Chapter 29
-
Itanium Architecture

INDEX

Introduction

When PA-RISC was released, HP began designing the architecture to replace it. Several years into the project, HP determined that the economics of microprocessor manufacturing favored partnership and decided to partner with Intel, the leader in volume IC manufacturing. Intel was working on their next generation architecture as well, so the timing was perfect. This joint development combined HP's strengths in system and architecture design with Intel's strengths in processor design and manufacturing.

Overview of the Itanium Architecture

Traditional microprocessor architectures (CISC and RISC) have fundamental attributes that limit performance. To achieve higher performance, processors must not only execute instructions faster, but also execute more instructions per cycle, referred to as “parallel execution”. Greater parallel execution allows more information to be processed concurrently - thereby increasing overall processor performance. In traditional architectures, the processor is often underutilized because of the compiler’s limited ability to organize instructions. Branches (instructions that change the flow of execution within the program) and memory latency (the time for data to arrive from memory) compound the already limited ability of today’s processors to achieve parallel execution.
To overcome these limitations, a new architecture was required. Traditional architectures communicate parallelism through sequential machine code that “implies” parallelism to the processor. Intel and HP jointly defined a new architecture technology called *EPIC (Explicitly Parallel Instruction Computing)* named for the ability of the software to extract maximum parallelism (potential to do work in parallel) in the original code and “explicitly” describe it to the hardware. Intel and HP have jointly defined a new 64-bit *Instruction Set Architecture (ISA)*, based on EPIC technology, which Intel has incorporated into Itanium, Intel’s 64 bit microprocessor architecture. The new 64 bit ISA takes an innovative approach combining explicit parallelism with techniques called predication and speculation to progress well beyond the limitations of traditional architectures.
The new architecture is called *Itanium*™, formerly known as IA-64.

Main features of EPIC

The main features of EPIC are *explicit parallelism*, *predication* and *speculation*.

Explicit parallelism

In traditional architectures like RISC the processor receives a sequential stream of instructions from the compiler and must reorder the instructions to prevent functional units from being idle. The processor can only reorder a small, fixed number of instructions. A particular functional unit may be idle even though there are instructions in the instruction stream destined for that functional unit.

The concept behind explicit parallelism is that instructions arrive at the processor explicitly ordered by the compiler. The compiler organizes the code for an entire program and makes the ordering explicit so the processor can execute instructions in the most efficient manner.

Simpler, smaller chip control structures are possible when parallelism is exposed by the compiler instead of the hardware. Space saved on the chip can be used for additional functional units, large numbers of registers, and large caches -further increasing parallelism and overall performance.

Predication

Another major performance limiter for traditional architectures is branching. A branch is a decision between two sets of instructions. Today's architectures use a method called *branch prediction* to predict which set of instructions to load. When branches are mispredicted the whole path suffers a time delay. While current architectures may only mispredict 5-10% of the time, the penalties may slow down the processor by as much as 30-40%. Branches also constrain compiler efficiency and underutilize the capabilities of the microprocessor.

The new 64 bit ISA uses a concept called predication. Predication effectively executes both branches, rather than trying to predict the correct branch. When the correct branch is known, unnecessary results are discarded.
Predication can remove many branches from the code and reduce mispredicts significantly. A study in ISCA 1995 by Scott Mahlke and others, demonstrated that predication removed over 50% of the branches and 40% of the mispredicted branches from several popular benchmark programs. Thus, predication enables increased performance resulting from greater parallelism and better utilization of an Itanium based processor's performance capabilities.

Speculation

Memory latency (the time to retrieve data from memory) is yet another performance limitation for traditional architectures. Memory latency stalls the processor, leaving it idle until the data arrives from memory. Because memory latency has not kept up with increasing processor speeds, loads (the retrieval of data from memory) need to be initiated earlier to ensure that data arrives when it is needed.

The new 64-bit ISA uses speculation, a method of allowing the compiler to initiate a load from memory earlier, even before it is known to be needed, thus ensuring data is available for use if needed. As a result, the compiler schedules to allow more time for data to arrive without stalling the processor or slowing its performance.

Because the Itanium ISA allows the compiler to expose maximum parallelism in the code and explicitly describe it to the hardware, simpler and smaller chip control structures are possible. Space saved on the chip can then be used for additional resources, such as larger caches and many more registers and functional units. These, in turn, supply the processor with a steady stream of instructions and data to make full use of its capabilities, greatly increasing parallel execution and overall performance.

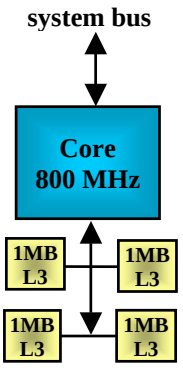
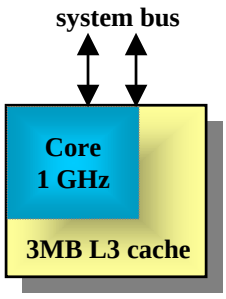
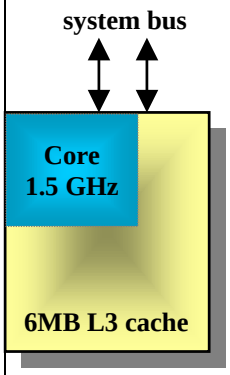
To provide more on-chip resources, Intel's Itanium based processors capitalize on both the strengths of explicit parallelism and the savings in chip space that the 64-bit ISA provides. Itanium based processors have massive resources, with 128 general (integer) registers, 128 floating-point registers, 64 predicate registers, 8 branch registers und 128 control registers. In contrast, today's RISC based processors typically have only 32 general registers and are

therefore forced to use register renaming or some other mechanism to create the resources necessary for parallel execution. In Itanium, the functional units attached to the large register file can also be replicated, making Itanium inherently scalable over a wide range of implementations. Of course, since replicated functional units increase the machine width, performance can be increased correspondingly. And with the more sizable caches and the many more read and write ports afforded to memory, the speed of Itanium based processors is no longer limited by the memory latency problems of traditional processors.

A more detailed description of the EPIC features including a code example can be found at http://www.software.hp.com/products/IA64/arch.html.

The Itanium processor family (IPF)

Intel announced several generations of the Itanium Processor Family (IPF). The following IPF processors are available as of today.

Feature	1st generation	2nd generation	
	Itanium	McKinley	Madison
CPU clock speeds	733/800MHz	900/1000MHz	1.3GHz/1.5GHz
System Bus			
width	64 bit	128 bit	128 bit
speed/transactions	133MHz/266 MT/s	200MHz/400 MT/s	200MHz/400 MT/s
bandwidth	2.1 GB/s	6.4 GB/s	6.4 GB/s
Width			
bundles per clock	2	2	2
integer units	4	6	6
loads/stores per clock	2 load or stores	2 loads and 2 stores	2 loads and 2 stores
issue ports	9	11	11
Caches			
level 1 size/latency	2x 16K / 2 clock	2x 16K / 1 clock	2x 16K / 1 clock
level 2 size/latency	96K / 12 clock	256K / 5-7 clock	256K / 5-7 clock
level 3 size/latency	2-4MB off die 20 clock	1.5-3MB on die 12-15 clock	3-6MB on die 14-17 clock
level 3 bandwidth	11.7 GB/s	32 GB/s	48 GB/s
Addressing			
physical	44 bit	50 bit	50 bit
virtual	50 bit	64 bit	64 bit
max. page size	256MB	4GB	4GB
Layout			

The following table shows available HP systems:

gen.	name	processors	clock speeds	chipset	systems
1st	Itanium	Itanium (fka Merced)	733/800MHz	Intel 82460GX	rx4610 (Ironman) rx9610 (Olympic/Azusa) i2000 (Bigsur)
2nd	Itanium2	McKinley	900/1000MHz	hp zx1	rx2600/zx6000 (Longs peak) zx2000 (Wilson peak) rx5670 (Everest)
2nd	Itanium2	Madison	1.5GHz	hp sx1000 (pinnacles)	rx1600 (Nemesis) rx4640 (Mt.Diablo) rx7620 (Eiger) rx8620 (Olympia) SuperDome (Orca)
3rd	Itanium3	Montecito	?	hp sx1000	tbd

NOTE: 1st generation itanium products were not intended as entry systems and have been discontinued meanwhile.

NOTE: McKinley systems do also support the Madison processor with UX 11.23 (rx2600 PSP)

Specifications of currently available HP Itanium systems can be found at:
http://www.hp.com/products1/itanium/servers_workstations

HP-UX Itanium Releases

These are the different HP-UX releases. The current PA release is UX 11.11. The current IPF release is UX 11.23.

release identifier	release name	architecture	supported processors
UX 11.00	n/a	PA only	PA7000/PA8000
UX 11.11	11i v1	PA only	PA8000
UX 11.20*	11i v1.5	IPF only	Itanium
UX 11.22	11i v1.6	IPF only	Itanium/McKinley
UX 11.23	11i v2	IPF only	McKinley/Madison
UX 11.31	11i v3	IPF & PA	?

*: UX 11.20 is obsolete meanwhile. It was intended for prerelease customers only.

Differences between HP-UX for PA and IPF

From the operating systems point of view there should be no noticable difference between HP-UX for PA and HP-UX for Itanium. The following table highlights the features that are not common between the OSes. Altough some applications are not bundled for the Itanium releases they might run in emulation mode through Aries. on the other hand there are some bundled applications, that are not Itanium native but do also run in emulation mode (e.g. SD-UX).

Feature	UX 11.11	UX 11.22	UX 11.23
Operating Environments	4 (EOE, MCOE, MTOE, TCOE)	2 (MTOE, TCOE)	5 (FOE, EOE, MCOE, MTOE, TCOE)
Compressed dumps	yes (with product CDUMP11i)	no	yes (by default)
Kernel configuration	GUI: SAM CLI: kmtune, mk_kernel	GUI: kcweb CLI: kmtune, mk_kernel	GUI: kcweb CLI: kctune, kconfig
Dynamic kernel tunables (*[1])	12	26 (UX 11.11 + 14)	32 (UX 11.22 + 6)
Automatic kernel tunables (*[1])	no	no	yes
Peripheral devices configuration	SAM	SAM	pdweb
System Configuration Repository	yes	replaced by System Inventory Manager (SIM)	replaced by System Inventory Manager (SIM)
Boot filesystem /stand	hfs	vxfs	vxfs
HP-UX Virtual Partitions (vPars)	yes	no	no
Instant Capacity on Demand (iCOD)	yes	no	yes
MxN threads (*[2])	no	yes	yes
HP-UX Workload Manager	yes	no	yes
Secure web console	yes	no	no
Central web console	yes	no	no
Hard partitions (nPars)	yes	no	yes
Autoport Aggregation (APA)	yes	no	no
ServiceGuard and SG Extension for RAC	yes	yes	yes
Metro-/ContinentalClusters	yes	no	yes
hptc/ClusterPack	no	yes	yes
Netscape Directory Server	yes	no	no
LDAP-UX Integration	yes	no	yes
HP-UX Kerberos Server	yes	no	yes
HP-UX Secure Shell	yes	no	yes
HP-UX AAA server	yes	no	no
IPFilter/IPSec	yes	no	yes

(*[1]): Refer to the Kernel chapter for details
(*[2]): MxN threading can boost the performance of Java applications

What needs to be considered on an IPF system

Usually you should not recognize if you are working at a PA system or an IPF system. HP-UX has the same “look and feel” on both platforms. The commands and the applications behave exactly the same from the user’s perspective. Anyway there are some specials that need to be considered. The new boot disk layout and the existence of the *Extensible Firmware Interface (EFI)* that is located between firmware and operating system impose some changes in typical admin tasks like setting up a root mirror or replacing a failed disk.

How an IPF system boots

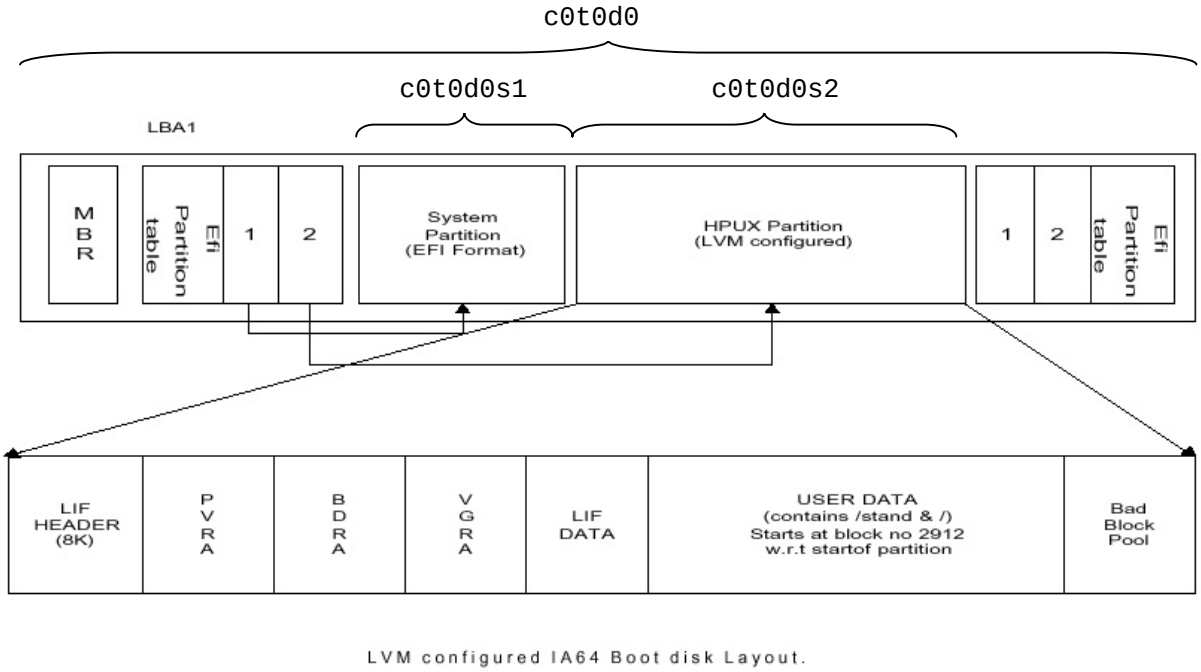
The IPF bootstrap process involves the execution of the following software components in that order:

- CMOS
- option ROM
- EFI
- EFI Boot Manager (EFI main menu)
- HP-UX Bootloader (hpux.efi)

After the processor is reset, firmware initializes and tests processors and platform. It then transfers control to EFI, the Extensible Firmware Interface. EFI, in turn, initializes EFI boot and runtime services and launches the Boot Manager. The Boot Manager, which allows loading of EFI application or drivers from EFI defined file system, loads and transfers control to hpux.efi, the HP-UX-specific bootstrap loader. hpux.efi then loads the HP-UX kernel object file from the HP-UX file system to memory and transfers control to the loaded kernel image.

Boot disk layout and EFI

The EFI (Extensible Firmware Interface) is an interface between HP-UX and the IPF platform firmware (BIOS). The file system supported by the Extensible Firmware Interface is based on the FAT file system. EFI encompasses the use of FAT-32 for a system partition, and FAT-12 or FAT-16 for removable media. The system partition is required on a bootable disk for the IPF platform. It is used by the IPF BIOS to locate the HP-UX bootloader (hpux). The following picture shows a Itanium boot disk under LVM control:



LVM configured IA64 Boot disk Layout.

For a hard disk, the system partition is a contiguous grouping of sectors on the disk, where the starting sector and size are defined by the EFI partition table, which resides on the second logical block of the hard disk, and/or by the Master Boot Record (MBR), which resides on the first sector of the hard disk.
The EFI System Partition can contain directories, data files, and EFI Images. The EFI system firmware may search the `\EFI` directory of the EFI system partition to find possible EFI images that can be loaded. The HP-UX bootloader is one example of an EFI Image.

From the OS point of view the device file /dev/dsk/c0t0d0 would represent the whole disk. c0t0d0**s1** would represent the *EFI system partitio*n (usually 100MB at UX 11.22 and 500MB at UX 11.23) and c0t0d0**s2** would represent the *HP-UX partition*. So the HP-UX partition on an IPF system represents a whole LVM boot disk on a PA system. Hence when dealing with Itanium boot disks you need to specify the s2 device file for e.g. pvcreate, pvdisplay, etc.

NOTE: Non-bootable disks are treated exactly the same on IPF and PA systems since they are not partitioned and do not contain EFI.

NOTE: As of **UX 11.23** there is a *HP Service Partition* c0t0d0**s3** (FAT-32) right behind the HP-UX partition. The service partition is created by Diagnostics installation. It's default size is 400MB. It can be accesses by the commands `efi_ls(1M)`, `efi_cp(1M)` etc.

The directory structure of a default EFI system partition containing HP-UX boot information is as follows:

`/startup.nsh`	The startup EFI shell
`/EFI/diag`	
`/EFI/tools`	
`/EFI/Intel_Firmware/fpswa.efi`	
`/EFI/HPUX/hpux.efi`	the HP-UX bootloader
`/EFI/HPUX/nbp.efi`	
`/EFI/HPUX/AUTO`	the AUTO file containing „boot vmunix“
`/EFI/hp/tools/network/README.network`	

```
/EFI/hp/tools/network/ftp.efi
/EFI/hp/tools/network/ifconfig.efi
/EFI/hp/tools/network/inet.nsh
/EFI/hp/tools/network/ping.efi
/EFI/hp/tools/network/route.efi
/EFI/hp/tools/network/tcpipv4.efi
```

These files can also be found below the directory `/usr/lib/efi/` on any Itanium HP-UX file system.

HP-UX contains a set of EFI utilities:

`efi_fsinit(1M)`	Initialize an EFI volume; i.e. create a header and an empty directory.
`efi_cp(1M)`	Copy files to and from an EFI volume.
`efi_mkdir(1M)`	Create directories in an EFI volume.
`efi_ls(1M)`	List the contents of an EFI volume.
`efi_rm(1M)`	Remove files from an EFI volume.
`efi_rmdir(1M)`	Remove directories from an EFI volume.
`idisk(1M)`	Creates partitions for IPF disks.

The EFI utilities are the only utilities in HP-UX where the internal structure of an EFI volume is known. To the rest of HP-UX, an EFI system partition is simply a partition containing unspecified data. The EFI volume cannot be mounted to HP-UX currently.
The files in the EFI file system can have up to 255 characters, they are not case sensitive.
More detailed information can be found in the `efi(4)` manual page. An EFI command overview can be found at
http://hprtdt58.grc.hp.com/documents/systems/longspeak/itanium_handly_trifold.pdf (HP internal)

How to configure the Console input/output devices

On Itanium systems there are basically three possible console screens:

- VGA-terminal
- On board serial console port
- Management port (MP) *only available for server, not on workstations*

From the EFI Boot Maintenace Manager menu you can enable or disable these ports:

```
EFI Boot Maintenance Manager ver 1.10 [14.61]

Main Menu. Select an Operation


        Boot from a File
        Add a Boot Option
        Delete Boot Option(s)
        Change Boot Order

        Manage BootNext setting
        Set Auto Boot TimeOut

        Select Active Console Output Devices
        Select Active Console Input Devices
        Select Active Standard Error Devices

        Cold Reset
        Exit
```

In this example all three devices are enabled. The asterix indicates this.

```
Select the Console Output Device(s)


      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(PcAnsi)
      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(Vt100)
    * Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(Vt100+)
      Acpi(PNP0501,0)/Uart(9600 N81)/VenMsg(VtUtf8)
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(PcAnsi)
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(Vt100)
    * Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(Vt100+)
      Acpi(HWP0002,700)/Pci(1|1)/Uart(9600 N81)/VenMsg(VtUtf8)
    * Acpi(HWP0002,700)/Pci(2|0)
    Save Settings to NVRAM
    Exit
```

Serial port

MP port

VGA port

KMINE doc IA64KBRC00011702 provides tables that show which of the devices display the output of:

- power on self test (POST)
- EFI menus
- HP-UX system bootup messages (i.e. VG activation, startup-scripts, login prompt)

depending on the different combinations of selected console devices.

Here is the example for the rx2600 server:

Console Port	Console input	Console output	POST messages	EFI menu	Boot messages
Serial Port	*none*	*none*	no	yes	no
Management Port (MP)	*none*	*none*	yes	yes	yes
VGA monitor	*none*	*none*	no	yes	no
Serial Port	*none*	*none*	no	no	no
Management Port (MP)	*none*	*none*	yes	yes	no
VGA monitor	*	*	no	yes	yes
Serial Port	*	*	yes	yes	yes
Management Port (MP)	*none*	*none*	yes	yes	no
VGA monitor	*none*	*none*	no	no	no
Serial Port	*none*	*none*	no	no	no
Management Port (MP)	*	*	yes	yes	yes
VGA monitor	*none*	*none*	no	no	no
Serial Port	*	*	yes	yes	no
Management Port (MP)	*	*	yes	yes	yes
VGA monitor	*none*	*none*	no	no	no
Serial Port	*	*	yes	yes	no
Management Port (MP)	*	*	yes	yes	yes
VGA monitor	*	*	no	yes	no

This can be summarized as follows:

- If you select no console input & output devices, EFI automatically selects them all.
- The VGA monitor does never show POST output.
- Even if the is no shell login on VGA you will always get a CDE login prompt presented in case the system is equipped with a supported graphics card.
- Boot messages are only printed to one single output device. If multiple devices are enabled the priority is: 1. MP port, 2. Serial port, 3. VGA monitor.
  So if you like to have boot messages on VGA you need to enable the VGA port only.
  EFI's console device settings can be overwritten by specifying the `vga` option at the boot loader:

  ```
  HPUX> boot -vga
  ```

  Now boot messages go to the VGA monitor. The MP port cannot be used after this, only on-board serial port.

NOTE (UX 11.23): This all seems to have changed in UX 11.23. It looks like outpu is going to all devices.

The boot process

The following section describes the boot process in detail.

NOTE: I recommed to use a vt100 compatible terminal type if you logon to the Management port console (former GSP). After powering on the system will perform the Power On Self Tests (POST) and then launch the EFI Boot Manager:

```
EFI Boot Manager ver 1.10 [14.60]  Firmware ver 2.21 [4306]

Please select a boot option

    EFI Shell [Build-in]
    HP-UX 11.22
    Red Hat Linux Advanced Server
    Microsoft Windows
    Boot option maintenance menu
    Security/Password Menu


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

In this case boot option `HP-UX 11.22` is configured. Selecting this will start the OS Loader - continue here.
If no boot option is configured yet you may select `EFI Shell [Build-in]`. The device mapping table will be displayed and the EFI shell script `STARTUP.NSH` will be executed:

```
Loading.: EFI Shell [Build-in]
EFI Shell version 1.10 [14.60]
Device mapping table
  fs0  : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig2B7F9A68-32D0-44...
  fs1  : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig183C0EF2-DBE4-41...
  fs2  : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun2,Lun0)/HD(Part1,Sig4F580000)
  fs3  : Acpi(HWP0002,100)/Pci(1|0)/Pci(1|1)/Scsi(Pun0,Lun0)/HD(Part1,Sig01947...
  blk0 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)
  blk1 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig2B7F9A68-32D0-44...
  blk2 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part2,Sig183C0EF2-DBE4-41...
  blk3 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part3,Sig8EDBCFDA-2916-11...
  blk4 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun0,Lun0)/HD(Part4,Sig8EDC2FC0-2916-11...
  blk5 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun2,Lun0)
  blk6 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun2,Lun0)/HD(Part1,Sig4F580000)
  blk7 : Acpi(HWP0002,0)/Pci(2|0)/Scsi(Pun2,Lun0)/HD(Part2,Sig4F580000)
  blk8 : Acpi(HWP0002,0)/Pci(2|1)/Scsi(Pun3,Lun0)
  blk9 : Acpi(HWP0002,100)/Pci(1|0)/Pci(1|1)/Scsi(Pun0,Lun0)
  blkA : Acpi(HWP0002,100)/Pci(1|0)/Pci(1|1)/Scsi(Pun0,Lun0)/HD(Part1,Sig01947...
  blkB : Acpi(HWP0002,100)/Pci(1|0)/Pci(1|1)/Scsi(Pun0,Lun0)/HD(Part2,Sig0194C...
  blkC : Acpi(HWP0002,100)/Pci(1|0)/Pci(1|1)/Scsi(Pun0,Lun0)/HD(Part3,Sig01951...

startup.nsh> echo -off

Welcome to HP-UX for IA64
  setting hpux path(\EFI\HPUX)...
  type 'fs[x]:' where x is your bootdisk (0, 1, 2...)
  type 'hpux' to start hpux bootloader
Shell>
```

NOTE: You may enter `help` at the shell prompt:

```
Shell> help
List of classes of commands:

boot          -- Booting options and disk-related commands
configuration -- Changing and retrieving system information
device        -- Getting device, driver and handle information
memory        -- Memory related commands
shell         -- Basic shell navigation and customization
scripts       -- EFI shell-script commands

Use 'help <class>' for a list of commands in that class
Use 'help <command>' for full documentation of a command
Use 'help -a' to display list of all commands

Shell>
```

If you do not know which fs represents your HP-UX boot disk just try all of them and search for the HP-UX loader:

```
Shell> fs2:

fs2:\> ls
Directory of: fs2:\

  10/28/02  11:10p <DIR>            512  EFI
  10/28/02  11:10p                  336  STARTUP.NSH
         1 File(s)         336 bytes
         1 Dir(s)

fs2:\> type startup.nsh
File: fs2:\startup.nsh, Size 336

#
# Copyright (c) 2000  Hewlett Packard Corporation
#
#  HP-UX EFI start-up shell script
#
echo -off
echo ""
echo "Welcome to HP-UX for IA64"
echo "  setting hpux path(\EFI\HPUX)..."
set path ".;\efi\hpux;\efi\tools;"
echo "  type 'fs[x]:' where x is your bootdisk (0, 1, 2...)"
echo "  type 'hpux' to start hpux bootloader"
echo -on

fs2:\> cd efi

fs2:\EFI> ls
Directory of: fs2:\EFI

  10/28/02  11:10p <DIR>            512  .
  10/28/02  11:10p <DIR>              0  ..
  10/28/02  11:10p <DIR>            512  HPUX
  10/28/02  11:10p <DIR>            512  Intel_Firmware
  10/28/02  11:10p <DIR>            512  DIAG
  10/28/02  11:10p <DIR>            512  HP
  10/28/02  11:10p <DIR>            512  TOOLS
  11/19/02  11:22a <DIR>            512  FC
         0 File(s)           0 bytes
         8 Dir(s)

fs2:\EFI> cd hpux

fs2:\EFI\HPUX> ls
```

```
Directory of: fs2:\EFI\HPUX

  10/28/02  11:10p <DIR>            512  .
  10/28/02  11:10p <DIR>            512  ..
  10/28/02  11:10p              419,545  HPUX.EFI
  10/28/02  11:10p               24,576  NBP.EFI
  10/28/02  11:10p                   12  AUTO
          3 File(s)     444,133 bytes
          2 Dir(s)


fs2:\EFI\HPUX> type auto
File: fs2:\EFI\HPUX\auto, Size 12

boot vmunix

fs2:\EFI\HPUX>
```

yes. This looks like the HP-UX boot disk.

Switch from the EFI shell to fs2:

```
Shell> fs2:

fs2:\>
```

From here you are able to switch to any other boot disk that has an EFI file system, e.g to switch to fs1 type `fs1:`. In order to go back to the EFI boot manager menu type `exit`.

To continue, invoke the HP-UX boot loader. It's an EFI executable (`hpux.efi`). EFI executables (`.efi`) and EFI shell scripts (`.nsh`) can be invoked without specifying the appendix. This is DOS like ;-)

```
fs2:\> hpux

(c) Copyright 1990-2002, Hewlett Packard Company.
All rights reserved

HP-UX Boot Loader for IA64  Revision 1.713

Press Any Key to interrupt Autoboot
\efi\hpux\AUTO ==> boot vmunix
Seconds left till autoboot -   3
```

Pressing any key at this point interrupts the 10 seconds autoboot timeout and invokes the OS Loader hpux.efi in interactive mode. At the `HPUX>` prompt you have several possibilities to interact, such as

Changing boot options:

`HPUX> boot -is`	for single user mode
`HPUX> boot -lq`	for LVM quorum mode
`HPUX> boot -lm`	for maintenance mode boot
`HPUX> boot vmunix.prev`	for booting previous kernel
`HPUX> boot -vga`	redirect console output (e.g. rc scripts) to VGA monitor.

Some of the options may be combined.

NOTE: At UX 11.22 you need to have PHKL_28787 installed to be able to boot to maintenance mode.

Getting help:

```
HPUX> help
 -- HPUX bootloader for IA64 Help --
  list of supported commands:

   boot [kernel] - boots HPUX kernel
   help [-d]     - help screen (-d to list debug commands)
   ls [-b] [dir]      - lists directory (-b for screen break)
   ll [-b] [dir]      - lists directory in long detail
   mmap          - show current EFI memory map
   setauto [-d] [str] - sets AUTO file (-d to delete AUTO file)
   showauto      - shows AUTO file
   ver           - prints version numbers
   exit          - exits bootloader
```

Gathering firmware versions:

```
HPUX> ver
Bootloader Version => 1.713
EFI Specification Revision => 1.10
Firmware Vendor  => HP
Firmware Revision=> 14.60
CPU Revision => 7
```

Displaying the content of the AUTO file:

```
HPUX> showauto
\efi\hpux\AUTO => boot vmunix
```

Listing the content of the boot directory /stand:

```
HPUX> ll
FILENAME                                SIZE   DATE(UTC)
.
..
.kminstall_lock                            0   2002/11/13
.kmmodreg_lock                             0   2002/11/13
.kmsystune_lock                            0   2002/08/09
boot.3D5429A80B4D/                             2002/08/09
boot.3DBE377F0857/                             2002/10/29
boot.sys/                                      2002/08/09
bootconf                                  21   2002/10/29
build/                                         2003/02/21
dlkm(LINK)                                24   2002/10/29
dlkm.3D5429A80B4D/                             2002/08/09
dlkm.3DBE377F0857/                             2003/02/21
ioconfig                               3,076   2003/02/21
kernrel                                   82   2002/08/09
krs/                                           2003/02/21
krs_lkg/                                       2003/02/21
krs_tmp/                                       2003/02/21
lost+found/                                    2002/10/29
rootconf                                  12   2003/02/21
system                                 1,041   2002/10/29
system.d/                                      2002/11/13
vmunix                            39,775,528   2002/10/29
```

If autoboot timeout was not interrupted the OS loader will be launched with the options as specified in the `AUTO` file and try to load the kernel:

```
AUTOBOOTING...
AUTO BOOT> boot vmunix
> System Memory = 2041 MB
loading section 0 ..................................... (complete)
loading section 1 ...... (complete)
loading symbol table
loading System Directory(boot.sys) to MFS ......
loading Kernel Boot Directory(boot.3DBE377F0857) to MFS. .....
Launching /stand/vmunix
SIZE: Text:19787K + Data:2610K + BSS:2381K = Total:24779K
```

If you are sitting in front of the VGA monitor and your system seems to hang at this stage the console output may not be configured properly. The "`SIZE:`" message above is printed by HP-UX bootloader who prints console output to all available devices. The subsequent message "`Console is on ...`" is printed by the HP-UX kernel's funtion `printf()`. This goes only to the configured console path. Refer to the section above in order to configure the console for VGA.

If boot messages are set up to be directed to your current console device, the boot processes will continue as follows:

```
Console is on a Serial Device
Booting kernel...
```

Here it takes some time, then the normal boot process as described in the Boot chapter continues until you get the Console login:

```
Loaded ACPI revision 2.0 tables.
NOTICE: nfs3_link(): File system was registered at index 4.
NOTICE: autofs_link(): File system was registered at index 5.
NOTICE: cachefs_link(): File system was registered at index 6.
td: claimed Tachyon XL2 Fibre Channel Mass Storage card at 0/7/1/0
Boot device's HP-UX HW path is: 0.0.2.0.2.0
legacyio_cdio_end: WARNING isc for function 0 not found!!!

    System Console is on the Built-In Serial Interface
igelan0: APA NOT SUPPORTED on HP A6794-60001 PCI 1000Base-T at hardware path
0/1/1/0/4/0
igelan0: INITIALIZING HP A6794-60001 PCI 1000Base-T at hardware path 0/1/1/0/4/0
Entering cifs_init...
Initialization finished successfully... slot is 10
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
    Swap device table:  (start & size given in 512-byte blocks)
        entry 0 - major is 64, minor is 0x2; start = 0, size = 8388608
Starting the STREAMS daemons-phase 1
Checking root file system.
file system is clean - log replay is not required
Root check done.
....... OK

...
...
...
...
```

```
The system is ready.

GenericSysName [HP Release B.11.22] (see /etc/issue)
Console Login:
```

How to mirror the root disk

The following procedure shows how to mirror the root disk. Let c3t2d0 be the existing primary disk and c2t1d0 the new mirror boot disk:

1) Setup the disk partitions

   At a cold-installed UX 11.23 system the partition sizes are different compared to UX 11.22. Use idisk(1M) to check the partition sizes. E.g. for a UX 11.23 system you would get:

```
# diskinfo /dev/rdsk/c0t0d0s1 | grep size
                size: 512000 Kbytes

# diskinfo /dev/rdsk/c0t0d0s3 | grep size
                size: 409600 Kbytes
```

   Create a partition description file:

at UX 11.22:	at UX 11.23 (cold-installed)

`# vi /tmp/partitionfile`
`2`
`EFI 100MB`
`HPUX 100%`
	`# vi /tmp/partitionfile`
	`3`
	`EFI 500MB`
	`HPUX 100%`
	`HPSP 400MB`

   Use idisk(1M) command to partition the disk according to this file:

```
# idisk -wf /tmp/partitionfile /dev/rdsk/c2t1d0
idisk version: 1.2
********************** WARNING **********************
If you continue you may destroy all data on this disk.
Do you wish to continue(yes/no)? yes
...
```

2) Create the new device files for the new partitions (c2t1d0s1, s2, (s3))

```
# insf -e -Cdisk
```

3) Use efi_fsinit(1M) to initialize the FAT filesystem on the EFI partition:

```
# efi_fsinit -d /dev/rdsk/c2t1d0s1
```

   NOTE: This step is not neccessary if it can be guaranteed that the mirror disk does not contain a valid EFI filesystem. In this case efi_fsinit(1M) will be done automatically by the subsequent mkboot(1M) command. But if you take e.g. an old UX 11.22 boot disk as mirror disk, mkboot will not automatically run efi_fsinit. As a result only 100MB of the 500MB EFI partition (s1) can be used.

4) Use mkboot(1M) to format the EFI partition (s1) and populate it with the EFI files below `/usr/lib/efi/` and to format the LIF volume (part of s2) and populate it with the LIF files (ISL, AUTO, HPUX, LABEL) below `/usr/lib/uxbootlf`:

```
# mkboot -e -l /dev/rdsk/c2t1d0
# efi_ls -d /dev/rdsk/c2t1d0s1                    (to check EFI)
FileName                              Last Modified             Size
EFI/                                   11/ 5/2003                 0
STARTUP.NSH                            11/ 5/2003               296

total space 523251712 bytes, free space 520073216 bytes

# lifls -l /dev/rdsk/c2t1d0s2                     (to check LIF)
```

5) Check the content of AUTO file on EFI partition:

```
# efi_cp -d /dev/rdsk/c2t1d0s1 -u /EFI/HPUX/AUTO /tmp/x; cat /tmp/x
```

NOTE: Specify the -lq option if prefer that your system boots up without interruption in case of a disk failure:

```
# mkboot -a "boot vmunix -lq" /dev/rdsk/c2t1d0
# mkboot -a "boot vmunix -lq" /dev/rdsk/c3t2d0
```

6) Copy the HP service partition (UX 11.23 only):
(skip this, if you don't have a service partition)

```
# dd if=/dev/rdsk/c3t2d0s3 of=/dev/rdsk/c2t1d0s3 bs=1024k
```

7a) LVM only:

- Initialize the LVM partition (s2) and add it to vg00:

```
# pvcreate [-f] –B /dev/rdsk/c2t1d0s2      (take care to use s2)
# vgextend vg00 /dev/dsk/c2t1d0s2
```

- Mirror the LVs to the s2 partition:

```
# for i in lvol1 lvol2 ... lvol8                    (specify each LV)
> do lvextend -m 1 /dev/vg00/$i /dev/dsk/c2t1d0s2
> done
```

- Check if content of LABEL file (i.e. root, boot, swap and dump device definition) has been initialized (done by lvextend) on the mirror disk:

```
# lvlnboot -v
Boot Definitions for Volume Group /dev/vg00:
Physical Volumes belonging in Root Volume Group:
        /dev/dsk/c3t2d0s2 (0/1/1/1.2.0) -- Boot Disk
        /dev/dsk/c2t1d0s2 (0/1/1/0.1.0) -- Boot Disk
Boot: lvol1     on:     /dev/dsk/c3t2d0s2
                        /dev/dsk/c2t1d0s2
Root: lvol3     on:     /dev/dsk/c3t2d0s2
                        /dev/dsk/c2t1d0s2
Swap: lvol2     on:     /dev/dsk/c3t2d0s2
                        /dev/dsk/c2t1d0s2
Dump: lvol2     on:     /dev/dsk/c3t2d0s2, 0
```

If not, then set it:

```
# lvlnboot -r /dev/vg00/lvol3
# lvlnboot -b /dev/vg00/lvol1
# lvlnboot -s /dev/vg00/lvol2
# lvlnboot -d /dev/vg00/lvol2
```

7b) VxVM only:

- Have VxVM see the new c2t1d0s2 disk:
  `# vxdctl enable`
- Confirm that you can see the new disk as “online invalid”:
  `# vxdisk list`
- Mirror the root disk. This may take some time:
  `# /etc/vx/bin/vxrootmir -v c2t1d0s2`
- Verify that all volumes are mirrored:
  `# vxprint -g rootdg`

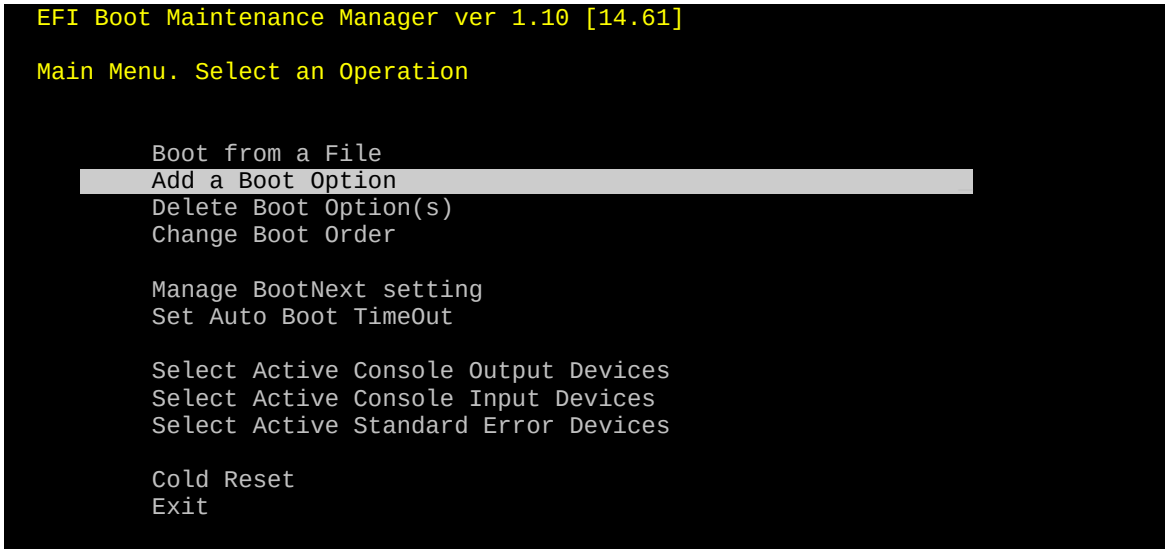
8) Specify the mirrored disk as alternate bootpath

```
# setboot –a <HW path of mirror>
# setboot                                   (to check it)
```

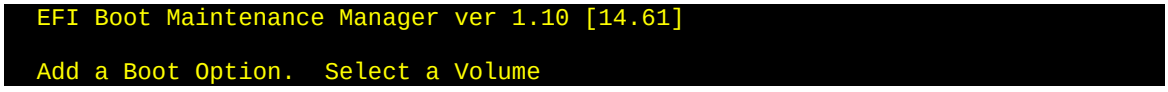
9) Create an EFI boot option (**UX 11.22 only**; done by setboot for UX 11.23)

Finally it is a good idea to set a EFI boot option for the mirror disk. This boot option will be stored in NVRAM:

Boot to EFI and enter the Boot maintenance manager menu:



Select the mirror disk (Pun1|Lun0) represents SCSI target 1, Lun 0 (c2t1d0) in our case. (Pun2,Lun0) is the original boot disk (c3t2d0)



```
      IA64_EFI [Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun1,Lun0)/HD
(Part1,SigB45A0000)
      IA64_EFI [Acpi(HWP0002,100)/Pci(1|1)/Scsi(Pun2,Lun0)/HD
(Part1,Sig958B0000)
      EFI DISK [Acpi(HWP0002,600)/Pci(1|0)/Pci(0|0)/Pci(0|0)/Pci(0|0)/
Scsi(Pun0,Lun0)/HD(Part1,Sig119E1A60-0B4C-01C3-507B-9E5F8078F531)
      Removable Media Boot [Acpi(HWP0002,0)/Pci(2|0)/Ata(Primary,Master
      Load File [EFI Shell [Built-in]]
      Load File [Acpi(HWP0002,0)/Pci(3|0)/Mac(00306E3809C6)]
      Load File [Acpi(HWP0002,100)/Pci(2|0)/Mac(00306E3889E3)]
      Exit
```

Now navigate to the HP-UX bootloader HPUX.EFI on the disk:

```
  EFI Boot Maintenance Manager ver 1.10 [14.61]

  Select file or change to new directory:

      05/28/03  09:38a <DIR>         512 EFI                          _
      [Treat like Removable Media Boot]
     Exit
```

```
  Select file or change to new directory:

        05/28/03  09:38a <DIR>         512 .
        05/28/03  09:38a <DIR>           0 ..
      05/28/03  09:38a <DIR>         512 HPUX                         _
        05/28/03  09:38a <DIR>         512 Intel_Firmware
        05/28/03  09:38a <DIR>         512 DIAG
        05/28/03  09:38a <DIR>         512 HP
        05/28/03  09:38a <DIR>         512 TOOLS
     Exit
```

```
  Select file or change to new directory:

        05/28/03  09:38a <DIR>         512 .
        05/28/03  09:38a <DIR>         512 ..
      05/28/03  11:52a           419,545 HPUX.EFI
        05/28/03  11:52a            24,576 NBP.EFI
     Exit


     Filename: \EFI\HPUX\HPUX.EFI

DevicePath:[Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun1,Lun0)/HD(Part1,SigB45A0000)/\EF
I\HPUX\HPUX.EFI]
      IA-64 EFI Application 05/28/03  11:52a     419,545 bytes

     BootFFFF:
Acpi(HWP0002,100)/Pci(1|0)/Scsi(Pun1,Lun0)/HD(Part1,SigB45A0000)/\EFI\HPUX\HPUX.
EFI
```

Now enter a description for this boot option, e.g. *HP-UX mirror boot disk*:

```
   Enter Description:  HP-UX mirror boot disk


   Current BootOption-->Main Menu. Select an Operation
   New BootOption Data. ASCII/Unicode strings only, with max of 240 characters
   Enter BootOption Data Type [A-Ascii U-Unicode N-No BootOption] : N
```

```
```

Finally save the setting to NVRAM:

```
Save changes to NVRAM [Y-Yes N-No]:  Y
```

10) Try to boot from the mirror disk by choosing the appropriate boot option.

How to replace a failed disk

There is no difference between PA and IPF systems when talking about replacing *non-root* disks. Even for root disks the procedure is quite similar, so for details refer to the LVM chapter. The partitioned disk layout and the existance of EFI needs to be considered though. Hence there are three major differences compared to the usual “hot-swap” procedure.

1) Partition the new disk

   You have to run idisk(1M) first before performing mkboot(1M), lvlnboot(1M) and vgcfgrestore(1M) in order to partition the new disk. Note that vgcfgrestore(1M) does not restore EFI or LIF. This is done by mkboot(1M).
   idisk(1M) usage is explained in step 1) of the mirror procedure above.

2) Recreate EFI boot option (**UX 11.22 only**; not neccessary for UX 11.23)

   idisk(1M) creates a new disk identifier (GUID) on the disk. If an EFI boot option entry (held in NVRAM) exists for the replaced disk it is still linked to the old GUID. Hence you have to remove the old boot option entry and add a new one.
   Refer to step 9) of the mirror procedure above.

3) Be careful to specify the `s2` device file (`c#t#d#s2`) when performing LVM commands.

The ARIES emulator

Aries is the HP-UX PA-RISC to HP-UX IPF binary emulator or *dynamic code translator*. Aries transparently emulates both 32-bit and 64-bit HP-UX PA-RISC 2.0 applications on HP-UX IPF, thus providing binary compatibility for PA-RISC binaries on IPF systems. It is transparent in that users need not do anything special to simply run their PA programs on IPF.

The HP-UX/IPF kernel (exec(2)) will recognize PA executable files (both 32- and 64-bit fields) and invoke Aries to emulate these processes. Aries will handle all instruction set architecture (ISA) emulation and enviournment emulation on behalf of the PA program, and present a virtual PA machine achitecture to the emulated program.
There are actually two versions of Aries: one for emulating PA-32 processes, and one for PA-64 processes. This is necessary because Aries must keep the address space layout that the emulated process expects; natively, then Aries-32 is a 32-bit process, and Aries-64 is a 64-bit process.

Because an emulator can never be as performant as native code Aries should be used where performance is not critical or where it is not possible to create a native IPF family binary.

If you intend to run PA binaries on IPF it is strongly recommended to apply the latest available Aries patch:

UX 11.20: PHSS_24463 or newer
UX 11.22: PHSS_28827 or newer

Further information about Aries can be found on the websites mentioned in the Additional information section.

IPF vs. PA-RISC Terminology

PA term	IPF term
Chassis Code	Event ID
Activity Log	Forward Progress Log (FPL)
BCH	Extensible Firmware Interface (EFI)
HPMC	Machine Check Abort (MCA)
Errorlog	System Event Log (SEL)
TOC	Non-Maskable Interrupt (NMI) or INIT
LPMC (corrected by CPU) e.g. Level 2 Cache Single Bit Error (SBE)	Correctable Machine Check (CMC)
LPMC (corrected by CEC) e.g. Memory SBE	Correctable Platform Error (CPE)

NOTE: CEC (Core Electronic Complex) is the chipset (zx1/sx1000).

Additional Information

General

HP's Itanium homepage
http://hp.com/go/itanium

Intel's Itanium homepage
http://developer.intel.com/design/itanium/family/ (non HP)

Itanium website by WTEC
http://wtec.cup.hp.com/~hpux/docs/ia64 (HP internal)

IA-64 Virtual Competency Center EMEA
http://hpux.uksr.hp.com/ceasst/ia64/ (HP internal)

Itanium Center of Expertise (Canada)
http://ia64.canada.hp.com/ (HP internal)

UNIX Support Knowledge Team website
(cntains Itanium related training material such as future HP-UX releases, EFI, ...)
http://cso.fc.hp.com/ssil/uxsk/hpux (HP internal)

Application availability (ISV site)
http://www.hp.com/products1/itanium/partners/isv.html

HP-UX 11i version 1.6 support plan
http://wwwpsp.atl.hp.com/lmx_mount/supplan/psp/12/psp12606.htm

New features with Madison systems
http://bcstraining.corp.hp.com/madison/tech/os/ (HP internal)

IPF Hardware Handbook (great!):
http://www.grc.hp.com/docs/wkstcc/ia64/products/technical_info/handbook.html

Software Development

IA-64 Transition Kit
http://www.software.hp.com/products/IA64

Software Transition Kit (STK)
http://devrsrc1.external.hp.com/STK/

EFI

Intel's EFI homepage:
http://developer.intel.com/technology/efi/

Hands on EFI training:
http://cso.fc.hp.com/ssil/uxsk/hpux/hpux_releases/11.22

EFI command overview (includes also POSSE, MP, BMC/CLI commands):
http://hprtdt58.grc.hp.com/documents/systems/longspeak/itanium_handly_trifold.pdf (HP internal)

Search http://docs.hp.com/ with the keyword "EFI" for further documents.

Performance

SPEC Performance Benchmarks
http://www.spec.org (non HP)

Performance Tuning Guide for Itanium systems
http://www.hp.com/products1/unix/operating/infolibrary/whitepapers/7206_IPF_tuning_wp_051403.pdf

UX 11.22 at docs.hp.com
http://docs.hp.com/hpux/os/11iV1.6

Aries

Aries homepage
http://cllweb.cup.hp.com/migration/training (HP internal)

Aries official page
http://devrsrc1.external.hp.com/STK/Aries.html