CRL: High-Performance All-Software Distributed Shared Memory

Kirk L. Johnson

University of Colorado, Boulder

M. Frans Kaashoek, Deborah A. Wallach

Massachusetts Institute of Technology

Introduction

- Goal: cost-effective high-perf computing
  distributed systems
  ease of programming

- Message passing
  portable, efficient, but hard to program

- DSM improves programmability

Distributed Shared Memory (DSM)

- Goal: DSM with portability, efficiency
- Tension between HW and SW
  performance vs. implementation effort

C Region Library (CRL)

- Shared memory model
- Portable
- Efficient

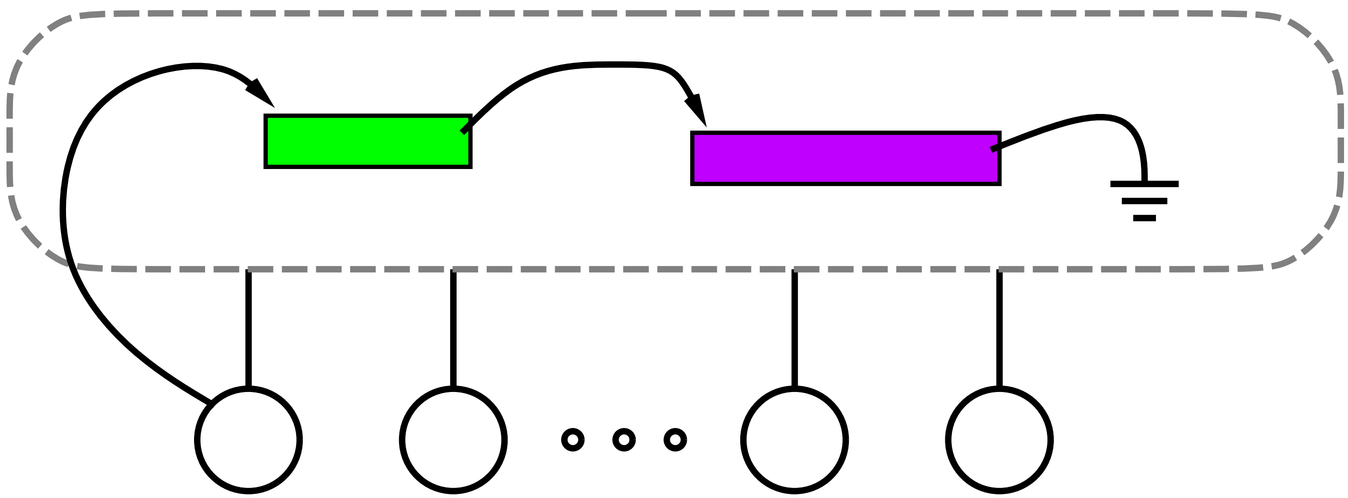
- Controlled comparison with HW DSM

$\Rightarrow$ CRL performance within 15%

Outline

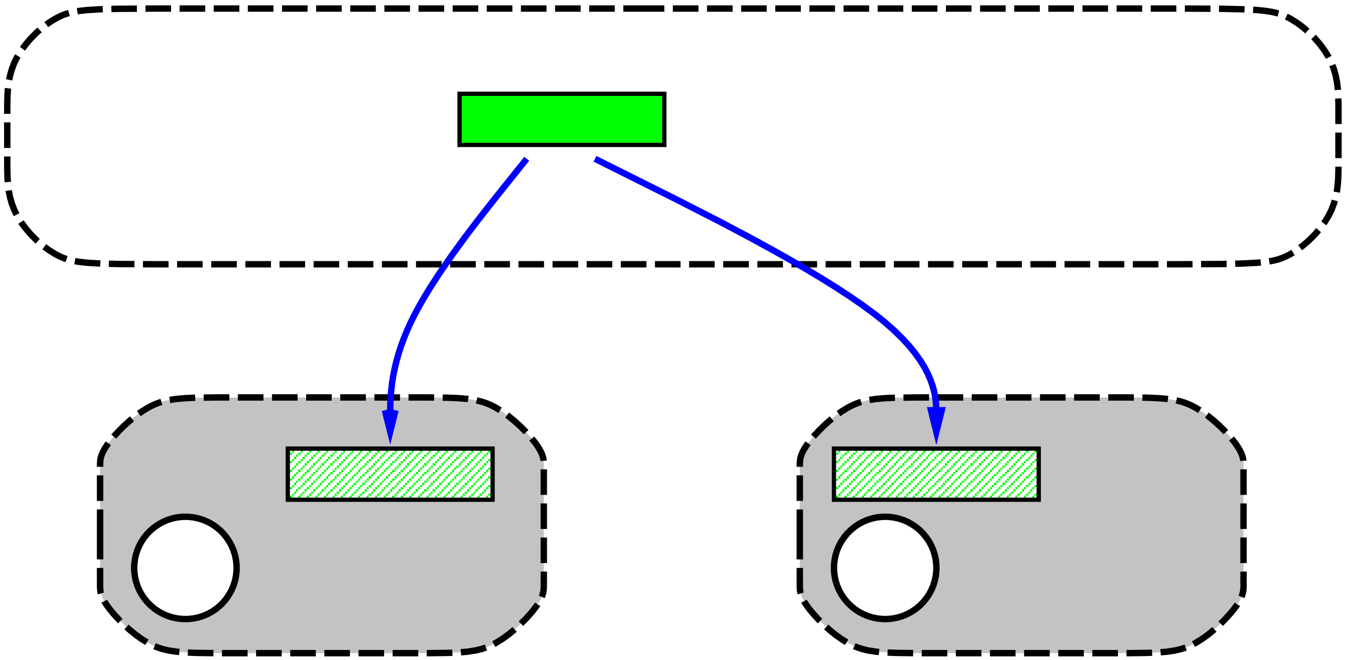
- Introduction
- The CRL approach
- Framework and methodology
- CRL vs. hardware DSM
- CRL on distributed systems
- Conclusions

Communicate through *regions*



- Contiguous area of memory
- Application defined, variable size
- Named by region identifiers
- Can be created dynamically

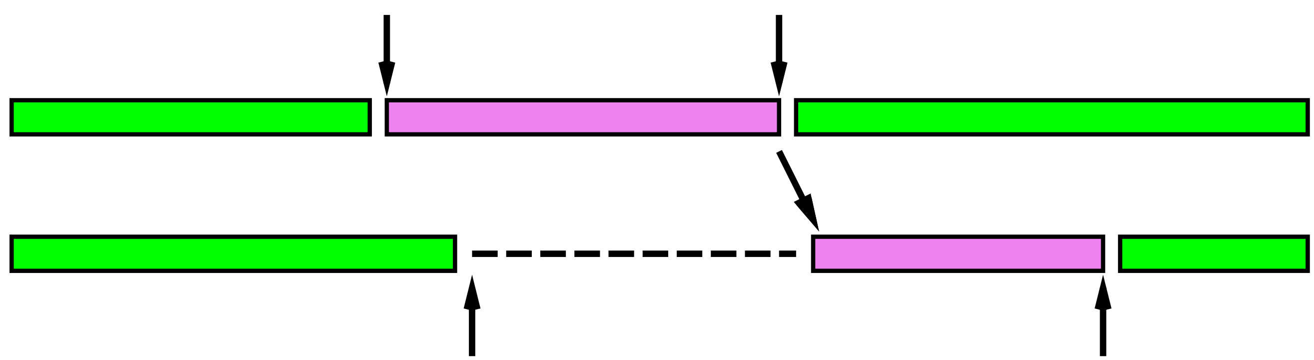
Mapping/unmapping



- Before accessing, regions must be mapped
- After accessing, they can be unmapped

Group accesses into *operations*

- Annotate program to delimit operations
- Read & write operations
- Integrate data access and synchronization



Programming model summary

Modest differences from 'standard' DSM

- Annotations delimiting operations
- 'Global' *vs.* 'local' pointers

Our experience: low programmer overhead

Prototype implementation

- Regions are cached
- Fixed-home, invalidate-based protocol
- Handles out-of-order message delivery
- Implemented entirely as a library
- Runs on three platforms
  (CM-5, Alewife, TCP/Unix)

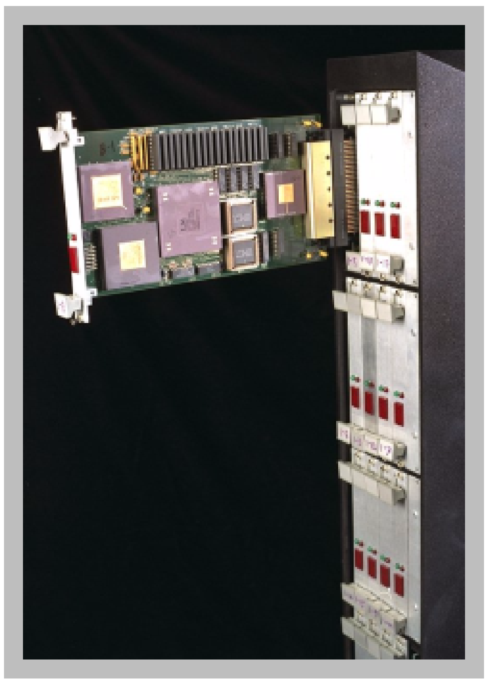
Thinking Machines CM-5

128 nodes

round-trip: 1088 cycles

bandwidth: 0.25 bytes/cycle

comparable to NOW



MIT Alewife Machine

32 nodes

round-trip: 528 cycles

bandwidth: 0.9 bytes/cycle

supports both SM and MP

Applications

Application	Region size (bytes)	Region number	Cycles/ CRL op
Blocked LU	800	2,500	11,000
Water	672	500	1,540
Barnes-Hut	100	16,000	436

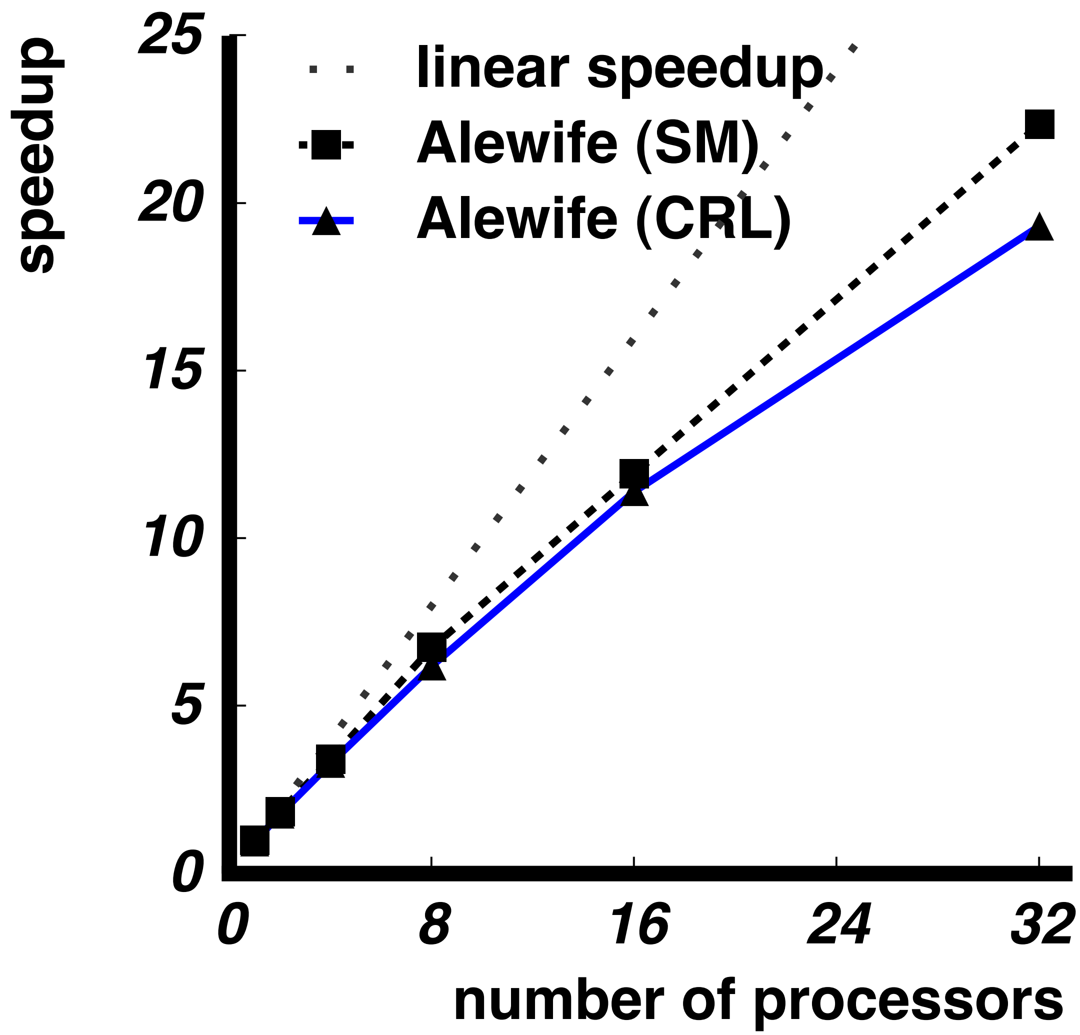
- Direct port of original shared memory code

CRL vs. Hardware DSM

Can CRL deliver performance competitive with hardware DSM?

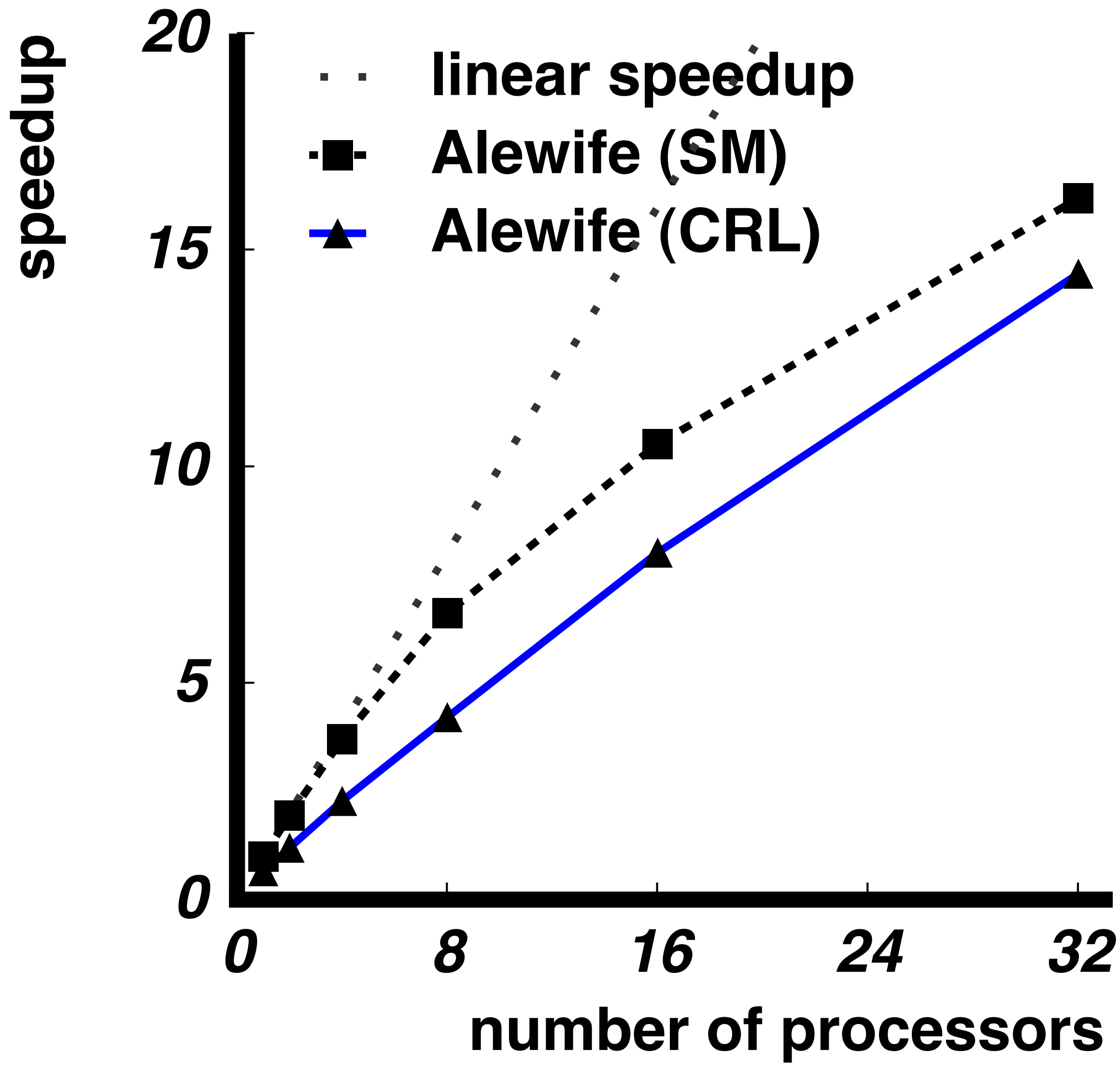
- Controlled comparison using Alewife

Water (medium grained)



Water (512 molecules)

Barnes-Hut (fine grained)



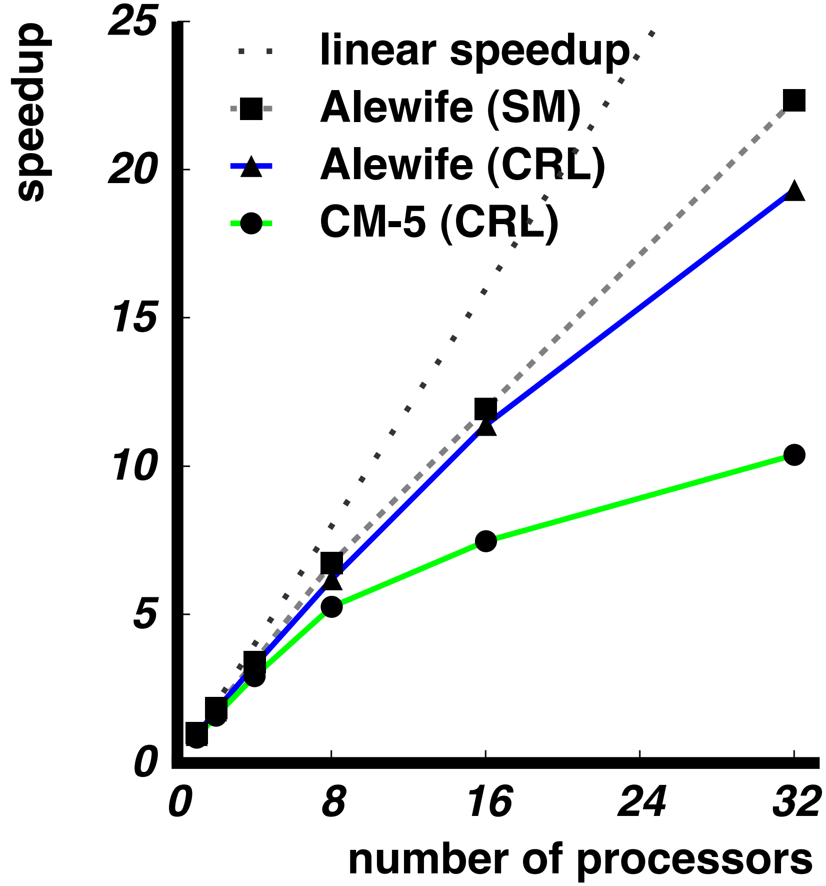
Barnes-Hut (4,096 bodies)

CRL on distributed systems

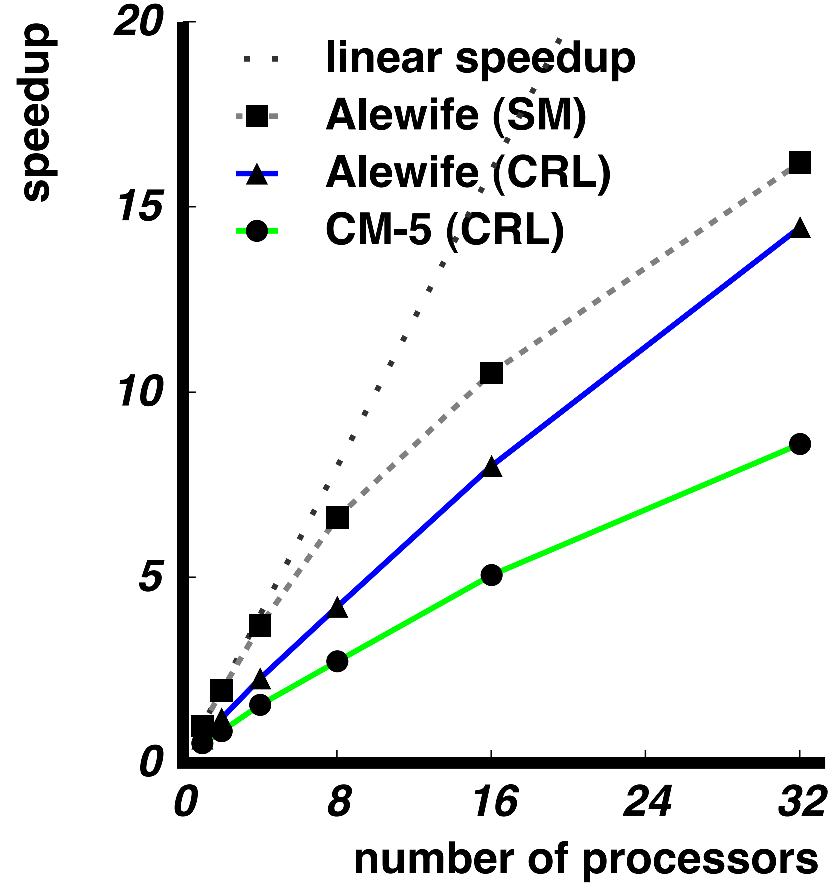
What about impact of increased communication costs on CRL?

- Compare CRL on Alewife and CM-5

CM-5 CRL vs. Alewife CRL

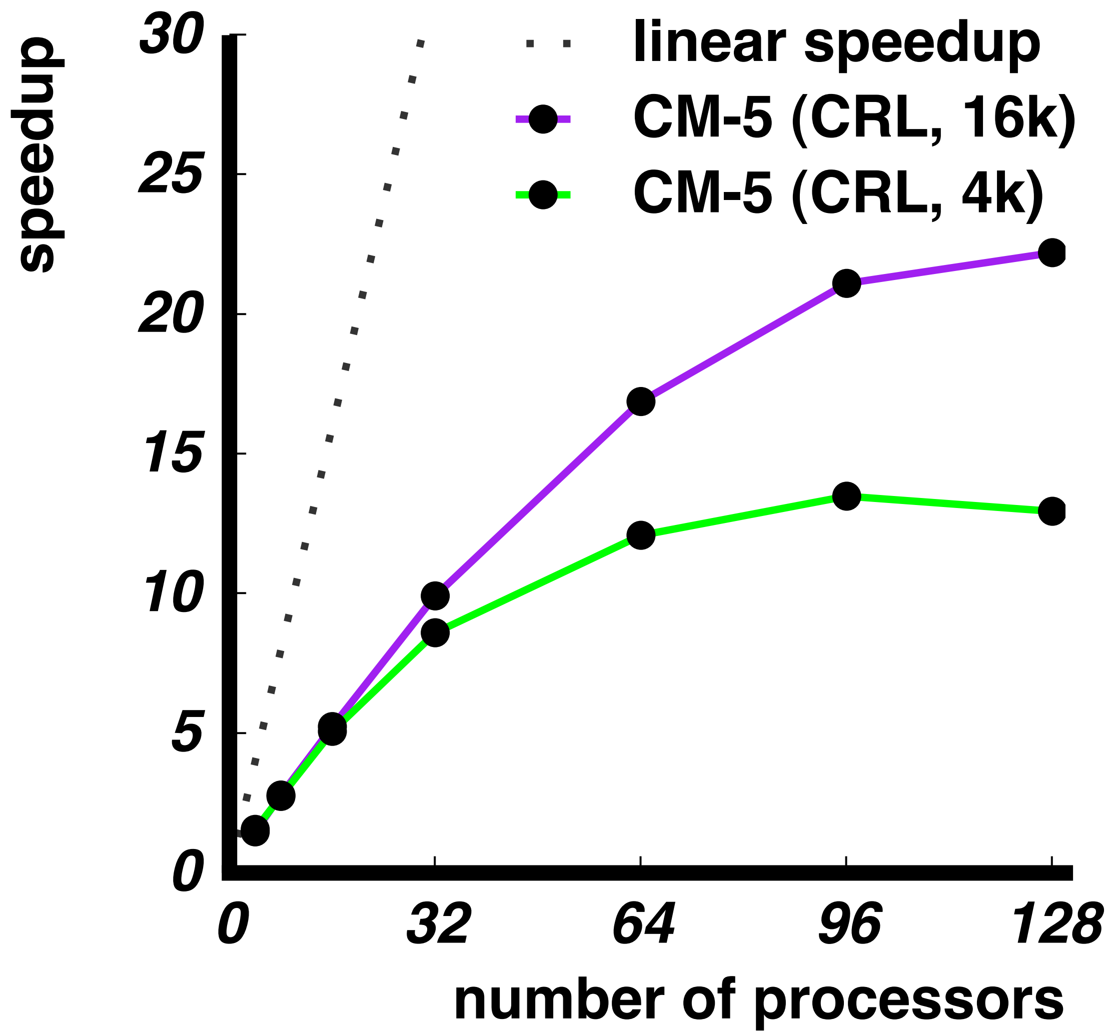


Water
(512 molecules)



Barnes-Hut
(4,096 bodies)

Larger problem and machine sizes



Barnes-Hut (4,096 and 16,384 bodies)

Why does CRL do well?

- Simple, efficient implementation
- Overhead amortized over many references
- No problems from fixed-size coherence units

Conclusions & contributions

- CRL (simple, portable, efficient, scalable)
- First controlled comparison of scalable hardware and software DSM systems
- CRL delivers competitive performance!
- Hardware support not necessary
  reduced implementation effort
  increased flexibility