**Receiving Memory Corrected Error Notification bypassing the BIOS or Firmware using Intel Machine Check Architecture in Firmware First Mode.**

# **Abstract**

Monitoring Datacenter servers for hardware errors, alarms and any other indications about system health is an essential part of planning maintenance activities or even predictive failure analysis of critical server components. Industry has explored several approaches thus far and documented challenges with respect to tracking down memory errors corrected by the hardware in real time. Traditionally, it has been a tradeoff between high granular error information and less overhead on the system while monitoring memory modules on servers.

Here we summarize various approaches to receive corrected error notifications related to memory modules with higher level of granularity but with much lesser overhead on the system while operating in firmware first mode. As a result, the firmware could still be leveraged to trigger reliability, availability, and serviceability measures supported by Intel Xeon Scalable processor family. However, the approach may not be limited to the bespoken processor family and may be leveraged on all such Intel processors that support Intel MCA and enhanced MCA2.

In the next sections, information has been structured to introduce Intel Machine Check Architecture (MCA), difference in error notification flows due to MCA versions and then a alternative method or monitoring corrected errors via interrupts if Vendor BIOS does not propagate the error information to OS yet.

# **Introduction**

Intel Machine Check Architecture (MCA) in general is used to communicate critical hardware events to the system software or the BIOS/Firmware through a set of hardware registers. Errors could be related to memory, caches, Bus, Interconnect logic o the CPU cores and Internal timers. As far as memory errors are concerned, following events, depending on processor model, could be monitored:

* Memory read error: Errors detected while reading from a memory location.
* Memory write error: Errors detected while writing to a memory location.
* Address or command error: Errors in the memory channel. They are typically communication errors due to electrical interferences.
* Memory scrubbing error: An error detected while scanning memory locations to find and fix any errors.

## **Corrected Errors**

Some memory errors are automatically corrected by the ECC hardware and therefore they are transparent to the application that consumes data from the memory region where such errors occur. However, corrected errors are reported to the System Software or BIOS/Firmware for advisory reasons. Corrected error notification with exact location of the error is a critical piece of information to isolate failing DIMMs and thus carryout any maintenance activity.

Diagram

Description automatically generated with medium confidence

The figure above on the left depicts error notification mechanisms in Legacy MCA mode and on the right is notification flow when Intel enhanced Machine Check Architecture-2 is enabled.

Legacy MCA does not necessitate Firmware First Mode; therefore, bypasses the BIOS/Firmware completely. Consequently, several RAS (Reliability, Availability and Serviceability) features such as Page off-lining, sparing and PPR (post package repair) that require BIOS intervention can no longer be leveraged. In this mode, corrected memory errors are notified to the System software through Corrected Machine Check Interrupt (CMCI) and uncorrected memory errors are notified via a Machine Check Exception (Interrupt 18) as defined in Intel SDM volume 3B.

In enhanced Machine Check Architecture-2 mode, as depicted on the right, Corrected errors are first notified to the BIOS or Firmware. Error information is further enriched with high granular location of where the error occurred. BIOS/UEFI FW can then optionally notify the System Software via CMCI.

## **Uncorrected Errors**

Below is the error notification flow of Uncorrected errors when in FFM and while using eMCA2.

Uncorrected errors are relatively less common when compared to transient corrected errors. Notification of corrected errors via BIOS/FW to the system software might overwhelm the server [References to existing works] and may also lead to issues such as OS hang as the system transitions to System Management Mode. As an optimization technique, vendors enable only “Uncorrected error” notification and disable corrected error notification in Firmware First Mode.

Timeline

Description automatically generated

# **Alternative/Direct approach to receiving Corrected Error Notification directly to OS in FFM mode.**

Uncorrected error notification remains unchanged; therefore, sent to the BIOS/firmware first and if deemed necessary the system software is notified. This enables Machine Check Architecture Recovery at software layers. Besides ability to perform post package repair by saving the Uncorrected Error location information is preserved. Since we continue to operate in FFM mode, other RAS features such as page off-lining and sparing could also be leveraged with the help of BIOS or UEFI FW as supported by Intel processors.

Preconditions to receiving CMCI notification for memory errors directly from the hardware by the System Software layer (OS or Hypervisor) in FFM mode:

* Global error signaling is enabled in Intel MSR 0x17F.
* Intel Machine Check Global Capability register MSR- 0x179 enables CMCI.
* Any other relevant control registers at individual machine check bank levels are not used to disable CMCI notifications to the System Software layer.
* CMCI to SMI morphing (as in eMCA2) is “not enabled” by the BIOS or UEFI Firmware. If enabled, direct CMCI notifications from hardware to the system software would not take effect.

If the above preconditions are met the system software (OS/Hypervisor) is allowed to subscribe for CMCI directly from the relevant machine check architecture banks of interest. The procedure for subscribing such CE notifications remains same as described in Intel SDM Volume 3B (version and exact section). CMCI should be enabled by setting the appropriate bit of MCi\_CTL2 MSRs of relevant banks (IMC and M2M banks for memory corrected errors).

# **Procedure to fetch high granular memory error location information – leveraging approach discussed earlier.**

Based on the CMCI notification, system software can extract the physical address location of the memory error from MCA status registers of relevant banks and further apply address decoding rules to fetch high granular information pertaining to the location of the error (Linux EDAC driver for skylake skx\_edac consists of reference code).

BIOS or UEFI FW could continue to use Rank level threshold counters defined in PCICONFIG space to monitor memory errors. Furthermore, memory error counters exceeding the predefined threshold could be used as a trigger to fetch location information using “error cloaking”. When error cloaking is enabled temporarily, BIOS/UEFI Firmware could consume the hardware error information and any reads of MCA Bank registers from the System Software to fetch error information would return invalid or no information. Alternatively, UEFI/Firmware could also use the shadow registers of M2M banks to fetch the location of errors when rank level memory counters exceed predefined threshold.

# Acronyms

MCA machine Check Architecture

eMCA-2 Enhanced Machine Check Architecture version 2

CMCI Corrected Machine Check Interrupt

MCE Machine Check Exception

FFM Firmware First Mode

SMI System Management Interrupt

SMM System Management Mode

CE Corrected Error

UCE Uncorrected Error

Intel SDM Intel Software Developer’s Manual

UEFI FW Unified Extensible Firmware Interface Firmware

BIOS Basic Input/Output System

PCICONFIG Peripheral Component Interconnect Configuration

M2M Mesh to Memory

IMC Integrated Memory Controller

RAS Reliability Availability and Serviceability

References:

<https://www.intel.com/content/dam/develop/external/us/en/documents/emca2-integration-validation-guide-556978.pdf>

<https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html>