mcelog
Memory error handling in user space

Sept. 2010

Andi Kleen

Linux Kongress 2010

Trends

Many cores need more memory

Important workloads need a lot of memory

Maximum memory capacity growing quickly

More memory means more memory errors

Error basics

Uncorrected versus corrected errors
- Data is lost or
- Error happened, but was corrected

Hard versus soft (transient) errors
- Bit got flipped once or
- Bit is permanently broken

For corrected and soft errors trends are important
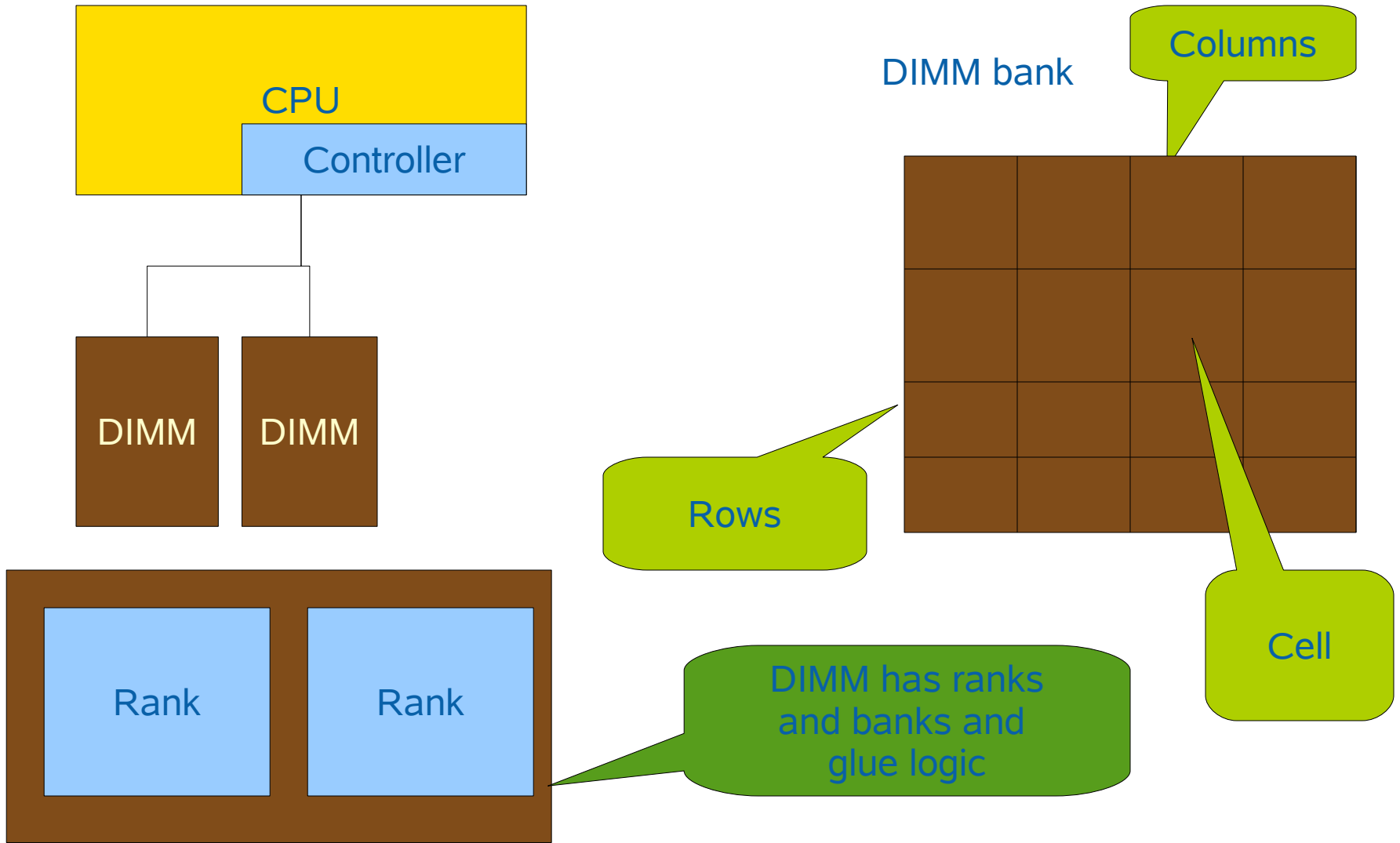- Corrected error rate is expected
- Better call them "events"

Clusters make uncommon errors common
- Large machines do to a lesser degree too

Memory overview

What can go wrong in memory?

Radiation flipping bits

Voltage fluctuations by the power supply

Silicon breaking down
   Cells, Columns, Banks, Dimms, Controllers

Lost information in data transfers

…

In general the more memory the more likely are errors
   And the trend is more and more memory

Cheap memory also has more errors

Longer uptime accumulates more soft errors.

A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility: U-Rochester
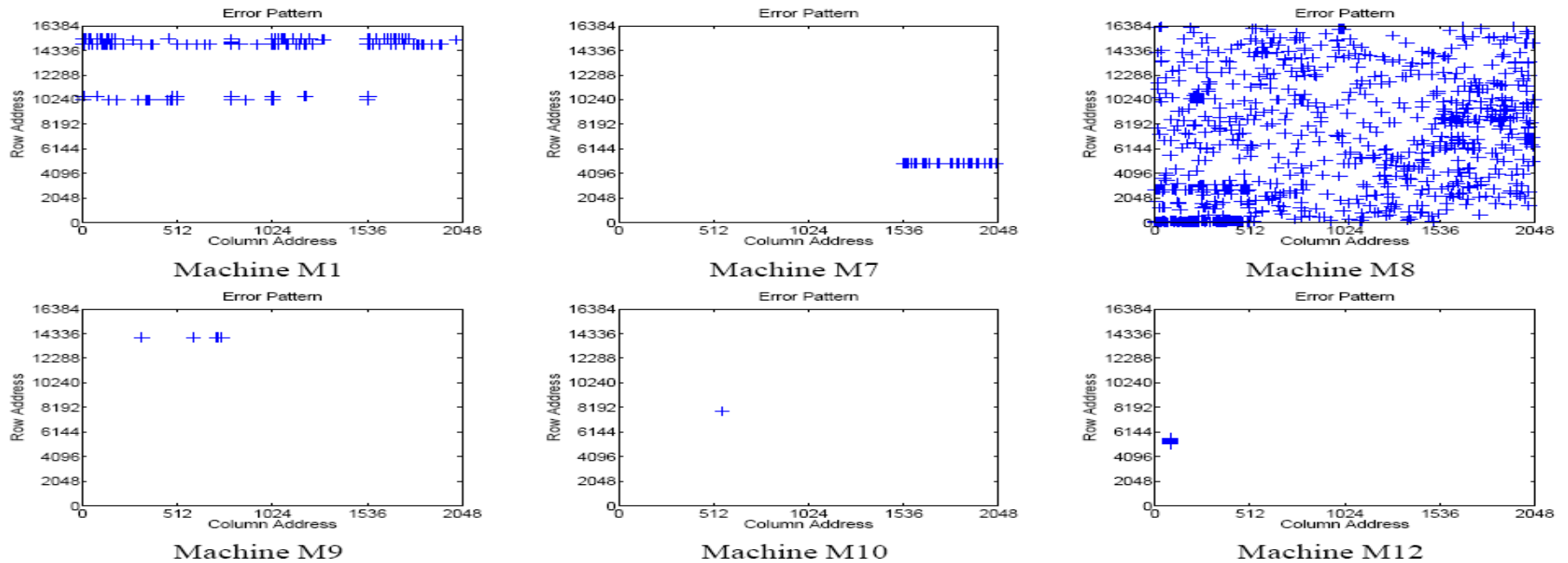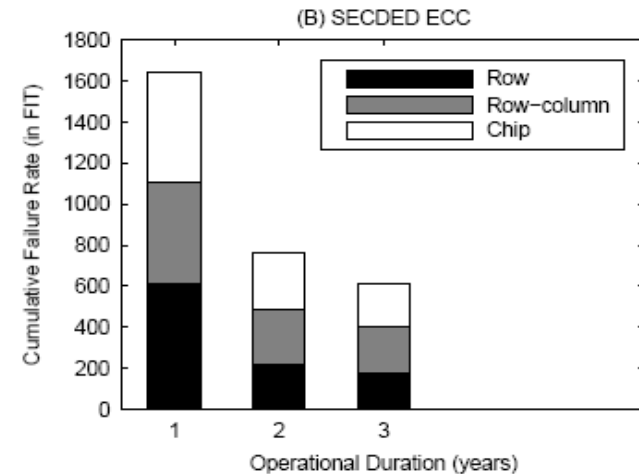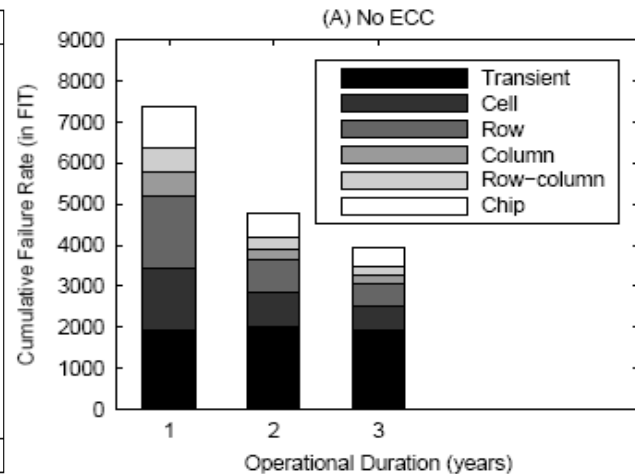## Evaluated 212 servers over 9 months

Figure 1. The visualization of error patterns on physical memory devices. Each cross represents an erroneous cell at its row/column addresses. The system address to row/column address translation is obtained from the official Intel

Machine	Single-cell	Row	Column	Whole-chip
M1	2	11	1	
M2		1		
M3	1 (transient)			
M4	1			
M5	1 (transient)			
M6				1
M7		1		
M8				1
M9		1		
M10	1			
M11	1			
M12			1	
Total	7 (2 transient)	14	2	2

Mcelog

User space daemon decodes, processes MCE events for known CPUs
   Both corrected and uncorrected

Standard component of 64bit x86 distributions for many years
   Now on 32bit too

Traditionally cronjob, now daemon
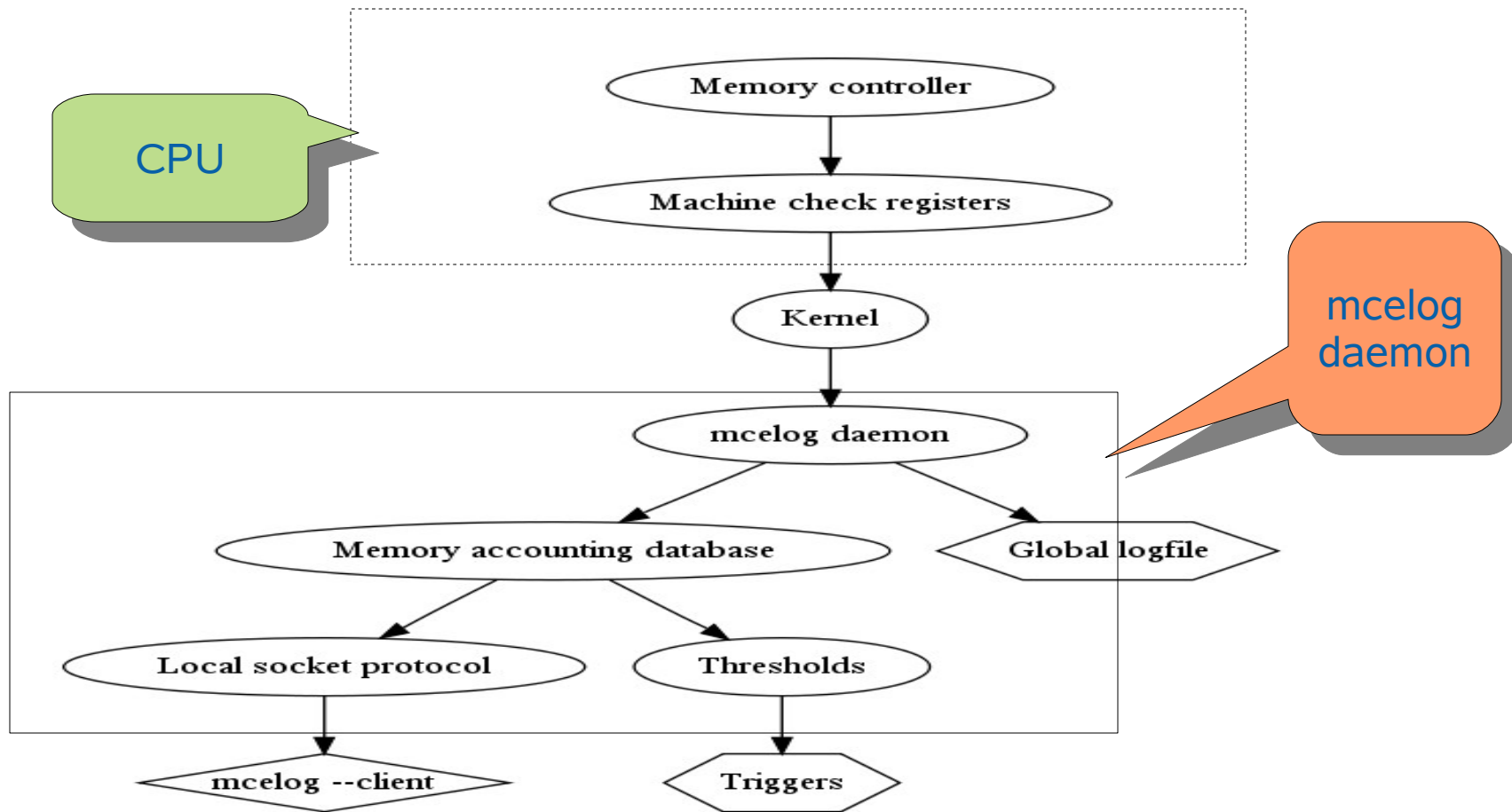   Daemon mode allows to keep state
   Many new features enabled by this

To get all the new features make sure to have the daemon running!

Can run offline to decode panic messages (mcelog –ascii)

mcelog needs to support the CPUs
   Not all features available with all CPUs
   Generally better with integrated memory controller

mcelog overview

Kernel architecture

Kernel machine check handler

Retrieves error information from hardware

Recovers or panics for uncorrected errors

Handles uncorrected errors directly

Modern kernels able to recover transparently from some cases

Or kill the process with the corrupted data

If error is not contained stop the system

Corrected errors are logged and passed to user space

Through /dev/mcelog error records

High level errors

Low level errors contain a lot of information

Needed by experts for analysis

But can be daunting to understand

Also not all errors are interesting

Interesting are error trends or serious errors.

Goal of mcelog is to provide higher level error analysis

/var/log/mcelog log file for details

"one liners" in syslog

Based on thresholds and specific actions

Predictive Failure Analysis

Watching corrected errors and try to guess future trends

Observation:

Hard uncorrected errors are often preceded by corrected errors

Stop using component in advance

All error counts have to be aged

Otherwise corrected transistent errors would lead to incorrect actions

Can be done for many units:

Memory page, DIMM, socket

Harddisks (e.g. SMART)

Full computer

Other hardware

Hardware memory protection mechanisms

Lowend:

No checksums for storage, limited protection of transfers

ECC: Detect 2bit, Correct 1bit (SECDED)

Needs ECC memory and server grade memory controllers
Scrubbing

Lockstep, Chipkill

Handle broken rank or DIMM with sophisticated encoding
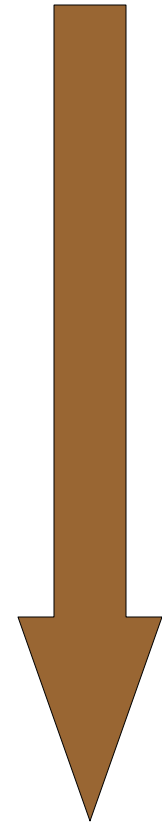Needs symmetric DIMM configurations, slower

DIMM, rank sparing

Highend: Full memory mirroring

Handles near all errors transparently
Needs twice as much memory

Cheap

Expensive

Software and Solutions Group

Stuck bits

Hardware redundancy handles most problems

Interesting case: single stuck bits in DIMM
   Corrected by ECC SECDED
   But error stays around in DIMM
   If there's another bit flip in unit it turns into a uncorrected (detected) error

Reasonably common occurrence
   Rest of the DIMM and system is often fine

Moving to a spare or mirroring can avoid these problems
   But it's a big hammer

Bad page offlining

Account errors per page

Accounting in daemon on demand (~64bytes/4K)

When page reaches error threshold in time period stop using page

Default offlining on 10 corrected events per 24h

Avoids triggering on transient soft errors

Handle stuck bits efficiently in software

Does not work well for larger scale corruptions

But handles common situation without impacting any redundancy

Compliments hardware memory protection and extends coverage

Avoids service interruptions: DIMM with a few stuck bits can be used without significant loss

Kernel soft offline code

mcelog daemon asks kernel to stop using a page

Transparent to applications, no killing

Kernel soft offlines page
- User data is migrated transparently to a new page
- Cached data is dropped
- Free pages are marked bad

Goal is to be able to offline 90+% of pages in common workloads
- Will never be 100%

Studies using "page-types" tool for different enterprise workloads
- Largest gap was huge page support, now nearly addressed

Basic code available in 2.6.33 or SLES11-SP1

Extended Cache Error Handling

x86 CPUs have large caches

Caches have internal error detection and correction mechanisms

CPU reports when redundancy runs low

Does not necessarily mean CPU is broken, could be external causes
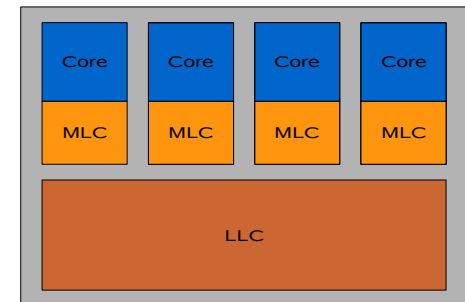But caches with no redundancy may have unrecoverable errors in the future

mcelog offlines cores using these caches

Stop using cores to stop using the caches
System will continue running with reduced capacity
Logs warning to system log

Can be customized with cache-error-trigger

Error accounting

Individual errors are not that interesting
- Errors often come in bursts and individual errors in a burst are not interesting
- Large clusters can generate a lot of data, which is difficult to process

Daemon accounts memory errors per component
- Currently DIMM*, socket, page*
- Reports "n errors in last x hours on component k"
- Triggers when thresholds are exceeded
- Accounting in memory only

Option to disable individual error logging for less data
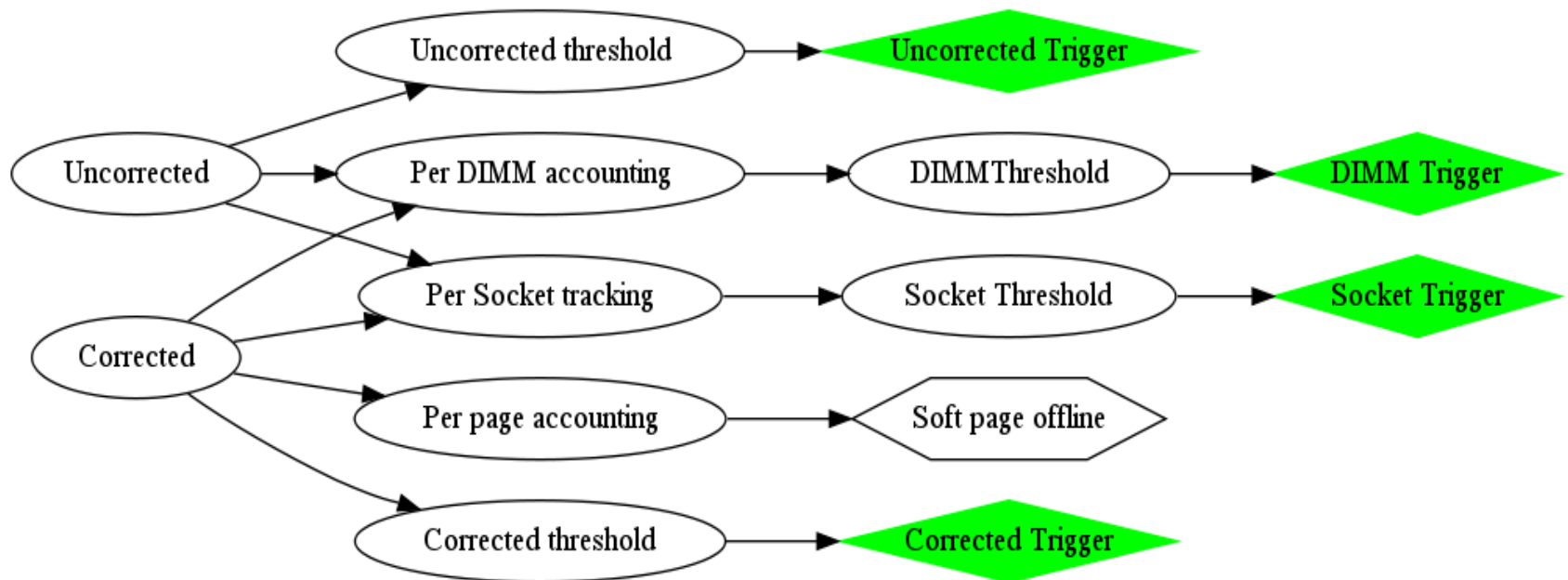- Can be important in clusters!

*If reported by CPU

Querying the database for memory errors

```
# mcelog --client
Memory errors
SOCKET 0 CHANNEL 2 DIMM 1
corrected memory errors:
    3 total
    3 in 24h
uncorrected memory errors:
    0 total
    0 in 24h

SOCKET 0 CHANNEL 0 DIMM 0
corrected memory errors:
    3 total
    3 in 24h
uncorrected memory errors:
    0 total
    0 in 24h
```

Triggers

mcelog DIMM trigger variables

```
THRESHOLD        human readable threshold status

MESSAGE          Human readable consolidated error message

TOTALCOUNT       total count of errors for current DIMM of CE/UC

LOCATION         Consolidated location as a single string

DMI_LOCATION     DIMM location from DMI/SMBIOS if available

DMI_NAME         DIMM identifier from DMI/SMBIOS if available

DIMM             DIMM number reported by hardware

CHANNEL          Channel number reported by hardware

SOCKETID         Socket ID of CPU that includes the memory controller with the DIMM

CECOUNT          Total corrected error count for DIMM

UCCOUNT          Total uncorrected error count for DIMM

LASTEVENT        Time stamp of event that triggered threshold (in time_t format, seconds)

THRESHOLD_COUNT Total umber of events in current threshold time period of specific type
```

Local trigger use cases

Simple
- Page administrator
- Alert network management system
- Blink the red LED
    - What else is the LED subsystem good for?

Complex
- In a VM cluster migrate VM away
- In a HA cluster trigger a failover

Other clever solutions?

Problems
- Finding a good threshold
- Depends on the memory used and environment
- Triggering too early can be expensive and cause unnecessary disruption

Future

Network management interface

Offlining of more page types

APEI support for chipset errors

Accounting for more components
   PCI devices, other devices

More database functionality for errors

Better support for error tree analysis

More space efficient data structure for page accounting
   Would be an interesting little project if someone is interested please contact
      me.

Ideas and contributions welcome

Resources

Intel Software Developer's Manual: 3A/B System Programming Guide
  Description of the x86 Machine check architecture

git://git.kernel.org/pub/scm/linux/kernel/git/ak/linux-mce-2.6
  MCE development tree (hwpoison, mce4)

git://git.kernel.org/pub/scm/utils/cpu/mcelog.git
  Mcelog development repository

ftp://ftp.kernel.org/pub/linux/utils/cpu/mce/
  Mcelog releases

git://git.kernel.org/pub/scm/utils/cpu/mce-test.git
  MCE test suite (or now part of LTP to (http://ltp.sourceforge.net)

git://git.kernel.org/pub/scm/utils/cpu/mce-inject.git
  MCE injector

http://halobtes.de
  More documentation about error handling

Backup

Error sources

Machine checks from the CPU

  Corrected (CMCI)

  Uncorrected

NMI

PCI-Express Advanced Error Handling (AER)

ACPI4 (APEI)

Device specific

  SATA errors

  Network

Modern x86 systems with integrated memory controllers report memory (usually) errors as machine checks

page-types on my workstation

```
# page-types  | awk ' { printf "%8s %8s %15s %30s\n",$2,$3,$4,$5 }' |
  sort -rn +1  | head
 1310720     5120                                              # total
  885248     3458 _____
  262144     1024 _____n_____                    nopage
   36400      142 __RU_l_____        referenced,uptodate,lru
   31530      123 ___U_l_____                 uptodate,lru
   28024      109 ___U_lA_____          uptodate,lru,active
   23955       93 ___U_lA_____Ma_b_____
                               uptodate,lru,active,mmap,anonymous, swapbacked
   17105       66 __RU_lA_____ referenced,uptodate,lru,active
    7787       30 __R__l_____             referenced,lru
    5534       21 _____S_____                       slab
```

mcelog triggers

Triggers
  Per DIMM, per Socket, per page, cache error
  Thresholds can be defined for corrected and uncorrected errors

Triggers are simple shell script in /etc/mcelog/
  Executed by the mcelog daemon

Default actions:
  Call "local" script if available
  Page: offlining: implemented in mcelog internally
  DIMM, socket: syslog. Disabled by default
  Cache error: offline affected cores

Local script can do arbitary action

See paper for more details on trigger syntax

Mcelog versus EDAC

EDAC old style driver for chipset memory controllers

- Exposes a lot of low level details
- New model: memory controller in CPU
- Memory errors integrated with machine checks
- Handled by standard MCE subsystem

EDAC needs driver for each platform

- And often accesses "non stable" registers that could change even with steppings
- Mcelog uses standardized interfaces as far as possible.

No integration with software

- Requires special configuration for each board to identify components

Cannot do a lot of things that user space (mcelog) can do

Testing

Testing machine checks is difficult

- Normal operation doesn't have enough errors
- So standard Linux community testing model doesn't work very
- Needs special injection support and test suites

Various injectors on software level (without hardware support)

- Low level machine check injector
- Page error injector for process and for arbitrary page

New injectors, test suite mce-test for testing MCEs

- Testing low level handler with mce-inject
- Testing hwpoison VM code in process context
  - Bring pages into specific states and test to see if they can be poisoned

Ongoing work to get the best test coverage

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2010, Intel Corporation. All rights are protected.

Open Source Technology Center

Software and Solutions Group