

Achieving predictable timing and responsiveness in a commodity operating system through cooperative scheduling

Anirban Sinha,
Department of Computer Science, University of British Columbia.
anirbans@cs.ubc.ca

March 26, 2007

Abstract

There was a time when one could make a clear distinction between general purpose computing and real-time computing. That distinction has become increasingly blurred as rich media applications have entered the mainstream. The performance requirements of these *soft real-time* applications differ from more traditional computation in numerous ways, notably in terms of the balance between long-term throughput and short-term responsiveness. Hence, these applications poses a significant scheduling challenge for a general purpose commodity operating system.

The present Linux kernel scheduling heuristics prioritize applications which it considers to be IO intensive ones over CPU intensive ones. However, this is a relative classification and is dependent upon the workload distribution at a given point in time. In our work, we present results that show that this heuristics does not really work well with a mix of multimedia applications and other such similar applications (such as databases and webservers) that are both IO and CPU intensive. To add to our woes, many multimedia applications are adaptive in nature. With a mix of adaptive real time and other non real time best effort applications, the CPU remains saturated most of the time. Thus, existing approaches to real time which assumes an a priori reservation and/or under load does not work. Moreover, along with introducing unpredictable timing behavior, traditional preemptive scheduling discipline breaks the application level adaptive logic and is also often cache unfriendly.

In this work, we intend to address the issues mentioned above. We propose a *cooperative process scheduling* scheme whereby real time applications cooperate with each other by respecting each other's deadlines with assistance from the kernel. The kernel, which supervises this cooperation, is responsible for policing badly behaving/non-cooperative processes. It also decides how to schedule non-realtime, best effort processes. For scheduling the later and for policing, we use a virtual time algorithm, which we call *periodic virtual time fair scheduling*. This approach helps to achieve work conservation (non a priori reservation) and also long term fair scheduling. Cooperation reduces the problem of unpredictable timeliness by minimizing involuntary preemption. It is also friendly towards application level adaptive logic during overload. Yielding at appropriate times (as decided by the application) can amortize the cost of cache pollution which comes as an added advantage.

We evaluate our approach using the existing adaptive streaming framework, QStream as the benchmarking tool. We show, with a series of benchmarks, that using our proposed solution, the time sensitive applications can not only maintain good timeliness but also can achieve uniform graceful adaptation across all processes in overload.