Read Write Semaphores

- Allows for multiple readers and only one writer
- They are fair locks
  - New readers will block if a writer is blocked
Read Write Semaphores

- Real Time converts them to a simple mutex
- Serializes readers
  - mainline can run parallel
- Affects various work loads drastically
- Note, mainline can be forced to serialize readers if a writer is blocked
  - Remember, they are fair locks
Read Write Semaphores

- Biggest culprit for performance issues
  - mmap_sem
  - Page faults
  - Lots of threads (Java!)
  - Peter Zijlstra has worked to avoid taking mmap_sem on page faults
- There may be other areas where rwsems are bad
Read Write Semaphores

- Priority inheritance is hard
- Doing PI for multiple tasks is even harder
  - was done before and was really complex
  - Tried to keep the fast past
    - use cmpxchg() to grab lock quickly when uncontended
Read Write Semaphores

- Priority inheritance is hard
- Doing PI for multiple tasks is even harder
  - was done before and was really complex
  - Tried to keep the fast past
    - use cmpxchg() to grab lock quickly when uncontended
- “train wreck!” - Thomas Gleixner
Read Write Semaphores

- Revisit Priority Inheritance
- Forget the fast path (rwsem's suck anyway)
- Greatly simplifies the algorithm
  - All must take the internal spinlock before taking lock
- But still complex, but reasonable