Shift the Last CPU: CPU0 Hot Plug

Fenghua Yu <fenghua.yu@intel.com>

Intel

Outline

- Introduction
- CPU0 Hot Plug Design
  - AP Hot Plug Kernel Path
  - Remove Assumptions of Not Hot Pluggable CPU0
  - Wake up CPU0
- Status
- Limitations
- Future Work
- References

Terms

- Usage of CPU hot plug and offline/online may not be very consistently used in Linux.
- To reduce confusion, the following terms are used in this presentation:
  - CPU hot plug: Hot add or hot remove a CPU during OS run time.
  - CPU (logical) offline/online: Same as CPU hot plug
  - Physical CPU hot plug: Physically hot add or hot remove a CPU. This needs BIOS hooks and something like attention button on the platform

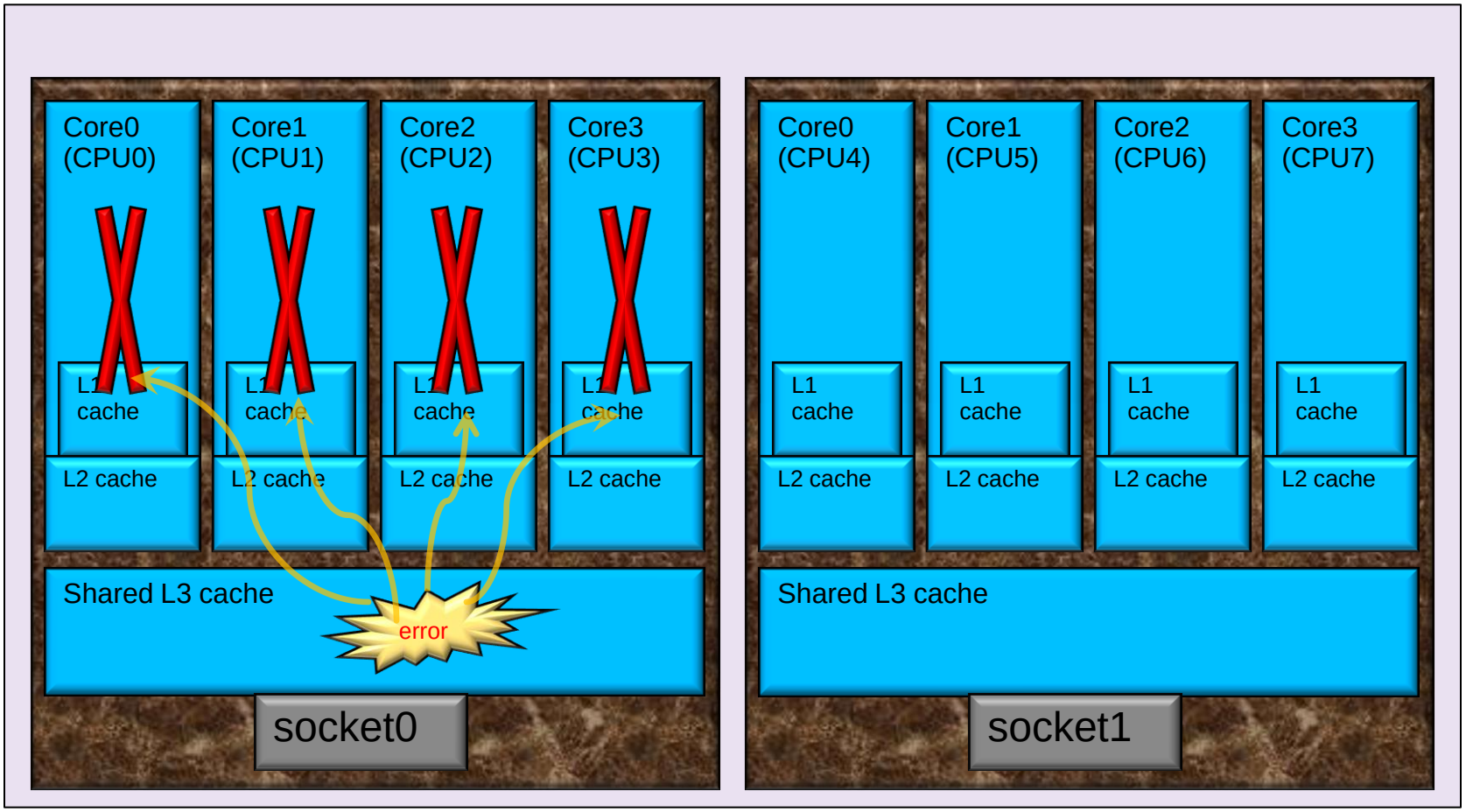
Introduction

- CPU0 or BSP (Bootstrap Processor) is the first CPU that starts Linux kernel.
- All other CPUs booting after CPU0 are APs(Application Processors). APs are named CPU1, CPU2, ….
- In 3.7 and older kernels, only APs can be hot plugged on x86.
- CPU0 or BSP has been the last processor that can not be hot pluggable on x86 platforms.
- This presentation will discuss CPU0 or BSP online and offline and how to remove this obstacle to CPU hot plug.

Why CPU0 Hot Plug?

- RAS Feature:
  - If socket0 needs to be hot plugged for any reason (any thread on socket0 is bad, shared cache issue, uncore issue, etc), CPU0 is required to be offline or hot replaced to keep the system running
  - Hot pluggable CPU0 is getting more useful in multi core era when CPU0 has more coupling with other components in a socket

An Example of CPU0 Hot Plug Usage

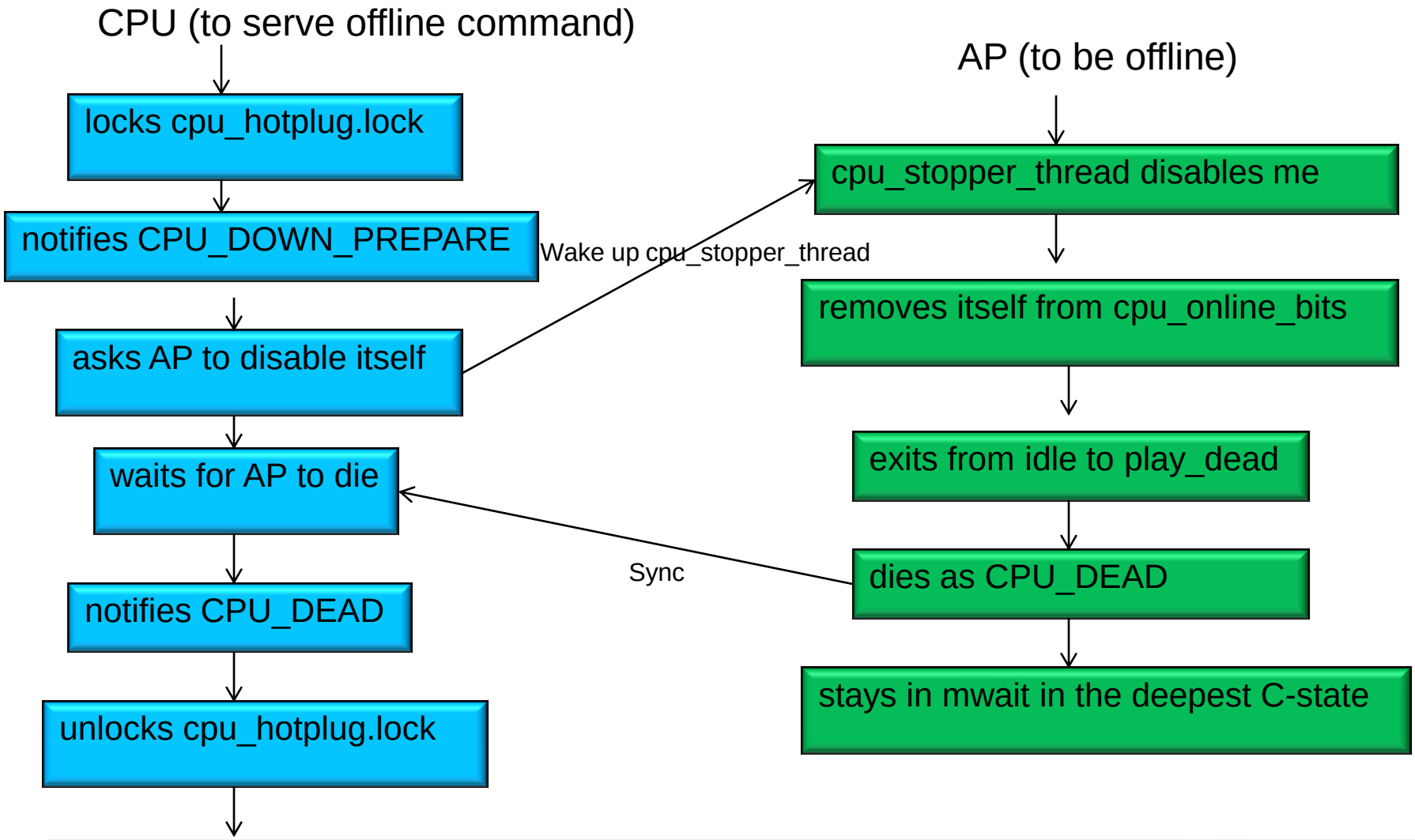


A yellow status error in shared L3 triggers CPU0~3 offline in socket0

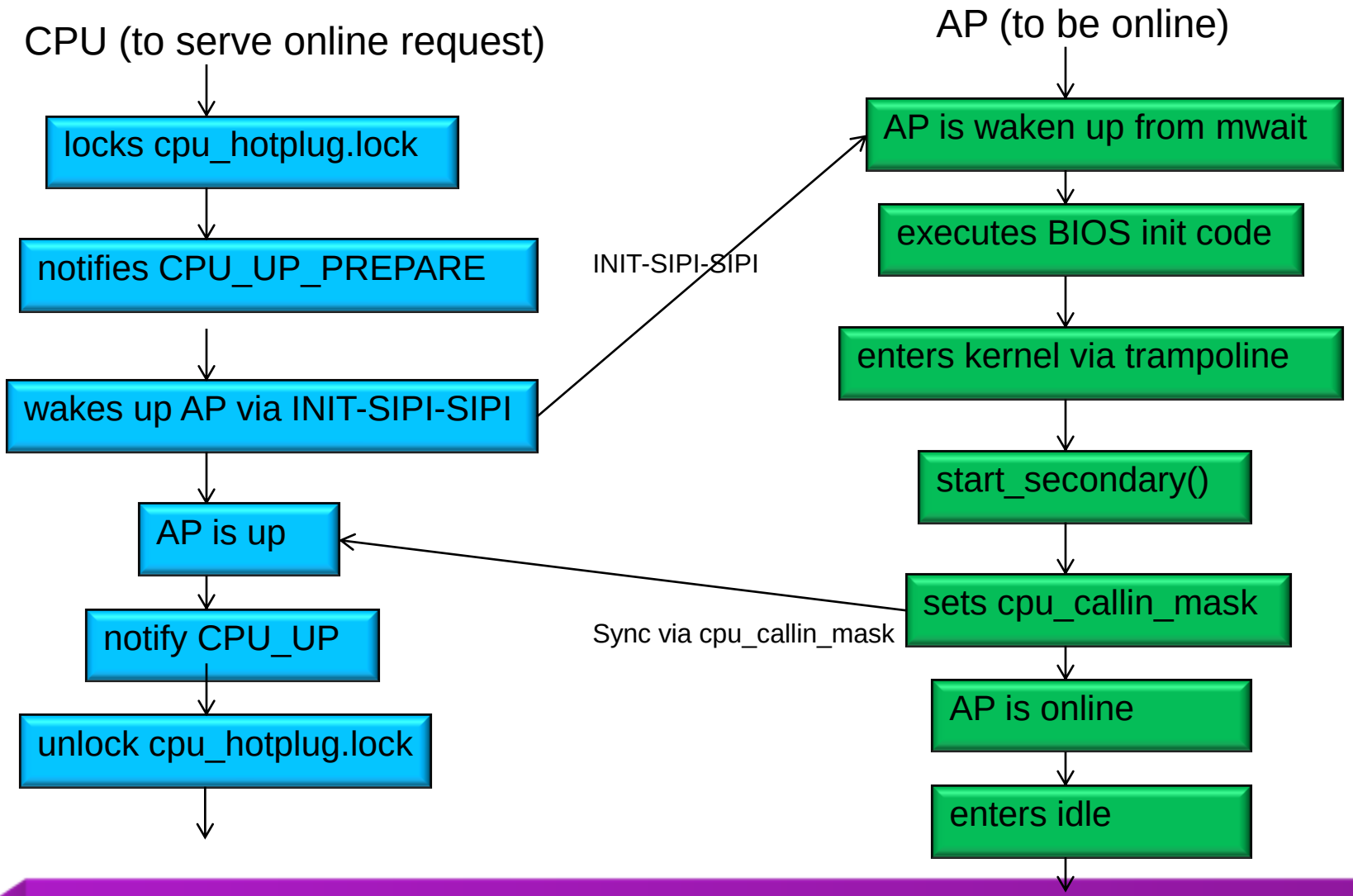
CPU0 Hot Plug Design

- We don’t introduce brand new method to hot plug CPU0
- Instead, we fit CPU0 hot plug code into existing method of AP hot plug by
  - eliminating the implicit assumption that CPU0 is not hot pluggable
  - and solving issues when CPU0 becomes hot pluggable
- In the next two slides, we will review AP hot plug kernel work flow before describing CPU0 hot plug

Simplified AP Offline Work Flow



Simplified AP Online Work Flow



Things to Do for Hot Pluggable CPU0

- CPU0 hot plug design is based on AP hot plug method
- To handle hot pluggable CPU0, we need to:
  - remove the assumption that CPU0 is not hot pluggable
  - fix issues when CPU0 becomes hot pluggable
  - contain limitations of this feature

CPU0 Is Hot Pluggable

- CPU0 is hot pluggable when there is no PIC mode irq on the platform
  - irq in PIC mode can only be serviced by CPU0
  - irq in IOAPIC mode can be serviced by any CPU
- CPU0 is hot pluggable on modern platforms
  - Modern platforms don’t have PIC mode irq any more.

CPU0 Is Hot Pluggable (cont.)

- CPU0 is set up as hot pluggable if there is no irq dependency:
  - Its online interface in sysfs is created
- CONFIG_BOOTPARAM_HOTPLUG_CPU0:
  - Sets default setting of cpu0_hotpluggable
  - Can enable CPU0 hot plug by opt-in kernel parameter “cpu0_hotplug”

Remove CPU0 Offline Assumption

- On AP hot plug path, there is an assumption that CPU0 can not be offline once system boots.
- We remove the CPU0 assumption to offline CPU0:
  - CPU0 can be disabled in native_cpu_disable()
  - Enable x2apic in cpu_init() on CPU0
  - Set numa node in cpu_init() on CPU0

Remove CPU0 Online Assumption

- Similarly, on the CPU online path, there is an assumption that CPU0 can not be online again once system boots.
- Remove the assumption to enable CPU0 online:
  - CPU0 can be online in native_cpu_up()
  - Store cpu info for CPU0 in identify_secondary_cpu(c) when it’s online.
  - Init thread xstate only once to avoid overriding xstate_size when CPU0 is up after offline

Find a Substitute When CPU0 is Offline

- In a few places, kernel always asks CPU0 for services.
  - With our design, kernel can not assume CPU0 is always online any more.
- Instead of always asking CPU0 for service, kernel asks the first available online CPU to do that:
  - Ask the first available online CPU to retrigger irq in ioapic_retrigger_irq()
  - Ask the first available online CPU to save mtrr in mtrr_ap_init()

Wake Up CPU0 from Offline

- Wake up CPU0 from offline via NMI:
  - CPU0 can not be waken up vi INIT-SIPI-SIPI sequence because BSP will execute the BIOS boot-strap code which is not a desired behavior
  - To avoid the BIOS boot-strap code, wake up BSP via NMI
    - Could wake up BSP via writing to monitored address…
- NMI can only wake up logically hot removed BSP:
  - For physically hot adding CPU0, we need another waking up method when real platform and request are available.

Do not Suspend/Hibernate While CPU0 Is Offline

- Suspend (S3) or hibernate (S4) can not be executed if CPU0 is detected offline:
  - Because x86 BIOS requires CPU0 to resume from sleep
- To successfully resume from suspend/hibernate, CPU0 must be online before suspend or hibernate:
  - Suspend or hibernate will fail and system can not go to S3 or S4 if CPU0 is offline

Debug BSP Online/Offline

- CONFIG_DEBUG_HOTPLUG_CPU0 is for debugging the CPU0 hot plug feature:
  - The switch takes down CPU0 as early as possible and boots user space up while CPU0 is offline.
  - User can online CPU0 back after boot time.
  - Default value of the switch is off.
  - Safe and earliest place to take down CPU0 is after all hot plug notifiers are installed and SMP boots.

Patches Status

- All patches were merged into the upstream 3.8 kernel.

Limitations of CPU0 Hot Plug

- Currently only CPU0 logical online/offline is supported
  - CPU0 needs to handle SMI
- Resume doesn’t work if BSP is offline
  - BIOS needs CPU0 to respond to resume interrupt

Future Work

- To remove the limitations, platform and BIOS are required not to bind BIOS services to BSP:
  - Handling SMI is not restricted to a specific BSP
  - Resume is not restricted to a specific BSP

Acknowledgements

Tony Luck, Asit Mallick, H. Peter Anvin, Bruce Schlobohm

(Intel SSG/OTC)

References

[1] Intel 64 and IA-32 Architectures Software Developer's Manual (Volume 1, 2, 3)

[2] Linux kernel source tree

[3] The BSP hot plug patches can be found at: https://lkml.org/lkml/2012/11/13/782

Backup