Distributed Shared Memory: Concepts and Systems

Jelica Protić, Milo Tomašević and Veljko Milutinović

IEEE Parallel & Distributed Technology, Summer 1996

Context:
distributed memory management
high-performance computing

Multiprocessor systems and shared memory

1. Shared-memory system (multiprocessor)
2. Distributed-memory system (multicomputer)—communication costs more of an issue. “Hardware problems easier, software problems more complex.”

Distributed shared memory (DSM) combines the two concepts.

Research Focus of DSM systems

- general approaches
- minimize average access time
- maintain data consistency

DSM System Structure

States and current locations of data blocks maintained in the *directory*.

Classification

Research from the mid-80s.

Multiprocessor (scalability) and multicomputer (communication latency) issues converging??

3 Key Issues:

1. DSM algorithm
2. Implementation level of DSM mechanism
3. What is the memory consistency model

DSM Algorithms

Two basic problems:

1. data is distributed to minimize access latency
2. coherent view of data must be maintained without too much overhead.

Strategies:

1. Replication—make copies of the data (best for read sharing)
2. Migration—move copy of the data (best when single node access at a time)

Algorithms

1. Single reader/single writer. Little used, either a central server or migration of data copy used.
2. Multiple reader/single writer (read-replication).
   - Central manager (apart from owner)—maintains copy set
   - Improved centralized manager—copy set maintained by owner
   - Fixed distributed manager—each site manages a portion of data blocks
   - Broadcast distributed manager—broadcast to find owner of block (owner and manager the same)
   - Dynamic distributed manager—(Li's algorithm) Each copy of block caches probable owner (which may need to be forwarded to real owner).
3. Multiple reader/multiple writer (full-replication). Could use reliable multicast

Algorithm Improvements

- Add sequence number to Li's algorithm—only transfer page if it is known to be out-of-date.
- competitive algorithm for replication—only replicate when number of accesses warrants replication costs???
- introduce time window to hold page at a site—reduce thrashing.

Implementation Level of the DSM Mechanism

Where is *lookup* of data done and an *action* must be taken on write to preserve coherence of shared data.

1. Software
2. Hardware
3. Hybrid

Software-Level Implementations

Three levels:

1. user-level library
2. operating system
3. programming language

Often done through virtual memory and page-fault handler. Relatively large pages can cause problems with *false sharing* when there is fine-grain parallelism.

Hardware-Level Implementations

Automatic replication of shared data—transparent for applications. Can handle finer-grain sharing. More for high-end machines.

Hybrid-Level Implementations

"Extreme" approaches can still involve some amount of mixed implementation.

Memory Consistency Models

Stronger Consistency Models

Sequential Consistency—all processors see the same order of reads and writes (e.g. through a single queue of accesses).

Processor Consistency—all processors must observe writes in the same order

Weaker Consistency Models

Requires memory becomes consistent only on explicit synchronization accesses. Otherwise reads/writes can be processed as received between synchronization accesses.

Release Consistency—separates acquisition and release for synchronization access. Modifications are propagated on execution of release (after an acquisition has been made).

Lazy Release Consistency—Make updates when the next relevant acquire is made.

Entry Consistency—Finer level of variable protection for better performance, but more programmer involvement.

Summary

Most, if not all work, is in the research domain—not in the commercial domain.

Work to be done in the three key areas plus some additional issues.

Tradeoffs between effort for programmer versus implementation, performance versus consistency.