1 Overview

- multicomputers – easier to build, harder to program
- multiprocessor – harder to build, easier to program

1.1 DSM(Distributed Shared Memory, Li 1986)

workstations connected by LAN
local page reference
page fault – fetch a page from remote machine
conceptually easy to use
poor performance

1.2 Goals of DSM research

minimize network traffic
reduce the latency between request and completion

- possible solutions
  1. share only a selected portion of the address space
  2. replicate the shared variables on multiple machines

1.3 Shared Memory Architectures

1. On-Chip Memory (Fig 6-1)
2. Bus-Based Multiprocessors (Fig 6-2)
3. Ring-Based Multiprocessors (Fig 6-5)
   Memnet (Delp, Farber) – DSM machines
4. Switched Multiprocessors (Fig 6-6)
   scalable, clusters
5. NUMA (Non Uniform Memory Access) Multiprocessors

Spectrum of SM machines (Fig 6-10, Fig 6-11)

2 Distributed Shared Memory (DSM)

- data in shared address space accessed as in traditional VM.
- mapping manager – maps the shared address space to the physical address space.

2.1 Advantage of DSM

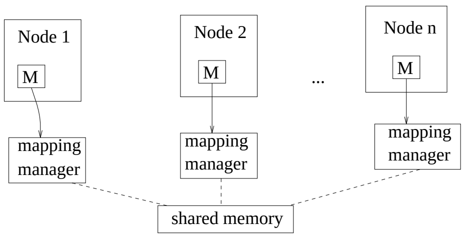
1. no explicit comm. primitives, send and receive, needed in program. It is believed to be easier to design and write parallel alg's using DSM

2. complex data structure can be passed by reference.

3. moving page containing the data take advantage of locality and reduce comm. overhead.

4. cheaper to build DSM system than tightly coupled multiprocessor system.

5. scalable – improved portability of programs written for multiprocessors.



2.2 Issues in implementing DSM

1. how to keep track of the location of shared data

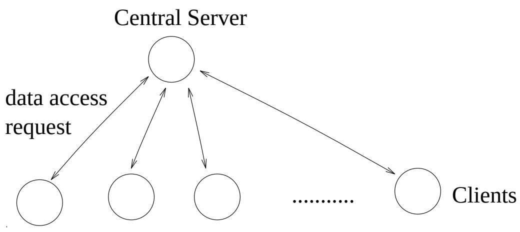
2. how to overcome comm. delays and protocol overhead when accessing remote data

3. how to make shared data concurrently accessible.

2.3 Basic Algorithms

2.3.1 The Central-Server Algorithm

1. a server maintains all shared data



2. read request $\longrightarrow$ send data to client.
   write request $\longrightarrow$ update shared data
   use timeout/retry for write req/ack and version number.

3. central server bottleneck

4. multiple servers, each server responsible for a partition of shared data.

2.3.2 The Migration Algorithm (Fig 6-25(b))

1. data is shipped to the location of the data access request and subsequent accesses to the data performed locally.

2. only one node to access shared data at a time

3. susceptible to thrashing.

4. could be integrated with the virtual memory. If there is a page fault and the memory address maps to a remote page, a fault-handler will migrate the page.

5. for locating a data block,

   (a) servers keep track of the location of page

   (b) use hints maintained at nodes

   (c) broadcast query to locate a page

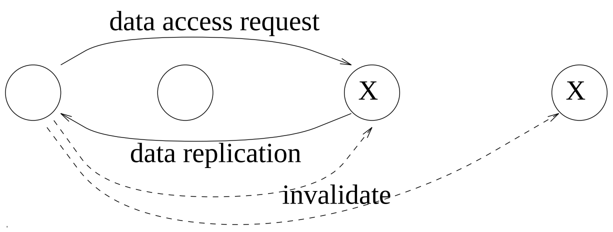
2.3.3 The Read-Replication Algorithm (Fig 6-25(c))

1. extends the migration alg.
   mutiple nodes to have read access.
   one node to have read-wirte access

2. improve performance for concurrent read.

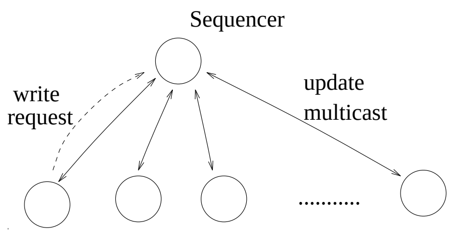
3. write is expensive.
   read copies must be invalidated and updated. DSM must keep track of all copies.

4. reduce avg cost of read
   when the ratio of reads and writes is large.



2.3.4 The Full-Replication Algorithm

1. extension of the read-replication alg.
2. multiple nodes with R/W access
3. access to shared data must be controlled.
   a gap-free sequencer.
   (a) send modified data to a sequencer.
   (b) the sequencer assigns unique sequence number and multicast the modification with the sequence number.
   (c) each node processes the mod.
   requests in the sequence number order.
   (d) if there is a gap, ask the sequencer to resend.



2.4 Memory Coherence

replicated shared data and concurrent access at many nodes.
Informally, a memory is coherent if the value returned by a read operation is always the value that the programmer expects. E.g. read most recently written value.

2.5 Intuitive Semantics

Strict Consistency – a read returns the most recently written value
$\Longrightarrow$ total ordering of requests.
$\Longrightarrow$ inefficient.

2.6 Relaxed Coherence Semantics

1. Strict Consistency – Any read to a memory location `x` returns the value stored by the most recent write operation to `x`.
2. Sequential Consistency – The result of any execution is the same as if the operations of all processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.

3. Causal Consistency – Writes that are potentially causally related must be seen by all processes in the same order. Concurrent writes may be seen iin a different order on different machines.
4. PRAM Consistency and Processor Consistency – Writes done by a single process are received by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.
5. Weak Consistency
   (a) Accesses to synchronization variables are sequentially consistent.
   (b) No access to a synchronization variable is allowed to be performed until all previous writes have completed everywhere.
   (c) No data access (read or write) is allowed to be performed until all previous accesses to synchronization variables have been performed.
6. Release Consistency
   (a) Before an ordinary access to a shared variable is performed, all previous acquires done by the process must have completed successfully.
   (b) Before a release is allowed to be performed, all previous reads and writes done by the process must have completed.
   (c) The acquire and release accesses must be processor consistent (sequential consistency is not required).
7. Entry Consistency
   (a) An acquire access of a synchronization variable is not allowed to perform with respect to a process until all updates to the guarded shared data have been performed with respect to that process.
   (b) Before an exclusive mode access to a synchronization variable by a process is allowed to perform with respect to that process, no other process may hold the synchronization variable, not even in nonexclusive mode.
   (c) After an exclusive mode access to a synchronization variable has been performed, any other process' next nonexclusive mode access to that synchronization variable may not be performed until it has performed with respect to that variable's owner.

2.7 Coherence Protocols

for concurrent access, replicated data need to ensure that replicas are coherent. (i.e. nodes do not access stale data)

2.7.1 Write-Invalidate Protocol

- invalidate alll copies then write.
- need to send inval. msgs. to all the nodes that have copies even if they may no longer use data.
- better if several updates between reads or high-degree of per-node locality of reference.
- inefficient if many nodes frequently access an object
- most DSM systems use this (IVY, CAPNET...)

2.7.2 Write-Update

- a write to a shared data causes all copies of that data to be updated.
- more network traffic overhead.

[Example] PLUS system

- write-update protocol
- general consistency
- the unit of replication is page (4K bytes)
  the unit of mem. access and coherence maintenance is one (32-bit) word.
- a list of replicas of a (virtual) page
  a designated master copy
- copy-list

Read op. – if local memory, the local memory is read. otherwise, the MCM (Memory Coherence Manager) send and req. to its counter part at the specified remote node.
Write op. – write always performed first on the master copy and then propagated to the copies linked by the copy-list.
Unifying Synch. and Data Transfer in Clouds. Fig 10.7.

2.7.3 The Dynamic Dist. Mang. Scheme

Every host keeps track of the ownership of the pages that are in its local page table. When a page fault, it sends a page req. to the processor (say i) indicated by the probowner field. If processor i is the true owner, O.K. otherwise, forward the request.

2.8 Design Issues

- Granularity
  large page size – improve locality. less network overhead.
  small page size – less false sharing
- Page Replacement
  LRU (Least Recently Used) not ideal
  favor remote page over local page

IVY (Integrated Shared Virtual Memory at Yale)

- 1K byte page
- address space – shared virtual memory address space and private space
- the coherence protocol.
  1. multiple-readers-single writer
     a reader always sees the latest value written. i.e. strict consistency
  2. write-invalidation protocol.
  3. when a process i has a write fault to a page p:
     (a) process i finds a owner of page p
     (b) the owner of p sends the page and its copyset to i
     (c) processor i sends out invalidation msgs to processors in the copyset.
  4. when a processor i has a read fault to a page p:
     - processor i finds the owner of page p
     - the owner of p sends a read-only copy of p to i and add i to the copyset of p.

2.9 Manager Schemes

2.9.1 The Centralized Manager Scheme

maintains all data ownership infor.
write req. to the manager.
the manager forwards the req. to the owner.
2 msgs to locate the owner of a page.
manager bottleneck

2.9.2 The Distributed Manager Scheme

Each manager keeps track of the owners of a predetermined set of pages.