Linux.conf.au

23 January 2003

My computer is bigger than yours!

Martin Schwenke

IBM OzLabs Linux Technology Center

<martins@au.ibm.com> · <martin@meltin.net>







Your computer...

- 1 CPU
- 256MB RAM
- 2 Ethernet interfaces
- 2 IDE disks
- 1 CD-ROM drive
- 1 CD-RW drive







My computer...

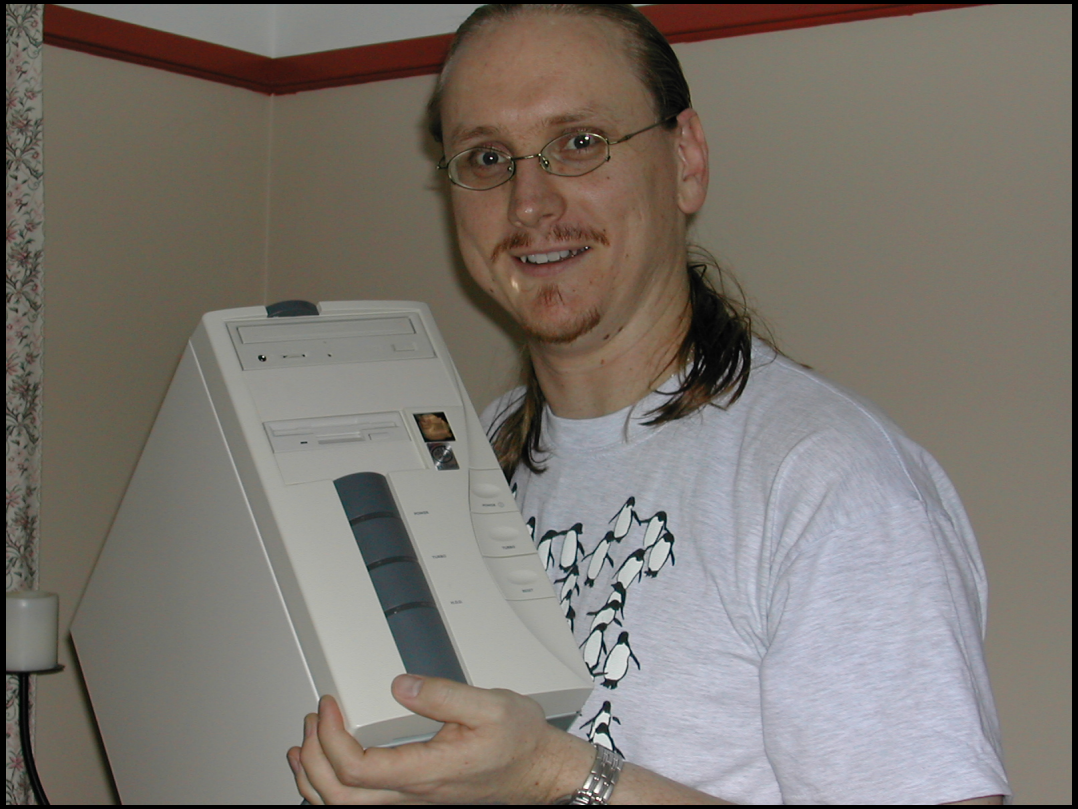
- 32 CPUs
- 64GB RAM
- 16 Ethernet interfaces
- 540 SCSI disks
- 1 CD-ROM drive
- 1 CD-RW drive







Your computer...



My computer...

Availability and Serviceability

- My customer wants 99.99% availability!







Availability and Serviceability

- My customer wants 99.99% availability!
- Yay hot-plug!







Availability and Serviceability

- My customer wants 99.99% availability!
- Yay hot-plug!
- OK, let's say `/dev/scsi/host0/bus0/target8/lun0` is failing.







Availability and Serviceability

- My customer wants 99.99% availability!
- Yay hot-plug!
- OK, let's say `/dev/scsi/host0/bus0/target8/lun0` is failing.
- Which one of my 540 disks is that?







Vital Product Data

Vital Product Data (VPD) is information that uniquely identifies hardware and, potentially, software elements of a system. The VPD can provide the system with information on various Field Replaceable Units such as part number, serial number, and other detailed information. The objective from a system point of view is to make this information available to the system owner and service personnel.

— PCI Local Bus Specification. Release 2.2. Page 289.







VPD — binary or text?

- PCI 2.2 specification defines a format for VPD to be used in PCI devices. This is a binary format containing ASCII fields.
- AIX uses a textual representation for VPD (`lsvpd`).







Example of VPD as text

```
*DS 16 Bit SCSI Disk Drive
*AX /dev/scsi/host0/bus0/target8/lun0
*MF IBM
*TM ST318305LC
*YL U1.9-P1/Z1-A8
*FN 09P4435
*RL 43353039
*SN 00043CD2
*EC H11936
*PN 09P4434
*Z0 000003129F00013E
*Z1 0211C509
*Z2 1000
*Z3 02041
*Z4 0001
*Z5 22
*Z6 162870 C
```

Example of VPD as text, explained!

```
*DS 16 Bit SCSI Disk Drive
*AX /dev/scsi/host0/bus0/target8/lun0
*MF IBM                 # Manufacturer
*TM ST318305LC          # Type/model
*YL U1.9-P1/Z1-A8       # Target 8, (integ.) bus 1, planar 1, drawer 9, rack 1
*FN 09P4435             # Field Replaceable Unit Number
*RL 43353039            # Firmware release level (hex encoded => C509)
*SN 00043CD2            # Serial number.
*EC H11936              # Engineering level of board.
*PN 09P4434             # Part number.
*Z0 000003129F00013E    # 1st 8 bytes of standard SCSI INQUIRY
*Z1 0211C509
*Z2 1000                # ... fields starting with 'Z' are device-specific.
*Z3 02041               # ... fields starting with 'Y' are system-specific.
*Z4 0001                # ... fields starting with 'V' are vendor-specific
*Z5 22
*Z6 162870 C
```







VPD as hardware inventory

- VPD for all (important) components $\Rightarrow$ *hardware inventory*.







VPD as hardware inventory

- VPD for all (important) components $\Rightarrow$ *hardware inventory*.
- Must be available before a service event takes place.







VPD as hardware inventory

- VPD for all (important) components $\Rightarrow$ *hardware inventory*.
- Must be available before a service event takes place.
- Persistent: boot-time hardware failures, change management.







Hardware inventory stages

1. Collection.
2. Storage and change management.
3. Rendering —boring.







Hardware Inventory pSeries Linux

- Copy of Open Firmware (OF) device-tree in `/proc/device-tree`.







Hardware Inventory pSeries Linux

- Copy of Open Firmware (OF) device-tree in `/proc/device-tree`.
- `ibm,vpd` properties for PCI 2.0/2.1 devices.







Hardware Inventory pSeries Linux

- Copy of Open Firmware (OF) device-tree in `/proc/device-tree`.
- `ibm,vpd` properties for PCI 2.0/2.1 devices.
- Collection easy: `find`.







Hardware Inventory pSeries Linux

- Copy of Open Firmware (OF) device-tree in `/proc/device-tree`.
- `ibm,vpd` properties for PCI 2.0/2.1 devices.
- Collection easy: `find`.
- Storage: database = `/var/lib/device-tree`.
- Render before storing, in textual format: `linux,vpd`.







SCSI

- All (most?) SCSI devices contain a certain amount of data.
- Not in PCI format.







SCSI

- All (most?) SCSI devices contain a certain amount of data.
- Not in PCI format.
- Collection: SCSI INQUIRY commands via SG interface.







SCSI

- All (most?) SCSI devices contain a certain amount of data.
- Not in PCI format.
- Collection: SCSI INQUIRY commands via SG interface.
- Storage: Rendering program + short template file, to `linux,vpd`.







SCSI

- All (most?) SCSI devices contain a certain amount of data.
- Not in PCI format.
- Collection: SCSI INQUIRY commands via SG interface.
- Storage: Rendering program + short template file, to `linux,vpd`.
- Currently:
  - Several hacks for AIX compatibility.
  - Only extended VPD for certain IBM devices.







VPD from PCI 2.0/2.1 devices

1. Check PCI configuration space to determine whether there is an expansion ROM.
2. If there is a ROM, but it has no address assigned, then assign one.
3. If the ROM is disabled, enable it.
4. Check the PCI data structure in the ROM to see if there is an address for VPD.
5. Read the VPD.
6. Disable, unassigned ROM as necessary.







VPD from PCI 2.0/2.1 devices

1. Check PCI configuration space to determine whether there is an expansion ROM.
2. If there is a ROM, but it has no address assigned, then assign one.
3. If the ROM is disabled, enable it.
4. Check the PCI data structure in the ROM to see if there is an address for VPD.
5. Read the VPD.
6. Disable, unassigned ROM as necessary.
7. Kaboom!







VPD from PCI 2.2 devices

- Devices expose VPD via capabilities list.
- I/O via configuration space pseudo-files under `/proc/bus/pci`.







VPD from PCI 2.2 devices

- Devices expose VPD via capabilities list.
- I/O via configuration space pseudo-files under `/proc/bus/pci`.
- Works like a beauty!







Exposing VPD through sysfs

- Expose VPD via sysfs?







Exposing VPD through sysfs

- Expose VPD via sysfs?
- One extreme: Expose binary PCI or textual format VPD.







Exposing VPD through sysfs

- Expose VPD via sysfs?
- One extreme: Expose binary PCI or textual format VPD.
- Other extreme: Expose no VPD.







Exposing VPD through sysfs

- Expose VPD via sysfs?
- One extreme: Expose binary PCI or textual format VPD.
- Other extreme: Expose no VPD.
- Suitable compromise: Expose some stuff currently exposed via ad hoc interfaces, such as SCSI INQUIRY data.







sysfs as a database format

- Replace augmented device-tree with sysfs-like structure?







Persistence and change management

- Hardware inventory needs to persist across boots.







Persistence and change management

- Hardware inventory needs to persist across boots.
- Detection of hardware changes: addition, removal, replacement.
- Fault diagnosis.







Persistence and change management

- Hardware inventory needs to persist across boots.
- Detection of hardware changes: addition, removal, replacement.
- Fault diagnosis.
- Assistance for persistent device naming.







Persistence and change management

- Hardware inventory needs to persist across boots.
- Detection of hardware changes: addition, removal, replacement.
- Fault diagnosis.
- Assistance for persistent device naming.
- AIX tools (e.g. `diag -a`), Red Hat's `kudzu`, ...







Persistence and change management

- Hardware inventory needs to persist across boots.
- Detection of hardware changes: addition, removal, replacement.
- Fault diagnosis.
- Assistance for persistent device naming.
- AIX tools (e.g. `diag -a`), Red Hat's `kudzu`, ...
- Run-time hardware changes tracked via hot-plug and above.







Persistent device naming

- Linux 2.4: aliases/links via `devfs`.







Persistent device naming

- Linux 2.4: aliases/links via `devfs`.
- Linux 2.5: hot-plug.







Persistent device naming

- Linux 2.4: aliases/links via `devfs`.
- Linux 2.5: hot-plug.
- IBM 'Device Naming Project' — `scsiname` is a userspace utility, but does its own SCSI INQUIRYs.







Persistent device naming

- Linux 2.4: aliases/links via `devfs`.
- Linux 2.5: hot-plug.
- IBM 'Device Naming Project' — `scsiname` is a userspace utility, but does its own SCSI INQUIRYs.
- Could be a thin layer around hardware inventory management system.







Investigation needed...

- Distributed Management Task Force's *Common Information Model* (CIM).
- Probably the database we want, provided it can be used early enough at boot time...







Status

- Originally Perl script that rendered `ibm,vpd` properties.
- Subsequent Perl version augmented `device-tree` with `linux,vpd` files for SCSI devices.







Status

- Originally Perl script that rendered `ibm,vpd` properties.
- Subsequent Perl version augmented `device-tree` with `linux,vpd` files for SCSI devices.
- Current implementation is about 2000 lines of C code and shell scripts.







Status

- Originally Perl script that rendered `ibm,vpd` properties.
- Subsequent Perl version augmented `device-tree` with `linux,vpd` files for SCSI devices.
- Current implementation is about 2000 lines of C code and shell scripts.
- Incomplete, more work needed.
- Plan to release as Open Source a little later this year.







Questions?

- ?