Cleaning Up Linux’s CPU Hotplug For Real Time and Energy Management

Thomas Gleixner (Linutronix)
Paul E. McKenney (IBM Linux Technology Center assigned to Linaro)
Vincent Guittot (ST-Ericsson assigned to Linaro)
CPU Hotplug Offline Process

- CPU_DOWN_PREPARE
  - Succeeded (CPU_DEAD)
  - Failed (CPU_DOWN_FAILED)

- CPU_DYING
  - Succeeded
  - Failed

- CPU_POST_DEAD

- All other CPUs spinning with interrupts off while outgoing CPU is in CPU_DYING notifiers

Notes:
- cpu_hotplug.lock held
- cpu_add_remove_lock held
CPU Hotplug Online Process

CPU_UP_PREPARE

- Succeeded
- Failed

CPU_STARTING

- Succeeded
- Failed

CPU_ONLINE

- cpu_hotplug.lock held

- cpu_add_remove_lock held
CPU Hotplug Is Not Atomic

Valid notifier order for online:
- IPIs
- RCU
- Scheduler
Must reverse order for offline

Reality will intrude...
- RCU depends on scheduler
- Circular dependency!
Must further decompose RCU and scheduler interaction
Source of CPU-Hotplug Latency

Parking Per-CPU kthreads

• Thomas Gleixner patchset:
  • kthread_create_on_cpu(...)  
  • kthread_should_park(void)  
  • kthread_park(struct task_struct *k)  
  • kthread_unpark(struct task_struct *k)  
  • kthread_parkme(void)  
  • smpboot_register_percpu_thread(struct smp_hotplug_thread *plug_thread);  
  • smpboot_unregister_percpu_thread(struct smp_hotplug_thread *plug_thread);  
  • smpboot_thread_check_parking(struct smpboot_thread_data *td);  

• This approach should remove per-CPU kthread creation overhead  
  • Also move notification to kthreads, offline in kthread
for_each_online_cpu()

- More than 300 uses of this primitive
  - Too many to reliably classify by hand
- Silas Boyd-Wickizer automating classification
  - Uses “sparse” static-analysis tool for Linux kernel
  - Leverage sparse “address spaces”
    - Identify usage contexts
    - For example, uses protected by get_online_cpus() need not change when CPU offline moves away from stop_machine()
Alternative Approaches Considered

- Continue using existing CPU hotplug
  - Not feasible: slow, unreliable, disruptive to real time
- Modify CPU hotplug to reverse offline notifier order
  - Doesn't help with kthread creation overhead
- Dump CPU hotplug in favor of something new
  - Still need to clear all current and future work from each CPU, so similar complexity required
  - Plus still need CPU hotplug for failing hardware
Possible Issues With Approach

- Old-style interrupt controllers
- Scheduler-RCU circular dependency
- Early-boot initialization (before kthreads can be created)
- X86 MTRRs on hyperthreaded systems still require quiesce (but faster than CPU_DYING)
- Scanning online CPUs and changing for_each_online_cpu() semantics