Abstract

Traditional processor scheduling mechanisms in operating systems are fairly rigid, often supporting only one fixed scheduling policy, or, at most, a few "scheduling classes" whose implementations are closely tied together in the OS kernel. This paper presents *CPU inheritance scheduling*, a novel processor scheduling framework in which arbitrary threads can act as schedulers for other threads. Widely different scheduling policies can be implemented under the framework, and many different policies can coexist in a single system, providing much greater scheduling flexibility. Modular, hierarchical control can be provided over the processor utilization of arbitrary administrative domains, such as processes, jobs, users, and groups, and the CPU resources consumed can be accounted for and attributed accurately. Applications as well as the OS can implement customized local scheduling policies; the framework ensures that all the different policies work together logically and predictably. As a side effect, the framework also cleanly addresses priority inversion by providing a generalized form of priority inheritance that automatically works within and among multiple diverse scheduling policies. CPU inheritance scheduling extends naturally to multiprocessors, and supports processor management techniques such as processor affinity [7] and scheduler activations [1]. Experimental results and simulations indicate that this framework can be provided with negligible overhead in typical situations, and fairly small (5-10%) performance degradation even in scheduling-intensive situations.