Restartable Sequences: Scaling Per-Core Shared Memory Use in Containers

Mathieu Desnoyers
EfficiOS Inc.

Restartable Sequences in a Nutshell

- Restartable Sequences (RSEQ) system call,
- Register a per-thread area,
- Allow user-space to create small “transactions” which are managed by the kernel,
- Allow fast access to per-cpu data in user-space,
- Used by glibc to speed up sched_getcpu(3) since glibc 2.35,
- Use cases: memory allocators (tcmalloc), user-space tracers and counters (LTTng).

Problem and Goals

- On large multi-core systems partitioned as many containers with cpusets, user-space per-CPU data structures waste memory.
- RSEQ critical sections enable use of per-cpu data structures in userspace.
- Minimize memory allocation when there is little concurrency for threads touching the data structure.
- Allocate user-space per-cpu data with min(nr threads, sched affinity, cpuset) entries.

Private Data Structures (Per-Memory-Space)

- “Restartable Sequences: Scheduler-Aware Scaling of Memory Use on Many-Core Systems”, LPC2022 Refereed Track
  - https://lpc.events/event/16/contributions/1391/

Private Data Structures (Per-Memory-Space)

- RSEQ vcpu_id allocation domain per memory space (typically per-process)
  - New struct rseq field “mm_vcpu_id”
  - Getter for mask of possible vcpus (invariant)
    - For max size allocation of per-vcpu-data,
    - Except for CRIU
      - *Maximum number of CPUs can change between checkpoint and restore.*
  - Getter for mask of online vcpus
    - Hint for preallocation
    - Consider sched affinity and cpuset

Shared Data Structures (Across Memory-Spaces)

- System-wide (root namespace)
  - use sched_getcpu(3) or rseq->cpu_id,
  - use either mask or max value of /sys/devices/system/cpu/possible,
  - getter for mask possible cpus (invariant) -> for max size allocation through /sys/devices/system/cpu/possible,
  - getter for mask online cpus -> for preallocation through /sys/devices/system/cpu/online.

Shared Data Structures (Across Memory-Spaces)

- Per-pid-namespace/cgroup cpuset
  - Typical use cases is to use cgroup v2 cpusets to restrict pid namespaces to a subset of cpus on the system.
    - Target number (with burst),
    - Partition.
  - Getter for mask possible vcpus (invariant) -> for max size allocation,
  - Getter for mask online vcpus -> for preallocation.
    - Consider sched affinity and cpuset

Shared Data Structures (Across Memory-Spaces)

- RSEQ vcpu_id allocation domain per:
  - pid namespace,
  - user namespace,
  - cgroup cpuset,
  - new namespace ?
- Hierarchical or not ?