

[RFC][PATCH] O(1) Entitlement Based Scheduler

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-02/6548.html>

From: John Lee (johnl_at_aurema.com)

Date: 02/25/04

Date: Thu, 26 Feb 2004 01:35:05 +1100 (EST)

To: linux-kernel@vger.kernel.org

Hi everyone,

This patch is a modification of the O(1) scheduler that introduces O(1) entitlement based scheduling for SCHED_NORMAL tasks. This patch is aimed at keeping the scalability and efficiency of the current scheduler, but also provide:

- Specific allocation of CPU resources amongst tasks
- Scheduling fairness and good interactive response without the need for heuristics, and
- Reduced scheduler complexity.

The fundamental concept of entitlement based sharing is that each task has an `_entitlement_` to CPU resources that is determined by the number of `_shares_` that it holds, and the scheduler allocates CPU to tasks so that the `_rate_` at which they receive CPU time is consistent with their entitlement.

The usage rates for each task are estimated using Kalman filter techniques, the estimates being similar to those obtained by taking a running average over twice the filter `_response half life_` (see below). However, Kalman filter values are cheaper to compute and don't require the maintenance of historical usage data.

The use of CPU usage rates also makes it possible to impose `_per task CPU usage rate caps_`. This patch provides both soft and hard CPU usage rate caps per task. The difference between a hard and soft cap is that when unused CPU cycles are available, a hard cap is `_always_` enforced regardless, whereas a soft cap is allowed to be exceeded.

Features of the EBS scheduler

=====

CPU shares

Each task has a number of CPU shares that determine its entitlement. Shares can be read/set directly via the files

/proc/<pid>/cpu_shares
/proc/<tgid>/task/<pid>/cpu_shares

or indirectly via setting the task's nice value using nice or renice. A task may be allocated between 1 and 420 shares with 20 shares being the default allocation. A nice value ≥ 0 is mapped to $(20 - \text{nice})$ shares and a value < 0 is mapped to $(20 + \text{nice} * \text{nice})$ shares. If shares are set directly via /proc/<pid>/cpu_shares then its nice value will be adjusted accordingly.

CPU usage rate caps

A task's CPU usage rate cap imposes a soft (or hard) upper limit on the rate at which it can use CPU resources and can be set/read via the files

/proc/<pid>/cpu_rate_cap
/proc/<tgid>/task/<pid>/cpu_rate_cap

Usage rate caps are expressed as rational numbers (e.g. "1 / 2") and hard caps are signified by a "!" suffix. The rational number indicates the proportion