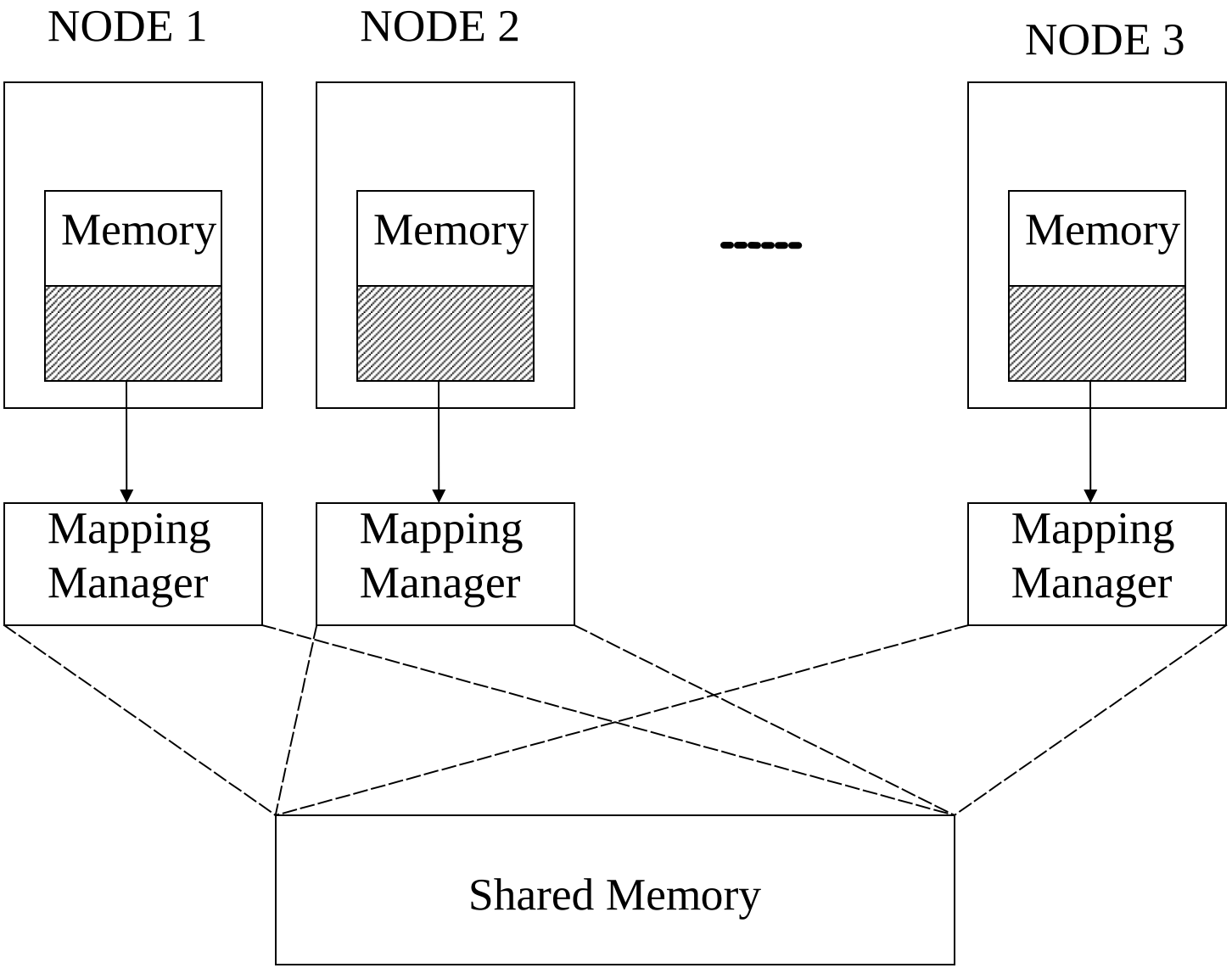
Distributed Resource Management: Distributed Shared Memory

Distributed shared memory (DSM)

- What
  - The distributed shared memory (DSM) implements the shared memory model in distributed systems, which have no physical shared memory
  - The shared memory model provides a virtual address space shared between all nodes
  - The overcome the high cost of communication in distributed systems, DSM systems move data to the location of access
- How:
  - Data moves between main memory and secondary memory (within a node) and between main memories of different nodes
  - Each data object is owned by a node
    - Initial owner is the node that created object
    - Ownership can change as object moves from node to node
  - When a process accesses data in the shared address space, the mapping manager maps shared memory address to physical memory (local or remote)

Distributed shared memory (Cont.)



Advantages of distributed shared memory (DSM)

- Data sharing is implicit, hiding data movement (as opposed to ‘Send’/‘Receive’ in message passing model)
- Passing data structures containing pointers is easier (in message passing model data moves between different address spaces)
- Moving entire object to user takes advantage of locality difference
- Less expensive to build than tightly coupled multiprocessor system: off-the-shelf hardware, no expensive interface to shared physical memory
- Very large total physical memory for all nodes: Large programs can run more efficiently
- No serial access to common bus for shared physical memory like in multiprocessor systems
- Programs written for shared memory multiprocessors can be run on DSM systems with minimum changes

Algorithms for implementing DSM

- Issues
  - How to keep track of the location of remote data
  - How to minimize communication overhead when accessing remote data
  - How to access concurrently remote data at several nodes

1. The Central Server Algorithm

- Central server maintains all shared data
  - Read request: returns data item
  - Write request: updates data and returns acknowledgement message
- Implementation
  - A timeout is used to resend a request if acknowledgment fails
  - Associated sequence numbers can be used to detect duplicate write requests
  - If an application's request to access shared data fails repeatedly, a failure condition is sent to the application
- Issues: performance and reliability
- Possible solutions
  - Partition shared data between several servers
  - Use a mapping function to distribute/locate data

Algorithms for implementing DSM (cont.)

2. The Migration Algorithm

- Operation
  - Ship (migrate) entire data object (page, block) containing data item to requesting location
  - Allow only one node to access a shared data at a time
- Advantages
  - Takes advantage of the locality of reference
  - DSM can be integrated with VM at each node
    - Make DSM page multiple of VM page size
    - A locally held shared memory can be mapped into the VM page address space
    - If page not local, fault-handler migrates page and removes it from address space at remote node
- To locate a remote data object:
  - Use a location server
  - Maintain hints at each node
  - Broadcast query
- Issues
  - Only one node can access a data object at a time
  - Thrashing can occur: to minimize it, set minimum time data object resides at a node

Algorithms for implementing DSM (cont.)

3. The Read-Replication Algorithm

- Replicates data objects to multiple nodes
- DSM keeps track of location of data objects
- Multiple nodes can have read access or one node write access (multiple readers-one writer protocol)
- After a write, all copies are invalidated or updated
- DSM has to keep track of locations of all copies of data objects. Examples of implementations:
  - IVY: owner node of data object knows all nodes that have copies
  - PLUS: distributed linked-list tracks all nodes that have copies
- Advantage
  - The read-replication can lead to substantial performance improvements if the ratio of reads to writes is large

Algorithms for implementing DSM (cont.)

4. The Full–Replication Algorithm

- Extension of read-replication algorithm: multiple nodes can read and multiple nodes can write (multiple-readers, multiple-writers protocol)
- Issue: consistency of data for multiple writers
- Solution: use of gap-free sequencer
  - All writes sent to sequencer
  - Sequencer assigns sequence number and sends write request to all sites that have copies
  - Each node performs writes according to sequence numbers
  - A gap in sequence numbers indicates a missing write request: node asks for retransmission of missing write requests

Memory coherence

- DSM are based on
  - Replicated shared data objects
  - Concurrent access of data objects at many nodes
- Coherent memory: when value returned by read operation is the expected value (e.g., value of most recent write)
- Mechanism that control/synchronizes accesses is needed to maintain memory coherence
- Sequential consistency: A system is sequentially consistent if
  - The result of any execution of operations of all processors is the same as if they were executed in sequential order, and
  - The operations of each processor appear in this sequence in the order specified by its program
- General consistency:
  - All copies of a memory location (replicas) eventually contain same data when all writes issued by every processor have completed

Memory coherence (Cont.)

- Processor consistency:
  - Operations issued by a processor are performed in the order they are issued
  - Operations issued by several processors may not be performed in the same order (e.g. simultaneous reads of same location by different processors may yields different results)
- Weak consistency:
  - Memory is consistent only (immediately) after a synchronization operation
  - A regular data access can be performed only after all previous synchronization accesses have completed
- Release consistency:
  - Further relaxation of weak consistency
  - Synchronization operations must be consistent with each other only within a processor
  - Synchronization operations: Acquire (i.e. lock), Release (i.e. unlock)
  - Sequence: Acquire

  Regular access

  Release

Coherence Protocols

- Issues
  - How do we ensure that all replicas have the same information
  - How do we ensure that nodes do not access stale data

1. Write-invalidate protocol

- A write to shared data invalidates all copies except one before write executes
- Invalidated copies are no longer accessible
- Advantage: good performance for
  - Many updates between reads
  - Per node locality of reference
- Disadvantage
  - Invalidations sent to all nodes that have copies
  - Inefficient if many nodes access same object
- Examples: most DSM systems: IVY, Clouds, Dash, Memnet, Mermaid, and Mirage

2. Write-update protocol

- A write to shared data causes all copies to be updated (new value sent, instead of validation)
- More difficult to implement

Design issues

- Granularity: size of shared memory unit
  - If DSM page size is a multiple of the local virtual memory (VM) management page size (supported by hardware), then DSM can be integrated with VM, i.e. use the VM page handling
  - Advantages vs. disadvantages of using a large page size:
    - (+) Exploit locality of reference
    - (+) Less overhead in page transport
    - (-) More contention for page by many processes
  - Advantages vs. disadvantages of using a small page size
    - (+) Less contention
    - (+) Less false sharing (page contains two items, not shared but needed by two processes)
    - (-) More page traffic
  - Examples
    - PLUS: page size 4 Kbytes, unit of memory access is 32-bit word
    - Clouds, Munin: object is unit of shared data structure

Design issues (cont.)

- Page replacement
  - Replacement algorithm (e.g. LRU) must take into account page access modes: shared, private, read-only, writable
  - Example: LRU with access modes
    - Private (local) pages to be replaced before shared ones
    - Private pages swapped to disk
    - Shared pages sent over network to owner
    - Read-only pages may be discarded (owners have a copy)

Case studies: **IVY**

- **IVY** (**I**ntegrated shared **V**irtual memory at **Y**ale) implemented in Apollo DOMAIN environment, i.e. Apollo workstations on a token ring
- **Granularity**: 1 Kbyte page
- **Process address space**: private space + shared VM space
  - Private space: local to process
  - Shared space: can be accessed by any process through the shared part of its address space
- **Node mapping manager**: does mapping between local memory of that node and the shared virtual memory space
- **Memory access operation**
  - On page fault, block process
  - If page local, fetch from secondary memory
  - If not local, request a remote memory access, acquire page
- Page now available to all processes at the node

Case studies: IVY (Cont.)

- **Coherence protocol**
  - Page access modes: read only, write, nil (invalidate)
  - Multiple readers-single writer semantics
  - Protocol
    - Write invalidation: before a write to a page is allowed, all other read-only copies are invalidated
    - Strict consistency: a reader always sees the latest value written
- **Write sequence**
  - Processor ‘i’ has write fault to page ‘p’
  - Processor ‘i’ finds owner of page ‘p’ and sends request
  - Owner of ‘p’ sends page and its copyset to ‘i’ and marks ‘p’ entry in its page table ‘nil’ (copyset = list of processors containing read-only copy of page)
  - Processor ‘i’ sends invalidation messages to all processors in copyset
- **Read sequence**
  - Processor ‘i’ has read fault to page ‘p’
  - Processor ‘i’ finds owner of page ‘p’
  - Owner of ‘p’ sends copy of page to ‘i’ and adds ‘i’ to copyset of ‘p’. Processor ‘i’ has read-only access to ‘p’

Case studies: IVY (Cont.)

Algorithms used for implementing actions for ‘Read’ and ‘Write’ actions

- **Centralized manager scheme**
  - Central manager resides on single processor: maintains all data ownership information
  - On page fault, processor ‘i’ requests copy of page from central manager
  - Central manager sends request to page owner. If ‘Write’ requested, updates owner information to indicate ‘i’ is the new owner
  - Owner sends copy of page to processor ‘i’ and
    - If ‘Write’, also sends copyset of page
    - If ‘Read’, adds ‘i’ to the copyset of page
  - On write, central manager sends invalidation messages to all processors in copyset
  - Performance issues
    - Two messages are required to locate page owner
    - On ‘Writes’, invalidation messages are sent to all processors in copyset
    - Centralized manager can become bottleneck

Case studies: IVY (Cont.)

Algorithms used for implementing actions for ‘Read’ and ‘Write’ actions (cont.)

- **The fixed distributed manager scheme**
  - Distributes the central manager’s role to every processor in the system
  - Every processor keeps track of the owners of a predetermined set of pages (determined by a mapping function *H*)
  - When a processor ‘i’ faults on page ‘p’, processor ‘i’ contacts processor *H(p)* for a copy of the page
  - The rest the protocol is the same as the one with the centralized manager

Note: In both the centralized and fixed distributed manager schemes, if two or more concurrent accesses to the same page are requested, the requests are serialized by the manager

Case studies: IVY (Cont.)

Algorithms used for implementing actions for ‘Read’ and ‘Write’ actions (cont.)

- **The dynamic distributed manager scheme**
  - Every host keeps track of the ownership of the pages that are in its local page table
    - Every page table has a field called *probowner* (probable owner)
    - Initially, *probowner* is set to a default processor
    - The field is modified as pages are requested from various processors
  - When a processor has a page fault, it sends a page request to processor ‘i’ indicated by the *probowner* field
  - If processor ‘i’ is the true owner of the page, fault handling proceeds like in centralized scheme
  - If ‘I’ is not the owner, it forwards the request to the processor indicated in its *probowner* field
  - This continues until the true owner of the page is found

Case studies: **Mirage**

- Developed at UCLA, kernel modified to support DSM operation
- Extends the coherence protocol of IVY system to control thrashing (in IVY, a page can move back and forth between multiple processors sharing the page)
- When a shared memory page is transferred to a processor, that processor will keep the page for ‘delta’ seconds
  - If a request for the page is made before ‘delta’ seconds expired, processor informs control manager of the amount of time left
  - ‘Delta’ can be a combination of real-time and service-time for that processor
- Advantages
  - Benefits locality of reference
  - Decreases thrashing

Case studies: **Clouds**

- Developed at Georgia Institute of Technology
- The virtual address space of all objects is viewed as a global distributed shared memory
  - The objects are composed of segments which are mapped into virtual memory by the kernel using the memory management hardware
  - A segment is a multiple of the physical page size
- For remote object invocations, the DSM mechanism transfers the required segments to the requesting host
  - On a segment fault, a *location system object* is consulted to locate the object
  - The *location system object* broadcasts a query for each locate operation
  - The actual data transfer is done by the distributed shared memory controller (DSMC)