46

Posted by u/nburgin 9 months ago

Why nice levels are a placebo and have been for a very long time, and no one seems to have noticed

(EDIT: I was assuming this configuration was default in most distros, but now I'm starting to think it just might be some Ubuntu/Mint-specific weirdness...)

Linux has a feature called 'autogrouping', which is enabled by default on ~~most~~ many systems, for scheduling processes (see manual page excerpt below). Essentially it causes the scheduler to act primarily on the nice level set for process *groups* rather than individual processes.

This generally improves responsiveness and fairness for typical use cases where the nice value is always left untouched at zero anyway, by not giving a task which is split into many processes a larger share of cpu time than one that is a single process despite having the same nice level. While your desktop session (including all apps launched through graphical launchers) typically shares one autogroup, opening a terminal window (which is typically where cpu-heavy background tasks are launched) creates a new autogroup, and background services generally have their own autogroups as well.

Are you with me so far? Here's where it gets screwy: when autogrouping is turned on, the standard per-process nice level **only** affects scheduling priority **relative to other processes in its group**. And the `nice` and `renice` commands (and their underlying system calls) are only aware of the traditional per-process nice value; they do **not** act on autogroups. Autogroup nice level can only be changed by writing to the virtual file at `proc/<pid>/autogroup`, and **none of the standard utilities for dealing with priority seem to take this into account**.

While autogrouping tends to ensure fairness, what if you don't want fairness? What if you want to run a background task in very low priority? So in your terminal, instead of running `make -j32` you run `nice 15 make -j32`. Except *oops*, that actually made no difference! Since its autogroup nice level is still zero and the build you just started has no other processes running in its autogroup, its nice level is *irrelevant*.

The dark side of autogrouping is that with it enabled, the conventional commands and system calls for setting priority mostly become *placebos* that don't actually do anything. This means that power users wanting to actually control the priority of their processes, are not getting the result they expect. Also the few programs that set their own nice level (such as Folding@Home, which kindly attempts to set itself to nice +19)... actually fail in their attempt to (de)prioritize themselves, and still receive a "fair" share of cpu time even when there are other sources of high CPU load running.

The only place where you'll find unrelated tasks running on the same autogroup and thus making the regular nice level somewhat relevant, is your main desktop session excluding terminal

r/linux

All things Linux and GNU/Linux -- this is neither a community exclusively about the kernel Linux, nor is exclusively about the GNU operating system.

535k	1.0k
Members	Online

Created Jan 25, 2008

JOIN

ADVERTISEMENT

ADVERTISEMENT

`proc/sys/kernel/sched_autogroup_enabled`;

- Hard-disable autogroups by custom-building a kernel with `CONFIG_SCHED_AUTOGROUP=N`;
- Start using the autogroup nice level at `proc/<pid>/autogroup` instead of the regular nice level. There seems to be no standard command for this, though I've jerry-rigged a crude solution
- The nuclear option of using `chrt -f` to bypass autogroups using `SCHED_FIFO` for something you don't want to have cpu time stolen from by lower-priority processes. Of course this only works on interactive processes as opposed to batch ones, or else your system could hang. While it's a clumsy solution, notably this is an option that someone who isn't aware of autogroups might still come up with on their own.

Excerpt from man 7 sched:

> **The autogroup feature**
>
> Since Linux 2.6.38, the kernel provides a feature known as autogrouping to improve interactive desktop performance in the face of multiprocess, CPU-intensive workloads such as building the Linux kernel with large numbers of parallel build processes (i.e., the **make**(1) **-j** flag).
>
> This feature operates in conjunction with the CFS scheduler and requires a kernel that is configured with **CONFIG_SCHED_AUTOGROUP**. On a running system, this feature is enabled or disabled via the file <u>/proc/sys/kernel/sched_autogroup_enabled</u>; a value of 0 disables the feature, while a value of 1 enables it. The default value in this file is 1, unless the kernel was booted with the <u>noautogroup</u> parameter.
>
> A new autogroup is created when a new session is created via **setsid**(2); this happens, for example, when a new terminal window is started. A new process created by **fork**(2) inherits its parent's autogroup membership. Thus, all of the processes in a session are members of the same autogroup. An autogroup is automatically destroyed when the last process in the group terminates.
>
> When autogrouping is enabled, all of the members of an autogroup are placed in the same kernel scheduler "task group". The CFS scheduler employs an algorithm that equalizes the distribution of CPU cycles across task groups. The benefits of this for interactive desktop performance can be described via the following example.

cycles at the expense of the other jobs on the system.

A process's autogroup (task group) membership can be viewed via the file <u>/proc/[pid]/autogroup</u>:

$ **cat /proc/1/autogroup** /autogroup-1 nice 0

This file can also be used to modify the CPU bandwidth allocated to an autogroup. This is done by writing a number in the "nice" range to the file to set the autogroup's nice value. The allowed range is from +19 (low priority) to -20 (high priority). (Writing values outside of this range causes **write**(2) to fail with the error **EINVAL**.)

The autogroup nice setting has the same meaning as the process nice value, but applies to distribution of CPU cycles to the autogroup as a whole, based on the relative nice values of other autogroups. For a process inside an autogroup, the CPU cycles that it receives will be a product of the autogroup's nice value (compared to other autogroups) and the process's nice value (compared to other processes in the same autogroup.

The use of the **cgroups**(7) CPU controller to place processes in cgroups other than the root CPU cgroup overrides the effect of autogrouping.

The autogroup feature groups only processes scheduled under non-real-time policies (**SCHED_OTHER**, **SCHED_BATCH**, and **SCHED_IDLE**). It does not group processes scheduled under real-time and deadline policies. Those processes are scheduled according to the rules described earlier.

The nice value and group scheduling

When scheduling non-real-time processes (i.e., those scheduled under the **SCHED_OTHER**, **SCHED_BATCH**, and **SCHED_IDLE** policies), the CFS scheduler employs a technique known as "group scheduling", if the kernel was configured with the **CONFIG_FAIR_GROUP_SCHED** option (which is typical).

Under group scheduling, threads are scheduled in "task groups". Task groups have a hierarchical relationship, rooted under the initial task group on the system, known as the "root task group". Task groups are formed in the following circumstances:

- All of the threads in a CPU cgroup form a task group. The parent of this task group is the task group of the corresponding parent cgroup.

> **CONFIG_FAIR_GROUP_SCHED**), then all of the processes on the system are notionally placed in a single task group.
>
> Under group scheduling, a thread's nice value has an effect for scheduling decisions <u>only</u> <u>relative</u> <u>to</u> <u>other</u> <u>threads</u> <u>in</u> <u>the</u> <u>same</u> <u>task</u> <u>group</u>. This has some surprising consequences in terms of the traditional semantics of the nice value on UNIX systems. In particular, if autogrouping is enabled (which is the default in various distributions), then employing **setpriority**(2) or **nice**(1) on a process has an effect only for scheduling relative to other processes executed in the same session (typically: the same terminal window).
>
> Conversely, for two processes that are (for example) the sole CPU-bound processes in different sessions (e.g., different terminal windows, each of whose jobs are tied to different autogroups), <u>modifying</u> <u>the</u> <u>nice</u> <u>value</u> <u>of</u> <u>the</u> <u>process</u> <u>in</u> <u>one</u> <u>of</u> <u>the</u> <u>sessions</u> <u>has</u> <u>no</u> <u>effect</u> in terms of the scheduler's decisions relative to the process in the other session. A possibly useful workaround here is to use a command such as the following to modify the autogroup nice value for <u>all</u> of the processes in a terminal session:
>
> $ **echo 10 > /proc/self/autogroup**

40 Comments Share Save Hide Report 82% Upvoted

This thread is archived
New comments cannot be posted and votes cannot be cast

SORT BY **BEST**

daemonpenguin 15 points · 9 months ago

On my system it definitely makes a difference whether I run some processes with nice (or not). One program in particular, you can watch it draw text line by line if it's run with a high nice value, when otherwise it draws the whole screen instantly.

So I was curious if this is because of a misunderstanding on the OP's part or a configuration issue.

unaffected :p

Continue this thread →

MiningMarsh 11 points · 9 months ago

Auto-grouping is not a default on any system as far as I am aware: systemd and openrc both

VIEW ENTIRE DISCUSSION (40 COMMENTS)

More posts from the linux community

Posted by u/raghukamath 2 days ago

495

Popular Application Krita 4.3.0 Released

krita.org/en/ite...

90 Comments Share Save Hide Report

Continue browsing in r/linux →