The History of Linux Task Isolation

Matt Fleming

WHAT IS TASK ISOLATION?

- Running userspace tasks and workloads without interruption
- There are various levels of isolation, it's kinda like a spectrum
- **Not** talking about *process isolation* which is a security thing (VMs/containers)
- Linux has supported some version of this for 18 years
- Used for HPC, real-time, and userspace drivers (networking)

Linux Task Isolation Through The Years

2002

Support added for **sched_setaffinity()** and **sched_getaffinity()** system calls in **v2.5.8**

2004

isolcpus allows CPUs to be removed from scheduler domains and load balancer. Added in **v2.6.9**

2013

nohz_full reduces the timer tick to 1HZ even while running tasks since **v3.10**

2018

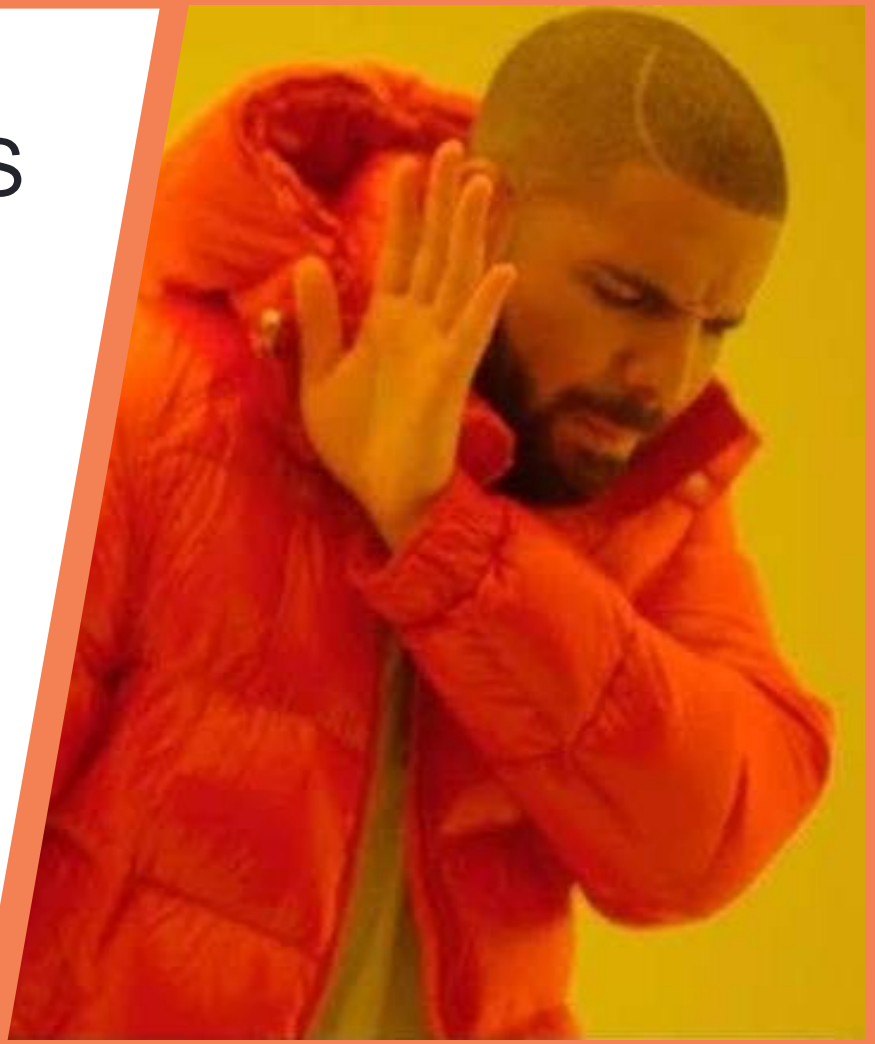
nohz_full support disables the 1HZ residual timer in **v4.17**

Today

prctl() task isolation?

BAD KERNEL SERVICES

- **Preemption:** Multitasking and sharing hardware resources between tasks
- **Interrupts:** Servicing device/CPU interrupts
- **Timers:** These maintain scheduler run queues and other statistics
- **RCU callbacks:** Major in-kernel synchronisation mechanism



GOOD KERNEL SERVICES

- **Allocating dedicated hardware resources:** This means we need CPU and IRQ affinity
- **Ability to disable kernel features:** We don't need things like the scheduler tick
- **Notification when we break isolation**: For "hard" isolation we need to know when we've exited isolation mode

CPU Affinity System Calls

CPU AFFINITY SYSTEM CALLS

- Robert Love added support to the Linux kernel for **sched_getaffinity()** and **sched_setaffinity()** in 2002

- These system calls let you control which CPU tasks can run on using a cpumask

- A cpumask with one CPU bit set *pins* that task to a single CPU

author Robert Love <rml@tech9.net> 2002-04-09 04:02:50 -0700
committer Linus Torvalds <torvalds@penguin.transmeta.com> 2002-04-09 04:02:50 -0700
commit 22e962f9b7a7abbc2d17ceaf3917bb8e67b68a8f (patch)
tree e5a8c26a9b54e7aa0edb64ec9df335dfabe72c71
parent b96ad24ad44794ee435987291c83b77e6c6f96ac (diff)
download history-22e962f9b7a7abbc2d17ceaf3917bb8e67b68a8f.tar.gz

[PATCH] cpu affinity syscalls v2.5.8-pre3

This patch implements the following calls to set and retrieve a task's CPU affinity:

```
int sched_setaffinity(pid_t pid, unsigned int len,
                                  unsigned long *new_mask_ptr)
int ched_getaffinity(pid_t pid, unsigned int len,
                                  unsigned long *user_mask_ptr)
```

CPU affinity provides dedicated hardware resources

Isolating CPUs With isolcpus

ISOLATING CPUS

- Support for removing CPUs from the kernel's scheduler domains added in v2.6.9 (2004) by Dimitri Sivanich
- **isolcpus=** kernel command-line param takes a CPU list and leaves those CPUs alone
- This option **disables the load balancer!**

author Dimitri Sivanich <sivanich@sgi.com> 2004-08-23 21:09:04 -0700
committer Linus Torvalds <torvalds@ppc970.osdl.org> 2004-08-23 21:09:04 -0700
commit 6f4c30b1ca21efd1d8bac1ee49100f75f840f5e0 (patch)
tree 6ddfecc1a3af42436e79a7c83d501081815522f5
parent c183e253bd5ca0c923060dd394c236f81e3b8690 (diff)
download history-6f4c30b1ca21efd1d8bac1ee49100f75f840f5e0.tar.gz

[PATCH] sched: isolated sched domains

Here's a version of the isolated scheduler domain code that I mentioned in an RFC on 7/22. This patch applies on top of 2.6.8-rc2-mm1 (to include all of the new arch_init_sched_domain code). This patch also contains the 2 line fix to remove the check of first_cpu(sd->groups->cpumask)) that Jesse sent in earlier.

Note that this has not been tested with CONFIG_SCHED_SMT. I hope that my handling of those instances is OK.

Signed-off-by: Dimitri Sivanich <sivanich@sgi.com>
Signed-off-by: Andrew Morton <akpm@osdl.org>
Signed-off-by: Linus Torvalds <torvalds@osdl.org>

You need to manually place tasks with sched_setaffinity()

Going Timerless With nohz_full

GOING TIMERLESS IN v3.10

- Even with CPU affinity, the *timer tick* (**CONFIG_HZ**) still fires periodically to do things like allow the scheduler pick new tasks to run. Linux has been able to disable the timer tick for years **but only when the CPU goes idle**

- **nohz_full=** kernel command-line param (2013) offloads the timer tick to a *housekeeping CPU* when only 1 task is running. It's misnamed though because a timer still fires once a second (1HZ)

- **nohz_full=** offloads RCU callbacks through the **rcu_nocb=** code

GOING TIMERLESS IN v4.17

- That 1HZ tick was still causing interrupts for applications and was removed in v4.17 (2018). Modern kernels can run without the 1HZ timer tick

- The timer tick can **still be enabled** for a bunch of reasons though!
  - Perf events
  - Timers
  - Scheduler (providing fairness, e.g. SCHED_RR and SCHED_OTHER)
  - RCU
  - Unstable clocks

SHOUT OUT!

- Frederic Weisbecker (frederic@kernel.org) has been working on reducing the interference from timers and tasks for years

- He's got some great talks that cover this work
  - CPU isolation - state of the art, Kernel Recipes 2015
  - State of CPU Isolation, Kernel Recipes 2018

Task Isolation Mode

TASK ISOLATION MODE

- Originally created by Chris Metcalf while at Mellanox. Picked up by Alex Belits at Marvell

- Designed to be a “hard” isolation mode and requires task or workload to issue new **prctl(PR_SET_TASK_ISOLATION)** system calls. Future work will add cgroups (cpusets) support

- If your task violates isolation (system call, fault, exception, interrupt), you get a **SIGKILL** signal

Upstream review has been good.
Could be merged in an upcoming release?

References

- A full task-isolation mode for the kernel (LWN.net) [link]
- Latest task_isolation mode patches (linux-kernel mailing list) [link]
- Dropping the timer tick -- for real this time (LWN.net) [link]
- (Nearly) full tickless operation in 3.10 (LWN.net) [link]
- Reducing OS jitter due to per-cpu kthreads (Linux kernel source) [link]

- Shielding Linux Resources (SUSE docs) [link]

- isolcpus is deprecated, kinda (personal blog) [link]

Slide design taken from “How agencies make money”, [link]

Thank You

@fleming_matt

mfleming