Distributed Queue-based Locking using Advanced Network Features

Ananth Devulapalli
Ohio Supercomputer Center
1 South Limestone St., Suite 310
Springfield, OH 45502
ananth@osc.edu

Pete Wyckoff
Ohio Supercomputer Center
1224 Kinnear Road
Columbus, OH 43212
pw@osc.edu

Abstract

A Distributed Lock Manager (DLM) provides advisory locking services to applications such as databases and file systems that run on distributed systems. Lock management at the server is implemented using First-In-First-Out (FIFO) queues. In this paper, we demonstrate a novel way of delegating the lock management to the participating lock-requesting nodes, using advanced network primitives such as Remote Direct Memory Access (RDMA) and Atomic operations. This nicely complements the original idea of DLM, where management of the lock space is distributed. Our implementation achieves better load balancing, reduction in server load and improved throughput over traditional designs.

1 Introduction

Locking is essential for serialization of access to shared resources. In an uniprocessor environment, this is achieved using services such as mutexes, semaphores and monitors [22] that rely on the centralized management provided by the operating system. In shared-memory multiprocessor systems, the architecture frequently provides mechanisms to extend this centralized locking scheme across the domain of the machine. Distributed multiprocessor systems, however, rarely use networks that provide such features and hence resort to using a dedicated process to provide synchronization, called a Distributed Lock Manager (DLM).

DLM is a critical component in many applications that run on distributed systems, such as databases and file systems. Today, many vendors like Oracle [20], IBM [2] and HP [15] provide DLM as an integral component in their systems. Current DLM implementations achieve greater scalability and fault-tolerance over single-site lock serving models by distributing locks among all nodes in the cluster. Each node serves as a server for a subset of locks and as clients for the rest. A server uses FIFO queues for serializing requests for a shared resource.

This work was supported by the US DoE ASC program.

In this paper we show how to further augment load-balancing by distributing the queue-management among the lock-requesting clients. We focus particularly on the use of DLM in non-shared-memory distributed systems and we take advantage of atomicity features provided by modern high-speed communication networks. We start by explaining the current issues that motivate our work in Section 2. In the next section we explain the abstract design of how we distributed the queue-management. Then we explain our implementation of distributed FIFO queue using Remote Direct Memory Access (RDMA) based primitives. In Section 5 we describe the experiments and follow up with their results and analysis.

2 Motivation

In this section we describe traditional DLM designs and advanced capabilities of modern interconnects that motivate our work.

2.1 Locking in distributed systems

In multi-processor distributed systems, processes frequently need to share global resources such as files, memory buffers and access to storage or network devices. One approach to achieve synchronization among these processes is to concentrate the management of locks at a single site. Any node needing to lock a resource sends a request to the centralized manager. While this approach is simple, it suffers from two critical disadvantages, namely the presence of hot spots in the network and a single point of failure at the manager node. In the case of loosely coupled clusters that generally have higher latency networks and low-cost and hence low-reliability nodes, these problems are aggravated. As a result it is difficult to implement high performance shared services like file systems and database servers that rely on this model of centralized lock management.

As an alternative, lock management responsibility can be distributed across multiple nodes in two different fashions: various independent locks can be managed at different nodes, and aspects of the lock management protocol for a single lock can be handled by multiple nodes. The first aspect of spreading the management of multiple locks across