

Is S-DSM Dead?

Michael L. Scott

Department of Computer Science
University of Rochester

Keynote Address
2nd Workshop on
Software-Distributed Shared Memory

In conjunction with ICS 2000
Santa Fe, NM, May 2000

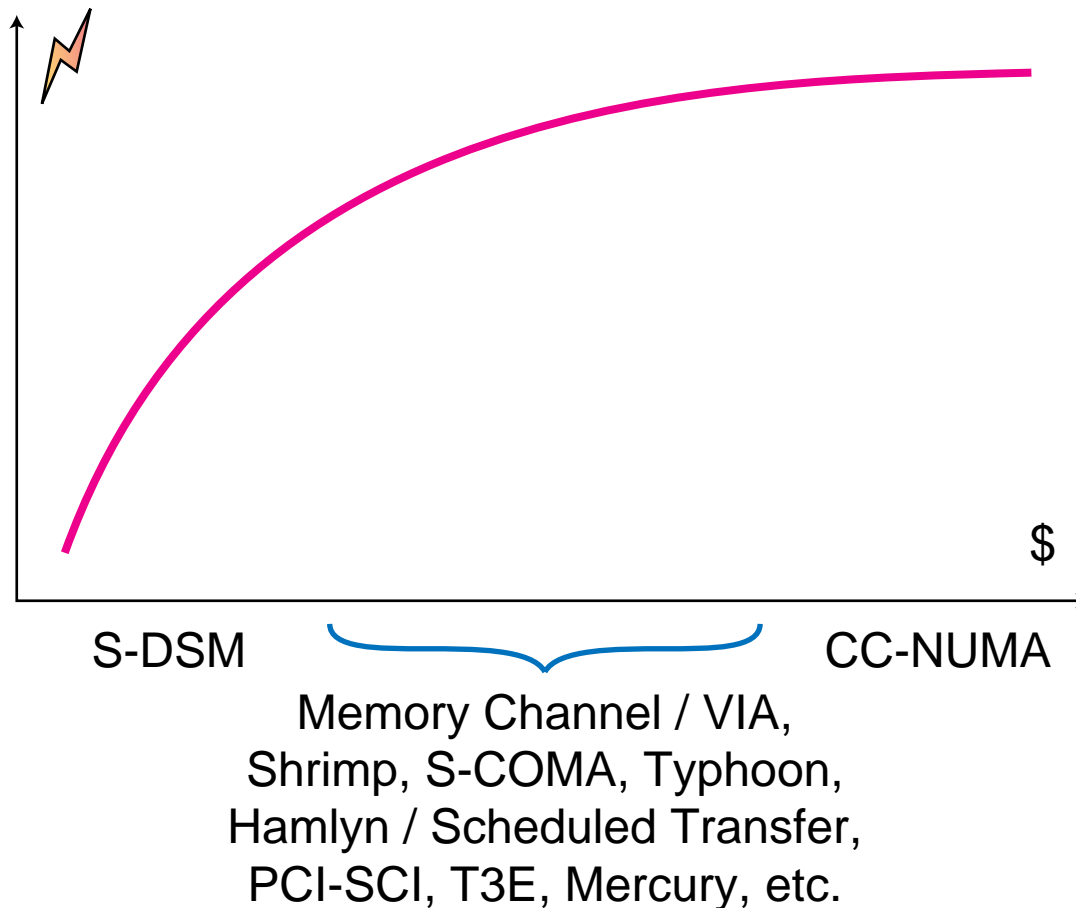
The Company We Keep?

- David Cheriton, SOSP 1999:
 - » Capabilities are dead; no
 - » S-DSM is dead; no
 - » Active networks are dead; no
 - » IPv6 is dead
- MLS:
 - » Interconnection network topology
 - » Byzantine agreement
 - » Load balancing and scheduling

Why Are We Doing This?

- Shared memory an attractive programming model
 - » Familiar
 - » Arguably simpler, esp. for non-performance-critical code
- Hardware coherence is faster, but software
 - » Is cheaper
 - » Can be built faster (sooner to market, faster processors)
 - » Can use more complex protocols
 - » Is easier to tune/enhance/customize
 - » Is the only option on distributed machines

The Price-Performance Curve



S-DSM may maximize “bang for the buck” for shared-memory parallel computing

Is S-DSM Dead?

- Yes

- » The key ideas seem to have been discovered; the rate of innovation is way down
- » Application writers are still choosing MPI
- » Program committees aren't looking for papers

- No

- » Speedups for well-written apps are very good
- » Wide-spread use awaits production-quality systems
- » The ideas are valuable in a wider domain; hence InterWeave (more later)

Outline

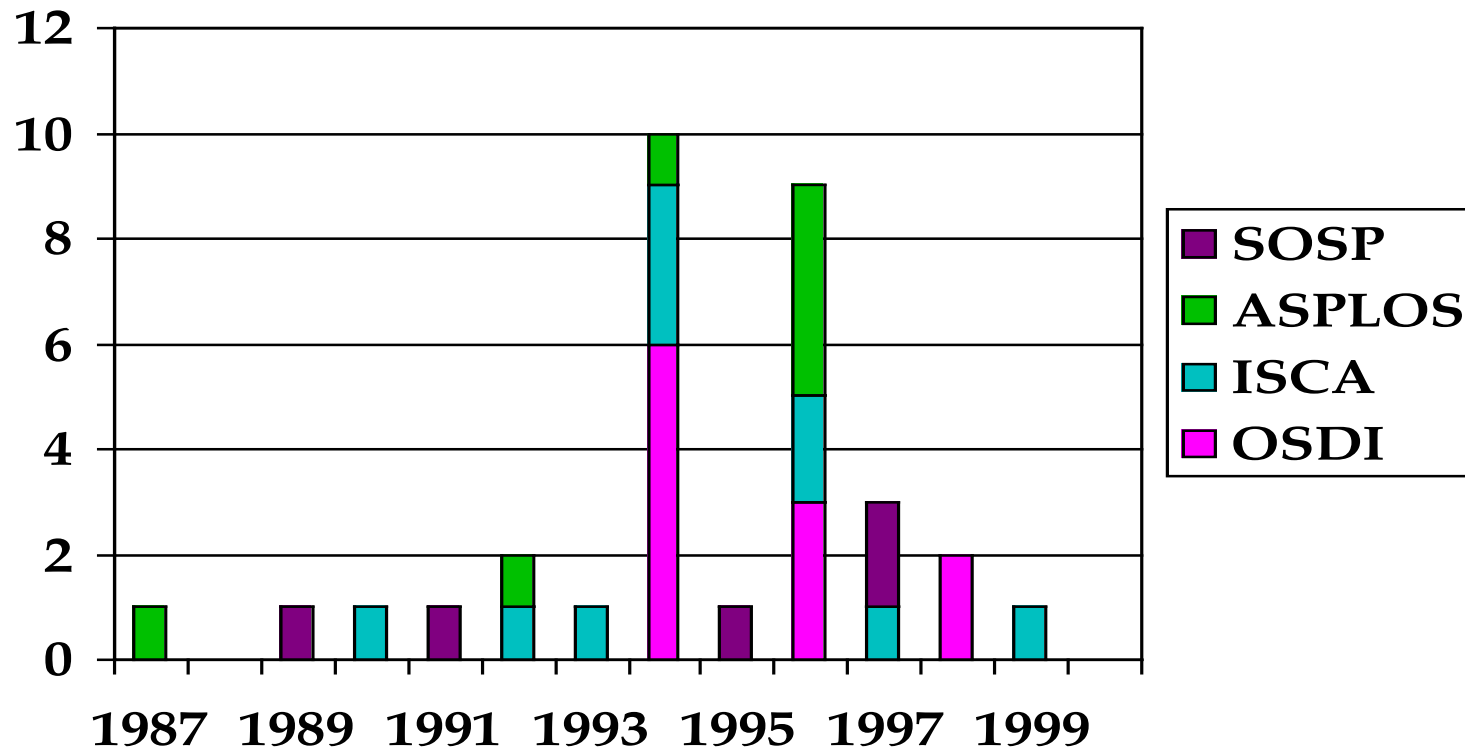
- Historical perspective
- Some thoughts about current status
- What we don't need
- What we do need
- A project that's doing some of it (joint work with Sandhya Dwarkadas and students)

Caveat: I'm trying to stir things up a bit; please don't take offense if I oversimplify, overstate things, or ignore your favorite project!

A Brief History of the Field

- » 1986 Ivy
 - » 1989 Shiva
 - » 1990 Munin
 - » 1992 LRC (TreadMarks)
 - » 1993 Sh. Regions, Midway
 - » 1994 AURC (Shrimp)
 - » 1995 CRL
 - » 1996 Shasta
 - » 1997 Cashmere
 - » 1998 HLRC
- The original idea (Kai Li)
- Relaxed memory model, optimized protocols
- Software-only protocols
- Leverage special HW (User-level messages, multiprocessor nodes)
-

Papers at Leading Conferences



Anybody see a trend?

Why Haven't We Conquered the World?

Been at this for 14 years, but:

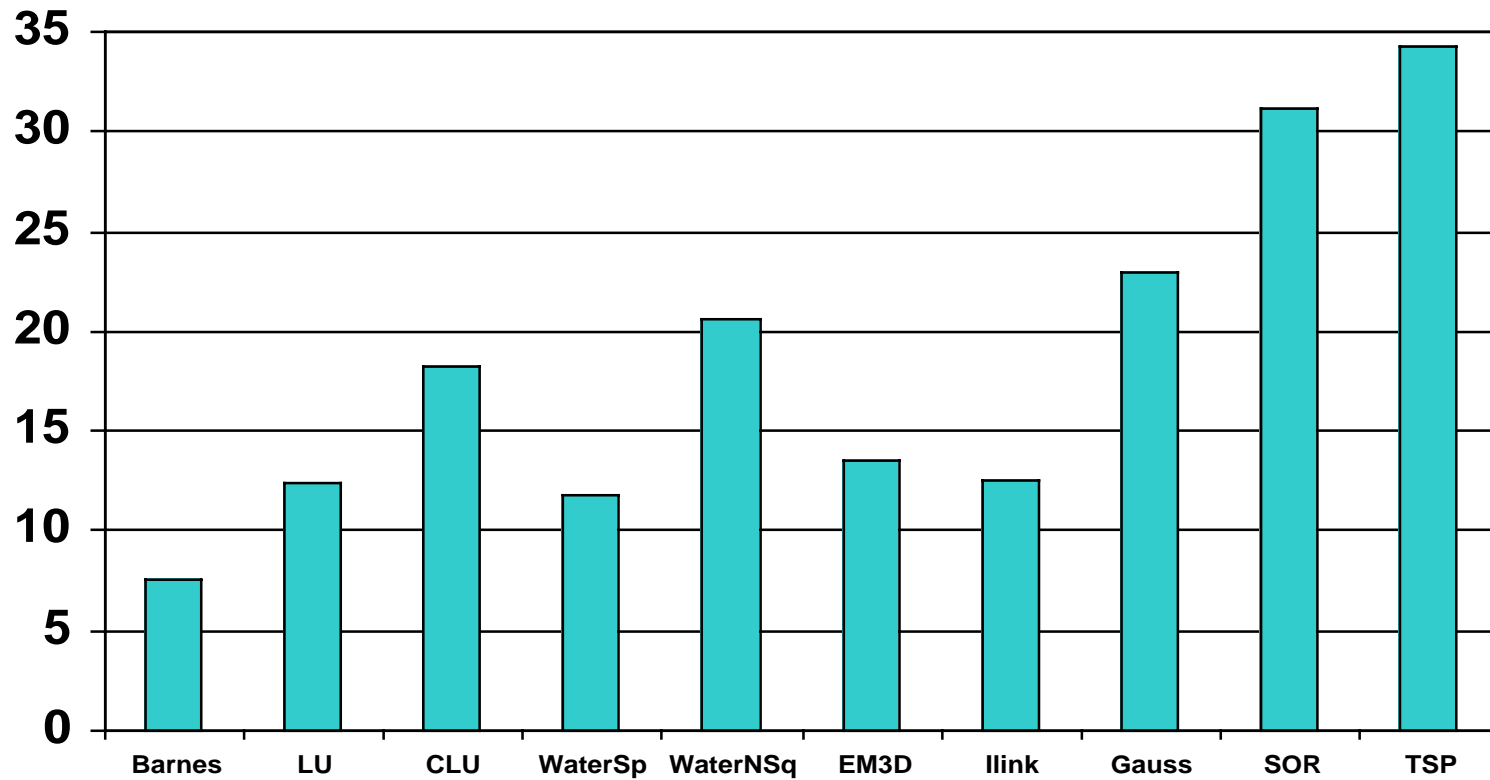
- No major OS vendor packages S-DSM
- TreadMarks the only commercially-available system
 - » Rice Spin-off
 - » Kuck and Associates OpenMP implementation
 - » Some noteworthy successes (e.g. NIH/FastLink)
- High-end users (who might tolerate research code) still stick to MPI

Where are the other success stories?

So Where Do We Stand Today?

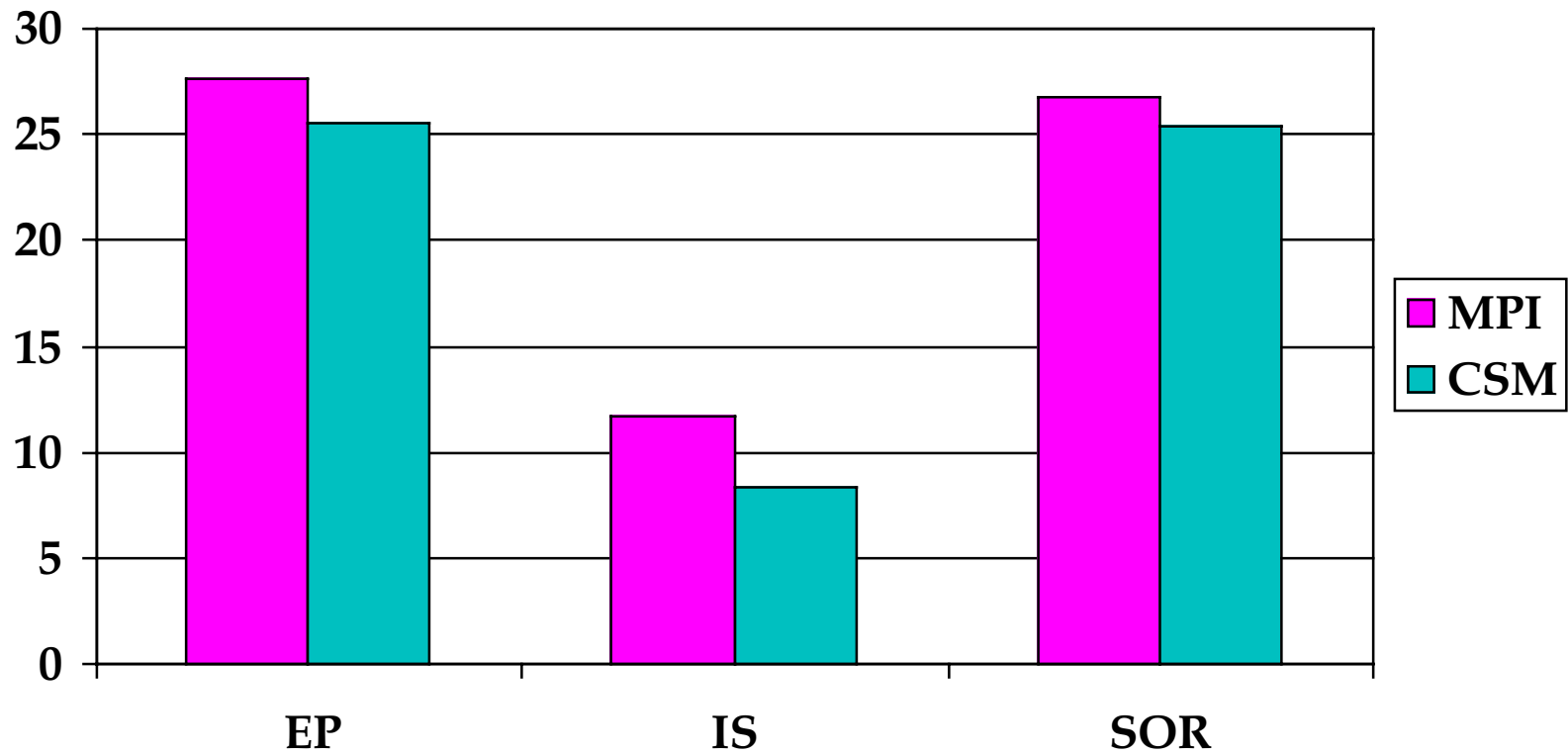
- Converging on the “right” implementation
 - » Relaxed memory model
 - » VM-based protocols (false sharing not a major issue)
 - » Multiprocessor nodes
 - » User-level network interface
- So-so performance
 - » OK for well-written apps on modest numbers of nodes, but
 - » Nobody has demonstrated real scalability

Cashmere Speedups



32 processors

Cashmere and MPI



HPC Is Not Our Niche

- We're not going to run S-DSM on 4000 nodes
- We're not going to match the performance of hand-tuned MPI code
- We're not going to convert the national labs

What is our niche?

Modest-sized clusters

and the applications they run

What We Don't Need

- More protocol tweaks
- New APIs
- More isomorphic implementations
- More SPLASH benchmarks
- More “scalable” systems tested on 16 nodes

Are there only four big ideas?

What We Need

- Single system image
 - » good debuggers
 - » process and memory management
- Compiler integration
- Non-scientific apps
 - » CSCW, OLTP, e-commerce, games
- Wide-area distribution
(for functionality, *not* performance)
 - » Heterogeneity
 - » Application-specific memory models
 - » Fault tolerance

Cashmere Plans

- Protocol tweaks; VIA, Linux, Myrinet ports
- Compiler integration (ARCH)
- Global memory management
- Applications
 - » CFD in Astrophysics
 - » Protein folding
 - » Laser fusion
 - » Object recognition
 - » Volumetric reconstruction
 - » Neural simulation
- MPI comparison
- InterWeave

InterWeave Motivation

- Convenient support for “satellite” nodes
 - » remote visualization and steering (Astrophysics)
 - » client-server division of labor (datamining)
- ★ True distributed apps
 - » intelligent environments (AI group)
- Speedup probably not feasible; convenience the principal goal

InterWeave Overview

- Sharing of persistent versioned segments, named by URLs
- User-specified relaxed coherence; reader-writer locks
- Hash-based consistency
- Full support for heterogeneity, using XDR and pointer swizzling (Eduardo's talk this afternoon)
- Cashmere functions as a single InterWeave node

Segment Creation

```
IW_handle_t h = IW_create_segment (URL);  
IW_wl_acquire (h);  
my_type* p = (my_type *) IW_malloc (h, my_type_desc);  
*p = ...  
IW_wl_release (h);
```

- Communicates with server to create segment (checking access rights)
- Allocates local cached copy (not necessarily contiguous)
- Can follow pointers to other segments

Coherence

- Relaxed reader-writer locks
 - » Writer grabs current version (does *not* exclude readers)
 - » Reader checks to see if current cached copy (if any) is “recent enough”
- Multiple notions of “recent enough”
 - » e.g. immediate, polled, temporal, delta, diff-based
 - » from Beehive [Singla97], InterAct [LCR '98]
- Cache whole segments; server need only keep most recent version, in machine-independent form
- Diff-based updates (both ways) based on block time stamps

Consistency

- Need to respect happens-before
- Invalidate cached version of A that is older than version on which newly-obtained version of B depends
- Ideally want to know entire object history
- Can approximate using hashing
 - » slot i of vector contains timestamp of most recent antecedent hashing to i
 - » invalidate A if $B.\text{vec}[\text{hash}(A)]$ is newer

InterWeave Related Work

- Distributed objects
 - » Language-specific (Emerald/Amber/VDOM, Argus, Ada, ORCA, numerous Java systems)
 - » Language-neutral (PerDiS, Legion, Globe, DCOM, CORBA, Fresco)
- Distributed state (Khazana, Active Harmony, Linda)
- Metacomputing (GLOBUS/GRID, Legion, WebOS)
- Heterogeneity, swizzling (RPC, LOOM, Java pickling)
- Multiple consistency models (Friedman et al., Agrawal et al., Alonso et al., Ramachandran et al., web caching work)
- Transactions, persistence (Feeley et al., Thor)

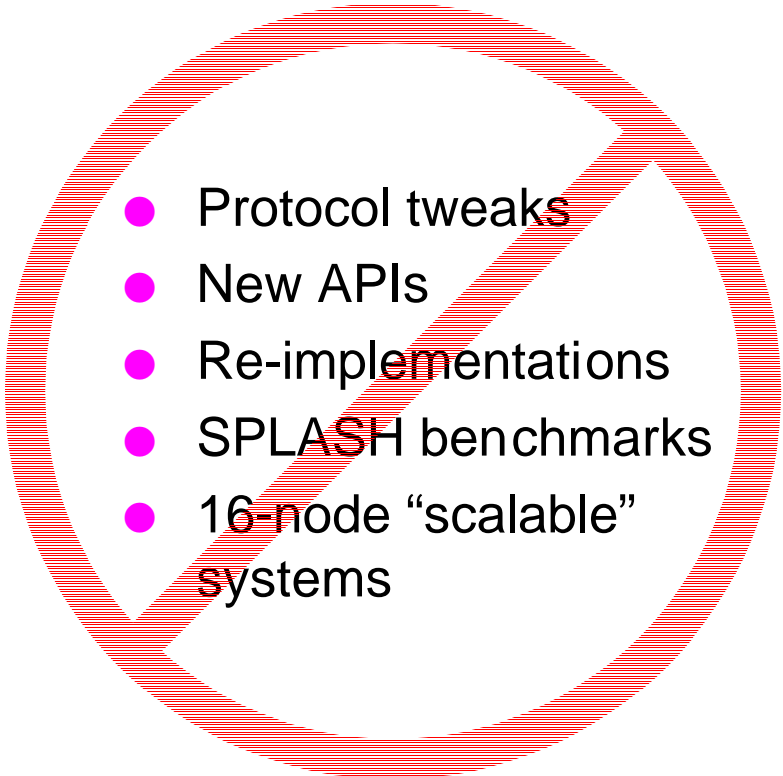
Status

- Prototype running on Alpha cluster
- Barnes-Hut demo running: Cashmere plus remote front end
- Distributed game in the works (MazeWars)
- Coherence models in; consistency in the works
- XDR compiler finished; heterogeneity in the works
- Extended abstracts at WSDSM and LCR

see <http://www.cs.rochester.edu/research/interweave>

What We Need (Reprise)

- Single system image
 - » good debuggers
 - » process and memory management
- Compiler integration
- Non-scientific apps
 - » CSCW, OLTP, games, e-commerce
- Wide-area distribution
 - » Heterogeneity
 - » Application-specific memory models
 - » Fault tolerance

- 
- Protocol tweaks
 - New APIs
 - Re-implementations
 - SPLASH benchmarks
 - 16-node “scalable” systems

A Plug for LCR

Languages, Compilers, and Run-Time Systems for Scalable Computers

May 25-27, 2000

Rochester, NY

Program and registration information available at
<http://www.cs.rochester.edu/~LCR2k>