Performance Evaluation of Two Home-Based LRC Protocols for SVM Systems

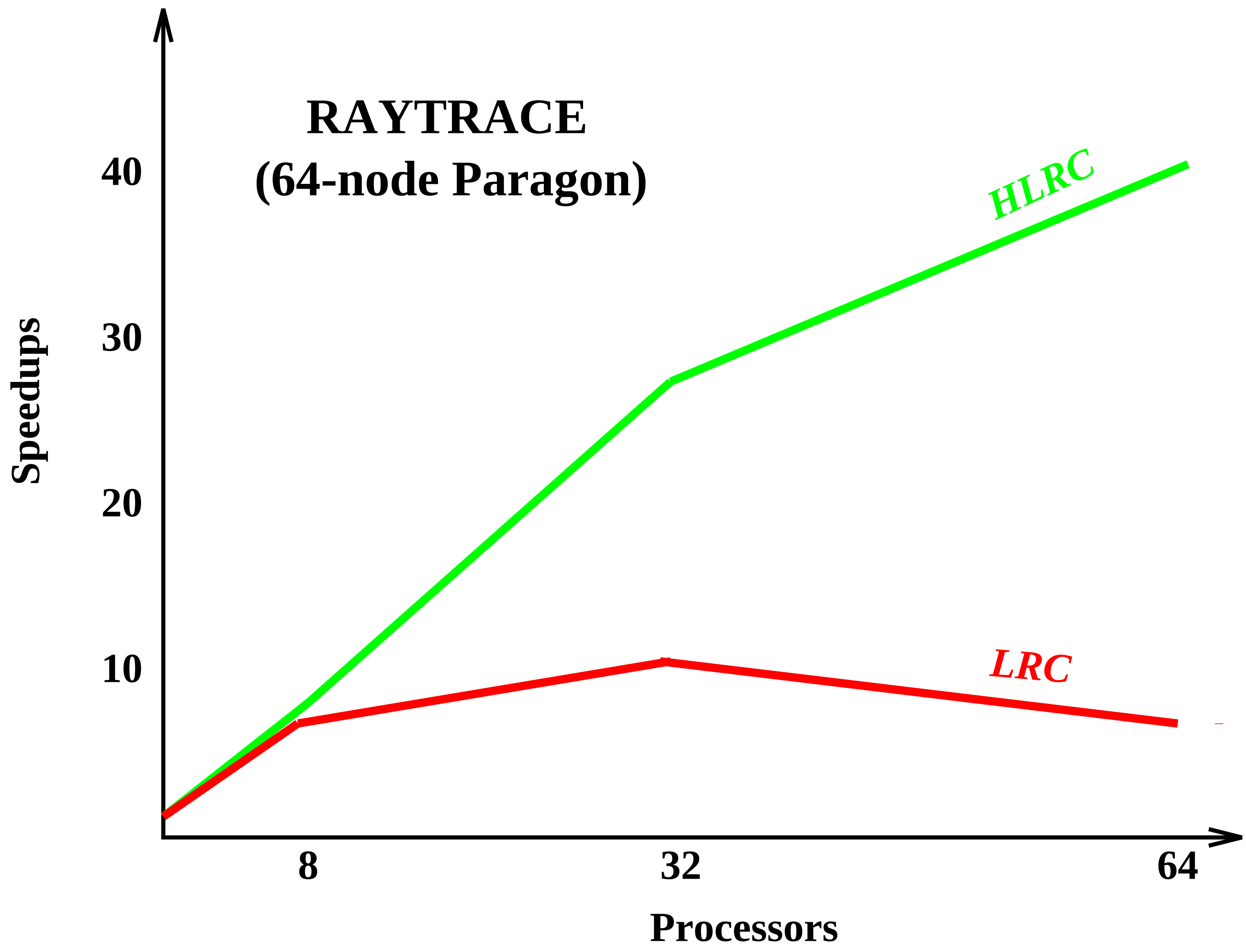
Yuanyuan Zhou, Liviu Iftode and Kai Li

Department of Computer Science
Princeton University

Motivation

- SVM: page-based software shared memory
- Performance limitations
  - False sharing
  - Protocol overhead
- Traditional approach: Lazy Release Consistency (LRC)
  - Release consistent memory
  - Multiple writer, invalidation-based protocol
  - Scalability is an open question
- Our approach: Home-based Lazy Release Consistency (HLRC)
  - One copy of shared page is eagerly updated

HLRC vs. LRC



RAYTRACE
(64-node Paragon)
Speedups
40
30
20
10
HLRC
LRC
8
32
64
Processors

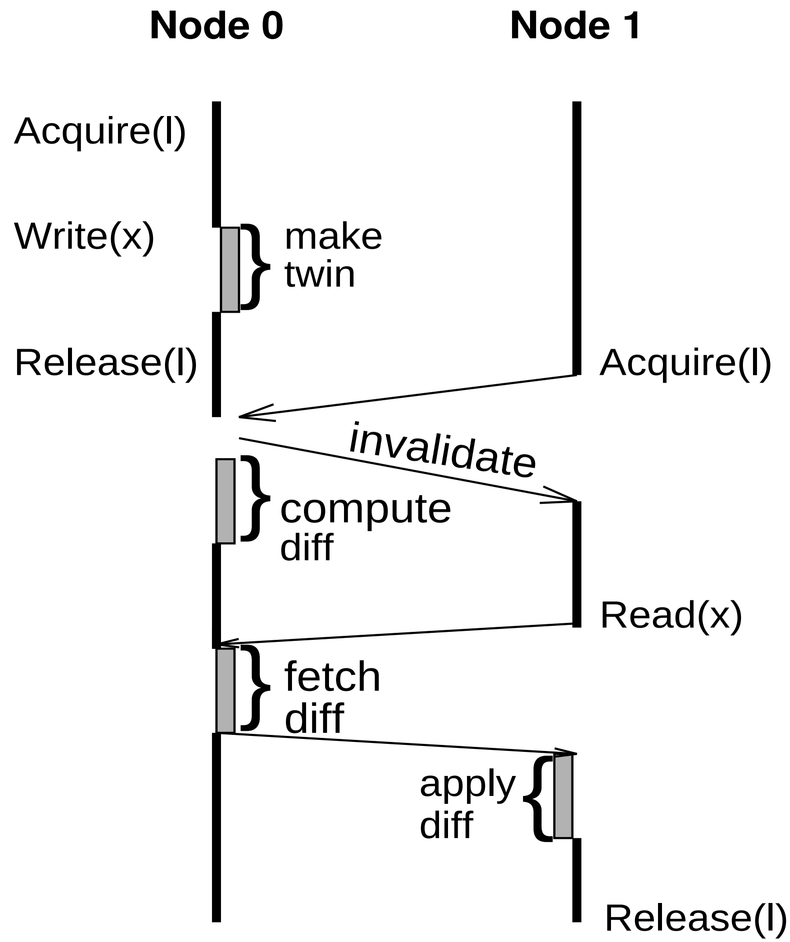
Contributions

- New protocol: Home-based Lazy Release Consistency (HLRC)
- First study on scalability of SVM protocols
- First SVM implementation on 64 nodes

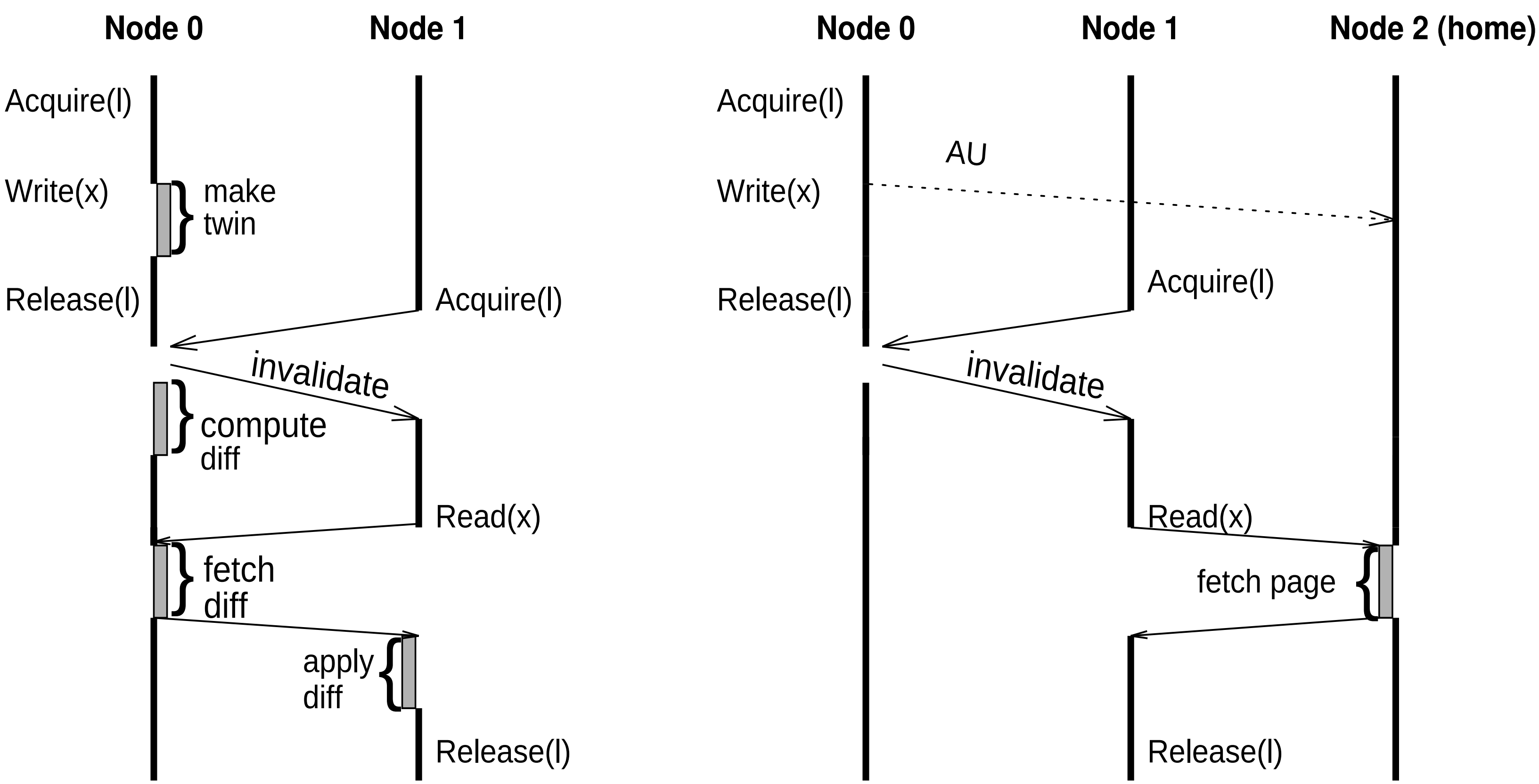
Outline

- Overview of LRC and A   URC
- Home-based LRC
- Prototype Implementation
- Performance
- Conc lusions

Lazy Release Consistency (*Keleher , ISCA '92*)

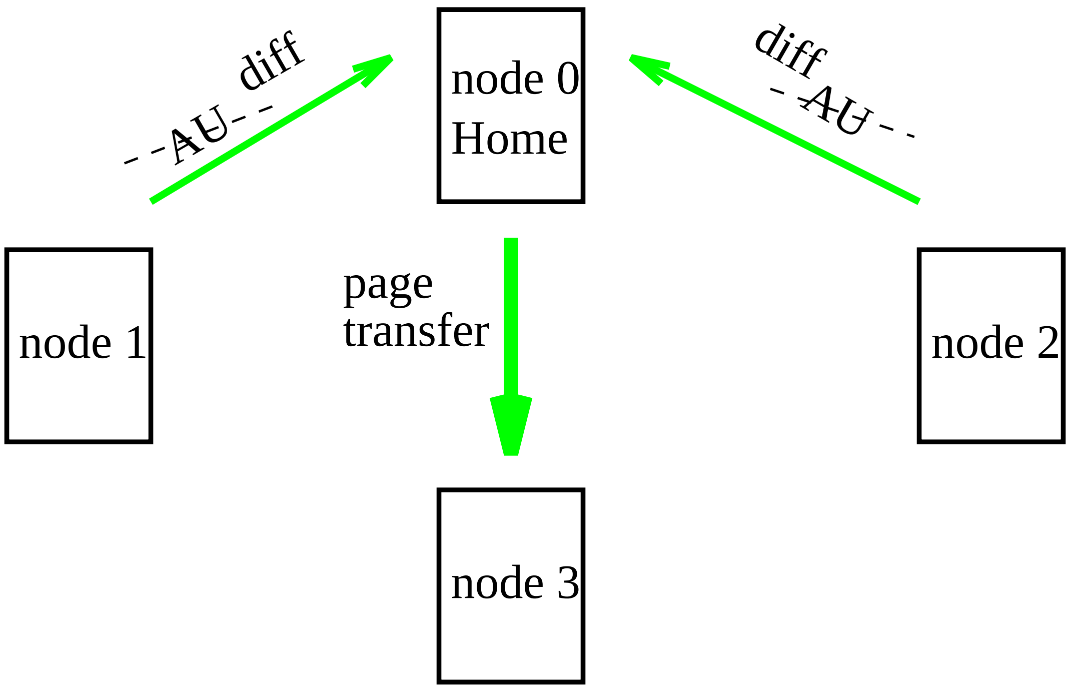


Automatic Update Release Consistency
(*Iftode, HPCA '96*)



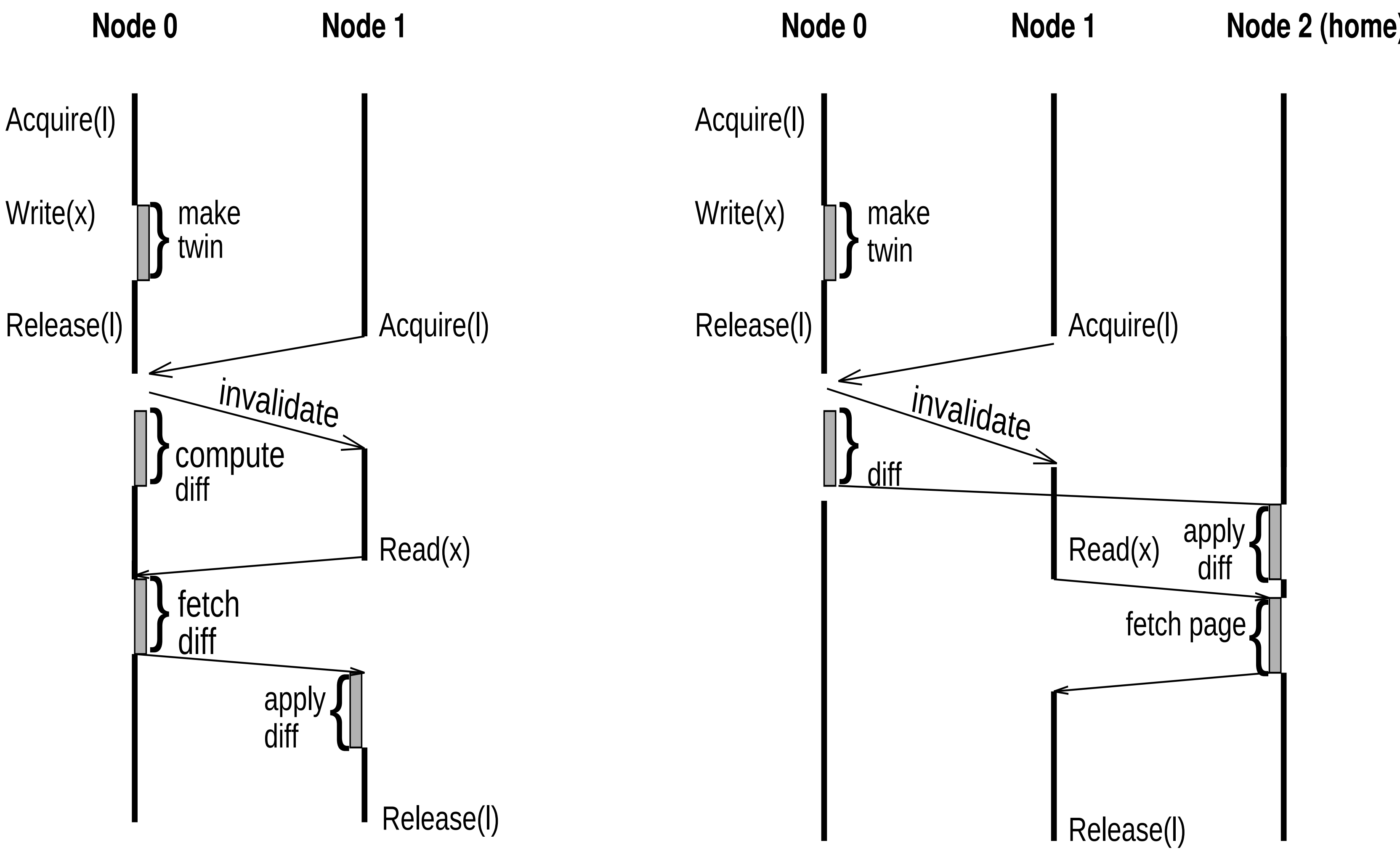
requires hardware support for automatic update (AU)

Home-Based Approach

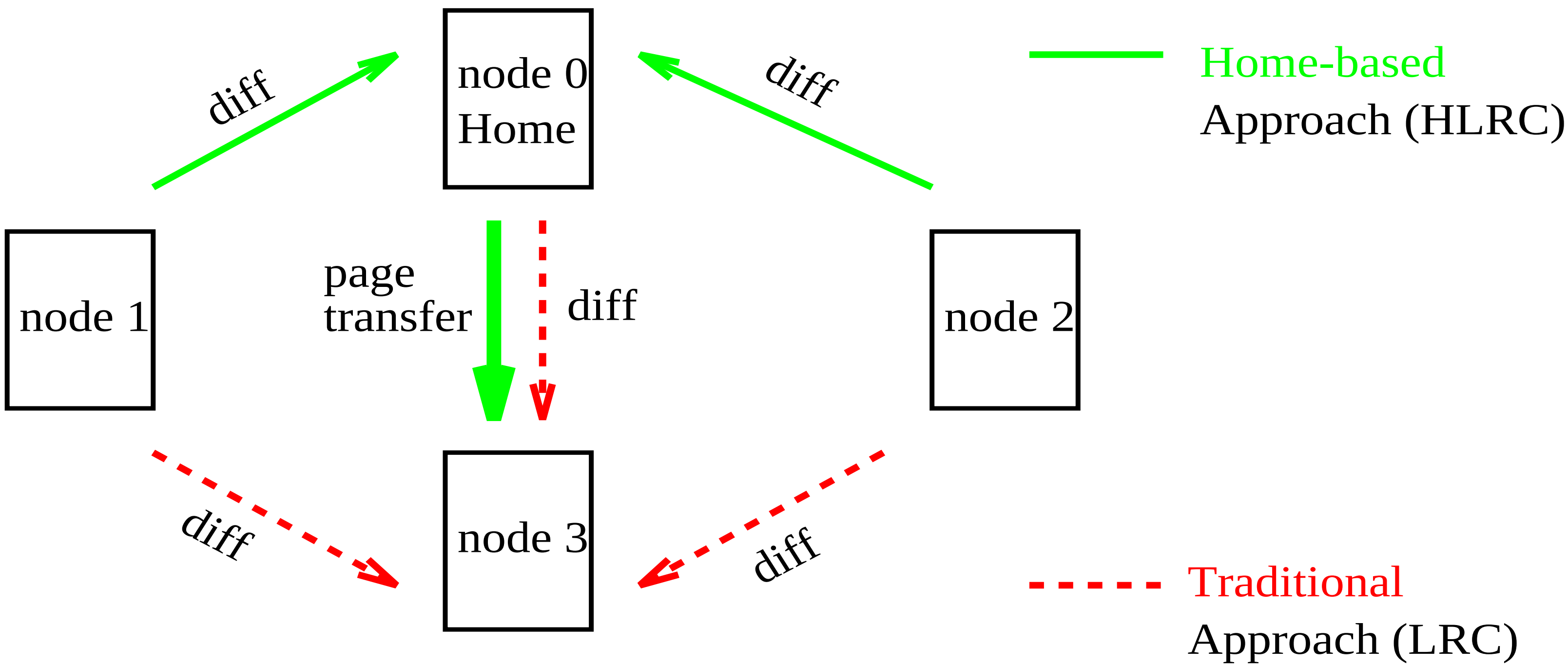


- each page has a home
- updates sent eagerly to home
- whole page fetched on demand from home
- hardware support is not necessary

Home-Based Lazy Release Consistency



Advantages of Home-Based Approach



- each diff is applied exactly once ➠ small overhead
- no diff gathering and ordering ➠ simple protocol
- diffs have short lifetimes ➠ low memory overhead
- fewer messages compared to LRC ➠ $O(p)$ vs $O(p^2)$

HLRC and LRC Prototype Implementations

- 64-node P aragon
- one compute processor and one communication co-processor (50 MHz i860) per node
- 400 MB/sec memory bus
- 200 MB/sec/link bandwidth
- micro-kernel based OSF/1 with 8 KB page size
- NX/2 message passing library
- protocol data is garbage collected at barrier time (LRC)

Basic Operations

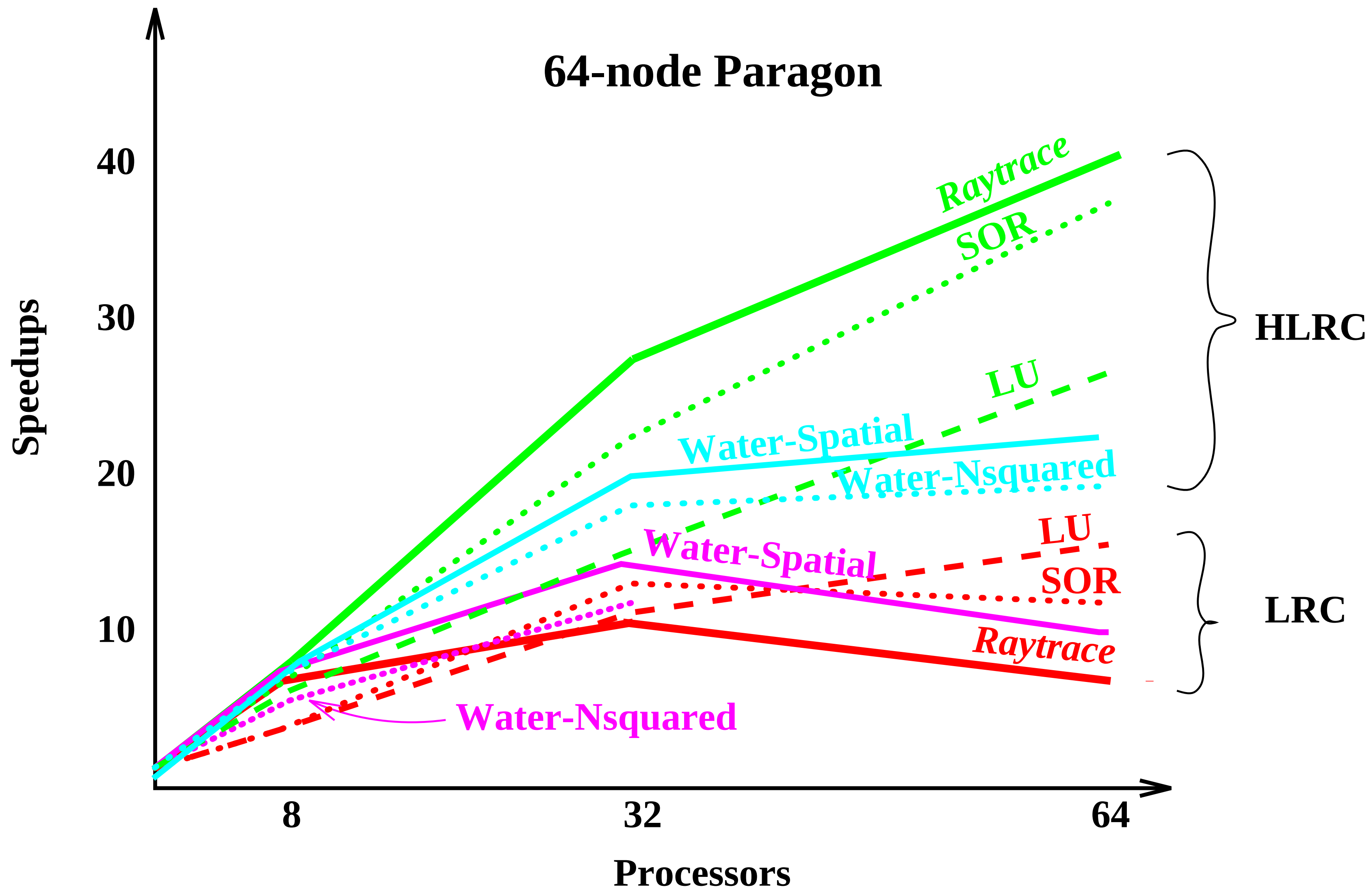
Operation	Time (μsec)
Message Latency	50
Page Transfer	92
Twin Copy	120
Diff Creation	380-560
Diff Application	0-430
Page Fault Exception	290
Receive Interrupt	690
Page Invalidation	200
Page Protection Change	50

page fault in HLRC = 290 + 50 + 690 + 92 + 50 = 1,172 μsec

Applications

Application	Source	Problem Size	Sequential Execution Time (sec)
LU	Splash-2	2048x2048	1,028
Raytrace	Splash-2	balls4.env	956
SOR	TreadMarks	1024*4096	1,360
Water-N^2	Splash-2	4096 molecules	1,130
Water-Spatial	Splash-2	4096 molecules	1,080

Speedups

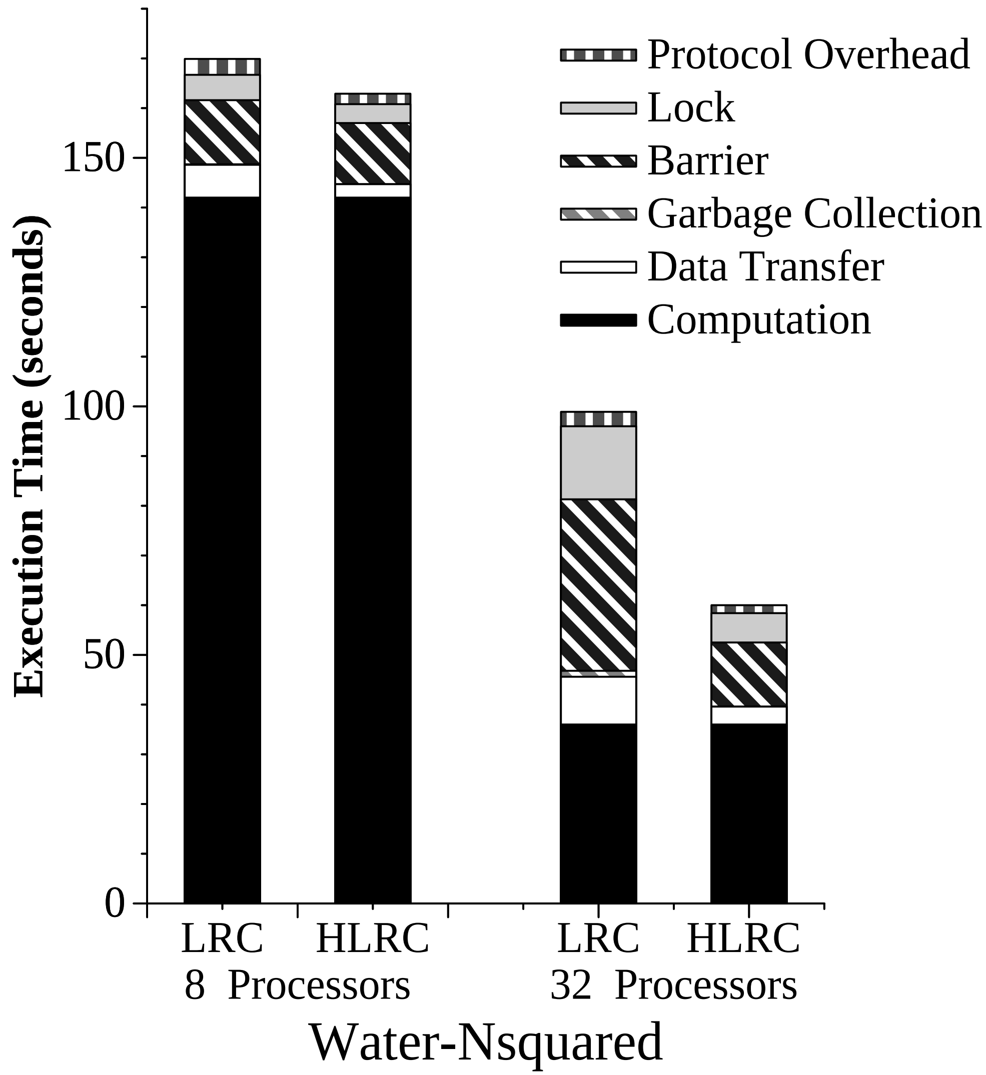

64-node Paragon
Speedups
40
30
20
10
8
32
64
Processors
Raytrace
SOR
LU
Water-Spatial
Water-Nsquared
HLRC
LU
SOR
Water-Spatial
Raytrace
LRC
Water-Nsquared

Frequency of Basic Operations

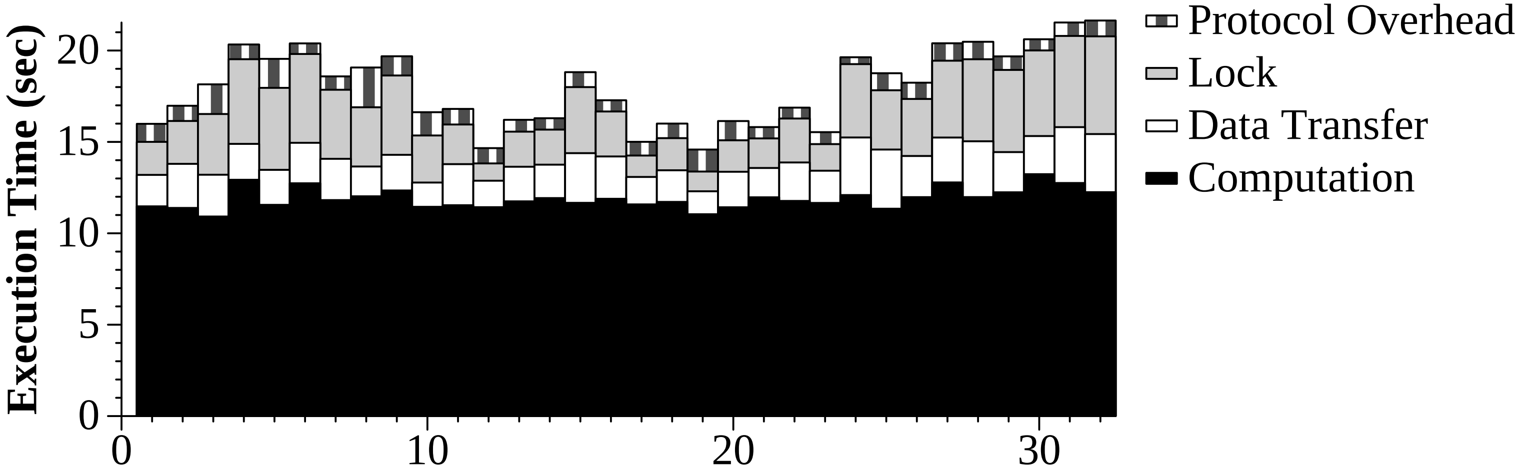
Application (64 processors)	Page Faults		Diffs Created		Diffs Applied	
	LRC	HLRC	LRC	HLRC	LRC	HLRC
LU	452	452	5,704	0	0	0
Raytrace	108	87	83	103	2,368	103
SOR	385	385	3,136	0	342	0
Water-N^2 (32 p)	893	859	956	815	8,742	815
Water-Spatial	918	878	122	66	3,948	66

- fewer page faults
- fewer diffs created (except Raytrace)
- substantially fewer diffs applied

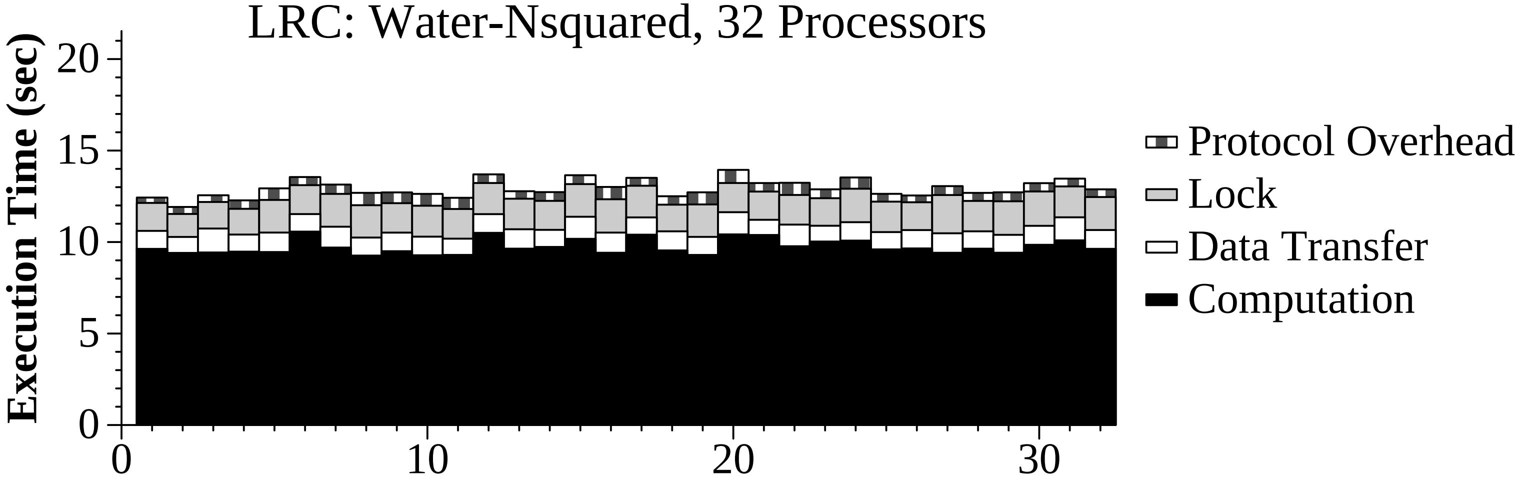
Execution T   ime Breakdown



Per-Processor T ime Breakdown

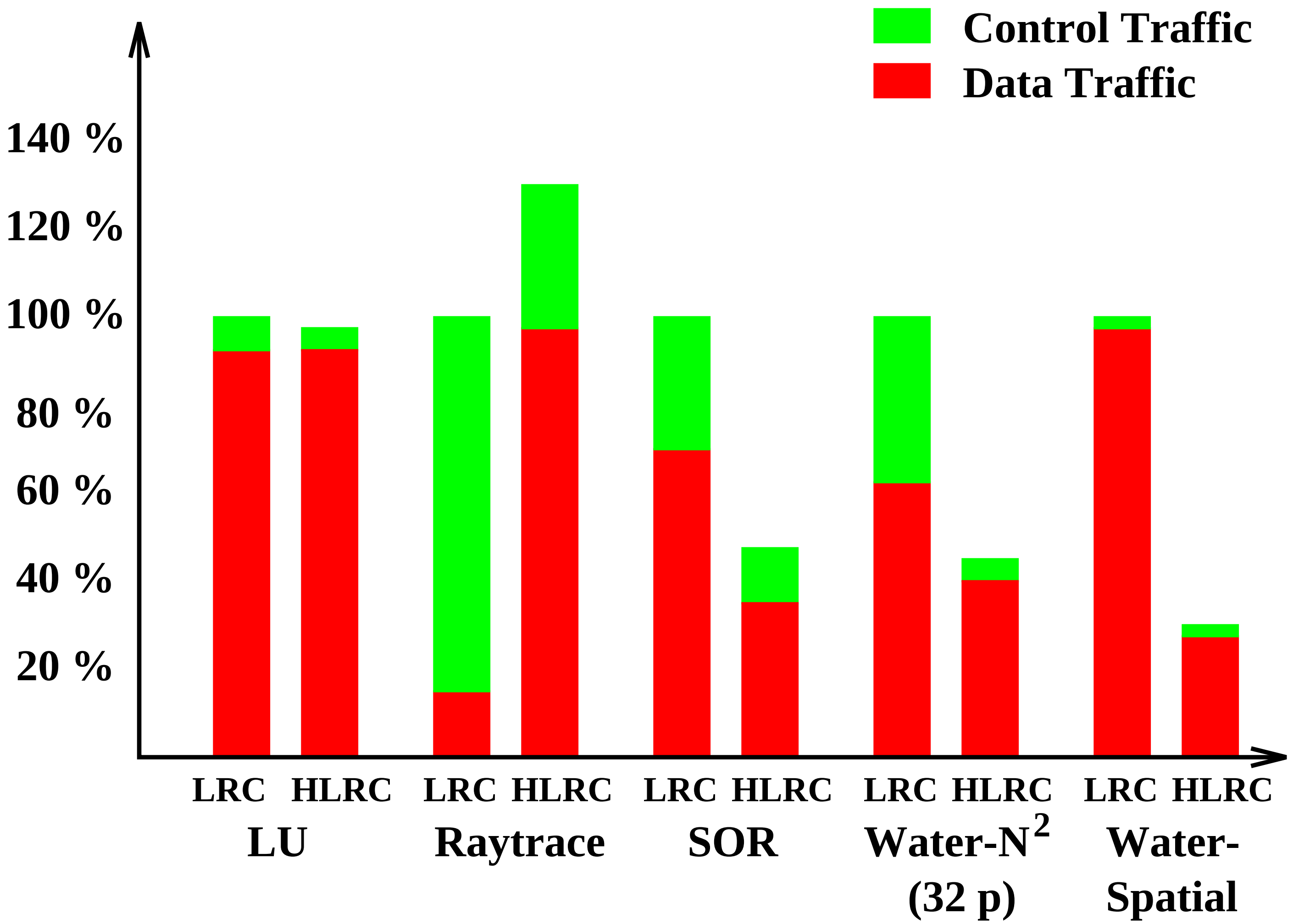


LRC: Water-Nsquared, 32 Processors



HLRC: Water-Nsquared, 32 Processors

Communication T raffic

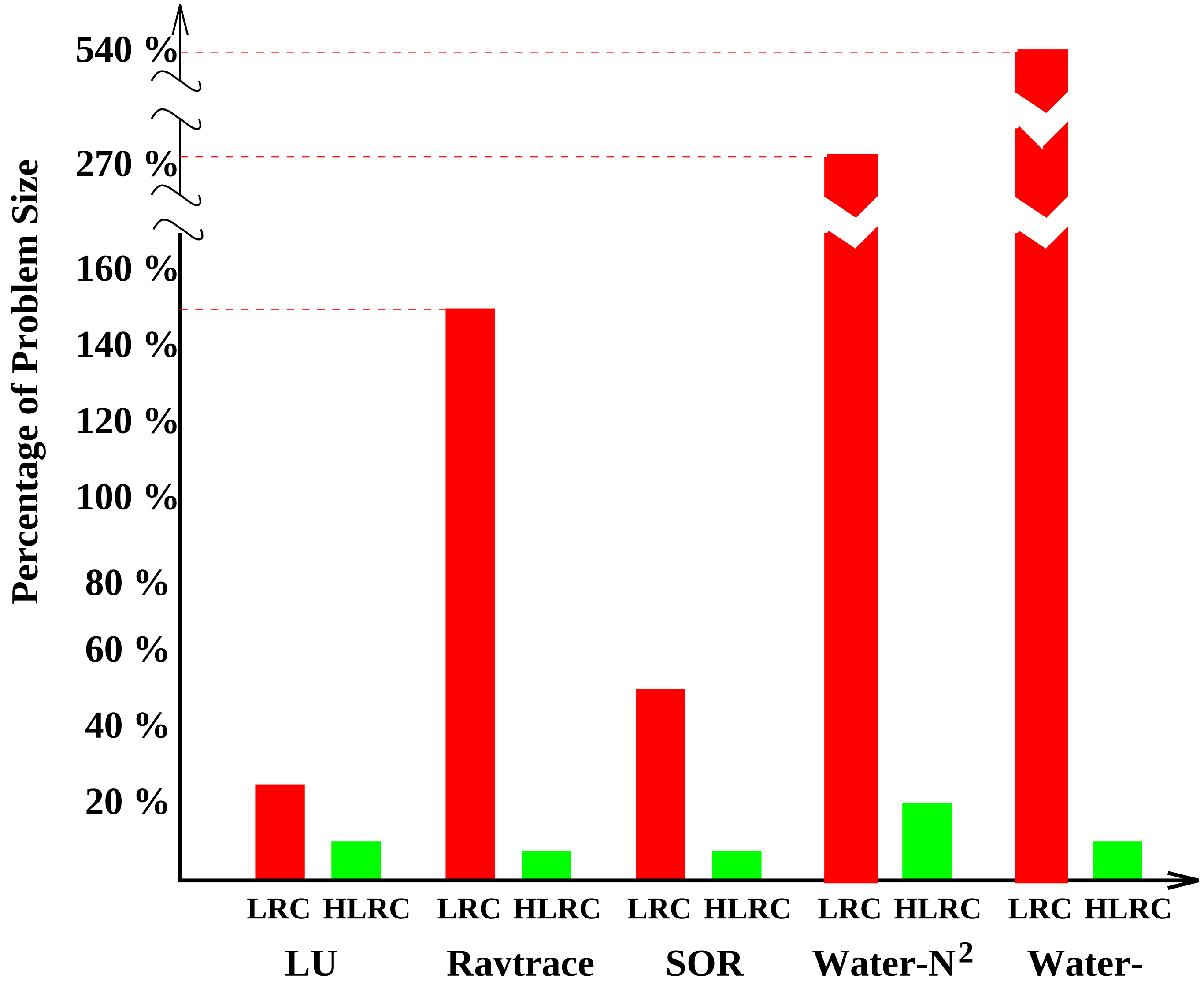


64 Processors

Number of Messages

Application (64 processors)	Data		Control	
	LRC	HLRC	LRC	HLRC
LU	452	452	1,178	1,178
Raytrace	2,368	190	403	313
SOR	385	385	581	581
Water-N^2 (32 p)	1,421	1,674	3,093	2,493
Water- Spatial	2,581	944	2,628	925

Protocol Memory Overhead



Memory Overhead Breakdown

Application (64 processors)	Twins		Diffs		Write Notices	
	LRC	HLRC	LRC	HLRC	LRC	HLRC
LU	2.5	0.0	2.7	0	1.3	0.2
SOR	2.3	0.0	5.2	0	0.9	0.01
Water-N^2 (32 p)	3.9	0.1	4.2	0	5.4	0.15
Water-Spatial	0.4	0.1	12.0	0	0.4	0.01
Raytrace	0.7	0.1	13	0	1.3	0.1

- twins and diffs require more memory in LRC and need to be garbage collected
- memory overhead for write notices is muc h larger in LRC and increases with the number of nodes

Conclusions

- HLRC outperforms LRC by more than a factor of 2 on a 64-node P aragon
- Home-based LRC scales better than LRC due to:
  - ➢ less data traffic
  - ➢ lower memory overhead
  - ➢ simpler protocol
  - ➢ no garbage collection
- Protocol overlapping using the communication coprocessor provides modest improvement
- Plan to release HLRC softw are