Open. Together.
Case Study: Alternatives for SMM Usage in Intel Platforms

Sarathy Jayakumar, Principal Engineer
Intel Corporation
Problem Summary

System Management Mode (SMM) issues to address

• Degrades performance & quality of service (QoS)
• SMM latency increases with core count
• Firmware-based reliability of service (RAS) features
• SMM model adds complexity to firmware
• Multi-core asynchronous events, no concept of interrupt priority or reentrancy, race conditions, handler code, …
• Security concerns due to higher SMM privilege level
OS View of SMM

OS / Drivers

- ACPI Tables (ex. PCCT)
- UEFI Runtime (RT) Services (ex: SetVariable)
- ACPI DSM Methods (ex: ARS)

SMM

Platform Hardware (Processor, Memory, I/O, ...)

OPEN SYSTEMS
FIRMWARE

Case Studies

Open. Together.
OS View of SMM

Software SMM elimination strategy should not impact OS to ‘Platform Abstraction’ interface.

ACPI/RT services provide platform abstraction.

Software SMI triggers are transparent to the OS.

Hardware SMI triggers are transparent to the OS.

ACPI Tables (ex. PCCT)

UEFI Runtime (RT) Services (ex: SetVariable)

ACPI DSM Methods (ex: ARS)

Platform Hardware (Processor, Memory, I/O, ...)

OS / Drivers

OPEN SYSTEMS FIRMWARE

Case Studies
Categories of SMM Handler

Current Model

1: Software SMI that do not require SMM privileges (ex: Address translation, NVDIMM DSMs, etc.)

2: Software SMI that require SMM privileges

3: Hardware SMI and RAS Handlers that do not require privileges

4: Hardware SMI and RAS Handlers that require privileges

ASL + PRM

Capsule Update + OS Driver

Firmware Update

UEFI Variable Services, Firmware Update

OOB

OOB + PRM
What about a Driver-based Model?

1. Do not want platform knowledge in OS driver
2. Requires intimate platform/silicon knowledge (ex: Address Translation for RAS)
3. Variance between platform implementation/generation
Examples of Driver-based Issues

**PSHED Plug-in:** Not a viable deployment model due to ACPI abstraction, which uses SMI for complex tasks.

**Address Translation:** Originally pushed to EDAC drivers. OS vendors prefer ACPI to keep driver generic. ACPI relies SMM to handle complex algorithms.

**NVDIMM Drivers:** Uses ACPI to keep NVDIMM drivers generic. Relies on ACPI (again) which (still) uses SMM to handle complex tasks (this is a trend).
Platform Runtime Mechanism (PRM)

- Mechanism to invoke native code from ACPI
- Uses ASL as a landing point for runtime events
- ASL will invoke PRM if required ("ASL Assist")

**ACPI Source Language (ASL)**

```
Notify 0x80
...
Device
  _HID ACPIxxx
```

**Bridge Driver**
- Binds to ACPI Object
- Calls RT Protocol

**PRM Handler (UEFI RT Protocol)**

Note: PRM is not a new capability. It is based on combining existing capabilities.
Case Study: Using PRM for Correctable Error (CE) Handling

Handling correctable errors (Option 1)

Memory Error Subsystem

Peripheral Controller Hub (PCH)

ERR0 Pin

GPIO

SMI

SCI

[lxx Method]

PRM Build HEST

APEI Notify to OS
Case Studies
Call to Action

Work together to accelerate SMM reduction.
Move software SMM Handlers to PRM.

Bridge driver and sample PRM handler available in GitHub:
https://github.com/tianocore/edk2-staging/tree/PRMCaseStudy

Please review & provide feedback!
Glossary

PCCT – Platform Communication Channel Table
DSM – Device Specific Methods
ARS – Address Range Scrubbing
OOB – Out Of Band
PRM – Protected Runtime Mechanism
PSHED – Platform Specific Hardware Error Driver
EDAC – Error Detection And Correction
SCI – System Configuration Interrupt
HEST – Hardware Error Sources Table
APEI – ACPI Platform Error Interfaces

*Other names and brands may be claimed as the property of others
© Intel Corporation