presented by

Server RAS and UEFI CPER

Spring 2017 UEFI Seminar and Plugfest
March 27 - 31, 2017
Presented by
Lucia, Mao (Intel)
Spike Yuan (Intel)

Agenda

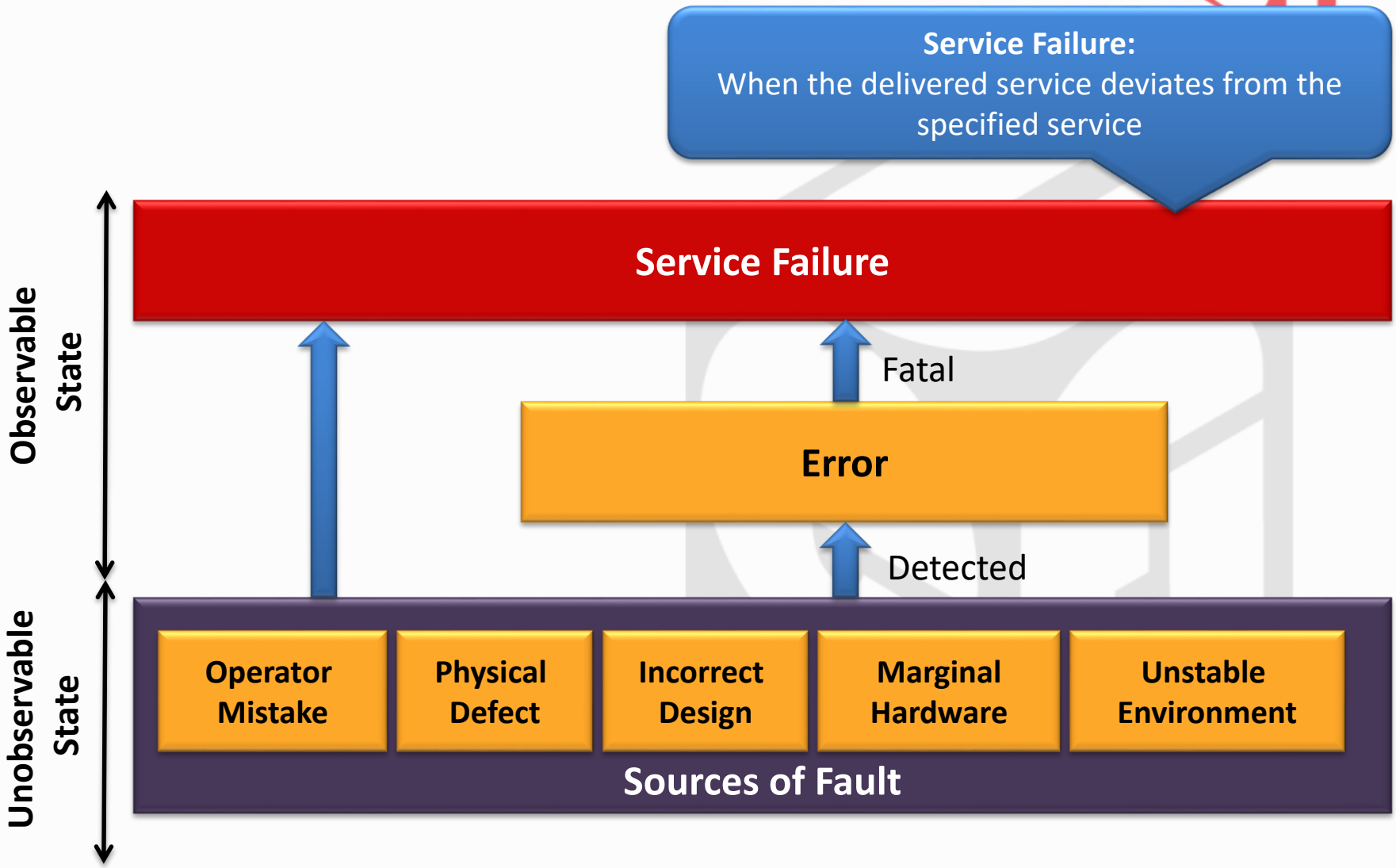
- RAS Basic
- Server RAS Challenges
- SW Building Blocks
- Possible Solutions
- Call to Action

RAS Basic

RAS101: Key Definitions

- Reliability:
  - System capability to detect errors, correct errors, and flag errors.
  - Measured in FITs (Failure in Time).
  - 1FIT = 1 Failure / 1 Billion hours (MTBF – Mean Time Between Failures = 114K Years!)
- Availability
  - System capability to stay operational even when error occur.
  - Measured in terms of ‘down time within a time interval’
  - Five 9’s (99.999% Up Time) => 22 seconds Down Time in One Month
- Serviceability
  - System capability to report failures for “FRU Isolation” and ease of repair. FRU – Field Replaceable Unit, e.g., DIMM, PCI Express* device

Fault, Error, and Failure

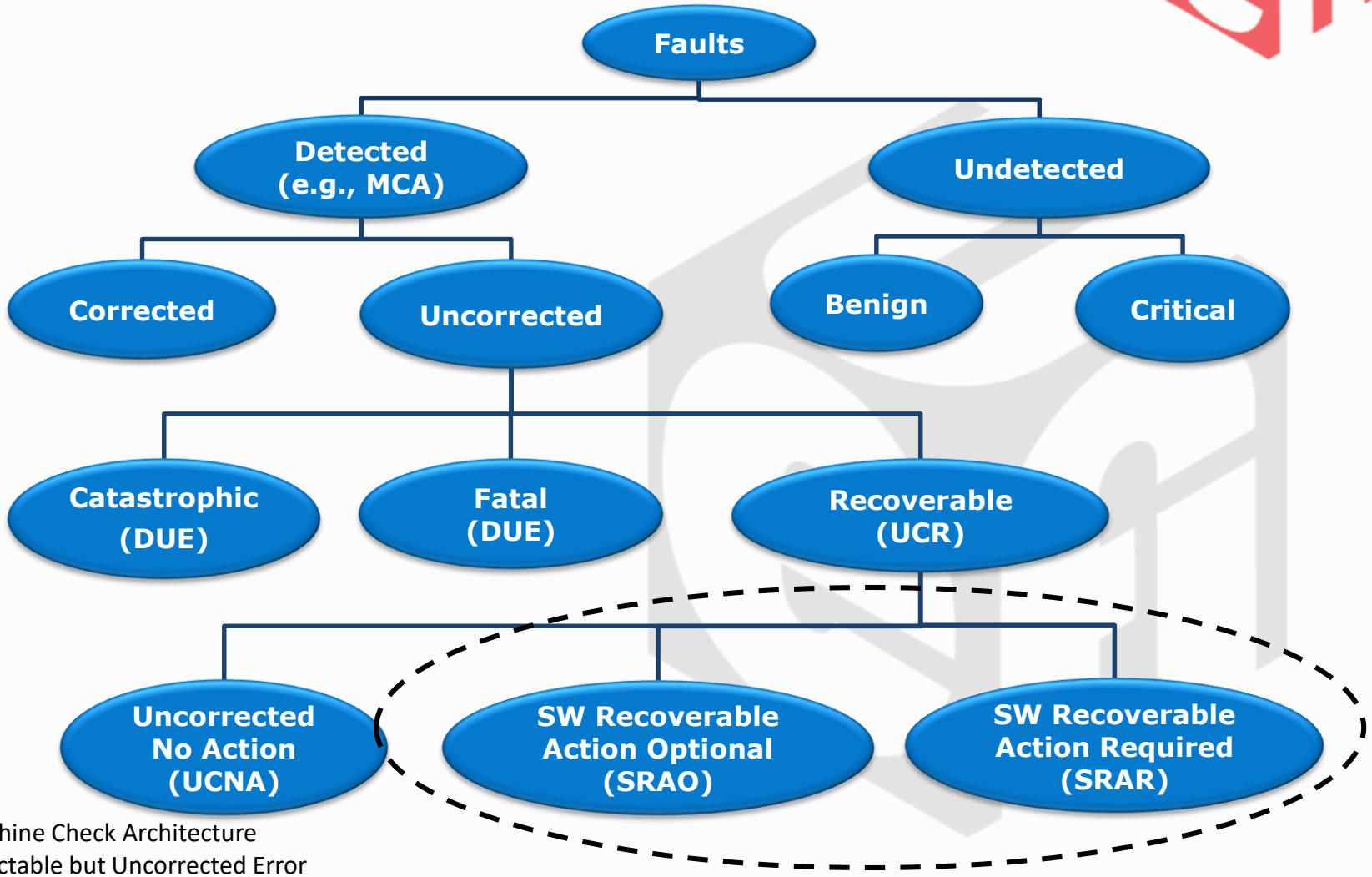


Sources of Fault/Error

SN	Error Source	Definition	Example
1	Transient Error	Electrical Noise induced faults mainly affecting links such as DDR Bus, or PCI Express links.	Transient errors on links may alter the data, Command or Address bits during read/write operation. Reads won't alter the DRAM stored value, but 'Writes' may alter.
2	Soft Error	Errors due to external high energy particle strike, e.g., Alpha particles, Neutrons. Soft errors could occur is any known good system	Result in affecting storage structures such as DRAM cell (SBE or MBE), L1/L2/L3 caches.
3	Hard (Device) Failure	Device failure due to marginality of the device or degradation over time.	Failure of entire device such as DRAM, memory buffer chip, or CPU chip

- SBE: Single-bit Error. MBE: Multi-bit Error

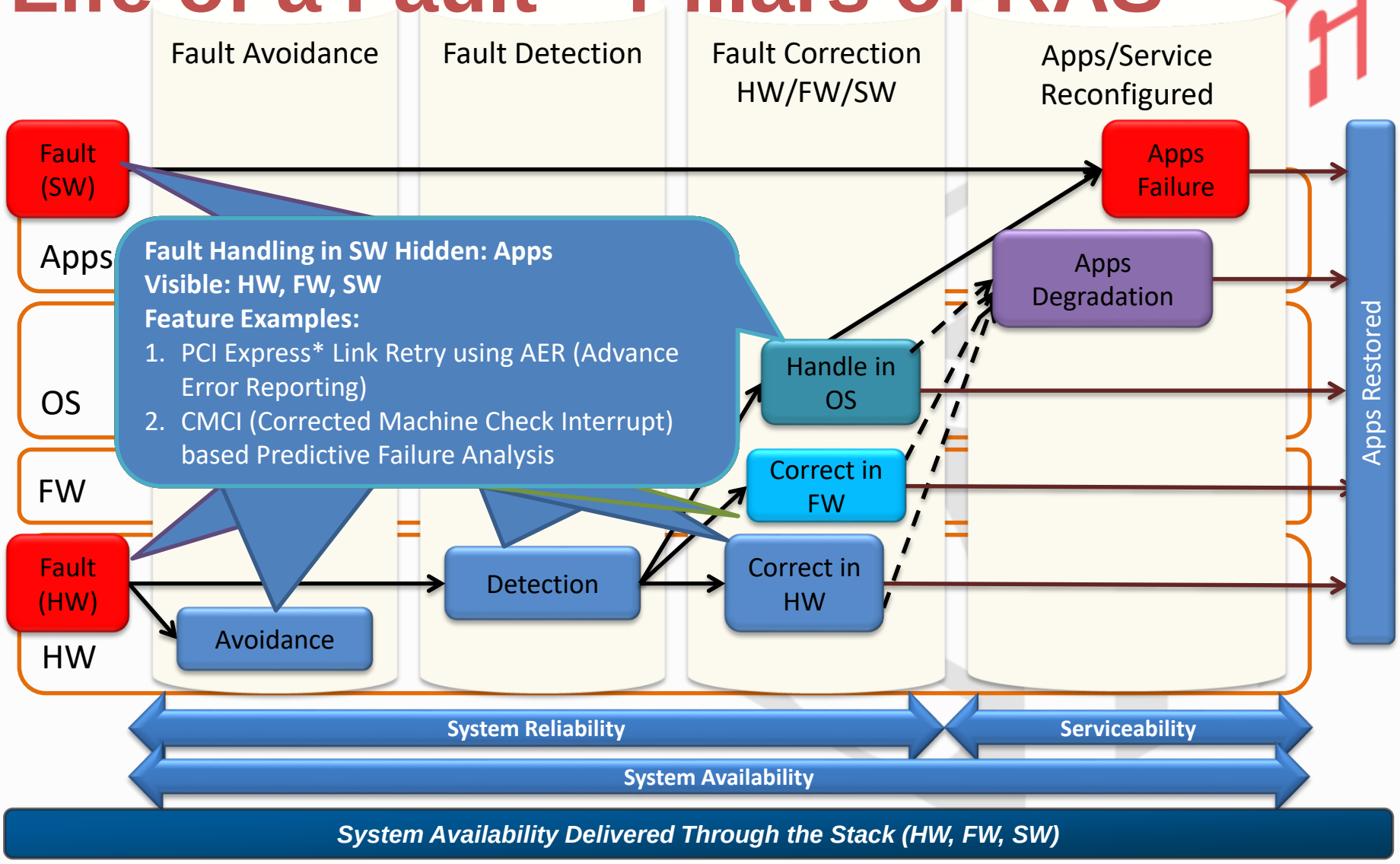
An Example - Intel® Xeon® Processor Fault Classification



MCA: Machine Check Architecture
DUE: Detectable but Uncorrected Error
UCR: Uncorrected Recoverable

Server RAS Challenges

Life of a Fault – Pillars of RAS



RAS Enabling Framework

Fault Handling = RAS (Four Pillars)

Fault Tolerance
1. Avoidance
2. Detection
3. Correction

Fault Management
4. Reconfiguration

System Stack

Fault Correction at App Layer

Fault Correction Through OS

Fault Correction Through FW

Application

OS/VMM

FW (OEM/IBV/BMC)

OS-based Fault Management

FW-based Fault Management

Error Signaling/Polling

Fault Avoidance, Detection, and Correction in HW

FW (Silicon RefCode)

Silicon

Error Logging

E.g., Memory ECC

E.g., Failed DIMM Isolation

System Reliability
Extending the Uptime

Diagnosability/Serviceability/Manageability
Minimizing Downtime

RAS Enabling Requires HW, FW/BIOS, OS/SW

RAS Needs of Cloud and HPC Clusters

Cloud Infrastructure	HPC Infrastructure
Need Fault Handling	Need Fault Handling
Check pointing is not used	Check-pointing is actively used
Applications can tolerate single machine failure	Applications can not tolerant single machine failure
Fault Management Capabilities for improving TCO	**Fault Tolerant** Capabilities for extending uptime
Example: Automated techniques for identify failed component	Example: HW/FW based self-healing techniques

RAS Needs of Mission Critical Segment

Prevent/minimize unplanned downtime

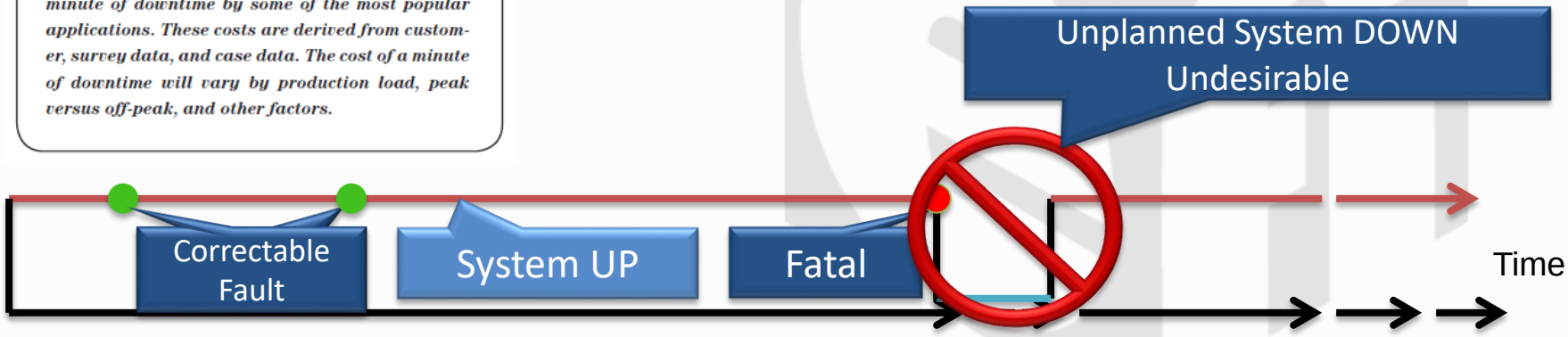
COST OF DOWNTIME

Application Name	Cost/Minute
Trading (securities)	$73,000
HLR	$29,300
ERP	$14,800
Order Processing	$13,300
E-Commerce	$12,600
Supply Chain	$11,500
EFT	$6,200
POS	$4,700
ATM	$3,600
E-Mail	$1,900

The above table shows the average cost of a minute of downtime by some of the most popular applications. These costs are derived from customer, survey data, and case data. The cost of a minute of downtime will vary by production load, peak versus off-peak, and other factors.

What would an outage cost?

Source: Trend in IT Value Report, Standish Group International, 2008



Intel® Xeon® RAS features directly impact end-user's bottom line!

FW/SW Building Blocks

Error Reporting Basics

- Error Reporting includes two functions:
  - Logging
  - Signaling

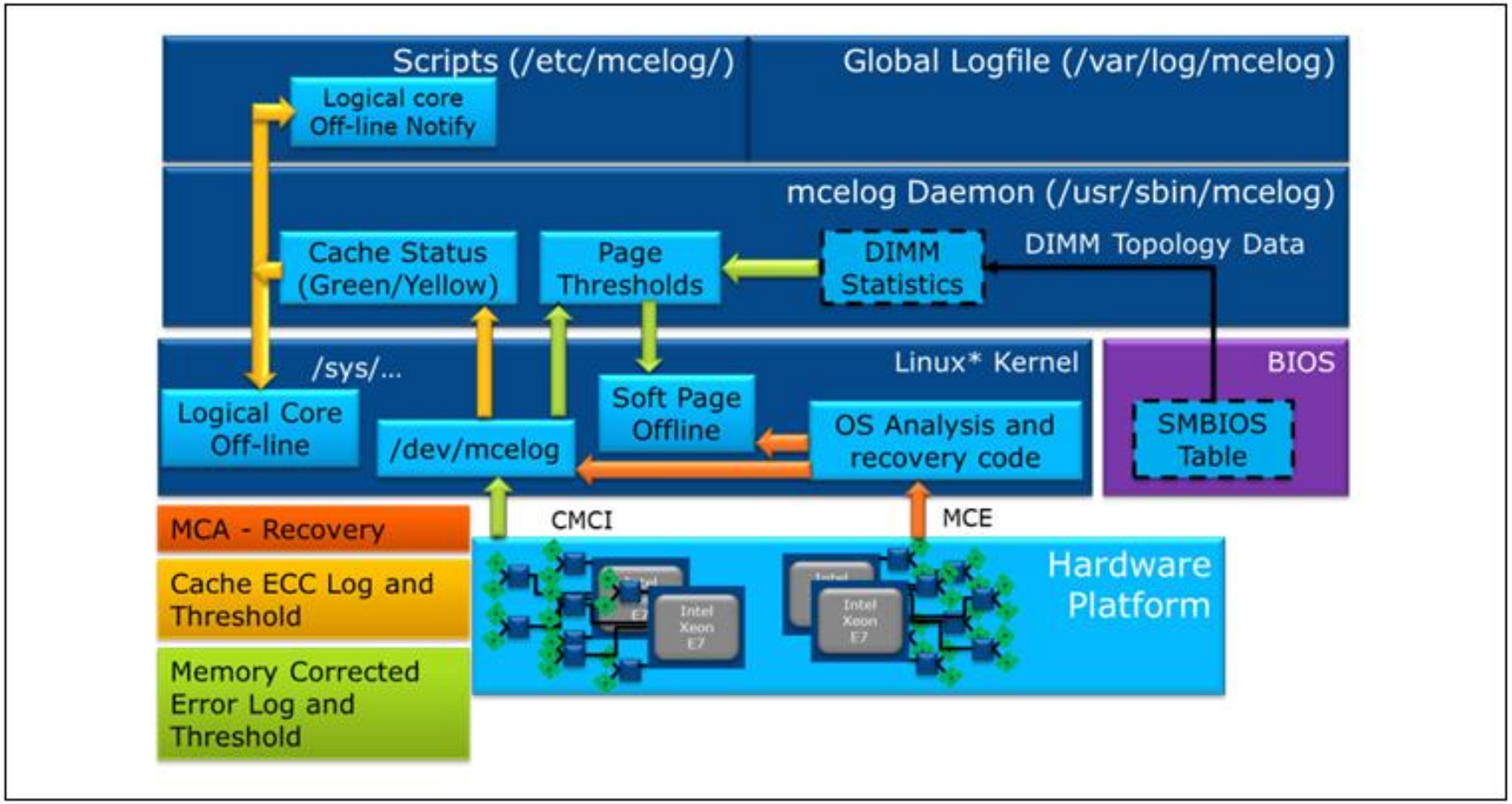
- Logging
  - Through MCA Banks, PCIe AER Registers, and Memory Corrected Error Registers
- eMCA2 Mode – Enhanced Error reporting to support Firmware-First mode;
- Signaling of Corrected Errors
  - CMCI (Corrected Machine Check Interrupt)
    - Threshold based
    - Enabled only in IA32-legacy MCA mode. Disabled in eMCA2 mode.
  - CSMI (Corrected SMI) for core/uncore (Part of the eMCA2 new Feature)
    - Enabled only in eMCA2 mode. Disabled in IA32-legacy MCA mode.
    - No Threshold
  - SMI (System Management Interrupt) for Memory errors
  - MSI (Message Signaled Interrupt) or external signaling for PCI Express* errors

Error Reporting Basics (Continued)

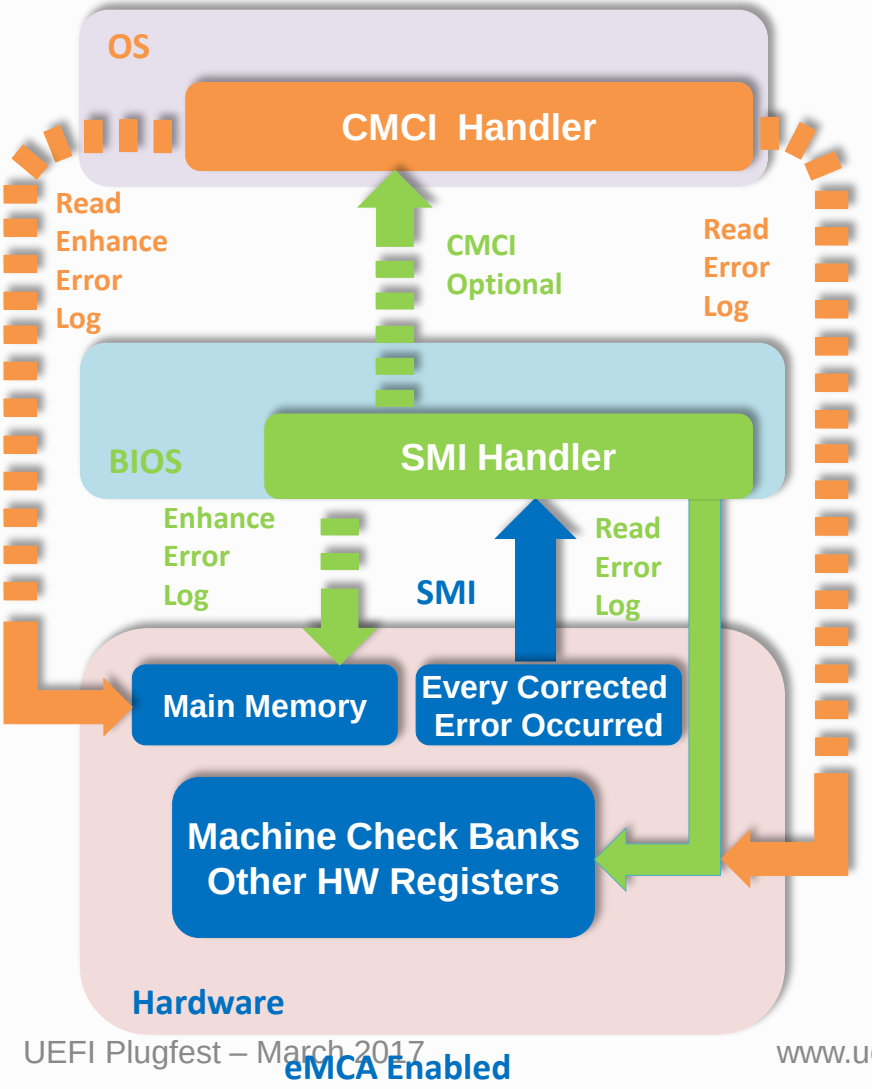
- Signaling of Uncorrected Recoverable Errors (e.g., UCNA)
  - CMCI for core/uncore errors at the source
    - Enabled only in IA32-legacy MCA mode. Disabled in eMCA2 mode.
  - MSMI (Machine Check SMI) for core/uncore errors at the source (Part of the eMCA2)
    - Enabled only in eMCA2 mode. Disabled in IA32-legacy MCA mode.
    - MSMI trigger (same as SMI).
  - MSI or external signaling for Severity1 PCIe AER nonfatal errors
- Signaling of Uncorrected Recoverable Errors (e.g., SRAO and SRAR)
  - MCERR (Machine Check Error) for core/uncore errors
    - External signaling – via CATERR_N pin (16 BCLK Pulse). Allows propagation to other sockets.
    - In-band signaling – MCE trigger (vector 18h). In-band SMI trigger if configured.
    - Enabled only in IA32-legacy MCA mode. Disabled in eMCA2 mode
  - MSMI (Machine Check SMI) for core/uncore errors at the source(Part of the eMCA2)
    - Enabled only in eMCA2 mode. Disabled in IA32-legacy MCA mode.
    - External signaling – via MSMI_N pin. Allows propagation to other sockets.
    - In-band signaling – MSMI trigger (same as SMI).

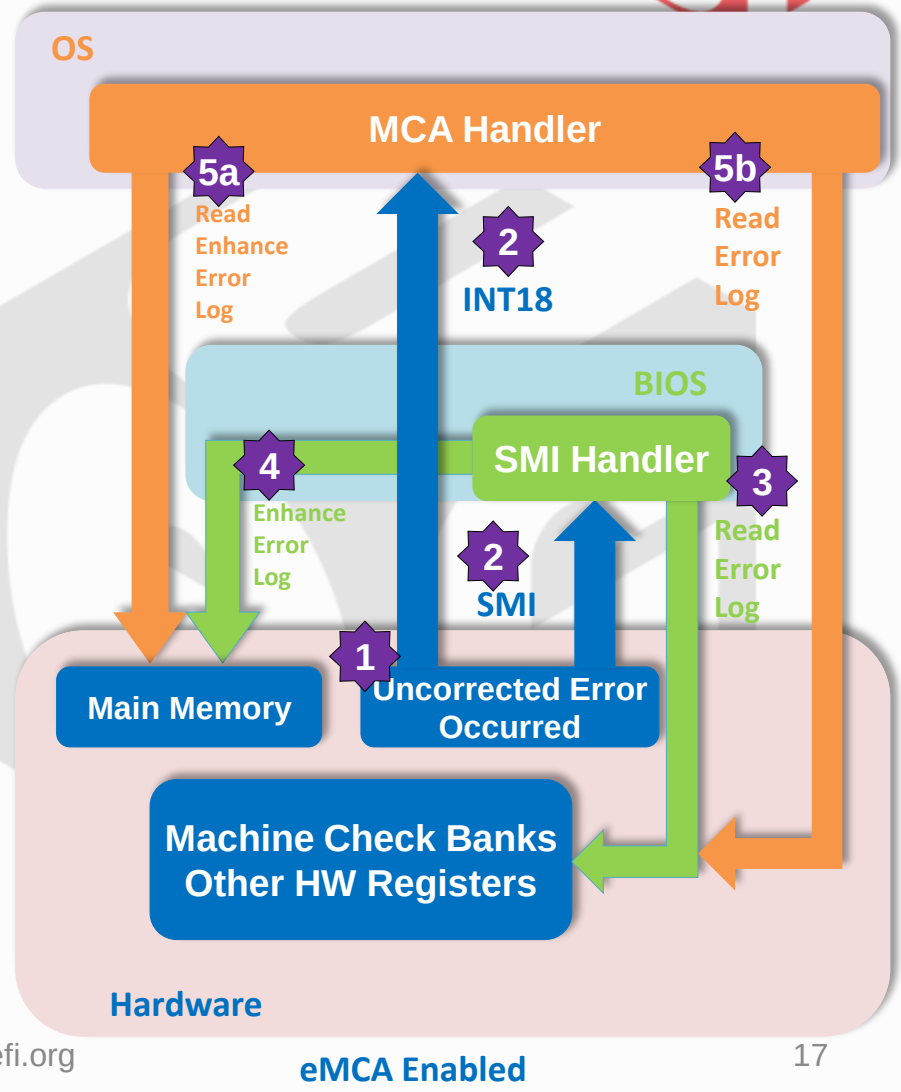
UCNA: Uncorrected No Action
SRAO: SW Recoverable Action Optional
SRAR: SW Recoverable Action Required

Linux RAS Blocks



Linux CMCI/MCA Handler with eMCA2





APEI Overview

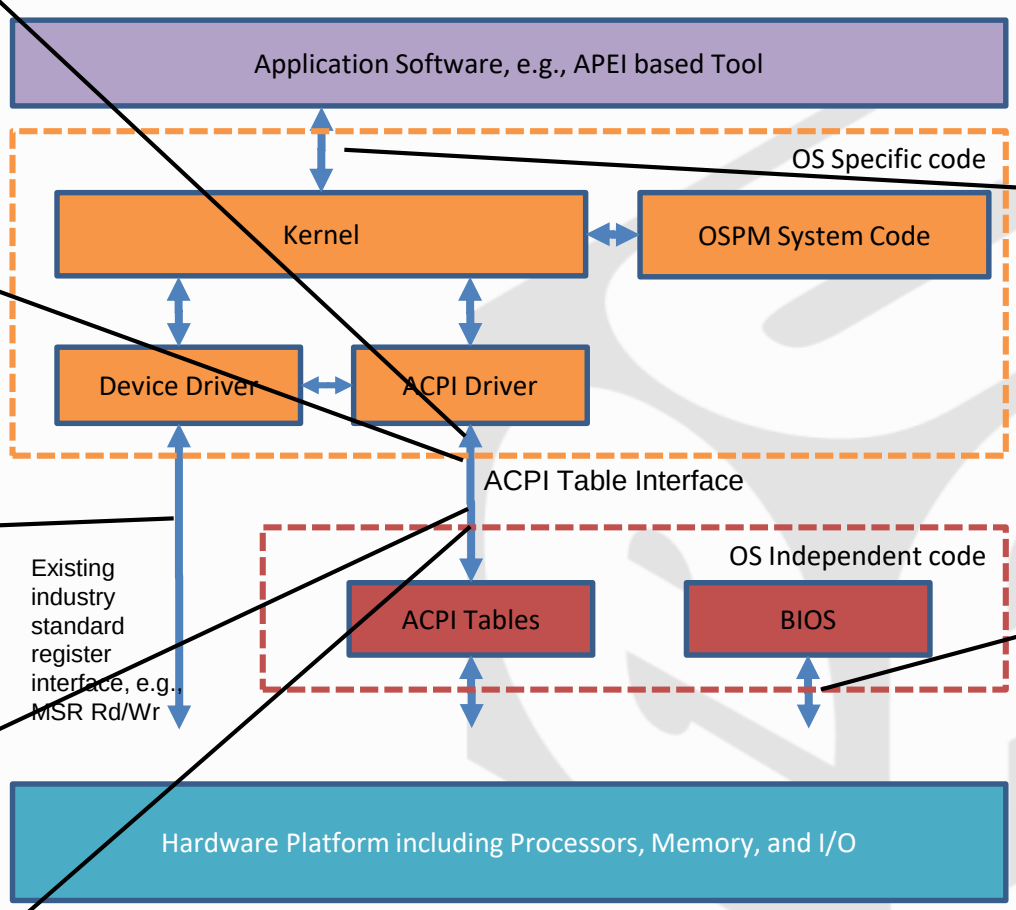
Boot-time:
Step 1a: BIOS/SMM presents APEI tables towards OS. BIOS checks if target processor supports APEI feature prior to presenting to OS

Run-time:
Step 2b: OS requests BIOS/SMM to enable APEI, i.e., enable machine-check bank non-zero value write capability

Step 2c: OS writes Machine-check banks values requested in

Step 2d: OS requests BIOS/SMM to either inject MCERR or CMCI. Note that actual trigger event occurs within OS context.

Step 2f: Once testing is completed, OS requests disabling MCERR/CMCI signaling.



Run-time:
Step 2a: APEI based Error Injection Tool requests OS to inject EINJ based error in selected Machine-check bank

Step 2e: BIOS discovers the banks updated by OS and program SMI triggering source registers.

UEFI CPER Overview

- Common Platform Error Record
- CPER is also the format used to describe platform hardware error by various APEI tables, such as ERST, BERT and HEST etc.

Linux CPER implementation

- Legacy MCA way:

In arch/x86/kernel/cpu/mcheck/mce-apei.c -

an error report given to the kernel by APEI and pass it into the normal Linux error logging code that invoke after finding an error in a machine check bank. This code didn't provide address information in the machine check bank.

- eMCA2 Mode:

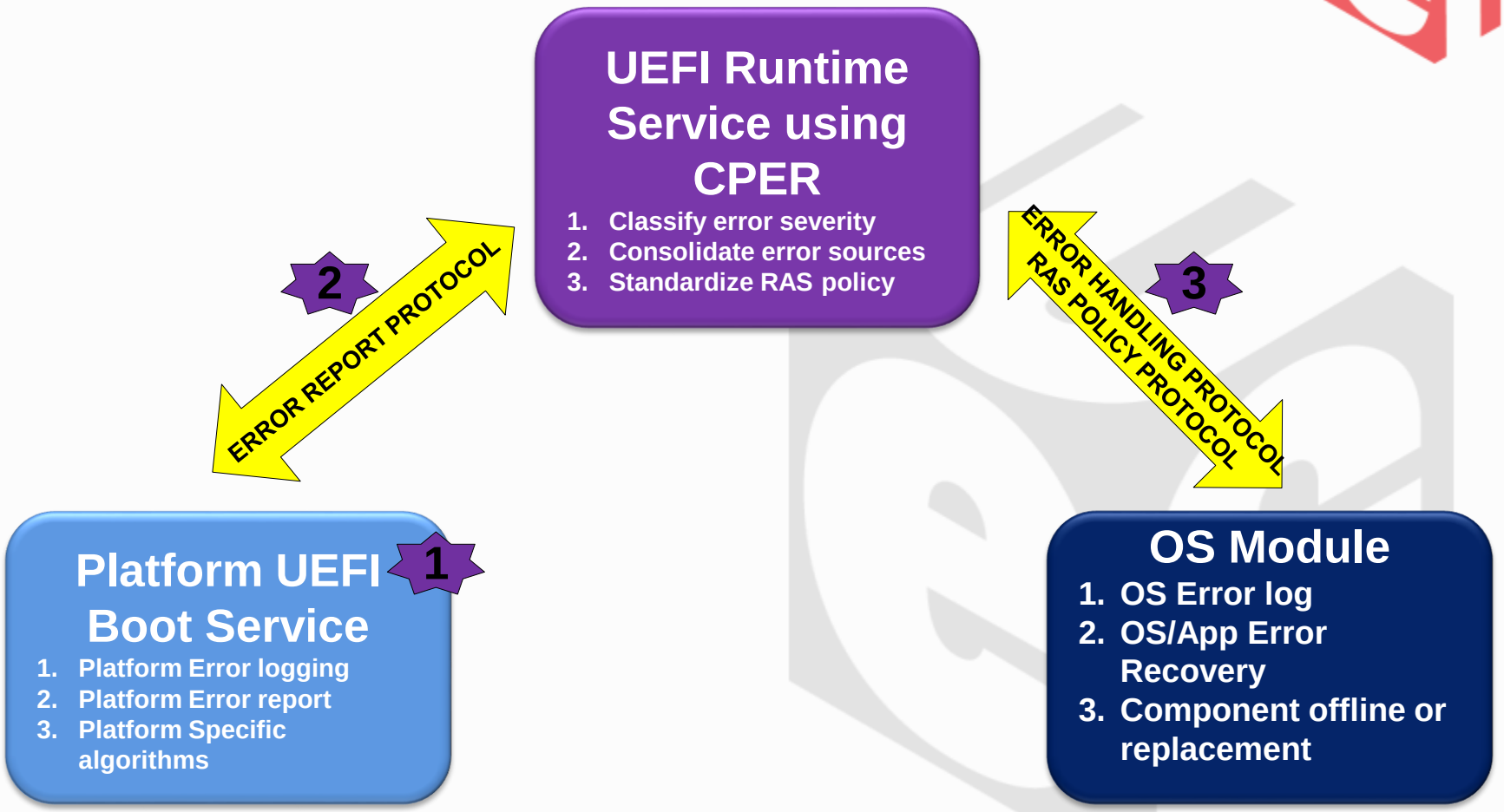
In drivers/acpi/acpi_extlog.c –

on recent generation servers with eMCA, this code picks up additional information provided by BIOS associated with each error logged. All Linux really looks for in the CPER record is the "handle" into the SMBIOS/DMI table so that, for example, it can report which DIMM is associated with the error.

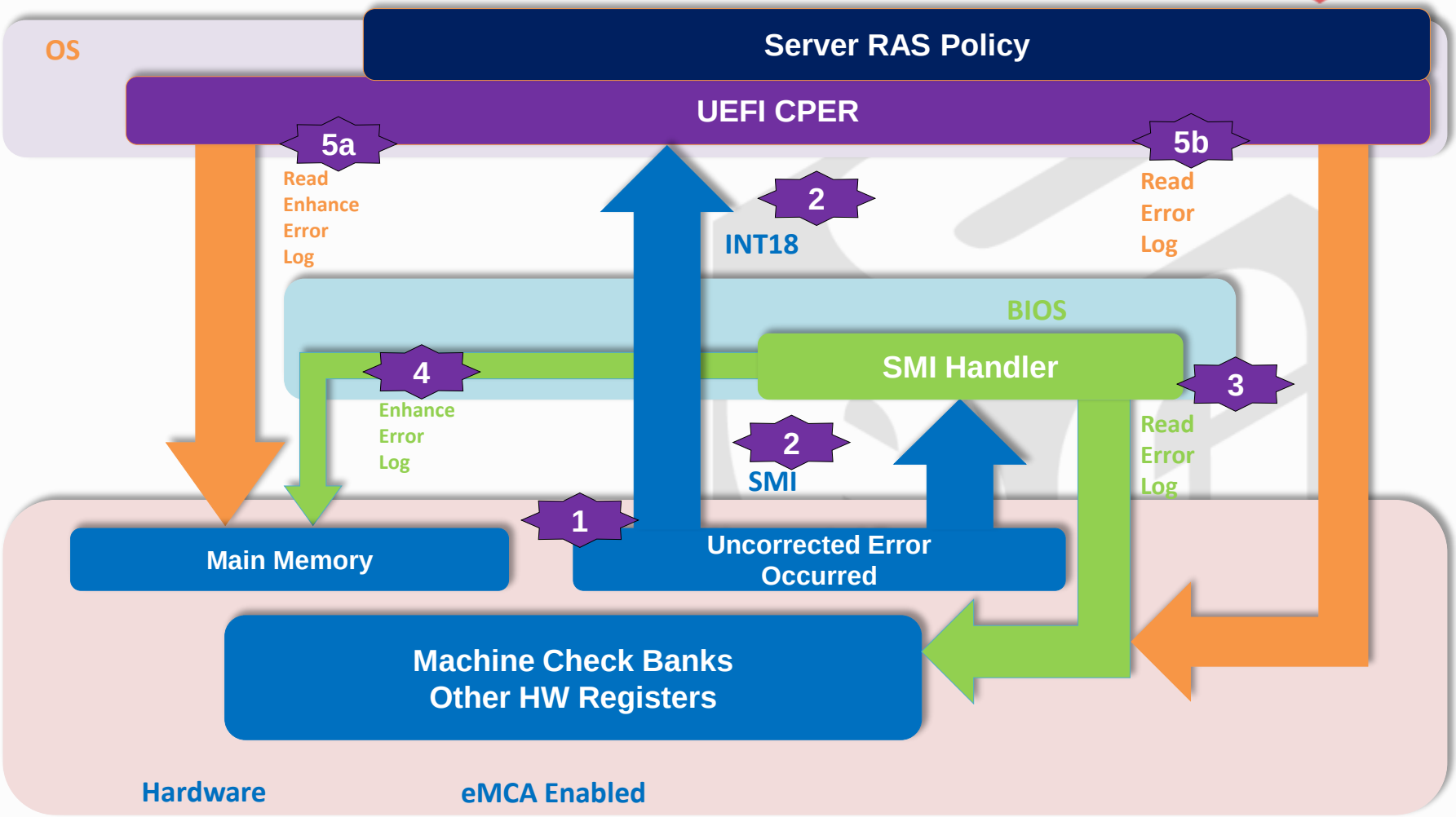
precisely map error vs. specific component

```
[   72.396625] {1}[Hardware Error]: Hardware error from APEI Generic Hardware Error Source: 0
[   72.396627] {1}[Hardware Error]: APEI generic hardware error status
[   72.396628] {1}[Hardware Error]: severity: 2, corrected
[   72.396630] {1}[Hardware Error]: section: 0, severity: 2, corrected
[   72.396632] {1}[Hardware Error]: flags: 0x01
[   72.396634] {1}[Hardware Error]: primary
[   72.396635] {1}[Hardware Error]: section_type: memory error
[   72.396637] {1}[Hardware Error]: error_status: 0x0000000000000400
[   72.396638] {1}[Hardware Error]: node: 3
[   72.396639] {1}[Hardware Error]: card: 0
[   72.396640] {1}[Hardware Error]: module: 0
[   72.396641] {1}[Hardware Error]: device: 0
```

Solution


UEFI Runtime Service using CPER
1. Classify error severity
2. Consolidate error sources
3. Standardize RAS policy
2
ERROR REPORT PROTOCOL
3
ERROR HANDLING PROTOCOL
RAS POLICY PROTOCOL
Platform UEFI Boot Service
1
1. Platform Error logging
2. Platform Error report
3. Platform Specific algorithms
OS Module
1. OS Error log
2. OS/App Error Recovery
3. Component offline or replacement

CPER w/ eMCA2


OS
Server RAS Policy
UEFI CPER
5a
Read
Enhance
Error
Log
2
INT18
5b
Read
Error
Log
BIOS
SMI Handler
4
Enhance
Error
Log
3
Read
Error
Log
2
SMI
1
Main Memory
Uncorrected Error
Occurred
Machine Check Banks
Other HW Registers
Hardware
eMCA Enabled

Call to Action

Call to Action

- Standardize UEFI Error Reporting/Error Handling/RAS Policy Protocol
- Centralize more Errors sources like PCIe AER/MCA etc.
- Connect APEI/CPER with OS-based policy like CPU/Memory/PCIe devices hotplug;
- OS-guided RAS policy back to FW and HW, like page offline.

Thanks for attending the Spring 2017 UEFI Seminar and Plugfest

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by