# ARM64 Server RAS Solutions

Jonathan (Zhixiong) Zhang
Cavium Inc.

# Agenda

- Overview
- Solutions
- Building blocks
- Reflections

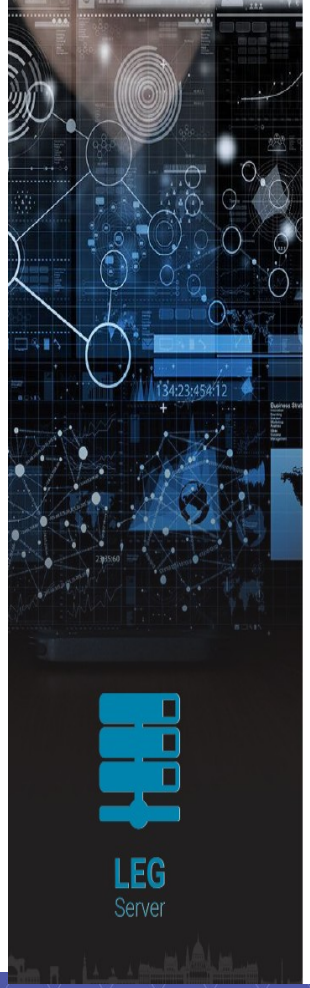
# Overview

Reliability, Availability, Serviceability

RAS is one of the most important aspects of ARM64 servers' success

# Progresses

- RAS on ARM64
- SFO17
- Fu Wei
- Focus on OS and ACPI

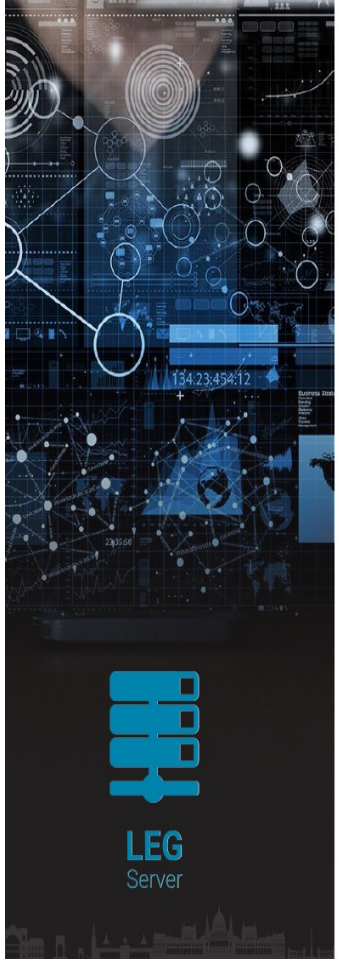



## AGENDA

- What is RAS?
  - Introduction: Definition, Importance, History on X86
  - Overview of RAS functions
- ARMv8 CPU requirements for RAS
  - CPU core, GICv2/GICv3
  - **RAS Extension(ARMv8.2)**
- SDEI(Software Delegated Exception Interface)
- APEI(ACPI Platform Error Interfaces)
  - BERT and CPER, HEST and GHESv2, EINJ/ERST
- SW components for RAS(in example)
  - All SW components
  - How it works: BERT, HEST
- Current status and Planning

# Progresses

- RAS on AARCH64
- SFO17
- Fu Wei, Supreeth Venkatesh
- Focus on prototype solution

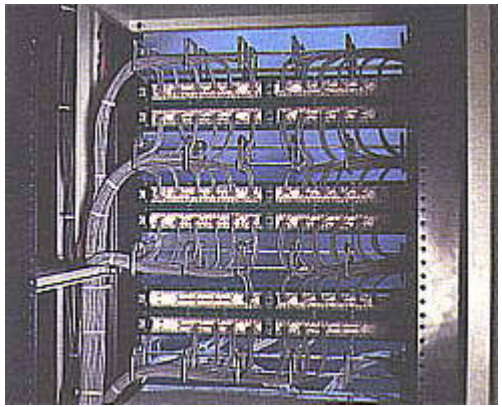



# AGENDA

1. Brief introduction of RAS
   - Definition, Importance, History
2. RAS on AArch64
   - Overview
     - Hardware support
       - RAS Extension
     - Software Architecture
       - ARM-Trusted-Firmware, UEFI, APEI tables
       - SDEI
   - Prototype Solution for Firmware First Error Handling
     - MM Secure Partition, Secure Partition Manager
     - Uncorrected error -- HEST & MM
     - Demo time
3. Status and Future Plans

# Discussion Focus

- RAS is a 3 dimensional problem!
- 2 Dimensions: from CPU, to board, to rack
- 1 dimension: from hardware, to firmware to software
- Product ready solutions



# Who needs what – end user, apps developer

- Accurate error reporting
- Right level error reporting
- Low cycle stealing
- Quarantined uncorrectable error

# Who needs what – data centre admin

- Accurate error reporting
- Reliable error reporting
- Uniformed error handling policy
- Actionable error reporting

# Who needs what – Platform Vendor

Validation through production software stack

- Example: Memory stability has to be validated through running all hardware threads, exercising all DIMMs through OS memory test tool.
- Example: PCIe tuning issue can be spotted through thousands of reboots.

# Who needs what – SOC Vendor

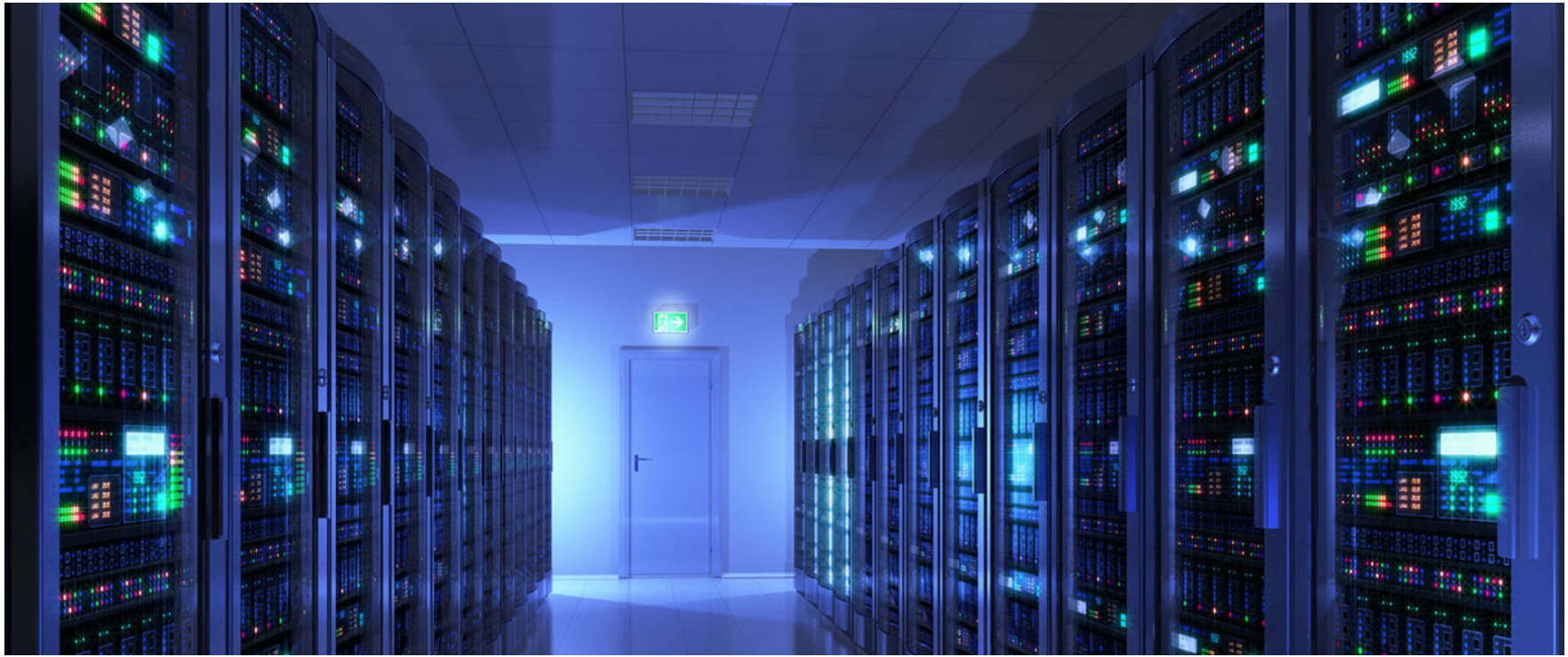
Validation through production software stack

- Validation/diag tool does not exercise the SOC enough.
- Subsystem design/programming issue needs to be unmasked earlier in the engineering cycle.

Cost-effective support process

- Expedited problem root cause.
- Reduced reliant on hardware debugger.

# Solutions



# Early boot time error notification

- Post code, to diagnose halted boot.
- SEL record, to diagnose successful boot with component failure

# Run time error notification

## Catastrophic Error

1. GPIO interrupt. Out Of Band.
2. SEL record. Out Of Band.
3. BERT record. In Band
4. Crash dump. Out Of Band and In Band; Catastrophic error only.

## Non-catastrophic Error

1. GPIO interrupt. Out Of Band.
2. SEL record. Out Of Band.
3. HEST record. In Band
4. GHESv2 notification. In Band
    a. Polled
    b. Interrupt
    c. SDEI
    d. SEA
    e. SEI

# DRAM error handling

## Boot time issue

- Reasons:
  - DIMM training failure.
  - SPD access failure.
- Reponses:
  - Halt the boot.
  - Continue booting with SEL record sent.

## Run time issue

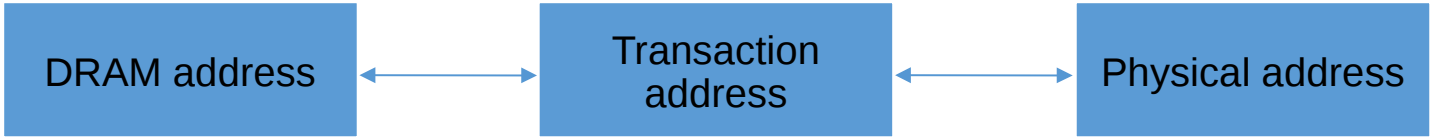
- Correctable error
- Uncorrectable error in non-secure memory
  - Recoverable.
  - Unrecovable, in non OS critical region
  - Unrecovable, in OS critical region.
- Uncorrectable error in secure memory

# DRAM error reporting

Error location types concerned:

a) Apps developer: Physical address
b) Data center admin: FRU (Field Replaceable Unit)
c) Platform vendor: DRAM address

Channel interleaving complicates address translation:



# Memory scrubbing

DRAM scrubbing prevents future error from happening.

a. Firmware initiated on-demand scrubbing, upon memory error.
b. Firmware initiated interval based scrubbing, as defined by user.
c. OS initiated on-demand scrubbing.
    1. Platform exposes scrubbing support capability to OS through ACPI RASF table.
    2. OS setup, start, stop scrubbing through RASF PCC channel.

# PCIe error handling -- workflow

Firmware First Handling being asked.

1. PCIe device (such as end point device) detects error.
2. The error is reported up through AER (Advanced Error Reporting) mechanism.
3. PCIe Root Complex on the SoC issues error interrupt.
4. Firmware secure interrupt handler analyzes and reports the error.

# PCIe error handling -- ACPI

a) OSC: OS conveys APEI support to platform.
b) OSC: When OS requests control over PCIe AER, platform does not give the control.
c) APEI error source: GHESv2 error source, no PCIe RP AER structure, no PCIe device AER structure.
d) CPER: PCIe error section.

# PCIe error handling – SEL sample

a) Multiple SEL records generated per error.
b) Some are generated from error producer, others from error consumer.
c) SEL records hold info such as error type (bus error, PERR, SERR, etc.), BDF numbers, vendor/device IDs, error ID (bad TLP, bad DLLP, etc.)

**SEL records generated**

632 | 02/14/2018 | 14:26:33 | Critical Interrupt #0xa1 | Bus Correctable error | Asserted
633 | 02/14/2018 | 14:26:33 | OEM record c0 | fd2000 | 7d1784af7375
634 | 02/14/2018 | 14:26:33 | Critical Interrupt #0xa1 | Bus Correctable error | Asserted
635 | 02/14/2018 | 14:26:33 | OEM record c0 | fc4000 | 8680c9107071
636 | 02/14/2018 | 14:26:33 | OEM record c0 | fc4000 | 8680c9107800
637 | 02/14/2018 | 14:26:33 | Critical Interrupt #0xa1 | Bus Correctable error | Asserted
638 | 02/14/2018 | 14:26:33 | OEM record c0 | fc4000 | 8680c9107071
639 | 02/14/2018 | 14:26:33 | OEM record c0 | fc4000 | 8680c9107800

# Power/Thermal management

Sensors

a. Platform sensors
b. DIMM sensors
c. On-chip sensors

Defenses

d. DIMM temperature based memory throttling
e. Other sensor temperature based Close Loop Thermal Throttling
f. Last defense: Software and hardware based thermal trip

Power optimizations

g. Autonomous DVFS
h. Turbo mode and CPPC

# Catastrophic Error -- Detection

a) Hardware error interrupt handler in Trusted Firmware – A (AKA. ARM Trusted Firmware).
b) Secure watchdog bite in on-chip management controller
    a) Watchdog signal 0 routed as SPI to GIC.
    b) Watchdog signal 1 routed to the platform, such as on-chip management controller
c) BMC (Board Management Controller) detected heart beat stop from on-chip management controller
d) OS initiated through ACPI PDTT mechanism

# Catastrophic Error – crash dump

a) Contains data needed for failure analysis.
b) Vendor proprietary tool provided to analyze the failure.
c) Modularized to provide fault tolerance.
d) Saved in both non-volatile storage and BMC.

# Building Blocks

# Early boot time considerations

Report issues happened before interrupt can be enabled:

a. Check error status register at certain check points.
b. Check error status register right before interrupt is enabled.
c. Enable hardware error interrupt as soon as possible.

Prevent false positives:

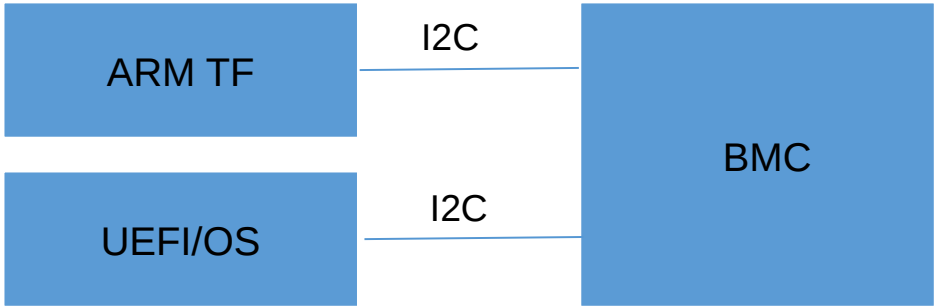
d. Filter out noises happened during training/initialization process.
e. Clear status registers before enabling hardware error interrupt.

# SEL record

## I2C bus access conflict resolution

a. Dedicated I2C connection between Trusted Firmware – A and BMC.
b. Dedicated I2C connection between non-secure world (UEFI/OS) and BMC.

## BMC to handle simultaneous SEL record access



# Configuration

Early boot loader configuration

a. Some customers want boot to be halted upon single DIMM training failure.
b. Other customers want boot to continue unless all DIMMs failed.

ARM TF configuration (For example, memory CE threshold needs)

c. Length of sliding window.
d. The threshold number.

Enablement of display/update of configurations in OS

e. Stored in a partition in BootRom, separated from UEFI variable partition.
f. Exposed through UEFI variable run time service.
g. Protected from firmware update.
h. Able to be set to factory default.

# Error injection

## Memory Error

- Inject into a specific physical memory
- Inject into a specific DIMM location

## PCIe Error

- Inject through Root Port
- Inject through End Point device
- Inject through PCIe analyzer

# Reflections

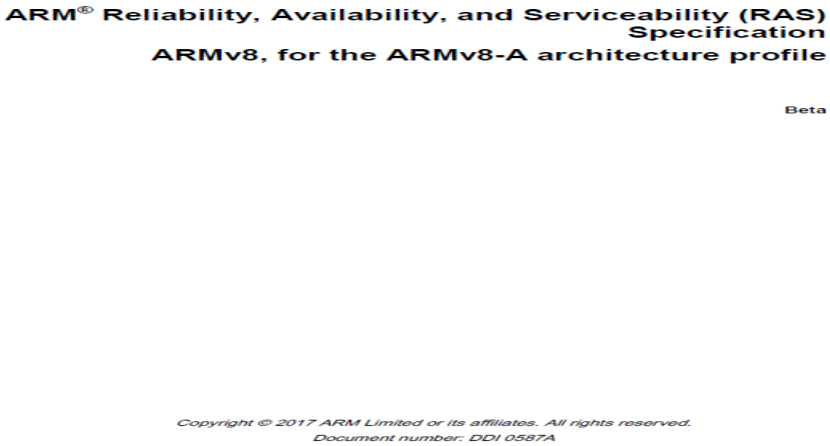
# Hardware design

## Reduce cycle stealing

a. Route hardware error interrupt to on-chip management controller
b. Manage Correctable Error threshold in hardware

## Standardized design

c. Support RAS extension


ARM® Reliability, Availability, and Serviceability (RAS) Specification
ARMv8, for the ARMv8-A architecture profile

Beta

Copyright © 2017 ARM Limited or its affiliates. All rights reserved.
Document number: DDI 0587A

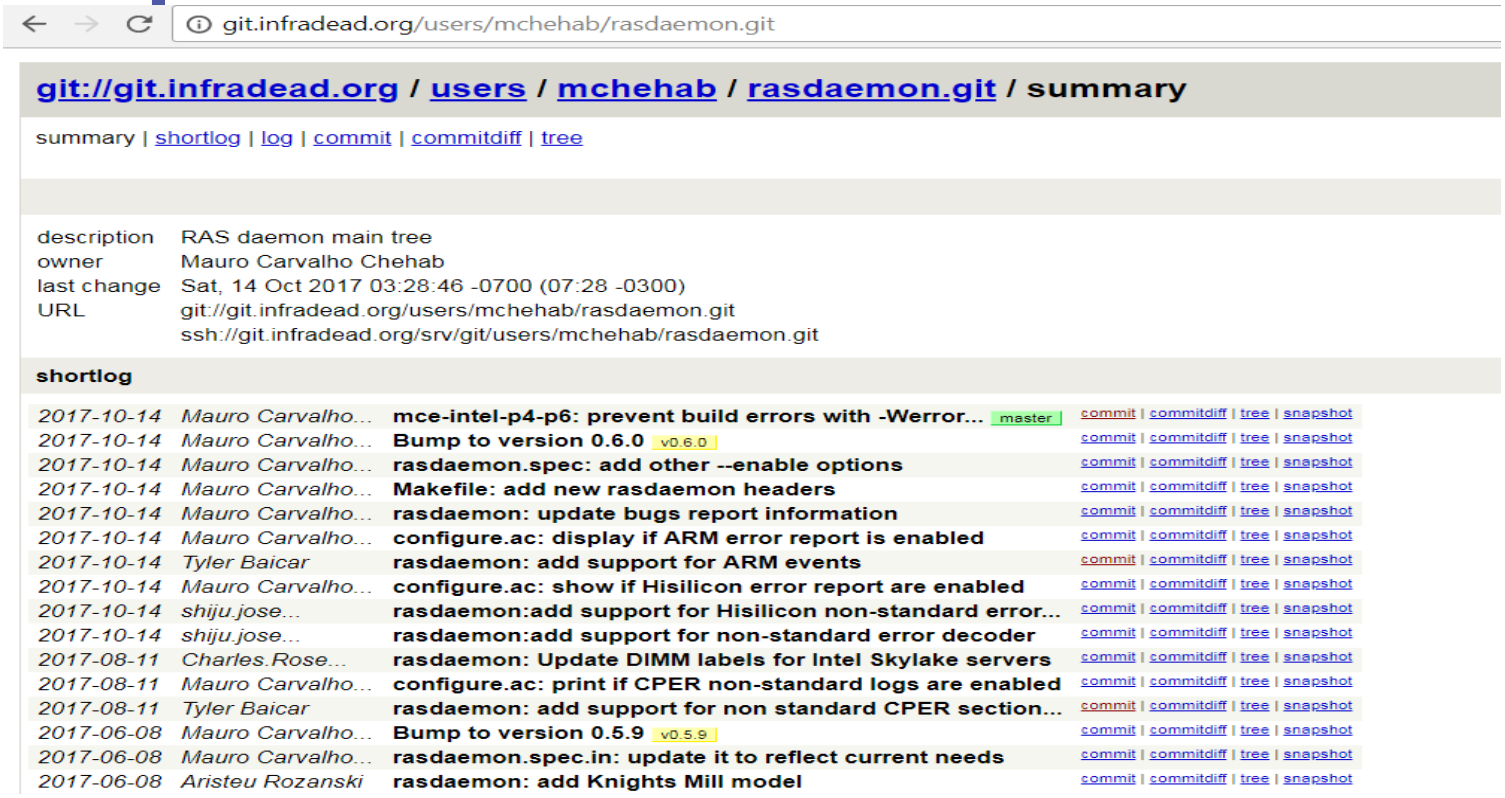
# Firmware RAS Features

- Exception Handling Framework in Trusted Firmware – A.
  - A framework for triaging RAS errors in EL3 prior to handling or delegation to lower EL
  - Up-stream support available for triaging RAS error interrupts and delegation through SDEI
  - Support for v8.2 RAS extension under review
    - Error synchronization barrier
    - Standard Error records
  - Support for v8.4 RAS extensions planned next
- Software Delegated Exception Interface dispatcher in Trusted Firmware – A.
  - A mechanism to deliver high priority non-maskable events to Normal world
  - Hooks into EHF as a way of delegating RAS errors to Normal world
  - Up-stream support available for physical SDEI dispatcher
- RAS error handling in a Secure partition
  - Support for CPER creation in a EDK2 Standalone MM partition in S-EL0 under development
  - Enables isolation of platform specific code in a EL3 managed sandbox
- DRAM error handling prototyped on SGI-575 using above components

# Kernel RAS Features

- Memory failure handling merged
- All Arm specific APEI notifications (SEA, SEI, SDEI) work
- Patches to make APEI support aware of multiple NMI sources under review
- Support for SDEI Client driver merged
- Support in APEI for registering SDEI notification handler under review
- Support for IESB in v8.2 RAS Extension merged
  - SErrors normally unmasked when in kernel
- Errors in KVM-Guest memory forwarded to Qemu/Kvmtool
- Support for notifying KVM-Guest under development

# User space daemon



git.infradead.org/users/mchehab/rasdaemon.git

**git://git.infradead.org / users / mchehab / rasdaemon.git / summary**

summary | shortlog | log | commit | commitdiff | tree

| | |
|---|---|
| description | RAS daemon main tree |
| owner | Mauro Carvalho Chehab |
| last change | Sat, 14 Oct 2017 03:28:46 -0700 (07:28 -0300) |
| URL | git://git.infradead.org/users/mchehab/rasdaemon.git |
| | ssh://git.infradead.org/srv/git/users/mchehab/rasdaemon.git |

**shortlog**

| | | | |
|---|---|---|---|
| 2017-10-14 | Mauro Carvalho... | mce-intel-p4-p6: prevent build errors with -Werror... master | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | Bump to version 0.6.0 v0.6.0 | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | rasdaemon.spec: add other --enable options | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | Makefile: add new rasdaemon headers | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | rasdaemon: update bugs report information | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | configure.ac: display if ARM error report is enabled | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Tyler Baicar | rasdaemon: add support for ARM events | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | Mauro Carvalho... | configure.ac: show if Hisilicon error report are enabled | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | shiju.jose... | rasdaemon:add support for Hisilicon non-standard error... | commit \| commitdiff \| tree \| snapshot |
| 2017-10-14 | shiju.jose... | rasdaemon:add support for non-standard error decoder | commit \| commitdiff \| tree \| snapshot |
| 2017-08-11 | Charles.Rose... | rasdaemon: Update DIMM labels for Intel Skylake servers | commit \| commitdiff \| tree \| snapshot |
| 2017-08-11 | Mauro Carvalho... | configure.ac: print if CPER non-standard logs are enabled | commit \| commitdiff \| tree \| snapshot |
| 2017-08-11 | Tyler Baicar | rasdaemon: add support for non standard CPER section... | commit \| commitdiff \| tree \| snapshot |
| 2017-06-08 | Mauro Carvalho... | Bump to version 0.5.9 v0.5.9 | commit \| commitdiff \| tree \| snapshot |
| 2017-06-08 | Mauro Carvalho... | rasdaemon.spec.in: update it to reflect current needs | commit \| commitdiff \| tree \| snapshot |
| 2017-06-08 | Aristeu Rozanski | rasdaemon: add Knights Mill model | commit \| commitdiff \| tree \| snapshot |

# BMC

GitHub, Inc. [US] | https://github.com/openbmc/openbmc

## Features of OpenBMC

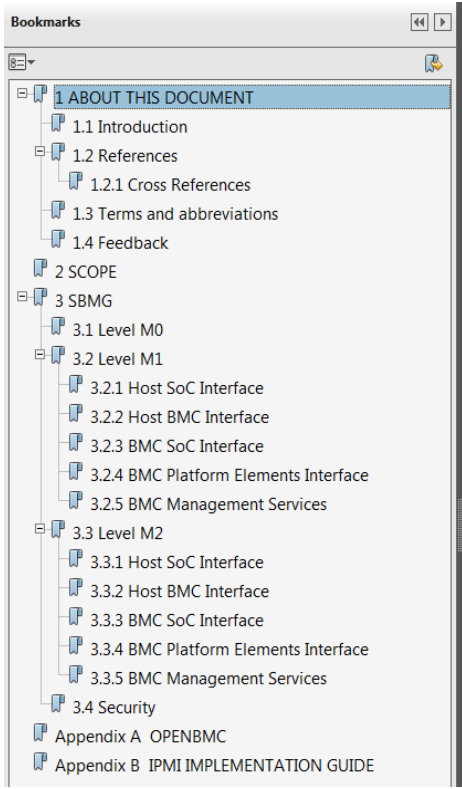
**Feature List**

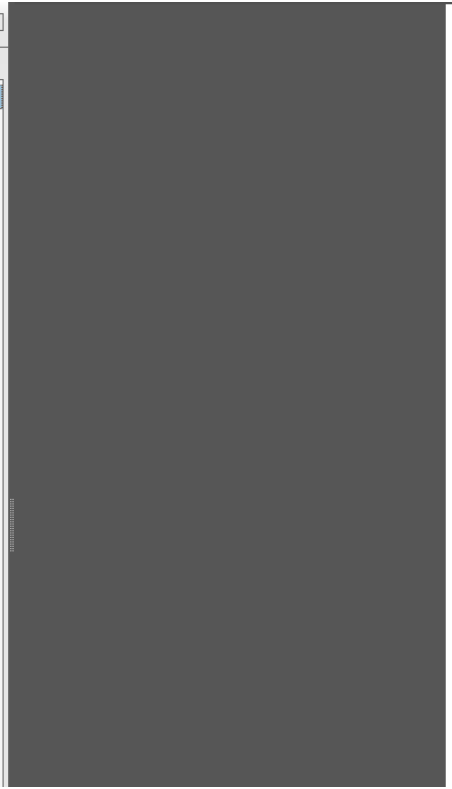
- REST Management
- IPMI
- SSH based SOL
- Power and Cooling Management
- Event Logs
- Zeroconf discoverable
- Sensors
- Inventory
- LED Management
- Host Watchdog
- Simulation
- Code Update Support for multiple BMC/BIOS images
- POWER On Chip Controller (OCC) Support

**Features In Progress**

- Full IPMI 2.0 Compliance with DCMI

# ARM Server Base Manageability Guide



ARM® Server Base Manageability Guide 1.0

**Platform Design Document**

CONFIDENTIAL

# Firmware First vs. Kernel First

- In general, firmware first is asked by server industry.
- Kernel first is desired for some products.
- ACPI table available for review that describes the RAS error node topology to the kernel.
- https://connect.arm.com/dropzone/systemarch/DEN0061A-RAS_ACPI-1.0alp1.pdf

# Challenges

## New technologies

a. NVDIMM: ACPI NVDIMM Sub Team working on a number of proposals.
b. New DIMM technologies: DDR5.
c. New bus technologies: PCIe 5.0; CCIX; Gen-Z.

## SoC complexities

d. SoC instead of chip set.
e. SoC interconnect.
f. Large number of cores, big size of caches, fast speed buses
g. On-Chip controllers (PCIe/SATA/USB, etc.)

# Acknowledgement


- Achin Gupta (ARM)
- Charles Garcia-Tobin (ARM)

# Thank You

**#HKG18**

HKG18 keynotes and videos on: connect.linaro.org

For further information: jon.zhixiong.zhang@gmail.comwww.linaro.org