

Linux RAS: Fight with hardware error

Huang Ying
2010/10

Agenda

- Introduction
- Hardware Error Logging
 - Machine Check
 - APEI
- Hardware Error Recovering
 - Hardware Poison
 - Predictive Fault Analysis

Introduction

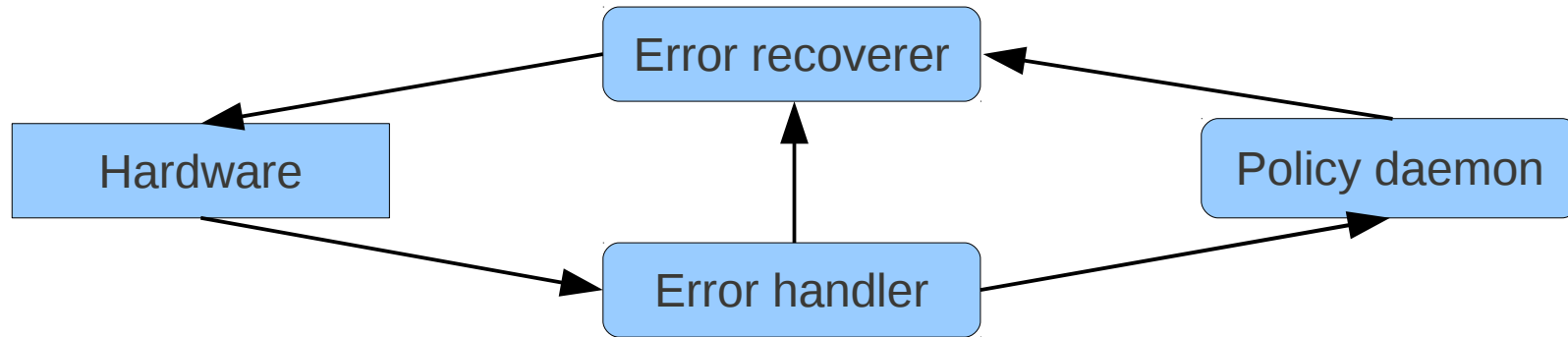
What is RAS

- Definition from Wikipedia
 - **Reliability:** Features that help avoid and detect hardware faults.
 - **Availability:** Amount of time a device is actually operating as the percentage of total time it should be operating.
 - **Serviceability:** Takes the form of various methods of easily diagnosing the system when problems arise.
- We will focus on hardware error detecting, logging and recovering

Why RAS

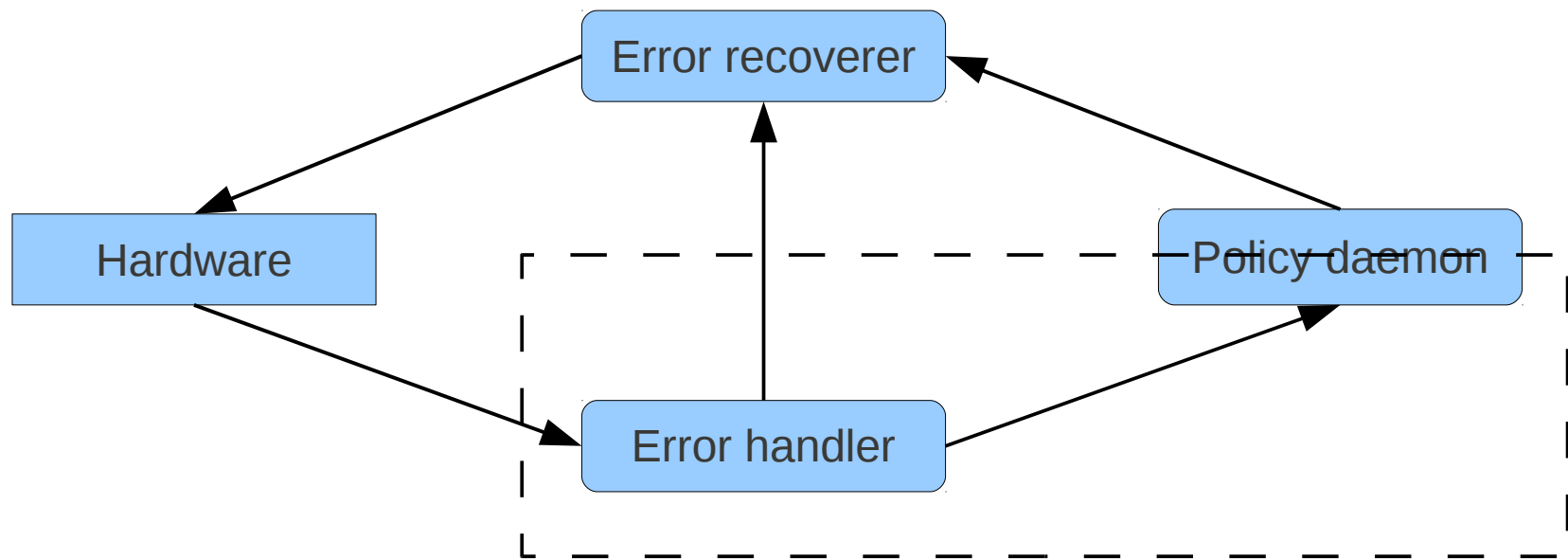
- Essential for Linux to be used in mission critical environment (bank, stock trading etc)
- Hardware error becomes more common with bigger system (clusters, cloud computing) and more transistor.
- Error reporting is even good for desktop (find fault hardware).

A Big Picture of RAS Processing



- Hardware detect the error, report it to Linux
- Linux hardware error handler collect error information from hardware, may trigger error recovery for some urgent recoverable errors
- Linux hardware error recoverer isolate/swap broken hardware
- Linux recovery policy daemon accounts errors for hardware components, trigger recovery operation accordingly.

Hardware error logging



Overview

- CPU, QPI and memory error: Machine Check
- Reporting error via firmware: APEI (ACPI Platform Error Interface)
- PCIe error: PCIe AER (Advanced Error Reporting)
- NMI: PCI SERR, IOCHK, etc

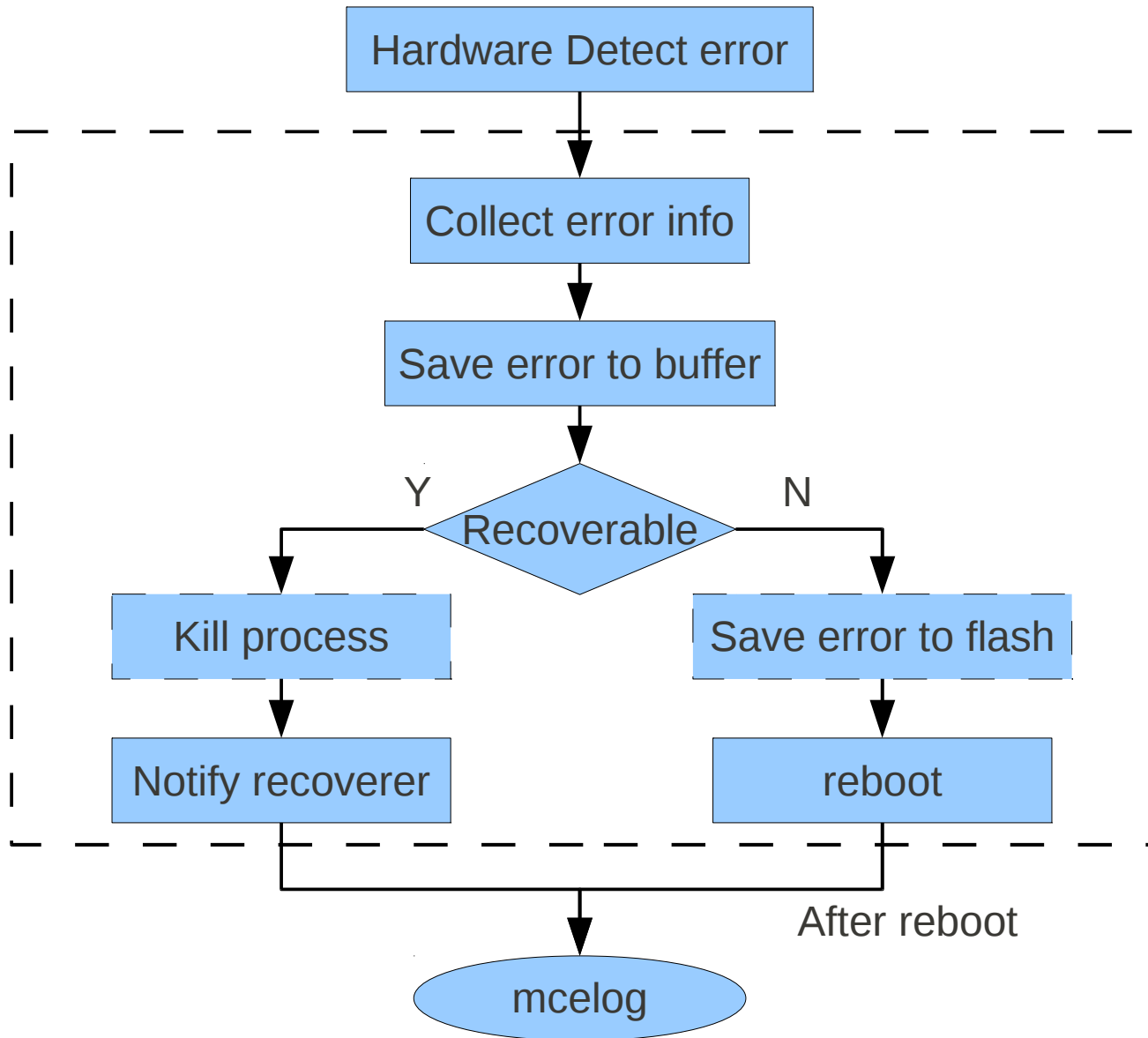
Hardware Error Severity

- Corrected
 - Corrected by hardware, 1 bit error for ECC memory
- Uncorrected
 - Recoverable
 - Recoverable by software, isolate the fault memory page
 - Fatal
 - Have to halt or reboot system

Machine Check: Hardware mechanism

- Report CPU, QPI and memory errors (corrected and uncorrected)
- Uncorrected errors are notified via exception 18
- Corrected errors are notified via interrupt or poll
- Deliver error information with a set of MSR (model specific register): severity, error code, fault memory address, etc

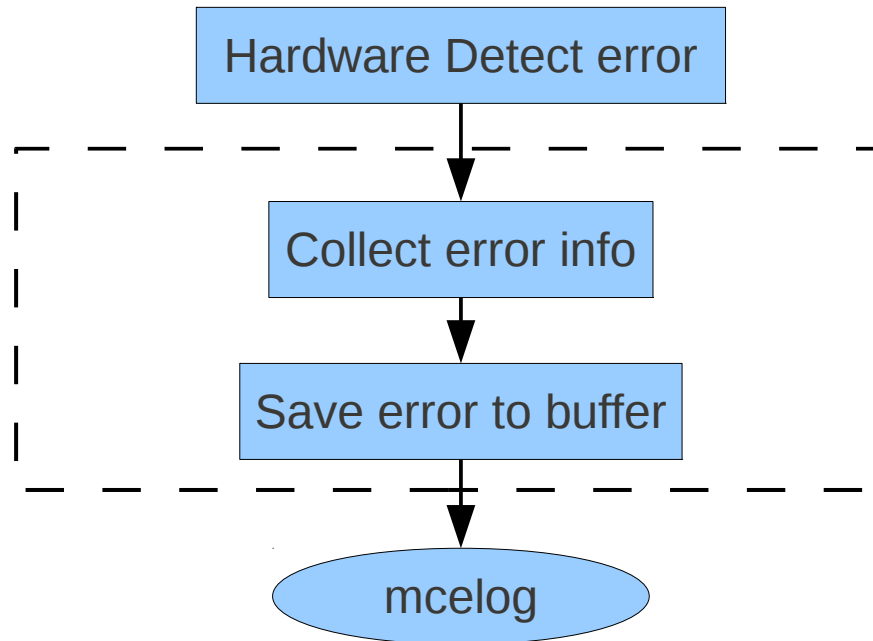
Machine Check: Exception handler 1



Machine Check: Exception handler 2

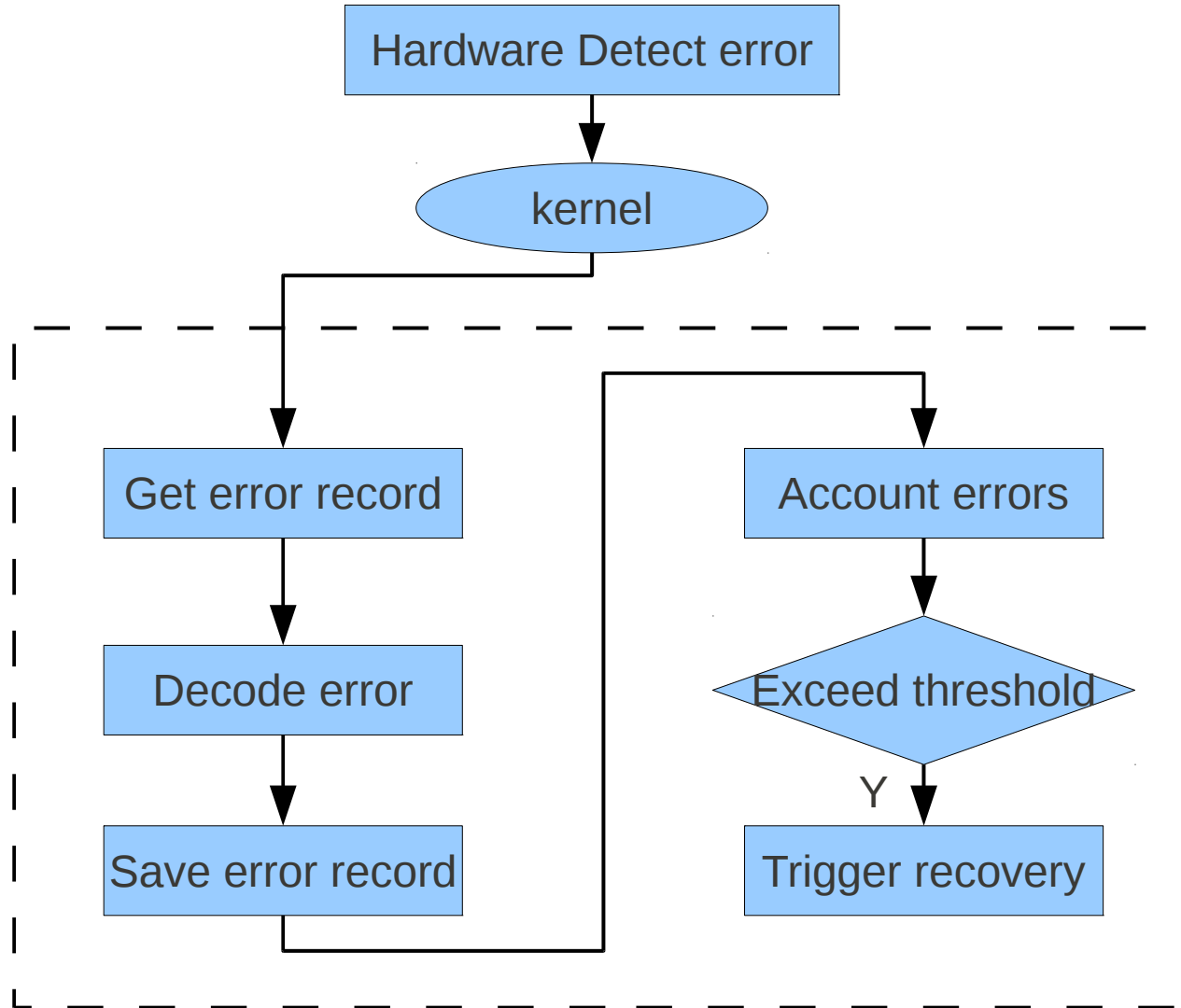
- Collect error information from MSR's
- Put error record into internal lock-less buffer
- If severity is recoverable
 - Kill current process if necessary
 - Notify recoverer (such as fault memory page offliner)
- Otherwise severity is fatal
 - Save error record into flash if possible
 - Panic and reboot to log (read from flash or sticky MSR's)

Machine Check: IRQ handler or poller



- Collect error information from MSRs
- Put error record into internal lock-less buffer

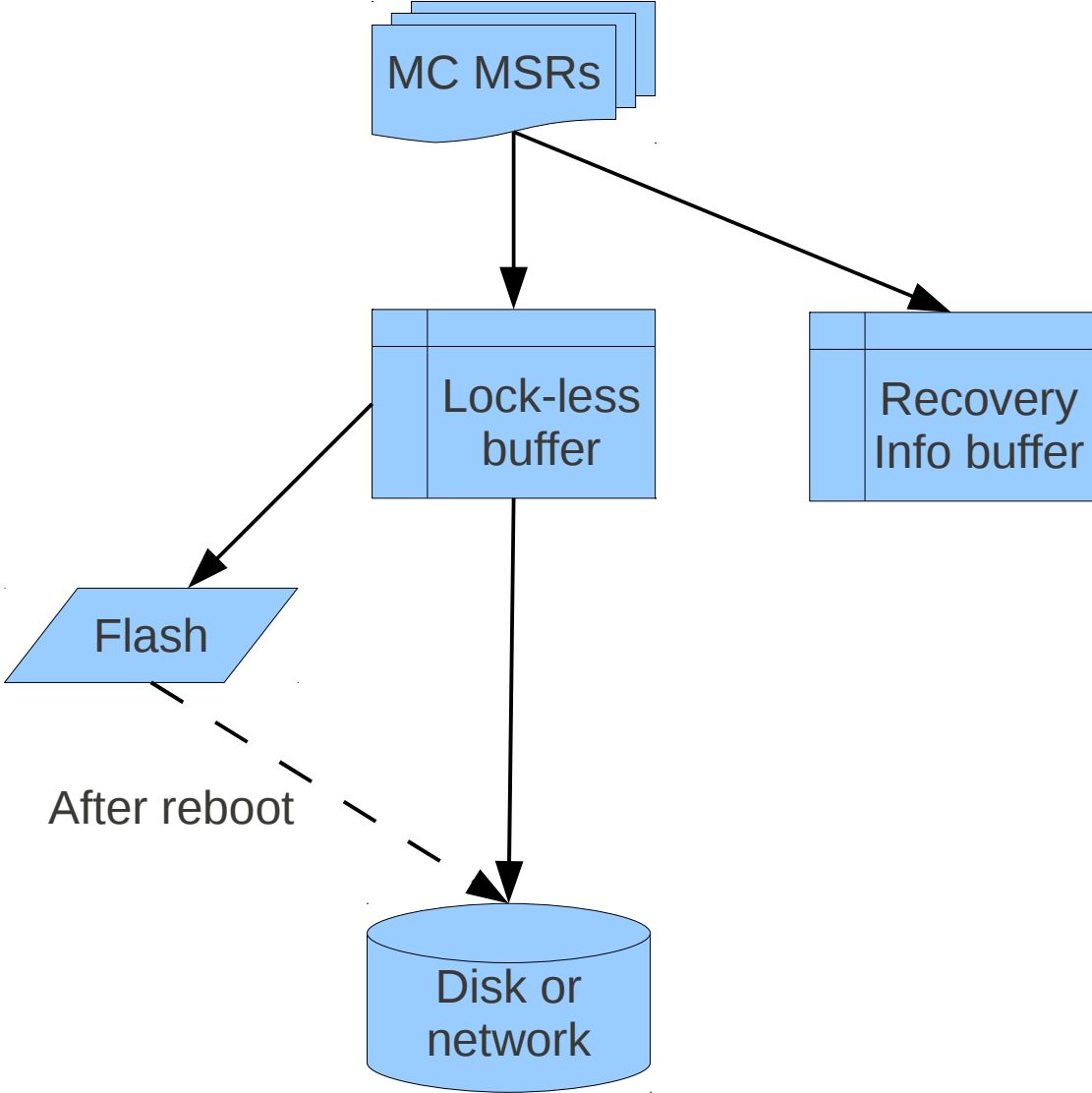
Machine Check: mcelog 1



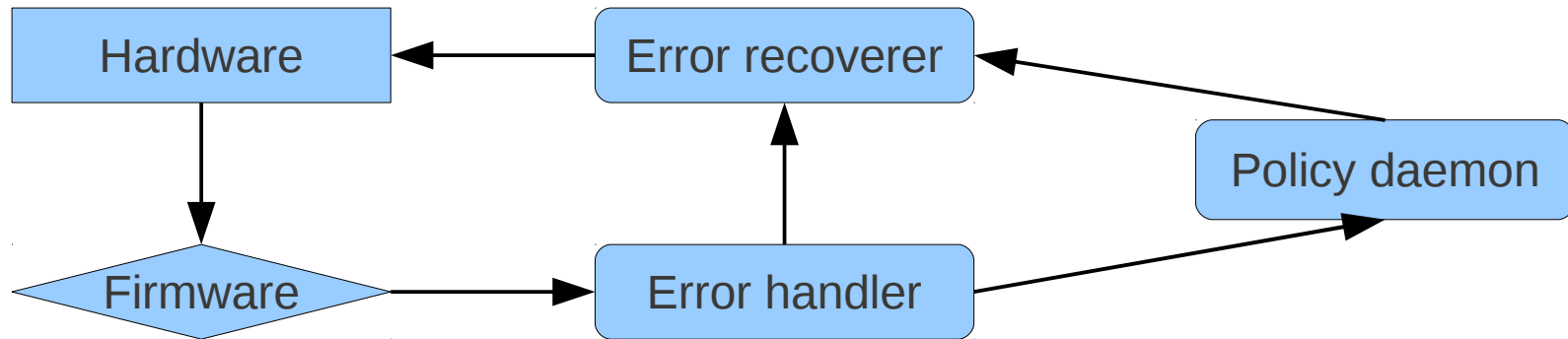
Machine Check: mcelog 2

- Get error record from kernel lock-less buffer or flash
- Decode error record
- Save error record to disk or network
- Account errors for hardware components
- If exceed threshold
 - Trigger error recovery operation

Machine Check: Data flow chart



APEI: What is APEI

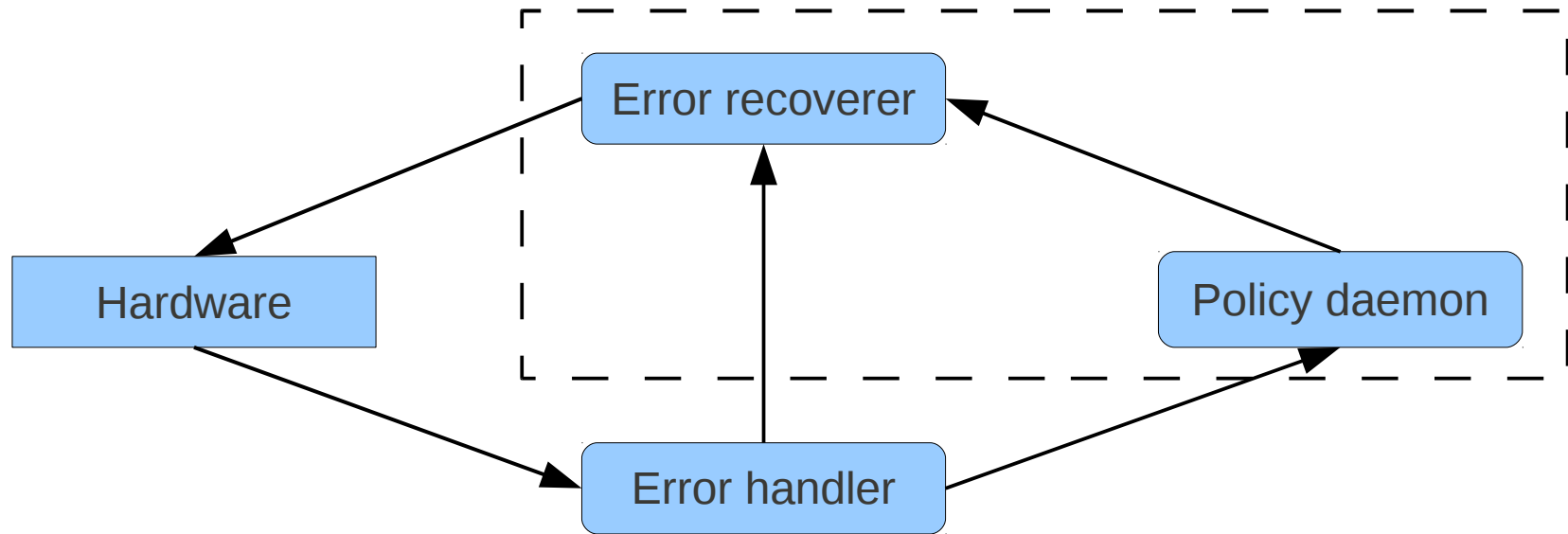


- APEI: ACPI Platform Error Interface
 - Described by several ACPI tables
- Error logging with APEI
 - Hardware detect error, report to firmware instead of Linux
 - Firmware collect error information, then report to Linux

APEI: Why?

- Can report non-standardized (chipset, etc) error
 - Chipset errors are hard to be standardized like machine check, chipset drivers are tedious
- More complete error information
 - Can access non-standard registers for standardized error
 - Firmware is more flexible than hardware

Hardware error recovering



Overview

- Halt, reboot: prevent corrupted data goes disk
- Hardware poison: isolate the poisoned error memory page
- Memory hotplug/hotremove: offline/swap the error memory
- CPU hotplug/hotremove: offline/swap the error CPU
- PCI hotplug: offline/replace the error PCI card
- Predictive Failure Analysis: predict error trend, and replace potential bad components

Hardware Poison: Background

- Example: Patrol scrub
 - Memory controller detect corrupted page in the background, such as 2 bit ECC error
 - Mark corrupted page as poisoned in hardware and notify system with MCE.
 - Further accessing to the page will trigger MCE. Need kill the process

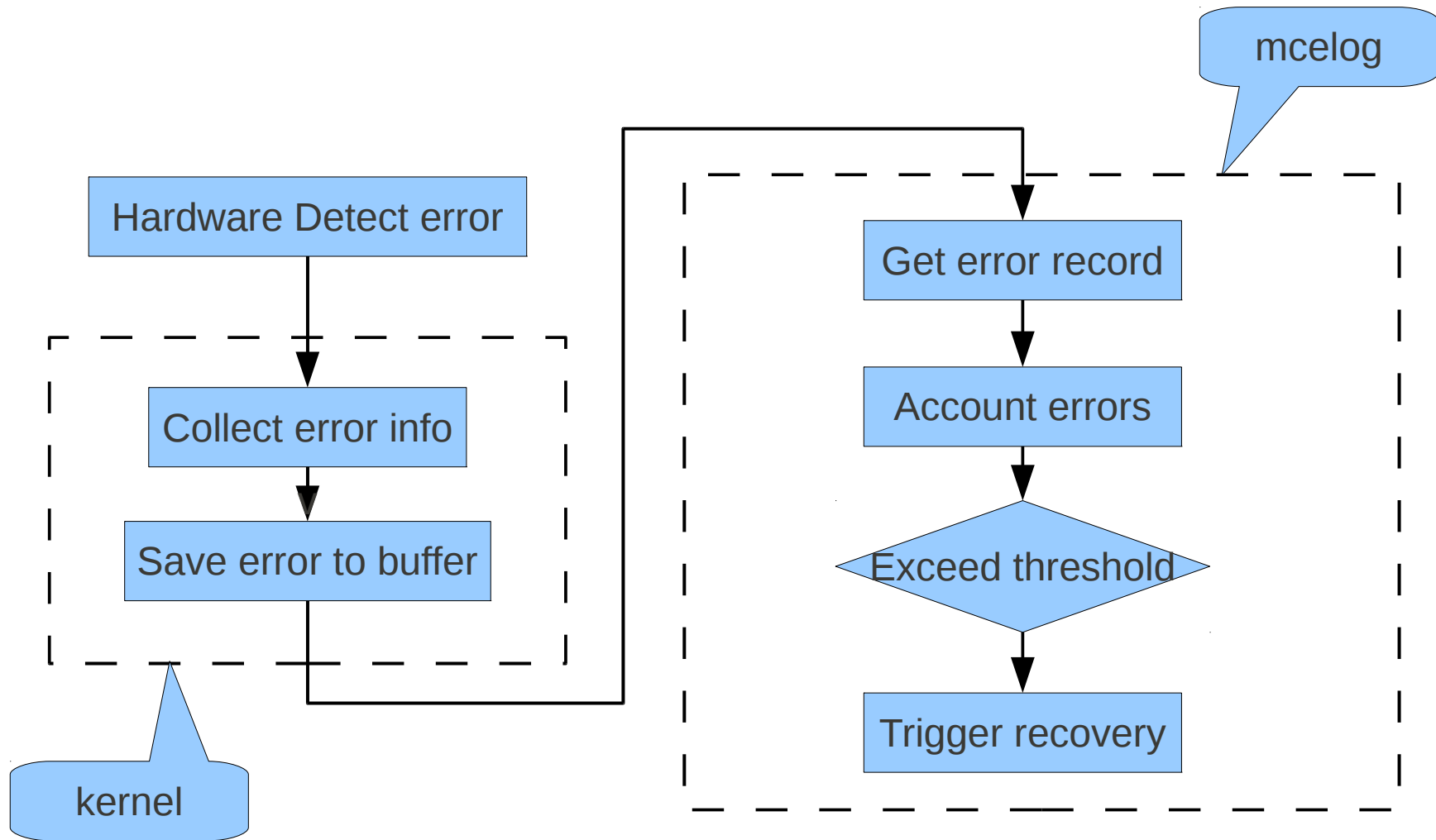
Hardware Poison: Recovering

- Mark page as bad
- Unmap page from the processes using the page, mark page table entry as poisoned, process will be killed when access the poisoned address later
- Send SIGBUS to the processes optionally
- Process based on page type
 - Clean page/swap cache: drop the cache
 - Free page: refuse to be allocated because marked bad
 - Anonymous page: kill the process if accessed
 - Dirty page cache: mark IO error, try drop the cache
 - Other pages: ignore, hope we are lucky

Predictive Fault Analysis

- Observation
 - Hard uncorrected errors are often preceded by corrected errors
- Idea
 - Stop using component in advance

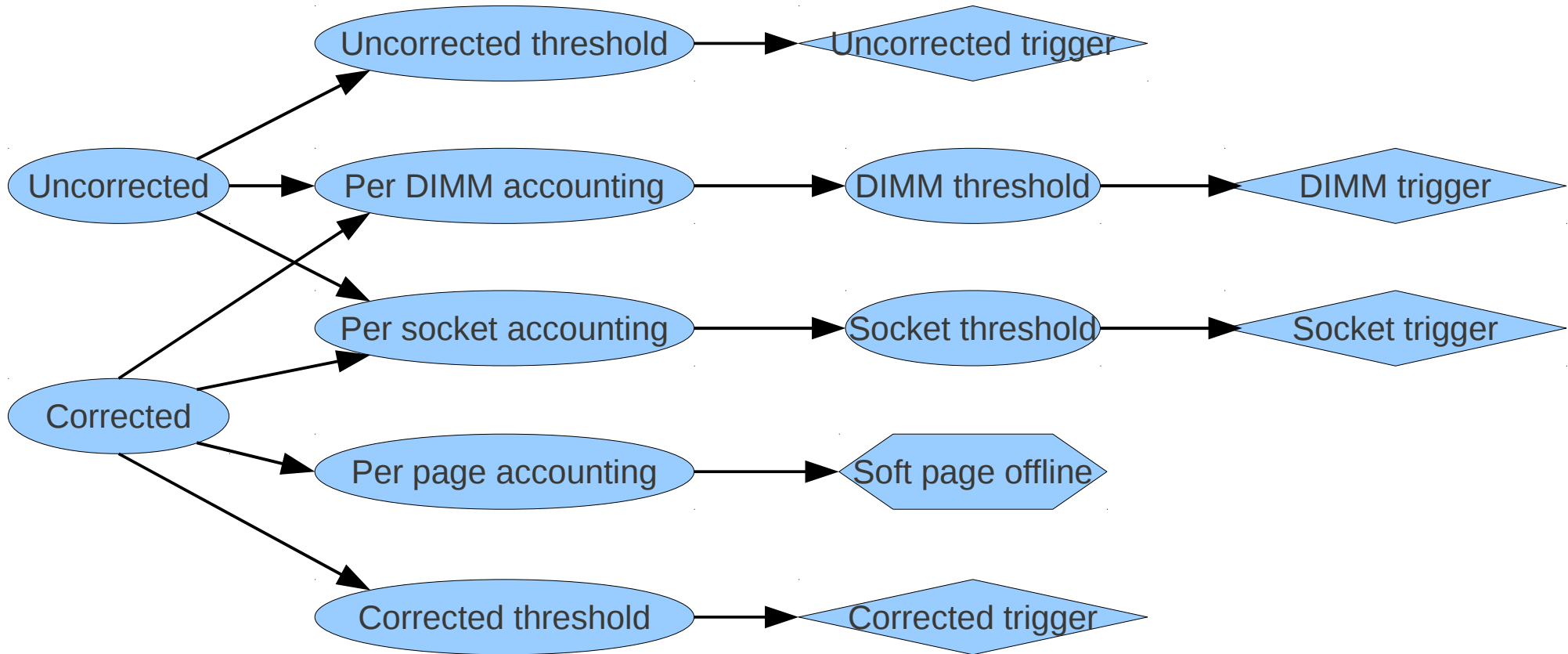
PFA: bad page offlining 1



PFA: bad page offlining 2

- Memory controller detect corrected error, report it to Linux via MCE or APEI
- Kernel error handler collect error information, save error record in lock-less buffer
- Mcelog get error record from kernel lock-less buffer
- Mcelog account errors for each page
- If exceed threshold, trigger page offlining
 - similar with hardware poison recovering, can write back dirty page, because we can touch them here

PFA: mcelog error accounting



References

- Intel software developers' manual: Vol 3A/B system programming guide
 - Description of Machine Check
- ACPI specification
 - Description of APEI
- [git://git.kernel.org/pub/scm/utils/cpu/mcelog.git](https://git.kernel.org/pub/scm/utils/cpu/mcelog.git)
 - Mcelog development repository
- [git://git.kernel.org/pub/scm/utils/cpu/mce-test.git](https://git.kernel.org/pub/scm/utils/cpu/mce-test.git)
 - MCE test suite
- <http://halobtes.de>
 - Andi Kleen's document about Machine Check and mcelog

Questions?

Backup

Challenges of RAS

- Ignore kernel locking
 - Lock-less data structure
- Needs to be handled quickly and with minimal kernel infrastructure
- Hard to test
 - Special hardware
 - Software based error injector