# Experiences and Proposals on RAS for RISC-V in Public Cloud

**Shuai Xue and Zhuo Song**

Alibaba Cloud

2025.05

# 01 Background

# What is RAS? -- Definition

**R**eliability

**A**vailability

**S**erviceability

# What is RAS? -- Definition

- Used originally by IBM to measure mainframe robustness
- **Reliability**
  - Probability that a system will produce correct outputs
  - Generally measured as Mean Time Between Failures (MTBF), 30000 hours for HDD
  - Enhanced by features that help to avoid, detect and repair hardware faults
- **Availability**
  - Probability that a system is operational at a given time
  - Generally measured as a percentage of downtime per a period of time
  - Examples:
    - 99.999% (“five nines”) means 5.26 minutes of downtime per year
- **Serviceability**
  - System capability to report failures for “FRU Isolation” and ease ofrepair.
  - FRU – Field Replaceable Unit, e.g., DIMM, PCI Express* device

## Why do we need RAS? -- Importance

### Impacts

Continuity, Computation needs be correct and reliable.

### Inevitability

Although faults are rare, enterprise systems can be very large. So failures are inevitable.

### Operating Expense (OPEX)

OPEX for maintenance is reduced by

1. replacing only failed parts
2. scheduled maintenance (is cheaper than unscheduled service outages)

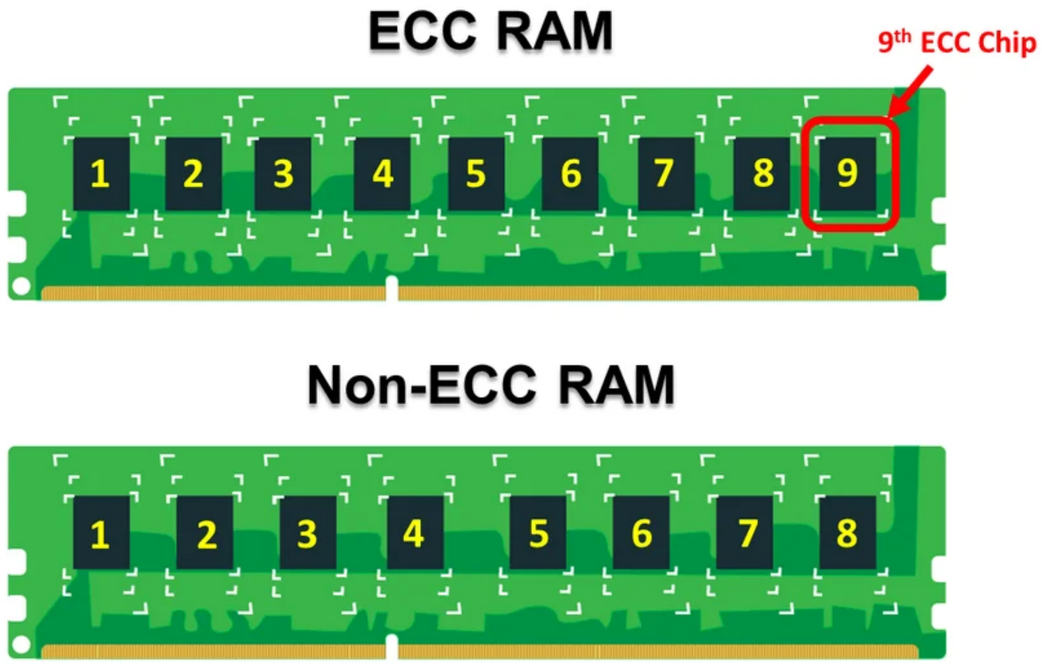
CPU Server: 2x Intel Sapphire Rapids

10K $

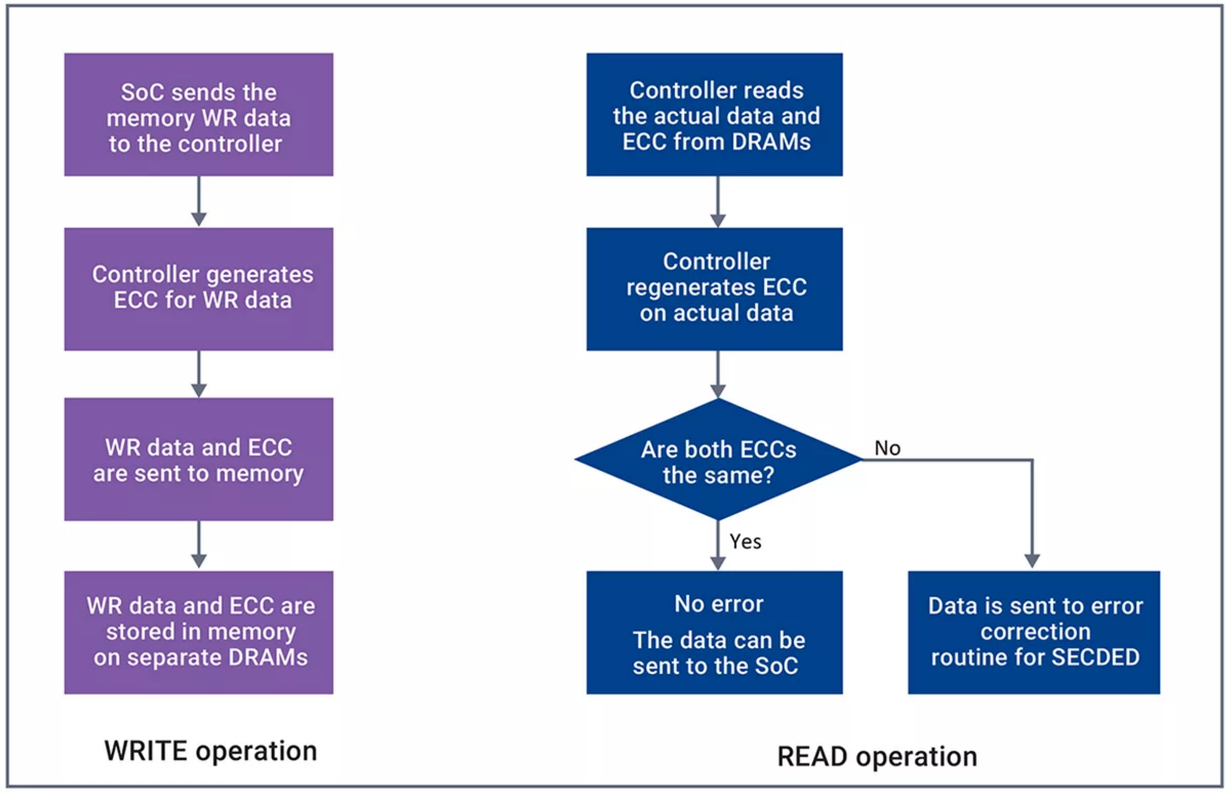
**27X cost**

AI Server: 8 NVIDIA H100 GPU + 4 NVSwitch

269K $




- ECC: a 64-bit data width, 8 additional bits are used for ECC storage
- ECC RAM contains an **additional DRAM chip** to store the ECC codes



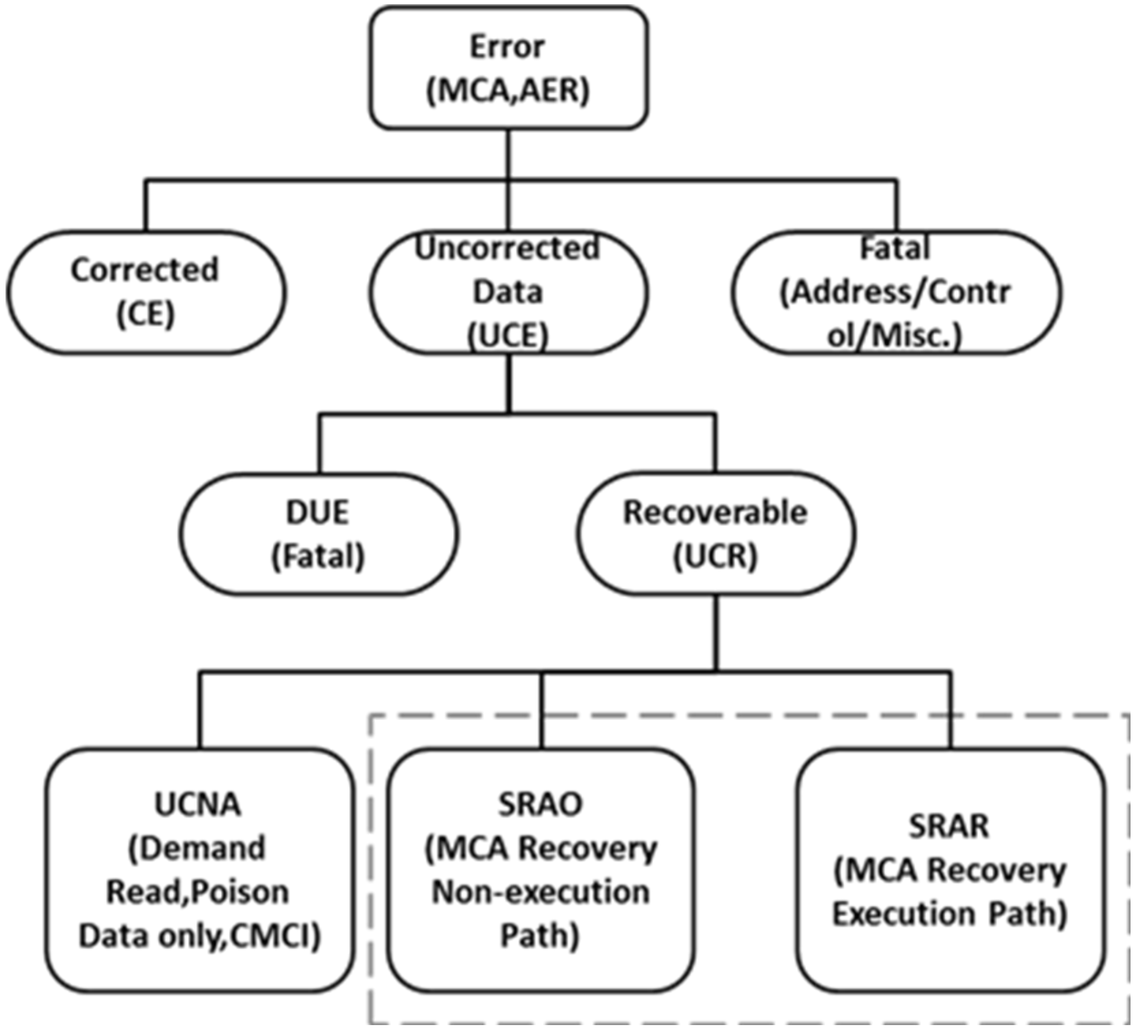

WR and RD operation flows with side-band ECC

ECC error:

- Corrected Error：1 bit error and corrected by hardware, software intervention is not required
- UnCorrected Error：2 bit error and can NOT corrected by hardware, software intervention is **required**

https://www.synopsys.com/articles/ecc-memory-error-correction.html

# Intel Xeon RAS Error Categories




- Error categories :
  - Corrected error (CE)
  - Uncorrected error(UCE)
  - Fatal

- DUE (Fatal): panic immediately
- UCNA (Uncorrectable Error No Action required): UCE detected by background scrubber
- No MCE interrupt assert, typically no action needs to be taken.

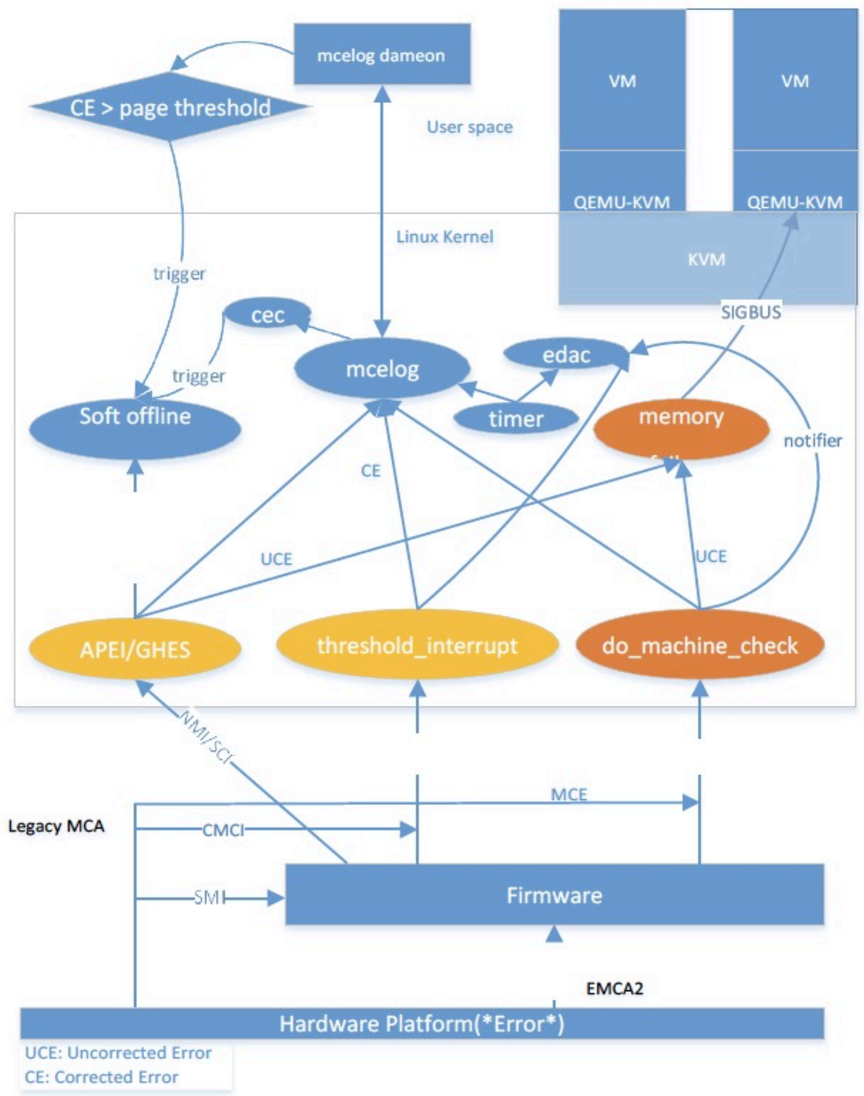
- SRAO (Software Recoverable Action Optional) – Action Optional:
  - Occur outside of program execution path.
  - OS/App is optional to take action (e.g., Offline failure page/Kill failure thread) to recover this uncorrectable error.
- SRAR (Software Recoverable Action Required) – Action Required:
  - Occur on the program execution path.
  - OS/App requires to take action (e.g., Offline failure page/Kill failure thread) to recover this uncorrectable error.

# Intel Xeon RAS Error categories


| | Mci_STATUS | | | | | | | MCG_STATUS | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Errored Thread | | Other Threads | | | | |
| | Val ID | UC | PCC | Service | AR (Action Required) | ADDRV | MISCV | RIPV | EIPV | RIPV | EIPV | Signaling | ADDR in Kernel Space | SW Action |
| Uncorrected Errors | 1 | 1 | 1 | x | x | x | x | x | x | x | x | MCERR | x | System crash. |
| SRAR – Instruction | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | MCERR | No | Take specific recovery action |
| SRAR – Instruction | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | MCERR | Yes | Kernel Panic |
| SRAR – Data Load | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | MCERR | No | Take specific recovery action |
| SRAR – Data Load | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | MCERR | Yes | May Kernel Panic |
| SRAO | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | MCERR | x | Optional for recovery action |
| UCNA | 1 | 1 | 0 | 1 | 0 | 1 | 1 | x | x | x | x | CMCI | x | Log the error and Optional for recovery action |
| CE | 1 | 0 | 0 | 1 | 0 | 1 | 1 | x | x | x | x | CMCI | x | Log the error and no corrective action required |

Key Observation : Error severity also includes two parts, hardware and **software context**

# X86 Machine Check Architecture




- Full stack involved: Hardware + Firmware + Kernel + APP
- MCA: CPU errors are provided on x86 machines since Pentium 4
  - Depending on the processor, it can also provide memory and bus errors
  - eMCA and eMCA2 are Firmware First
- CE Handling:
  1. Hardware signal Common Storage Management Interface (CSMI) in every CE in Core, Uncore, DRAM and IIO
  2. SMM handler to able to set per MCA Bank basis SMI, read and write MC Banks and enhance error logs
  3. BIOS to create APEI(WHEA) or ELOG entry for OS
  4. BIOS signals CMCI to OS
  5. OS level CE error handling
- UCE Handling:
  1. Signal Machine-check System Management (MSMI ) for every uncorrected errors;
  2. BIOS trigger the OS level error signals such as MCE, or SCI for further error diagnosis or recovery in OS Machine Check handler.

Key Observation: The trend is moving towards a Firmware First approach.

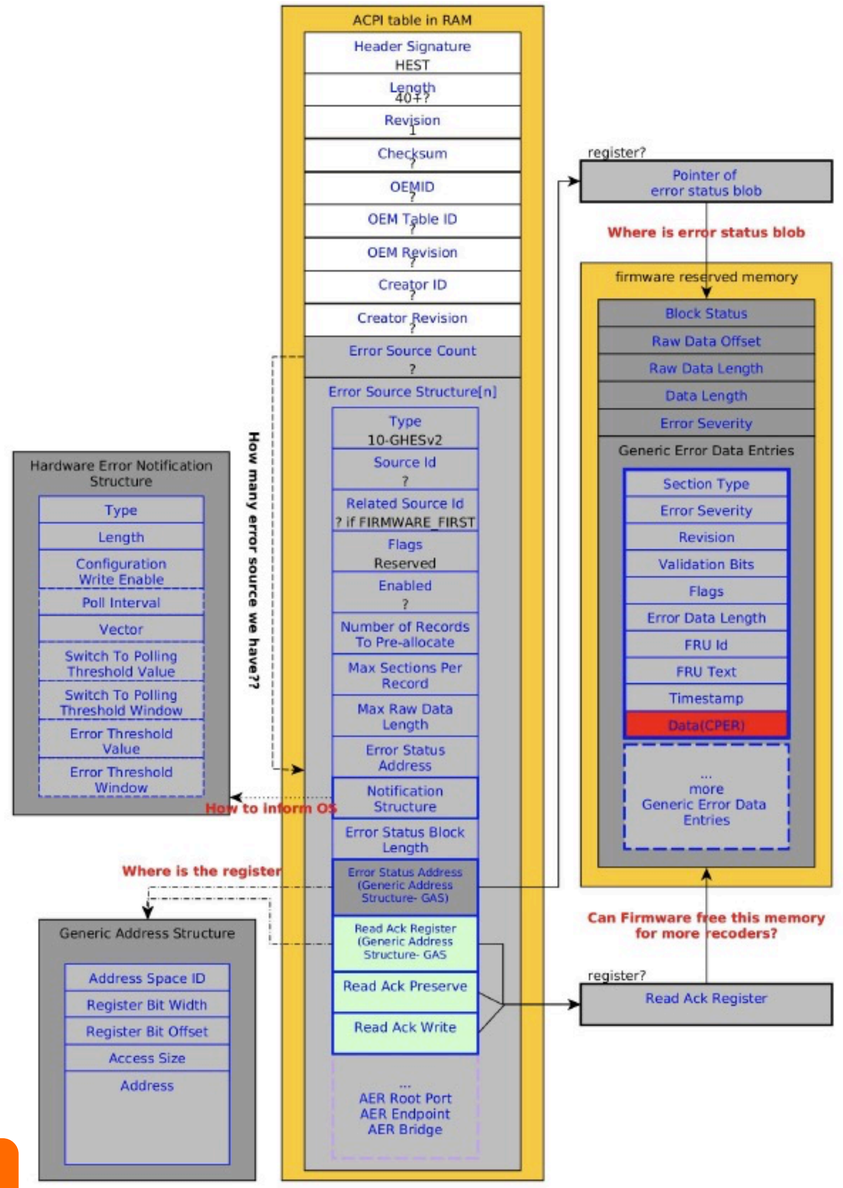
# ACPI Platform Error Interfaces

APEI (ACPI Platform Error Interfaces) defines:

- Boot Error Record Table: record errors in emergency (OS crash/reset)
- Hardware Error Source Tabe: record errors in runtime (OS still can work)
- Error Record Serialization Table: record and retrieve errors in persistent storage
- Error injection table: Test OSPM error handing stack

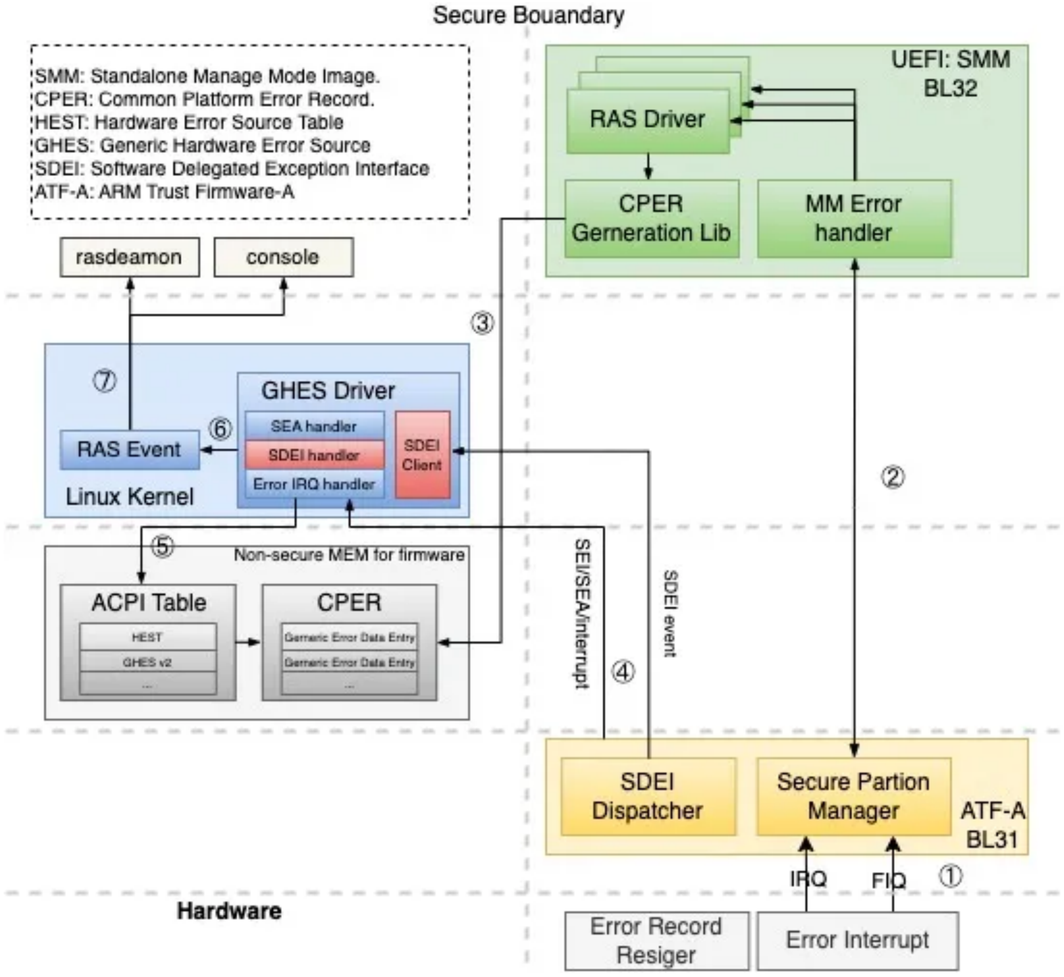
APEI HEST:

- Error Source structure:
  - IA-32: MCE/CMC/NMI
  - ARM: SEA/SDEI
  - For PCI: AER
- OS drivers: GHES driver
  - HOW to inform: Notification Structure
  - WHERE are the error records: Error Status Address
  - WHEN records can be free: Read Ack Resister




Key Observation: APEI is standard interface and used in ARM and X86

# ARM RAS Flow

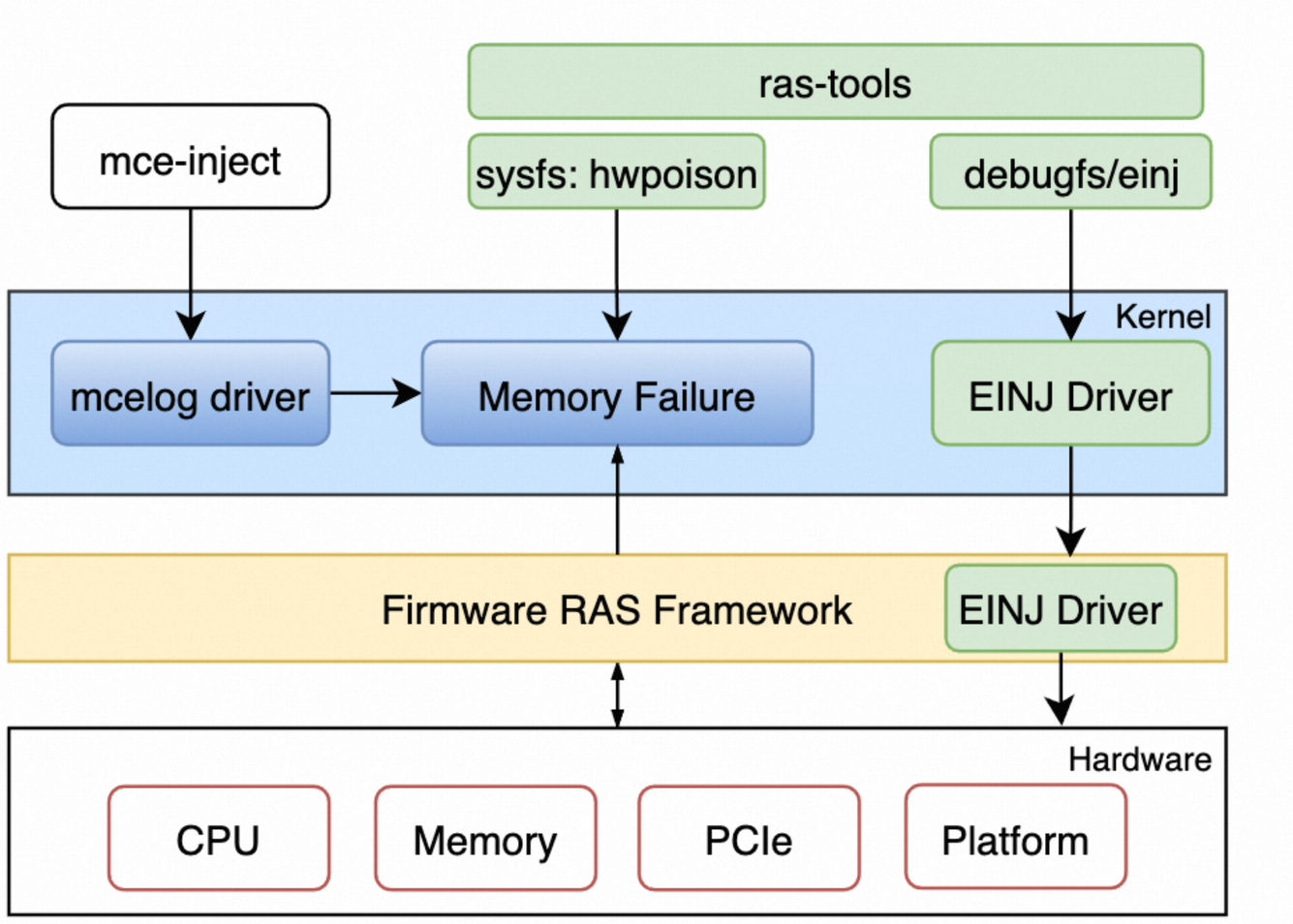



Error handling on ARM64 is Firmware First mode

1. UE occurred, the event will be routed to EL3 (SPM)
2. SPM routes the event to RAS error handler in S-EL0 (MM partitions)
3. MM Foundation (CperLib) creates the CPER blobs by the info from RAS Extension
4. SPM notifies SDEI to call the corresponding OS registered handler
5. OS gets the CPER blobs by Error Status Address block, process the error, try to recovery.
6. OS report the error event by RAS event
7. Rasdaemon log error info from RAS event to recorder

Key Observation: The trend is moving towards a Firmware First approach.

# Error Injection（EINJ）




- mce-inject based on mcelog driver
  - mcelog driver is support only on x86 and deprecated now
- Hwpoison sysfs and API：
  - Memory failure sysfs to hard or soft offline pages
  - Madvice API
- APEI Error INJection：
  - ✓ Hardware error injection
  - ✓ Hardware error signal, firmware error report and OS error containment
  - ✓ Standard APEI，support on both X86 and ARM

https://docs.kernel.org/firmware-guide/acpi/apei/einj.html

https://docs.nvidia.com/networking/display/bluefieldbsp490/ras

https://gitee.com/anolis/ras-tools

yum install ras-tools

**Error Injection**

BlueField-3 error handling may be tested by injecting errors using the following methods:

- Error injection (EINJ) ACPI table (for memory and processor errors)
- `ras-tools` (for memory and processor errors)

✓ **Note**

`ras-tools` has been verified to run with Anolis only. Use it with other OSs

Key Observation: HW/SW RAS Validation reports need to be incorporated as a basis for a new server.

- Most (58%~92%) of DIMMs with UEs have experienced preceding CEs in history.
- Over 98% of DRAM faults map to a single bank on DRAM
- 63% of nodes encounter DRAM errors from a single VM, while 90.9% experience errors from no more than three VMs.
- When error bits go beyond the error correction capability of ECC, an uncorrectable error (UE) then happens and typically leads to a node unavailability

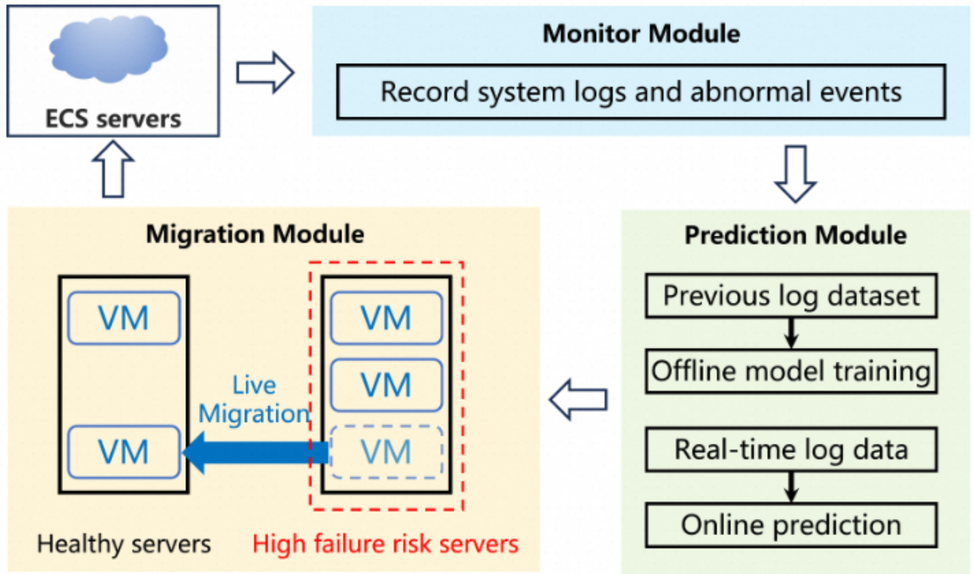



**Figure 1: Workflow of Alibaba Cloud operation and maintenance system.**

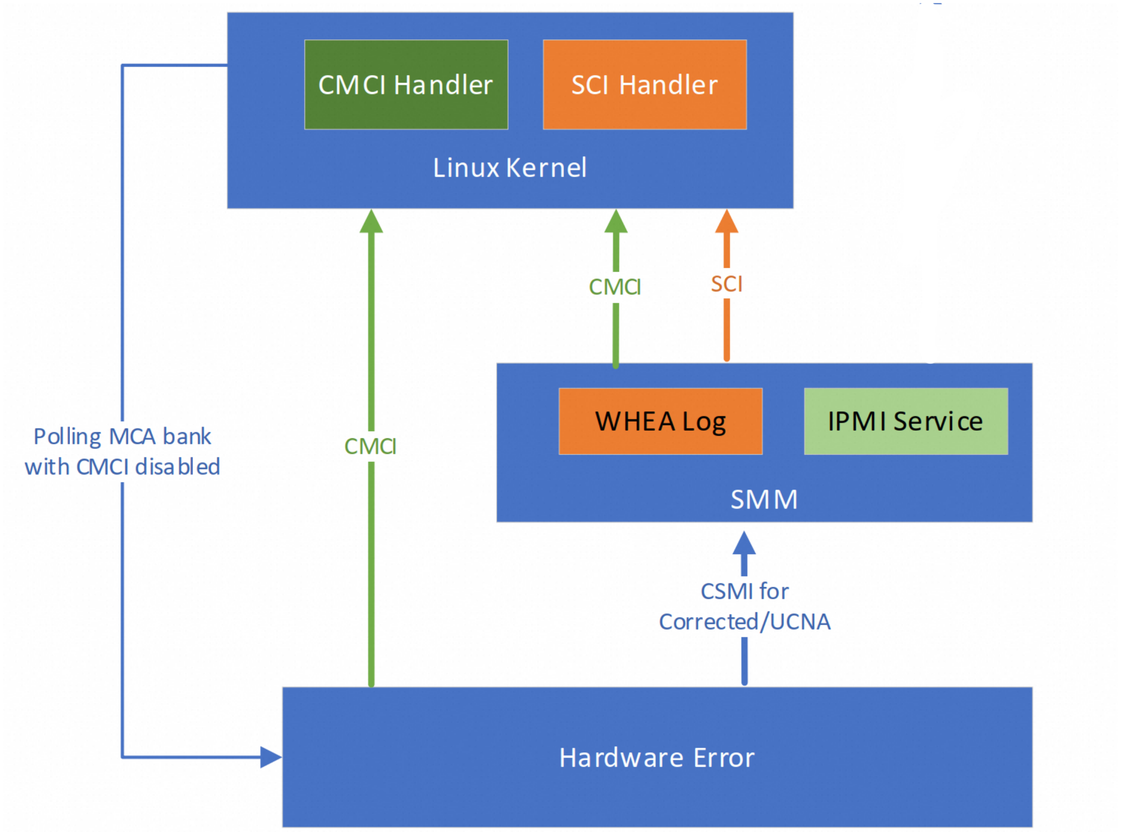
- CE samples play a crucial role in UE prediction.
- UE prediction models require extensive datasets.
- Key attributes of CE samples include error patterns, **row/column characteristics**, and ECC-related properties.
- Prioritize collecting each CE error whenever possible, instead of relying on threshold-based reporting.

Key Observation: CE samples are used to train UE prediction models.

Intel: Predicting Uncorrectable Memory Errors from the Correctable Error History

Alibaba. Predicting DRAM-Caused Risky VMs in Large-Scale Clouds. Published in HPCA2025

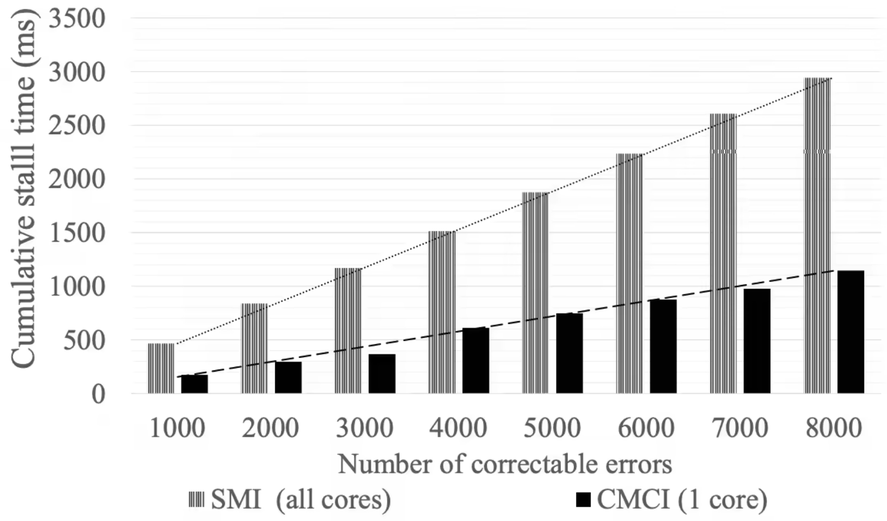
# Kernel First: Correct errors reporting

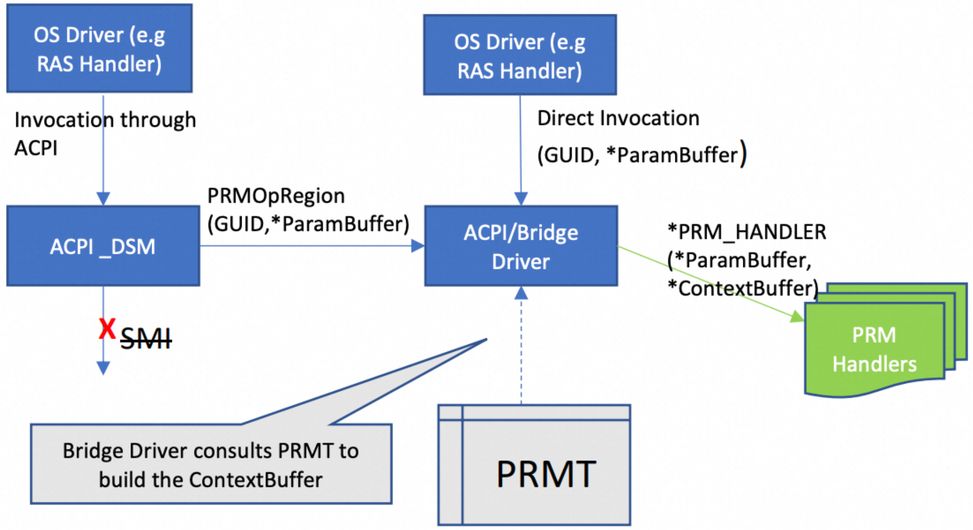



- On X86 CE is reported through two threshold-based pathways:
  - CMCI, with a threshold set to 1, and
  - CSMI, with thresholds set at 3000 or 5000.
- Kernel can switch to polling mode from interrupt mode when CMCI storm occurs.
  - Disable CMCI interrupt
  - Polling MCA bank with a timer
- ARM also adds Arm Error Source Table (AEST) for kernel first mode.
- RISC-V RERI Architecture Specification:
  - The *ces* used to enable signaling of CE when they are logged.
  - An overflow of cec is signaled using the signal configured in the ces field.
  - The physical manifestation of the signal is UNSPECIFIED by this specification.

Key Observation: Kernel First is also need to collect every CE in lightweight.
(e.g. AEST ARM or CMCI on x86)

- Correctable Errors (CE), occur more often than UE.
- Field Replaceable Unit : CE system address need to translate to DIMM address, e.g. row, column, bank, to identify faulty DIMM.
- Intel:
  - Native Address translation: hard decode by MSIC registers
  - ADXL DSM: EDAC SW SMI from EDAC driver
    - More expensive interrupt to translate address, all cores stall 3 seconds at most
    - Platform Runtime Mechanism (PRM): designed to invoke class one software handlers that do not need SMM privileges
- AMD SMA: normalized address => system address =>DIMM address




Key Observation: Implementing a lightweight address translation service in production.

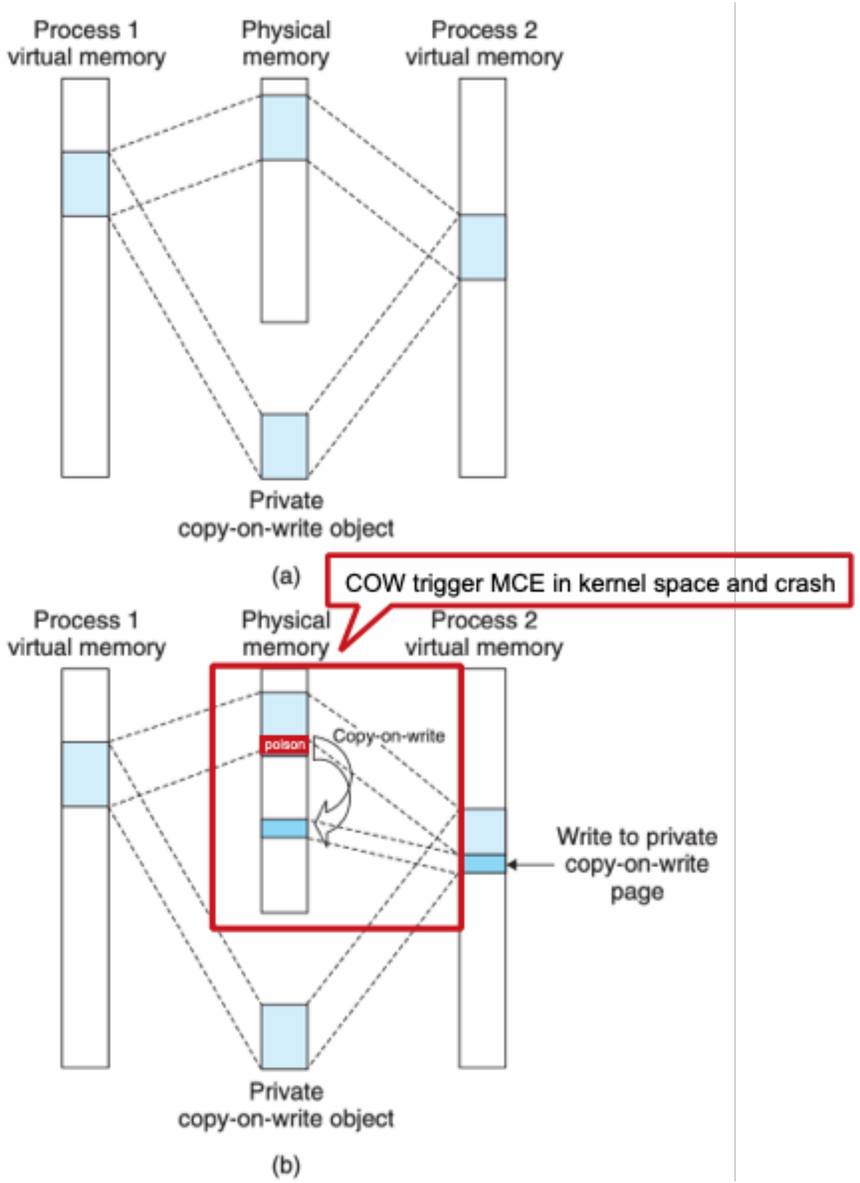
https://uefi.org/sites/default/files/resources/Platform%20Runtime%20Mechanism%20-%20with%20legal%20notice.pdf

# Context Problem: MCE trigger in Linux Kernel context

- Problem Statement: A private copy-on-write trigger page fault and copy object
  - (a) Both processes have mapped the private copy-on-write object, e.g. fork(2)
  - (b) Process 2 (child) writes to a page in the private area which triggers a protection page fault.
  - (c) The fault handler creates a new copy of the page in physical memory, updates the page table entry to point to the new copy, and then restores write permissions to the page.

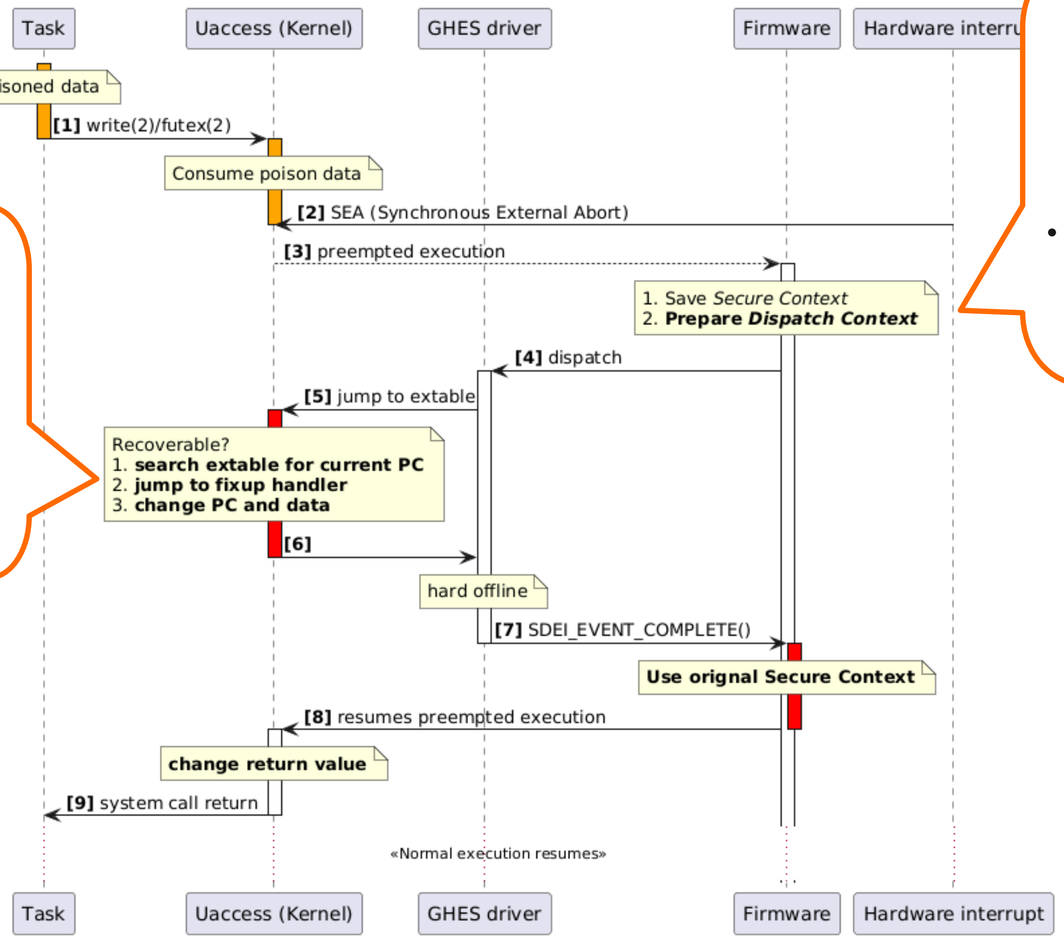
- Enhanced Actions:
  - Add a new copy_user_highpage_mc() function that uses copy_mc_to_kernel() on architectures
  - Send SIGBUS to task due to VM_FAULT_HWPOISON
  - Asynchronously take the page with the uncorrected error offline

- Similar scenarios: CoW, KSM copy, coredump copy, khugepaged, uaccess copy



# Sync Error handling limitation with SDEI/SSE notification

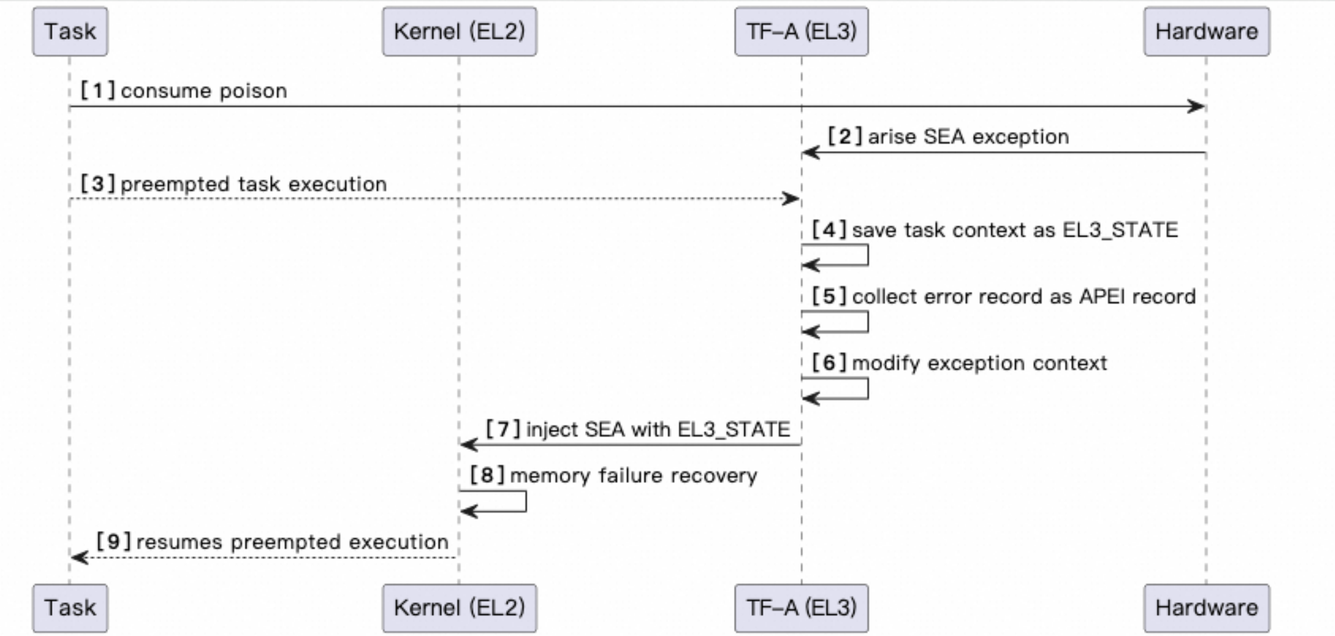
- **Software Context**
  - PC
  - General purpose register : X0-X31
- SDEI event: No Exception type?
  - Synchronous error: kill the process
  - Async error: defer to later kill




- **Hardware context**：Exception Syndrome Register (EL3)
  - Trap from EL0, EL1, EL2
  - Data abort or instruction abort
  - Recoverable state or unrecoverable state (containable)
- **Software Context**
  - PC
  - General purpose register : X0-X31

Key Observation: Kernel addressed context loss issues arising from dispatch and resume operations.

# ARM64: Sync Error handling




- Key improvements to address SDEI limitation
  - STEP 5:
    - Notify Type with ACPI_HEST_NOTIFY_SEA
    - ACPI_HEST_NOTIFY_SEA only used for synchronous errors
  - STEP 7:
    - Overwrite EL2 context
    - ERET to kernel exception vectors
  - No need to jump back to TF-A to resume

# Sync Error handling limitation with SDEI/SSE notification

- mcause 19: hardware error
  - A Hardware Error exception is a synchronous exception triggered when corrupted or uncorrectable data is accessed explicitly or implicitly by an instruction.
- M mode：
  - Save context to sse_interrupted_state
    - software context PC, hardware context mstatus
    - Inject to S mode kernel
- S mode kernel: memory failure recovery
  - **May change context PC**
  - **General purpose register for return value, e.g. –EFAULT**
- M mode：
  - Restore context from sse_interrupted_state
  - Recovered context is missing
- HART containable?

- RFC Proposal 1:
  - Determining Error Type (Sync or Async):
    - Introduces a new Notify Type: ACPI_HEST_NOTIFY_HEC
      - HEC means Hardware Error Check.
    - This new notify type is specifically used for synchronous errors.
  - Continuing Execution with Recovered Context:
    - The idea is to overwrite the S-mode context and directly jump to the kernel exception vectors for error handling.
    - This method avoids the need to switch back to M-mode to resume execution, potentially streamlining the recovery process.
- RFC Proposal 2:
  - Determining Error Type (Sync or Async):
    - a specific Software Event ID
  - Continuing Execution with Recovered Context:
    - Need to jump back to M-mode to resume recovered context


**Thank you.**