Beyond per-CPU atomics and rseq syscall: subset of eBPF bytecode for the do_on_cpu syscall

EfficiOS

mathieu.desnoyers@efficios.com

Restartable Sequences (RSEQ) in a nutshell

- System call registering user-space TLS data,
- TLS data acts as ABI between kernel and user-space,
- Enables user-space to implement efficient per-CPU data accesses.

The need for a system call fallback to RSEQ

- Concurrent update of remote user-space per-CPU data,
  - Aware of CPU hotplug,
- Early/late per-CPU data use in libc initialization and thread life-time,
- Single-stepping through RSEQ with existing debuggers.

```
SYSCALL_DEFINE5(do_on_cpu,
                struct bpf_insn __user *, ubytecode, u32, len,
                int64_t __user *, uresult, int, cpu, int, flags)
```

do_on_cpu RSEQ fallback requirements

- Not a fast-path,
- Large number of eBPF programs can exist in user-space memory:
  - Preloading them into the kernel is impractical wrt memory consumption,
- Received as parameter from a system call for single-use,
- Execute on a specific CPU received as parameter,
- Preemption disabled critical sections (exclusive per-CPU data access),
- Only access user-space memory and interpreter registers: ***may fault with preemption disabled***.

do_on_cpu runtime interpreter

- Upstream Linux eBPF infrastructure not useful for do_on_cpu:
  - Load/store of stack, kernel data,
  - All calls to external functions,
  - Most of eBPF verifier,
  - eBPF bytecode to native code JIT,
- Currently, do_on_cpu implements its own:
  - Bytecode validation,
  - Bytecode interpreter (with loops support),
  - User-space to kernel memory mapping translation.

EfficiOS

Additional eBPF extensions required

- Define an eBPF memory model,
- New instructions specifying memory ordering:
  - Load-acquire,
  - Store-release,
  - Memory barrier,
- Preemption disable/enable:
  - Allow disabling preemption for short bounded critical sections,
  - Minimize scheduler latency impact for preempt-RT.

EfficiOS

Additional Slides (if required by discussion)

- Handling page-faults with preemption disabled,
- Handling execution mismatch between passes.

Handling page-faults with preemption disabled

- Multi-pass scheme:

1) Create kernel mapping of memory:
    - Grab reference to each user-space page touched by bytecode,
    - Create vmap aligned on same page colour as user-space pages (for virtually-aliased architectures),
    - Enable preemption and restart bytecode interpretation each time a new page is added to the set,
2) Perform store side-effects.

Handling execution mismatch between passes

- Caused by changes in data loaded from user-space (tainted register):
  - Address for load/store from/to user-space memory,
  - Conditional branch,
- Handling of changes detected within pass (2) (store side-effects):
  - Restart if change detected before any store side-effect,
  - Return EIO (corruption detected) if change detected after side-effect is visible to user-space.

EfficiOS