

Scalable Linux Scheduling

Stephen Molloy
University of Michigan
smolloy@engin.umich.edu

Peter Honeyman
University of Michigan
honey@citi.umich.edu

in collaboration with

Ray Bryant
IBM Linux Technology Center
raybry@us.ibm.com

For most of its existence, the Linux operating system has been used primarily as a personal desktop operating system. In recent times, however, its use as a cost-efficient alternative for network servers, distributed workstations, and other large scale platforms has been increasing. Although its rise in popularity is remarkable, Linux still exhibits many undesirable traits. One such trait is its inability to scale. As a result, high load systems are limited by poor software design rather than physical resources. The primary focus of this research is to improve the scalability and robustness of the Linux operating system to support greater server workloads more reliably.

Concerned about the scalability of multithreaded network servers running on Linux, we are investigating improvements to the Linux scheduler. Our goal is to improve the scalability of the Linux scheduler to prepare it for enterprise-scale server workloads. Currently in Linux, every time the scheduler is called, it calculates a "goodness" value for each ready task; the task with the highest "goodness" value is the next task to run. This is an $O(n)$ operation where n is the number of ready tasks. Experiments by IBM indicate that as much as 30% of the total CPU time in the system can be spent in the scheduler when the number of tasks is high.

We are developing a new scheduler that limits the number of tasks examined on each scheduler invocation. The new scheduler will organize ready tasks in a manner which facilitates the selection of the best task to run. This organization groups tasks together based on parts of the "goodness" calculation which will not change. As a result, not only does the scheduler examine fewer tasks, but it also performs less computation per examination.

Our modified scheduler maintains the existing interfaces to the rest of the Linux kernel, closely matches the heuristics of the "goodness" calculation and compares favorably when running very light loads (desktop machines). At the same time, we can handle large numbers of threads and scale to many processors more gracefully. We will present benchmark comparisons between the two schedulers under both light (desktop) and heavy (server) workloads.

This work is a joint effort of the Linux Scalability Project at the University of Michigan Center for Information Technology Integration and the IBM Linux Technology Center.