Libamicontained - a library for reasoning about resource restriction

tandersen@netflix.com & sdabdoub@netflix.com
Repo: https://github.com/Netflix/libamicontained

Fosdem 2024

How many (and which) CPUs do I have?

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

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
  - No dependencies (golang doesn’t want libc, etc.)
  - Small
  - Correct

libamicontained

- An cgroup/container aware API for getting resources (i.e. cpu count).
- Consolidates well known algorithms for calculating cpu count from cgroup controllers
- Statically linked, c ABI, written in rust for safety.
- Meant to be used by language runtimes and applications in place of syscalls or /proc files.
- repo: https://github.com/Netflix/libamicontained

Why do runtimes ask?

- mostly to size thread pools/GC threads
- Size arenas/allocators

What can go wrong (example)

- 10 containers requesting on a host
- Host has 100 cpus, each container has 10% cfs quota
- containers see 100 cpus, create 100 threads
- Each bursts through all of quota in first time quantum
  - Or starve their own threads

How to compute the answer?

- num_cpus() - expected sched_getaffinity call. Takes into account cpusets, affinity mask, online cpus, etc.
- recommended_threads() - num_cpus() further constrained by cgroup CFS quota (i.e. threads = quota/period). Similar to systemd and lxcfs calculation.

Example with recommended_threads()

- 10 containers requesting on a host
- Host has 100 cpus, each container has 10% cfs quota
- containers see 10 cpus, create 10 threads
- All is well ;)

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

A step further…

- In a container world with quota+shares, cpu count is not static
  - “Core equivalent” computation of quotas
  - May get the whole box then throttled
  - Unused CPU time is lost
  - Allowed to change cgroup/cpus on live process, nothing takes that into account
- Why not have dynamic threadpools?
  - Not the way it works today

Merci!

tandersen@netflix.com & sdabdoub@netflix.com

Fosdem 2024

Question: how to integrate?

- Integrate with libresource?
- .a? .so? Other interfaces?