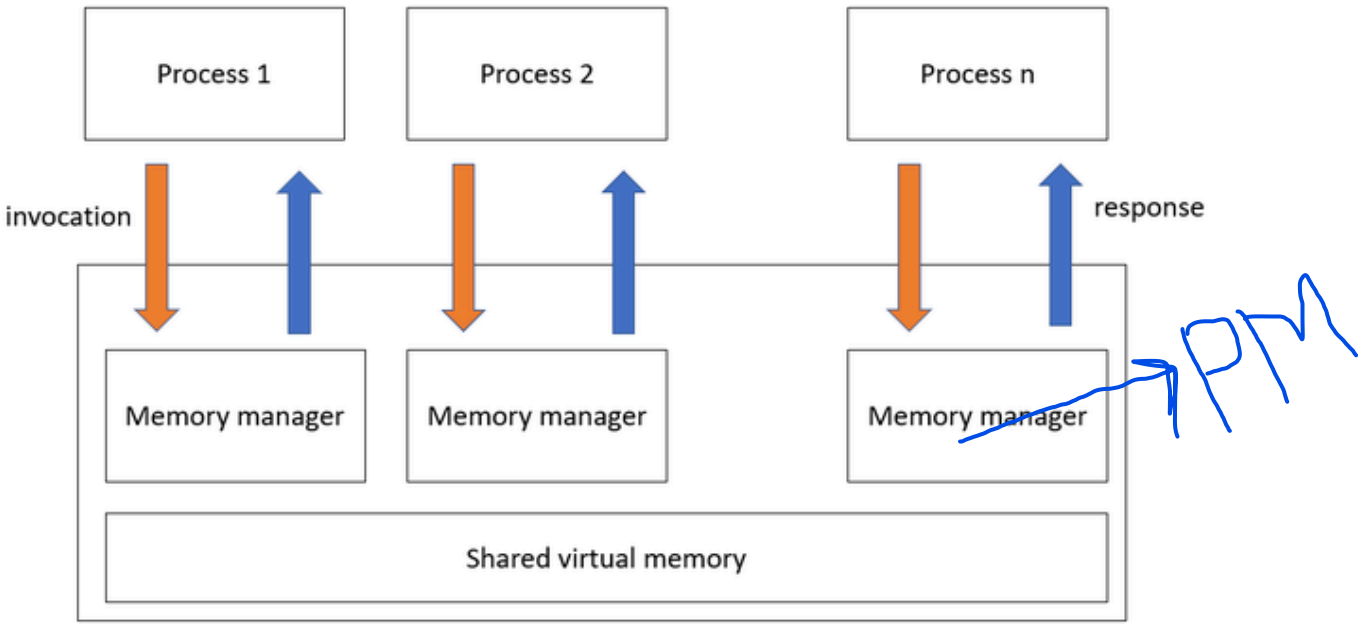
What is Distributed Shared Memory?

DSM is a system that allows multiple computers (nodes) to share memory as if they were using a single shared memory space. It helps different programs communicate without needing direct messaging. Even though each node has its own physical memory, DSM creates a virtual shared memory across all nodes.

How DSM Works

- Each node provides memory read and write services.
- A consistency protocol ensures that all nodes see updated data.
- Data moves between the memory of different nodes as needed.



Types of Distributed Shared Memory

Based on Granularity:

Granularity refers to the size of memory blocks that can be shared.

- **Fine-grained DSM**: Shares small memory blocks (e.g., individual variables or small objects).
- **Coarse-grained DSM**: Shares larger memory blocks (e.g., pages or segments).

Based on Implementation:

The implementation mechanism determines how shared memory is provided.

- **Hardware-based DSM**: Uses specialized hardware to provide a shared memory abstraction (e.g., NUMA systems).
- **Software-based DSM**: Implements DSM in software, typically using page-based memory management (e.g., TreadMarks).

Based on Data Distribution and Access

- **Page-based DSM**: Memory is divided into pages, and pages are moved between nodes as needed.
- **Object-based DSM**: Shared memory is managed as objects rather than pages.
- **Variable-based DSM**: Allows sharing of individual variables instead of entire pages or objects.

Based on Replication and Update Policies

- **Write-invalidate DSM**: When a process writes to a shared memory location, other copies of that data are invalidated.
- **Write-update DSM**: Updates are propagated to all copies of the shared data.

Advantages of Distributed Shared Memory

- **Simpler Abstraction:** Programmers need not concern about data movement, as the address space is the same it is easier to implement than RPC.
- **Easier Portability:** The access protocols used in DSM allow for a natural transition from sequential to distributed systems. DSM programs are portable as they use a common programming interface.
- **Locality of Data:** Data moved in large blocks i.e. data near to the current memory location that is being fetched, may be needed in the future so it will be also fetched.
- **On-Demand Data Movement:** It provided by DSM will eliminate the data exchange phase.

- **Larger Memory Space:** It provides large virtual memory space, the total memory size is the sum of the memory size of all the nodes, paging activities are reduced.
- **Better Performance:** DSM improves performance and efficiency by speeding up access to data.
- **Flexible Communication Environment:** They can join and leave DSM system without affecting the others as there is no need for sender and receiver to existing,
- **Process Migration Simplified:** They all share the address space so one process can easily be moved to a different machine.

Disadvantages of Distributed Shared Memory

- **Accessibility:** The data access is slow in DSM as compared to non-distributed.
- **Consistency**: When programming is done in DSM systems, programmers need to maintain consistency.
- **Message Passing:** DSM uses asynchronous message passing and is not efficient as per other message passing implementation.
- **Data Redundancy:** DSM allows simultaneous access to data, consistency and data redundancy is a common disadvantage.
- **Lower Performance:** CPU gets slowed down, even cache memory does not aid the situation.

Issues to Design and Implementation of DSM:

- Granularity
- Structure of shared memory space
- Memory coherence and access synchronization
- Data location and access
- Replacement strategy
- Thrashing
- Heterogeneity

1. Granularity: Granularity refers to the block size of a DSM system. Granularity refers to the unit of sharing and the unit of data moving across the network when a network block shortcoming then we can utilize the estimation of the block size as words/phrases. The block size might be different for the various networks.

2.Structure of shared memory space: Structure refers to the design of the shared data in the memory. The structure of the shared memory space of a DSM system is regularly dependent on the sort of applications that the DSM system is intended to support.

3. Memory coherence and access synchronization: In the DSM system the shared data things ought to be accessible by different nodes simultaneously in the network. The fundamental issue in this system is data irregularity. The data irregularity might be raised by synchronous access. To solve this problem in the DSM system we need to utilize some synchronization primitives, semaphores, event count, and so on.

4. **Finding and Accessing Data:** In a DSM (Distributed Shared Memory) system, data needs to be easy to find and retrieve when accessed by users or programs. To make this possible, the system must have a way to locate and manage data blocks efficiently. This ensures that data can be shared across the network while following the rules that keep memory consistent.

5. **Replacing Data in Memory:** If a node's local memory is full and it needs new data from another node, it must remove some existing data to make space. This means the system must decide which data to replace. Having a smart strategy for replacing old data is important when designing a DSM (Distributed Shared Memory) system.

6. **Thrashing:** In a DSM (Distributed Shared Memory) system, data blocks move between nodes when needed. If two nodes keep trying to write to the same data, the data may keep moving back and forth too quickly, slowing everything down. This problem is called **thrashing**. The system should have a way to prevent this so work can continue smoothly.

7. Heterogeneity: The DSM system worked for homogeneous systems and need not address the heterogeneity issue. In any case, assuming the underlined system environment is heterogeneous, the DSM system should be designed to deal with heterogeneity, so it works appropriately with machines having different architectures.