Algorithm for implementing Distributed Shared Memory

Shared memory is the memory block that can be accessed by more than one program. A shared memory concept is used to provide a way of communication and provide less redundant memory management.

Distributed Shared Memory abbreviated as **DSM** is the implementation of shared memory concept in distributed systems. The DSM system implements the shared memory models in loosely coupled systems that are deprived of a local physical shared memory in the system. In this type of system distributed shared memory provides a virtual memory space that is accessible by all the system (also known as **nodes**) of the distributed hierarchy.

Some common challenges that are to be kept in mind while the implementation of DSM −

- Tracking of the memory address (location) of data stored remotely in shared memory.
- To reduce the communication delays and high overhead associated with the references to remote data.
- Controlling the concurrent access of the data shared in DSM.

Based on these challenges there are algorithms designed to implement distributed shared memory. There are four algorithms −

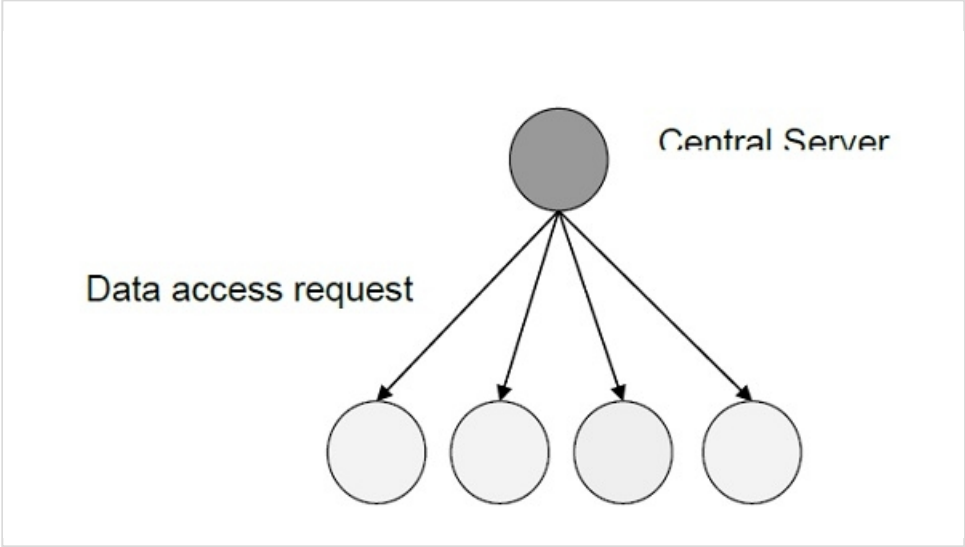
- **Central Server Algorithm**
- **Migration Algorithm**
- **Read Replication Algorithm**
- **Full Replication Algorithm**

Central Server Algorithm

All shared data is **maintained by the central server**. Other nodes of the distributed system **request for reading and writing data** to the server which serves the request and updates or provides access to the data along with **acknowledgment messages**.

These acknowledgment messages are used to provide the status of the data request is served by the server. When the data is sent to the calling function, it acknowledges a number that shows the access sequence of the data to maintain concurrency. And time-out is returned in case of failure.

For larger distributed systems, there can be more than one server. In this case, the servers are located using their address or using mapping functions.

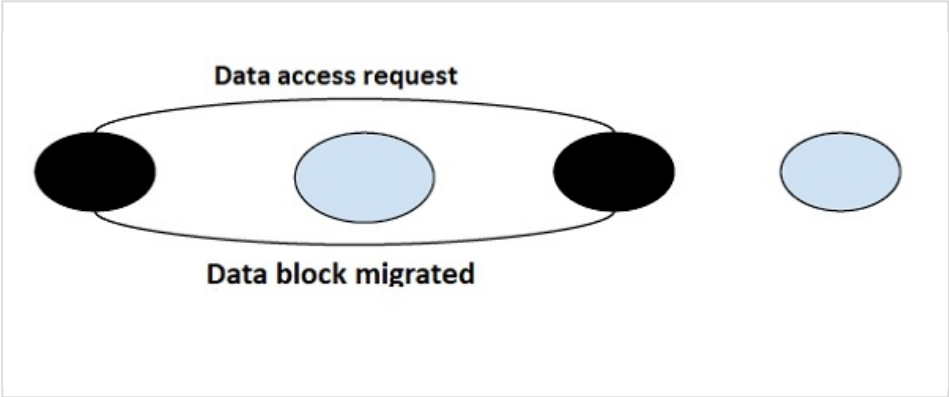


Migration Algorithm

As the name suggest the migration algorithm does the work of migration of data elements. Instead of using a central server serving each request, the **block containing the data requested by a system is migrated** to it for further access and processing. It migrates the data on request.

This algorithm though is good if when a system accesses the same block of data multiple times and the **ability to integrate virtual memory** concept, has some shortcomings that are needed to be addressed.

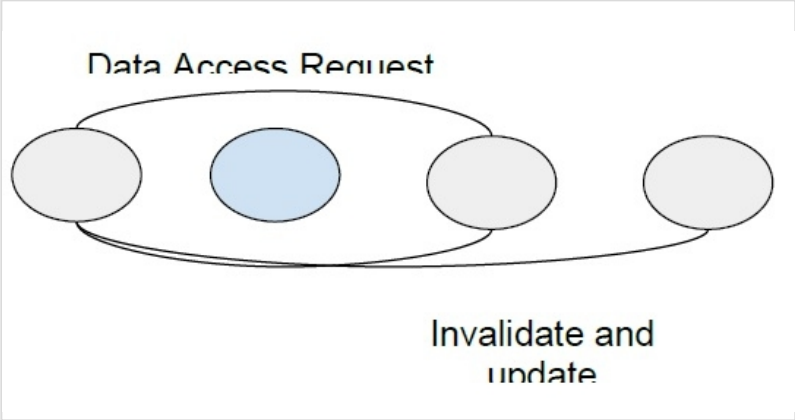
Only one node is able to access the shared data element at a time and the whole block is migrated to that node. Also, this algorithm is more **prone to thrashing** due to the migration of data items upon request by the node.



Read Replication Algorithm

In the read replication algorithm, the data block that is to be accessed is **replicated** and only **reading is allowed** in all the copies. If a write operation is to be done, then all read access is put on halt till all the copies are updated.

Overall system performance is improved as **concurrent access is allowed**. But write **operation is expensive** due to the requirement of updating all blocks that are shared to maintain concurrency. All copies of data element are to be tracked to maintain consistency.



Full Replication Algorithm

An extension to read the replication algorithm allowing the nodes to perform both **read and write** operation on the shared block of concurrently. But this access of nodes is controlled to maintain its consistency.

To maintain consistency of data on concurrent access of all nodes sequence is maintained and after every modification that is made in the **data a multicast** with modifications is reflected all the data copies.