Two-Phase Conflict Detection for Transactional Memory on Clusters

Ruibo Wang, Kai Lu, Xicheng Lu
Computer School
National University of Defense Technology
Changsha, China
ruibo@nudt.edu.cn, kai_lu@263.net, xclu@nudt.edu.cn

1. BACKGROUND
Most TM research has focused on multi-core processors; limited research has been aimed at the clusters. Manassiev et al. described a TM designed for clusters [1], but their approach is based on software distributed shared memory (DSM). Cluster-STM [2] and DiSTM [3] do not rely on the DSM mechanism to achieve memory coherence. The essential differences are: a) Cluster-STM is word-based and DiSTM is object-based; b) Cluster-STM is designed for large-scale cluster and DiSTM mainly focus on small cluster.

The performance of the existing cluster's TM is very poor [4]. The conflict detection, which is the most frequent operation of TM, is highly depending on the memory access. But the cluster's remote memory access speed is very slow. The TM design derived from the multi-core processor is not suitable for the clusters. Now we introduce a hierarchical conflict detection strategy. The detection process was divided into two phases. By using different detection method with the two-phase, the hierarchical strategy provides a good practical performance on clusters.

2. DESIGN
A typical node of a cluster is an SMP machine with several multi-core processors. The nodes are connected with high-speed network. Inside a node, the memory access speed is fast. Remote memory access need network communication, which is much slower. We divide the conflict detection process into two phases.

2.1 Conflict Detection inside a node
We use visible reader and eager writer strategy. The detection is followed by every read/write operation. The conflicts will be detected as early as possible. This policy can benefit from the fast memory access inside a node. The committed transaction will enter the next phase.

2.2 Conflict Detection among the nodes
A transaction reaches here will be queued to do the global commitment. First, the transaction will ask for a global token. It means only one transaction can commit globally at a time. Secondly, the transaction will broadcast its writeset to the other nodes. All the data will be Bloom filtered in order to reduce the network communication. The other nodes will check if there are any conflicts. The conflicted transactions will abort and retry. After all the acknowledgements received, the transaction will update the data globally and return the token. This is a lazy policy like TCC [5].

3. EVALUATION
We run our experiments on a 4-nodes cluster. Each node has 2 processors and 4GB RAM. Each processor has two 1GHz x86-64 cores. Simics is used to do the simulation while the benchmark is STMBench7.

The figure shows the total throughput in operations per second with thread count ranging from 2 to 64. All the threads are evenly distributed on the nodes. The hierarchical strategy shows better performance and scalability when the thread count is greater than 4, which is exactly the node number of the cluster.

4. REFERENCES