Relative Performance of Hardware and Software-Only Directory Protocols Under Latency Tolerating and Reducing Techniques

Håkan Grahn[1] and Per Stenström[2]

Dept. of Computer Science and Business Administration
University of Karlskrona/Ronneby, Sweden
Email: Hakan.Grahn@ide.hk-r.se
WWW: http://www.ide.hk-r.se/~nesse/

Dept. of Computer Engineering
Chalmers University of Technology, Sweden
Email: pers@ce.chalmers.se
WWW: http://www.ce.chalmers.se/~pers/

Motivation

Background:

- Latency tolerating and reducing techniques are useful for hardware-based directory protocols
- Software-only directory protocols have lower hardware overhead, but also lower performance as a result of protocol handler invocations

Problem:

- Are latency tolerating and reducing techniques successful also for software-only directory protocols, despite the protocol execution overhead

Focus and Contribution

Addressed how three types of latency tolerating techniques impact the performance of software-only directory protocols

We consider techniques that

- increase (prefetching),
- decrease (migratory optimization), and
- do not affect (release consistency)

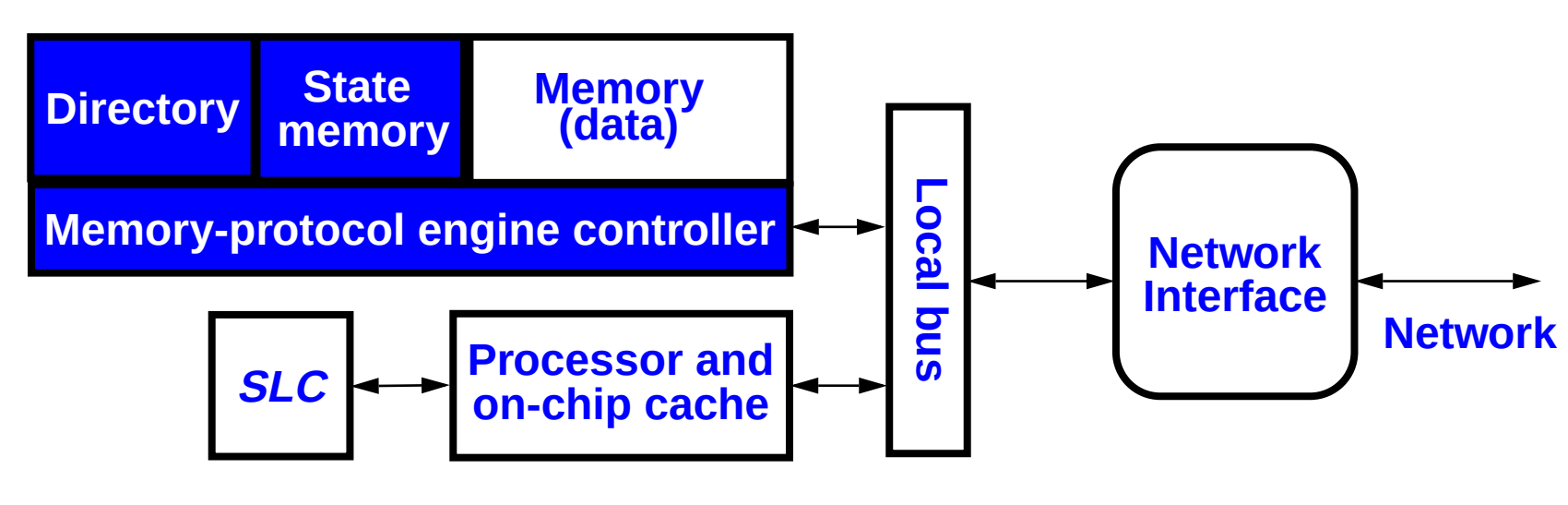
the protocol execution overhead

Evaluated the performance effects of the techniques using architectural simulations of a CC-NUMA multiprocessor model

Outline

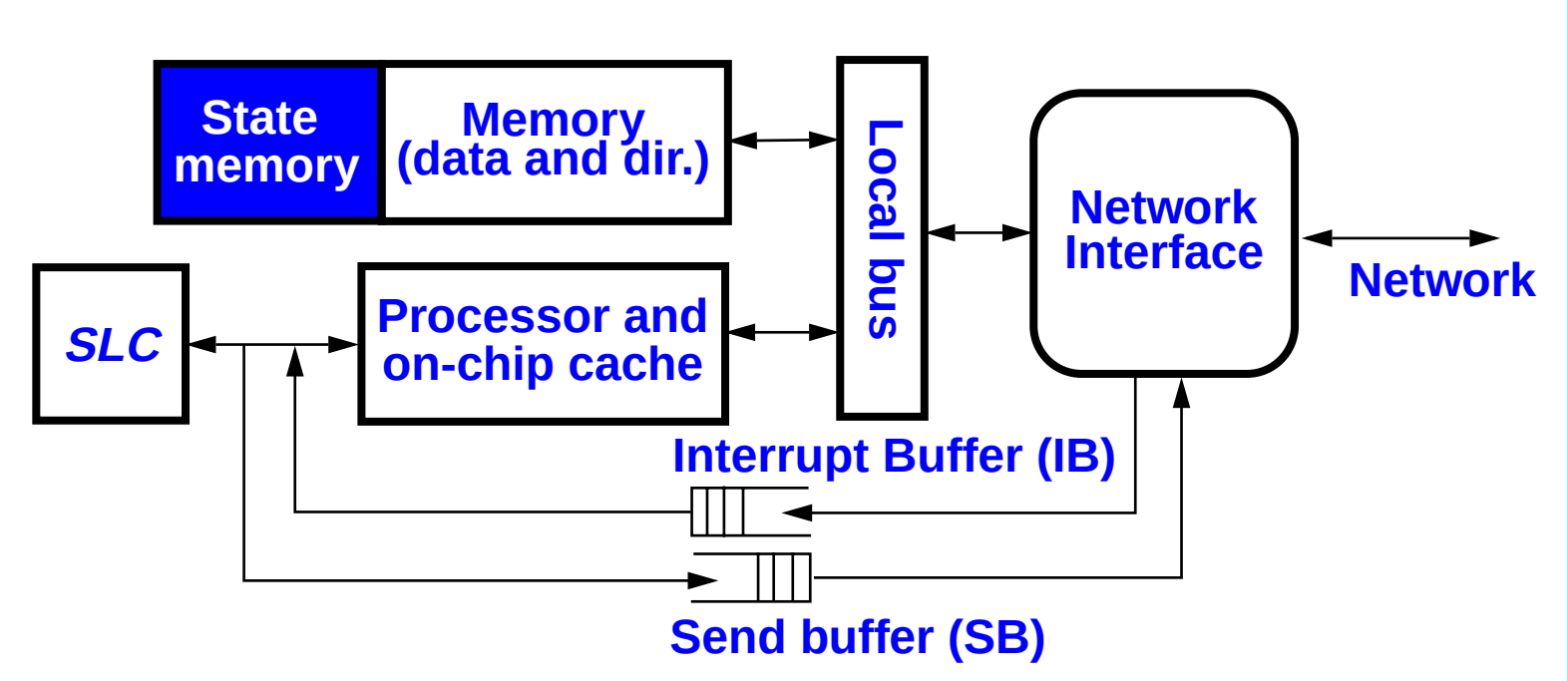
- Baseline hardware-only and software-only directory protocols
- Experimental methodology
- Performance effects of prefetching
- Performance effects of migratory optimization
- Performance effects of release consistency
- Conclusions

Baseline Hardware-Only Protocol



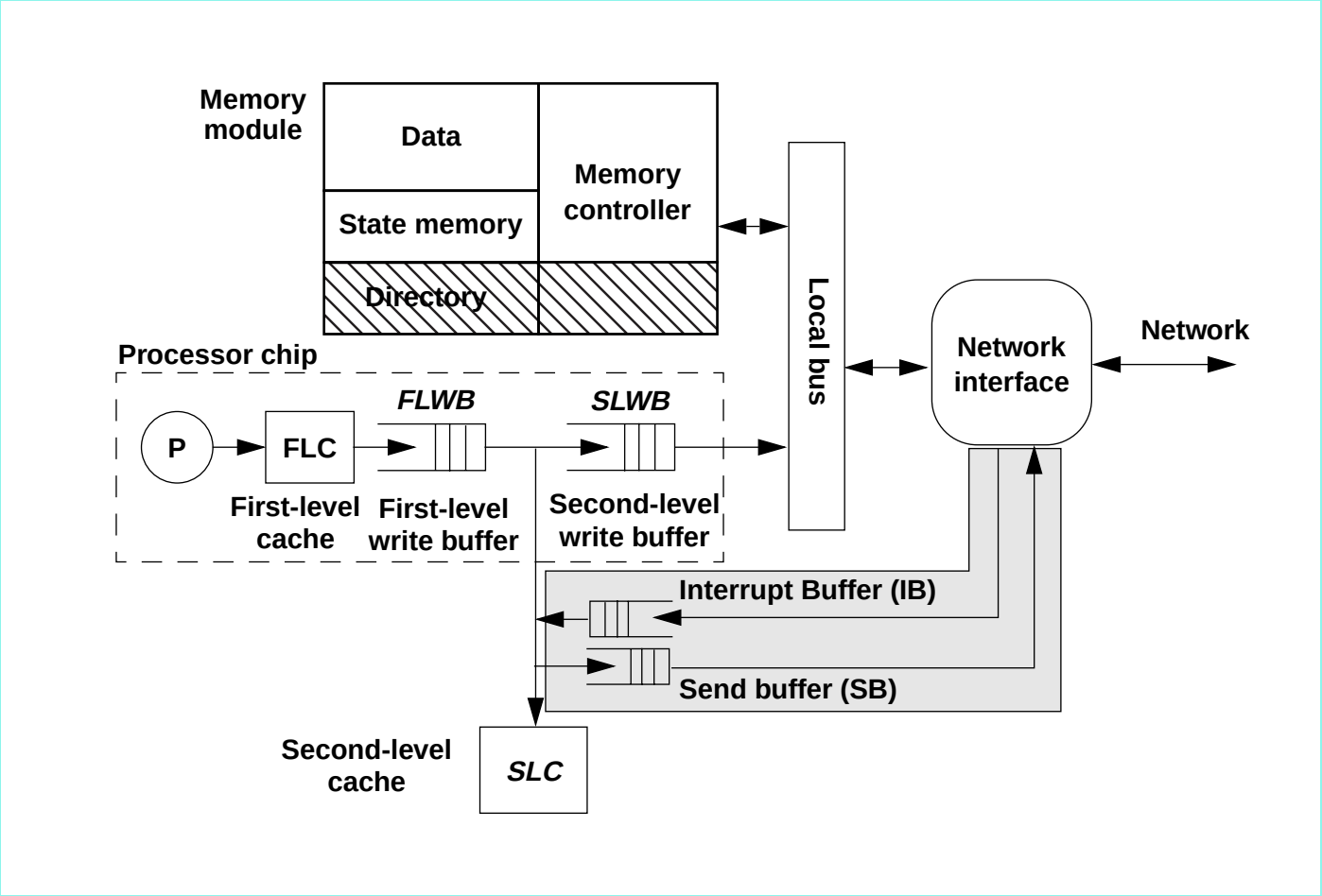
- Cache-coherent NUMA architecture with a write-invalidate, full-map four-hop protocol
- The memory-protocol engine consists of a controller, a directory, and a state memory
- The network interface routes messages but also processes invalidation acknowledgments

Baseline Software-Only Protocol

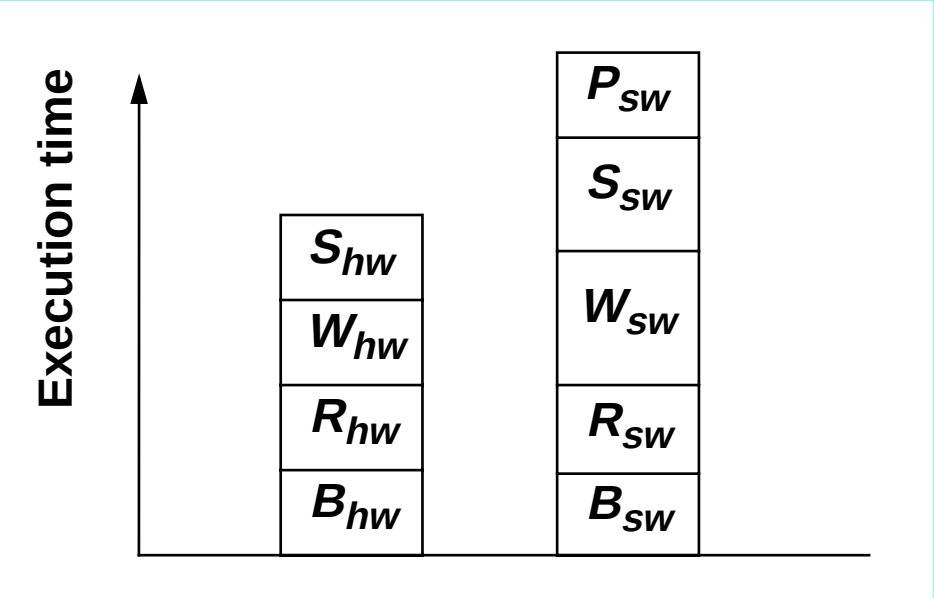


- The memory-protocol engine is emulated by software handlers on the compute processor
- The network interface has same function as in HW but also routes messages to/from IB and SB, and handles some requests to dirty blocks

Organization of a Processor Node



Execution Time Breakdown



- The busy time and read stall time is the same, i.e., $B_{hw} = B_{sw}$ and $R_{hw} = R_{sw}$
- The write and synchronization stall times are longer for SW, i.e., $W_{hw} < W_{sw}$ and $S_{hw} < S_{sw}$
- Protocol execution overhead for SW, i.e., P_{sw}

Experimental Methodology

Detailed architectural simulations of the baseline systems and the enhanced systems

Architectural parameters:

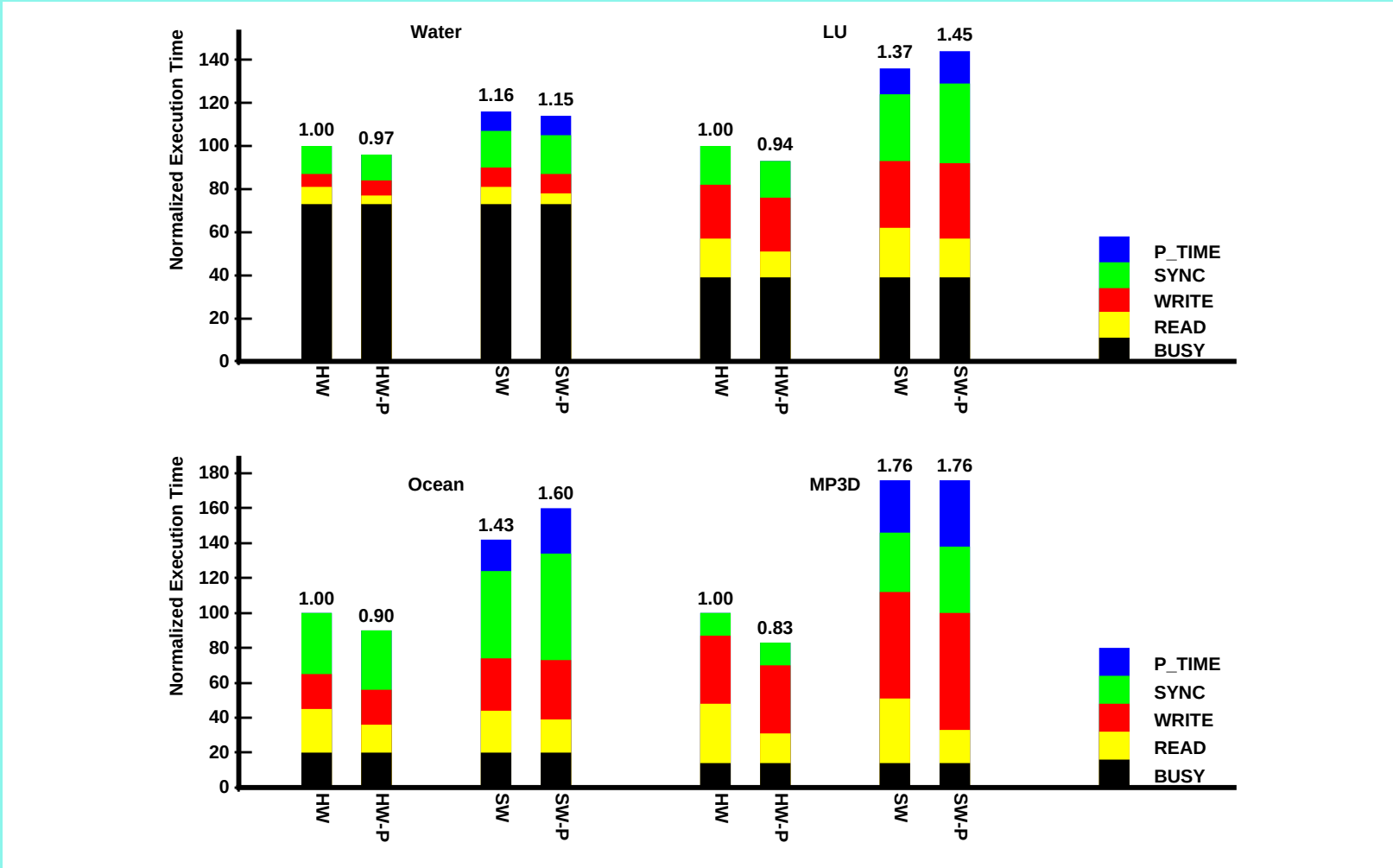
- 16 node system using 100 MHz processors, 30 pclocks network latency
- SW latency: 50 pclocks + DRAM accesses to directory + msg sending (read miss: ~100 pcl)
- 64 Kbytes SLCs using 64 bytes blocks
- Sequential consistency

Applications:

- Water, LU, Ocean, MP3D from the SPLASH suite

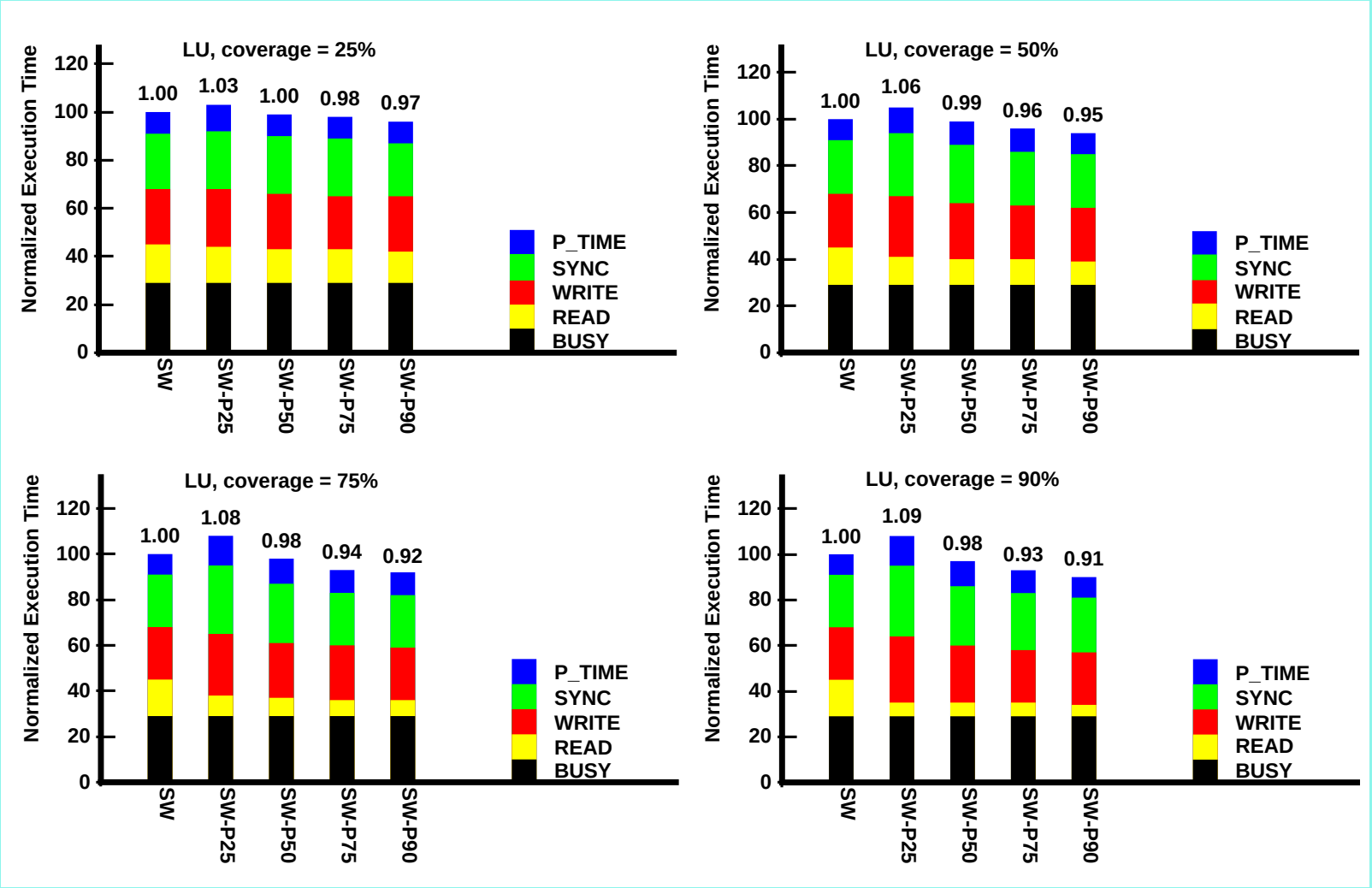
Prefetching

(Coverage = 50%, prefetch efficiency = 25%)



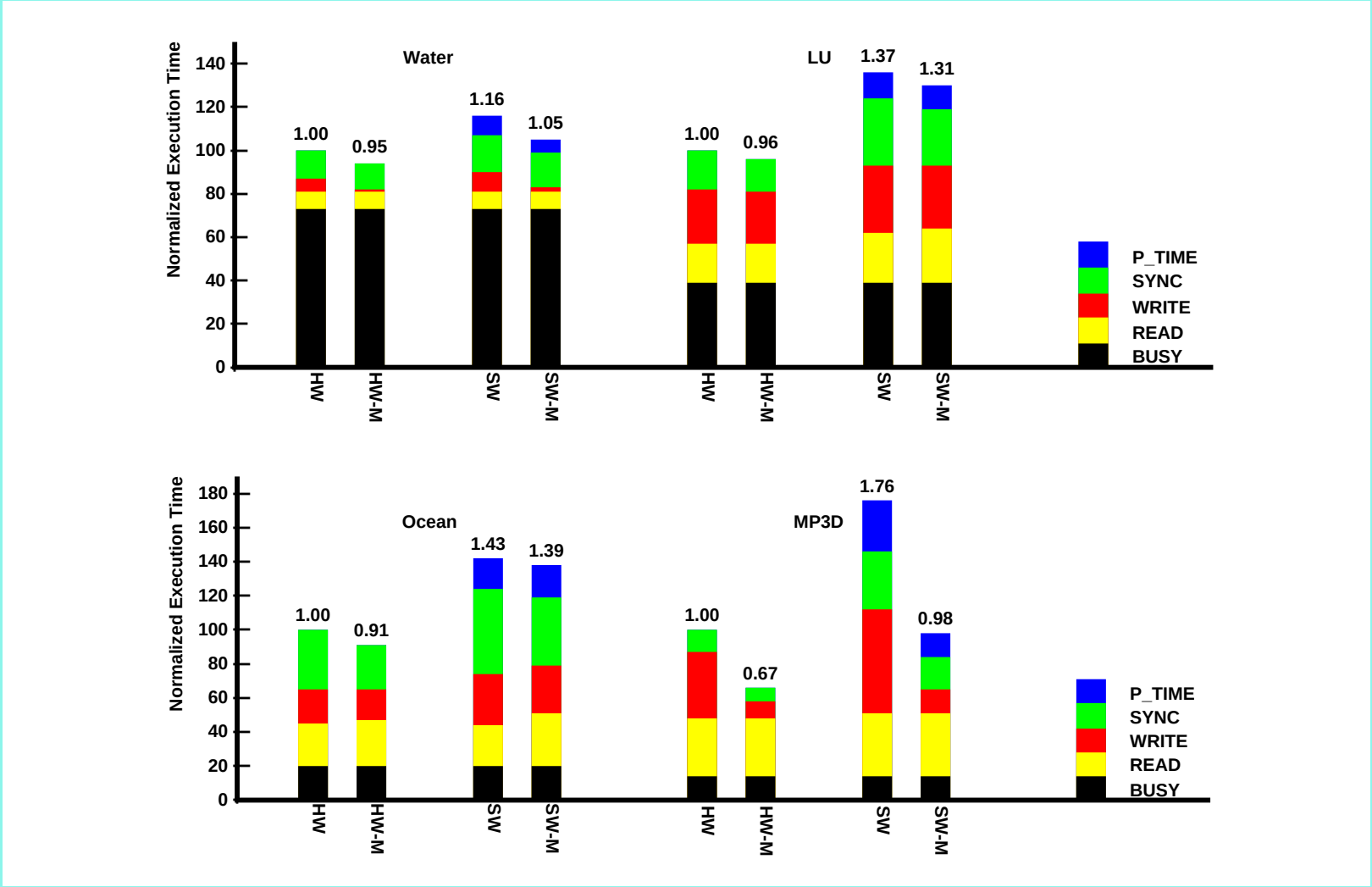
The protocol execution overhead removes the gains from the read stall time reduction for SW protocols

Variation of Efficiency and Coverage



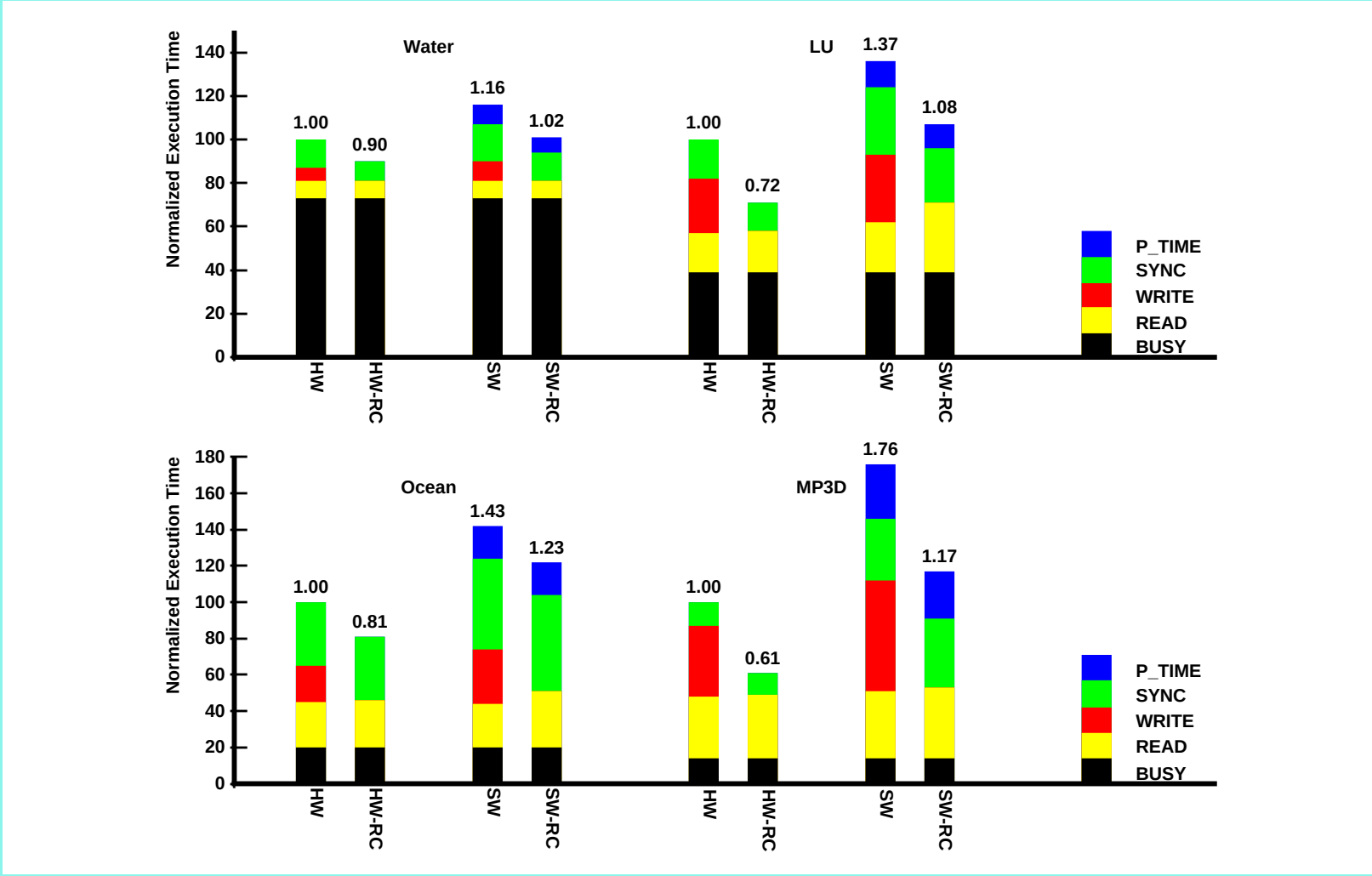
A high prefetch efficiency is very important, especially when the coverage is high

Migratory optimization



Migratory optimization usually reduces the execution time relatively more for SW than for HW

Release Consistency



RC successfully hides all write latency also for SW, but the protocol overhead is relatively larger under RC

Conclusions

Evaluated latency tolerating and reducing techniques for software-only directory protocols

Techniques that

- *increase* the protocol overhead, e.g., prefetching, must be very efficient and used with care
- *decrease* the protocol overhead, e.g., migratory optimization, are relatively more efficient for SW
- *do not affect* the protocol overhead, e.g., RC, are relatively more effective for HW

Latency tolerating and reducing techniques must be chosen with more care for software-only directory protocols than for HW