Ticket to a Tar Pit

Jeremy Fitzhardinge
jeremy@goop.org
jeremy.fitzhardinge@citrix.com
Spinlocks are awesome

- Great way to synchronize
- Almost always very low cost
- Straightforward to use
Spinlocks are Awful

- Spinning is a waste of time
- Spinning for no reason is a complete waste
- Why spin for no reason?
Old-style byte locks

• Not fair
• First CPU to check lock wins
• Locking:
  
  while (test_and_set(&lock))
  
  relax();

• Unlocking:
  
  lock = 0;
Ticket Locks

• Guaranteed FIFO granting of lock
• Introduced in 2.6.24
• Basic lock algorithm:
  myticket = claim_ticket(&lock);
  while (!my_turn(&lock, myticket))
    relax();
• Unlock:
  grant_ticket(&lock, myticket + 1);
Lockholder Preemption (LHP)

• If a VCPU has no PCPU while holding a lock everyone else wastes time
• Variation of priority inversion
• Can be annoying source of inefficiency, but not a box-killer
• Applies to all spinlock implementations
Lock Claim Scheduling

• Big problem when releasing a lock:
  – How to make sure next person gets CPU?
  – VCPU scheduler doesn't know
• Can easily get to 90%+ time spent spinning
Paravirtualizing Spinlocks

• Current approach: completely replace spinlocks
• pv_lock_ops intercepts:
  – spin_lock
  – spin_unlock
  – spin_trylock
  – spin_is_locked
  – spin_is_contended
Xen PV Spinlocks

- `spin_lock`: spin for a while, then block on event channel
- `spin_unlock`: unlock, then check to see if anyone blocked
  - If so, kick them with an event
  - event never delivered; just a blocking poll operation
- Per-VCPU array of who's waiting on what
  - Checking = linear scan
- Keep counter of waiters in lock
Downsides of PV Spinlocks

• Adds indirection to all lock operations
  – Better than an indirect call, but still an extra call
  – Measurable performance hit on some architectures

• Completely new lock implementation
  – Old-style lock
  – Different characteristics from native lock
  – Sleazy hack in relying on same initializer
Paravirtualized Ticket locks

• Leave fast-path of ticketlocks intact
• Only put pv-ops in the slow path
  – lock_spinning
  – unlock_kick
• Removes a layer of complexity in common code
• Much less per-hypervisor code
Lock Details

- myticket = claim_ticket(&lock);
  for (;;) {
    int count = THRESHOLD;
    do {
      if (my_turn(&lock, myticket))
        goto out;
      relax();
      out:
    } while(--count);
    pv_lock_ops.lock_spinning(&lock, myticket);
  }
Unlock details

• next = lock->tail + 1;
  grant_ticket(&lock, next);
  if (are_waiters(&lock))
    unlock_kick(&lock, next);

• Implementing are_waiters()
  – Check for any queued lock
    • Unchanged lock size, but lots of spurious kicks
  – Add “waiters” counter to lock
    • Fewer kicks, but increase lock size
Xen PV ticketlocks

• Per-VCPU vars of which lock, and which ticket
• lock_spinning records lock+ticket for VCPU, blocks on event channel
• unlock_kick scans for matching lock+ticket and kicks any it finds
Performance

• Very preliminary numbers
• +1 - -2% on native vs no PV ticketlock
  – About the same as PV spinlocks
• About the same as PV spinlock under Xen
  – Same good properties
• Overall, a bit disappointing
• Still seems like a better approach architecturally
• (What's a useful benchmark?)
Look, A Graph!

System Time Percent

- Blue: system max load
- Red: system optimal load
- Orange: system half load