Machine check handling on Linux

http://www.firstfloor.org/~andi/mce.pdf

Andi Kleen, SuSE Labs
ak@suse.de

What is a machine check?

- Hardware error

- Hardware is error correcting (ECC), but fails sometimes
- Internal errors, Memory, Cache, IO, Busses
- Can be an exception or "silent"
- IO errors can be caused by software (but usually not on PCs)
- NMI, Thermal

Why do we care?

- With bigger systems they become more common (64bit, clusters)

- With more transistors they will become more common
- Kernel developers and supporters need to know.
- Useful to diagnose hardware
- May predict failures early ("SMART for your memory")
- Some can be handled better than just panicing.
- Error reporting is even good for desktops

A live example (2.4)

kernel: Northbridge Machine Check exception = b60ea00100000813 0
ecc error
    link number 0
    err cpu1
    uncorrected ecc error
    processor context corrupt
    error address valid
    error enable
    error uncorrected
    previous error lost
    error address 00000001826ac018
Address: 00000001826ac018
CPU 1: Machine Check Exception: 0000000000000000
Kernel panic: Unable to continue

Why is it hard to handle them?

- Asynchronous - error happens later
- Imprecise - doesn’t report correct
- Ignore kernel locking
- Needs to be handled quickly and with minimum infrastructure
- Hard to test

History (PC centric)

- IBM PC had parity memory and caused NMI (Non Maskable Interrupt)

- Ignored IO errors (still a problem)
- Earlier x86 CPUs like Pentium 5 had simple machine check setups
- PPro added Intel generic machine check architecture
- Logged by BIOS in "server class" hardware
- First machine check handler for Linux/i386 by Alan Cox
- Currently split into different drivers
- 2.4 x86-64 handler with Opteron specific code

x86 machine check architecture

- Standard exception (18)
  - Standard registers (MSRs): address, status, misc
  - Banks with sub errors and own status (e.g. CPU, cache, bus, northbridge)
  - Some generic bits, but interpretation CPU specific
  - Silent errors - need a timer
  - Not integrated
- Thermal (using APIC vector)
- NMI for IO errors (optional)

User interfacing issues

- Looks like a software failure

- Users report it as software failure
- Need to separate them because they’re an different issue
- But software people still want to know
- Separate Log
- Disk logging after reset (NFI)

Old code issues

- Can deadlock in printk

- Would panic more often than needed
- Always in standard kernel log, often lost
- BIOS logging hard to manage

x86-64 2.6 rewrite

- Closely follows Intel/AMD recommendations

- Single driver for all CPUs.
- Lockless racefree logging infrastructure
- Logs binary log items
- Logs left over errors
- Tries to avoid panic, kills process
- Runtime configurable

User interface of the new code

- Cronjob runs mcelog to decode /dev/mcelog
- /sys/devices/system/machinecheck/machinecheck0/*
- tolerance level:
  - 0 always panic
  - 1 panic if deadlock
  - 2 try to avoid deadlock
- bankNctl
- check_interval
- oops=panic, panic=timeout

Future work: Handling RAM errors properly

- Use address to look up failed page using rmap
- Dirty/Kernel page: kill, Clean: reload, Free: ignore
- Needs to synchronize context
- Works better with imprecise errors
- May involve application using signals

Future work: IO error handling

- Enable NMI in chipset to report PCI level errors
  - Asynchronous delayed reporting makes it hard
  - Reliability
- Disable broken devices to prevent further damage
  - Better IO errors
- Driver debugging
  - Already works on non PC platforms to some extent (HP zX, IBM PPC, ...)
  - Callback or checking IO operations
  - NMI sources are not well defined
  - Would need an IOMMU to be really good

More work to do

- Simple NMI handling
- Thermal for x86-64
- Port it to 32bit x86
- Expand mcelog to decode more

URLs

http://www.firstfloor.org/~andi/mce.pdf

ftp://ftp.x86-64.org/pub/linux/tools/mcelog/

http://www.kernel.org