dominicrath
Beginner

05-27-2010 02:59 AM
3 Views

Halting/Restarting a BSP

Hi,

I hope this is the right forum for a question like mine.

In the process of writing some experimentation software for multi-core systems I wanted to be able to selectively halt and restart individual "logical processors" (threads, cores). The code currently runs "bare metal", i.e. with no OS, started like a boot-loader. Halting a logical processors should be possible without the cooperation of the victim processor, i.e. having a defined "shutdown" flag or a "shutdown" IPI isn't an option, because the processor might ignore the shutdown request.

Halting and restarting an AP (application processor) can be achieved easily by sending the AP an INIT IPI (puts the AP into wait-for-SIPI state) followed by two STARTUP IPIs (lets the AP begin executing at vector << 12). By using the warm-reset vector method described below as a safe-guard this should work for all processors, including systems with a discrete APIC that ignore the STARTUP IPI (which means they start executing immediately after an INIT, I believe).

The situation is more difficult for a BSP (bootstrap processor). By programming CMOS/RTC RAM offset 0xF with shutdown code 0xA and the warm-reset vector 40:67h in the BDA with the address of some code I want to restart from I can achieve the same for BSPs. This works fine on multiple systems including a D945GCLF2 with Intel BIOS, an IMEI board with Core2 Quad processor and AMI BIOS and in simulated platforms using VMWare and BOCHS.
Unfortunately it doesn't work on a Fujitsu Celsius R570 workstation with dual Xeon-5520 processors and a Phoenix SecureCore BIOS. The BIOS in the Celsius apparently ignores the shut-down code and resets the whole system after restarting execution following an INIT sent to the BSP.

What I'm looking for is a way to achieve a halt followed by a restart of a BSP without affecting the APs. The restart shall be initiated from one of the APs, persumably by sending an IPI to the BSP.

It looks like the warm-reset vector at 40:67h isn't used by modern BIOSes (at least by the one in the Celsius R570). Is there an any alternative method?

Best Regards,

Dominic Rath

0 Kudos

Share Reply

All forum topics < Previous Topic Next Topic >

4 Replies

jimdempseyatthecove
Black Belt

05-27-2010 05:57 AM
3 Views

Dominic,

Can you flip the role playing around?

Have the AP assume the role of the BSP and BSP assume the role of the AP.

A potential problem with this is APIC numbering would be incorrect for code examining APIC numbers to inquire what processor it was running on. If this is problematic, then potentially you could reassign the APIC numbers when you swap roles. As to how you would do this, I haven't a clue. It is something you can look into. The restart would have to bypass the BIOS hard restart as this would undo your intentions. Some of the system boards and BIOSs may expressly inhibit such activity since this could be used as an exploit by a virus.

Jim Dempsey
http://www.quickthreadprogramming.com

0 Kudos

Share Reply

dominicrath
Beginner

05-27-2010 07:03 AM

Hey Jim,

thanks for your reply. I'm not completely sure what you mean by flipping the role playing around. Do you mean "have the BSP tell the AP to halt/restart"? Or do you mean "make one of the APs _be_ the BSP"? The latter would be great - I just couldn't find anything on how to accomplish this. Apparently it should be possible, at least on some processors, as some chipset manuals say that the BIOS elects the BSP.

What I'm trying to achieve is an environment where it doesn't matter what part of the software runs on which core. Ideally the BSP wouldn't have to be treated special in any way - once it got the APs running for the first time they should all be equal.

The problem I'm trying to solve is that of catching a runaway-core, no matter what code it executed. The core might not have a valid IVT/IDT set up, or it might have interrupts disabled "forever" (cli; hlt;). Sending an NMI IPI would help, but only if the core has a valid IVT/IDT. Sending an INIT IPI works great, except that the BSP decides it should start executing at 0xFFFFFFF0 immediately, and apparently not every BIOS supports the warm-reset vector at 40:67h.

Regards,

Dominic

dominicrath

Beginner

05-27-2010 07:46 AM

Could it really be as simple as clearing the BSP flag in IA32_APIC_BASE? I'm still verifying my latest tests, but it looks like once the BSP flag in the MSR is cleared, the former BSP behaves like an AP, i.e. it enters wait-for-SIPI after an INIT, and honors the STARTUP message by branching to vector << 12.

Thanks a lot Jim!

Regards,

Dominic

jimdempseyatthecove
Black Belt

05-27-2010 07:53 AM

3 Views

It has been a while since I looked at the multi-processor boot so I may be totally wrong on the following:

I believe on hard boot that all cores start at 0x(FFFFFFFF)FFFFFFF0 with that area of memory initialized by BIOS (either as ROM or FLASH or...). I do not know of the APIC numbers have been assigned as of yet. The code there typically jumps to a secondary set of start-up that does your processor identification/arbitration, one of which takes (or is given) roll of BSP, the remainder processors fall into a monitor(location). Then BSP jumps or falls through tothe "single processor" bootstrap code.

What is unknown to me is if you are permitted to alter the code loaded at 0x(FFFFFFFF)FFFFFFF0 or at the secondary cold boot locations. If you can, then the trick (hack) will be to devise a reliable technique that can determine if the cold boot is actually a cold boot or if the cold boot is your run away core restart.

If the RDTSC is not effected by the software issued "cold boot" (note RDTSC may be different in different sockets), then you could potentially store the readout values in multiple locations, and your secondary start-up could test the current RDTSC against what was stored in those locations. If each stored location is within nnn ticks then assume this is coming from your restart of runaway processor. Note, this depends on the time stamp counter not being reset during this technique. If this isn't possible (due to reset) then write something unique into the keyboard buffer and read it back out in your secondary boot code. Or choose other device registers that persist across the restart and would have an almost impossibility of randomly containing your runaway processor restart signature.

Jim Dempsey

a10_ref BSP not compatible with Ubuntu 18.04
DongWang-BJTU
12-25-2019 01:04 AM

Creating BSP for Stratix 10
NPund
06-18-2019 06:02 PM

BSP for MinnowBoard MAX
Mustafa_A_Intel
08-27-2016 12:04 PM

BSP compilation is failing.
HSing22
01-02-2015 02:45 PM