Zhenyu Yan

The Performance of Spin Lock Alternatives for Shared-Memory Multiprocessors

From

Thomas E. Anderson

In this paper, Anderson points out is there a valid algorithm software rotation waiting to give hardware support atomic instructions, or performance requires more complex hardware support? He test the performance of many spin-waiting algorithm. He also propose a novel method for explicitly queueing spinning processors in software by assigning each a unique sequence number when it arrive at the lock. At the end, he examine the performance of several hardware solutions that reduce the cost of spin-waiting.

In shared-memory multiprocessors, each processor can directly address memory. For consistency of the data structure, we need a method to serialize the operations done on it. Shared-memory multiprocessors provide some form of hardware support for mutual exclusion - atomic instructions. He discuss five different tests and solutions, Spin on Test-and-Set, Spin on Read, Delay Alternatives, Queuing In Share Memory and Single Bus Multiprocessors. For spin on test and set, performance decreases as the number of spinning processors increases. The lock retainer must compete with the spinning processor to access the locked position and other positions for normal operation. For spin on Read, it uses a cache to reduce the cost of spinning. When the lock is released, each cache is updated or invalid, waiting for the processor to see the changes and perform tests and settings. When the key part is small, this is as bad as the spinning on the test and setup. For delay alternative, we can insert a Delay between each memory reference after the lock is released or after each separate access to the lock. It can improve performance when there are few spinning processors. For queuing in share memory, each processor inserts itself into a queue and then spin on a separate memory location flag. When the processor completes the critical section, it sets a flag for the next processor in the queue. For Single Bus Multiprocessors, Systems use either single bus multiprocessors, while others use multistage network multiprocessors to counterbalance the tradeoffs. Lock latency may be better than software queuing.

For result, as the number of rotating processors increases, the simple method of spin-waiting can degrade performance. Software queuing and backoff have good performance even for a large number of spinning processors. Backoff has better performance when there is no contention, and queues perform best when there is contention. Special hardware support can also improve performance.

I think this article is still very useful even it is out dated. In homework 3, there is a question to run two copies of spocks, the first spock will print the first message, the second spock will print the second message, the first spock will print the third message and so on. We can use spin lock to solve this problem.

The presenter Shreyas Shetty and Forrest Williams were good. Their slide were detail and highlighted, but there were too many slides to read. They also include graphs in the presentations. Their slides covered all the important information about the whole paper, but I wish I can learn more form beyond this paper from the presentations.