LECTURE- 24
Distributed Shared Memory

Shared Memory

We have already seen some kinds of shared memory mechanisms

- shared memory symmetric multiprocessors:
  - all memory is shared and equidistant from processors
  - caching is used for performance

There are also other kinds of shared memory approaches

- NUMA (non-uniform memory access) machines
  - each machine has its own memory but the *hardware* allows accessing remote memory
  - remote addresses are no different than local accesses at the assembly level
  - access to remote memory is much slower
  - no hardware caching occurs (but the software may do caching)

Why Distributed Shared Memory

- collaborative or concurrent applications use two main mechanisms to communicate or synchronize: message passing (send/receive) or shared memory
- to port these apps to systems without physical shared memory, you need support for DSM
  - need this for scaling across multiple disjoint systems/nodes (supercomputers, MPP are too costly)
  - even some many core systems don’t have uniform memory access from all cores
  - need to take advantage of RDMA mechanisms

Distributed Shared Memory

- objectives: *minimize latency* and *keep coherent*
- software presents the abstraction of shared memory
- either an OS or a language run-time manages the shared memory
- there are many different DSM granularities
  - page-based DSM: like regular virtual memory (e.g., Clouds here at Tech!)
  - shared-variable DSM and object-based DSM: managed by a language run-time system

Implementation

- Think of how you would implement such a "virtual" shared memory among workstations
  - Software: at the OS level for unsuspecting processes
    - how can this be done efficiently?
  - Software: at a language runtime level (e.g., Java VM)
    - how can it be done without hardware support?
  - Hardware level support
    - most efficient, but costly; many exotic interconnects have some support (Quadrics, Infiniband…)
  - Hybrid

Granularity of sharing

- cache/bus line -> overkill in distributed environments
- page-based
- object-based -> typically language/runtime support

Granularity tradeoffs

- finer granularity
  - improve concurrency, increase communication and frequency of execution of consistency related protocols
- coarser granularity
  - limit concurrency (especially is single writer only), reduce comm/consistency protocols
  - issue with false sharing
    - may be able to reduce with careful layout of data structures, hard if implemented at system/hardware level

Objective: Reduce Latency

- Approach:
  - migration
    - migrate shared unit (page, object, etc.) to node which has current access -> need state to determine current location
  - replication
    - keep multiple copies -> need per shared unit state for keeping track of replicas, and types of access at each replica

Access Algorithm

- Single Reader – Single Writer
  - simplest, migration can suffice, not very efficient
- Multiple Readers – Single Writer
  - more general, need to track current/most recent writer (aka owner)
- Multiple Readers – Multiple Writers
  - maximum concurrency, need to resolve write conflicts through consistency protocols

DSM vs. other shared memory mechanisms

- We have seen simple cache consistency mechanisms in two different settings:
  - SMPs (snooping caches, shared bus, write-update or write-invalidate protocols, write buffers, etc.)
  - distributed file systems (client or server-based protocols, leases, invalidation, delayed writes)
- Similar mechanisms apply to DSM, but they have to be more sophisticated
  - DSM will be used for synchronization and inter-process communication
  - DSM has to be very fast
  - comparable in speed to memory, not to a file system
  - caching is paramount
  - no central resources exist (like a bus we can snoop on or lock it cheaply)

Page-based DSM

- The issues in implementing DSM are similar to what we have seen so far, but some complications arise in the page-based case
  - which is the most common case for getting DSM running on machines with no shared memory support in hardware (e.g., workstations over ethernet)
- Pages are coarse grained—false sharing may occur
  - extra invalidations in a write-invalidate protocol
  - possibly overwriting other data in a write-update protocol
- Realistically, write-invalidate is the only option for page-based DSM
  - hard to do updates on every write: protection is at page granularity

Coherence Policy

- write update
  - allows multiple readers and writers
  - multicast message to update peer copies
  - processes have to agree on a total order for multicast writes for SC
- write invalidate
  - multiple readers, single writer
    - first invalidate peers
    - write to local cache
    - subsequent reads will get this new value

Write-invalidate protocol

- Not particularly exciting—usually straightforward and exactly what you would expect
- Interesting point: the “owner” of a page can be changing (e.g., can be the last writer to the page)
- The owner is responsible for blocking “writes” until all outstanding copies have been invalidated
- How to find the owner?
  - directories
  - distributed directories (statically or dynamically)
  - hints (probable owner, owner links, and periodic broadcasts so that all processors know a recent owner)

Synchronization

- Strict synchronization (e.g., mutual exclusion) can be too costly in DSM
  - spinlocks are a disaster as they require transferring pages only to find out that the lock is still busy
- A centralized synchronization manager may exist

Sample Implementation Techniques

- DSM mapped to same VA on all nodes
- kernels at individual nodes responsible for page level protection
- page states: none, read-only, read-write
- two data structures:
  - owner(P): last writer of page P
  - copyset(P): current sharers for P
- home node (also called manager node)
  - where directory info for a page is kept
  - m nodes and N pages, distribute the work, i.e each node responsible for m/N pages
  - given a page, you can get the home node
- owner node
  - the node that has write permission for a page, at any given time

ASSIGNMENT

- Q: Explain page based DSM.