CHAPTER 5

MEMORY CONSISTENCY MAINTENANCE

Memory consistency model is a trade-off between minimizing access order constraints and the complexity of the programming model. According to the programmer, the memory consistency enables the correct reasoning about the memory operations in a program. According to the system designer, the memory consistency specifies acceptable memory behaviors for the system. The memory consistency not only influences many aspects of system design but also influence the design of programming languages, compilers, and the underlying hardware[3, 11, 13]. In order to enhance performance, multiprocessors tend to implement sophisticated memory structures. These memories may replicate data through constructs such as caches and write buffers. The replication of data means reading/writing of the same data that are executed in parallel. But this activity produces inconsistent result when write operations are executed [1, 2, 6].

Any of the architectural features allow processes to have inconsistent views of memory, which, in turn, can result in unexpected program outcomes so it needs to maintain the consistency of memory in proper way. For maintaining memory consistency memory coherency is required. The coherency protocol ensures the serialization of write
operation and subsequent reads or writes access to update data. These updates are propagated by the remote site. The coherency protocol takes these update and incorporate these updates to local copies.

The coherency protocol is mainly used to keep multiple copies of on cache block consistent to all processors, while the role of memory consistency is specifying how memory behaves with respect to read and write operations for multiple processors. The implementation of memory consistency is different in different DSM architecture. In hardware DSM system memory consistency model emphasis on relaxing the restriction of event ordering, while in software DSM system memory consistency model focus on relaxing on coherence protocol [2, 5, 12]. A coherency scheme defines what values can be returned by a read operation. It also tells that event ordering in each processor, determines when a written value will be returned by a read operation.

The coherency protocol doesn’t tell us the event ordering in each processor; it is difficult to determine that the written value is seen by the other processor [4, 8]. For making the behaviors of memory system accurate it is needed to restrict both coherence and event ordering. This is done by the memory consistency model. The condition of coherence is a subset of the conditions of memory consistency model. Coherency considers the different events from different processors to the same location only, while consistency model not only considers the different events to the same location but also imposes constraints on the ordering of event within each processor, i.e. the execution order in each processor [7, 9]. So we can say by combining the coherency and event ordering, memory consistency model is evolved.
Memory Consistency Model = Coherency Protocol + Event Ordering

The memory consistency gives the proper behavior of memory system. So for the proper behavior following two things are required:

1. All memory access appears to execute automatically in some total order.
2. All memory accesses of each processor appear to execute in an order specified by its programmer.

The first condition is ensured by the coherence protocol. It ensures either that all processors which keep copies will see the new value simultaneously with modified processor or invalidate all other copies if there is any write operation. The second condition is done by the executing order of events. It is done by the restricting all the events in one processor to be issued, performed in program order. But this activity may affect the performance of the system. So it is become necessary to maintain this condition. For this we have designed a framework with the use of coherency protocol which is discussed in next section.

5.1 Framework to Maintain Memory Consistency

The approach taken for framework for maintaining the consistency is the relationship between memory consistency and memory coherency protocol for DSM system. The basic relation between memory consistency and memory coherency is shown in the Figure 5.1. The read, write and synchronization operation defines the relationship between
consistency and coherency. The memory consistency in this relation defines the level of consistency provided by the system while the memory coherency is responsible for managing data so that the required level of consistency is actually provided.

**Figure 5.1** Relationship between consistency and coherency

### 5.1.1 Issue Involved

For any system there should be five issues transparency, flexibility, reliability, performance and the scalable must satisfy. So the proposed framework also satisfies these properties.

1. **Transparency:** It is transparent and it has the single system image view. It works in the different transparency concepts, with *Location Transparency*: all the processors know where the most recent value is. For *Migration Transparency*: the processor will not feel the existence of other processors in the system using the same shared object if it has read only access, but it will feel it if it have write access to the shared object. Finally, *Parallelism Transparency*: it is achieved in read & writes both operations.

2. **Flexibility:** In general it is flexible, since the micro kernel is used for interposes communication and helps memory management.

3. **Reliability:** It considered being reliable for the availability concept, since the fraction of time the system is used in asking for the
counter value and receiving the answers is not too large if the system contains a small number of machines. For the Security concept, no other machine can access the shared object if it does not have authorization.

4. **Performance:** It shows performance in general with small number of machines in the system, but it may need a big bandwidth if the system has a large number of machines. If more than one machine asks for the shared object at the same time, the performance may become lower. Generally, in this model consistency is achieved over the performance.

5. **Scalability:** It is scalable for a small system i.e. the system where the importance is given to memory.

5.1.2 **Basic Concept**

The basic concept kept in mind to design the framework is the cost of consistency maintenance. The cost of consistency is taken to be minimal in order to achieve a good DSM performance. The consistency maintenance concept differentiates between a requesting site and holding site [10, 14]. The proposed framework is based on one-to-many relationship to inform all sites that are involved in a DSM application about a DSM data movement to request site. The proposed DSM framework is designed on the base of centralized consistency maintenance approach, where a requesting site will be updated about the holding site through centralized knowledge site. The consistency maintenance deals with replication to handle read access faults. The knowledge site possesses a copy of the requested page. The required
data could be copied to the requesting site without involving the third site. In this framework the processor-exclusive mapping of application and consistency maintenance sites can take place. The framework handled the access fault without interrupting a second application site. The framework is best suitable to handled access faults when data replication is used.

The proposed framework mainly concentrates on write operations. So the consistency maintenance mainly concern about controlling the write access fault and parallelism. The controlling of write access fault is done by centralized consistency maintenance. In this scheme the required page is copied to requesting site as well as knowledge site. By this the data replication is done.

5.2 Designing Issue of Framework

There are four components are used in the framework Memory consistency, Coherency Protocol, Concurrency Control and
Communication Service, which are shown in Figure 5.3. The framework is designed on the basis of the relationship between memory consistency and memory coherency as shown in the figure 5.1.

The read/write synchronization is done by the memory consistency. It controls the order of read and writes operations. The concurrency control supervises the concurrent accessing. It controls the multiple write operations which are being executed simultaneously. The coherency protocol specifies the memory location to read/write operations. This is done by updating all the read operations which is to be occurring after a write operation. The communication between coherency protocol and concurrency protocol is done by the communication service. It supervises all the communication done between these components.

### 5.2.1 Memory Consistency
It is a tradeoff between minimizing memory access order constraints and the complexity of the programming model as well as the complexity of the memory model itself. Memory consistency imposes ordering restriction on access depending on a number of attributes. These attributes are discussed here as the issues involved in memory consistency and on this basis the framework is designed. The issues involved are divided into two parts, one which is based on the basis of access and the other on the basis of properties of consistency. The accesses issues depend upon the following:

- Location of access
- Direction of access (read, write or both)
- Value transmitted in access
- Causality of access
- Category of access

The location, direction and value transmitted in access meant that what value is read or written or both on which address. For example direction of accesses which are memory read operations are of the form “R(a,d)” and memory write operations are of the form “W(a,d)”. The location (address) of the operation is represented by ‘a’ and ‘d’ is the value (data) read/written by the operation. Both read and write operations are represented by “RW(a, dr, dw)” where dr is the data read and dw is the data written.

The causality issue defines the relation that if two accesses are causally related so which access will occur first. The category
access defines the static property of accesses. The memory access may be shared or private. Private access is very easy to deal as it deals with one processor with its own memory. The shared access is difficult to implement as it deals with shared memory so may further be divided into competing and noncompeting access. A pair of access is said to be competing if they access the same location, at least one of them is a write access and not in order. A competing access is again be either synchronizing or nonsynchronizing. A synchronizing access enforces order. The location, direction and value access are used in every system. The problem arises for using the category of access. As we are concentrating on the memory consistency for DSM system which combines the best feature of shared memory as well as distributed memory and requiring the shard access. So we have taken shared/competing/synchronization category access issues for our framework. The memory consistency for DSM system defines the order of access with giving concentration to the concurrent access. So we have concentrated on mainly four properties of memory consistency i.e. Order of access, Concurrency, Scope and Atomicity. The order of access defines the sequence in which access are seen by the different processor. Concurrency of access defines if nodes can concurrently access data and the modes in which they can access it. Scope determines the set of data that is to be kept consistent. Atomicity defines whether the propagation of updates is done on a per access basis. The properties are for describing the implementation of memory consistency and the connection with the coherence.

5.2.2 Coherence Protocol
The coherency protocol works as interface between communication service and the memory consistency model. The core of the framework is the coherency protocol because it finds what update is actually required and what are the destination of updates. Coherency protocol is used in the framework to maintain memory consistency for ensuring the serialization of write operation and that any subsequent reads or writes access the update data. The coherency protocol described in the framework is based on three component memory management, synchronization and ownership management as shown in the Fig. 5.4.

![Coherency Protocol Diagram](image)

**Figure 5.4** Memory Coherence Protocol (MCP)

### 5.2.2.1 Memory Management

The memory management component is responsible for the management of local copy of data items and it performs two major operation; the processing of access operation and the handling of updates. The access operations comprise of read and write operation to the local copy. The synchronization is provided by connected consistency model. Write operation comprises the memory management component to update a location of local copy with a given value. Two concept is used for the proper memory
management; the first one defines what values can be returned by read operation and the second concept defines the event ordering in each processor. Memory management component ensures that multiple processors see a coherent view of the same location. for example if there are multiple write in the system then the memory management component will select those writes operation among all the writes which can be accessed by more than one processor. Now the memory management component with help of memory consistency finds the destinations of all selected writes.

5.2.2.2 Synchronization

The synchronization component is used as distribution component interchangeably. The synchronization component is responsible of exchange of data between copies. It is linked closely to the communication mechanism. The synchronization component provides method to send and request update from remote nodes. The method send update takes as a parameter an update and set the destination. The updates are sent to the address of the given destination. The event ordering in a processor describes that when other processors see a value that has been updated by this processors.

5.2.2.3 Ownership Management

The ownerships management component is for the proper administration of ownership of copies. The main activity of ownership component is to find the location of other copies shared data. For enabling this service it provides the following methods:
1. A method to announce that a local copy is being shared.

2. A method to determine the source of update.

3. A method to determine the destination of update.

The actual definition of ownership management is defined by the activity of synchronization component and communication services.

5.2.3 Communication Service

The communication service is used for processor communication, message passing making input and output. It provides the communication between processors, between memories, between input/output (I/O). Apart from this the communication between processor-memory, I/O-memory and I/O-processor is done through this component. The communication process completes in following three steps:

1. **Connection**: Before data exchange between two nodes, the connection should be established between these two nodes.

2. **Data Exchange**: After connection establishment, the two communicating entities can then exchange their data in any direction.

3. **Disconnection**: After the data exchange one of the communicating entities may issue a disconnection call and disconnect the communicating path.
For the proper communication it requires two messages, a request and a reply, and that a reliable communication protocol is used so that once a processor sends a request, it eventually receives a reply. Here the communication service component sends the message only to those processors which are accessing the location of any particular write operation. So instead of broadcast communication anycast communication method is used.

5.2.4 Concurrency Control

Concurrency control component of the framework ensures that correct results for concurrent operations are generated. It works as an interface between memory consistency and communication services. It is designed on the basis of consistency as well as coherency rule. The concurrency controller controls the concurrent access by:

1. A read by a processor P, to a location X that follows a write by P to X, with no writes to X by another processor occurring between the write and the read by P, always returns the value written by P.

2. A read by a processor to location X that follows a write by another processor to X returns the written value if the read and write are sufficiently separated and no other writes to X occur between the two accesses.

3. Writes to the same location are serialized i.e. two writes to the same location by any two processors are seen in the same order by all processor.
All the above are done with the help of memory consistency and coherency and the concurrency in the framework is controlled.

5.3 Maintaining Consistency

The framework provides a general interface that allows the memory consistency model to communicate with coherency protocol. The core method is to update and ensure memory consistency. As we have taken write-update coherence protocol so main concern is on the update operation. An update is a collection of data and descriptor. A descriptor holds the references to the location in the local copies of shared data. References may describe the single location or memory address or set of locations. An update holds descriptor that point to various locations in local copies of shared data. In these locations the corresponding data from the update have to insert in order to update a local copy. The update operations here depend upon the protocol implemented by synchronization component.

The working of the framework is described in two parts. First part finds the destination of node to propagate and incorporate the updates and the second part propagates the updates to all nodes.

5.3.1 Finding the destination and incorporating updates

Propagating updates to all nodes the framework first find the actual updates and their destination. After getting the destination of all
updates the framework incorporates all updates to local copies of all nodes. This activity is described in the following steps.

**Step 1.** The method update distributed update propagation from remote site.

**Step 2.** **Finding the Actual Update** (Figure 5.5)

a) The synchronization (SYN) component takes these update propagation and incorporate these update and asks to memory management (MM) component what to update and what not to be update.

b) The memory management component takes these updates and verifies these updates by seeing what the update requests are. It also takes the help of read and write operation provided by the memory consistency model.

c) After the verification, the verified updates are returned to the synchronization components by the memory management components.

**Step 3.** **Finding the destination** (Figure 5.6)
a) Now the destinations of verified updates are required.

b) The synchronization component asks the ownership management about the destination of these updates.

c) The ownership management finds the destination of these updates by access event which is provided by the memory consistency model.

d) The destinations of updates are sent to the synchronization component.

![Diagram](image)

**Figure 5.6** Disbursement of updates between SYN and OM Component

### Step4. Incorporating updates to local copies

a) The updates are sent to its destination by the synchronization component.

b) Then source is then asked for update by using the request update of the synchronization component.

c) The return update is now incorporated into local copies by the memory management component.

### 5.3.2 Propagation of updates
After incorporating updates to local copies now write update coherence protocol propagates the updates to all nodes. The propagation of update is started with block’s master node and proceeds to copy-list chain of nodes. The copy-list chain of a node tells about the master of that particular node and the next copy of that node. The write request is done by the initiator and propagates to all nodes according to the copy list information. The write request is completed when the last node in the copy-list chain acknowledges about the completion of the updates to the initiator of the write request.

For example (fig.5.7.) let A, B, C, D are four nodes. A is the master of all node. Node D is initiator as it wants to write and C is last node as it doesn’t have any next copy. The propagation of updates in the said example completes in following steps:

1. MCP of node D sends write request to node B.
2. MCP of node B sends this update to its master node which is A.
3. As node A is the master node so MCP of node A updates its copy-list X.
4. Now MCP of node A sends this update message to its next copy i.e. node B.
5. MCP of node B updates its copy-list and sends this update message to its next copy i.e. node C.
6. MCP of node C updates its copy-list. As it contains no next copy so it is the last node. Here the propagation of update message completes.
7. MCP of node C sends an acknowledgment to initiator node (D) that updates is completed.

If the propagation of updates happens successfully it means that all node are now aware about the write operation initiated by any node. So all nodes except the initiator node invalidate its write operation and the consistency of memory is maintained.
REFERENCE


