Update on futex2

LCA 2020

André Almeida

Kernel Developer

andrealmeid@collabora.com

Overview

- What’s futex?
- Why do we need futex2?
- Current status
- Current results
- Next steps

Fast Userspace Mutex

- System call for creating sync primitives in userspace (e. g. mutexes, semaphores)
- Kernel just provide ways to sleep/wake thread, all the logic is done in userspace
- Created in 2002, no new features since 2008
- Modern workloads requires new functionalities

Why do we need futex2?

- Current code in “maintenance mode”, no new features/redesign will happen
- Legacy features, fragile code, hard to test and track for regressions
- Limitations: no NUMA awareness, only 32bit sized futexes, wait on a single futex

Why do we need futex2?

- Solution: new interface
- Fix previous limitations
- Add new functionalities
- No more multiplexing
- Code (mostly) from scratch

Current status

- Features completed: wait, wake, waitv, timeout, shared futexes
- Selftests and perf tests ported to futex2
- Ported Wine/Proton to use futex2

Current status

- Waitv: also known as “Futex Wait Multiple”
- A single waiter can wait for multiple futexes
- Similar to WaitForMultipleObjects from WinAPI
- Work sponsored by Valve to get Windows games running faster on Linux

Current results

- Performance: comparing with original futex
  - Hash operations/sec: +2.84% operations
  - Wake calls: -4.89% time to complete
  - Wake-parallel: -13.06% time to complete
  - -3% kernel cycles on futex while running games
- Stability: futex2 can run modern AAA games
  - 42k futex2 calls/sec

Next steps

- Implementing remaining features
  - NUMA, variable size, requeue
- More testing
- Upstreaming: RT tree

Thank you

```
Message {
  config {
    priority: "high"
    body: "Collabora is hiring"  // Many open positions
    recipient: "you"             // Please join us
    calltoaction: "http://col.la/join"
  }
}
```