OpenVMS Integrity
Boot Environment

Thomas Siebold, Senior Software Consultant
Business Critical Systems
Transition Engineering and Consulting Group
thomas.siebold@hp.com

© 2005 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice

Topics
- Hardware
  - Systems
  - Options
- Boot Environment
  - EFI
  - Disk structure
  - Booting
OpenVMS v8.2 orderable today on these servers
Firmware Revisions

- System       Syst. F/W  BMC  MP
- rx1600 (Nemesis)  1.10  2.33  E.02.29
- rx1620 (Onyx)  2.11  3.48  E.03.13
- rx2600 (Long's Peak)  2.31  1.52  E.02.29
- rx2620 (Badger Peak)  3.10  3.47  E.03.13
- rx4640/Madison&Hondo  2.13  2.37  E.02.29
- rx4640/Mad9M  3.11  3.47  E.03.13

CPUs supported

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx1600</td>
<td>1.0Ghz</td>
</tr>
<tr>
<td>Rx1620</td>
<td>1.3 (3MB), 1.6 (3MB) [Mad9m]</td>
</tr>
<tr>
<td>Rx2600</td>
<td>1.0, 1.3, 1.4, 1.5</td>
</tr>
<tr>
<td>Rx2620</td>
<td>1.3 (3), 1.6 (3), 1.6 (6) [Mad9m]</td>
</tr>
<tr>
<td>Rx4640</td>
<td>1.3 (3/6), 1.6 (3/6), Hondo 1.1</td>
</tr>
<tr>
<td>Rx4640</td>
<td>1.5 (4), 1.6 (6), 1.6 (9) [Mad9m]</td>
</tr>
</tbody>
</table>
Memory expansion

- Rx1600/1620: 512MB – 16 GB
- Rx2600/2620: 1GB – 24GB
- Rx4640: 1GB – 64 GB/128GB (4Gb DIMMs)

Options

- Single and 4 port 10/100 Tulip (A5230A, A5506B)
- Broadcom 5701 Gbit copper and fiber NICs (A6847A, A6825A)
- Intel Gbit copper and fiber NICs (A7011A, A7012A)
- Qlogic 2-port FC adapter (A6826A)
- U320 2p SCSI adapter (A7173A)
- Radeon 7500 graphics card (AB551A)
- FC/Broadcom 5703 copper and fiber combo cards (A9782A [Fiber], A9784A [Copper])
- Fermat 6402 and 6404 2 and 4 port U320 RAID controllers (A9890A, A9891A) [schedule - around May, 2005, on OpenVMS/Itanium; January, 2005 on Alpha]
and now for something NEW….

„EFI“

The Extensible Firmware Interface

- A new standard from Intel
- Replaces BIOS
- EFI firmware on the system
  - Includes a user interface called the “shell”
  - EFI commands native in the firmware
  - Interface to the system hardware
  - Runs EFI applications from the EFI system partition
Benefits of EFI abstraction

- Abstraction of OS from firmware
- Abstraction free of legacy interfaces
- Coherent, scalable platform interface

- No Collision
- No Space Limitation
- Support Speedy Boot
- Provide Drivers to OS

EFI structural model
EFI operational model

- Boot starts with hardware initialization

- Continues with a sequence of loads
  - Each successive loader is a bit “smarter”

- Culminates in the loading of an operating system

EFI system partition

- An EFI system partition is a FAT32 file system

- EFI directory in the root directory `\efi`

- Vendors use subdirectories to store their OS loaders and applications

- On HP OpenVMS I64 systems, the boot loader filename is `vms_loader.efi` and is located in `fs0:\efi\vms`
EFI Shell [Built-in]

```
EFI Boot Manager ver 1.18 [14.6.1] Firmware ver 2.28 [43]1
Please select a boot option

- EFI Shell [Built-in]
- Boot Option Maintenance Menu
- System Configuration Menu

Use ^ and v to change option(s). Use Enter to select an option.

Loading: EFI Shell [Built-in]
EFI Shell version 1.18 [14.6.1]
```

Device mapping:
- fs0: /dev/loop0
- blk0: /dev/loop0

```
Shell>
```

EFI system partition

```
Shell> dir
1: Cannot open current directory - No mapping
Exit status code: No mapping

Shell> fs0:
fs0> dir
Directory of: fs0:
11/21/83 07:38 <DIR> 0 bytes
          1 Dir(s)
```

```
Shell> fs0:
```

```
```
EFI system partition files

- EFI requires GUID Partition Table (GPT) disk format
  - GUID = Globally Unique Identifier
- EFI requires one FAT32 partition
  - Contains EFI system partition files
- OpenVMS requires ODS-2 or ODS-5 format disk with Files-11 file structure
- OpenVMS does not support partitioned disks!
- Both formats co-exist, independent of each other
Hybrid system disk: EFI view

- GPT format with one or more FAT32 partitions
  - EFI system partition
  - Diagnostics partition (optional)
- Remaining space marked allocated

Hybrid system disk: OpenVMS view

- ODS Files-11 format disk
- One container file for each FAT32 partition
- Optical media use ISO9660 format instead of GPT
OpenVMS view of EFI system partition

Seen from OpenVMS:
- SYS$EFI.SYS
- PC style boot block

Seen from EFI:
- FAT32 partition
- OS Loader: VMS_LOADER.EFI

GPT.SYS:
- MBR
- Pointer to EFI partition
- GPT (GUID Partition Table)

Hybrid system disk: dual view
OpenVMS I64 bootstrap sequence

EFI System Partition (FAT32)  OpenVMS File System (ODS-2 or ODS-5)

VMS_LOADER.EFI  IPXE.EXE  SYSBOOT.EXE  SYS$BASE_IMAGE

Booting OpenVMS I64
Console

Integrity Servers – Hardware Overview

- No “Vax like” or “Alpha like” console
- Has multiple consoles:
  - Management Processor (MP)
  - Baseboard Management Console (BMC)
  - Both attempt to be common across the entire hardware range
- Uses Extensible Firmware Interface (EFI) rather than BIOS.
MP console

- Runs with box level power, even with system off.
- Local, remote (modem) and network connectivity
- Console configuration (terminal type, etc.)
- Network configuration (hostname, IP address, etc.)
- Multiple console sessions (one writer, many readers)
- Provides ability to copy files over the network (firmware updates)

BMC

- Runs with main board powered up
- Local connectivity (9 pin serial)
- Power up, self tests
- Device detection
- Console configuration
- No graphics console
EFI

- Mini operating system
- FAT formatted file system (FAT12, FAT16 and FAT32), VMS presents FAT16 partition to EFI
- Boot menu and defaults
- Environment variables (VMS_FLAGS, etc.)
- VMS_LOADER.EFI finds and loads IPB.EXE
- IPB.EXE understands the OpenVMS file system, EFI does not.

Boot and Device detection

- EFI boot loader from FAT partition (hidden as a container file on the system disk)
- Boot flags passed through environment variables
- Reads executive into memory
- Passes control to the executive
- The system uses ACPI (Advanced Configuration and Power Interface) for device detection by the firmware
- Devices appear as a set of CSRs (Control and Status Registers) in physical memory – the I/O space.
Boot and Device detection

- Devices have interrupt vectors which connect a device interrupt request to the device driver interrupt service routine. Device data obtained from ACPI data.
- ACPI data indicates device type.
- SYSMAN IO AUTO will query ACPI data to find devices and set up OpenVMS device drivers to communicate with the hardware.

Now Let’s take a look, how the past 6 slides look at real life….

Booting VMS from the EFI Shell

- Select EFI Shell from the boot menu
- Set Boot flags environment variable, stored in NVRAM
  - IA64 flag values are generally the same as Alpha and VAX
  - Shell> set vms_flags “0,0”
- Select disk and directory
  - Shell> fs0:
  - fs0:> cd efi\vms
- Start the boot of VMS
  - fs0:> vms_loader
- Override environment variable
  - fs0:> vms_loader –flags 0,1
VMS_SHOW

- EFI utility to associate VMS device name with EFI pathnames

```
fs0:\efi\vms> vms_show devices
VMS: EIA0
  EFI: Acpi(000222F0,0)/Pci(3|0)/Mac(00306E3829B5)

VMS: DKA0
  EFI: fs0: Acpi(000222F0,100)/Pci(1|0)/Scsi(Pun0,Lun0)

VMS: EWA0
  EFI: Acpi(000222F0,100)/Pci(2|0)/Mac(00306E38F938)

VMS: DKC200
  EFI: fs1: Acpi(000222F0,200)/Pci(1|0)/Scsi(Pun2,Lun0)
```

VMS_SET

- EFI utility to allow selection of dump devices and HLL debugger port:

```
fs0:\efi\vms> vms_set dump_dev dka0,dkc200
VMS: DKA0
  EFI: fs0: Acpi(000222F0,100)/Pci(1|0)/Scsi(Pun0,Lun0)

VMS: DKC200
  EFI: fs1: Acpi(000222F0,200)/Pci(1|0)/Scsi(Pun2,Lun0)

fs0:\efi\vms> vms_show dump_dev
VMS: DKA0
  EFI: fs0: Acpi(000222F0,100)/Pci(1|0)/Scsi(Pun0,Lun0)

VMS: DKC200
  EFI: fs1: Acpi(000222F0,200)/Pci(1|0)/Scsi(Pun2,Lun0)
```
Forcing a system crash

- **History**
  - On VAX and early Alpha system, ^P was detected by special hardware and handled by the console
  - Commodity serial ports pushed ^P detection in console software, if IPL < DEVICE_IPL

- EFI console ignores ^P, no way to return to EFI Shell

- IA64 VMS console terminal driver handles ^P
  - Calls XDELTA, if loaded
  - Prompts for forced crash, if IPL < DEVICE_IPL

- Remote console command generates Transfer of Control signal (BMC:TOC, MP:TC)
  - Handled by IA64 VMS as a non-maskable exception and

From this point on....

VMS is VMS is VMS.....