# Distributed Shared Memory

# CS239
# Advanced Operating Systems

## November 16, 1999
## Dr. Richard Guy



# Lecture Source Materials

Singhal, Ch. 10: **Distributed Shared Memory**

**Towards Transparent and Efficient Software DSM**, Scales et al, SOSP 97.

**Cashmere-2L:Software Coherent Shared Memory on a Clustered Remote-Write Network**, Stets et al, SOSP 97.



# Outline

- Introduction and Motivation
- Basic Approaches
- Memory Coherence and Protocols
- Design Issues
- Example Systems
- Recent Results in DSM Research
- Summary

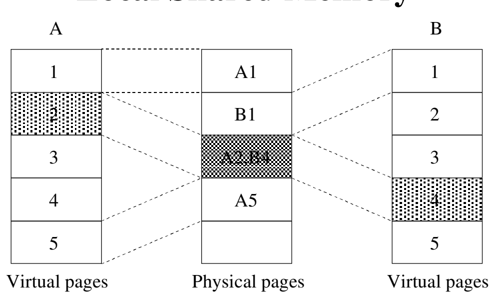


# Introduction and Motivation

- Local shared memory
- Basis for distributed computation?



# Local Shared Memory






# Local Shared Memory

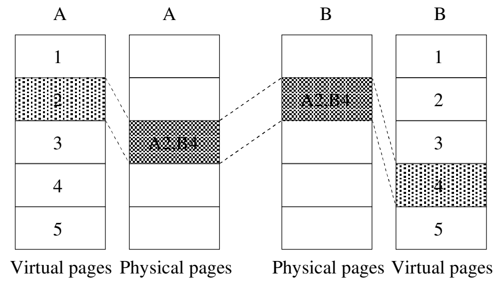
- features
  - "cheap" IPC for cooperating processes
  - synchronization left to processess
  - useful in tightly-coupled multiprocessors
    - exploit concurrent execution
    - exploit common physical memory pool

## Local Shared Memory, cont.

- potential problems
  - coherence
    - read/write access serialized by memory hardware
  - VM page thrashing
    - shared pages are "stickier" than non-shared pages



## Distributed Shared Memory






## Motivation

- DSM hopes to use loosely-coupled processors to get benefits of SM without paying high $$ cost for multiprocessor



## DSM Advantages (claimed)

- simpler abstraction than message passing
- easier to access complex data structures
- exploits locality of reference properties
- pools all physical memory in one huge memory
- scalable (e.g., no common bus



## DSM disadvantages (possible)

- Applications know a lot about synchronization methods/tools
- Performance will suffer if too much remote memory access



## The Challenge

Can cooperating processes on processors that do not share a common physical memory be presented a coherent shared memory space?

Yes,

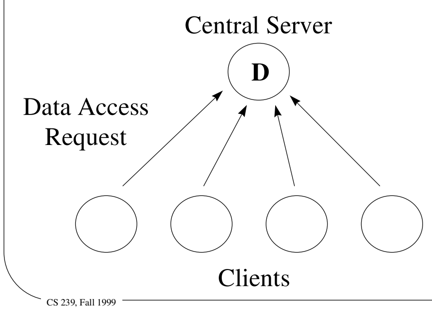
… and sometimes with a reasonable cost in performance!

## Approaches to DSM

- Central server
- Migration
- Read replication
- Full replication



## Central Server DSM






## Central Server DSM

- A central server maintains all data pages
  - returns data on reads
  - makes updates on writes
- Care needed to avoid duplicating writes
- Server duties can be split between several sites, by statically partitioning the shared memory page pool

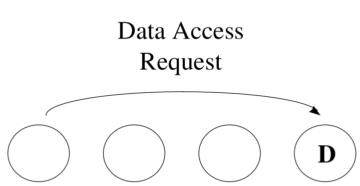


## Pros/Cons of Central Server DSM

+ Relatively simple to implement
+ Synchronization methods pretty clear
– Most data references are remote
– Prone to bottlenecks

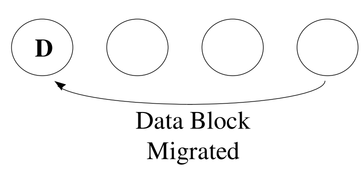


## Migration DSM






## Migration DSM



## Migration DSM

- Data lives where most recently used
- When “off-site” memory is requested, it migrates to the requesting site
- Granularity is usually block or page level
- With locality of reference, works well
- But possibility of thrashing
- Servers keep track of data locations

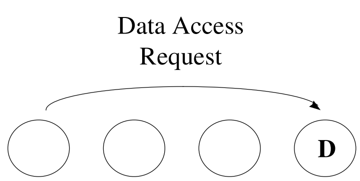


## Pros/Cons of Migration DSM

+ If locality good, most accesses can be local
+ Synchronization behavior fairly easy
– Between-node thrashing dangers
– Possible bottlenecks at location servers
– May put heavy burden on communications

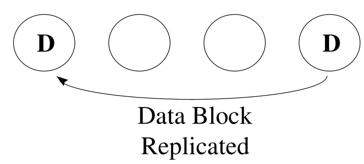


## Read Replication DSM



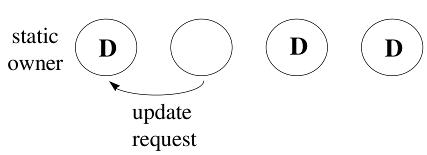



## Read Replication DSM



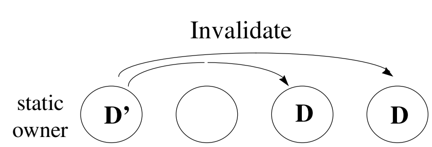



## Read Replication DSM with update






## Read Replication DSM with update



## Read-Replication DSM

- Multiple copies of readable blocks stored
- If block is written, invalidate other copies
- Each block has owner that permanently stores it (and it is the only updatable replica)
- If writes are frequent, invalidation can

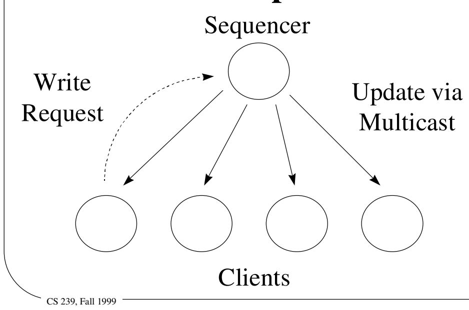


## Pros/Cons of Read Replication DSM

+ Reads are usually local
+ Synchronization not too difficult
– Requires invalidations
– Possibly heavy burden on owners of popular pages
– Writes are only local for page owner



## Full Replication DSM with update






## Full-Replication DSM

- Full-service replicas of data blocks/pages stored at several machines
- Requires consistency mechanism
  – for same reasons as replicated files
  – different mechanisms may be appropriate



## Pros/Cons of Full Replication DSM

+ Access usually local
+ Bottlenecks generally less likely
– Synchronization tricky
– Requires full replication mechanism
– Likely to be overkill unless data sharing very common



## Memory Coherence in Replicated DSM Schemes

If more than one copy of a data item exists, problems arise when data is written and “later” read

*Consistency model* describes what level of consistency a DSM system provides

## Consistency Models for DSM

- Strict consistency (as if only one copy)
- Sequential consistency (serializability)
- General consistency (eventual consistency)
- Processor consistency (per machine)
- Weak consistency (up to programmer)



## Protocols to Provide DSM Coherence

- Write-invalidate protocol
  - Invalidate other copies before write
  - Expensive for those other copies
- Write-update protocol
  - Update other copies on write
  - Higher network traffic results
  - Requires some kind of locking



## Controlling Thrashing in DSM

How to prevent 2 sites sharing a page for write from thrashing the net?

One way is to lock a written page down at a site for some period $\Delta$

If another site wants to write it, that site must wait for $\Delta$ to expire

Used in Mirage system



## Design Issues

- Granularity of shared item
- Page replacement policy
- Exploiting semantic hints



## Granularity in DSM

How big should the unit of shared memory be?

- Should be related to hardware's page size
- Size of page has many important implications...



## Pros/Cons of Large Pages for DSM

+ Takes advantage of locality of reference
+ Minimizes cost of running paging code
+ Can minimize communications cost
  - If lots of data being moved

– Leads to greater contention

# Pros/Cons of Small Page Size for DSM

+ Less contention
+ Minimizes *false sharing*
+ Lower cost to move small data objects
– Greater overhead per byte

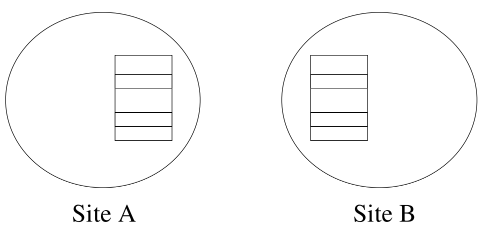


# False Sharing in DSM

- Occurs when an apparent, but not actual, write conflict occurs between two copies of a page
- Caused by viewing replication unit as a page, rather than a memory location (or object)

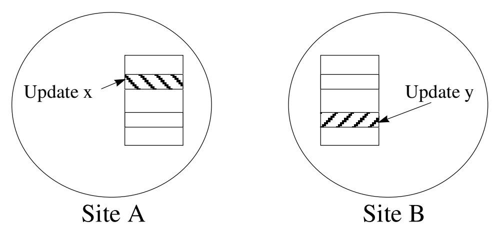


# False Sharing Example






# False Sharing Example, Con't




There appears to be a conflict, but there really isn't



# Effects of False Sharing

- Depends on approach used
  - In migration systems, causes more page migration
  - In write-invalidate replication systems, causes more invalidations
  - In update-propagation replication systems, causes more locking



# Page Replacement in DSM

- When a local node needs to bump a shared memory page, what does it do?
- LRU alone not a sufficient answer
- Must consider private vs. shared pages
  - And replication issues
  - And read-only pages
- LRU within classes often used

## Exploiting Semantic Hints

- Write-once
- Private
- Write-many
- Result
- Synchronization
- Migratory
- Producer-consumer



## Semantic hints, cont.

- Read-mostly
- General read-write
  - invalid
  - unowned
  - owned exclusively
  - owned nonexclusively



## Some DSM Systems

- Ivy (Yale)
- Mirage (UCLA)
- Clouds (Georgia Tech)
- Munin (Rice U.)
- Memnet (U. of Delaware)
- Telegraphos (Greece)



## Recent DSM Research

- SOSP 97
  - Shasta
  - Cashmere-2L
- Emergence of low-cost small-scale SMP with hardware shared memory
- Low-latency remote-write networks (e.g., DEC's Memory Channel)



## DEC's Memory Channel

- Low-latency hardware interconnect
- Knows about process-process connections (up to 64K)
- Appears as both device (outgoing) and physical memory (incoming)
- Write-only paradigm…
  - how to do a remote read?



## Shasta

- Goals
  - transparent execution of standard binaries (no special knowledge of software DSM services)
  - fine-grained sharing (smaller than page-level)

## Shasta techniques

- Cache coherence protocol
  - private v shared virtual address space
  - shared states
    - invalid vs shared vs exclusive
    - shared miss
- Code insertion to detect shared miss load/store before it actually occurs



## Shasta Techniques, continued

- Avoid OS involvement where possible
  - use inline code for shared misses, not OS/VM interrupt handling
  - use polling for Memory Channel, not standard device interrupt



## Cashmere

- Goal
  - two-level shared memory support
    - hardware within SMP “node”
    - software across nodes
  - uses release consistency, a la earlier DSM implementations



## DSM Summary

- Basic idea of DSM is…
- Primary problems of DSM are…
- Recent HW trend responses are...