TECHNOLOGY GUIDE

Intel Corporation

Reliability, Availability, and Insights - Reliability, Availability, and Serviceability in Telecommunication Networks

Authors

David Kennedy

1 Introduction

Ensuring reliable and available equipment is one of the hallmarks of telecommunications networking infrastructure in order to provide exceptional quality of service for customers. This is increasingly important given the growing traffic demands on data networks, the ubiquitous use of fixed and mobile infrastructure in our everyday lives, the broadening of services dependent on always-on access, and the growth of IoT infrastructure for services at the edge. Issues relating to system failures in live infrastructure, network configuration issues, degraded quality of service, reliability, and performance and challenges around manageability of hardware in the field contribute to increased support overheads, and impact on customer relations. Network outages are disruptive to almost every aspect of our modern lives, both personal and professional, with the potential to damage the credibility of Comms Service providers (CoSPs).

The cost of network downtime not only potentially results in reputational damage to network operators but can also result in huge fines for CoSPs. Recent cases involving significant downtime of between 24-48 hours have resulted in fines in the order of tens of millions of dollars. The gradual transition to a Software Defined Network (SDN) over the past years has created a state where issues in a virtualized environment are mitigated by redeploying the affected instance. However, while Network Function Virtualization (NFV) architecture is critical in delivering 99.999% reliability, it creates a space where hardware is abstracted from the troubleshooting process. Progress in NFV can leverage the advancements in telemetry from MANO (Management and Orchestration), to provide an infrastructure where telemetry can generate abundant amounts of system level data to help orchestrate and manage network health.

When systems in the field fail and reach an unrecoverable state, there are massive amounts of overhead in terms of (1) identifying the root cause and (2) repairing the issue(s) as efficiently as possible to ensure minimal disruption. Building upon decades of Intel's advancements in Reliability, Availability, and Serviceability (RAS) competencies, the 4th Gen Intel® Xeon® Scalable processors provide an enhanced set of hardware features to not only improve the overall reliability of the hardware, but also expose additional aspects of availability to reduce unplanned downtime, increase access to debug level information, and to provide additional indicators on the overall health of the underlying Silicon. In addition, troubleshooting tools in the 4th Gen Intel Xeon Scalable processors can be utilized to further mitigate the necessity to physically be present whenever a hardware failure occurs, and provides the ability to remotely access the infrastructure.

This paper intends to present the architecture for a RAS capable network, introduce some of the new features in the 4th Gen Intel Xeon Scalable processors, and explore how these features can be deployed in a customer network environment to:

- Maximize availability, reduce unplanned operation downtime scenarios over the lifespan of the equipment, and return systems to service quickly
- Deliver insights to help monitor systems, drive actions and optimizations for operations as well as workload execution
- Provide uncompromised quality to immediately and correctly identify anomalies, safely contain, and remediate deviations from normal operations, with the ability to perform updates with minimal impact or downtime

This document is intended for communication service providers who are planning and deploying network infrastructure running on the latest Intel Xeon Scalable Processors, and who would like to greater understand the capabilities for a more reliable network.

This document is part of the Network Transformation Experience Kit, which is available at https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits.

Table of Contents

1 Introduction ... 1
1.1 Terminology ... 2
1.2 Reference Documentation ... 3

2 Overview ... 3
2.1 Use Cases ... 5
2.2 Technology Description ... 6
2.2.1 Enhanced Processor Error Pins ... 6

3 Technology Overview ... 7
3.1 System Updates ... 7
3.1.1 Seamless Firmware Update (SFU) ... 7
3.2 Error Reduction, Handling, and Containment ... 8
3.2.1 Scan-at-Field ... 8
3.3 Analysis and Monitoring ... 9
3.3.1 Intel® Platform Analysis Technology (Intel® PAT) ... 9
3.3.2 Intel® Platform Monitoring Technology (Intel® PMT) ... 10
3.3.3 Memory Failure Prediction ... 10
3.4 Service Assurance ... 10
3.4.1 Intel® Resource Director Technology (Intel® RDT) ... 10
3.5 Root Cause Analysis ... 11
3.5.1 Intel® At-Scale Debug (Intel® ASD) ... 11
3.5.2 Crash Dump ... 11
3.5.3 Crash Log Enhancements ... 11

4 Deployment ... 12
4.1 Out of Band Management ... 13
4.2 OOB Management Services Module ... 14
4.3 MCTP ... 15
4.4 Redfish ... 16
4.5 Telemetry ... 16

5 Summary ... 17

Appendix A Seamless Firmware Update ... 17

Appendix B BIOS Configuration ... 18

Appendix C Platform Firmware Resiliency ... 21

Appendix D 4th Gen Intel Xeon Scalable Processor Features ... 21
D.1 128B ECC Mode Partial Poisoning ... 21
D.1.1 Patrol Scrub Handling of Errors ... 22
D.1.2 DDR5 On-Die Error Check and Scrub ... 22
D.1.3 Error Correction Flow ... 23

Appendix E Standard RAS Features ... 23

Appendix F 4th Gen Intel Xeon Scalable Processor Advanced RAS Features ... 25

Figures

Figure 1. Overview of Types of Faults ... 4
Figure 2. 4th Gen Intel Xeon Scalable Processors RAS Coverage ... 6
Figure 3. Sideband Signal Flow for Error Management ... 7
Figure 4. Seamless Firmware Update Blackout Period ... 8
Figure 5. Components for Scan-At-Field Deployment ... 8
Figure 6. Intel® At-Scale Debug Interface ... 11
Figure 7. Crash Log Hardware and Software Components ... 12
Figure 8. RAS Architecture Overview ... 12
Figure 9. Overview for Out of Band Management ... 14
Figure 10. Out of Band Management State Machines ... 15
Figure 11. MCTP Platform Management Subsystem ... 16
Figure 12. Telemetry and Insights Infrastructure ... 17
Figure 13. Seamless Firmware Update Runtime Requirements ... 18

Figure 14. Error Correction Flow 23

Tables

Table 1. Terminology 2
Table 2. Reference Documents 3
Table 3. RAS Usage Models 5

Document Revision History

REVISION	DATE	DESCRIPTION
001	May 2022	Initial release.
002	July 2022	Revised the document for public release to Intel® Network Builders.

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
AER	Advanced Error Reporting
ASD	At-Scale-Debug
BCLK	Bit Clock
BMC	Baseboard Management Controller
BKC	Best Known Configuration
CATERR	Catastrophic Error
CHA	Caching and Home Agent
CLI	Command Line Interface
CoSP	Comms Service Providers
CSR	Control Status Register
DMTF	Distributed Management Task Force
DPM	Defects Per Million
DTLB	Data Translation Lookaside Buffer
DUE	Device Under Error
EID	Endpoint ID
eSPI	Embedded Serial Peripheral Interface
FRU	Field Replaceable Unit
IIO	Integrated Input/Output
IOSF	Intel On-Chip System Fabric
IPMI	Intelligent Platform Management Interface
ITP/XDP	In Target Probe
LMCE	Local Machine Check Error
LLC	Last Level Cache
M2M	Mesh to Mesh
M2IOSF	Mesh to Intel On-Chip System Fabric
MCA	Machine Check Architecture
MCE	Machine Check Error
MCTP	Management Component Transfer Protocol
ME	Management Engine

MFP	Memory Failure Prediction
MSR	Model Specific Register
NEBS	Network Equipment Building System
NVD	Non-volatile DRAM
OOB	Out of Band
OOBMSM	Out of Band Management Services Module
PAT	Intel® Platform Analysis Technology
PECI	Platform Event Controller Interface
PMT	Intel® Platform Monitoring Technology
Psys	Power into System
RAS	Reliability, Availability, and Serviceability
RDT	Resource Director Technology
RMCA	Recoverable Machine Check Architecture
SAF	Scan At Field
SHC	System Health Check
SFU	Seamless Firmware Update
SRAO	Software Recoverable Action Optional
SRAR	Software Recoverable Action Required
TMA	Top Down Micro-architecture Analysis
TOR	Timeout Re-Order
Ubox	Utility Box (Configuration Agent)
UCNA	Uncorrected No Action
UCR	Uncorrected Recoverable
UPI	Ultra Path Interconnect

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Platform Analysis Technology	https://www.intel.com/content/www/us/en/developer/topic-technology/platform-analysis-technology/overview.html
Redfish Specification	https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.15.0.pdf

2 Overview

Reliability, Availability, and Serviceability (RAS) traditionally breaks down into three specific areas:

1. Reliability ensures that network hardware can detect, repair, and avoid errors to deliver a defined probability of system uptime (often referred to as Mean Time Between Failures). Its primary objective is to ensure that a performant system is maintained through suppression, isolation, and repair of errors on the platform.
2. Availability is equated to the amount of time a system is available over the course of a period of time. Typically, network providers insist on hardware being available up to 99.999% per year, equating to around 30 minutes downtime for planned and unplanned downtime.
3. Serviceability relates to the ability to update and maintain equipment in the network. This is particularly important to equipment in the network located in remote locations where a hard down scenario can be on the order of hours in order to allow a service engineer to physically access the site.

More recently, considerations around secure-ability, manageability, debuggability, and sustainability have come to the forefront when delivering a robust and safe network.

Reliability is critical to the function of every aspect of the network, ranging from the data center infrastructure at the core of the network, to the thousands of converged edge locations. While telecoms grade platforms are fundamentally designed to achieve the highest standard of reliability through Network Equipment Building System (NEBS) compliance (encompassing thermal margin, vibration, failover, and RF interference) and redundancy. However, systems are still vulnerable to unplanned downtime through software faults, software misconfigurations, as well as hardware faults, with the latter being primarily susceptible to environmental conditions. Hardware errors can range from transient errors, e.g. electrical noise from high speed interfaces, soft errors, e.g. from high energy particle strikes in cache and memory, to persistent faults, e.g. stuck at faults and device failures. Memory continues to be one of the highest risk platform elements to be exposed to these types of errors, and in addition to the incremental optimizations made over the last number of generations to increase fault coverage and scrubbing capabilities for potential errors, new techniques are in development to accurately predict and report failing areas in memory at the network level.

Detectable errors can be further categorized into correctable (generally with a negligible effect on the platform) and uncorrectable, which may scale from anything from a recoverable fault on the system to a catastrophic failure, which through a controlled power down and migration will result in a system having to be taken out of service. The impact for these scenarios ranges from depreciated performance within the network to critical hardware downtime, which may result in service interruptions.

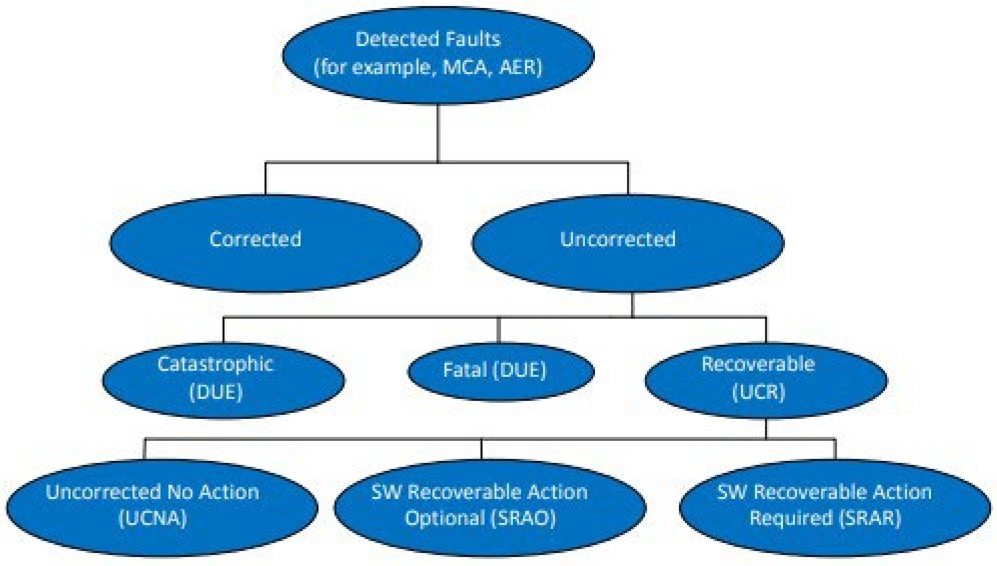


Figure 1. Overview of Types of Faults

Faults can be generated from a vast array of resources, and can potentially range from a correctable error, which can be seamlessly handled by the system, to uncorrectable errors, which span everything from recoverable (UCR) through software, Fatal, which may impact on system behavior to Catastrophic, which will result in a system hang condition.

The transition towards a Software Defined Network, with a highly orchestrated infrastructure, can ensure that any performance impact to a virtual machine(s) or container(s) can quickly be remediated through live migration or replacement to ensure 99.999% reliability. This can lead to scenarios where, for example, errors are worked around rather than fully understood and contribute over time to an overall system failure.

While the orchestrated software approach is useful for mitigating minor errors, consideration needs to be provided for complex systems errors. Typically, these scenarios would require physical presence and in-depth technical knowledge. With enhancements to Intel's extensive RAS capabilities in the Intel® Xeon® Scalable processor product line, customers can expedite a multi-vendor debug approach to respond and debug complex issues in a remote environment with the right interfaces and infrastructure in place.

This paper will aim to provide a foundation for implementing RAS infrastructure on 4th Gen Intel® Xeon® Scalable processors, looking at the key components to optimize hardware error management and reporting, outline new technologies to predict failures in both test as well as in-field environments, and provide details regarding tools to debug complex issues.

2.1 Use Cases

The below table provides an outline of the primary use cases for RAS technologies, as well as some of the resulting actions that a system can enforce to mitigate them.

Table 3. RAS Usage Models

	RAS Usage Model		Benefit	Key Use Cases (System Level)	RAS Scope
1	Meet Service Level Agreement in the presence of machine faults		Maintain the SLA and minimize the duration of downtime	I. Perform Predictive Failure Analysis II. Isolate Faulty FRU	Serviceability
2	Achieve Target Service Availability in the presence of machine faults	a. Reduce duration of service downtime (Serviceability)	Reduce downtime and increase service availability		
		b. Increase service uptime in presence of machine faults	Increase service uptime and increase service availability	III. Maintain the service when transient (HW Correctable Error) fault is detected IV. Maintain the service when persistent fault is detected V. Recover the service when HW Uncorrectable Error (UC) is detected VI. Throttle Mem BW in case of higher than target temp to prevent machine failure	Reliability
		c. Maintain data integrity in presence of DUE	Minimize risk of data corruption	VII. Prevent Corrupted Data to exit an I/O port to persistent storage device	
		d. FRU replacement for nonstop operation	Maintain target availability	Replace Compute node without bringing down the whole system	

3	Field Debug	Simplify Debugging in Field	IX. Identify failed FRU in the field deployed system	Debug
4	RAS Validation using Error Injection	Systematic and fast method	X. Validate RAS Tech Using Error Injection Capabilities	Validation

2.2 Technology Description

Figure 2 provides an outline of the numerous hardware features available on the 4th Gen Intel Xeon Scalable processors. As can be seen from the below diagram, error management encompasses almost every aspect of the core, with numerous logical blocks dedicated to RAS functionality for core, IO, and memory functions.

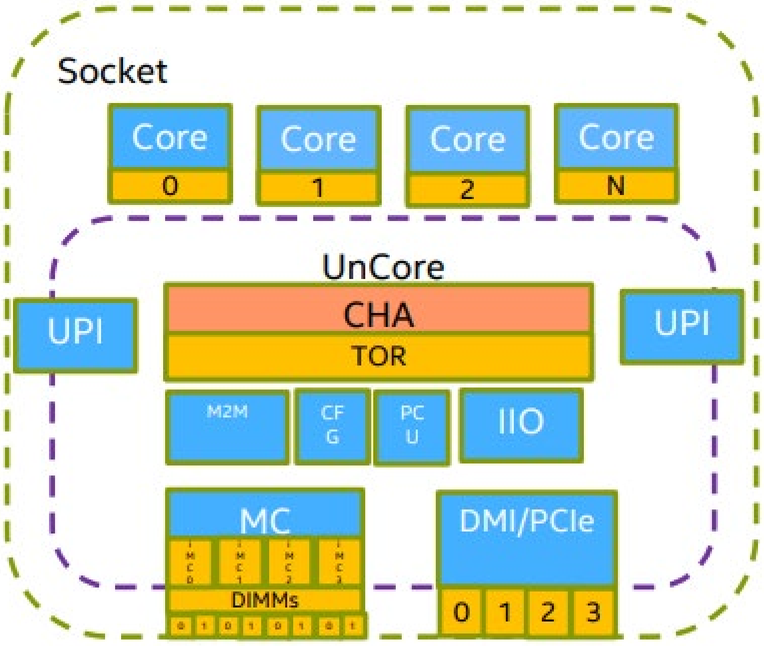


Figure 2. 4th Gen Intel Xeon Scalable Processors RAS Coverage

The above block diagram details the fault coverage of Intel® Xeon® architecture incorporating a number of key blocks for monitoring core, uncore, and memory errors:

- Machine Check Errors (MCE) provides coverage across both the core and uncore functional blocks
  - Core MC banks provide error description for execution engines, L1 and L2, which capture errors such as timeout conditions, retries and corrupt data containment
    - Pipeline Errors/I-Cache/D-Cache/Data Translation Lookaside Buffer (DTLB)/Snp, fetch, fill
  - Uncore MC Banks spans the Last Level Cache (LLC), IO blocks (PCIe, UPI) as well additional silicon components
    - Caching/Home Agent (CHA) (Parity/Request Transaction ID, Intel® Ultra Path Interconnect (Intel® UPI), Integrated Memory Controller (iMC), Utility Box (uBox)
  - Advanced Error Reporting (AER) provides an extended range of error reporting for PCIe for conditions such as root complex considerations, masking errors and uncorrectable errors

Some of the new features outlined above for the 4th Gen Intel Xeon Scalable processor core include an improved implementation of 128B ECC Partial Poisoning, enhanced patrol scrub error handling for logging of repeated failures in memory, and support for DDR5 error check and scrub (ECS). In addition, numerous new technologies at a platform level provide a comprehensive suite of capabilities to enhance the network architecture. (Refer to Section 3).

The following section outlines some of the new package level features incorporated in this generation, which requires adherences from a hardware implementation.

2.2.1 Enhanced Processor Error Pins

The 4th Gen Intel Xeon Scalable processor redefines the functionality of the sideband error signaling as follows:

- CATERR_N: Catastrophic error signal combines the signaling of the uncorrected non-recoverable and fatal error to remote processors and Baseboard Management Controller (BMC):
    - The pin combines the functionality of the legacy Cathern and the MSMI_N pins.
    - A 16-Bit Clock (BCLK) pulse signals the occurrence on an uncorrected non-recoverable error condition.
    - If permanently asserted, this signal indicates the detection of an IERR or catastrophic error in the core or uncore.
  - RMCA_N: Recoverable Machine Check Architecture (MCA) event:
      - Availability of 16-Bit Clocks (BCLKs) pulse signaling an Uncorrected Recoverable Error (URE).
      - It will not toggle on the Local Machine Check Exceptions (LMCE) nor the Uncorrected No Action Required (UCNA) event
      - Note: The CATERR_N and RMCA-n are not expected to be asserted for the same error
  - Advanced Error Reporting (AER) Domain: Error_N[0, 1, 2]
    - Error_N[0]: Reports the corrected errors in the IIO, the iMC's, or the Intel® Optane™ PMem DDRT alert and the SMBus hang condition
    - Error_N[1]: Reports the Uncorrected errors in the IIO
    - Error_N[2]: Signals fatal errors in the IIO

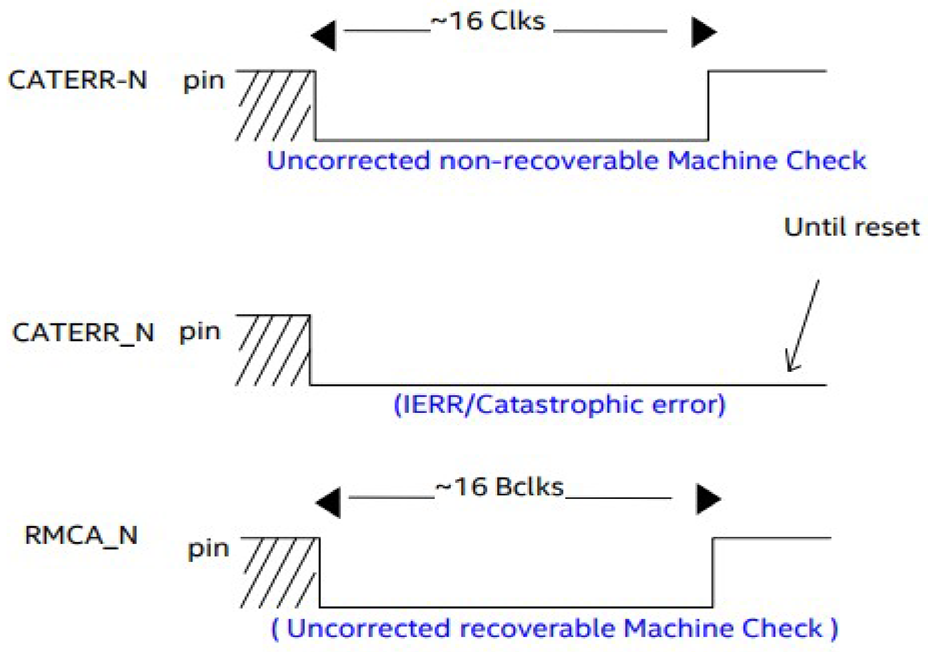

~16 Clks
CATERR-N pin
Uncorrected non-recoverable Machine Check
Until reset
CATERR_N pin
(IERR/Catastrophic error)
~16 Bclks
RMCA_N pin
(Uncorrected recoverable Machine Check)

Figure 3. Sideband Signal Flow for Error Management

3 Technology Overview

3.1 System Updates

System upgrade technologies are intended to minimize the customer impact of firmware upgrades on Intel Xeon platforms and to apply changes in the network faster and at scale. The new features of the 4th Gen Intel Xeon Scalable processor deliver higher system availability, improved reliability, better security, and SLA compliance.

3.1.1 Seamless Firmware Update (SFU)

SFU allows for updated firmware components and microcode with no perceived degradation to services running on the platform, effectively eliminating blackout periods during the upgrade process. This reduces the update from multiple minutes to less than ten seconds, while allowing applications to continue uninterrupted and targets all Intel firmware blocks. Updates can be carried out by a remote operator and the platform is required to have Out Of Band provisioning enabled with OpenBMC.

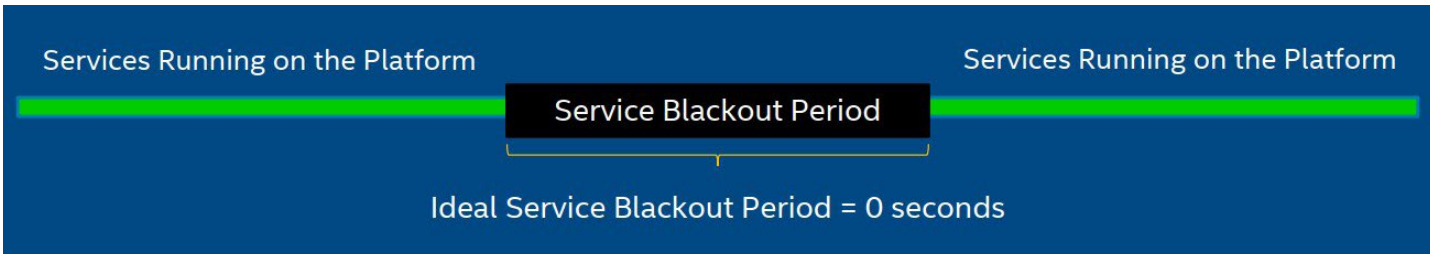

Services Running on the Platform
Service Blackout Period
Services Running on the Platform
Ideal Service Blackout Period = 0 seconds

Figure 4. Seamless Firmware Update Blackout Period

Seamless Firmware Update (SFU) reduces the service blackout period in a virtualized environment by activating a kernel pause and resume, with VM memory state preserved, to update the affected firmware. For non-virtualized environments, this is effectively able to operate without any intrusion on the running VMs.

3.2 Error Reduction, Handling, and Containment

To maximize the platform availability, the goal of error reduction, handling, and containment is to ensure high quality error suppression and to provide a resilient and robust platform utilizing unintrusive system health checks, diagnostics, and automated scanning tools to identify potential areas of risk.

3.2.1 Scan-at-Field

Scan-at-Field is intended to detect defective CPUs in the field at runtime with minimal overhead on applications. By applying scan patterns in post-production environments, it can detect stuck-at-faults for cores to enhance early life failure detection (DPM) and help improve the overall product lifecycle. A Scan Test Pattern File (blob) runs a low impact algorithm from User Space to interrogate the cores to assess any faults. When an error occurs on a platform, it is notified through the Machine Check Architecture Registers as an uncorrectable error and upstreams the event to an operator.

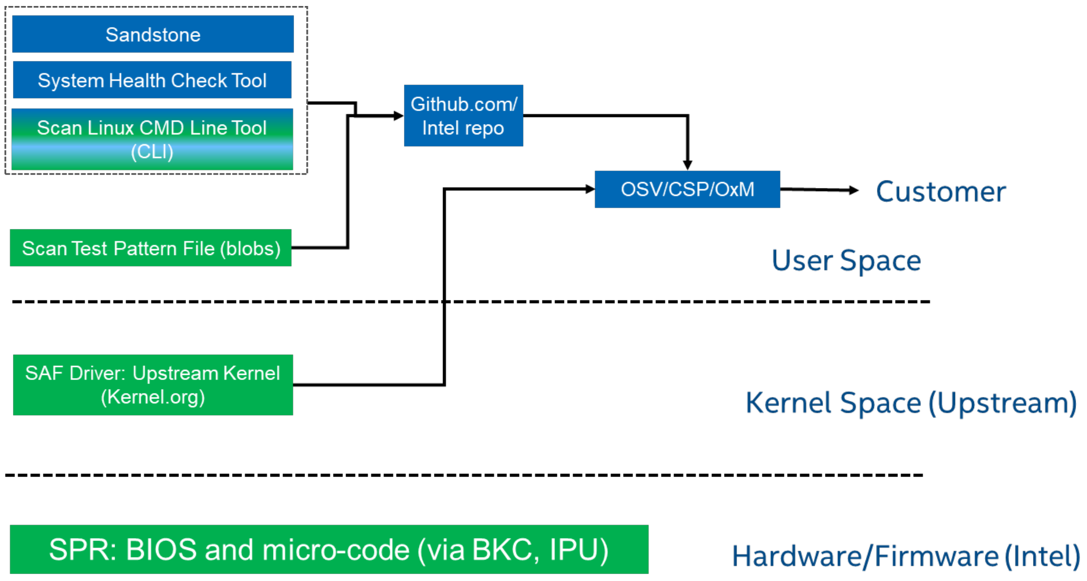

Sandstone
System Health Check Tool
Scan Linux CMD Line Tool (CLI)
Github.com/ Intel repo
OSV/CSP/OxM
Customer
Scan Test Pattern File (blobs)
User Space
SAF Driver: Upstream Kernel (Kernel.org)
Kernel Space (Upstream)
SPR: BIOS and micro-code (via BKC, IPU)
Hardware/Firmware (Intel)

Figure 5. Components for Scan-At-Field Deployment

It has a coverage of greater than 90% of core stuck-at-faults, and targets 200 ms end to end runtime at lowest frequency for all test content to ensure minimal disruption to availability and performance. Scan test pattern is authenticated prior to use and stored in a secure memory address range accessible only to the microcode. The Linux Kernel Mode driver and the Linux CLI (Command Line Interface) support Test Tool: System Health Check (SHC)

Scan At Field requires enablement as part of the BIOS Boot Setup along with additional blobs to be stored in memory. Upon software initialization, the UCODE retrieves SCAN data from its location in DRAM (Dynamic Random Access Memory), authenticates it and writes it into the protected DRAM range (PRMRR). Chunks are stored in an encrypted state after authentication. Software then verifies the authentication of SCAN data by examining the state of the rdmsr (CHUNKS_IDENTIFICATION_STATE).

During execution, software writes to all logical cores via **wrmsr** commands, transitions the core to the **MWAIT** state, and executes the **Scan at Field** test. Test results are made available via *msr* registers for handling in software or alternatively can be configured to trigger an uncorrectable error through the MCA registers.

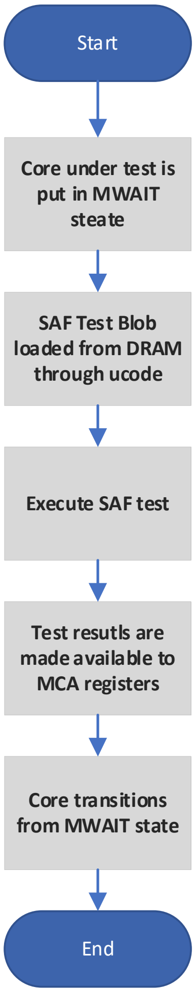

Start
Core under test is put in MWAIT steate
SAF Test Blob loaded from DRAM through ucode
Execute SAF test
Test resutls are made available to MCA registers
Core transitions from MWAIT state
End

Figure 4. Scan at Field Test Pattern Flow

3.3 Analysis and Monitoring

Additional tools for Intel platforms have been made available to help improve monitoring by making data collection more discoverable and easier to manage as well as simplifying telemetry gathering, processing and reporting. The intention is to unify and log exposure across Intel® architectures to provide a consistent approach across devices and deployments.

3.3.1 Intel® Platform Analysis Technology (Intel® PAT)

Intel® Platform Analysis Technology (Intel® PAT) delivers functionality to enable monitoring, diagnosis, and optimization of platforms based on Intel architectures. Intel PAT can also be used for data center management, cloud resource management, and security monitoring.

Platform Analysis Technology provides a set of software tools to manage, monitor, diagnose, and optimize Intel platforms including the following:

- **Architectural Last Branch Records (LBRs):** Records the last known branches and events in processor registers, including source address, destination address, cycle time, and branch type for tuning and debug purposes with respect to optimizations without requiring updated profiling and debug tools for each CPU generation
- **PT Software Instrumentation (PTWRITE):** Provides a low overhead means for software to instrument processor tracing and provides flexible, enriched trace data for debug and profiling purposes
- **PERF_METRICS Extension:** Extends the interface to cover Level2 of the Top-down Microarchitecture Analysis Method (TMA) to deliver higher accuracy characterization (especially for non-steady state workloads) at lower profiling overheads

- **Chassis PerfMon Framework:** Uncore performance telemetry feature providing hardware feedback on workload activity and performance for advanced system and platform performance monitoring and analysis for management and optimization
- **PEBS Enhancements:** Precise Distribution Processor Event-Based Sampling (PEBS), Load Latency Extensions, and New Store Latency
  - Facility Software focused per core and per logical processor information to enable PEBS.

Below is a list of available tools built on Intel PAT.

- Intel® VTune™ Profiler
- Intel® Graphics Performance Analyzers
- Processor Counter Monitor (PCM)
- Intel® Power Gadget
- PowerTop

For further details, visit https://www.intel.com/content/www/us/en/developer/topic-technology/platform-analysistechnology/overview.html.

3.3.2 Intel® Platform Monitoring Technology (Intel® PMT)

Platform monitoring provides performance, power, and resource utilization data.

- **Intel® Trace Hub (Intel® TH) –** Debug and optimize code by exposing accurate and detailed traces to quickly identify where problems are occurring. Intel® TH leverages a consistent framework to aggregate and output the system's trace data to system memory and industry-standard trace ports.
- **Intel® Processor Trace (Intel® PT) –** Logs information about a program's performance without significantly slowing the system. Intel® PT obtains precise software behavior information including timing and instruction pointer information.
- **Architectural Event Trace (AET) –** Delivers a trace of events that occur while running software, which could include modification of the processor state, reaction of the processor to external events, or interaction between the processor and external devices. Contact your Intel representative to learn how you can get access to AET technology.
- **Debugging –** Enables run control, JTAG connectivity, and Direct Connect Interface (DCI) for closed chassis access.

3.3.3 Memory Failure Prediction

In a large server cluster, memory errors, while seemingly benign at a local level, can have a detrimental impact on overall network performance, reducing reliability with the potential to result in system failures and/or outages. Memory failure prediction identifies potential failures over time through multi-dimensional models and algorithms to detect memory errors and assign health scores for the purposes of analysis and response.

This allows customers to migrate critical workloads, identify DIMMs for replacement, and predict the offlining of memory pages.

3.4 Service Assurance

Resource constraints and locking of shared memory spaces can have a detrimental effect on performance for critical applications. Providing greater visibility and control over shared resources (LLC, memory BW, interconnect BW, I/O devices) helps to ensure that applications can maximize performance. CPU memory and caching controls deliver consistent and deterministic performance via Quality of Service (QoS) enforcement. For further details on service assurance, see the below details:

- https://networkbuilders.intel.com/docs/nfv-platform-service-assurance-intel-infrastructure-managementtechnologies.pdf
- https://networkbuilders.intel.com/solutionslibrary/enhanced-service-assurance-intel-platform-with-opnfv-barometerproject

3.4.1 Intel® Resource Director Technology (Intel® RDT)

Intel® Resource Director Technology (Intel® RDT) is a key feature set to optimize application performance and enhance shared resource monitoring along with the capabilities of service assurance.

Constituent features of Intel RDT include:

- Cache Monitoring Technology (CMT) and Cache Allocation Technology (CAT), which provide the hardware framework to, for instance, monitor and control the use of shared Last Level Cache (LLC).
- Memory Bandwidth Monitoring (MBM) and Memory Bandwidth Allocation (MBA) which provide the framework to monitor and control memory bandwidth.

3.5 Root Cause Analysis

To ensure that downtime is minimized, it is important to quickly and accurately diagnose and triage system failures through the capabilities of improved, accurate crash log mechanisms.

3.5.1 Intel® At-Scale Debug (Intel® ASD)

The Intel® At-Scale Debug (Intel® ASD) feature allows for the use of any host system to run a debug tool stack whilst connecting to a target system across the network. Intel® ASD is designed to enable customer self-sufficiency at scale in order to perform critical data collection, minimize the need for the traditional use of Intel In Target Probe (ITP), which may require physically accessing systems, and allows greater flexibility for instrumentation, reproduction, and environmental conditions surrounding the debug process. It allows customers to reduce debug complexity at the manufacturing, qualification, and production phases for hardware deployments.

As outlined in the block diagram below, as a minimum requirement, the target system must have a BMC with physical connectivity to the JTAG* pins. These are the high-level steps and ingredients that are needed to enable the Intel ASD BMCJTAG* solution.

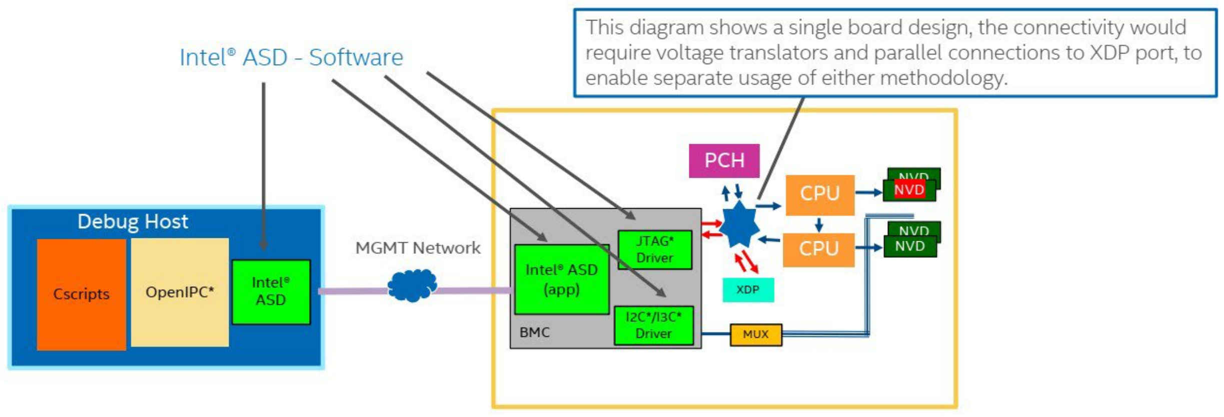

Intel® ASD - Software
This diagram shows a single board design, the connectivity would require voltage translators and parallel connections to XDP port, to enable separate usage of either methodology.
Debug Host
Cscripts
OpenIPC*
Intel® ASD
MGMT Network
Intel® ASD (app)
JTAG* Driver
I2C*/I3C* Driver
BMC
PCH
CPU
CPU
NVD
NVD
NVD
NVD
XDP
MUX

Figure 6. Intel® At-Scale Debug Interface

3.5.2 Crash Dump

Intel's System Crash Dump is an OOB method for the BMC to collect data from the CPU immediately upon system failure. Upon the occurrence of a crash condition, the BMC can autonomously trigger a crash dump on-demand, through CATERR, ERR[2:0] and SMI timeout assertions, reducing the overhead of having to wait multiple weeks to detect and reliably collect the data. Once the crash data is collected and organized into a standardized JSON format, which is stored by the BMC until it can alert the operator and upload the data for triage prior to reboot. In addition, the functionality can be enhanced to allow access to Cscripts over IPMI for more targeted data extraction by the administrator.

3.5.3 Crash Log Enhancements

Intel® Crash Log Technology is a mechanism for the aggregation of debug level data into a single location, providing access to the data through multiple methods. Some of the potential benefits and applications of Intel® Crash Log Technology include:

- Field use cases for critical issue resolution in order to reduce the time and cost expenditures required for reproduction, debugging, and triaging
- The ability to collect and aggregate system failure information for triage, first-level debug, and trend analysis purposes into a single location
- The ability to access failure information via multiple methods, including Intel Architecture FW (IAFW), the Operating System (OS), and OOB access via Joint Test Action Group (JTAG) ports or eSPI
- Crash Log vs. Crash Dump:

  o Crash log is an extension of crash dump introduced in wave1/2 of OOBMSM. Unlike crash dump (which is still supported), crash log is the automated process of dumping state in the event of an IERR or an MCA.

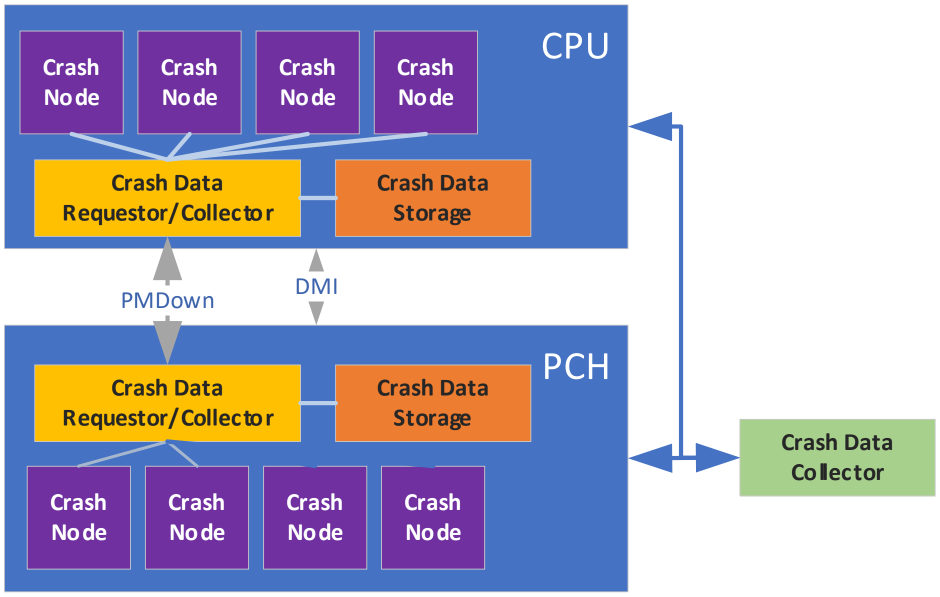


Figure 7. Crash Log Hardware and Software Components

4 Deployment

The diagram below gives an overview of the numerous technologies per platform to enable a robust platform management infrastructure. Further details on the physical implementation of these interfaces are outlined in the Eagle Steam Platform Design Guide (Document ID 610826).

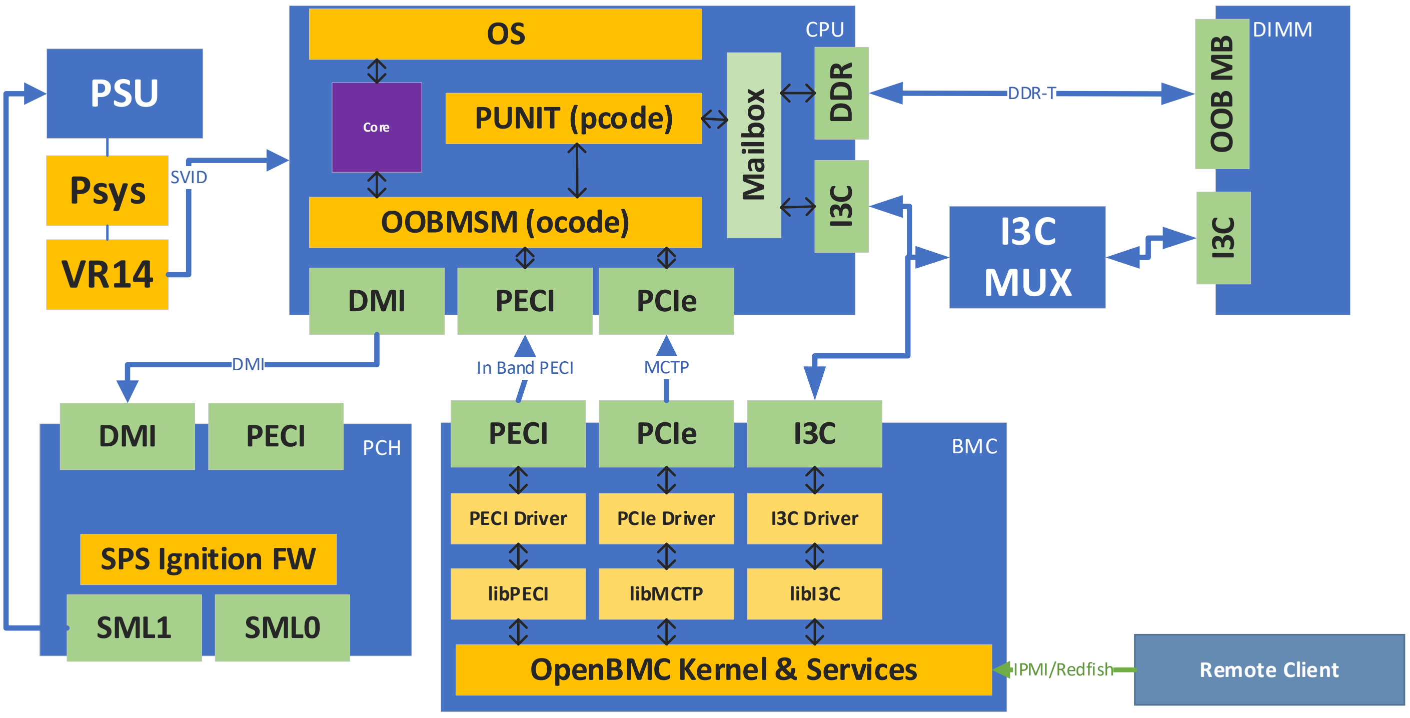


Figure 8. RAS Architecture Overview

Firmware is a key element to a RAS architecture and helps to enable access to services. The 4th Gen Intel Xeon Scalable processor BIOS implements various interactions with the RAS architecture including the following:

- Chipset Ingredients

        - SPS/Ignition Firmware
            - Located in the PCH, this provides the initialization of the CPU after reset, but also incorporates monitoring of peripheral devices during runtime (Power, thermals) over the SMBus/SMLink interface
        - PUNIT/Pcode
            - The PUNIT/Pcode provides the implementation of power management on the CPU, closely interacting with the BIOS, ME, and OOB management portions of the system
        - Out of Band Management Services Module (OOBMSM)
            - Provides access to the Intel capabilities described in the PCIE vendor specific extended capability registers (both VSEC and DVSEC). These capabilities include features like Intel® Platform Monitoring Technology (Intel® PMT) as well as others that are not supported by the intel_pmt driver. i.e. PSys
            - The PSys hardware requirement is new to the Pulse Width Modulation VR14 spec and is used to monitor and report digitized total 12V DC system power and provides more responsive power delivery. The VR14 spec details the requirements for Psys Sense Circuits. It is required for Psys hardware to be connected between the wall/PSU and Socket 0 Pcode over SVID bus #0.

- Platform Ingredients
    - PECI Interface
        - A proprietary Intel management bus that communicates data on platform level indicators and thresholds to the OpenBMC
    - I3C
        - A next generation management interface, incorporating the key capabilities of I2C and SPI into an advanced, consolidated specification, with higher performance and lower power requirements
        - Refer to https://members.mipi.org/wg/All-Members/document/84923 for further details
    - OpenBMC implementation
        - Open source implementation of the Baseboard Management Controller FW with defined API drivers and interfaces (PCIe/PECI/I3C etc.) to provide for portable development across platforms and architectures
        - 4th Gen Intel Xeon Scalable processors use the ASPEED2600 chipset for implementing OpenBMC
        - Refer to https://github.com/openbmc/openbmc for further details
    - Redfish
        - DMTF framework to provide a RESTFUL interface for the management of servers, networking, and storage in a converged architecture. Redfish is widely adopted across the industry and provides a secure and simplified integration along with improved performance and remote management capabilities
        - Refer to https://www.dmtf.org/standards/redfish and https://pythonhosted.org/python-redfish/installation.html for further details

4.1 Out of Band Management

OOB BIOS configuration provides the user with the ability to view and modify the BIOS setup configuration parameters remotely via the BMC LAN channel at any host state, for example when the server in booted to the OS booted and in a running state. Modifications to the parameters take place upon the next system reboot. The Insight Toolkit (ITK) and Syscfg tools are available to update BIOS settings with the following known limitations:

- ITK can be used to restore the BIOS default settings but does not update the current settings.
- Syscfg tools run with the shell and OS environment, and although updates to the BIOS settings are applied immediately, it still requires a mandatory reboot. Administrators must be aware of the BIOS setup variables, as only a small subset are supported via Syscfg. This limitation can be overcome by reading/writing the current BIOS settings from/to the BMC over the LAN interface. Essentially, the idea is to communicate the current BIOS settings to the BMC, process the request from the BMC regarding BIOS setting updates, and communicate back to the BMC the updated BIOS settings. Before performing remote configuration, the utility or the web client needs to establish a secure connection with the BMC by providing the BMC administrator username and password. The following diagram shows an overview of the implementation of the OOB BIOS configuration feature.

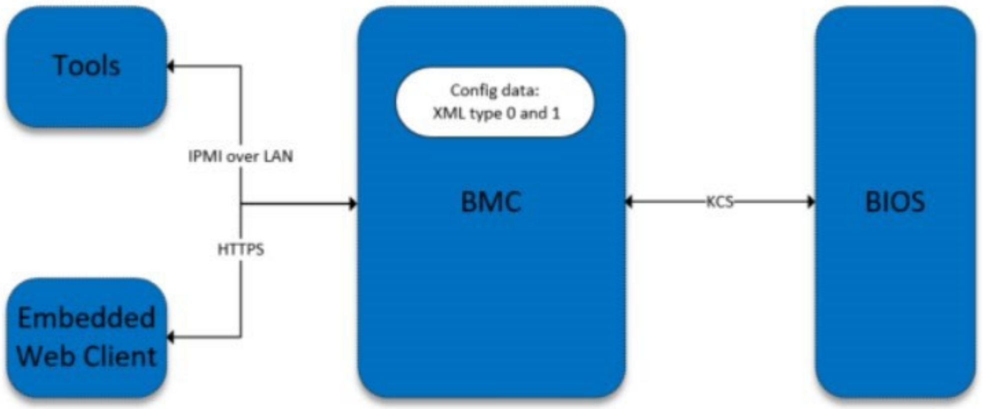


Figure 9. Overview for Out of Band Management

4.2 OOB Management Services Module

The System Management Service Module (OOBMSM) in the 4th Gen Intel Xeon Scalable processors provides out of band access for various management, telemetry, and crash log services. It supports the Intel® System Management Specification, Rev 1.0, with a set of services available via the PECI electrical interface or MCTP over PCIe. It is designed to support processor and memory thermal management, platform manageability and processer interface tuning and diagnostics.

It supports the below functions:

- Crash log
  - The crash log is a mechanism for aggregation of debug level data.
  - The data can be accessed through multiple methods including in-band via the OS as well as OOB via PECI, JTAG, or Espi
- MCTP
  - OOBMSM supports MCTP bridge and MCTP endpoint mode
- PECI
  - PECI interface outlines commands to the CPU and PCH through a dedicated Hardware interface.
- PECI over MCTP support
  - Provides Telemetry via PECI over MCTP support
- OOBMSM is built as a modular architecture with three key interfaces

  a. PCIe* Compliant Primary Interface (High Bandwidth)

  b. Two Sideband Interfaces (GPSB and PMSB)

     i. uController : Tensilica*

     ii. SRAM Module : SRAM size is 96 KB in Eagle Stream. (Data Memory)

  c. Instruction memory ~

     i. OOBMSM includes the Local Translation Module (LTM) as a On-Die “PhoneBook”

     ii. OOB-MSM FW is part of CPU Microcode

     iii. OOB-MSM FW size ~200 KB

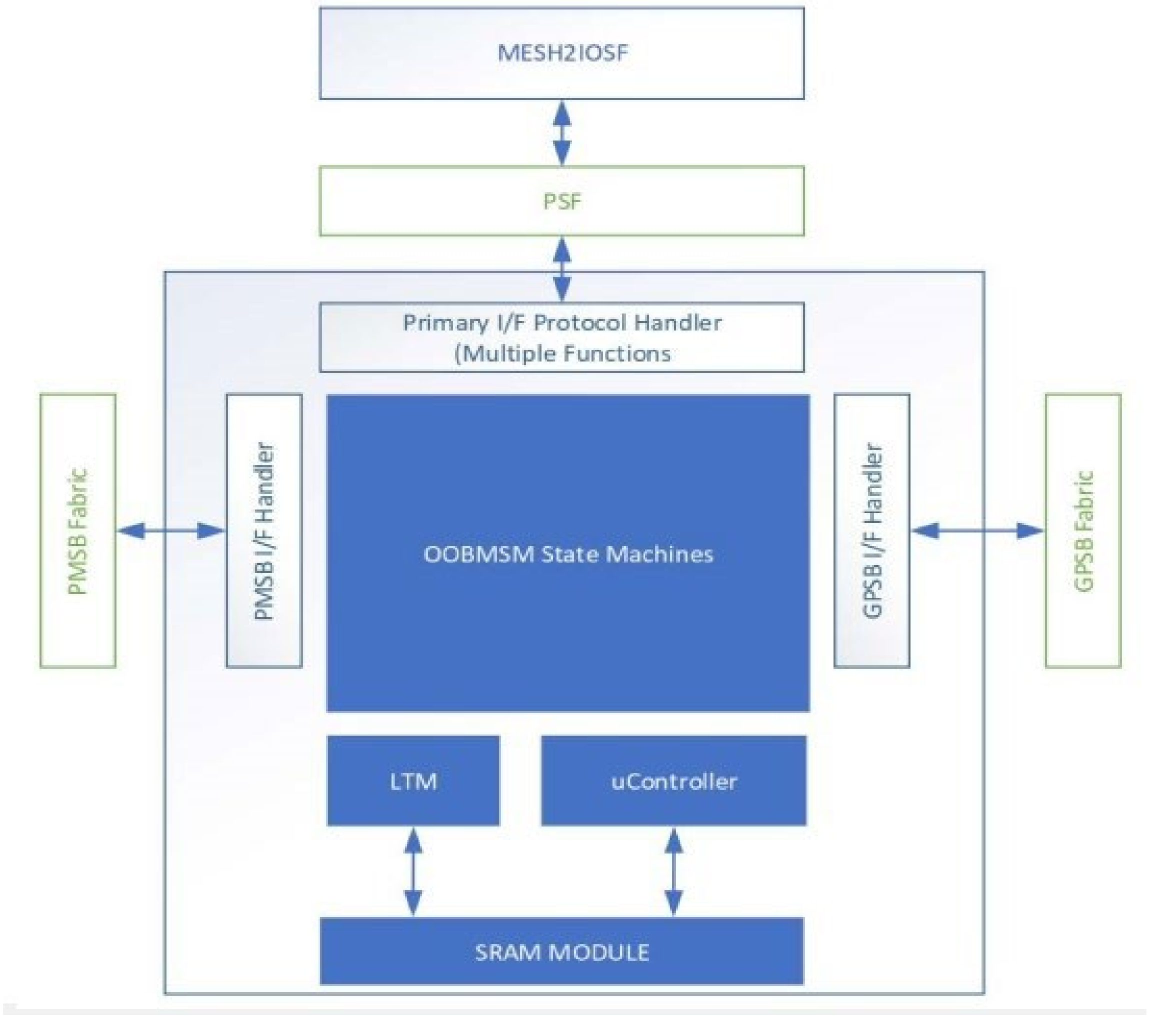


Figure 10. Out of Band Management State Machines

4.3 MCTP

The MCTP is a media-independent protocol for intercommunication among intelligent devices within the platform management subsystem of a managed computer system. This protocol is independent of the underlying physical bus properties, as well as the "data-link" layer messaging used on the bus. The physical and data-link layer methods for MCTP communication across a given medium are defined by companion “transport binding” specifications, such as MCTP over PCIe* Vendor Defined Messaging and MCTP over SMBus/I2C*. This approach enables future transport bindings to be defined to support additional buses such as USB, Reduced Media Independent Interface (RMII), and others, without affecting the base MCTP specification. MCTP has been designed to carry multiple types of manageability-related traffic across a common medium. The base MCTP specifications define message types for supporting the initialization and configuration of MCTP itself, and to support vendor-specific messages over MCTP. Other message types, including message types to support a Platform Level Data Model (PLDM), network controller sideband communications, and so on, are planned to be specified in the future by the DMTF PMCI work group.

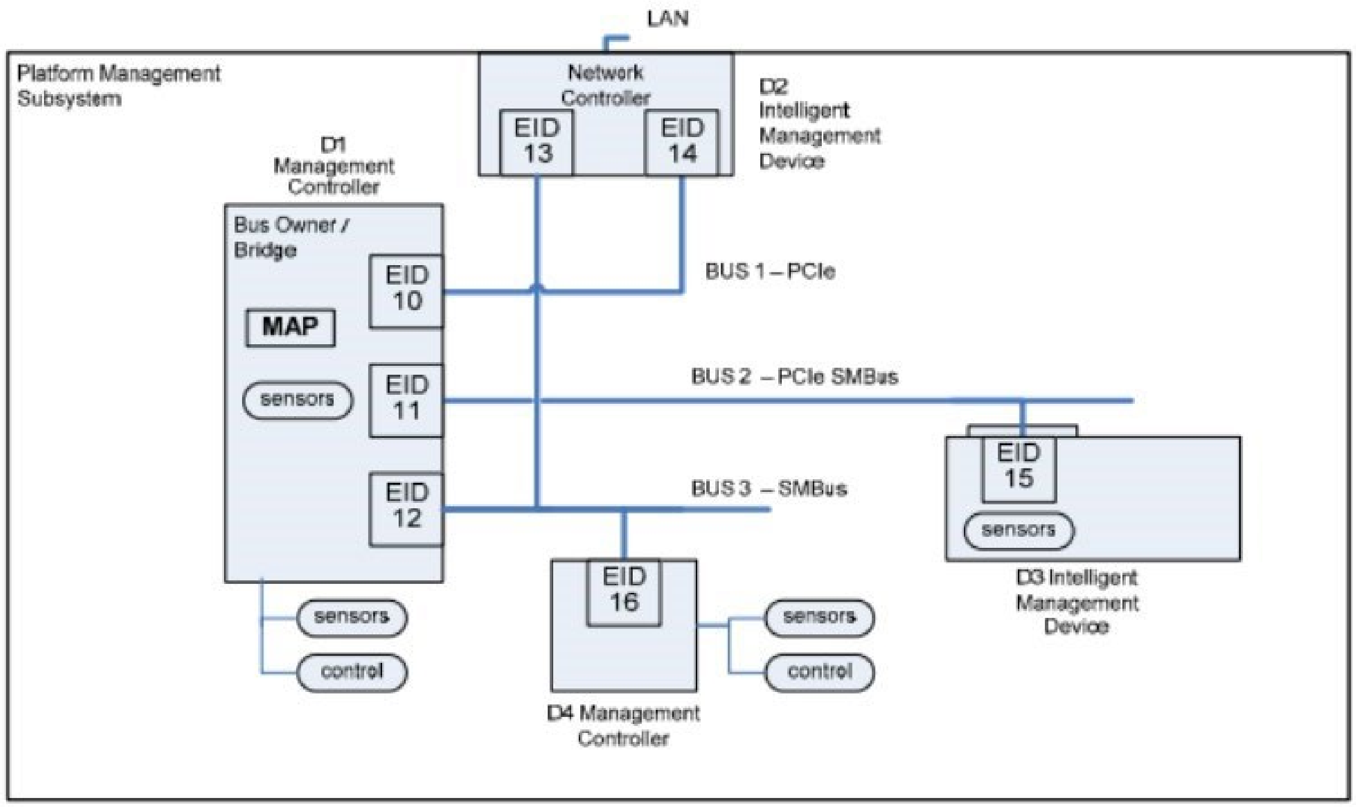


Figure 11. MCTP Platform Management Subsystem

MCTP endpoints and Endpoint IDs (EIDs) are the functions within devices that terminate the communication protocol of MCTP and handle MCTP control commands. MCTP uses a logical address called the EID for addressing and routing MCTP packets to and from endpoints. MCTP, a bus, is defined as an interconnect between platform components that share a common physical layer address space. A bus may be made up of multiple segments. A bus segment is a portion of a bus that is electrically separated from other segments that form a bus, but still shares a common physical address space with other segments.

4.4 Redfish

Redfish is a standard that uses RESTful interface semantics to access a schema based data model to conduct management operations. It is applicable to a wide range of devices and is suited for the infrastructure requirements for fixed and mobile networks.

Redfish is intended to supersede the IPMI interface. For further details, refer to the following specification: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.15.0.pdf

4.5 Telemetry

Telemetry reporting software, available from Intel, translates platform level metrics into networking as well as operational metrics and provides insights into platform reliability, utilization, congestion, along with potential configuration issues. These insights can be used to notify network operators and provide key inputs for potential remediation actions by automated control systems as part of an observability solution within closed loop systems.

The telemetry reports provide insights that distill IA metrics into networking and operational metrics and allow the integration of insights into automated control systems. The distilled networking and operational metrics can be grouped into four categories:

1. Platform health insights
2. Utilization insights
3. Congestion/overload insights
4. Platform configuration checks

The distilled metrics and insights can be consumed by multiple management, orchestration, and control systems, including, among others, Software Defined Networking (SDN) controllers, VIMs including Kubernetes and OpenStack, Root Cause Analysis (RCA) systems, Virtual Network Functions Managers (VNFMs), Network Functions Virtualization Orchestraters (NFVOs), capacity planners, online and offline analytics systems.

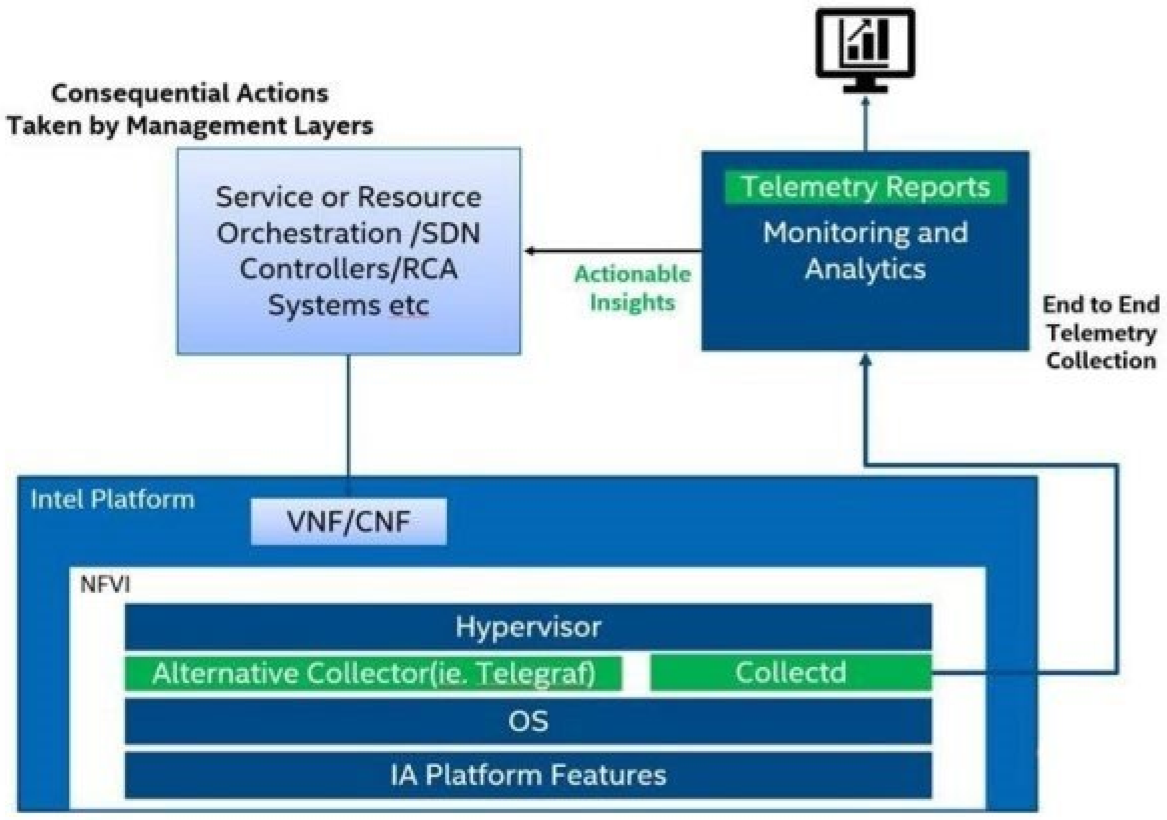

Consequential Actions
Taken by Management Layers
Service or Resource Orchestration /SDN Controllers/RCA Systems etc
Actionable Insights
Telemetry Reports
Monitoring and Analytics
End to End Telemetry Collection
Intel Platform
VNF/CNF
NFVI
Hypervisor
Alternative Collector(ie. Telegraf)
Collectd
OS
IA Platform Features

Figure 12. Telemetry and Insights Infrastructure

5 Summary

This document provides an overview of the technologies to develop a robust and reliable platform and ensure a platform with a high degree of debuggability. In addition to the incremental technologies in the 4th Gen Intel Xeon Scalable processors, adjacent technologies can supplement the set of tools to extract maximum performance from the system. These technologies can work in conjunction with the telemetry and orchestration frameworks to allow network operators to report and act upon potential errors in order to achieve target network KPIs. It is important when defining and designing next generation platforms to ensure that the system is fully optimized and configured to take advantage of these features.

Appendix A Seamless Firmware Update

Blackout periods for a firmware upgrade can cost in the order of >10 minutes to allow the active firmware and microcode components to be upgraded, as well as a cold reset of the system to allow the new images to activate. In addition to the downtime and loss of service, this is also costly from losing context for memory and migration to an alternate machine.

SFU allows the user to implement a firmware upgrade in around 10 seconds. This will allow the virtual machines (VMs) to continue to run during the upgrade on the platform and only a brief pause is needed during the update. VM memory and state is maintained through the update.

In order to enable SFU capabilities, the following elements are required to be enabled on the platform:

- System is required to support Out of Band update on an Intel® Platform Firmware Resilience (Intel® PFR) supported platform. This option is available as part of the BKC releases for the platforms. Similarly, Intel® Server Platform Services (Intel® SPS) is required to be supported.
- Platform Runtime Mechanism is required. Server Management Mode (SMM) is a special-purpose operating mode provided for handling system-wide functions like power management, system hardware control, or proprietary OEM-

designed code. It is invoked through the System Management Interrupt (SMI). However, with SMM users can get unfettered access to system resources, stall processors, overhead of switching across processor modes, and save or restore context and the OS has zero visibility and no control over the SMI software handlers. The Platform Runtime Mechanism (PRM) methodology is designed to invoke class one software handlers that do not need Server Management Mode (SMM) privileges to invoke native code through the Advanced Configuration and Power Interface (ACPI) context of runtime events.

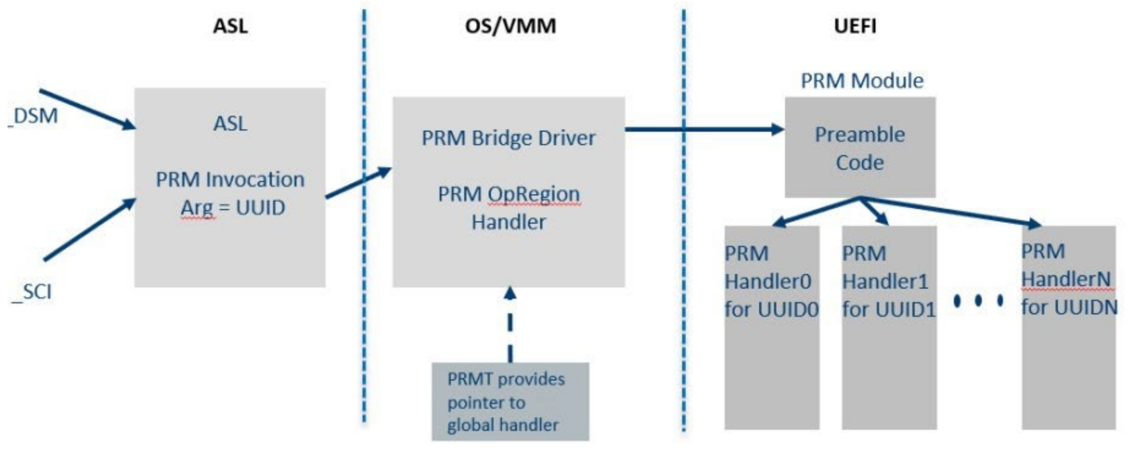


Figure 13. Seamless Firmware Update Runtime Requirements

Details on implementing PRM are available on the UEFI website - https://uefi.org/sites/default/files/resources/Platform%20Runtime%20Mechanism%20-%20with%20legal%20notice.pdf

SFU requires OpenBMC enablement.

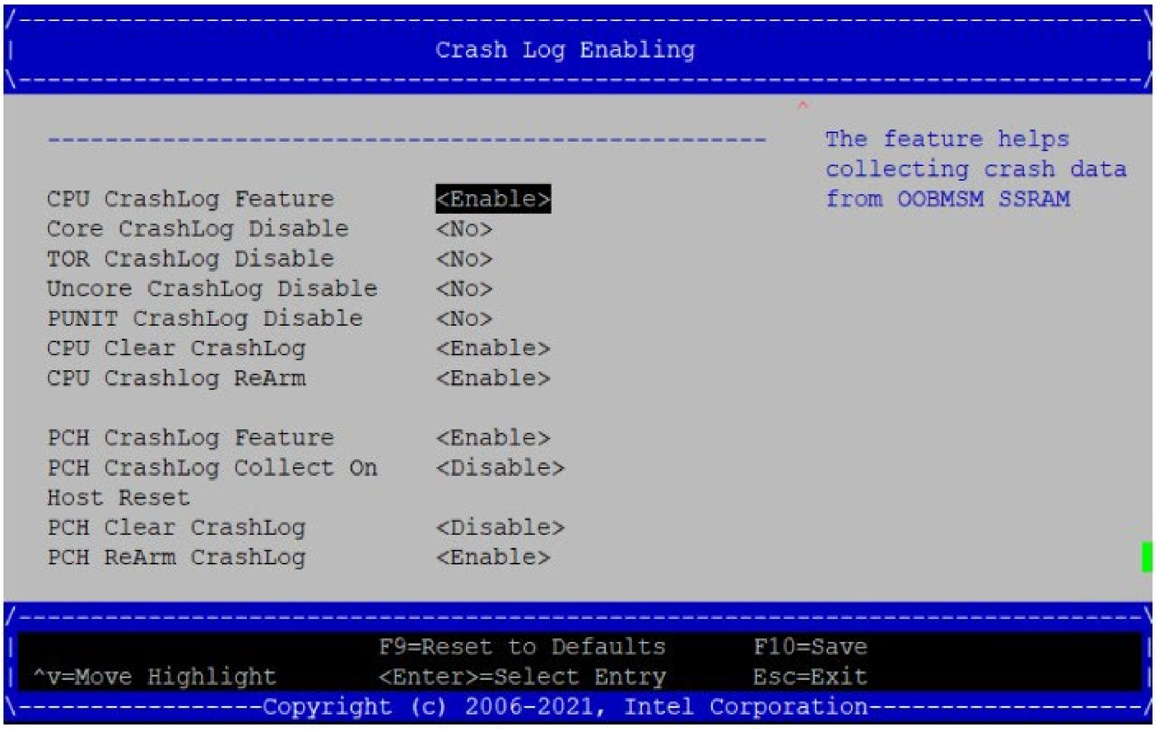
Appendix B BIOS Configuration

- Intel® PFR technology is designed C-Scripts Settings. The following is a quick reference of the key BIOS knobs for CScripts:
  - EDKII > Socket Configuration > Processor Configuration > Lock Chipset = Disable
  - EDKII > Socket Configuration > IIO Configuration > IIO DFX Configuration >
    - EV DFX Features = Enable
    - Disable BIOS Done = [X]
- BIOS Settings
  - EDKII > Platform Configuration > System Event Log > WHEA Settings >
    - WHEA Support = Enable
    - WHEA Log Memory Error = Enable
    - WHEA Log Processor Error = Enable
    - WHEA Log PCI Error = Enable
  - EDKII > Platform Configuration > System Event Log >
    - System Errors = Enable
    - System Memory Poison = Enable
  - EDKII > Platform Configuration > System Event Log > Error Injection Settings >
    - MCA Bank Error Injection Support = Enable
    - WHEA Error Injection Support = Enable
    - WHEA Error Injection 5.0 Extension = Enable
    - WHEA PCIe Error Injection Support = Enable
    - WHEA PCIe Error Action Table = Enable
  - EDKII > Platform Configuration > System Event Log > IIO Error Enabling >

        - IIO/PCH Global Error Support = Enable
        - IIO MCA Support = Disable
        - IIO eDPC Support = Disabled
        - PCIE Corrected Error Threshold Counter = Disable
    - EDKII > Platform Configuration > System Event Log > EMCA Settings > LMCE Support = Enable
    - EDKII > Platform Configuration > System Event Log > Memory Error Enabling > Memory Corrected Error = Enable
    - EDKII > Platform Configuration > System Event Log > EMCA Settings >
        - EMCA Logging Support = Enable
        - EMCA CMCI-SMI Morphing = EMCA gen2 CSMI
        - EMCA CMCI-SMI Threshold = 0
        - EMCA MCA-SMI Enable = EMCA gen 2 - MSMI
    - EDKII > Platform Configuration > System Event Log > IIO Error Enabling > OS Native AER Support = Disable
- Crash Log
    - BIOS Setup
        - Platform Configuration > System Event Log > Crash Log Enabling > PCH Crash Log Feature = “Enabled”
        - Platform Configuration > System Event Log > Crash Log Enabling > PCH Clear Crash Log= “Disabled”
        - Platform Configuration > System Event Log > Crash Log Enabling > PCH Crash Log Collect on All1 Resets = “Enabled”
    - OS Setup
        - To trigger on host resets, initiate a warm reset to the system. Some examples:
            - Push the reset button of the SUT
            - IPC resettarget() command
                - BMC shell ipmitool chassis power reset
                - Return to the command prompt to issue intel_crashlog extract and intel_crashlog triage commands
                    - The output of triage should show “trigger on all resets” as the cause
            - From the Debug and Test System (DTS) OS command prompt or shell, type intel_crashlog –h
    - Samples
        - BIOS Extraction
            1. The Intel® System Debugger release includes a crashlog-extract.efi utility in its installation directory. Copy it into a USB drive that can be accessed from the SUT BIOS/UEFI shell.
            2. Use the SUT configuration to facilitate injection of a Host Partition Reset (HPR) Timeout (TO).
            3. Perform an AC cycle (G3) on SUT to clean any leftover Crash Log record from PCH SRAM.
            4. After SUT boots, from the Debug Host, issue intel_crashlog extract command in verbose mode (-vv).
                a. The output should show “Crash Log region in PCH PMC SSRAM is not populated” and “no valid Crash Log can be extracted” messages. This ensures that no leftover records are present in PCH.
            5. Enter intel_crashlog --product EBG -vv trigger hprto on the Debug Host system. This will inject an HPR TO error via PCH JTAG* (IPC2 mailbox).
            6. The SUT may reset multiple times, just as in the previous exercise. BIOS rearms PCH Crash Log upon rebooting from Global Reset.
            7. Boot the SUT to BIOS/UEFI shell and navigate to the USB drive directory on the SUT.
            8. Find and execute the crashlog-extract.efi utility. This will extract Crash Log records from the ACPI Boot Error Record Table (BERT) created by BIOS at boot time.
            9. The utility should find the PCH Crash Log record created by the Host Partition Reset (HPR) Timeout (TO) error (Step 5) and create a .crashlog file in the root directory of the USB drive.
            10. Run an intel_crashlog triage on the extracted Crash Log file from the Debug Host system (not SUT). The output of the triage should show Host Reset Timeout as the cause.

- Back to Back Testing
  1. Disable in BIOS EDKII* Menu: Platform Configuration > System Event Log > Crash Log Enabling > PCH Crash Log Collect on All Resets = “Disabled”. Restart/Reset to apply BIOS changes.
  2. On a command prompt or shell, trigger the PCH Crash Log flow manually via IPC2 (PCH JTAG*) as described in Slide 10 (Manual/On-Demand Trigger).
  3. Next, extract and analyze the PCH Crash Log data using the intel_crashlog extract and intel_crashlog triage commands as shown in Slide 10. The output of the triage should show “Manual Trigger” as the cause.
  4. Alternatively, use the intel_crashlog summary command to verify the trigger reason.
  5. Enable in BIOS EDKII* Menu: Platform Configuration > System Event Log > Crash Log Enabling > PCH Crash Log Collect on All1 Resets = Enabled”. Restart/Reset to apply BIOS changes.
  6. Execute the Trigger Rearm command using the IPC2 (PCH JTAG*) interface: intel_crashlog -vv --product EBG trigger –rearm.
  7. Execute a warm reset using one of the methods (Trigger on Host Resets). This will trigger the PCH crash log flow on the warm or host reset.
  8. Extract and analyze the data using the intel_crashlog extract and intel_crashlog triage commands shown in Slide 10 (Manual/On-Demand Trigger). The output of the triage should show “Trigger on All Resets” as the cause.

- Integration and Deployment
  - Hardware Requirements
  - BIOS Configuration
  - Kernel Enablement
  - Remote Access Enablement
  - Debug Enablement
- Crash Log
  - Crash Log BIOS Enablement



```
/----------------------------------------------------------------------------\
|                           Crash Log Enabling                               |
\----------------------------------------------------------------------------/
                                                           ^
  --------------------------------------------------   The feature helps
                                                       collecting crash data
  CPU CrashLog Feature         <Enable>                from OOBMSM SSRAM
  Core CrashLog Disable        <No>
  TOR CrashLog Disable         <No>
  Uncore CrashLog Disable      <No>
  PUNIT CrashLog Disable       <No>
  CPU Clear CrashLog           <Enable>
  CPU Crashlog ReArm           <Enable>

  PCH CrashLog Feature         <Enable>
  PCH CrashLog Collect On      <Disable>
  Host Reset
  PCH Clear CrashLog           <Disable>
  PCH ReArm CrashLog           <Enable>

/----------------------------------------------------------------------------\
|                         F9=Reset to Defaults      F10=Save                 |
| ^v=Move Highlight       <Enter>=Select Entry      Esc=Exit                 |
\----------------Copyright (c) 2006-2021, Intel Corporation------------------/
```

Appendix C Platform Firmware Resiliency

Intel® PFR technology is designed to support the National Institute of Standards and Technology* (NIST*) requirements of the SP 800-193 document as well as to provide security assurance for Intel® Server Platforms against known and unknown vulnerabilities based on extensive research done both internally and externally. The goal of Intel® PFR technology is to provide resiliency by protecting platform assets, detecting corrupted firmware and malicious or erroneous behavior and recovering the platform to a known good state. 11.1 NIST SP 800-193* Requirements NIST* lists the following three guiding principles in order to support resiliency of platforms against potentially destructive attacks: • Protection: Mechanisms for ensuring that Intel® PFR code and critical data remain in a state of integrity and are protected from corruption, such as the process for ensuring the authenticity and integrity of firmware updates. • Detection: Mechanisms for detecting when Intel® PFR code and critical data have been corrupted. • Recovery: Mechanisms for restoring Intel® PFR code and critical data to a state of integrity if any such firmware code or critical data are detected to have been corrupted, or when forced to recover through an authorized mechanism. Recovery is limited to the ability to recover firmware code and critical data. In addition, NIST SP 800-193* document provides guidance on meeting those requirements via three main functions of a Platform Root of Trust (RTU): • RTU which is responsible for authenticating firmware updates and critical data changes to support platform protection capabilities, this includes signature verification of firmware updates as well as rollback protections during update. • Root of Trust for Detection (RTD) which is responsible for firmware and critical data corruption detection capabilities. • Root of Trust for Recovery (RTRec) which is responsible for recovery of firmware and critical data when corruption is detected or when instructed by an administrator. This document presents a platform feature architecture created to support the NIST* requirements and create a resilient platform able to self-recover upon detection of attack or firmware corruption. 11.2 Intel Security Requirements In addition to NIST* requirements, Intel® PFR technology lists the following security requirements in order to provide highest level of resiliency for Intel® Server Platform: 1. Must provide automatic, local recovery from known and unknown software attacks against critical-to-boot platform firmware and configuration data. 2. Must provide protection against software attacks that aim to inflict physical damage to critical-to-boot platform level components.

https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.15.0.pdf

Appendix D 4th Gen Intel Xeon Scalable Processor Features

D.1 128B ECC Mode Partial Poisoning

In the 128b ECC mode, the silicon continues to support the poison mode of the operation with the following flow characteristics:

- Poisoned data is written to the DDR memory with a unique poison pattern
- The “poisoned pattern” never overlaps with the bit patterns of a “good data”

- Poisoned data is read from memory
- The poisoned data pattern is treated as an uncorrectable error.
- The logic will log the Uncorrected No Action Required (UCNA) error.
- The data is sent to the requestor with a poison bit attached.

D.1.1 Patrol Scrub Handling of Errors

Uncorrectable Errors (UCE) are logged and signaled as a UCNA, and the poison pattern is generated and stored in the DRAM. Patrol scrubbing detects a poison pattern, handles it as an UCE, and logs and signals the error as a UCNA.

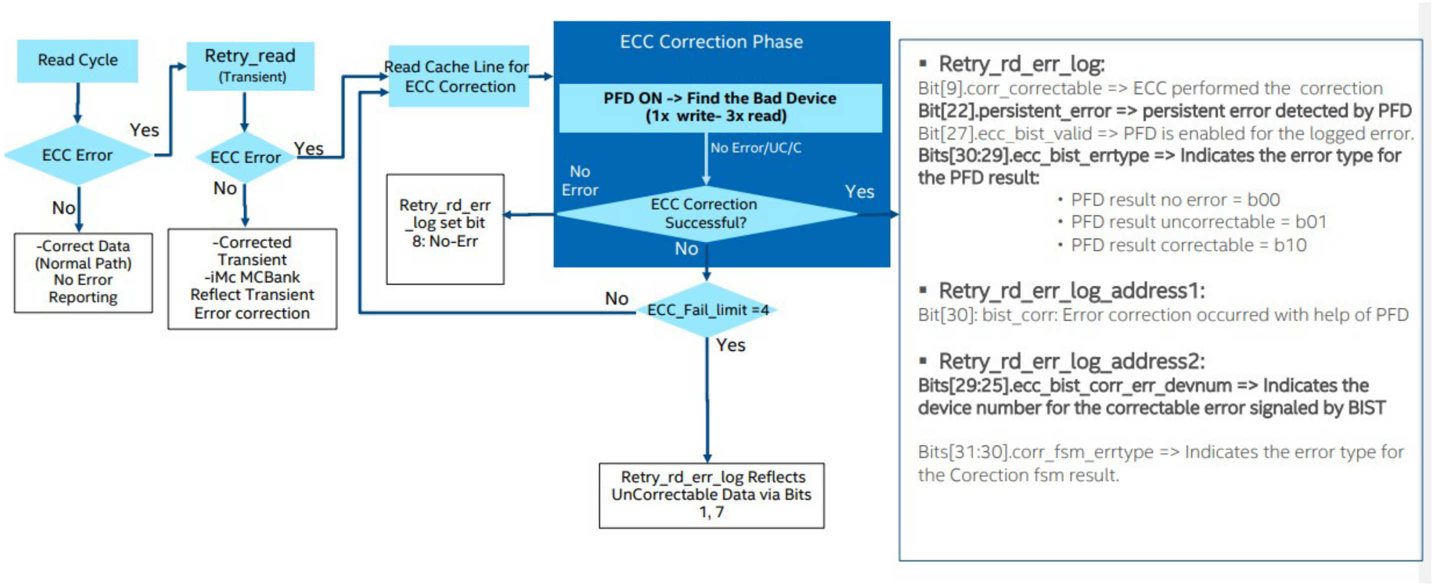
- BIOS Handler
    - With the partial poisoning, the iMC will log the Uncorrectable Error (UCE) on every instance of the detection.
    - The error handler can repeatedly find the signature of the same error source in the log:
        - The error due to the DRAM fault will have the same signature as a poisoned line placed in the DRAM.
        - The patrol scrubber will repeatedly detect the same line over its programmed cadence and will log and signal the error on every instance of the detection.

D.1.2 DDR5 On-Die Error Check and Scrub

iMC support extracting the DDR5 ECS registers. The DDR5 rank crossing the threshold will be reflected in the ECS and updates once every 24 hours.

- BIOS flow for the ECS:
    - Program all the ECS bank refresh intervals in rdimmtimingcntl.ECS_REFab_interval
    - The 10b field should be programmed to tECSint/tREF-offset. Value of 0 means that the ECS is disabled.
    - Enable the ECS automatic mode in the DRAM: MR14.OP[7]=0
    - Enable the Server Management Interrupt (SMI) signaling on patrol completion. Use the iMC patrol scrub 24-hour time counter to capture the ECS result.
- Runtime System Management Mode (SMM) handler:
    - For each socket, channel, sub-channel, rank, or sub-rank read the ECS error information from MR16 to MR20 using the Machine Check (MC) Memory Rank (MR) interface CSR mr_config. Save the valid data of each MR. For 3DS, read the MRs from each of the 3DS die. MR14.OP[3:0] provides the 3DS die selection.
    - MR read result is logged in the following CSRs: mr_read_result_dq12to0, mr_read_result_dq28to16, mr_read_result_dq44to32, mr_read_result_dq60to48, mr_read_result_dq68to64. This set of mr_read_result registers is per sub-channel.

D.1.3 Error Correction Flow


Read Cycle
ECC Error
Yes
No
-Correct Data (Normal Path) No Error Reporting
Retry_read (Transient)
ECC Error
Yes
No
-Corrected Transient -iMc MCBank Reflect Transient Error correction
Read Cache Line for ECC Correction
Retry_rd_err_log set bit 8: No-Err
ECC Correction Phase
PFD ON -> Find the Bad Device (1x write- 3x read)
No Error/UC/C
No Error
ECC Correction Successful?
Yes
No
No
ECC_Fail_limit =4
Yes
Retry_rd_err_log Reflects UnCorrectable Data via Bits 1, 7

- Retry_rd_err_log:

Bit[9].corr_correctable => ECC performed the correction
Bit[22].persistent_error => persistent error detected by PFD
Bit[27].ecc_bist_valid => PFD is enabled for the logged error.
Bits[30:29].ecc_bist_errtype => Indicates the error type for the PFD result:

  - PFD result no error = b00
  - PFD result uncorrectable = b01
  - PFD result correctable = b10

- Retry_rd_err_log_address1:

Bit[30]: bist_corr: Error correction occurred with help of PFD

- Retry_rd_err_log_address2:

Bits[29:25].ecc_bist_corr_err_devnum => Indicates the device number for the correctable error signaled by BIST

Bits[31:30].corr_fsm_errtype => Indicates the error type for the Corection fsm result.

Figure 14. Error Correction Flow

Appendix E Standard RAS Features

Category	Platform RAS Feature
IIO	PCI Express Link Retraining and Recovery
IIO	PCI Express Link CRC Error Check and Retry
IIO	PCI Express Corrupt Data Containment (Data Poisoning)
IIO	PCI Express ECRC
IIO	PCIe Enhanced Downstream Port Containment (eDPC)
IIO	PCI Express Card Hot-Plug (Add/Remove/Swap)
IIO	PCI Express Card Surprise Hot-Plug
IIO	Error Reporting via IOMCA
IIO, PCH	Integrated Error Handler (IEH)
PCH	PCH PCIe Advanced Error Reporting (AER)
Intel® UPI	Intel® UPI Link Level Retry
Intel® UPI	Intel® UPI Failing Lane Isolation
Intel® UPI	Intel® UPI Protocol Protection via CRC (16 bit)
CPU	Error Detection and Correction (coverage at socket level)
CPU	Corrupt Data Containment - Uncore
CPU	Corrupt Data Containment - Core
CPU	Advanced Error Detection and Correction (AEDC)
CPU	Time-out timer schemes
CPU	Error reporting (MCA, AER) - Core, Uncore, and IIO
CPU	Error reporting through MCA 2.0 (EMCA Gen2)

CPU	Processor BIST
CPU	MCA Bank Error Control
CPU	First Corrected Error Mode of Error Reporting
CPU	PCI Express Corrected Error Reporting
CPU	Thresholding for Corrected Errors (all uncore MCBank for CSMI)
CPU	CSR Error Log Cloaking (using DEVHIDE)
CPU	DCU/IFU Poison Enhancements
CPU	DCU Scrubbing
CPU	Core Disable for FRB
CPU	Socket Disable for FRB
CPU	Enhanced SMM (ESMM)
System	Failed DIMM Isolation
System	OOB access to Error Logs
System	Error Injection Capability
System	Predictive Failure Analysis
System	Pre-GOS1_IERR warning
System	Suppress Inbound Shutdown
System	Demoted Warm-Reset
System	MCA Recovery
Memory	Memory Single Device Data Correction (SDDC)
Memory	DDR Command/Address Parity Check and Retry
Memory	Memory Data Scrambling with Command and Address
Memory	Memory Demand and Patrol Scrubbing
Memory	DDR Memory Multi Rank Sparing
Memory	Memory Thermal Throttling
Memory	Memory Mirroring
Memory	Adaptive DDDC - Single Region (ADDDC - SR)
Memory	Post Package Repair (PPR)
Memory	Partial Cache Line Sparing (PCLS, HBM only)
Memory	Memory Disable/Map-Out for FRB
Memory	Memory SMBus Hang Recovery
CXL	CXL Link CRC Error Check and Retry
CXL	CXL Link Retraining and Recovery
CXL	CXL Corrupt Data Containment (Data Poisoning)
CXL	CXL ECRC
CXL	Error Reporting via IOMCA

Appendix F 4th Gen Intel Xeon Scalable Processor Advanced RAS Features

Category	Platform RAS Feature
Intel® UPI	Intel® UPI Dynamic Link Width Reduction
Intel® UPI	Intel® UPI Quiescence
Intel® UPI	Intel® UPI Hot-Plug
System	Viral Mode
System	Error Injection, including MCBank Spoofing
System	Error Reporting Through eMCA 2.0 with Ability to Write MSR
System	Static DMI/PCH Failover
System	Physical CPU + Memory + I/O Board Hot-Plug
System	OS CPU + Memory + IIO On-Lining (Capacity Change)
System	Electronically Isolated (Static/Hard) Partitioning
System	Dynamic Partitioning (Via Resource/Capacity Addition)
Memory	DDR 5 Address Range/Partial Memory Mirroring
Memory	x4 DRAM: ADDDC Multi-Region

0722/DN/WIPRO/PDF 728996-002US