Distributed Shared Memory

Motivation

- DSM is a system module that allows machines with no physical shared memory to share a single virtual address space
- benefits
  - easier to design and implement parallel algorithms
  - easier to share data by reference
  - amortizes access to remote data by exploiting locality of reference
  - cheaper than shared physical memory multiprocessor systems
  - huge physical memory available to all nodes
  - data bus is not a bottleneck
  - program written for shared memory multiprocessors can run on DSM
**Basic DSM Algorithms**

- central-server
  - all memory access requests go through a central server that maintains the shared memory
- migration
  - migrates memory page to requestor
  - needs to keep track of node having a page
  - thrashing is possible
- read-replication
  - multiple nodes have a read copy of a page, one node has a write copy
  - keeping track of owner and replica consistency
- full-replication
  - gap-free sequencer

**Memory Coherence**

- Memory is coherent if the value returned by a read is what the programmer expected
- coherence types
  - strict consistency: value read is the most recently written
  - requires total ordering of requests \( \Rightarrow \) expensive
  - sequential consistency (e.g. serializability)
  - general consistency (all copies eventually have same value)
  - weak/release consistency (ala two-phase locking) with
Coherence Protocols

- Protocols for maintaining memory coherence
- write-invalidate
- write-update
  - PLUS
    - master copy is updated first, then all replicas in replica set
    - writer is not blocked unless it wants to read a location with a pending update
  - Clouds
    - locking is used in a readers-writers setting
    - combines locking with data transfer from/to owner

Write-Update Coherence Protocols

- Berkeley ownership protocol (ala locking with modes)
  - each object has an owner
  - an object at a node can be in one of these states
    - invalid
    - unowned (but with valid data)
    - owned exclusively (valid data - single copy)
    - owned shared (valid data - multiple copies)
  - operations on an object can be
    - read-shared
    - read-exclusive
    - write
Type-Specific Memory Coherence

- Exploit application specific memory object semantics
- object types in Munin
  - write-once
  - private
  - write-many (multiple writes between locks)
  - result (multiple updates to different parts before a read)
  - synchronization
  - migratory (accessed in phases by one process in each phase)
  - producer-consumer (eager object movement)
  - read-mostly (update via broadcast)
  - general read-write (use Berkeley ownership protocol)

Write-Invalidate Coherence in IVY

- Each page has
  - an owner
  - a copyset (set of nodes with read-only copies)
  - state (invalid, read-only, write)

- on a write fault for page p
  - writer finds owner of p
  - owner of p sends p to writer together with copyset
  - writer invalidates all copies in copyset

- on a read fault for page p
  - reader finds the owner of p
  - owner adds reader to copyset and sends p to reader
Maintaining Page Ownership in IVY

- Centralized manager
  - faulting node sends its read/write request to CM
  - CM maintains the owner of each page
- Fixed distributed manager
  - faulting node for page p contacts a CM identified by H(p)
- Dynamic distributed manager
  - every page table entry has a probable owner field
  - true owner of a page is found by chasing the probowner pointers
  - probowner is updated when node receives invalidations, transfers ownership, receives a page, or forwards a request
  - double fault due to a successive read/write may occur; avoid them by using sequence (version) numbers

DSM Granularity & Replacement Policies

- Size of shared memory page
  - large ==> lower overhead
  - small ==> less contention for shared access
- Page Replacement policy
  - LRU with classes
  - swapping out must ensure no data loss