Limiting Memory Use of Userspace Per-CPU Data Structures in Containers

Mathieu Desnoyers, EfficiOS

Containers and Checkpoint/Restore Microconference
Linux Plumbers Conference 2024
September 18-20
Vienna, Austria

Userspace CPU Number Use-Cases

- Userspace Per-CPU data use-cases:
  - Tracing ring buffers,
  - Memory allocators
    - tcmalloc, jemalloc,
    - GNU C Library malloc(3),
  - Caches (e.g. NPTL thread stack caches),
  - Schedulers,
  - Statistics counters.
- Userspace uses the observable number of CPUs to automatically scale the number of threads.

Problem Context

- New machines with 512+ hardware threads (and thus logical CPUs) bring interesting challenges for user-space per-CPU data structures due to their large memory use.
- The RSEQ per-memory-map concurrency IDs (upstreamed in Linux v6.3) allow indexing user-space memory based on indexes derived from the number of concurrently running threads.
- I plan to apply the same concept to IPC namespace.

cpuset(7)

- This provides memory use upper bound when limiting containers with cpusets (e.g. cpuset: 0-31),
- It does not work when limiting containers that have many threads with time slices (e.g. cpu.max 2000 1000),
- Cpusets are far from ideal to describe the constraints in a cloud-native way:
  - those are bound to the machine topology,
  - hard to compose containers expressed with cpuset constraints,
  - tricky with big.LITTLE, p-core/e-core CPUs.

Discuss Proposal: cpu.max.concurrency

- Introduce a new “cpu.max.concurrency” interface file to the cpu controller, which defines the maximum number of concurrently running threads for the cgroup.
- Track the number of CPUs concurrently used by the cgroup.
- Extend the scheduler to constrain migration to the currently used set of cpus when the number of concurrently used CPUs reaches the maximum threshold.

Discuss Proposal

- Can be achieved by counting the number of threads in each runqueue belonging to the cgroup with per-CPU counters.
- Track the total number of used CPUs in a global counter within the cgroup.
- Track the set of used CPUs in a per-cgroup cpumask.

Discuss Proposal

- If sched_setaffinity or cpuset is used within the cgroup to add a thread affinity constraint that would require the scheduler to go beyond concurrency limits, fail with -EINVAL.
- Need to be able to move a set of threads across runqueues to allow migration when an affinity constraint is added without going beyond the concurrency limits.
- Should it be allowed to change cpu max concurrency limits dynamically ? If so, it may fail if trying to set the limit to a number that is not feasible due to the current cpuset/sched affinity masks.