SMART: A Processor Scheduler for Multimedia Applications

Jason Nieh and Monica S. Lam
Computer Systems Laboratory
Stanford University
Stanford, CA 94305
{nieh, lam}@cs.stanford.edu

Applications that manipulate digital audio and video represent a growing class of computations executed by workstation users. This class of computations is known as continuous media. Their distinguishing characteristic is that they must process and transport media samples within application-specific time constraints. To support these real-time activities, the operating system must manage resources so that their time constraints can be satisfied whenever possible.

The integration of continuous media activities into the workstation environment requires that the operating system not only support their time constraints, but that it continue to manage resources effectively for the interactive and batch activities already found in conventional applications. These conventional activities have no application-specific time constraints but they are expected to make reasonable forward progress. Despite the diverse performance characteristics and resources requirements of real-time and conventional activities, the operating system must allow all classes of computations to co-exist and share resources effectively. Furthermore, as continuous media activities alone can consume the resources of an entire machine, the operating system must manage resources effectively not just when all time constraints can be satisfied, but more importantly, when the system is overloaded. As processor cycles are often the most oversubscribed resource, effective processor scheduling is of paramount importance.

Previous attempts at processor scheduling have failed to provide adequate support for multimedia applications. Their failures highlight the difficulty of effectively scheduling the combination of real-time and conventional activities. Some of these attempts simply ignore time constraints by just "getting out of the way", or arbitrarily timeslicing on a per activity or per class of activity basis. Others impose artificial rate and deadline parameters on conventional activities that the user is required to hand-

tune by trial-and-error for each mix of activities. Their drawbacks are even more egregious when the system is overloaded, resulting in pathological behavior in the worst case.

To effectively support multimedia applications, we have created SMART (Scheduler for Multimedia And Real-Time). SMART manages resources so that time constraints can be satisfied whenever possible, and provides graceful degradation when the system is overloaded. At the same time, it does not simply allow real-time activities to monopolize the processor to the starvation of conventional activities, but rather it manages resources so that all classes of activities, with and without time constraints, can co-exist and share the processor effectively.

A key aspect of the SMART design for scheduling real-time and conventional activities is its separation of importance and urgency, in contrast to previous approaches which confuse these two dimensions. Importance determines how much processing time an activity should be allowed to use. SMART gives all activities, whether real-time or conventional, their fair allocations of processing time by default. Urgency determines when an activity needs to execute. SMART uses urgency to sequence activities so that the more urgent real-time activities are given their fair allocations first in order to satisfy their deadlines. Our combination of fair allocation with deadline sequencing ensures that real-time activities can satisfy their deadlines fairly while conventional activities make fair forward progress, without the need for any user parameters.

As different users may have different preferences for the behavior of a mix of applications, simple controls are provided that allow the user to bias the allocation of resources away from fairness. These SMART controls are priorities and shares. Priorities allow important activities, which can be either real-time or conventional, to monopolize the resources. Shares allow different types of activities to obtain a portion of the machine in proportion to their number of shares. The resulting high degree of predictable user control over the dynamic sharing of processing resources allows a wide range of desirable behaviors not achievable with previous approaches.