Hardware Management for CG-OpenRack-19

Suzanne Kelliher, Product Line Manager, Radisys
Nilan Naidoo, Principal Engineer, Radisys
Agenda

- Hardware Architecture Overview
- Hardware Management Overview
- Rack Level Hardware Management
- Next Steps
CG-OpenRack-19 High Level Architecture

Switching

Usable Compute / Storage Capacity

Power

Standard 19" Rack

Data Plane Switch
Dev Mgmt Switch
App Mgmt Switch

Full Shelf
½ Shelf

Vertical 12VDC bus bar in frame mates with power connector located on sled

4 x optical fiber ports via blind mate rear connector to sled
• Create Cohesion Across CG-OpenRack-19 Implementations

• Leverage OCP hardware management premise
  - Leverage existing HW management standards: IPMI 2.0, DCMI 1.5 and Redfish
  - Each node is independently managed by BMC
    - Includes cooling of shelf containing the node

• Add Options as Necessary for Simple, Efficient Rack Management
  - Device Management switch can be used to run Rack Management applications
    - Example, Location Aware Discovery
  - Rack Agent Module provides access to PSU & PDU, and additional physical security features, i.e. door locks

• Options for Rack Management
  - Provide basic rack level management using Redfish API based on open sourced Intel® RSD framework
  - Intel® RSD Architecture Compliant

  Data Plane Switch
  Data Plane Switch
  Dev Mgmt Switch
  App Mgmt Switch
  ½ Shelf ½ Shelf ½ Shelf ½ Shelf ½ Shelf ½ Shelf
  Full Shelf Full Shelf Full Shelf Full Shelf Full Shelf
  Rack Agent (Optional)
  PSU 12V, 1U
  PSU 12V, 1U

  • Connects to dedicated BMC port on each node, Rack Agent & Management port of other switches
  • One uplink out of rack provides OOB management access to all devices in rack
  • Open Linux environment enables Rack Level Management applications

  • Shelf HW Management provided by Server BMC
  • FRU Inventory
  • Sensor Data
  • Power on/off/reset
  • Power consumption
  • Boot order control
  • Remote Console (SOL, KVM)
  • Virtual media
  • Front Panel Indicators
  • Interfaces: IPMI 2.0, DCMI 1.5, Redfish

  • Rack Agent provides Ethernet access to PSU & PDU
  • Abstracts PSU & PDU management standard interface (IPMI, Redfish, SSH CLI)
  • PSU & PDU Inventory
  • Rack level power
• **Intel® RSD is a logical architecture that disaggregates compute, storage, and network resources:**
  - Introduces the ability to pool these resources for more efficient utilization of assets
  - Provides the ability to dynamically compose resources based on workload-specific demands from a set of compute, fabric, storage, and management modules that work together to build a wide range of virtual systems

• **The design uses four basic pillars:**
  - POD Manager for multi-rack management
  - Pooled system of compute, network, and storage resources are composed based on workload requirements
  - Pod-wide storage built on Ethernet-connected storage
  - A configurable network fabric of hardware, interconnect with cables and backplane, and management software

• **Intel RSD based on open industry standard Redfish**

• **Intel has open sourced reference implementation of following components on:**
  - Pod Manager
  - Pooled System Management Engine (PSME)
  - Rack Management Module (RMM)
  - Validation Test Suite (VTS)

Source code: [https://github.com/01org/intelRSD](https://github.com/01org/intelRSD)
• **A key attribute of Intel® RSD management is location-aware discovery**
  • A mechanism for numbering each component is required

• **Each Rack has a unique ID**
  • Configured by operator

• **RSD defines a 3 level hierarchy for modeling computer systems**
  • Drawer – maps to a shelf
  • Module – logical entity
  • Blade – maps to server motherboard

• **Numbering scheme for blades:**
  • <Drawer Row>.<Drawer Column>.1.<Blade Id>
Rack Agent Architecture

- **Rack Agent module consists of a Controller module following I/O:**
  - I2C interface to interface to PMBus
  - Ethernet Interfaces for uplink to device management switch
  - Serial console for debugging & initial setup
  - GPIO signals to monitor PSUs and Circuit Breakers on PDU
  - Other sensors required to monitor health of the module
  - OpenBMC is a good fit

- **PSU/PDU Management**
  - Presence & Inventory info of PDU & PSU
  - PSU Input and Output Voltage/Current
  - PDU Circuit Breakers
  - Temperature
  - Fan speed & status
Extended RSD PSME reference code to run on Device Management switch
- Extended Chassis and Compute GAMI IPMI interfaces to interact with BMC
- Extended Network Agents to run on Cumulus Linux on Data switches

Added Location Aware Discovery application to discover and determine blade locations
- Monitors switch ports to determine presence/absence of devices in the rack
- Uses Port-to-Device Mapping configuration file to map learned MAC addresses to Blade & Switch location
  - MAC -> Port -> Location
- To overcome limited visibility of blade inventory through IPMI, uses device profile files for each Product Id
  - Profile describes device tree of server

PSME Interfaces to Location Aware Discovery application through API
- Retrieves BMC parameters
- PSME will use contents of BOM file to fill in information not accessible via IPMI
- Listens for device state changes
• **Discovery**
  - Chassis
  - Computer systems
  - Managers

• **Server Information**
  - Server identification and asset info
  - Host Network MAC addresses
  - Local storage
  - Power supply and fans
  - State and Status

• **Common Manageability**
  - Change boot order / device
  - Reboot / power cycle server
  - Power usage and thresholds
  - Temperature

• **BMC Infrastructure**
  - View / configure BMC network settings

• **Access and Notification**
  - Subscribe/publish event model
To provide cohesion across CG-OpenRack-19 implementations

We are considering contributing the location aware discovery application and Intel® RSD enhancements
- It enables basic hardware management of rack using Redfish

Please join us on in the Radisys booth to see DCEngine and see a demonstration of this work.
Thank You