# A New Cgroup cpu.max.concurrency Controller Interface File

**Mathieu Desnoyers**, EfficiOS

**FOSDEM 2025**
February 1 & 2, 2025
Brussels, Belgium

EfficiOS

# Current/Max CPU Number in Userspace

- 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*** (logical CPUs) bring interesting challenges:
  - Memory footprint of user-space per-CPU data structures and
  - Scaling the number of worker threads.
- The RSEQ per-memory-map concurrency IDs allow ***indexing user-space memory*** with indexes bounded by the ***number of concurrently running threads***.
  - upstreamed in Linux v6.3

# Current Approaches: cpuset(7)

- This provides memory use upper bound when limiting containers with cpusets (e.g. cpuset: 0-31),
- 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.

# Current Approaches: CPU Cgroups

- Allow limiting containers to a specified portion of time slice:
  - e.g. cpu.max 2000 1000
  - maximum 2000 µs per 1000µs slice
  - 200% of CPU time
- Does not restrict migration, which meant it can result in either:
  - 2 CPUs running the workload 100% of the time, or
  - 200 CPUs running the workload 1% of the time.
- Cannot effectively restrict the number of concurrent CPUs that can be used by the cgroup.

# 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.
- ***Extend the Linux scheduler*** migration and load balancer to:
  - track the number of CPUs concurrently used by the cgroup,
  - constrain migration to the currently used set of cpus when the number of concurrently used CPUs reaches the maximum threshold.

# Additional Cgroups Data Structures

- Count of the number of threads in each runqueue belonging to the cgroup with per-CPU counters within each CPU cgroup.
- Count the total number of used CPUs in a global counter within the cgroup.
- Keep track of the set of used CPUs in a cpumask within the cgroup.

# ABI Error Handling (Limit Cases)

- If ***sched_setaffinity(2) or cpuset(7)*** are used within the cgroup to add a thread affinity constraint that would require the scheduler to go beyond concurrency limits (e.g. disjoint affinity set), ***fail with EINVAL***.
- ***Decreasing cpu.max.concurrency*** to a value that would not have any task placement solution due to the current affinity or cpuset limits ***fails with EINVAL***.
- ***Increasing cpu.max.concurrency*** should never fail.