A Brief Overview of Scalable Distributed Concurrency Services for Hierarchical Locking

Nirmit Desai and Frank Mueller
Department of Computer Science, North Carolina State University, 448 EGRC, Raleigh, NC 27695-7534
mueller@cs.ncsu.edu, phone: +1.919.515.7889, fax: +1.919.515.7925

ABSTRACT

As researchers share computational resources, such as objects in large distributed environments, it becomes difficult to achieve scalability of synchronization, and commonly used concurrency protocols currently lack scalability. Our peer-to-peer protocol enhances middleware concurrency services to provide scalability of synchronization enabling resource sharing and computing with distributed objects in systems with a large number of nodes.

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process Management—*mutual exclusion*

General Terms

Algorithms, Experimentation

Keywords

Hierarchical Locking, Distributed Mutual Exclusion

1. SUMMARY

Distributed computing is rapidly becoming a commodity to share resources, such as objects, on a large scale. Recent trends follow peer-to-peer computing paradigm with distributed objects supported by middleware to provide distributed services. One of the main challenges in such environments is to achieve scalability of synchronization. Our technical contribution is a novel, fully decentralized, hierarchical locking protocol to enhance concurrency in distributed resource allocation. The results of our work impact scalability for distributed computing facilities such as embedded computing with distributed objects over peer-to-peer computing environments, arbitrating accesses in very large database environments and distributed web servers.

Several exclusive locking schemes for distributed systems exist [3, 2]. However, in practice, systems generally require the distinction between lock modes to increase the potential for parallelism. Most systems distinguish read (R) locks and write (W) locks with parallel and exclusive access, respectively. An upgrade (U) lock represents an exclusive read lock that is followed by a request for promotion to a write lock. Upgrade locks avoid conflicts between threads that intend to modify (write) objects after having held a read lock.

Hierarchical locking schemes enhance parallelism and reduce locking overhead by distinguishing lock modes based on structural data representation. Entities at a higher level of granularity are associated with *intention* versions (IR and IW) of the aforementioned lock modes (R and W). For example, a node wishing to read an attribute of an object will request an intention read (IR) lock on the object itself and, once acquired, it will request a disjoint read (R) lock on the attribute it wants to read, without releasing IR. Lock requests may proceed in parallel if the modes requested are compatible. We refer to the compatibility matrix specified in [1], which is a de-facto standard hierarchical locking model. The compatibility rules are based on the fundamental principle that *a unit of data* cannot be both, read from and written to concurrently at either same or different sites.

Our protocol is based on prior work described in [2] with a set of additional rules. These rules are designed to solve a fundamental problem in distributed systems, viz. the lack of global knowledge. A node has no knowledge about the modes in which other nodes are holding the lock. By virtue of our protocol, any parent node *owns* the strongest (Mode M1 is said to be *stronger* than mode M2 if the former constrains the degree of concurrency over the latter) of all the modes owned in the tree rooted at that node. The token node owns the strongest lock mode of all other owned modes. As stronger modes have reduced compatibility, while granting a request at node A, it is safe to test the compatibility with the owned mode of A only, *i.e.*, local knowledge is sufficient to ensure correctness. The protocol was implemented, benchmarked and compared to the Naimi's protocol [3], which has the best known average case message complexity of $O(log(n))$. Figure 1 indicates significantly better scalability and savings in message overhead, particularly for the systems with a large number of nodes. The multi-granular nature of our protocol, combined with the low message complexity, is the prime causes of this difference, which represents the major contribution of our work.

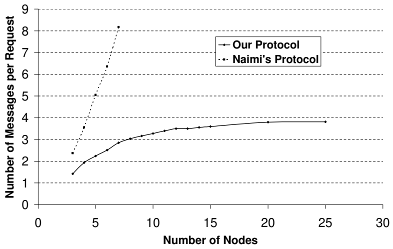

Figure 1: Scalability Behavior

2. REFERENCES

[1] O. M. Group. Concurrency service specification. www.omg.org/technology/documents/, Apr. 2000.

[2] F. Mueller. Prioritized token-based mutual exclusion for distributed systems. In *International Parallel Processing Symposium*, pages 791–795, 1998.

[3] M. Naimi, M. Trehel, and A. Arnold. A log(N) distributed mutual exclusion algorithm based on path reversal. *J. Parallel Distrib. Comput.*, 34(1):1–13, Apr. 1996.