Ch 10

• Shared memory via message passing

• Problems
  – Explicit user action needed
  – Address spaces are distinct
  – Small Granularity of Transfer

• Distributed Shared memory approach can help with these. Also, unlike tightly coupled multiprocessors
  – Cheaper to build using COTS
  – Memory pooled together is significant than local workstation memory
  – More scalable since data bus is not a bottleneck
  – SMM based programs can be easily ported
Implementation Approaches

• Central Server Based
  – A central server maintains all shared data. Provided to processors using request/response model with timeouts.
  – Problem: Scalability
  – Solution: Partition data, allocate each partition to a processor and have it coordinate requests for that partition.
  – Need a “mapping function” to map VM address to corresponding processor.
• **Migration Algorithm**
  – Instead of sending request to data, send data to request
  – Send a “larger” block of data than needed
    • Locality of reference
  – Access is serialized
  – Can lead to thrashing
    • Avoid by using hold downs
  – Can allow for integration with local VM mechanisms
    • Need to have conforming page sizes in VM and DSM.
  – How do you locate a block
    • Centralized mapping server
    • Hints
    • Broadcast based discovery
• **Read Replication**
  – Enhance basic migration by allowing multiple read copies and one write copy.
  – Invalidation on read?
  – Useful when read/write >> 1

• **Full Replication**
  – Allow multiple readers and writers
  – Consistency?
    • Use a sequencer
    • Process in sequence order
Memory Coherence

• In a DSM with replication, what is the semantics of memory access?
  – Need to define a memory consistency model
  – Strict Consistency – read returns latest write
  – Sequential Consistency – the result of any execution of operations of all processors is the same as if they were executed sequentially, and operations of a particular process happen in sequence
  – General Consistency – All copies of the memory location eventually contain the same data when all writes have completed
- Processor Consistency – writes issued by processors occur in order, but not across processors. So simultaneous reads on different processors can lead to different values.

- Weak Consistency – Synchronization access are sequentially consistent. Regular data accesses and synchs aren’t mixed. Synch. Up to the programmer.

- Release Consistency – Acquire/manipulate/release paradigm. Can mix in some combinations. Synchs are processor consistent.
Coherence Protocols

- Write Invalidate or Write Update
- Coherence in PLUS system
  - Page is the unit of replication, word is unit of consistency
  - One replica is the “master”. Each replica points to the master and to the next replica. This forms a distributed copy list.
  - On read fault for remote memory, MCM sends message to remote processor and receives data
  - On write, the operation is first performed at master, and then propagated to replicas.
  - Writer is not blocked unless it wants to read from that location.
  - Guarantees in process ordering, but not across processors.
• Clouds system uses synchronization locks for memory coherence. Locking process gets the data segment. Reverts back to owner upon release.

• Application Specific hints
  – Write once objects
  – Private objects
  – Write Many (use delayed updates, weak consistency)
  – Result Objects are a subset of write many, which are read after writes.
  – Synchronization Objects – proxies used for lock management.
  – Migratory objects – accessed in phases (critical section)
  – Producer consumer objects – eager movement.
  – Read Mostly objects – broadcast updates
  – General Objects
• General Objects
  – Invalid
  – Unowned – have valid data and may be replicated. Need to take ownership before updating
  – Owned exclusively -- has valid data and updatable locally. Must be shared if requested.
  – Owned non-exclusively – has valid data, but need to invalidate others before updates.
  – Read operations can be shared or for ownership.
Design Issues

• Granularity
  - Multiple of underlying page?
  - Tradeoff between size and contention
  - Combination by separating coherence from replication
  - Adaptivity?

• Replacement
  - Can’t use things like LRU directly because of sharing modes
  - Avoid disk swapping by memory reservation.
IVY Case Study

- Strict Consistency using multiple reader, single writer semantics and write invalidation
- Read Fault: Contact page owner. Owner adds you to its copyset and sends replica.
- Write Fault: Contact owner. Owner sends page and copysets, and invalidates its own entry. You store the page and send invalidation message to all in copyset.
- Manager can be centralized, distributed or dynamic distributed