



Linux
Plumbers
Conference | Richmond, VA | Nov. 13-15, 2023

Userspace adaptive spinlock with rseq

André Almeida, Igalia
Mathieu Desnoyers, EfficiOS



Use case

- Calling a `mutex_lock()` -> `futex()` requires a context switch
- Context switch can be more expensive than the critical section ("microcontention")
- Some apps have a complete userspace lock implementation without syscalls to avoid this cost
- Let's allow userspace to correctly spin!
- `rt_mutex` uses adaptive spinlock



The challenge

- A lock contender should spin if the lock owner is running
- Or sleep otherwise
- **How can we tell an userspace thread is running or not?**
- In a very fast way, without syscalls



rseq

- rseq code is already integrated with task scheduler and has a fast uAPI
- No syscalls to read information set by the kernel
- In a similar fashion, rseq was reused for getcpu as well



sched_state

- A new field for struct rseq: sched_state
- Updated by the kernel when process is scheduled out and in
- Now, the only thing left for userspace is to check if a thread is on a given CPU or not.
- If it is, it can safely spin, else, it goes the normal path (sleep using futex())



rseq struct cache line

```
struct rseq_sched_state {  
    __u32 version;  
    __u32 state;  
    __u32 tid;  
};
```

```
struct rseq {  
    [...]  
    __u32 mm_cid;  
  
    __u32 padding1;  
    __u64 sched_state_ptr;  
  
    char end[];  
}
```



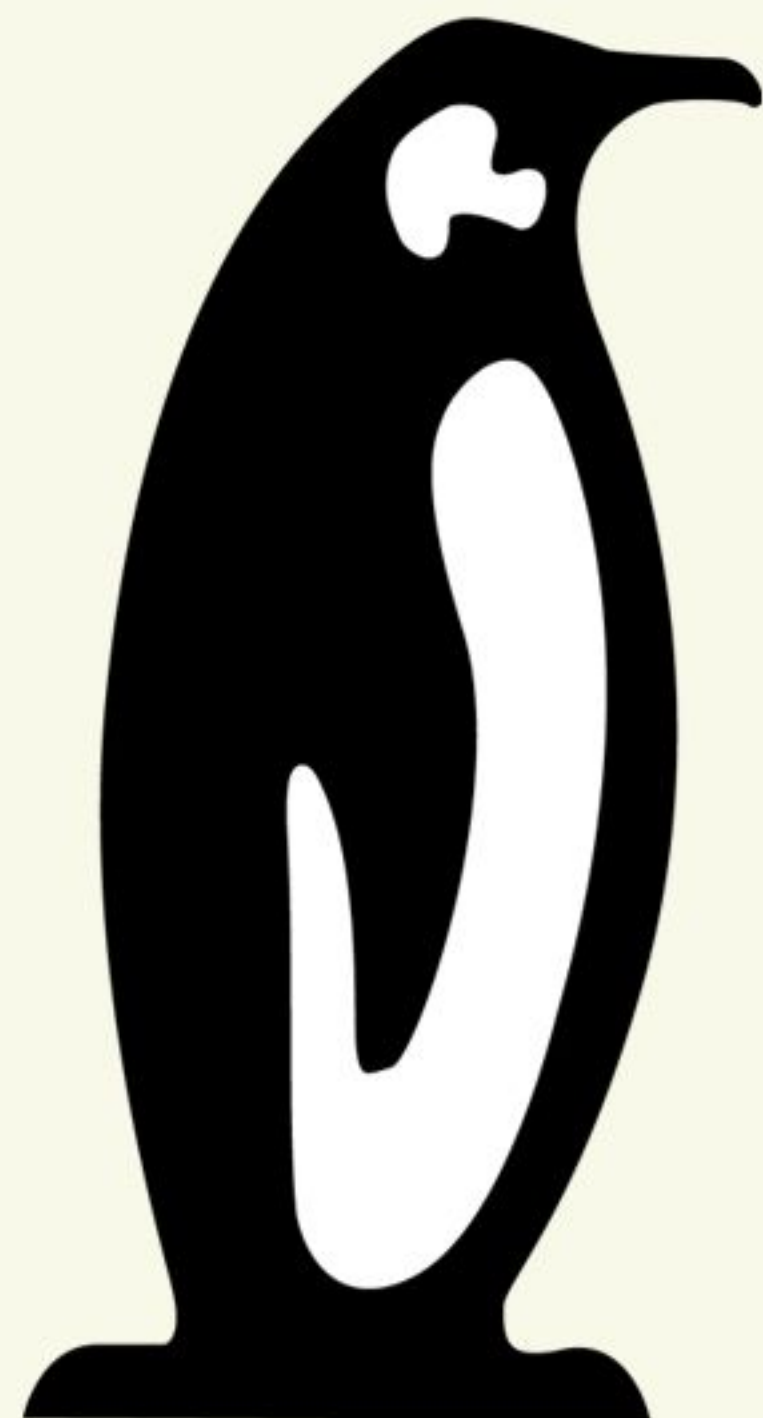
robustness?

- This interface requires shared thread information
- Before a thread dies, it free the lock in the exit path
- Barriers around the sched_state ensure that every reader is reading something meaningful
- However, this mechanism isn't robust for shared memory/multi process
- But we don't even have access to other processes' thread area in the first place
- **Is this a concern?**



Correctness

- Scheduler cannot take a page fault, so there's no guarantee that the sched_state **is always correct**
- **Statistically it should work** anyways, it's an edge case
- Worse case: it gets faulted in on next return to userspace
- Workaround: hook into the page fault handler, and populate it with new data when it gets faulted in for read



Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023

