# How many CPUs do I have?

...and other perplexing questions applications running containers must answer

Tycho Andersen <tandersen@netflix.com>, <tycho@tycho.pizza>

## Why do language runtimes ask?

- mostly to size thread pools/GC threads
  - JVM’s JIT spawns N threads
- Size arenas/allocators
  - E.g. tcmalloc wants to know how many per-thread arenas to allocate

## How do we do it today?

- So many interfaces
  - isol_cpus= kernel command line
  - /sys/devices/system/{cpu,memory}/online
  - /proc/stat, /proc/cpuinfo
    - lxcfs
  - sched_getaffinity()
- What’s missing?
  - cgroup info
  - cfs quota
  - SCHED_EXT / scx_*

## Hard to answer

- tcmalloc: https://github.com/google/tcmalloc/issues/188
  - segfaults on non-sequential cpu assignments
- JVM’s implementation
  - https://bugs.openjdk.org/browse/JDK-8322420
  - Queries cpuset.cpus (not .effective)
  - No .effective for memory, must recurse up the tree
  - 2CPU jobs with 384G heaps
  - https://stackoverflow.com/questions/75327454/how-do-i-read-the-effective-cgroups-limits-for-the-current-process-using-sys-fs/77234728#77234728

## Hard to answer (even more)

- (g)libc (aka nprocs, sysconf(NPROC_ONLIN))
  - Used to use /sys/devices/system/node, switched to sched_getaffinity() https://sourceware.org/bugzilla/show_bug.cgi?id=15630
  - Used by lots of libraries (e.g. jemalloc) to reason about memory arena counts, incorrect number of memory arenas wastes memory
  - Florian Weimer “Should be done by the kernel” https://bugzilla.kernel.org/show_bug.cgi?id=151821
- Musl
  - sched_getaffinity()

## Hard to answer (still more)

- libuv (nodejs)
  - Looks at /proc/stat, /proc/cpuinfo https://github.com/libuv/libuv/issues/2351
- lxcfs renderings incorrect in /proc/stat, /sys/devices/system/cpu
  - https://github.com/lxc/lxcfs/pull/557
  - https://github.com/lxc/lxcfs/pull/558
  - Causing crashes in libuv, jvm
  - `cpu_view` feature to reason about cfs shares/quota

## Where should this computation live?

- Nowhere
- Container runtime
  - Traditional
- Kernel: mechanism not policy
  - Mechanisms exist! Lots of them!
  - sched_ext would mean the algorithm itself is dynamic
- Userspace: one place so people don’t have to reimplement
  - libresource
  - systemd
  - util-linux

## Prior art

- Runtime implementations
  - Sometimes incorrect :)
- Lxcfs
  - Can only do file-based masking
  - Could add some seccomp fixing of sched_getaffinity()
- Libresource
  - Not container aware
  - Pairs well with lxcfs endpoints

## Two approaches: IPC vs. library

- IPC (aka varlink) via systemd or container engine
    - Has to be running on the host
        - Not all containers run systemd
    - Multiple implementations can exist
    - No extra dependencies: most people have a json parser handy
    - RFE github.com/systemd/systemd#31810

- Library
    - Need to get people to link against it
    - Potential home in libresource, util-linux
    - Generally new dependency for most applications

## Cgroups: “the delegation boundary”

- i.e. `Delegate=yes` from man 5 systemd.resource-control
- Runtime creates `/containers.slice/<id>`
- Application creates `/containers.slice/<id>/myapp and applies CPU limit`
- Application launches a JVM inside …/myapp
- JVM queries varlink API: what happens?
  - Looking past the delegation boundary means unprivileged code controls privileged code
  - Not looking past the delegation boundary means the answer is wrong: `/containers.slice/<id>` vs `/containers.slice/<id>/myapp`

# dankeschön

Tycho Andersen <tandersen@netflix.com>,
<tycho@tycho.pizza>