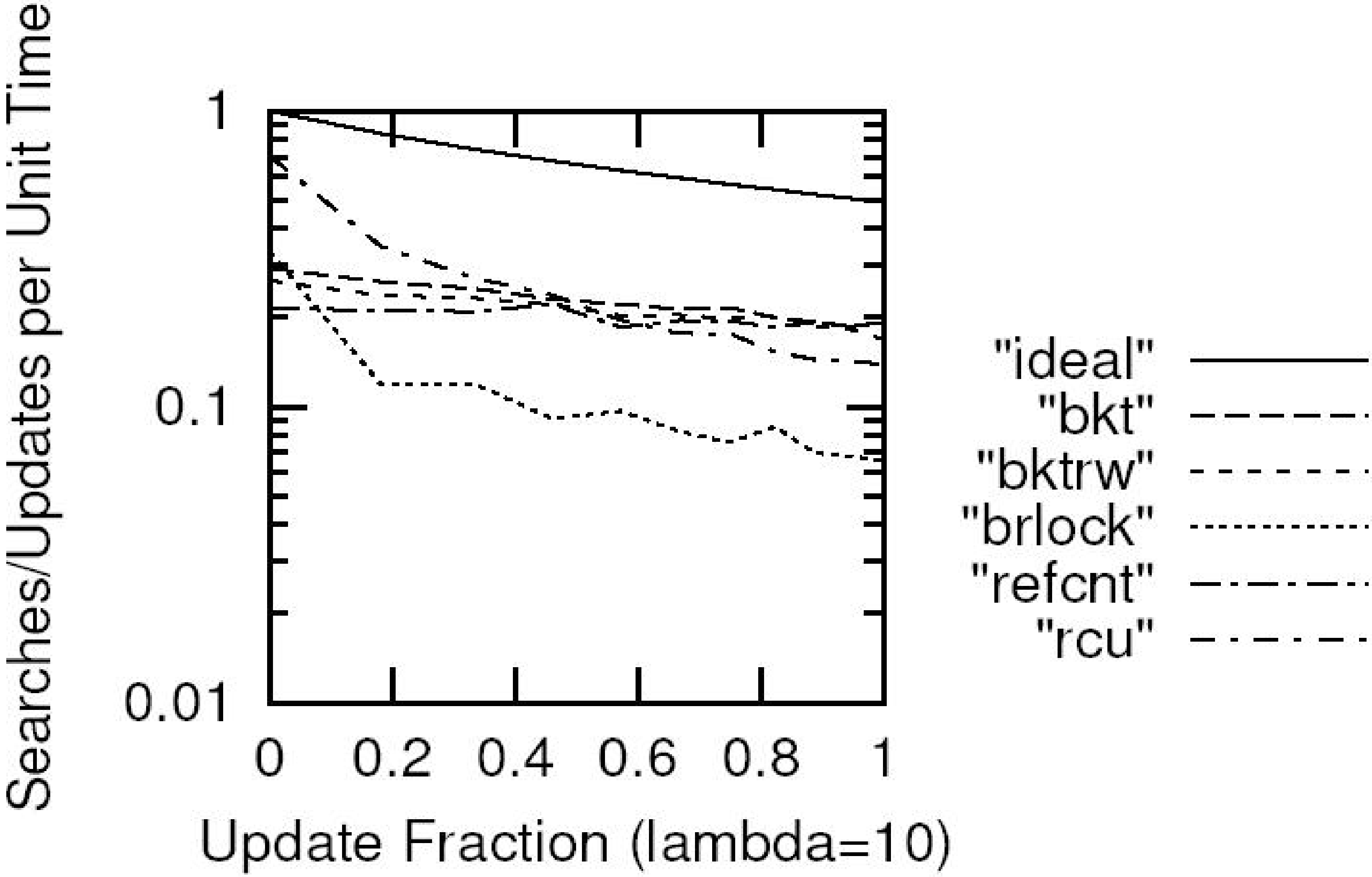
IPF Results for Mixed Workload



Opteron Results for Mixed Workload



PPC Results for Mixed Workload



Summary and Conclusions (1)

- RCU is great for read-mostly data structures
  - But not so great for update-mostly situations
  - RCU optimal when less then 10% of accesses are updates
- RCU updates cannot exclude readers
  - Good for deadlock avoidance and scalability
  - Adds complexity in some cases
    - But need 1,000s of instructions to make rwlock pay!!!
- RCU best when designed in from the start

Summary and Conclusions (2)

- Future/Ongoing Work
  - Testing RCU on more algorithms and data structures
  - Decreasing RCU grace-period overhead
    - Make things faster, increase RCU usefulness
  - Make RCU safe for realtime use (e.g., 250 *microseconds scheduling latency)*
  - *Enlist RCU to prevent DoS attacks*
  - Improve RCU ease of use

Legal Notice

- The views expressed in this paper are the author's only, and should not be attributed to IBM.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- IBM and AIX are registered trademarks of International Business Machines Corporation in the United States and/or other countries
- Linux is a registered trademark of Linus Torvalds
- Other company, product, and service names may be trademarks or service marks of others

Use

the right tool

for the job!!!