CS443 Paper Review
Lei Yang
2005-4-20

****

Title

Lottery Scheduling: Flexible Proportional-Share Resource Management

Author

Carl A. Waldspurger and William E. Weihl

Summary

This paper presents lottery scheduling, a randomized mechanism that provides responsive control over the relative execution rates of computations.

Most important ideas

Lottery scheduling is a proportional-share resource management algorithm, to provide the performance assurances present in traditional non-real-time process schedulers. Lottery scheduling enables flexible control over relative process execution rates with a ticket abstraction and provides load insulation among groups of processes using currencies.

The key idea bechind lottery scheduling is quite simple: resource rights are represented by lottery tickets. Each allocation is determined by holding a lottery; the resource is granted to the client with the winning ticket. This effectively allocates resources to competing clients in proportion to the number of tickets that they hold. The authors have demonstrated that scheduling by lottery is probabilistically fair: a client's throughput is proportional to its ticket allocation and a client's average response time is inversely proportional to its ticket allocation. In addition, as computation speeds continue to increase, shorter time quanta can be used to further improve accuracy while maintaining a fixed proportion of scheduler overhead.

In addition to process scheduling, lottery tickets can also be used in modular resource management. Serveral techniques are necessary for lottery-based resource management: ticket transfers, ticket inflation, ticket currencies, and compensation tickets.

A prototype lottery scheduler was implemented on Mach 3.0 microkernel. A centralized lottery scheduler randomly select a winning ticket, and then search a list of clients to locate the client holding that ticket. It's interesting that their lottery scheduling policy co-exists with the standard timesharing and fixed-priority policies, thereby allowing a few high-priority threads to remain at their original fixed priorities. They have verified the effectiveness of lottery scheduler and lottery-based resource management through experiments.

Flaws/Qustions

I absolutely like the idea presented in this paper: it is neat and elegant. However, lottery scheduling was developed with primarily CPU-bound jobs in mind and thus may not work well for I/O-bound jobs. Lottery scheduling succeeds at scheduling CPU intensive jobs that are currently active. However, when interactive jobs, idling for most of their time, become active again, they have to compete for the CPU with other jobs and thus may suffer a long response time. In other words, it is straightforwad to expect that interactive jobs should have a higher priority to reduce the response time. But always giving higher priority to interactive jobs is not a good solution because jobs do not stay interactive continuously and may become CPU bound. Although compensation tickets could protentially help interactive and I/O bound jobs, it does not satisfactorily improve the response times of these jobs. Also, since lottery scheduling does not differentiate between interactive and CPU bound jobs, and it could often fail to schedule interactive jobs first. Thus to utilize the lottery scheduler to run interactive and I/O jobs, the scheduler should differentiate between various jobs and give a higher priority to interactive and I/O jobs when they come back from the blocked state.

Relevance/Potential Future Research

It would be interesting to see lottery scheduling be implemented in Unix/Linux kernel. (I think such patch already exists, however, the new scheduler doesn't co-exist with the original, priority-based scheduler.) In addition, it might be worthwhile to combine conventional priority-based scheduling schemes with lottery scheduling by assigning different priorities to groups of processes (I/O bound and interactive processes have higher priorities) and use lottery scheduling within each priority group. Actually, I am quite interested in this approach and might evaluate it in the course project.