# Operating System Support for Shared Hardware Data Structures

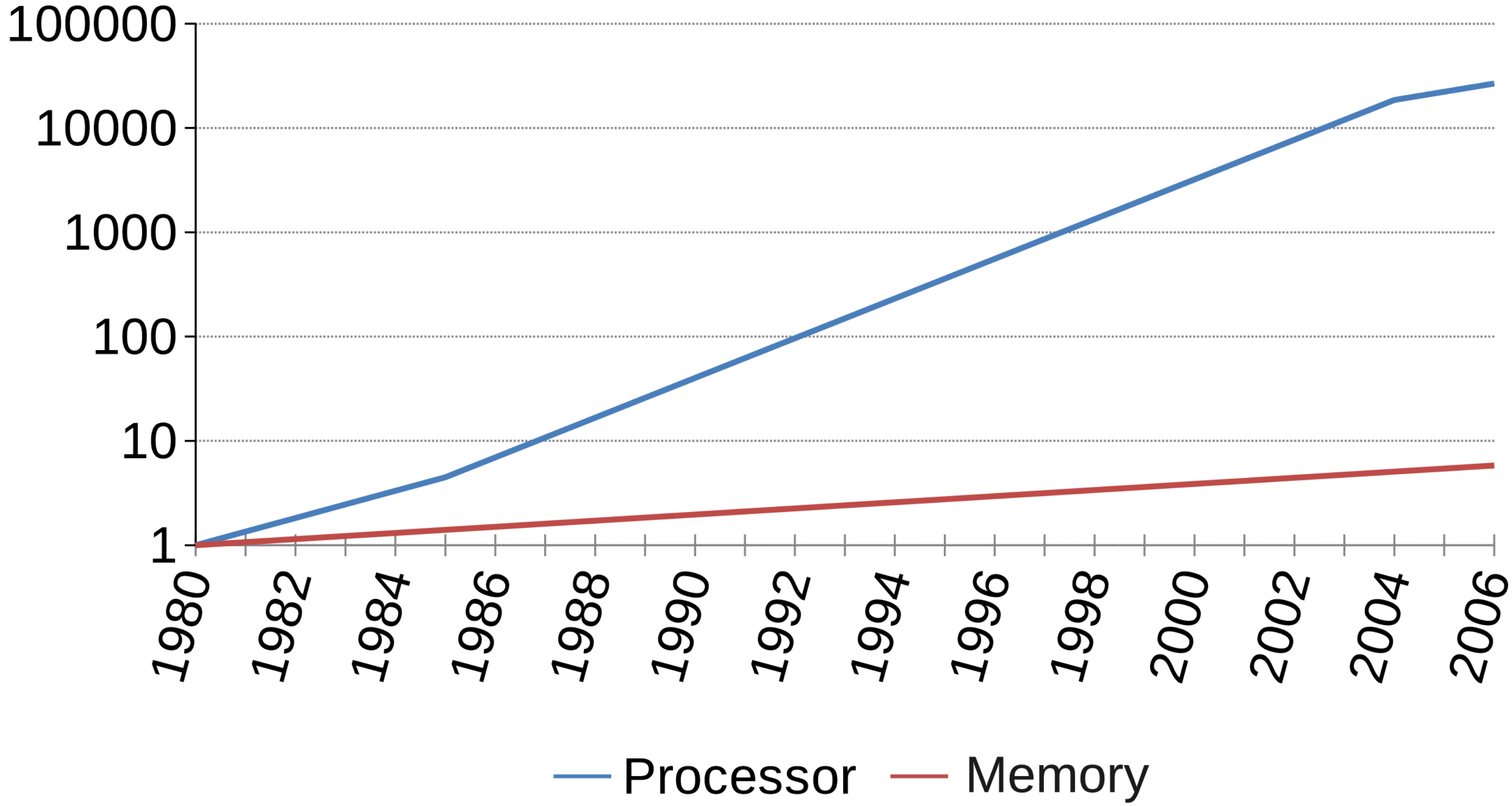
A dissertation thesis by Gedare Bloom

Advised by Bhagirath Narahari and Rahul Simha

Committee Members:
Gabriel Parmer, GWU CS
Evan Drumwright, GWU CS
Guru Venkataramani, GWU ECE

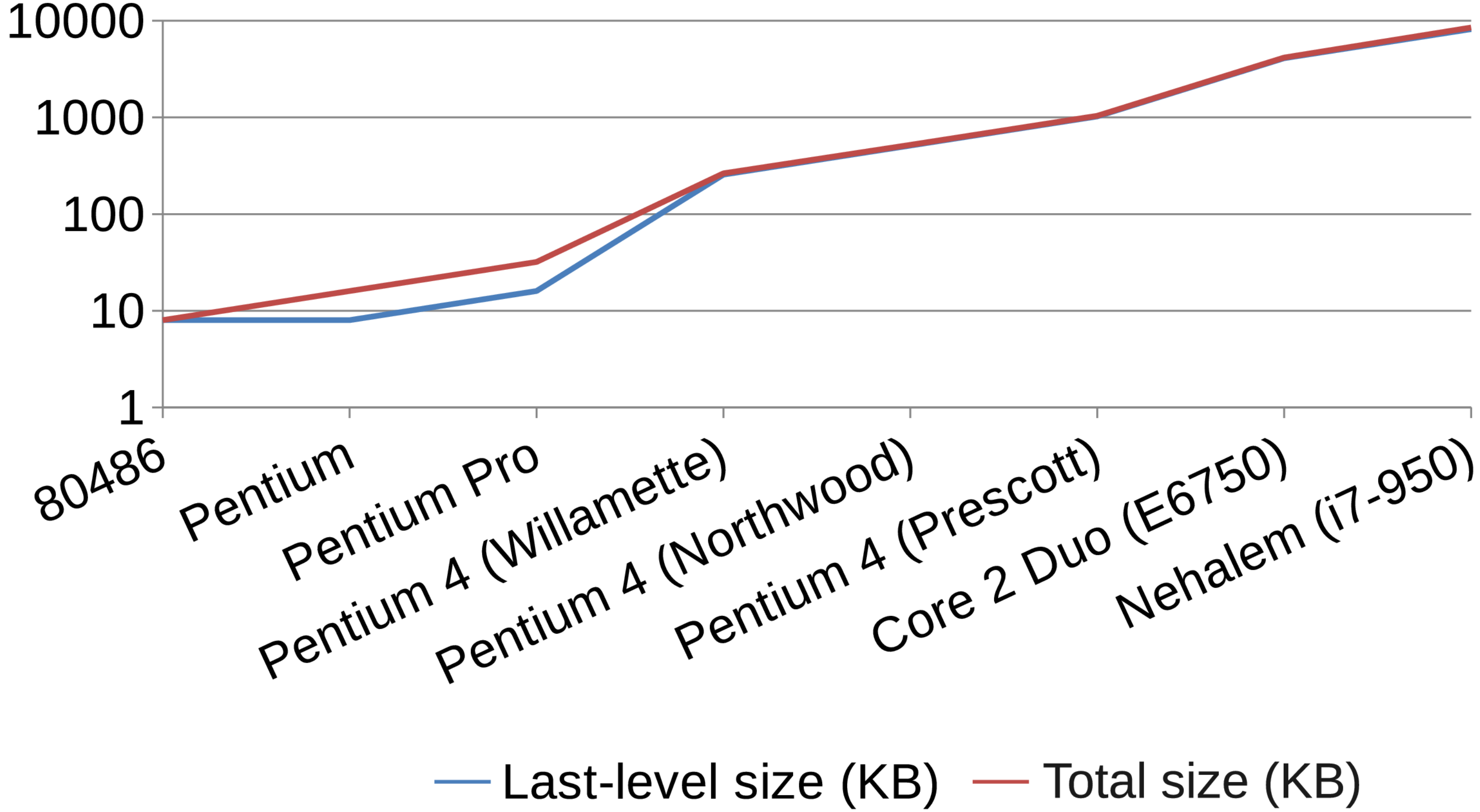
George Washington University
SEAS / Computer Science
19 Nov. 2012
Tompkins Hall 205

# Memory wall

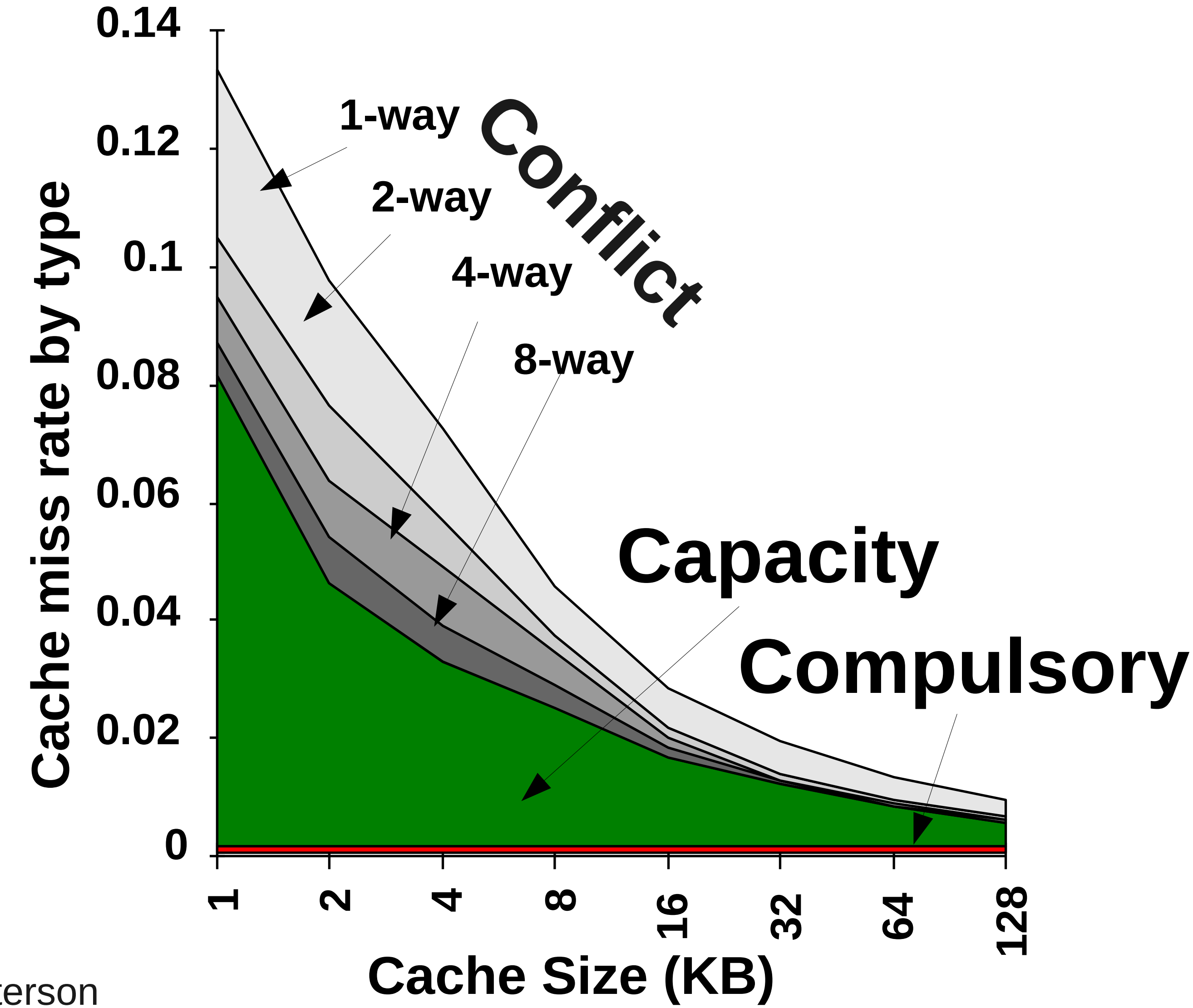



Hennessy and Patterson,"Computer Architecture: A Quantitative Approach," 4th Ed., 2007, Morgan Kaufman Publishers.

# Cache grows with bandwidth

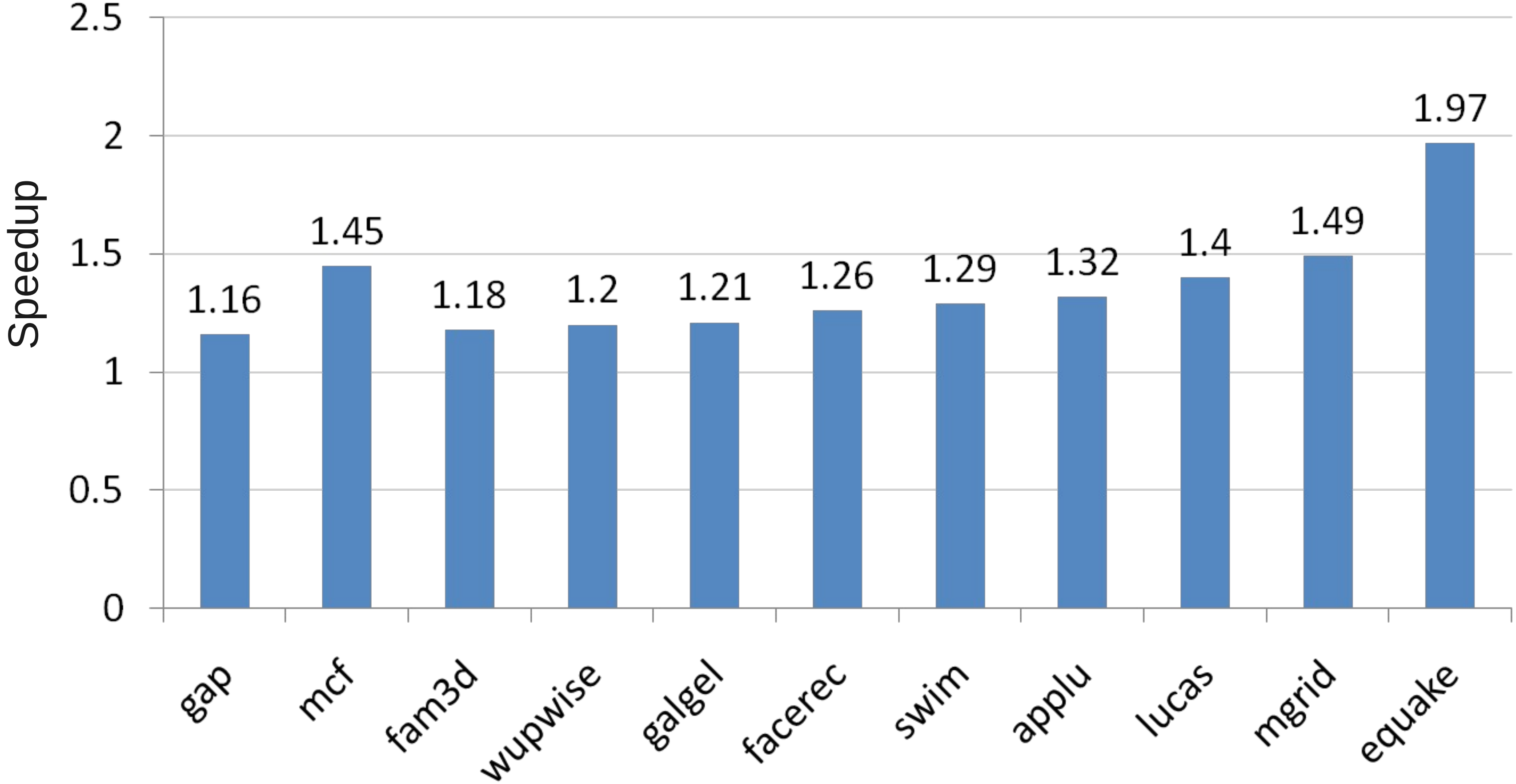


# Diminishing returns of cache growth



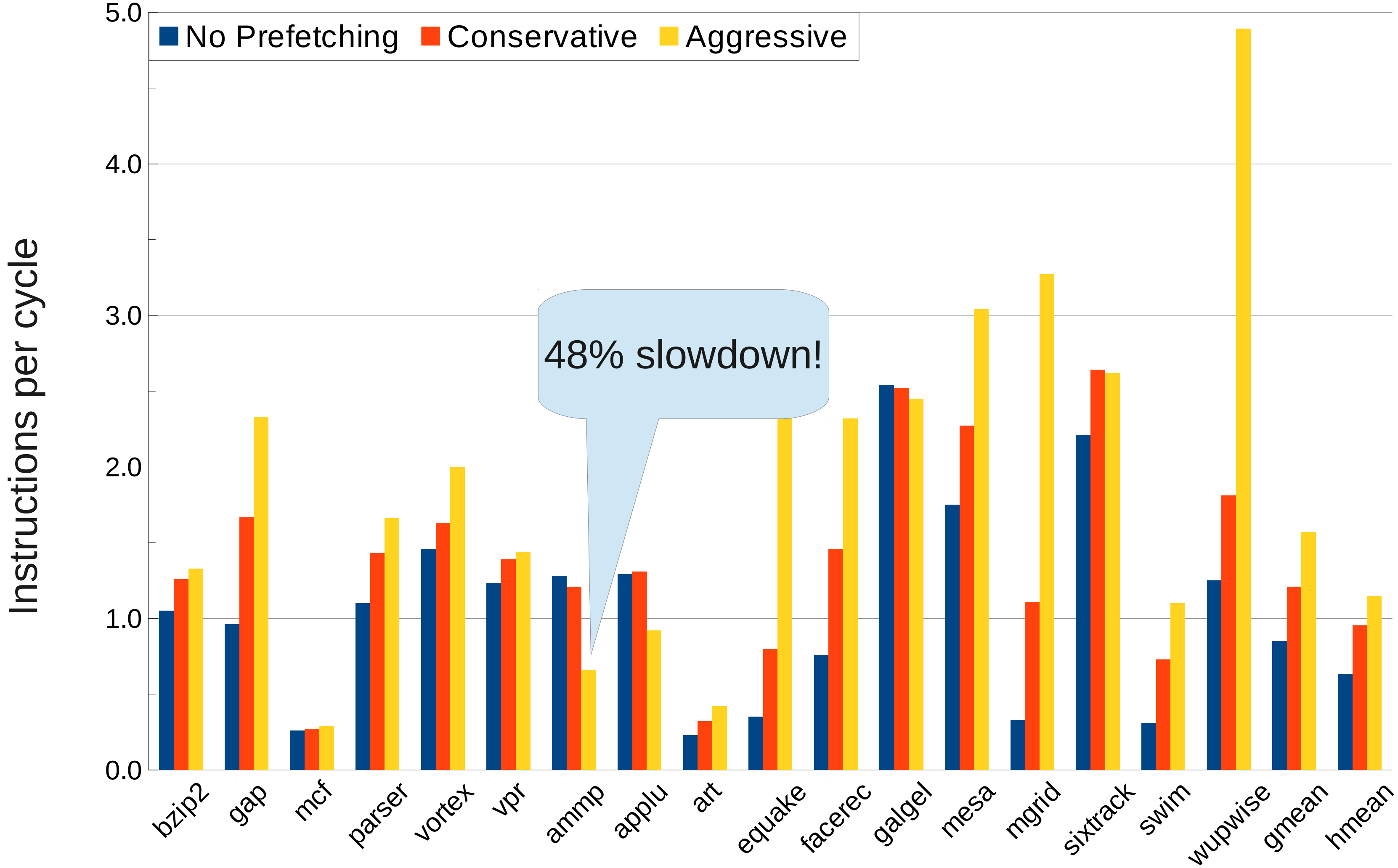

Hennessy and Patterson

# Prefetching picks up the slack




Hennessy and Patterson

# Prefetching not always beneficial




S. Srinath, O. Mutlu Kim Hyesoon, Y.N. Patt “Feedback Directed Prefetching” 2007

# Multicore game changer

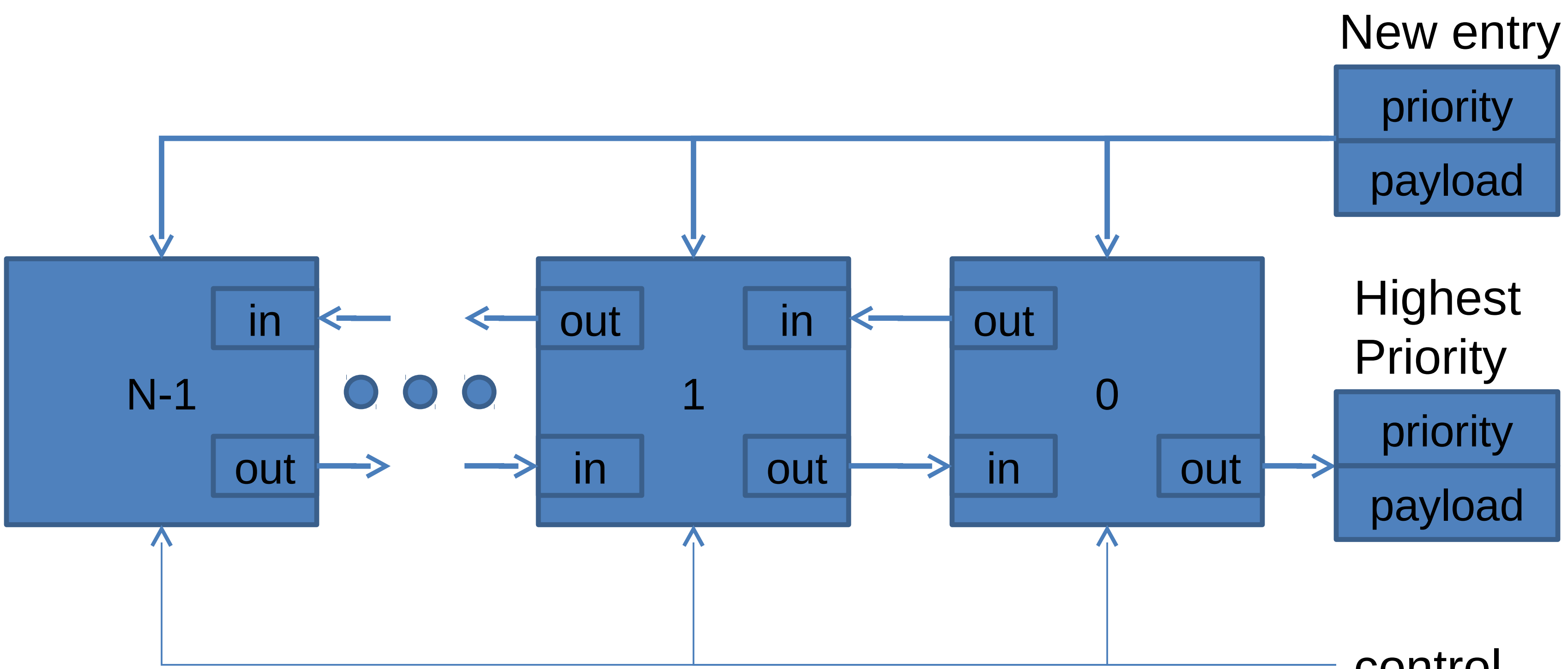
- Industry: “use spare transistors for cores”
- Problems
  - Parallel programming is hard
  - More data sharing: bad for cache
  - More bus contention: bad for prefetching

# Hardware data structures (HWDSs)

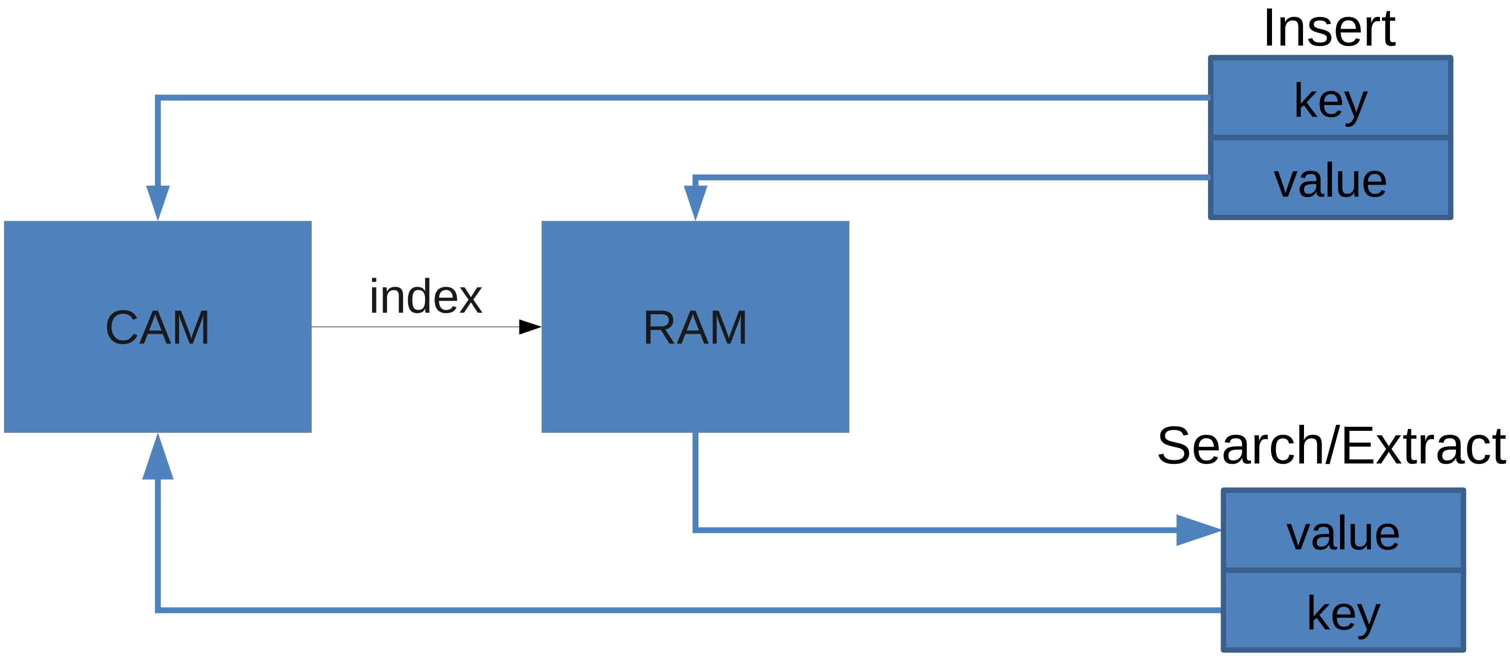
- HWDS = parallelism + smart storage
  - Advantage: reduce algorithmic complexity
  - Disadvantage: devotion of chip space

- New use for spare transistors

# Priority Queue HWDS



# Map HWDS



# Why PQ and Map?

- Critical to important software
  - 50-60% Dijkstra's algorithm
  - 30% grey-weighted distance transform
  - 40% discrete event simulation
  - 18% real-time task scheduling
  - 12% web browser
  - 20% physics simulations / scientific apps
  - 30%-900% referent object (bounds) checker

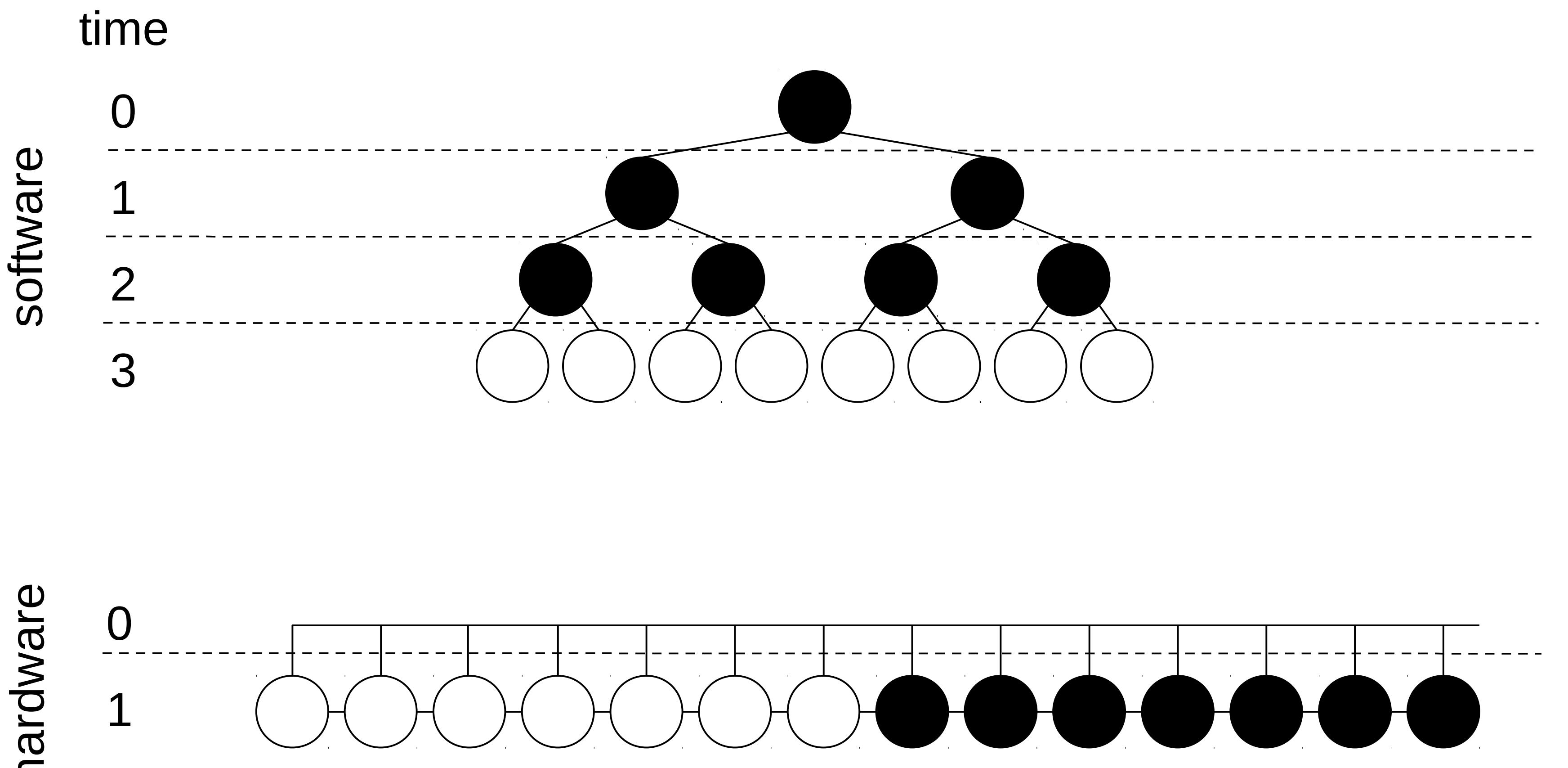
PQ

Map

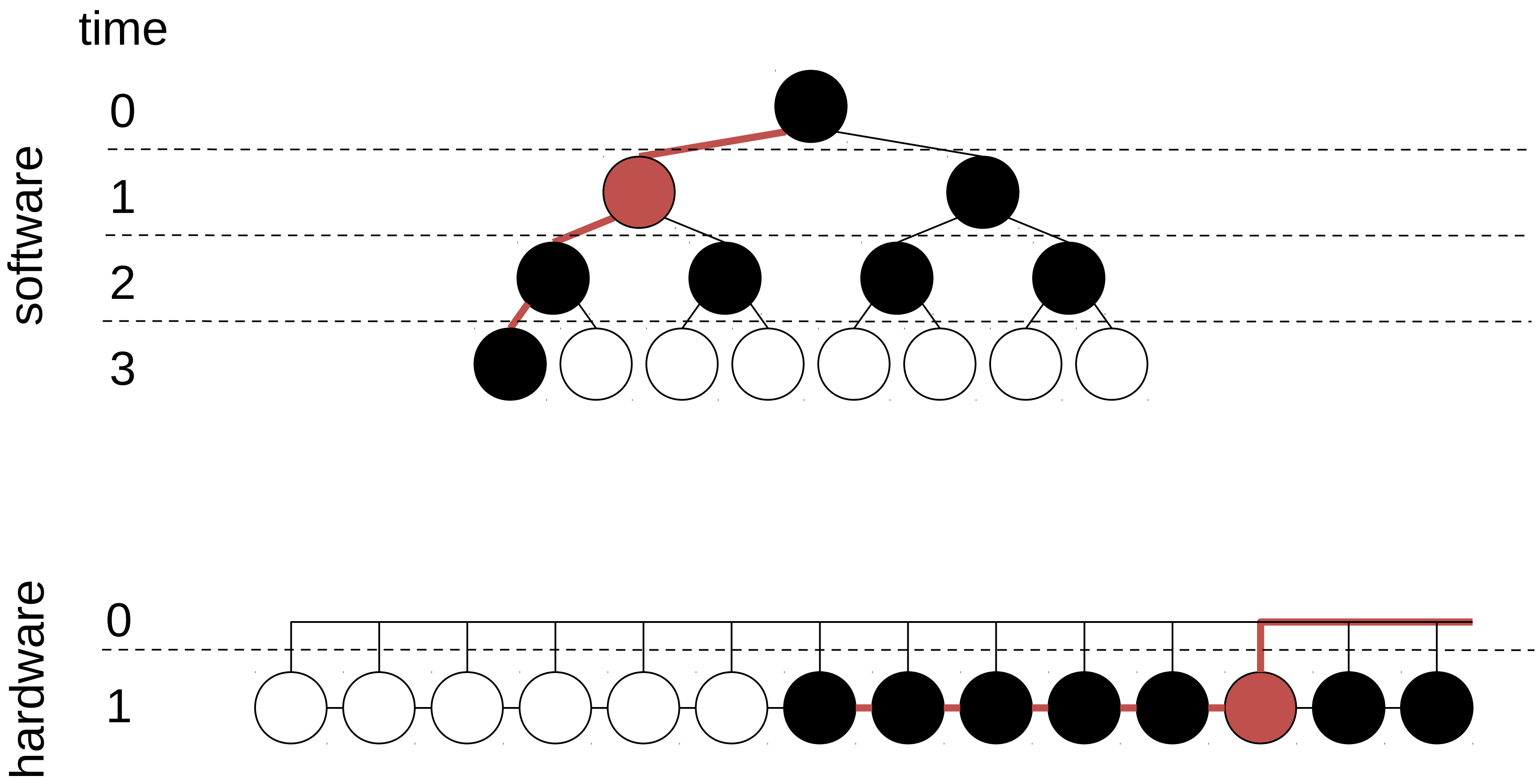
# OS Support for HWDSs

- This thesis contributes
  - DS operation API
  - Spilling HWDS overflow
  - HWDS Assignment for sharing hardware
  - Multiple kinds of HWDSs
  - Improved predictability for real-time systems
  - Cycle-accurate evaluation with real-world data

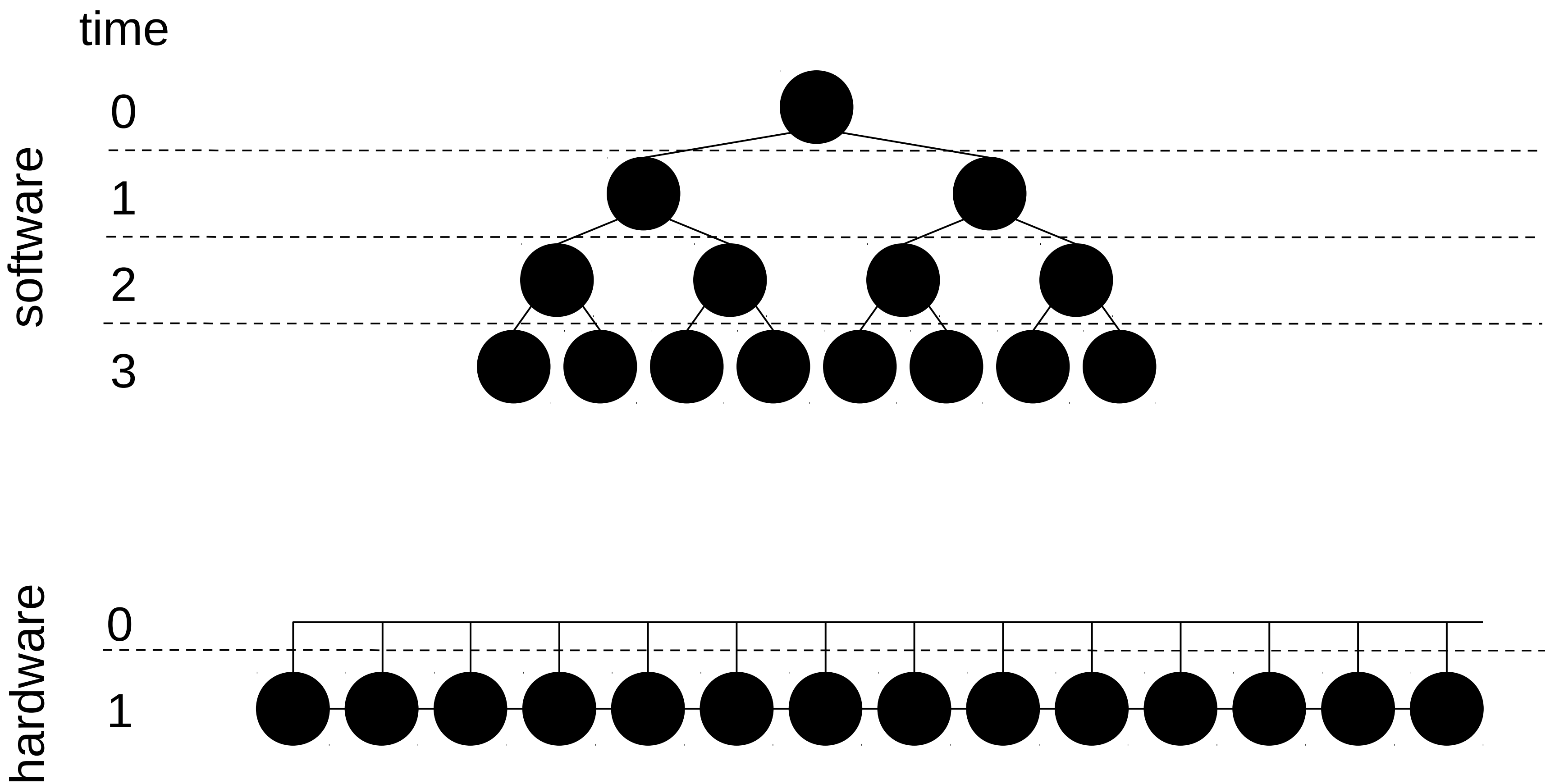
# Hardware’s advantage: parallelism



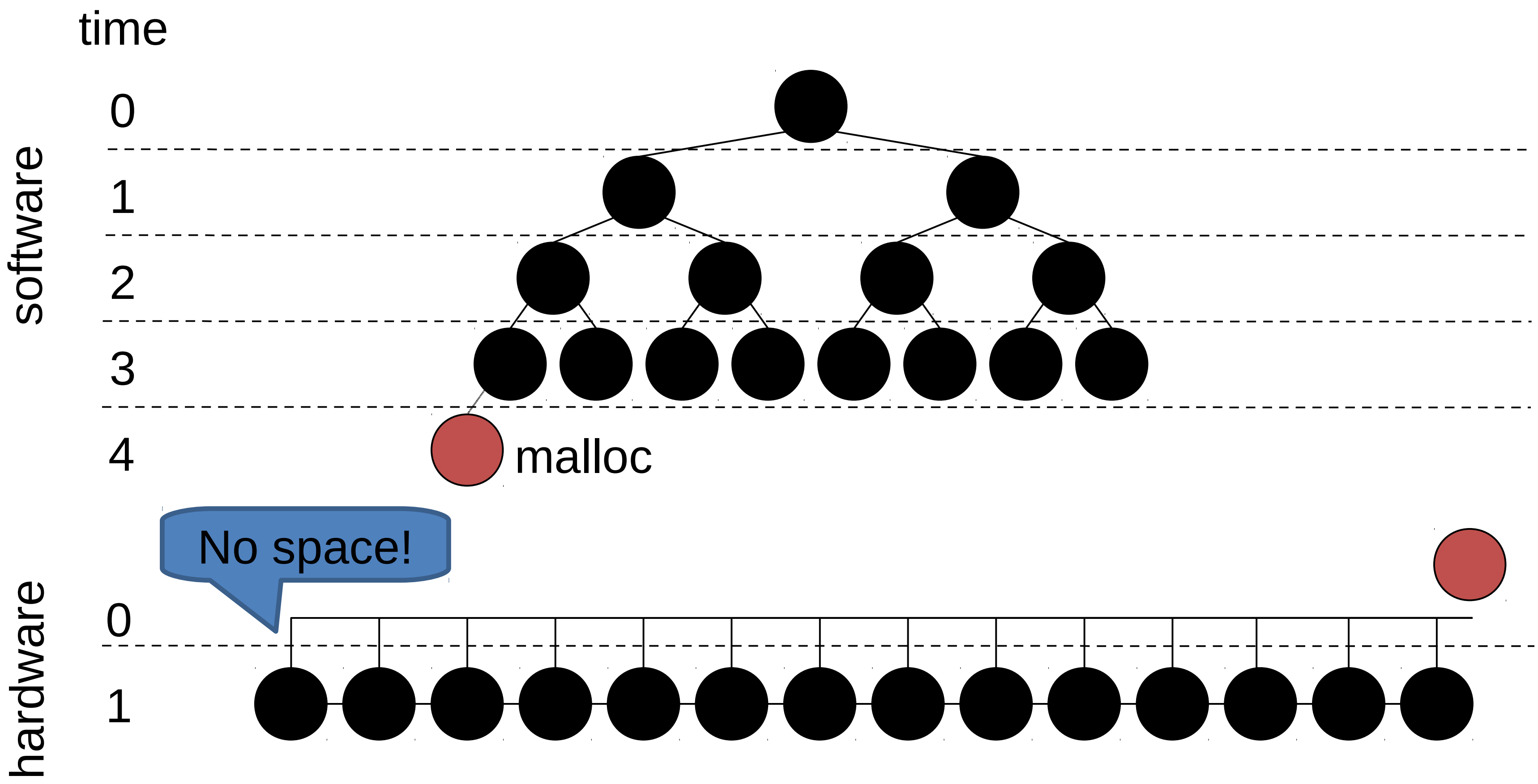
# Hardware's advantage: parallelism



# Hardware's disadvantage: capacity



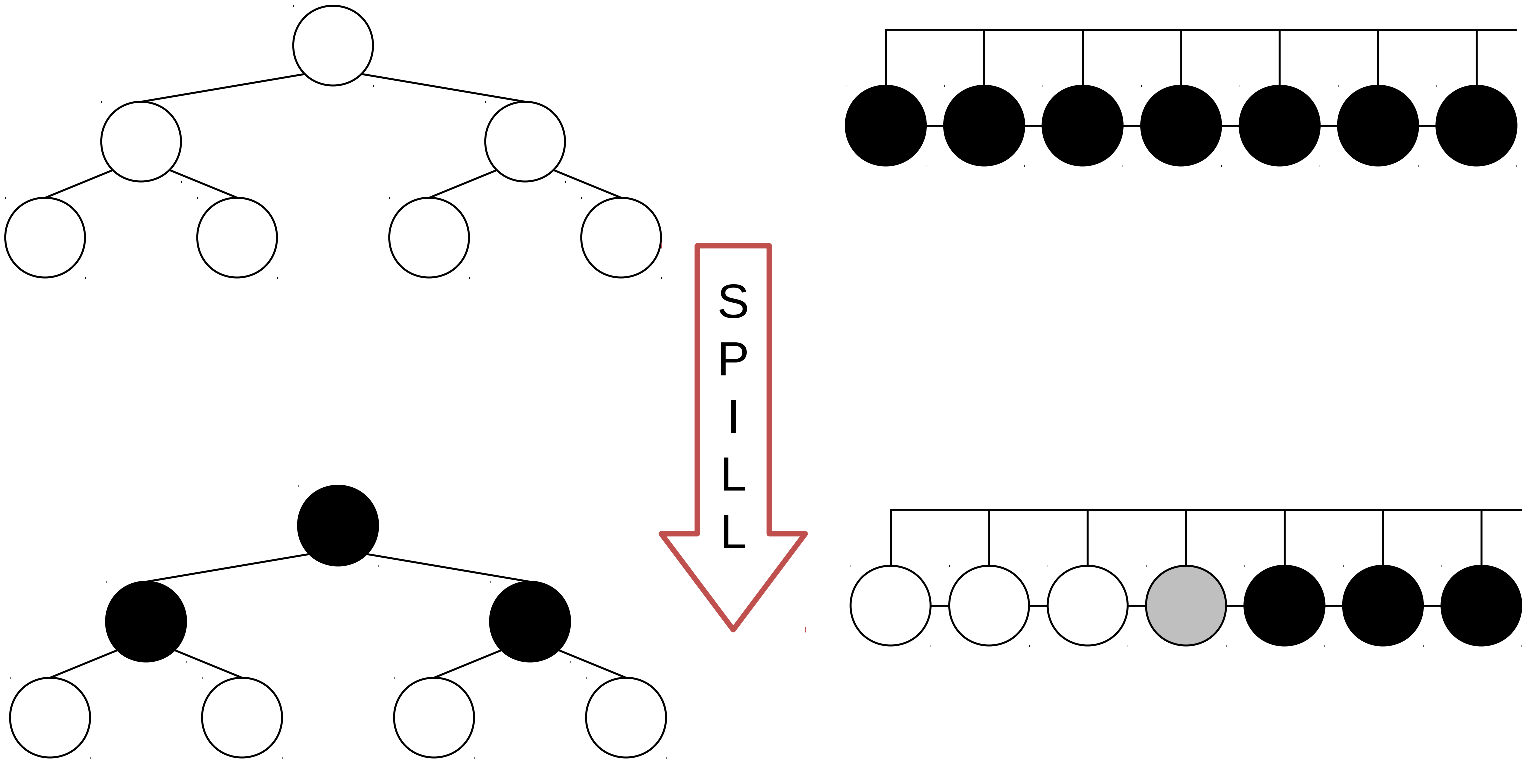
# Hardware's disadvantage: capacity



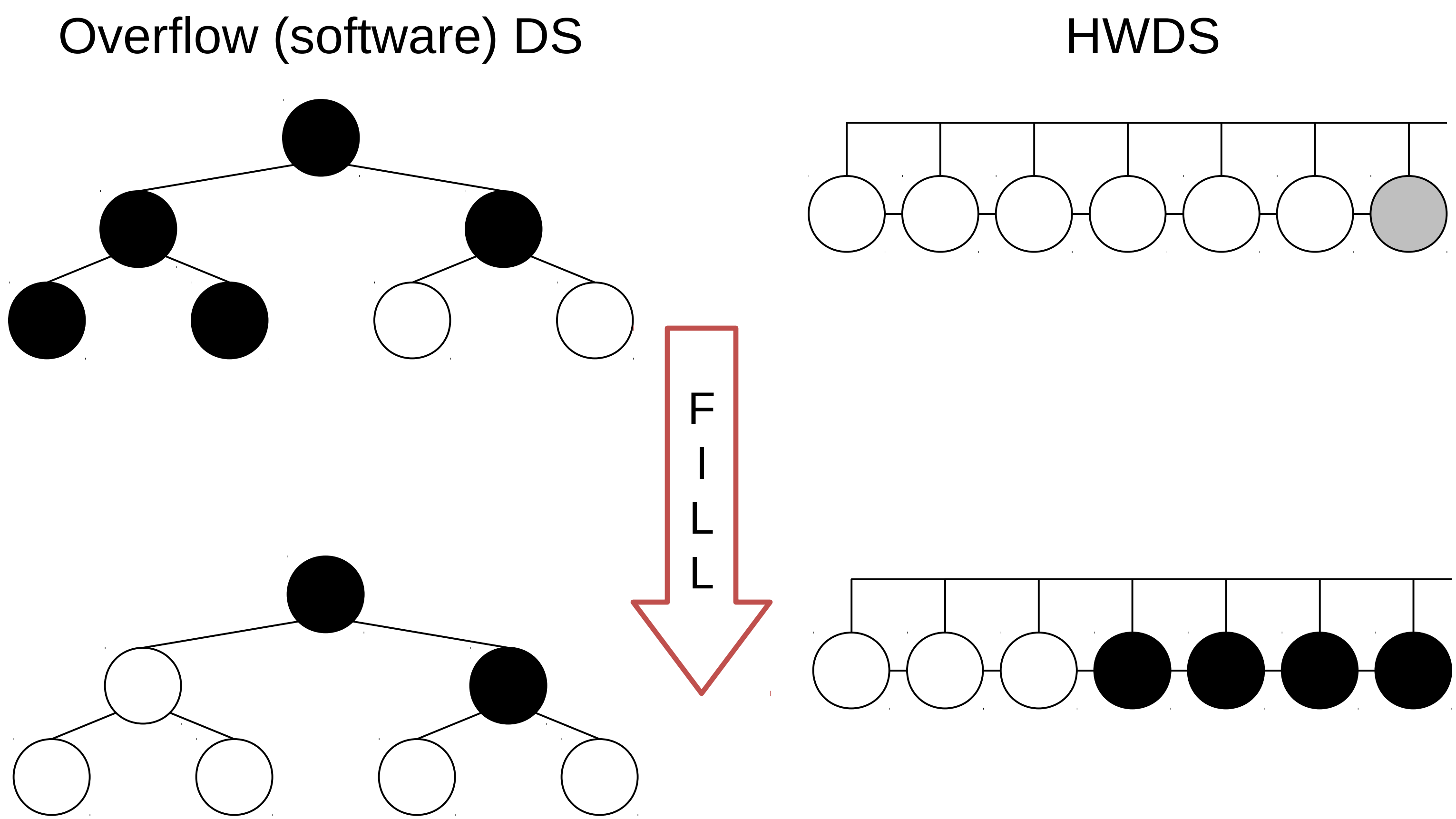
# Solving capacity by spilling

Overflow (software) DS

HWDS



# Solving capacity by filling



# HWDS Overflow: spill and fill

- United HWDS
  - Structural locality, HWDS knowledge
- Split HWDS
  - Spill inserts to overflow DS
- Overflow support needed
  - **spill / fill** HWDS instructions
  - **Exception handling** / interposition
  - Size limits
  - **What** and how much to spill / fill

# PQ Overflow

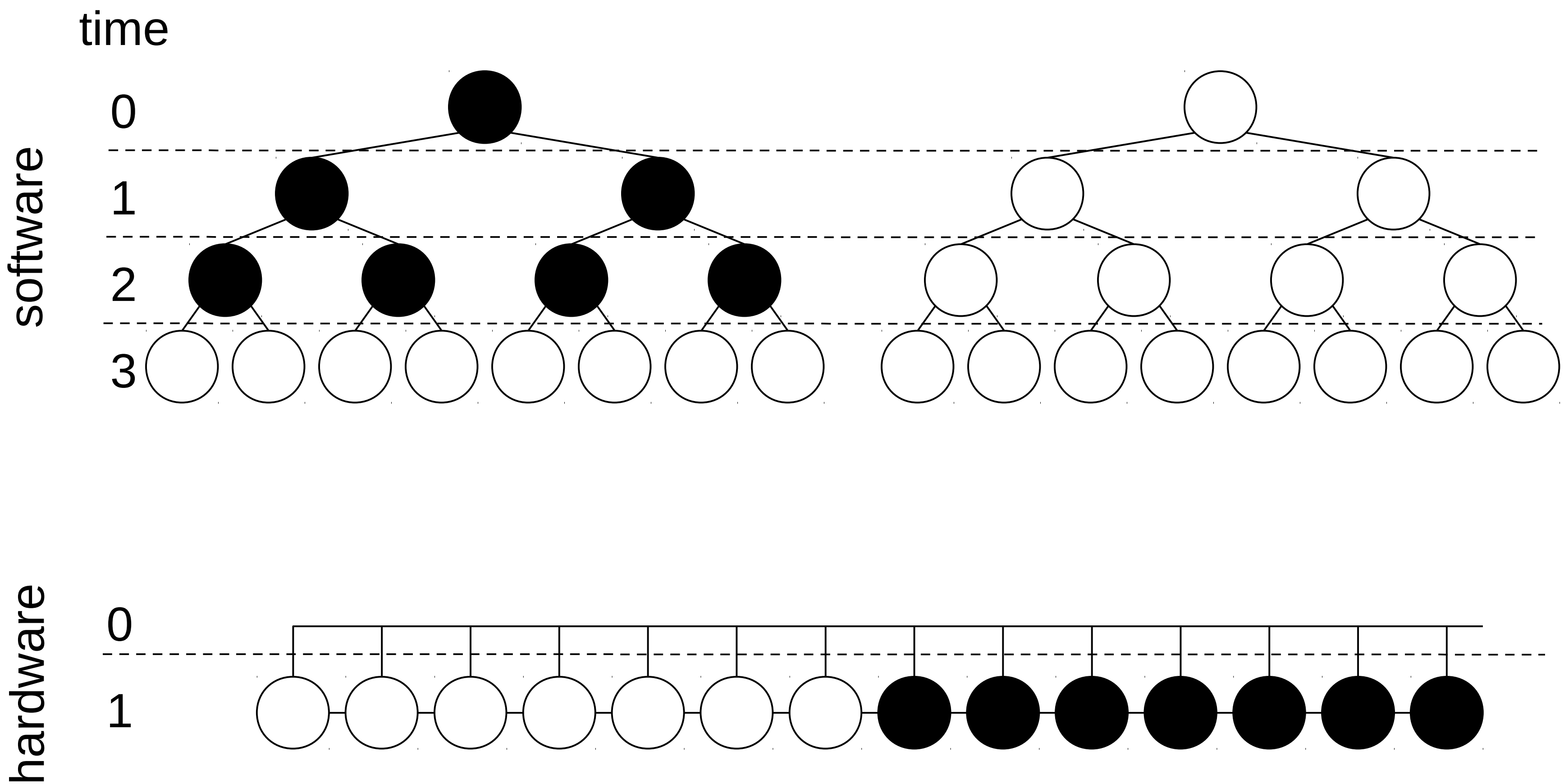
- Structural locality
  - Spill lowest priority node
  - Fill highest priority node
- Insert-after-spill violates ordering
  - Hardware marks ordering violations
  - Filling clears the marks
- United HWDS: merge-sorted linked list

# Map Overflow

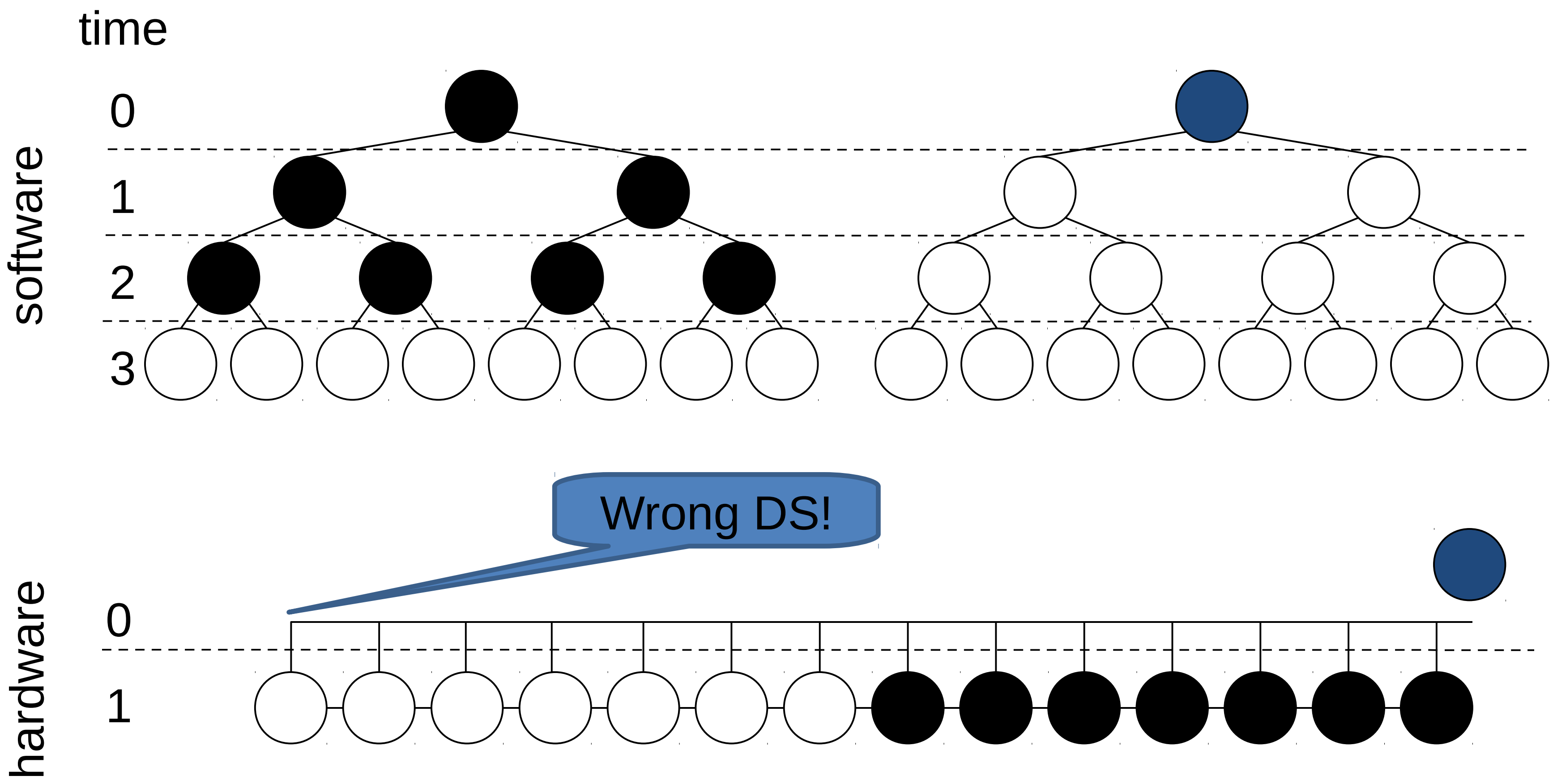
- Spill locality
  - **Least recently used (LRU)**
  - Least frequently used (LFU)
- Fill locality
  - Most recently used (MRU)
  - Most frequently used (MFU)
- Search fail-over
  - **Fill after search (FAS)**

United HWDS?

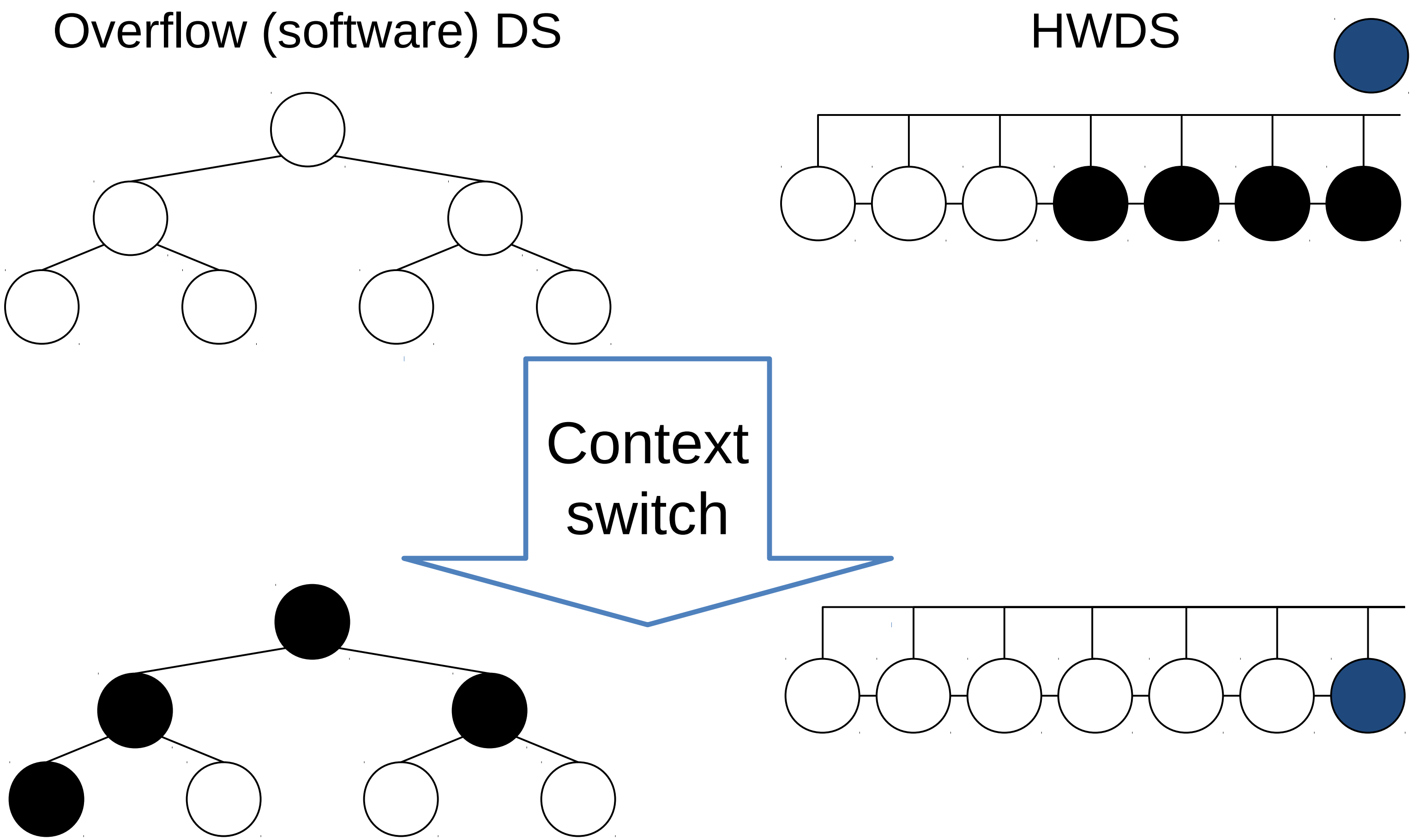
# Hardware's disadvantage: sharing



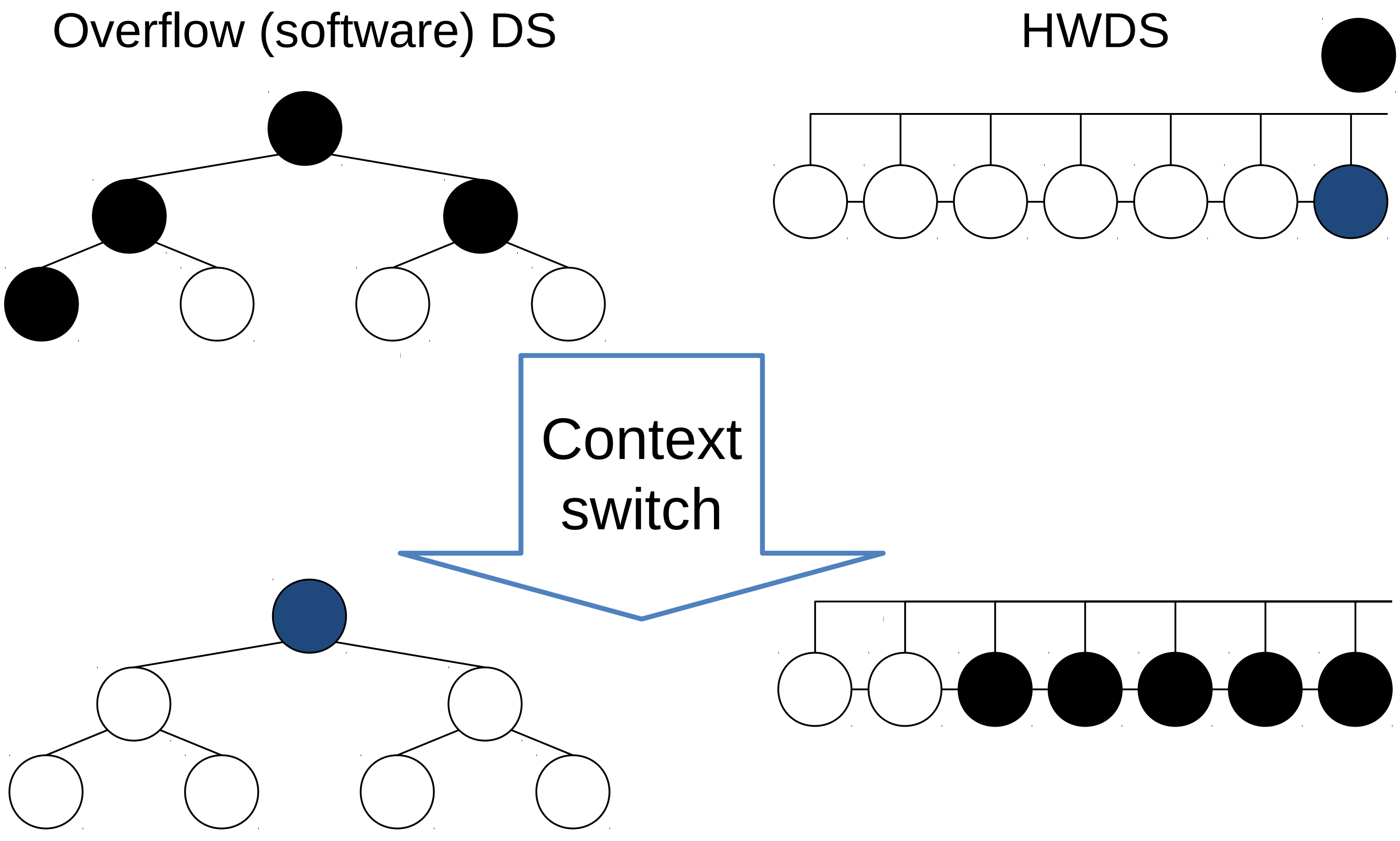
# Hardware's disadvantage: sharing



# Sharing hardware: context switch



# Sharing hardware: context switch



# Sharing hardware: Assignment

- HWDS or software implementation?
- Static / dynamic algorithms
  - **Context switch**
    - **How much** to restore
    - Pinning
    - **Eviction**
  - **Size limits**
  - **Interposition**
  - Stalling

# HWDSs for Hard Real-Time

- HWDS reduces variance in operation times
- Apply OS support for HWDS to real-time
  - Four HWDS assignment algorithms
    - Software-only assignment (SOA)
    - Hardware-only assignment (HOA)
    - Priority-aware assignment (PAA)
    - Context switch cost-aware assignment (CSCAA)
- Real-world applications improve by 5–15% utilization (see [1])

1. G. Bloom, G. Parmer, B. Narahari, and R. Simha, “Shared Hardware Data Structures for Hard Real-Time Systems,” 12th International Conference on Embedded Software. EMSOFT 2012. October 2012

# Experiments: Setup

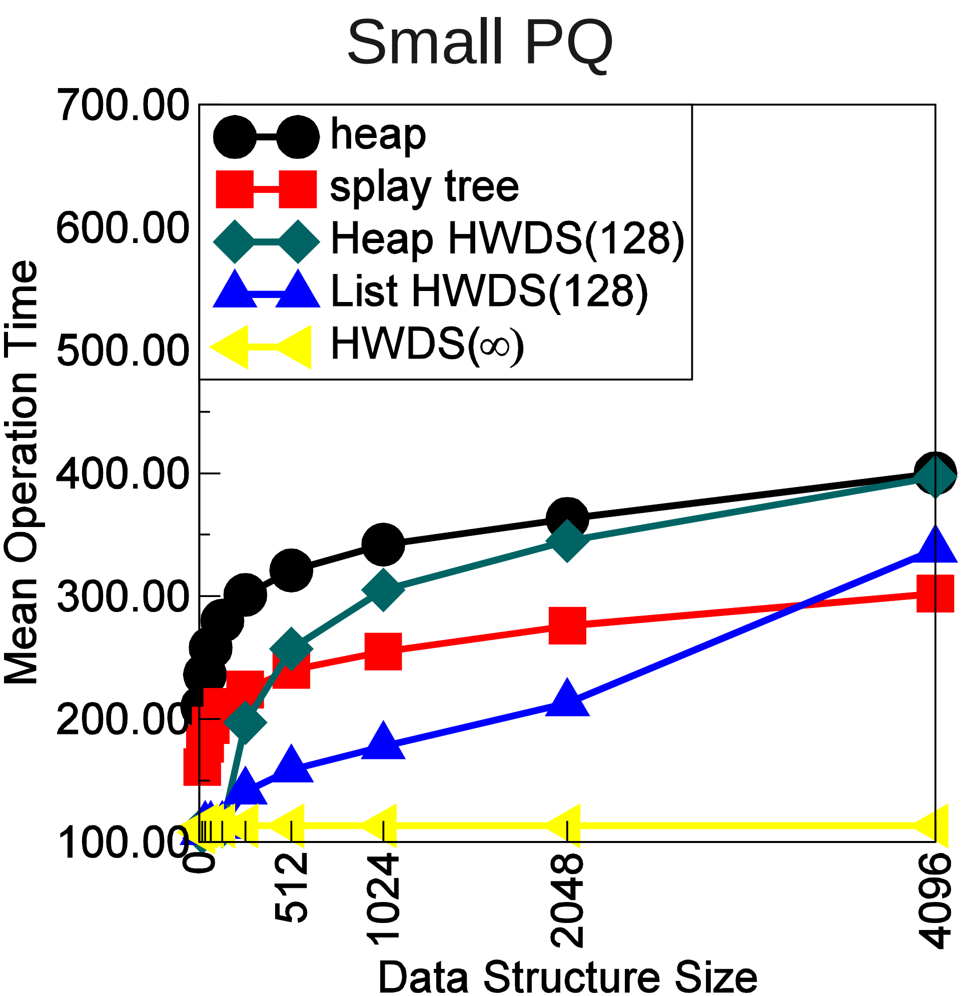
- HWDSs implemented in Simics/GEMS
  - New functional unit: atomic, non-speculative
  - 12-cycle HWDS instructions
- OS support in RTEMS
  - Exception handling and interposition library
  - Overflow DS
  - HWDS context switch and assignment
  - Task scheduling

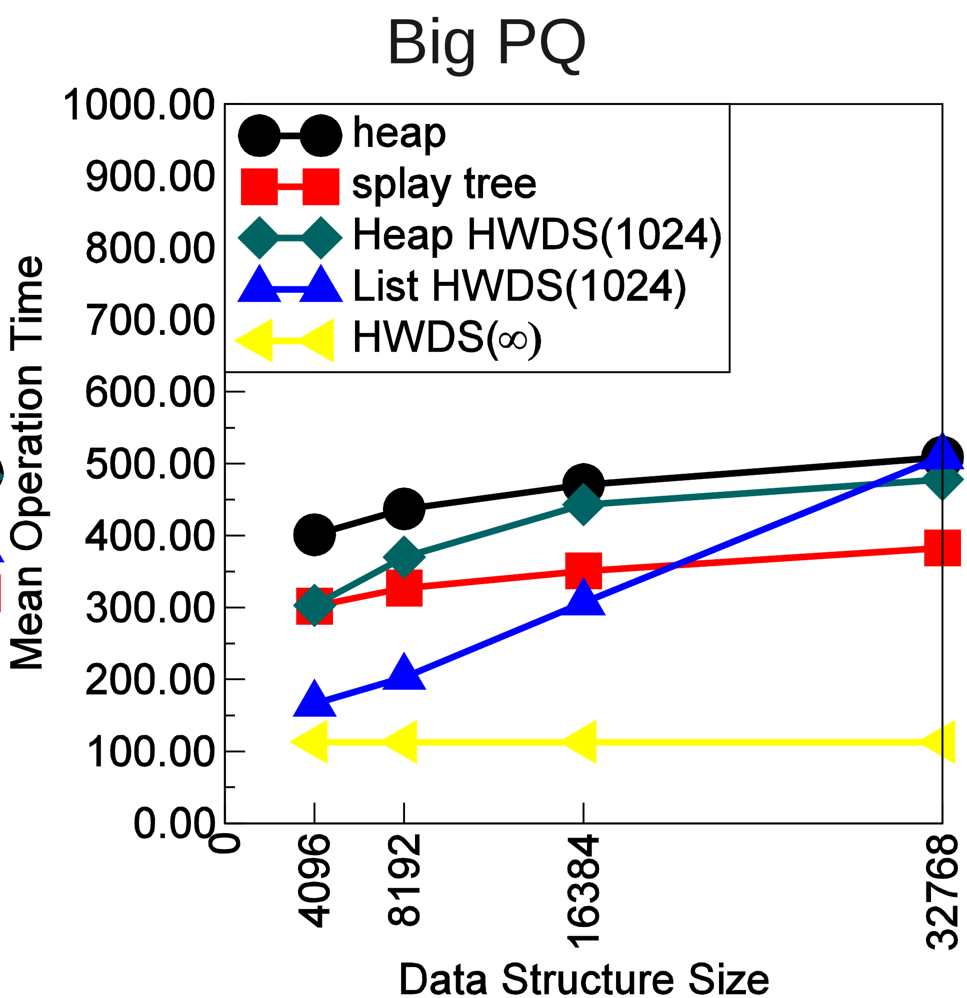
# Synthetic benchmarks

- Pending event set: classic hold [1]
  - Build PQ to max size
  - Execute *hold* operations
- Skewed search [2]
  - Build map to max size
  - Execute search and update operation

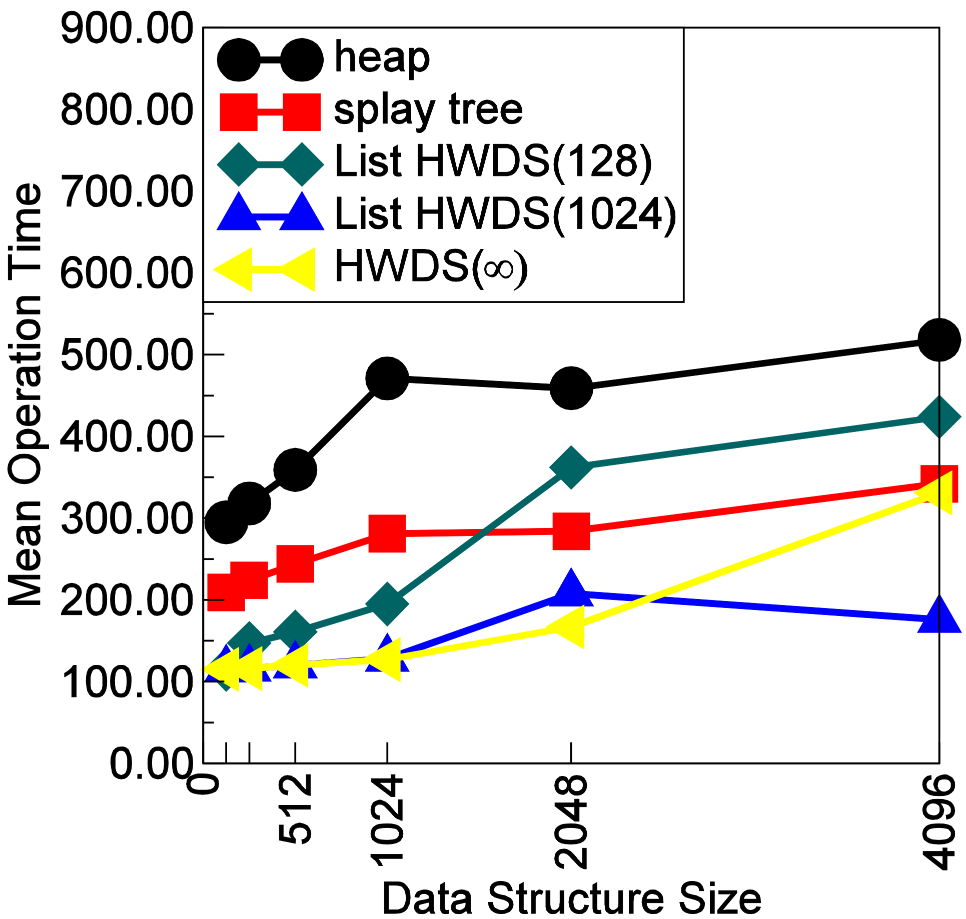
1. Douglas W Jones. An empirical comparison of priority-queue and event-set implementations. Commun. ACM, 29(4):300311, April 1986.
2. Jim Bell and Gopal Gupta. An evaluation of self-adjusting binary search tree techniques. *Software: Practice and Experience*, 23(4):369–382, April 1993.

# Overflow with HWPQ



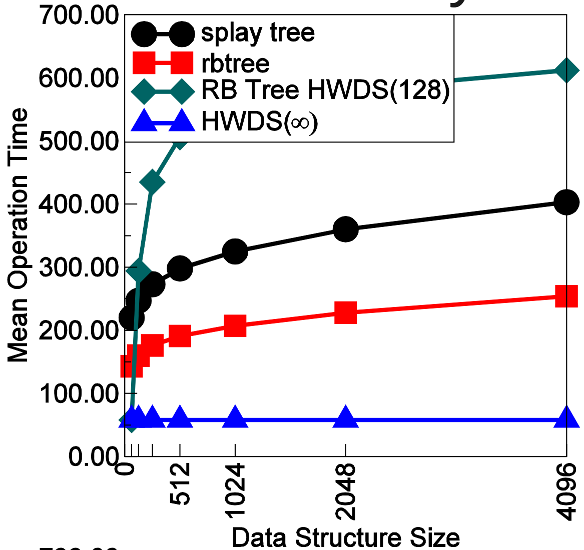
# HWDS sharing: 4 same-sized PQs

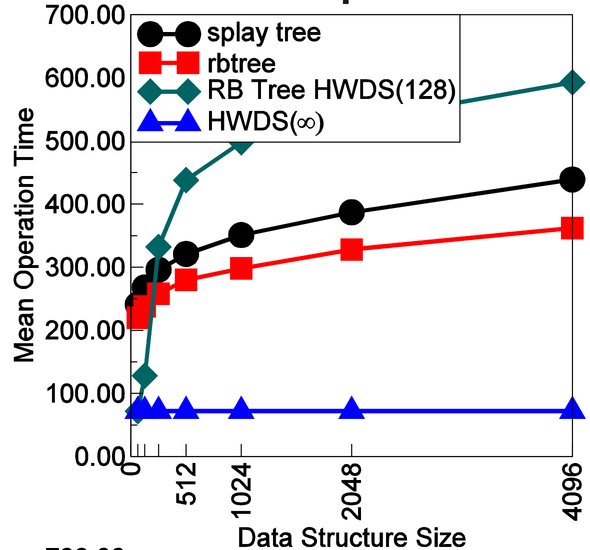


# Map overflow: spill last

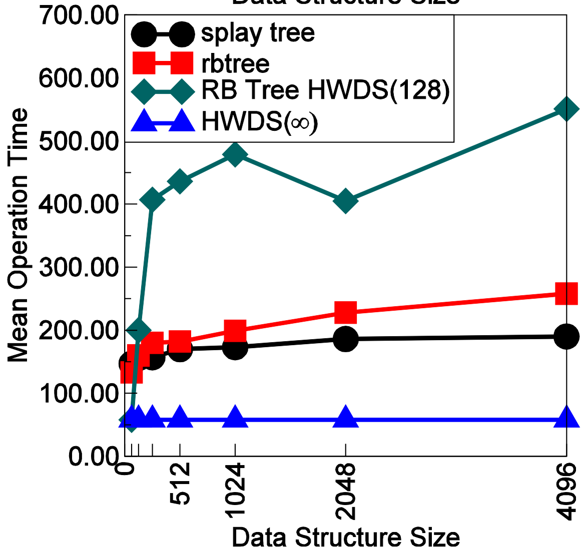
Read-only | 80% updates

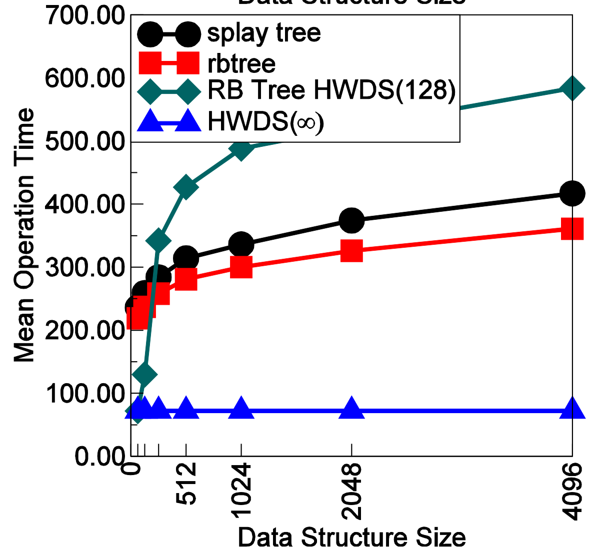
No skew




Large skew



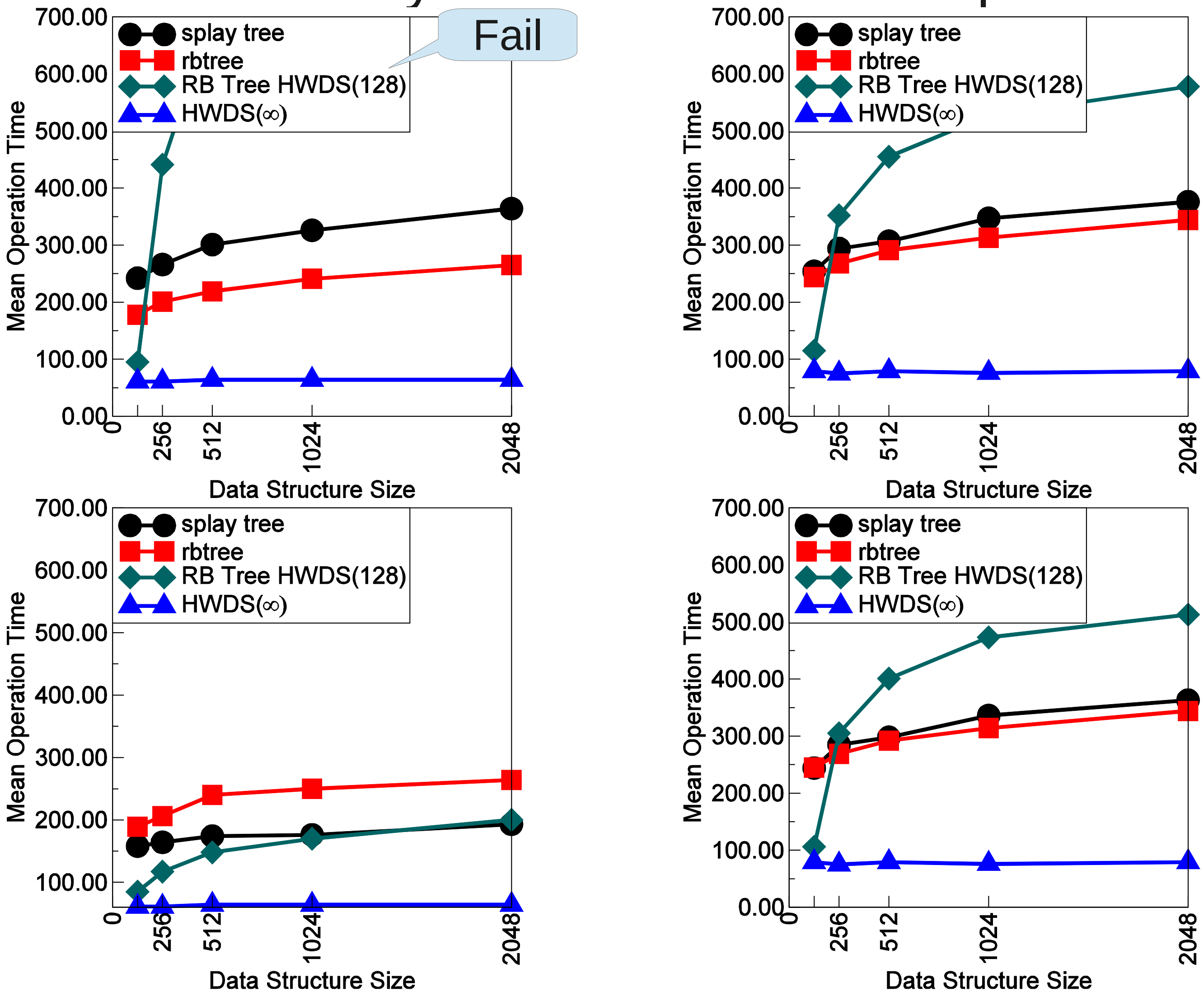



# Map overflow: LRU, FAS

Read-only | 80% updates

No skew

Large skew

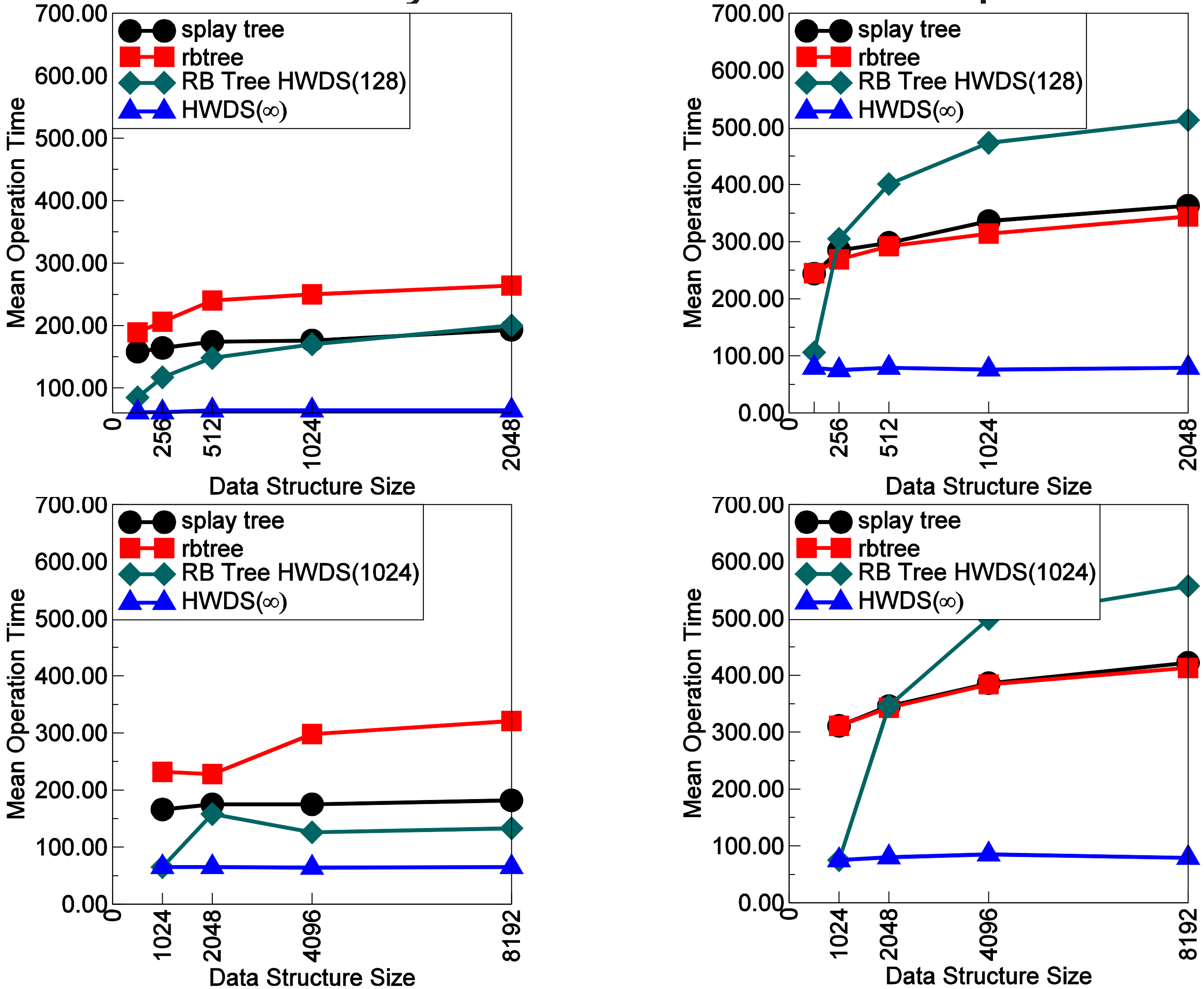


# Bigger HWDS, LRU FAS, skew

Read-only 80% updates

128

1024

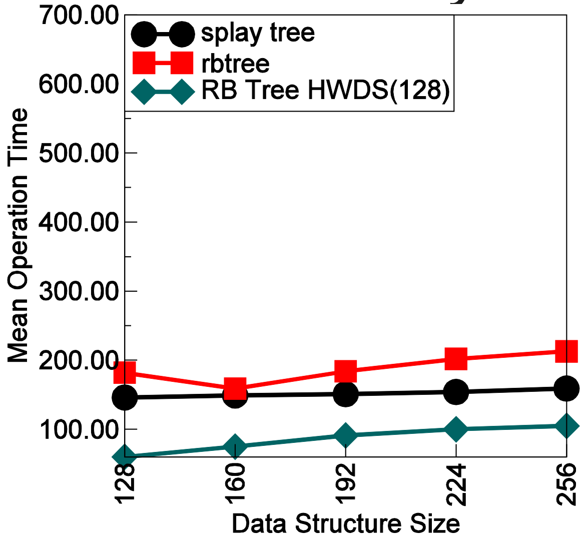


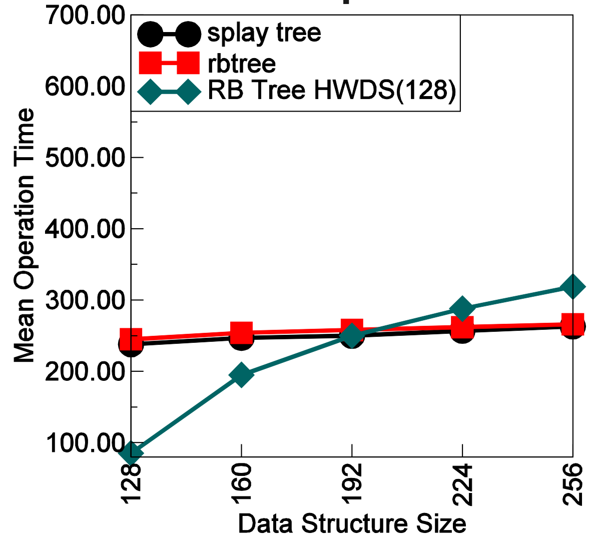
# Drill Down: LRU FAS w/ large skew

Read-only

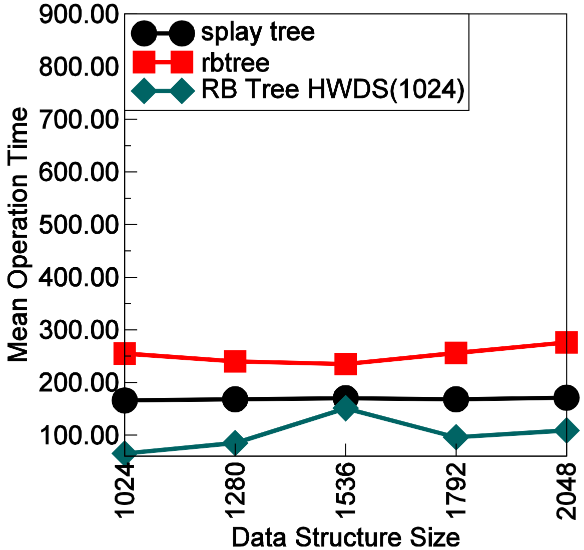
80% updates

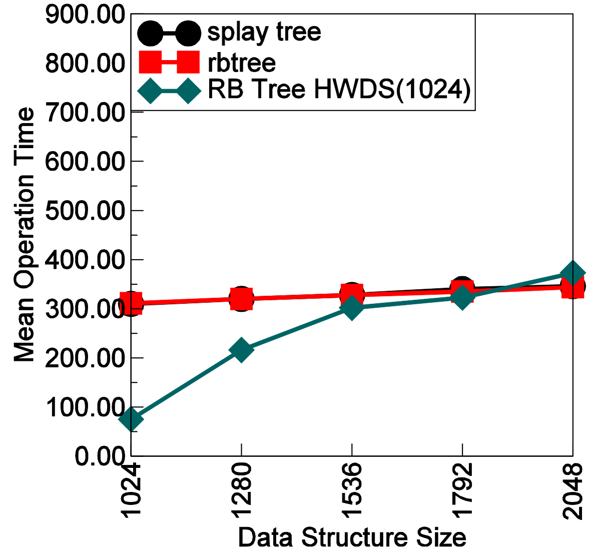
128



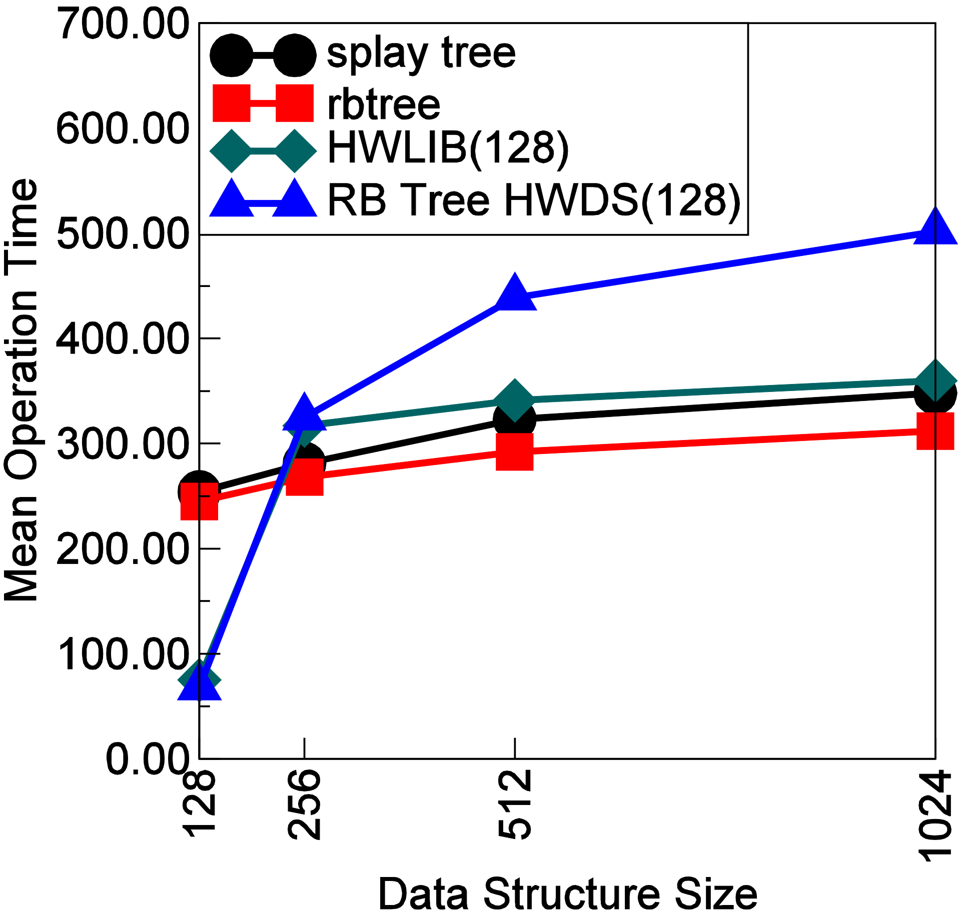

1024



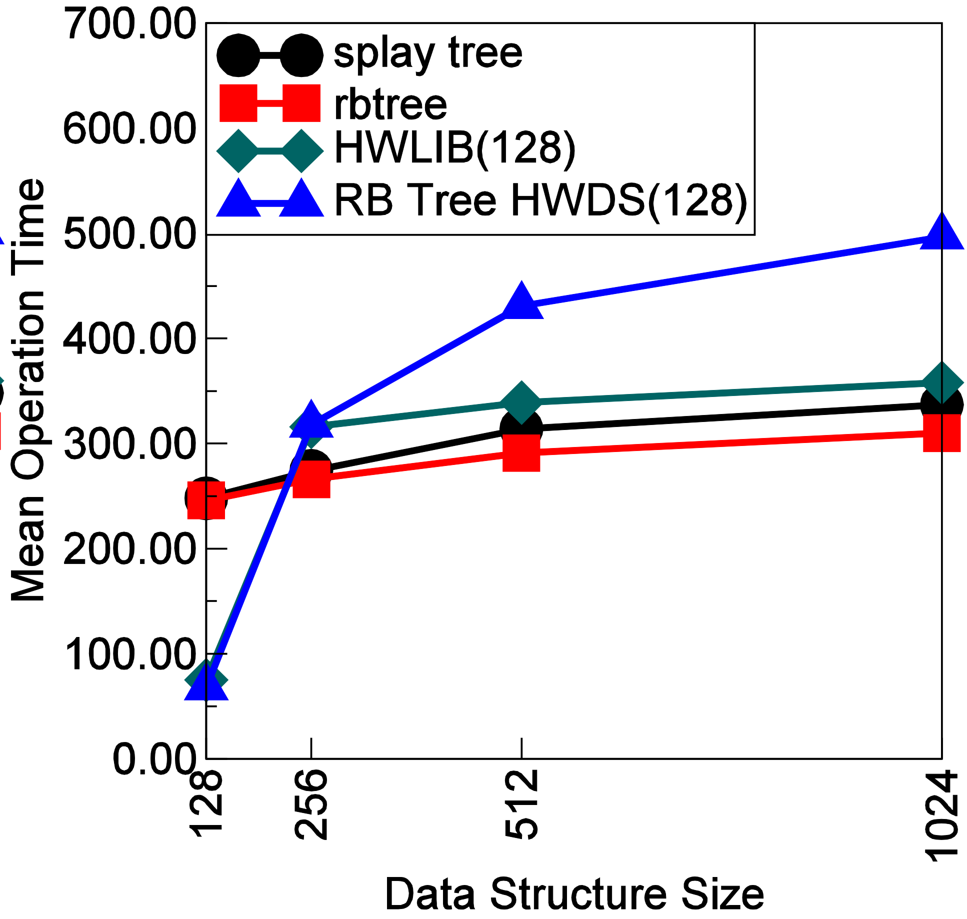



# Evict on extract (4Kops, 80% updates)

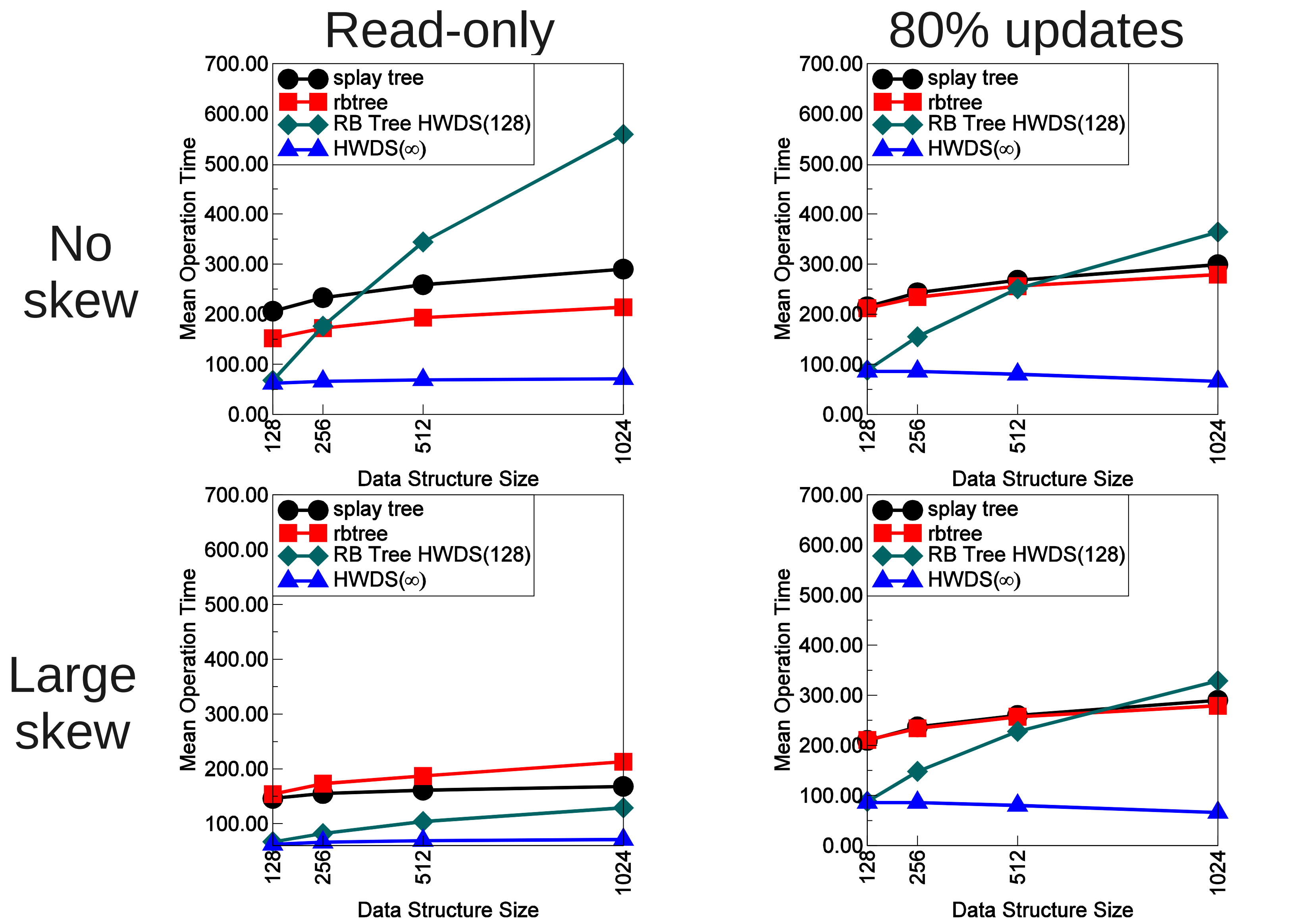
No skew




Large skew



# Sharing different size maps (4Kops)



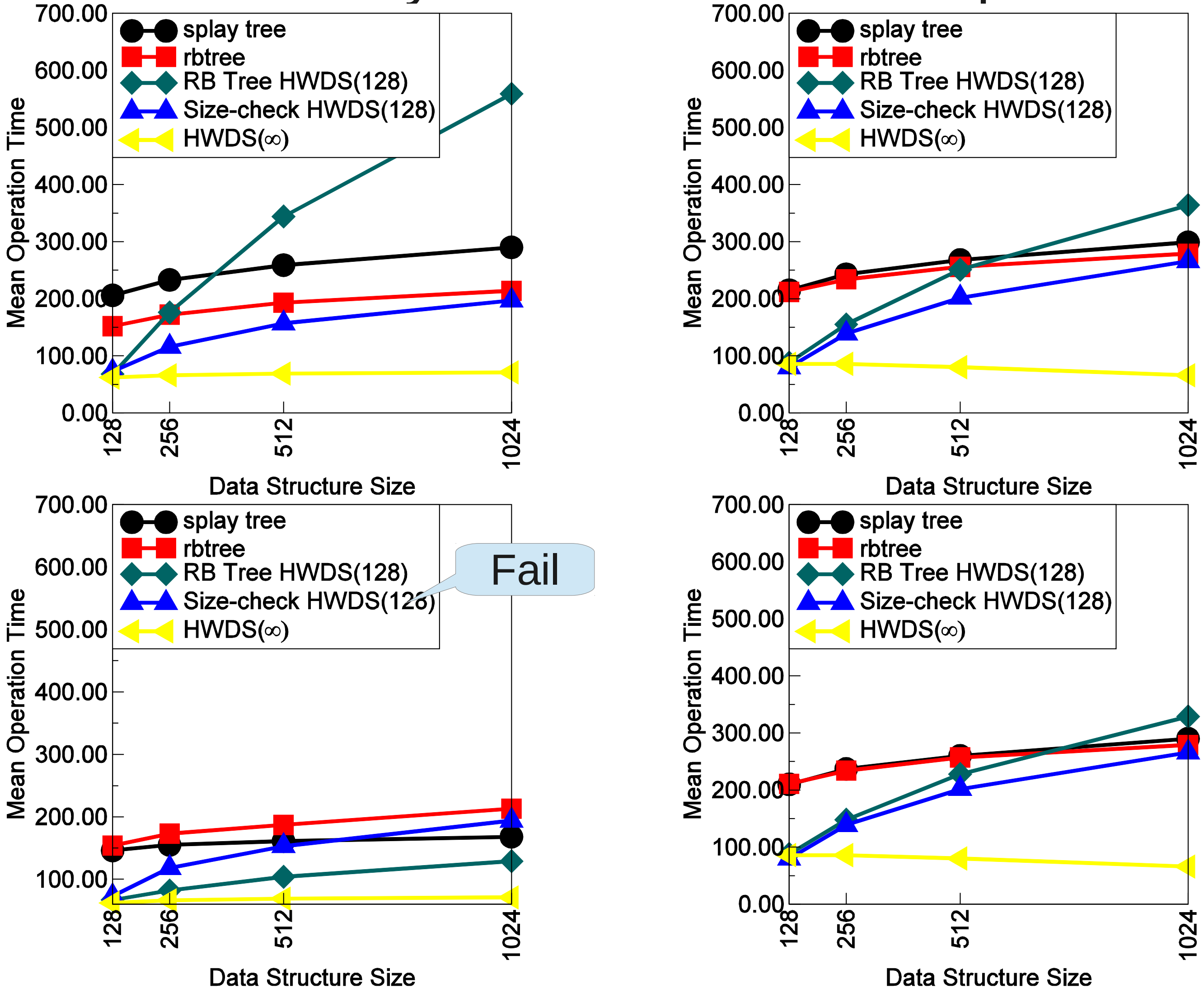
# Size checked Map HWDS (4Kops)

Read-only

80% updates

No skew

Large skew



# Synthetic Benchmarks: Summary

- Overflow and useful SW:HW ratios
  - < 16:1 for PQ – advantage United HWDS
  - < 1.5:1 for update-heavy skewed search
- Shared HWDSs are effective
- Policies can avoid performance loss

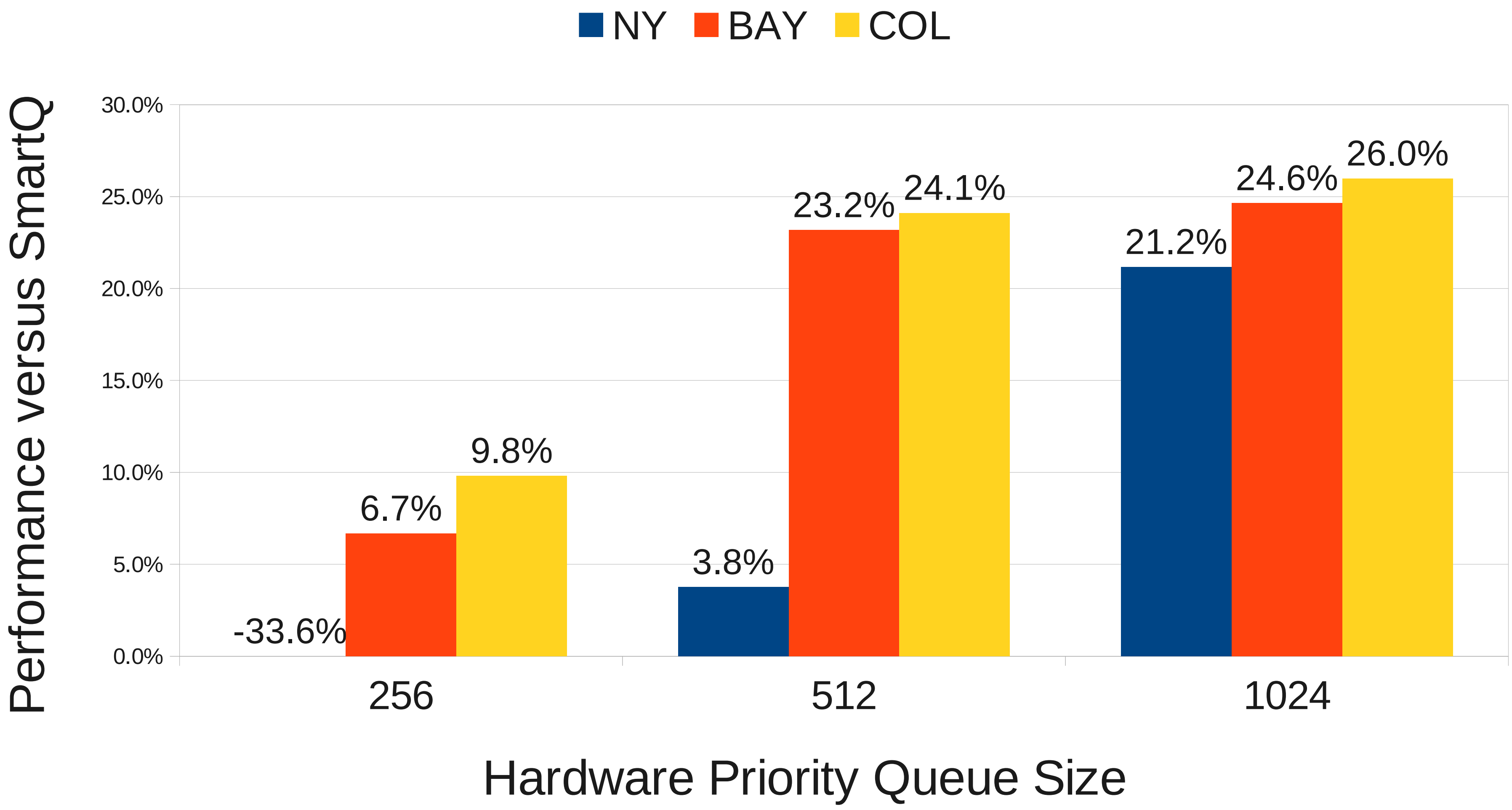
# Real-World Applications: Planning

- Dijkstra's Algorithm, A* search
  - PQ time can be 50% or more of total
  - Benefits from *change-key* operation
- 9th DIMACS challenge: GPS navigation
  - USA road maps

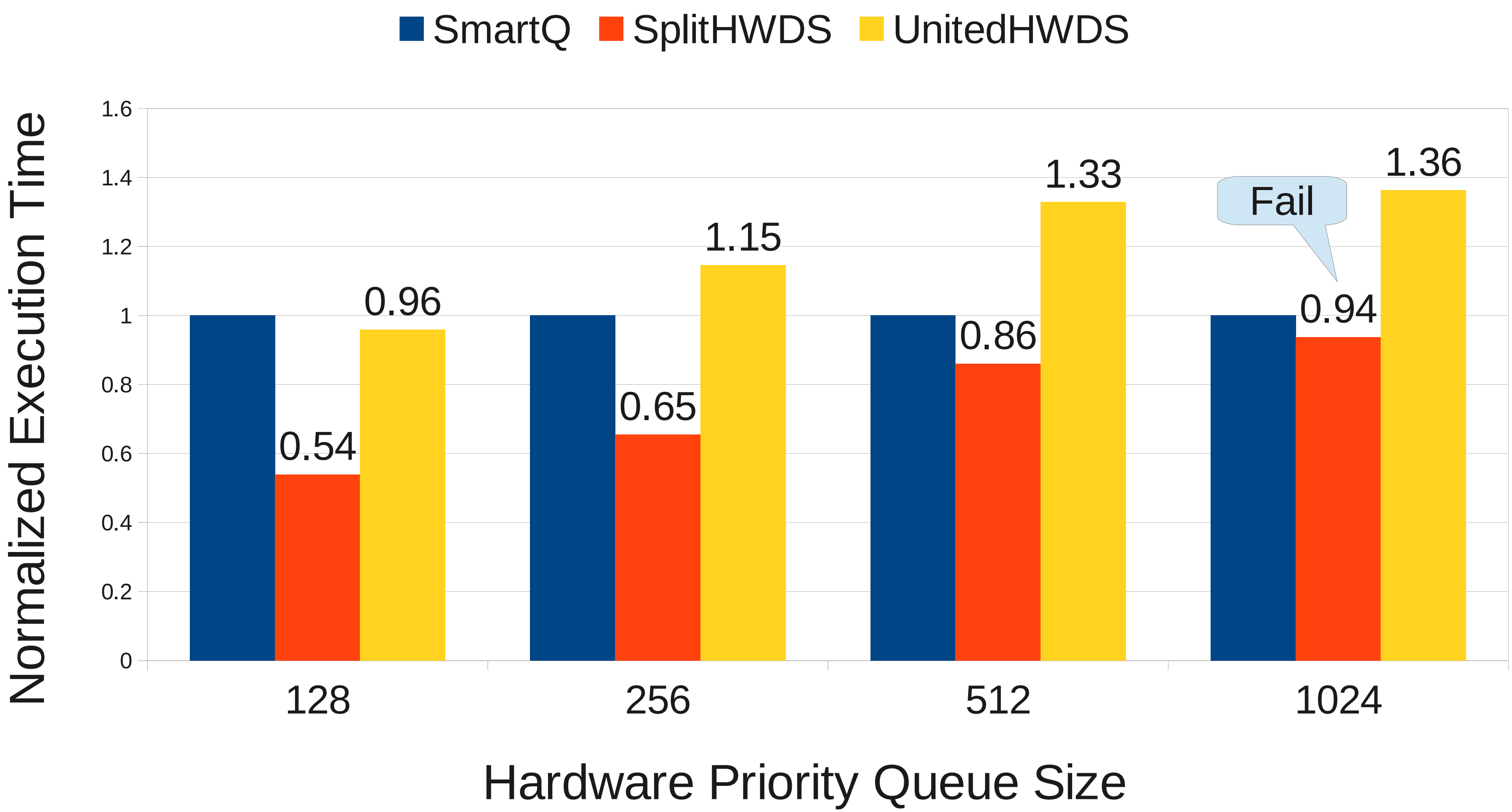
| Input | PQ Size | PQ Operations | PQ time |
|---|---|---|---|
| NY | 925 | 528693 | 28.50% |
| BAY | 886 | 642540 | 27.10% |
| COL | 945 | 871332 | 30.10% |

Behavior of 3 smallest USA inputs

# United HWDS: 5 queries



# Colorado benchmark: 1 query



# GPS Application: Summary

- Performance improves despite overflow
- Prior art does not use HWDS effectively
- This thesis benefits real-world applications

# Conclusion

- HWDS can improve memory workloads
- OS support necessary for applications
- This thesis:
  - Handles overflow better
  - First to support HWDS sharing
  - Demonstrates benefit for real applications
  - Evaluates overheads with cycle precision
  - Opens a new door for future explorations

# Future Work

- Memory access policies like with cache
- HWDS assignment algorithms
- Sharing data in a HWDS
- OS optimizations from HWDS knowledge
- Language and library integration
- Hardware improvements

# Publications


**G. Bloom**, G. Parmer, B. Narahari, and R. Simha, "Shared Hardware Data Structures for Hard Real-Time Systems," 12th International Conference on Embedded Software. EMSOFT 2012. October 2012.

E. Leontie, **G. Bloom**, B. Narahari, and R. Simha, "No Principal Too Small: Memory Access Control for Fine-Grained Protection Domains," 15th Euromicro Conference on Digital System Design. DSD 2012. September 2012.

**G. Bloom**, E. Leontie, B. Narahari, and R. Simha, "Chapter 12 - Hardware and Security: Vulnerabilities and Solutions," in Handbook on Securing Cyber-Physical Critical Infrastructure, Boston: Morgan Kaufmann, 2012, pp. 305–331. ISBN: 978-0-12-415815-3.

E. Leontie, **G. Bloom**, R. Simha, "Automation for Creating and Configuring Security Manifests for Hardware Containers," 4th Symposium on Configuration Analytics and Automation. SafeConfig 2011. October 2011.

**G. Bloom**, G. Parmer, B. Narahari, and R. Simha. "Real-Time Scheduling with Hardware Data Structures," Work in Progress, IEEE Real-Time Systems Symposium, 2010. RTSS 2010. December 2010.

**G. Bloom**, B. Narahari, and R. Simha. "Fab Forensics: Increasing Trust in IC Fabrication," IEEE International Conference on Technologies for Homeland Security, 2010. HST '10. November 2010.

E. Leontie, **G. Bloom**, O. Gelbart, B. Narahari, and R. Simha. "A compiler-hardware technique for protecting against buffer overflow attacks," Journal of Information Assurance and Security, vol. 5, no.1, pp. 1-8, 2010.

E. Leontie, **G. Bloom**, B. Narahari, R. Simha, and J. Zambreno. "Hardware-enforced Fine-grained Isolation of Untrusted Code," Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code. SecuCode '09. November 2009.

**G. Bloom**, B. Narahari, R. Simha, and J. Zambreno. "Providing secure execution environments with a last line of defense against Trojan circuit attacks," Computers & Security, vol. 28, no. 7, pp. 660-669, October 2009.

E. Leontie, **G. Bloom**, B. Narahari, R. Simha, and J. Zambreno. "Hardware Containers for Software Components: A Trusted Platform for COTS-Based Systems," 2009 IEEE/IFIP International Symposium on Trusted Computing and Communications. TRUSTCOM 2009. August 2009.

**G. Bloom**, B. Narahari, and R.Simha. "OS Support for Detecting Trojan Circuit Attacks," 2$^{nd}$ IEEE International Workshop on Hardware-Oriented Security and Trust. HOST 2009. July 2009.

**G. Bloom** and S. Popoveniuc, "Information leakage in mix networks with randomized partial checking," 2009 International Conference on Information Security and Privacy. ISP-09. July 2009.

# Thanks!

“programming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself but where to find it”

— John Backus, 1977 ACM Turing Award Lecture

“Advances in microelectronics have made the realization of “smart” data structures a practical reality.”

— Charles Leiserson, *Systolic Priority Queues*, 1979

Indeed, I believe that virtually *every* important aspect of programming arises somewhere in the context of sorting or searching!

— Don Knuth, *The Art of Computer Programming*