The State of Play for Linux 2.6 on PA-RISC

James Bottomley
Product Architect
SteelEye Technology, Inc.

HP WORLD 2003
Solutions and Technology Conference & Expo

Meet Lux, the palinux Mascot

- PA-RISC linux is still “Under Construction”.
- But then---so is Linux.
- This talk will cover:
  - PA-Linux
  - The new features in 2.6.
  - The stability of palinux. (or, should I trust my data to it).
  - The missing features.
  - What can I do to help?

Introduction

- PA-RISC is the proprietary HP RISC processor architecture originally designed for HP-UX.
- Project to run Linux on these machines started in October 1998.
- Linux first booted on PA-RISC on 25 June 1999.
- The PA-RISC kernel is now integrated into both the 2.4 and 2.6 mainline (a small patch set is still required for each).
- Two linux distributions now support PA-RISC: Debian (woody 3.0, released) and Gentoo (experimental).
- PA-RISC linux still not “officially” supported by HP.

What is palinux

- Palinux describes the project for getting Linux running on the PA-RISC processor. There are two elements:
  - The kernel which supports the hardware
  - And the user applications and booting system
- The kernel comes in two flavours: 32 bit and 64 bit.
  - The older machines (PA 7xxx processors) only run the 32 bit kernel
  - Newer machines (PA8xxx processors) may run either 32 or 64 bit kernels.
  - Some machines (depending on the PDC firmware) may only run the 64 bit kernel.
- The user land is currently only 32 bit ELF (with some support for HP-UX 32 bit SOM).

Tool Chain

- In order to build applications you need:
  - A compiler: gcc has build parisc-linux targets since version 3.x
  - A linker: GNU binutils supplies the linker, assembler and various binary utilities
  - A library: again GNU libc supplies this
- These elements are called the “tool chain”.
- The palinux tool chain is somewhat temperamental. However, palinux builds are now self hosting (the installed system can compile and build itself)
- The foundation for all of this is a document called the processor specific ELF supplement.

The ELF Supplement

- This defines the processor specific parts of the Linker format.
- There's no defined standard for this (although the Linux Standards Base is thinking about it).
- PA-RISC ELF32 is well defined.
- PA-RISC ELF64 isn't
  - We have enough of the definition to build and link a 64 bit kernel.
  - We don't have enough for a 64 bit user space (particularly all the pieces that make shared libraries function).
  - The problem is mainly that no-one has written the missing pieces.

What's new in 2.6

- The primary, user visible, changes are:
  - I/O Subsystem improvements:
    - Block I/O is completely renovated.
    - SCSI and IDE redone.
    - Arbitrary memory limits removed for “zero copy”.
    - Large sector (64 bit offset even on 32 bit platform)
  - VM Subsystem improvements
    - VM Overcommit detection prevent system promising resources it can't deliver
    - Swap partitions may be up to 2GB.
  - Scheduler updates
    - Algorithm rewritten for stability under load
    - New Native Posix Threading Library

What's new in 2.6

- Changes with less user visibility are
  - New device model
    - Unified device tree abstracts system view from bus type.
    - Provides common semantics for all probeable buses.
    - Control now done via sysfs filesystem (/sys).
    - Hotplug built into the model.
  - Re-written kernel build system
    - Dependencies are now computed on the fly
    - Make configure is fast
  - New module loader
    - We now have an in-kernel ELF linker

Palinux and 2.6

- A large amount of work had to be done simply to make palinux compile in 2.6
  - A new module loader had to be written
  - The VM and Scheduler updates affect the processor specific areas quite a bit
  - Many changes are done for x86 alone with no regard for other architectures. A lot of work has to be done just sweeping up the pieces behind this.
- The above means a lot of ongoing work just to keep it compiling too.
- In spite of this there has been some work done implementing new features in palinux for 2.6

What's Interesting about PA (from a hacker's point of view)

- From a kernel developer's standpoint, there are several interesting features about the PA-RISC architecture:
  - It's not fully coherent: This makes it a good candidate for testing the DMA API.
  - Some CPUs (PA7100) cannot even make coherent memory for dma_alloc_coherent.
  - Some machines have a very complex bus architecture (EISA/GSC/LASI/PCI all in one box). This makes palinux an ideal study ground for the generic device model.
  - It's the only Linux architecture to be able to compile 64 and 32 bit kernels out of the same tree.
  - It has an upward growing stack (all other architectures' stacks grow down).

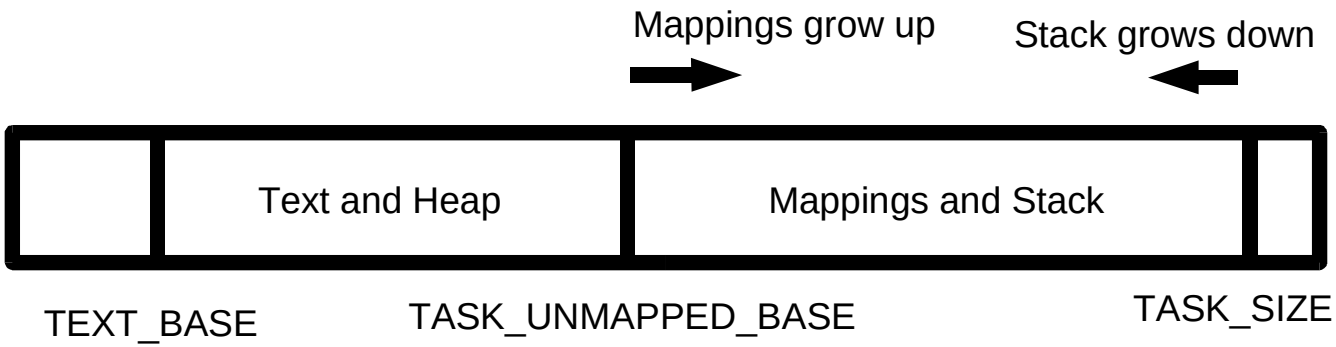
New palinux features in 2.6

- Stack layout finally looks similar to the other architectures for user executables.
- Native consumer of the new generic device DMA API
  - Unifies the bus view of the entire system.
  - Has a special API for fully non-coherent drivers (i.e. drivers that must work on the 715 series of machines).
  - Palinux provided the testbed for getting this API into the vanilla linux kernel.
- Fully integrated into the new kernel subsystem init code
- Hopefully, runtime detection of Narrow PDC in 64 bit kernel.

Upward Growing Stack

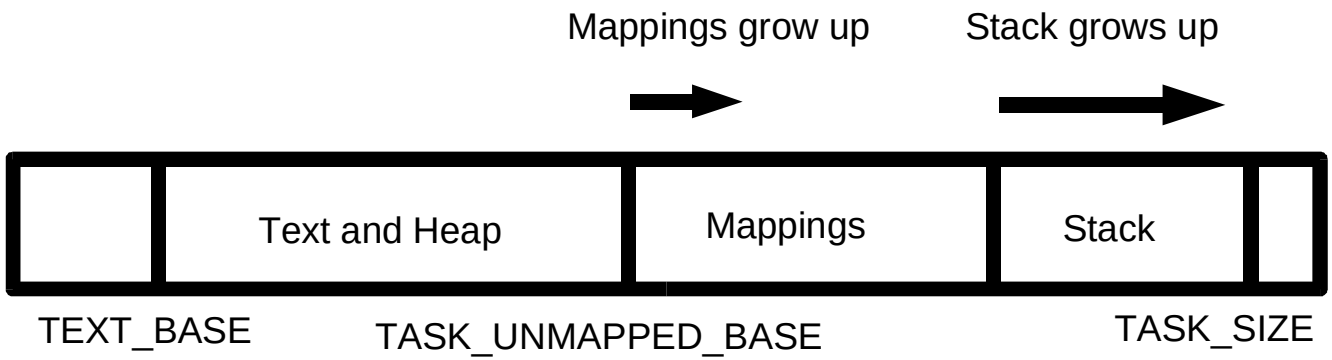
- The PA processor doesn't actually have a stack. However, the compiler improvises one.
- HP-UX had an upwards growing stack, so gcc uses this convention too.
- Upwards growing stacks provide security:
  - If the stack grows down, a buffer overrun expands into the caller's stack frame
  - If there were code there, the overrun may replace this and gain control of the applications
  - A buffer overrun in an upwards growing stack just runs off the end of the stack and can only damage the existing stack frame (it cannot extend into the caller's stack frame)

Traditional Stack Layout



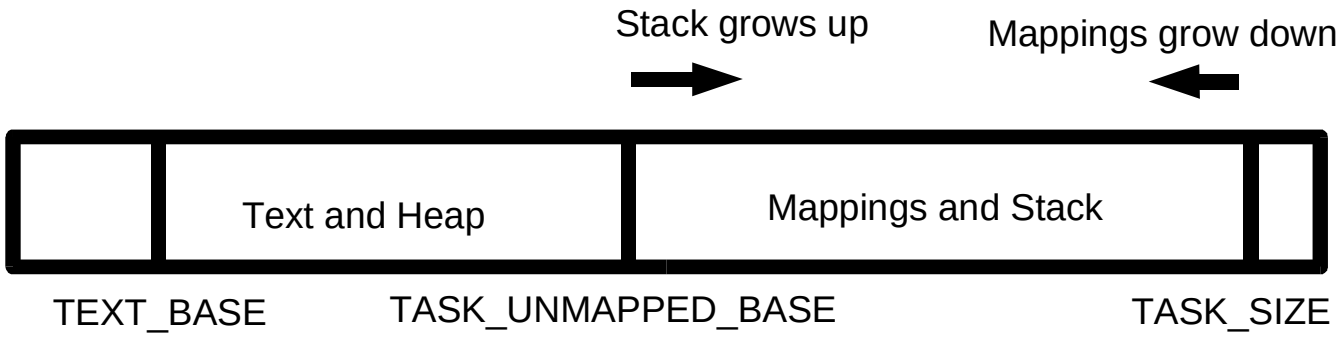
- Traditional Flat ELF binary layout
- Stack and Mappings share area
- Because mappings grow up and stack grows down, the space is optimally utilised (problems only occur when they meet).

Palinux Current Stack Layout



- For an upward growing stack, the base of the stack must be sized by the stack's ulimit
- Large stacks may cut into the area reserved for mappings (which is where shared libraries go)
- Layout requires users to be careful with ulimit -s (most of us blithely ignore this setting).

Alternative Flat Layout



- Well, if the stack layout is going to be strange, why not the mappings layout as well.
- This ELF layout is under consideration for 2.6; it has exactly the same properties now as the corresponding ELF layouts for all other architectures
- Having mappings grow down may be an issue.

The 2.4 DMA API

- Originally, in 2.4, a new DMA mapping API was introduced.
  - This was required for architectures which have MMUs on the I/O bus (basically everything but x86)
  - Was also required for architectures that are not “coherent” (which means the driver has to manage the processor cache to make sure it has the most current data from the device)
- Envisaged as a per-bus type API
  - PCI bus has `pci_dma_...`
  - SBUS has `sbus_dma_...`

The problems this causes

- Some PA-RISC boxes have five or more bus types (C360 has LASI, GSC, PCI, EISA and Runway).
- Having an API for each would be unwieldy.
- Also, the buses depend on each other.
  - For example: The PCI bus in the C360 is parented to the Cujo/Dino device on the GSC bus and needs to control the IOMMU that sits between Cujo/Dino and Runway.
- Worse, some drivers (notably the 53c710 driver for the LASI SCSI chip) drive multiple buses (MCA, LASI, EISA) but don't want to be bothered with the details of which they're currently connected to.
- All of this confusion is solved by a simple 2.5 addition...

The Generic Device Model

- In this model, all devices embed the same structure (`struct device`).
- These device structures form a view of the entire bus tree.
- How the platform DMA model interacts with the bus tree can now be defined per architecture
  - This is ideal for PA-RISC where more than one bus type may need to interact to set up an IOMMU transaction.
- Bus type now becomes a property of the device structure, not vice-versa.
- PA-RISC was the first architecture to implement the new generic device DMA mapping scheme.

4GB/4GB Kernel User Split

- This is a big deal on x86, where the whole memory layout is split 3GB/1GB because the kernel must be invisibly resident.
- The problem is x86 only (induced by hugely expensive TLB cache flushing on the chip). However since x86 is the master implementation, it drives all others
- The PA-RISC processor doesn't suffer from the problem, so it is possible to have the kernel and the user space live in separate “spaces”.
- This Layout alleviates PA-RISC from doing all the “HIGHMEM” contortions beloved of x86
- One of the few architectures to do this---follow x86 is the easiest course of action for most.

What Still Needs to be Done for palinux

- As has been stated before, to be truly enterprise class (or at least in the same league as the hardware), we need to support 64 bit ELF user applications.
- SMP support is currently broken
  - We have all the pieces, and believe they could be put together correctly
  - The change to the init subsystem broke the early configuration scanning necessary for SMP
  - We could regain SMP but only at the expense of reverting our init changes (no simple task).
  - Alternatively, we could initialise SMP using the SMP hotplug API (unfortunately, this didn't make it into 2.6)

What Still Needs to be Done for palinux Continued

- Frame Buffers and Input Layers:
  - 2.6 revamped these completely, and it's been a struggle just to keep up.
  - Palinux uses the STI framebuffer for X (Which is required for any graphical desktop you might wish to run).
  - Older machines require the HIL keyboard/mouse interface, which is having teething problems with the new input layer.
  - Some of the old Visualize graphics cards just don't work.
- Bus problems: HP-PB
  - Due to lack of documentation, there is no support at all in Linux for the old Precision Bus.
  - If you have the doc's we'll (try) to make it work.

The Biggie: Is it Safe to Run 2.6?

- The answer to this is, of course, Subjective, so let me tell you what I do.
- Thanks to HP I have a C360 machine which always runs the latest 2.5/2.6 kernel (on debian Woody)
- I use this machine as the bitkeeper repository for the linux-scsi project.
- Since ironing out the last few of the glitches around the 2.5.49 timeframe, the machine has performed flawlessly.
- Since the whole of linux-scsi is dependent on this, you could say I'm trusting everyone else's data to it aswell.
- In summary, my answer is “yes”.

What Can I Do to Help

- Assuming I've convinced you that running 2.6 on PA-RISC will be safe (and fun)...
- If you want to get involved as a programmer, we need both compiler/library experts to help us sort out the 64 bit user land.
- If you have kernel coding experience and would like to look at some of the rough edges (like SMP), that would be extremely welcome.
- Even if you can't contribute as a programmer, you can help us simply by running a palinux distribution with the 2.6 kernel.
  - Try out different devices and drivers. Tell us what works and what you have problems with.

How Do I get involved

- The core resource for palinux is the project website:
  - www.parisc-linux.org
- The website hosts the archives for the mailing list as well as easy install instructions for obtaining installation CD's
- The site also contains:
  - A list of hardware known to boot palinux
  - Unfortunately, a list of hardware known not to boot palinux.
  - Some of the dos and don'ts when bringing palinux up for the first time.
  - It even contains prebuilt 2.6 kernel binaries.

Installing Palinux for the First Time

- OK, so you've decided you want to try it.
- The easiest way is to download the CD ISO images, burn a CD and install to a spare local disc drive
  - One of the latest distribution CD's is usually best (either Debian Woody or Gentoo)
  - Boot your PA box from the CD
  - Partition the local disc and just follow the (fairly automated installation menus)
  - Once you've installed and successfully booted palinux, you're ready to install a 2.6 kernel
  - Download one of the 2.6 kernel binaries, install it, and boot from it.

Booting from 2.6

- To install the 2.6 kernel, the binary kernel image needs to be installed in `/boot` (usually with a nice distinctive name like `/boot/vmlinux-2.6.0-pa1`)
- To boot this image, interrupt the boot sequence in the IODC (type ESC when told) and then
  - `boot pri ipl`
  - If the system asks if you want to interact with IPL, type Y
  - When palo halts, it gives you a boot line to edit. Choose 0 (the default choice)
  - Backspace (with Cntrl-H) over the kernel name and type your 2.6 kernel name (e.g. `2/vmlinux-2.6.0-pa1`)
  - Press Enter
  - Backspace over the 0 and enter b to Boot.

Booting 2.6 Continued

- The system should boot up on the 2.6 kernel you just installed after this.
- If you need to use modules, you need an updated set of module tools (available from <insert URL>)
- If everything went according to plan, the system should come up and you should notice no difference from the 2.4 kernel you got from the installation
- If you like it, make it permanent
  - Edit `/etc/palo.conf`
  - Replace the 2.4 kernel path with your 2.6 one
  - Now run `/sbin/palo`
  - That's it, the system will permanently boot 2.6

What If You Don't Have a Spare Disc

- Don't dispair, there's also a set of unofficial “net boot” ISO images which allow the machine to boot discless from an NFS server (which may be a HP-UX or a Linux system)
- Download the “net boot” ISOs from www.parisc-linux.org
- Follow the installation instructions on the site **very carefully**, really, this is complex.
  - I mean this...wrong file name, or dhcp mis-configuration can result in nastily cryptic error messages (like “`Invalid Lifimage`”).
  - Once you escape into IODC, the boot command is simply `boot net`. If you have configured bootp/dhcp correctly, the machine should find it's IP and boot.

Now You've Booted

- That wasn't so hard, was it.
- Now, the fun begins. Try using palinux as you would have used your HP-UX workstation.
- Just trying out palinux 2.6 in everyday situations is the best testing you can do for us.
- Don't worry if you break something, we'd love to hear about it (and fix it...we will even let you keep the pieces).
- Try out the problem areas (like X, HIL, etc.) who knows, you might spot a solution that we missed.
- Most of all, if it works for you, tell your friends (especially those who have PA-RISC hardware).

If You Have Problems

- Don't be discouraged.
- The first step to solving the problem is finding it.
  - Or, to put it another way: we can't fix it if we don't know about it.
- Finding and clearly reporting problems is the biggest service you can do for the palinux project.
- See the website (www.parisc-linux.org)
  - First, check the Frequently Asked Questions (FAQ)
  - If it's not in the FAQ, congratulations, you found a new one.
  - Read “My kernel {crashed|hung}, how do I submit a useful bug report?”
  - Send it in (To: parisc-linux@parisc-linux.org).

Conclusions

- Palinux isn't quite as stable as, say, x86 linux, but it's getting there.
- **You** can help us improve the stability.
- Almost certainly, it won't eat your data, destroy your disc or even launch a precision guided nuclear missile strike against China.
- Palinux is a new frontier in linux.
  - Linux on x86 is old, well tested and now mainstream.
  - If you want fun, excitement and really wild things...Join us.