The Effect of Network Total Order, Broadcast, and Remote-Write on Network-Based Shared Memory Computing

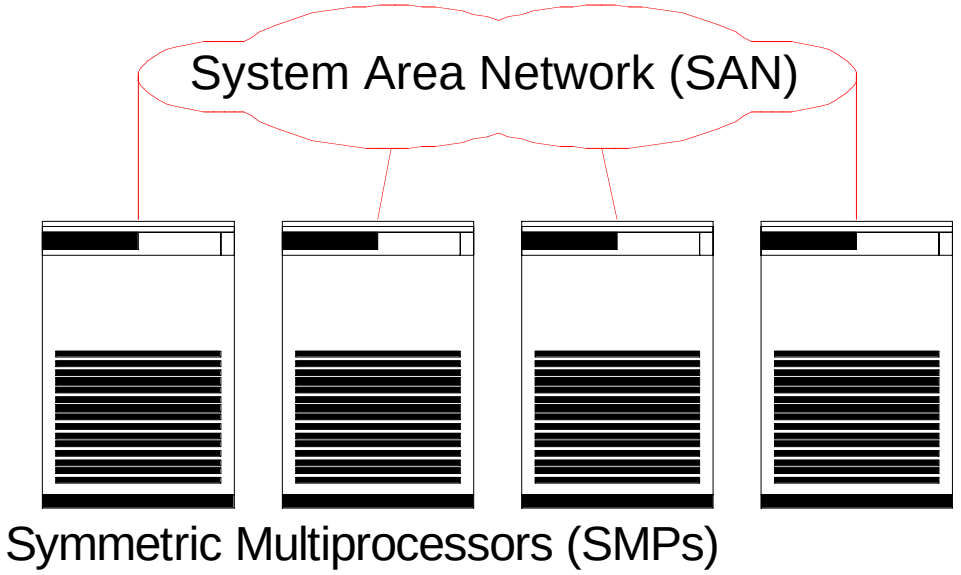
Robert Stets, Sandhya Dwarkadas,
Leonidas Kontothanassis,
Umit Rencuzogullari, and Michael L. Scott

University of Rochester and Compaq Research

HPCA, 2000

High Performance Cluster Computing

- Cost-effective parallel computing platform



- Software Distributed Shared Memory (SDSM) system provides attractive shared memory paradigm

- *In SDSM, what is the performance impact of the SAN?*

Cashmere SDSM

- Designed to leverage special network features:
  remote write, broadcast, total message order

- Use network features to reduce receiver and ack overhead
  – provides an 18-44% improvement in three of ten apps
  Benefits are outweighed by protocol optimizations!

- Use broadcast to reduce data propagation overhead
  – provides small improvement on eight node cluster
  Provides up to 51% on emulated 32-node cluster!

Cashmere: Network Features

I. Introduction

II. Cashmere design (use of network features)

III. Performance of full network features

IV. Performance of Adaptive Data Broadcast

V. Conclusions

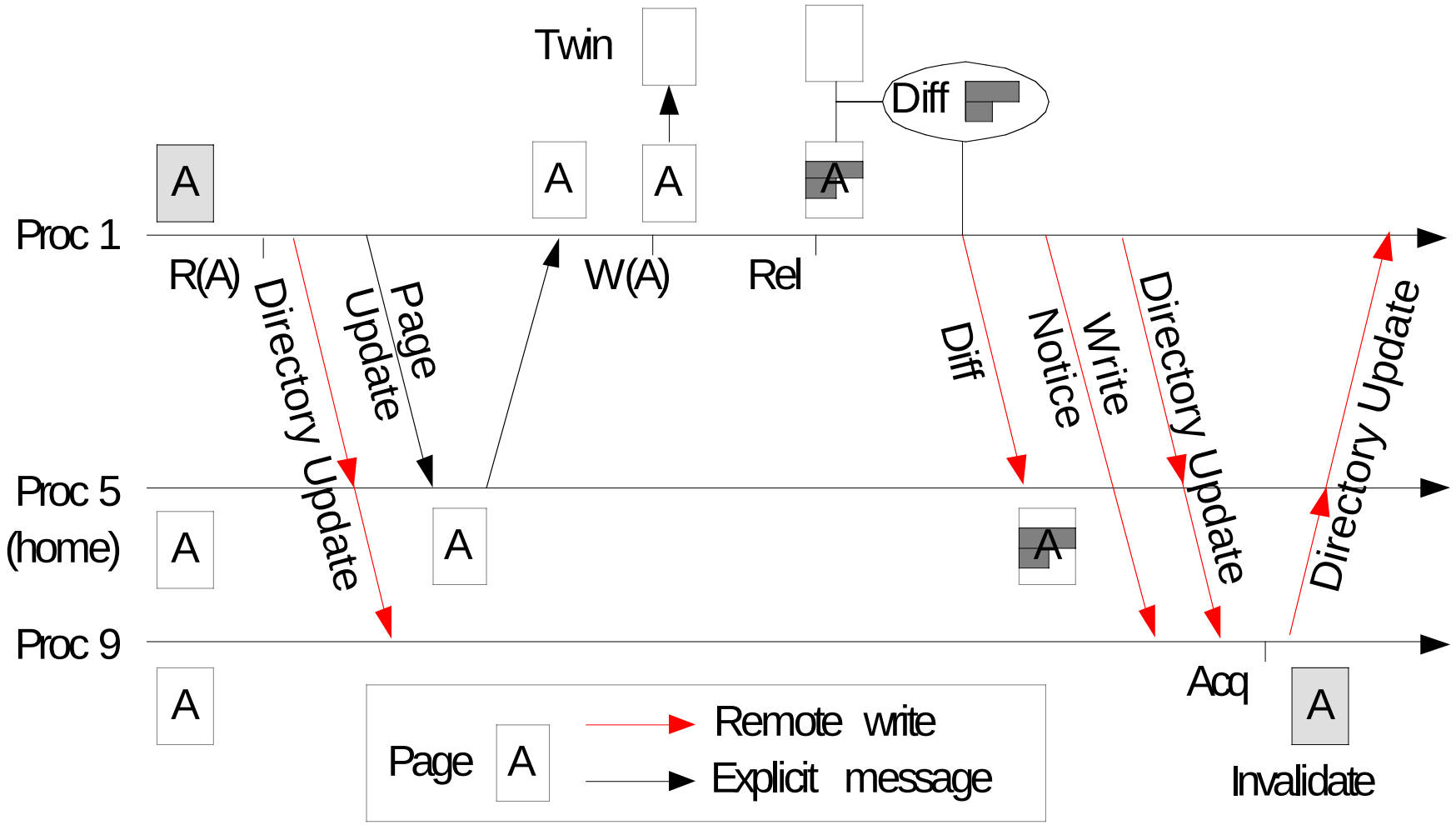
II. Cashmere Design

- Implements a Lazy Release Consistency model
- Applications must be free of data races and must use Cashmere synchronization ops (Acquire, Release)

- Traps data accesses through Virtual Memory hardware
- Uses invalidation messages (write notices)
- Employs home nodes and global page directory

- Prototype: AlphaServer SMPs, Memory Channel SAN

Use of Network Features in Cashmere



Cashmere Design Variants

- "Memory Channel" Cashmere
  - shared data propagation (diffs)*
  - meta-data propagation (directory, write notices)
  - synchronization mechanisms (locks, barriers)

  **Remotely-accessible shared data space is limited!*

- "Explicit messages" Cashmere
  - diffs, write notices, locks, barriers: use plain messages
  - directory: maintain master entry only at the home
  - hide ack latency by pipelining diffs, write notices

Home Migration Optimization

- Reduce twin/diff overhead by *migrating* home node to the active writers
  - migration is not possible when using remote write to propagate data
- Send migration request as part of each write fault
  - home will grant migration request if it is not actively writing the page
  - old node will forward any incoming requests to new home
- *Migration very effectively reduces twin/diff operations!*

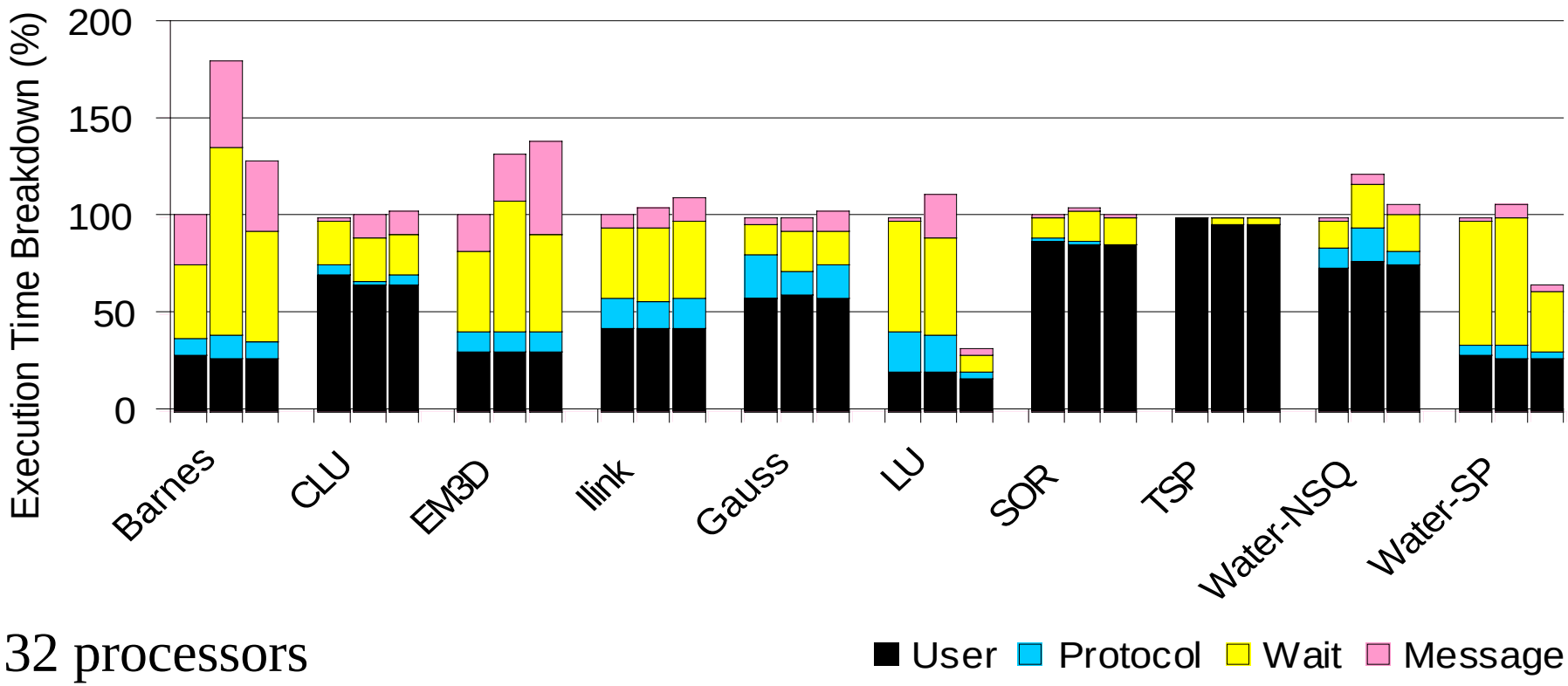
III: Performance of Full Features

- Platform: AlphaServer 4100 cluster: 32 procs, 8 nodes
  - Alpha 21164A 600 MHz
  - Memory Channel II SAN
- Microbenchmarks:
  Round-trip null message latency: 15 μsecs

	MC features (μsecs)	Explicit messages (μsecs)
Diff	31-129	70-245
Lock Acquire	10	33
Barrier (32 procs)	29	53

Results for Full Network Features

Left, middle, right bars:
MC Features, Explicit Msgs, Explicit Msgs Migration



32 processors

IV. Adaptive Broadcast (ADB)

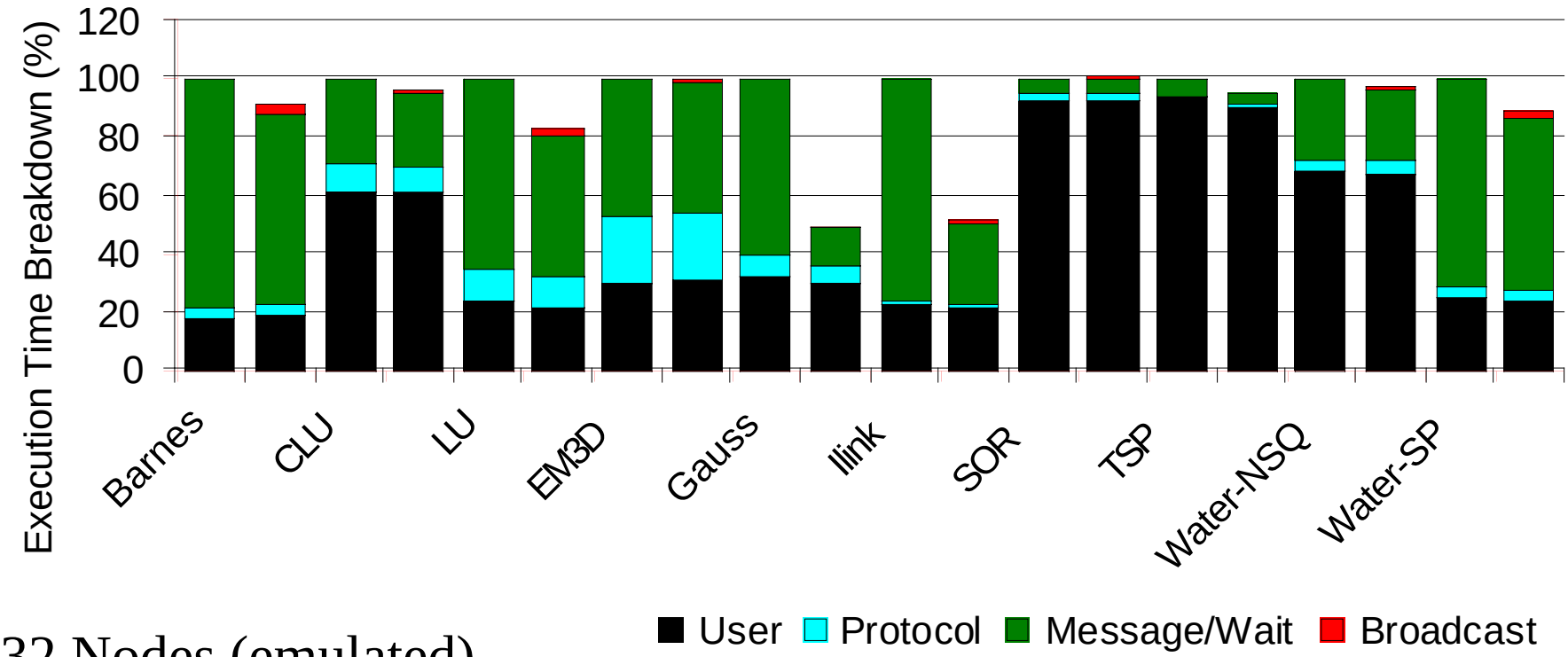
- Use broadcast to reduce contention for widely-shared data

- Augment Cashmere to include a set of broadcast buffers (avoid mapping data into remotely-accessible memory)
- Identify widely shared data
  - multiple requests for same page in the same interval
  - two or more requests for same page in the last interval

- Little performance improvement (<13%) at eight nodes.
- *Does ADB have a larger impact beyond eight nodes?*

Results for Cashmere-ADB

Left, right bars: Cashmere, Cashmere-ADB



32 Nodes (emulated)

V. Conclusions

- Special network features provide some benefit, but can limit scalability
  - three of ten applications improve by 18-44%
- Home node migration can largely recover benefits of the network features
  - can even lead to as much as 67% improvement
- In larger clusters, a scalable broadcast mechanism may be worthwhile
  - three of ten applications improve by 18-51%