The Linux Kernel and Internals for Scientific Cluster Users

AHPCC Linux Superclusters Users Conference

Tramm Hudson

Pete Beckman

Andreas Dilger

TURBOLINUX™

Special thanks to: Bill Saphir, Patrick Bozeman, Rémy Evard

Who Are We?

- Pete Beckman <beckman@turbolinux.com>
  - Formerly an employee at Los Alamos National Laboratory in the Advanced Computing Laboratory
  - Began to work on Linux Clusters several years ago, founded the Extreme Linux workshops
  - Director of TurboLabs, started in Santa Fe, April 2000
- Tramm Hudson <thudson@turbolinux.com>
  - Formerly an employee at Sandia, stunt programmer for Cplant Linux Alpha cluster
  - Joined TurboLinux in May, 2000
  - Soon will have his private pilot’s certificate (he has already flown solo)
- What is TurboLabs?
  - An advanced computing laboratory for TurboLinux
  - Currently, we are focused on R&D for Linux clusters...

Outline & Schedule for Tutorial
(what you should learn)

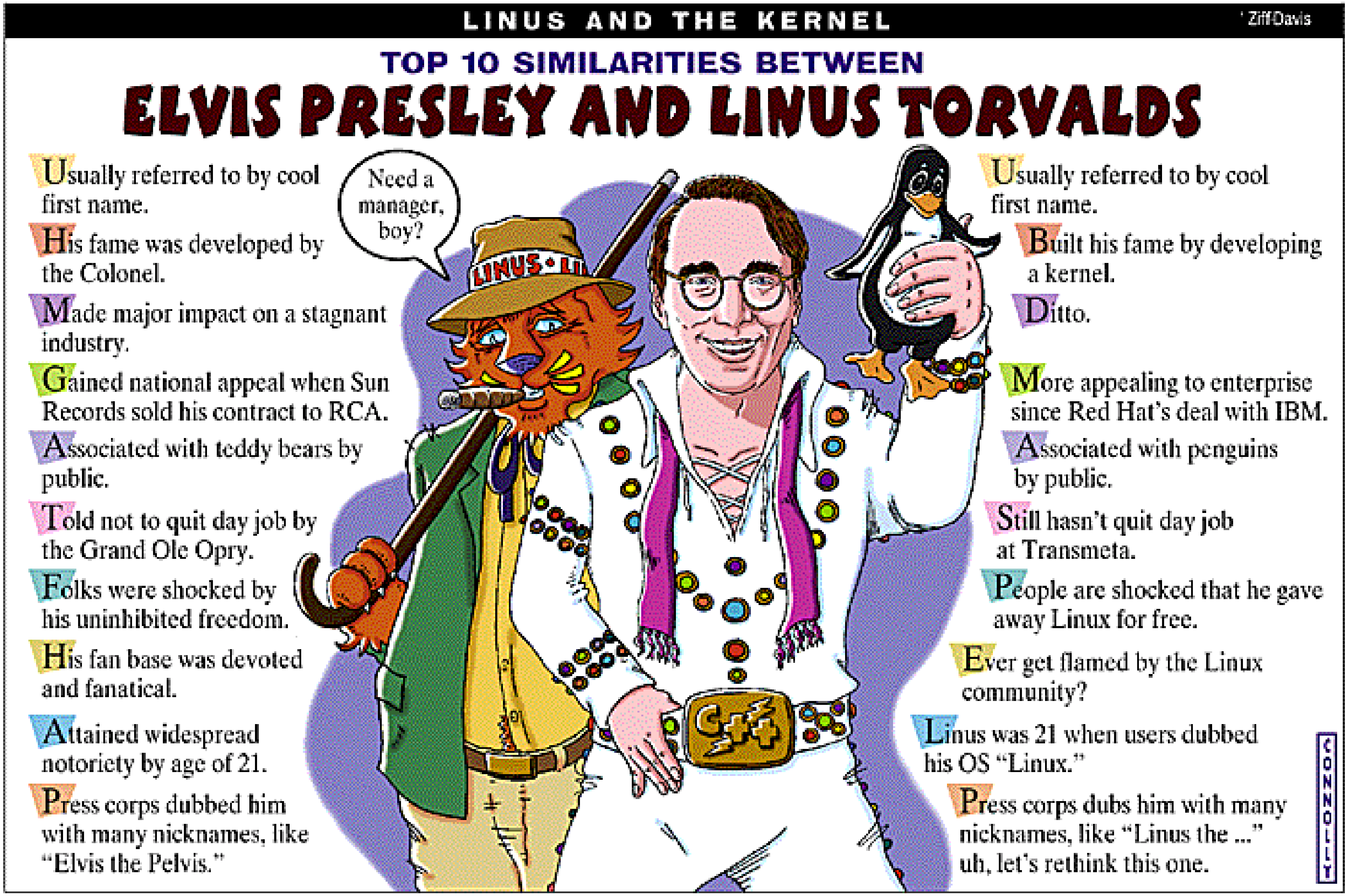
1:30 – 2:00:	An Introduction to Linux:
	Tech, Philosophy, Culture, Timeline, Religion
2:00 – 3:00:	The Linux Kernel
	Guts, modules, building, patching, testing
3:00 – 3:15:	Break
3:15 – 3:30:	Linux Filesystems
3:30 – 3:45:	A Linux distribution & configuration
3:45 – 4:30:	Overview: Linux for Scientific Clusters
	Arch, limits, theory, costs, networking
	Creating, running, programming, debugging
4:30 – 4:45	The Future of Linux and the world
4:45 – 5:00	Questions & Discussion (open mike...)

What is Linux?

- Another flavor (clone) of the original Unix kernels
- A GPL-licensed kernel (more on this later...)
- Mature: Officially released on the Internet October 5th, 1991
- Originally created by Linus Torvalds who continues maintenance with cast of thousands
- "Linux" sometimes refers to the entire "distribution" (strictly speaking, that not correct)
- "Disruptive technology"?

http://www5.zdnet.com/zdnn/stories/news/0,4586,2504419,00.html?chkpt=zdnntop

Who is Linus Torvalds?



LINUS AND THE KERNEL

' ZiffDavis

TOP 10 SIMILARITIES BETWEEN

ELVIS PRESLEY AND LINUS TORVALDS

Usually referred to by cool first name.
His fame was developed by the Colonel.
Made major impact on a stagnant industry.
Gained national appeal when Sun Records sold his contract to RCA.
Associated with teddy bears by public.
Told not to quit day job by the Grand Ole Opry.
Folks were shocked by his uninhibited freedom.
His fan base was devoted and fanatical.
Attained widespread notoriety by age of 21.
Press corps dubbed him with many nicknames, like "Elvis the Pelvis."

Usually referred to by cool first name.
Built his fame by developing a kernel.
Ditto.
More appealing to enterprise since Red Hat's deal with IBM.
Associated with penguins by public.
Still hasn't quit day job at Transmeta.
People are shocked that he gave away Linux for free.
Ever get flamed by the Linux community?
Linus was 21 when users dubbed his OS "Linux."
Press corps dubs him with many nicknames, like "Linus the ..." uh, let's rethink this one.

The Kernel and “A Distribution”
They are distinct and separate

- The Linux Kernel (the operating system):
  - memory management, hardware control, processes, TCP/IP networking, filesystems, IPC, multitasking
  - There is no integrated web browser in the OS
  - Lines of code: entire x86 kernel = 1.7 M, 468K without drivers
- A Distribution:
  - Assembled by companies or groups:
    - TurboLinux, SuSE, Redhat, Slackware, Debian, etc
  - The expected “Unix-like” programs: ls, cp, tar, csh, perl, gcc
  - Other apps: X, emacs, apache server, StarOffice, etc
  - Lines of code: apache: 100K, Perl: 151K, X (xfree86): 1.73M
- Kernel version and Distribution version not related
- Distribution versions between vendors are not related

The Kernel and "A Distribution"
Two different development methodologies

- The Linux Kernel
  - Linus Torvalds: Benevolent dictator
    - A loosely knit group of programmers:
      - Donald Becker, Alan Cox, Dave Miller, Ted Tso, Stephen Tweedie, etc
    - A loosely knit group of evangelists:
      - Maddog Hall, Eric Raymond, Bob Young, Cliff Miller
    - Email patch submissions
    - Patches may be rejected, delayed, or ignored
    - AC: "I got tired of waiting for Linus to bless 2.X, so I released 2.X.preZ"
- A Distribution:
  - Anyone can package up and release a distribution (kernel + Unix environment)

Kernel Versioning & Source

- Kernel source can be downloaded from www.kernel.org
- There are always two “latest” kernels: “development” and “stable”
- Stable kernels use a even minor number while development kernels use odd numbers (v2.2 or v2.3)
  - Some vintages are “good”.
- Companies that package a distribution (TurboLinux, Redhat, SuSE, etc) often have combined the stable kernel with certain patches
- Believe it or not, this is almost exactly like what happens when you buy a Win98 box from Dell, IBM, or Compaq
- Grabbing the latest kernel and shoving it onto your machine may be a bad idea...

Linux Gains Critical Mass, Why?

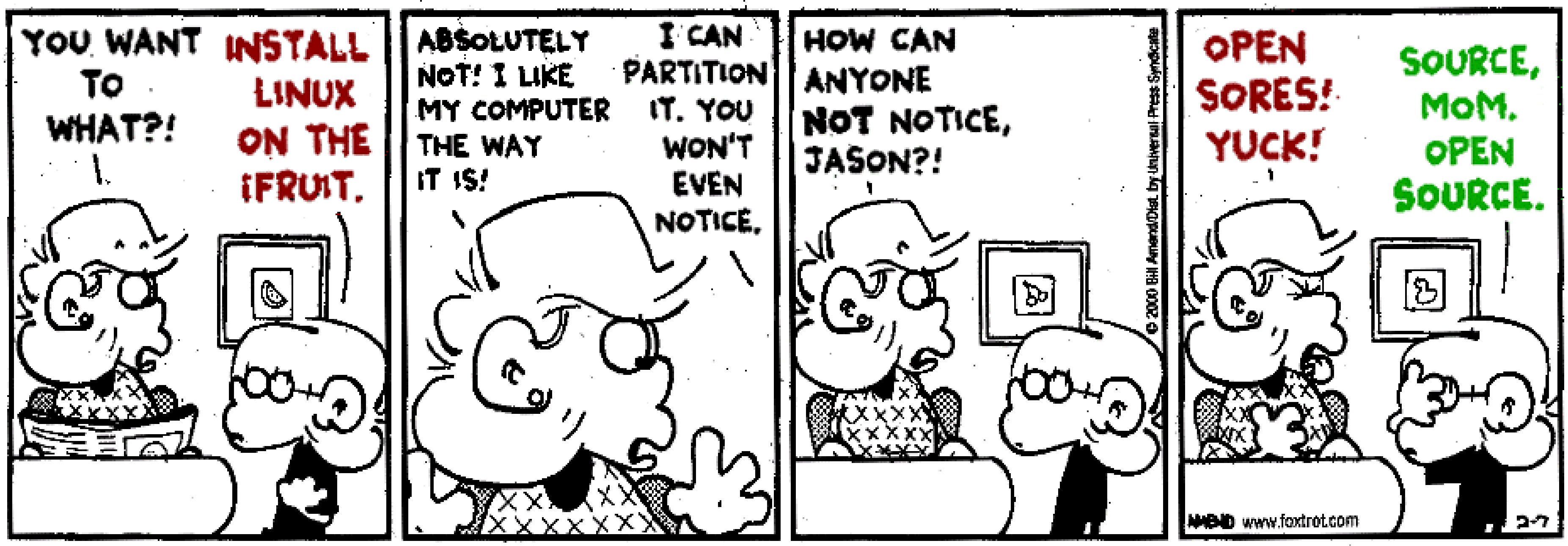
Some humble opinions....

- GPL Licensing (more on this in a moment)
- Existing GNU software (bash, gcc, ls, etc)
- The Internet provided communication backbone for developers
- Desktop Intel architecture no longer completely brain dead (the 386 emerged on the desktop)
- Traditional Unix platforms (HP, IBM, Sun, DEC, etc) were very expensive and out of the reach of most student programmers
- Community needs inexpensive Internet servers

Open Source and Free Software

A short explanation & history, why it matters...

FoxTrot: Bill Amend


YOU WANT TO WHAT?!
INSTALL LINUX ON THE iFRUIT.
ABSOLUTELY NOT! I LIKE MY COMPUTER THE WAY IT IS!
I CAN PARTITION IT. YOU WON'T EVEN NOTICE.
HOW CAN ANYONE NOT NOTICE, JASON?!
OPEN SORES! YUCK!
SOURCE, MOM. OPEN SOURCE.
© 2000 Bill Amend/Dist. by Universal Press Syndicate
AMEND www.foxtrot.com
2-7

- It is effecting business strategy
- The popular media is has noticed
- Radical Decentralization of Authorship
  - Hackers from more than 50 countries have contributed to the Kernel

Richard Stallman
www.gnu.org

- In 1984 RMS begins “GNU” project (wrote emacs, gcc, and gdb)
- RMS uses the term “Free Software” to describe freedom not price (gratis vs liberty).
- “Ethical” and “Moral” are terms often used by RMS to describe Free Software
- GPL or “copyleft” (GNU General Public License)
  - “Viral” or “sticky” license for derivative works
  - Source must be made available
- LGPL (Library/Lesser General Public License)
  - Permits linking to a non-GPLed code
- Insists people refer to “GNU/Linux” when talking about the entire OS and Unix environment
- IMHO: Trading one orthodoxy for another (GNU is often about religion)

The “Open Source” Movement

- Spring 1997: Eric Raymond insists a marketing campaign and new term is needed
- The Open Source Definition is created
  - Version 1.7: http://www.opensource.org/osd.html
- The GPL is one Open Source license, but the BSD License is also considered to be Open Source
- Open Source has become the generic term for describing software that is modifiable without restriction and freely distributable

Why Should You Care About Licensing: Open Source, GPL, etc?

- You don't want to violate a license
  - Modifying the kernel and then releasing it without making the modification available in source form is forbidden
  - Closed source loadable kernel modules are ok, but difficult to maintain
  - Linking to GPLed code or using bits of GPLed code *could* force you to GPL your source code (viral nature)
- Choosing a license for your software development is **very** important: GPL or BSD?
- Is Open Source a fad? Will it eventually collapse? What is the business model?

The Linux Kernel

A Guided Tour of /usr/src/linux

Kernel Directories in /usr/src/linux

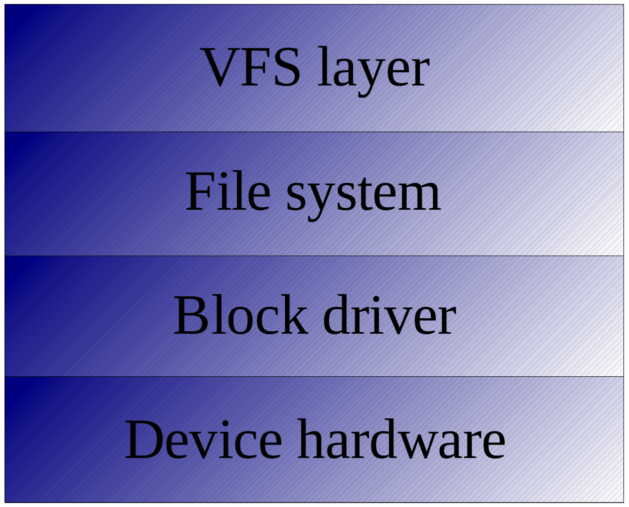
./arch	Architecture specific code
./drivers	All device drivers
./fs	Filesystems
./net	Networking protocols
./mm	Arch ind. memory mgt

/usr/src/linux/arch

- Architecture/processor specific code and management routines.
  - Listed below roughly in decreasing usage
  - ./i386
  - ./alpha
  - ./sparc
  - ./mips
  - ./ppc
  - ./s390
  - ./m68k

/usr/src/linux/drivers/block

- Devices that deal with large data blocks at a time, like hard disks.
- No filesystem information

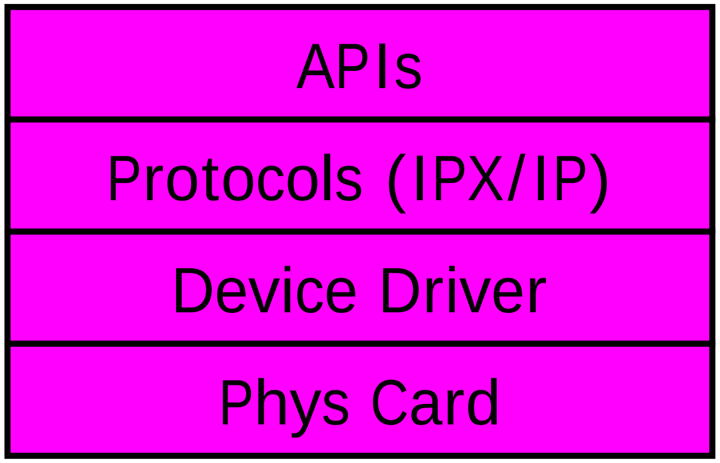


/usr/src/linux/drivers/char

- Character-at-a-time device
  - TTY
  - Mice
  - Joysticks
  - Sound cards (but not here, they are kept in a different directory (yes, this is inconsistent))

/usr/src/linux/drivers/net

- Network interface device drivers.
- No protocol information (like block devices and filesystems)



/usr/src/linux/drivers/

- ./scsi
  - SCSI controllers (still no filesystem info)
  - Generic SCSI devices
- ./sound
  - Sound card char drivers are here, not in ./drivers/char
- ./pci, ./usb, ./sbus
  - Different bus controllers for different architectures
- ./net
  - Network cards, no protocol information here

Correct, for now

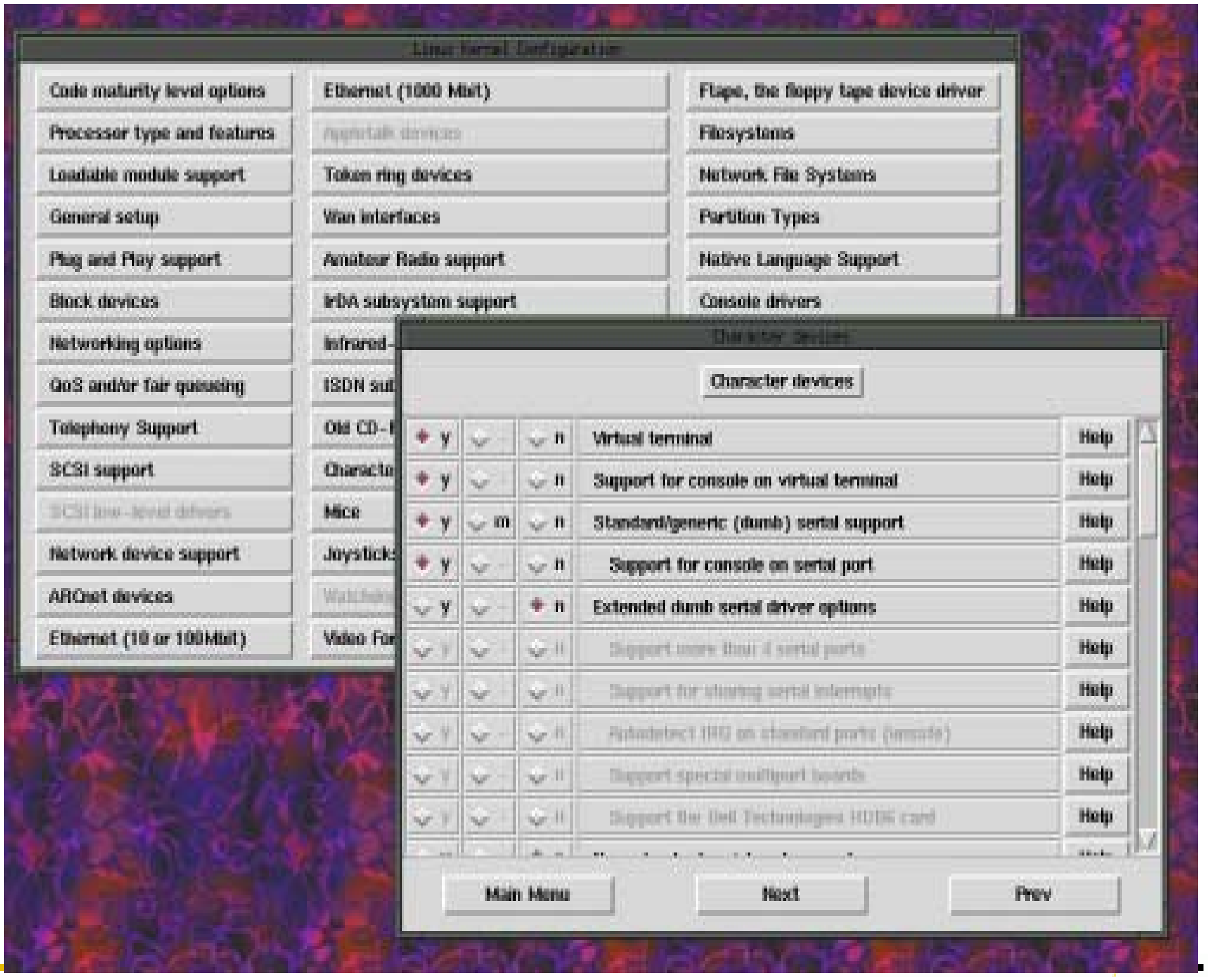
- Traffic on linux-kernel mailing list suggests that reorganization may be likely very soon.
- None of this controls which drivers get built into the kernel.
- Some device drivers get built elsewhere
  - PCMCIA
  - Alsasound
  - Third party drivers

So, you want to build a kernel?

- Step one is fetch the "mainline" kernel from http://www.kernel.org
- Remove /usr/src/linux symlink.
- Unpack in /usr/src
- ***This is not the same source that shipped with your distribution***
  - Extra patches for features that have not been blessed by Linus
  - Specialty device drivers for vendor hardware

Configuring Your Kernel

```
# cd /usr/src/linux
# make xconfig
```



[✓ y]		n	Virtual terminal
[✓ y]		n	Support for console on virtual terminal
[✓ y]	m	n	Standard/generic (dumb) serial support
[✓ y]		n	Support for console on serial port
y		[✓ n]	Extended dumb serial driver options
y		n	Support more than 4 serial ports
y		n	Support for sharing serial interrupts
y		n	Autodetect IRQ on standard ports (unsafe)
y		n	Support special multiport boards
y		n	Support the Bell Technologies HUB6 card

Custom Kernels for Clusters

- You should explore the options and features being compiled into your compute nodes
- Some changes are required for clusters
- Save memory: remove useless features
- Save time: optimise for specific processor

Custom Kernels: Necessary Features

- Serial console
  - VERY handy unless you enjoy pushing shopping carts with VGA monitors around the machine room
- Kernel IP config
  - Permits bootp, etc
- NFS root
- Myrinet / Gig-E / ATM
- APM and Wake-on-LAN
  - If required

Custom Kernels: Unneeded Features

- Sound
  - Unless you want *real* surround sound
- CD-ROM
- QoS
- Telephony, Video
- ISDN, ARCnet, Appletalk, Token ring, WAN, AX.25
- IrDA
  - Unless you have a wireless IR cluster

More Unneeded Features

- USB, Mice, Joysticks
- SCSI, ftape
- NLS
- Extra network cards
- Extra filesystems
  - Only NFS and ext2 required.

(M)odules versus (Y)es

- Most drivers and devices have an option to be built as a loadable module, instead of in the kernel.
- For general systems, this is very useful
  - Saves boot time kernel memory
  - Allows seldom used features to be demand loaded
  - Easier development for custom drivers

...But Not for Clusters

- For HPC clusters, features are generally not loaded and unloaded
- For clusters it simplifies start up to have everything in the kernel.
  - Kernel IP and NFS root require network drivers to be in the kernel
  - Root filesystem device driver and FS driver must be in kernel
- Bootp kernel / minimal root filesystem

A Quick Tour of Loadable Kernel Modules

- They are loaded on demand from a (possibly) user-space triggering event
  - User-level MP3 player can load sound module
- Several basic utilities:
  - lsmod, rmmod, insmod
  - depmod, modprobe: Manipulates dep. Info
- /lib/modules/2.2.14
- Versioning information is very important, yet still pretty tricky

Building the Kernel

- Select the needed features and prune the legacy baggage.
- If you are booting from floppy / harddisk
  # make bzImage
- If you have better network hardware
  # make bootpfile
- Go get some coffee....
  - Depending on your hardware, you may fly to Brazil and grow a crop of coffee.

Installing the Kernel

- For a local machine

```
# cp ./arch/i386/boot/bzImage  /boot/vmlinuz.new
# vi /etc/lilo.conf
# lilo
# shutdown -r now
```

- Clusters are more complicated and depend on the specifics of the cluster software.

Setting up the cluster

- Necessary daemons
  - dhcpd / bootpd
  - nfsd, export NFS Root
  - Build boot floppies / configure BIOS
- Full tutorial... There are **LOTS** of gory details that are very specific to your cluster.
- Hopefully in the future, methods will settle into SOP

Some Low-Level Details of the Boot Process

The boot process (i386)

- BIOS initializes hardware
- BIOS loads bootblock from first disk
  - Unless you're running Linux BIOS by Ron Minnich
- Bootblock starts LILO (Linux Loader)
- LILO reads its configuration file from known disk block to find which kernels are configured.
- LILO waits for options and kernel selection.
  - Some clusters use serial console to input LILO options

Useful kernel boot time options

- single
- root=/dev/nfs
- ip=192.168.1.13
- init=/bin/bash
- console=/dev/ttyS0,9600

Loading the Kernel

- LILO has a list of kernel blocks on the device and loads them into memory.
  - Does not go through filesystem
  - You must rerun LILO after changing or moving kernels, modifying /etc/lilo.conf, or restoring from backup to different disk geometry.
- LILO may use the zlib library to decompress the kernel image.
  - If the kernel too large, it must be compressed.
- LILO jumps into start_kernel() in ./init/main.c

start_kernel()

- Calls architecture specific code to setup command line and initialize memory
- Initializes VM system, scheduling, interrupts and other system services.
- Parses command line options

Parsing command line options

- Architecture specific means of retrieving them from the boot loader.
- ./init/main.c:parse_options() has a dispatch table of option name to handler functions.
- You can add your own options to the `cooked_param[]` with this prototype:

```
extern void foo_setup(char *str,
int *ints);
{ "foo=", foo_setup },
```

Back in start_kernel()

- Sets up SMP, if configured
- Spawns kernel thread init() and goes into idle loop.
- Will be woken up when system calls are made

init()

- Locks kernel
- Frees unused kernel memory
- Calls do_basic_setup()

do_basic_setup()

- Initializes all bus drivers
  - There has to be a joke in here... Donuts? Coffee?
- Launches kernel threads
  - Bdflush
  - Kupdate
  - Kswapd
  - Kpiod
- Mounts root filesystem / loads ramdisk
- Returns to init() thread

Back in init()

- Attempts to exec user specified init (passed in as init=...)
- Otherwise uses /sbin/init, /etc/init, /bin/init, /bin/sh
- If none of these work, it panics with an error message:
  - No init found. Try passing init= option to kernel.
- This can happen if your custom init can't load its libraries or if the root filesystem gets wedged.

Diskless boots are ~~different~~ better

- Ideally, the BIOS does a bootp request to find the nodes address and tftp's the kernel and starts it running (this is not common yet)
- ./net/ipv4/ipconfig.c:ip_autoconfig_setup() gets called during startup
  - Does another bootp request for node information.
- ./fs/super.c:mount_root() notices that NFS root is enabled and attempts to mount it.
  - If it fails, mounting the floppy is attempted
  - If that fails, the kernel panics

Advantages of Diskless

- No spinning parts in compute node
- No per-node management or configuration is required
  - No images are pushed to disks
  - LILO does not have to be run on the nodes
  - Down nodes get "updates" as soon as booted.
- If feasible, no floppies are required, either.

Setting Up a Minimal NFS Root

- It is easy to produce a single application NFS root -- statically link your app and call it as init=foo.
- General purpose ones are harder and may require library wrangling.
  - libc.so will be required, 5.25 MB of space
  - ld.so will also be needed, 500 KB
- Other than those, you need to use ldd to find what is required.
  - dlopen() libraries are not listed. Must find by trial and error.

Example: What does passwd need?

-
  ```
# ldd /usr/bin/passwd
  libdl.so.2 => /lib/libdl.so.2 (0x4001f000)
  libpam.so.0 => /lib/libpam.so.0 (0x40022000)
  libpam_misc.so.0 => /lib/libpam_misc.so.0
(0x4002a000)
  libpwdb.so.0 => /lib/libpwdb.so.0 (0x4002d000)
  libc.so.6 => /lib/libc.so.6 (0x40075000)
  /lib/ld-linux.so.2 => /lib/ld-linux.so.2
(0x40000000)        libcrypt.so.1 =>
/lib/libcrypt.so.1 (0x4015a000)
  libnsl.so.1 => /lib/libnsl.so.1 (0x40188000)
```
- Total: 6.7 MB (not counting PAM modules)

How to build a bootfloppy

- First ask yourself -- are you reinventing Tom's Root/Boot disk?
  - http://www.toms.net/rb/
- Make filesystem on floppy
  - mke2fs /dev/fd0
  - mount /dev/fd0 /mnt/floppy
- Copy all of the programs that you'll need and libraries that are required.
  - ldd to find libraries
  - Inspection of shell scripts for commands
- Make device files on floppy
  - You can run /dev/MAKEDEV, but 99% of the files are not needed on a floppy.
- Copy the custom kernel to floppy and write a lilo.conf to use it.

Example lilo.conf

```
boot    = /dev/fd0
map     = /boot/map
install = /boot/boot.b
backup  = /dev/null
delay   = 50
compact
read-write

image   = vmlinuz
        label   = linux
        append  = "root=/dev/nfs ide0=dma"
        root    = /dev/fd0
```

Scripts to Automate Boot Floppies

- Many are out there.
- Look at Tom's, LRP, etc
- TurboLinux will be releasing one
  - Currently used internally

Tuneable kernel parameters in /proc

- Runtime parameter changes are possible
- Fetch parameters with 'cat'
  - cat /proc/sys/kernel/hostname
  - cat /proc/meminfo
- Modify parameters with 'echo'
  - echo 1 > /proc/sys/net/ipv4/ip_forward
  - echo 5 > proc/sys/net/ipv4/tcp_max_syn_backlog

Linux Filesystems

Andreas Dilger is Our File System Kernel Guru

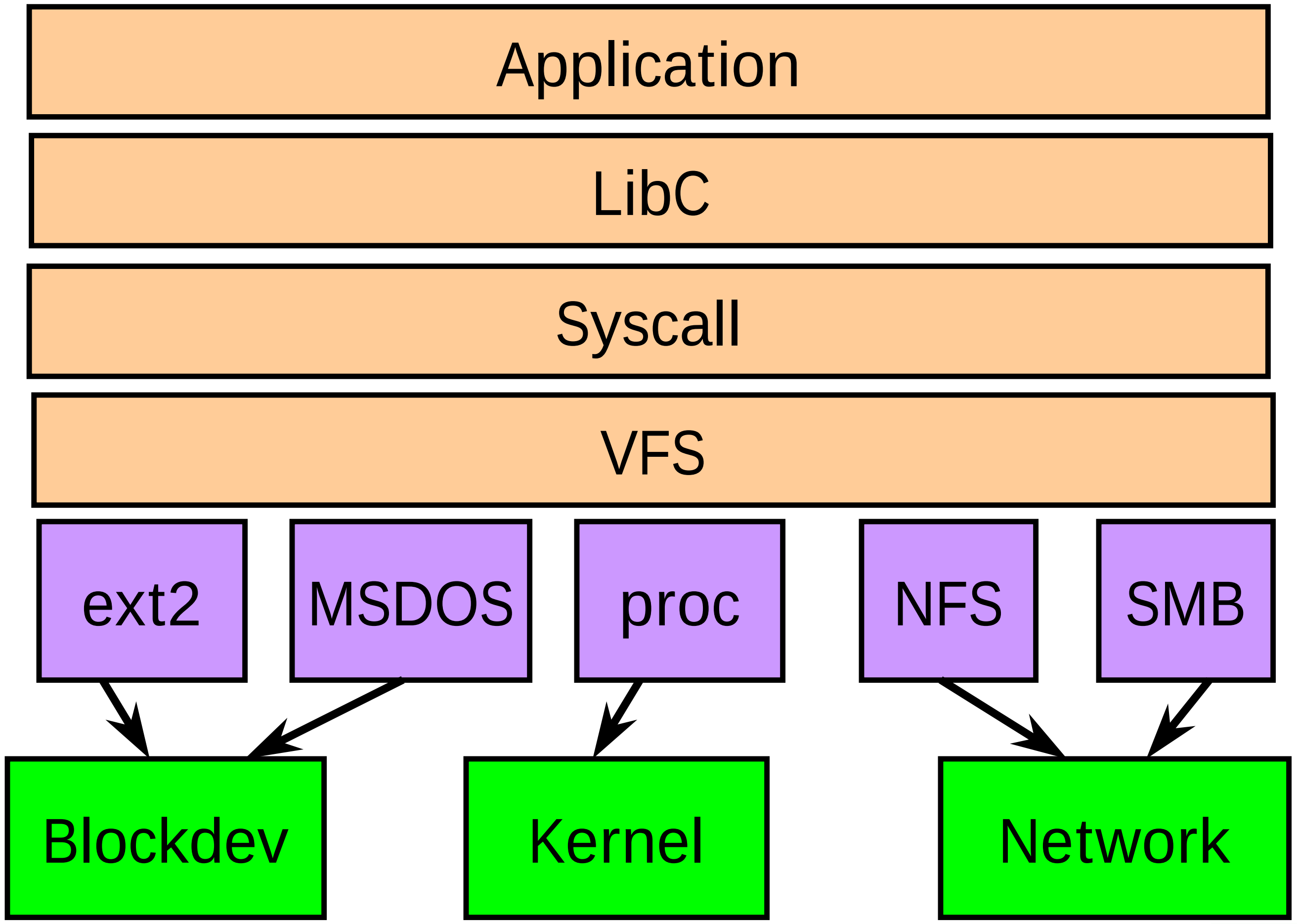
Linux Filesystems

- Local disk filesystems
  - Ext2, msdos/vfat, isofs/udf, ntfs/hpfs, ufs, hfs...
- Newer journaling filesystems
  - Ext3, reiserfs, XFS, IBM JFS
- Network filesystems
  - NFS, AFS, SMBFS
- Distributed filesystems
  - Coda, InterMezzo, GFS
- Others
  - Proc, devfs, shmfs, ramfs

Virtual Filesystem Layer

- “Object Oriented” abstraction of filesystem internals
- Filesystems modular, except boot fs
  - Module name = fs type in /etc/fstab
- VFS does not know fs specifics
- VFS works with generic superblock & inode
  - Superblock/inode hold pointers to fs data/functions
  - VFS calls method in inode by name

VFS and Filesystem



Life of a Filesystem

- Mount of fs checks /etc/fstab for type
- Kernel load module for filesystem
- Filesystem registers itself with kernel
  - VFS only knows fs type, fs read_super method
- VFS calls read_super
  - Reads superblock from disk, initializes generic sb
  - Superblock points to fs-specific operations
    - Read/write/update/delete inode
    - Write superblock
    - Statfs (returns used and free space, used ad free inodes)

Life of a Filesystem (slide 2)

- read_super loads root inode
- inode has fs-specific data, operations
  - Inode operations
    - Create/lookup/link/unlink file
    - mkdir/rmdir
    - Rename
  - File operations
    - Seek/read/write/sync
    - Pool/mmap/ioctl

Our Old Friend: Ext2

- Most commonly used Linux FS
- Long life based on performance / features
- Always maintained with the kernel
- Block based
  - 1, 2, 4K blocks selected at creation time
  - 64-bit file support for 2.4, 2.2+patches
  - Uses direct, indirect, double-, triple-indirect blocks for very large files

Our Old Friend: Ext2 (slide 2)

- Allocation algorithms localize files/data
  - Files near parent directory
  - Blocks near files
  - Preallocation of blocks reduces fragmentation
- Lots of development for new features
  - EA/ACL support (Posix)
  - Compression
  - Online resize
  - Threaded block/inode allocation

What is Journaling?

- Why do you care?
  - e2fsck 1hr/100GB, journal recovery in seconds
  - Have you ever corrupted a filesystem?
- Ensures FS is in a consistent state
- Usually does not prevent data loss
  - Last I/O before crash may be discarded
- Journaling of metadata only is fast
- Journaling of data may preserve app data better

Multi-stage Commit

1) Start transaction with unique ID
2) Write data to journal
3) Close transaction, start new transaction
4) Write trans. close record, update journal transaction ID
5) Copy data to FS
6) Transaction dropped

Linux JFS (not IBM JFS)

- JFS is generic journaling developed for ext3
- Writes full blocks into journal
- Groups multiple FS ops into 1 journal trans.
- Ensures ordered writes of blocks to journal and filesystem
  - Multi-stage commit
- Handles recovery by handling full blocks
  - Discard all I/O if transaction not complete
  - Copy full blocks to FS if complete
- No need to understand FS structure
- Can keep journal in a file or block device
- Needs kernel modes – not in 2.4 stock kernel

Journaling Ext2 (Ext3)

- Uses identical FS, layout as Ext2
- Uses 95% identical code as Ext2
- Journaling handled by JFS
- No fsck needed if kernel supports JFS
- Supported by recent versions of e2fs progs
- To Do:
  - Metadata-only journaling
  - Online resize
  - Port to 2.4
  - Performance turning

Reiser Filesystem

- Designed to put “database in filesystem”
- Handles small files well
- Handles large directories well
- Dynamic inode allocation
- Journalled
- Dedicated development team
- Not in 2.4 kernel because of journaling
- Doesn’t work with with NFS

Network Filesystems

- NFS
  - Server is a kernel thread in 2.4
  - Supports NFS V3 over TCP/IP in 2.4
- AFS
  - A binary client has been available for a long time
  - Source released by Transarc/IBM in August
  - Reliable, many companies/universities use AFS
- SMBFS
  - Win95/NT/2000
  - Can have Linux SMBFS client to Windows server
  - Can have Linux server to Windows clients

Distributed Filesystems

- Coda
  - From developers of AFS at CMU
  - Handles synchronization/authorization better than NFS
  - Clients cache data locally to disk
  - Disconnected operation permitted
- InterMezzo
  - Similar to Coda, but less complex
  - Caches data locally to filesystem, can work disconnected
- GFS
  - Uses shared disks
  - Serverless
  - Direct disk I/O improves performance
  - Needs special FCAL/SCSI disks

Other Filesystems on Linux

- Non-device filesystems
  - /proc exports kernel config to user space
  - devfs handles dynamic allocation
  - Shmfs makes IPC shm visible as files
  - Ramfs is page-cache based fie system
- Other filesystems on Linux
  - SGI XFS is available
  - IBM JFS is available
  - Many custom ones have been written

Linux Block Devices

- Meta Disk Driver
  - Block-to-block translation driver
    - Append, Striping (RAID 0), Mirroring (RAID 1), Parity (RAID 5)
- Loopback driver
  - Turns regular file into block device
  - Good for accessing disk/floppy images & testing
  - Can do encryption with kernel patches
- RAMdisk
  - RAM-based block dev
  - Used when booting install CD/floppy
  - Can build FS on RAMdisk

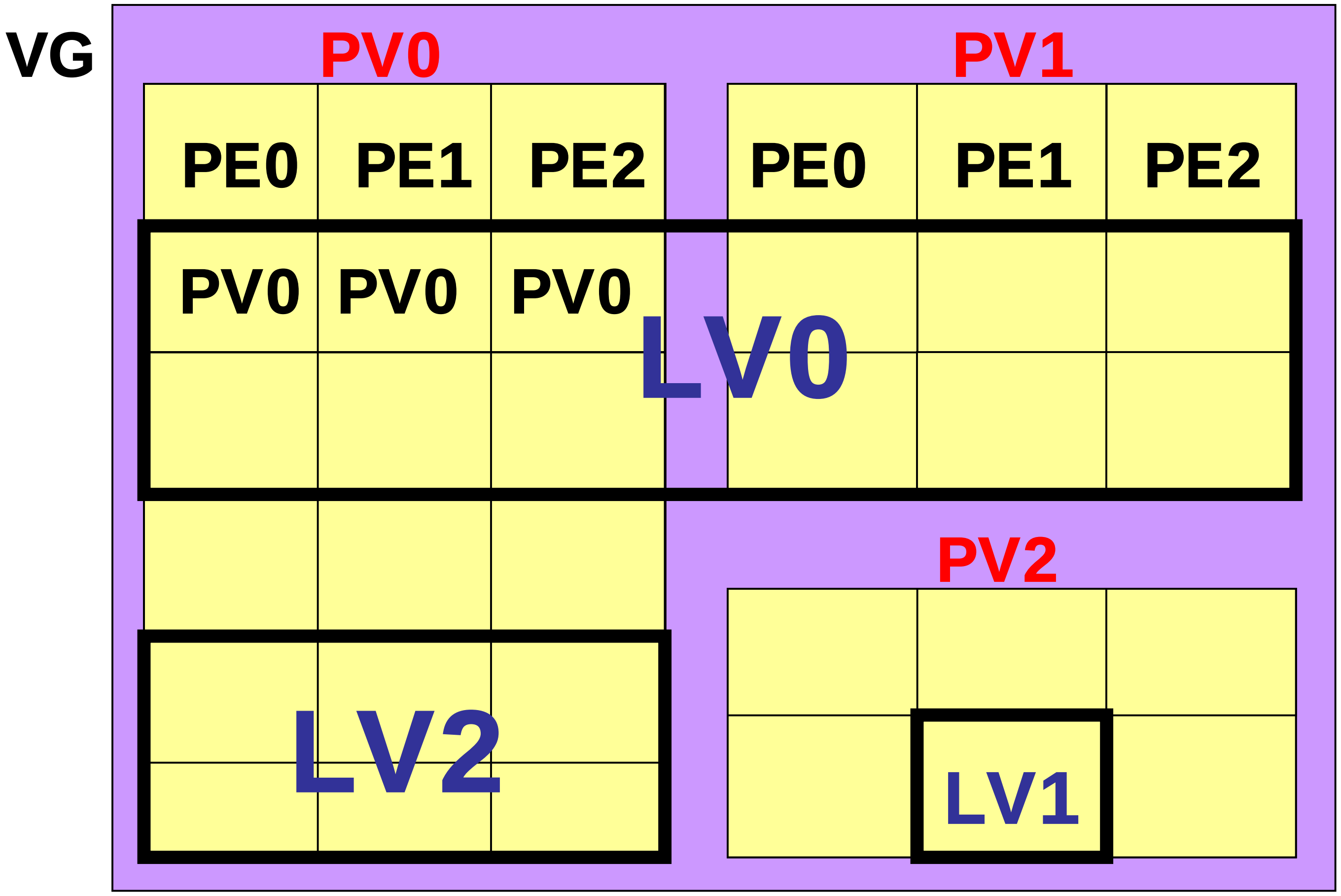
Logical Volume Manager

- Removes 1 to 1 mapping of FS to disk/partition
- Eases management of disks/filesystems
  - Allocate disk space as needed
  - Filesystems smaller/larger than disks
  - Move filesystems between disks
  - Filesystems don’t need to be contiguous
  - Can resize FS online with FS support (ext2 + patch, reiserfs)
- Built on top of MD/RAID block devices

How a LVM Works

- Disks are physical volumes (PV)
  - PVs are split into fixed-sized physical extents (PE)
  - A volume group (VG) is a virtual disk
  - PVs are groups into VGs
- A VG is divided into a logical volume (LV)
  - LV made of 1 or more logical extents (LE)
  - Les are mapped to PEs in arbitrary order and location
- Filesystem sees LV as a single contiguous device
  - LV exists in LE order
  - LVM kernel module remaps LE to PE
  - LE can be moved to a different PE
  - LV can be extended by adding LE/PE at end.

LVM Example



A Linux Distribution & Configuration

What Defines A Distribution?

Unlike legacy operating systems, there is not a sole source supplier for the operating system kernel, utilities and applications

Who Produces Distributions?

- Redhat: The ubiquitous
  - Known for its RPM package format
  - May be the next Microsoft.
- TurboLinux: High Performance Linux
  - Initially a RedHat derivative
  - Very popular in Asian markets
- SuSE
  - Has a cute green mascot
  - Popular in Europe

Other Distributions of Note

- SLS: Soft Landing System
  - One of the earliest (I installed Linux 0.88 with it)
  - No longer developed
- Debian
  - Devote followers of RMS
  - No non-free (speech) software included
- LRP
  - No user interface
  - For embedded systems

What Goes into the Distribution?

- Latest mainline Linux kernel
  - With some patches
  - An ENORMOUS selection of loadable modules (see module slides)
- Stable versions of most GNU packages
  - Emacs, gcc, ls, cp, rm, bash
- Installation tool
- Value added applications
  - Netscape, WP, etc. for end users
  - Apache, ClusterServer, etc. for servers

How Are They Differentiated?

- Irrationally
  - Religious views, ideology
    - Debian “Social Contract” for licensing
  - Package format preferences
  - File locations
    - SysV init versus BSD init does not help a business make money. Apache does.
- Rationally
  - The value added pieces
  - Installation ease
  - QA of complete, complex system of independent programs

The Past

- My first Linux install involved
  - Downloading twenty floppies worth of data over a 9600 modem.
  - Cross compiling a custom kernel from a VAX
  - Producing a bootable floppy by hand
  - Writing a partition table by hand
  - Reverse engineering video card specs, modifying video card driver, futzing with clock chips.
  - Hacking the ISA bus address assignment and hard coding the ports in the kernel.

The Now

- GUI Installers are X11 based
- CD-ROM (and DVD-ROM) installations eliminate downloading time.
- Thousands of packages are easily available
  - # ./configure && make && make install
  - # rpm -ivh foo-3.1.4.rpm
  - # appget install foo-3.1.4
- Most hardware is supported "out of the box"
  - There used not to be a box

Great, but...

- These distributions are all optimised for single machines, with real VGA consoles, keyboards and mice.
- The embedded market has little representation
  - LRP is an exception, but niche usage
- Clusters are not handled at all
  - Or they are handled as multiple machines

Clusters Are More Than Boxes

- The ideal cluster management tool scales such that n boxen are as manageable as 1 box.
- This means that many files are shared, configurations are replicated or retrieved from servers.
- Administrators should not have to think about individual boxes -- no personalities.
- Growing or shrinking the cluster, either by design or automatically because of failure should be trivial

Installing a Kernel on a Cluster

- Ideally, it is just a matter of:

  -
    ```
    # cp ./arch/i386/boot/bootpfile /tftpboot/vmlinuz
    ```
- But for many i386 clusters:

  -
    ```
    # for all node in $nodes; do
     scp bzImage root@$node:/boot;
     ssh root@$node lilo;
    done
    ```
  - Or the moral equivalent

Installing a Package on a Cluster

- For NFS rooted clusters, or those with automated replication (InterMezzo, rsync, etc.), just install into the image. Viola, all nodes have it.
- For local disk clusters, it comes down to a hodgepodge of shell scripts to replicate the changes.
- Or, the administrator installs each one by hand...
  - Admit it -- you've done it, too.

What Else Makes a Cluster Special?

- Single point of administration
- Power management
  - May be fancy BayTech RPC's
  - Or simple APM
- Tightly couples
- Interconnect
- Jobs scheduling
- (stay tuned for section on clusters....)

Cluster distribution ideas

- Install diag node with necessary servers
  - dhcpd / bootpd
  - NFS server ready
  - tftpd server (for network configuration)
  - OpenLDAP
- Install read-only NFS root-able cluster node directory separate from diag node root filesystem.
- Node management scripts are required
  - Out of the box, ready to run.

Who is doing it right?

- Sandia's Cplant has made the investment in hardware to have a managable cluster with thousands of nodes.
  - Multiple levels of hierarchy with separate networks.
  - Power control hardware
  - Division of labor
  - Hardware that support diskless network booting
- SETI@Home Offloads all management to the end user.
  - Which works if your users are capable.

But these are not typical clusters...

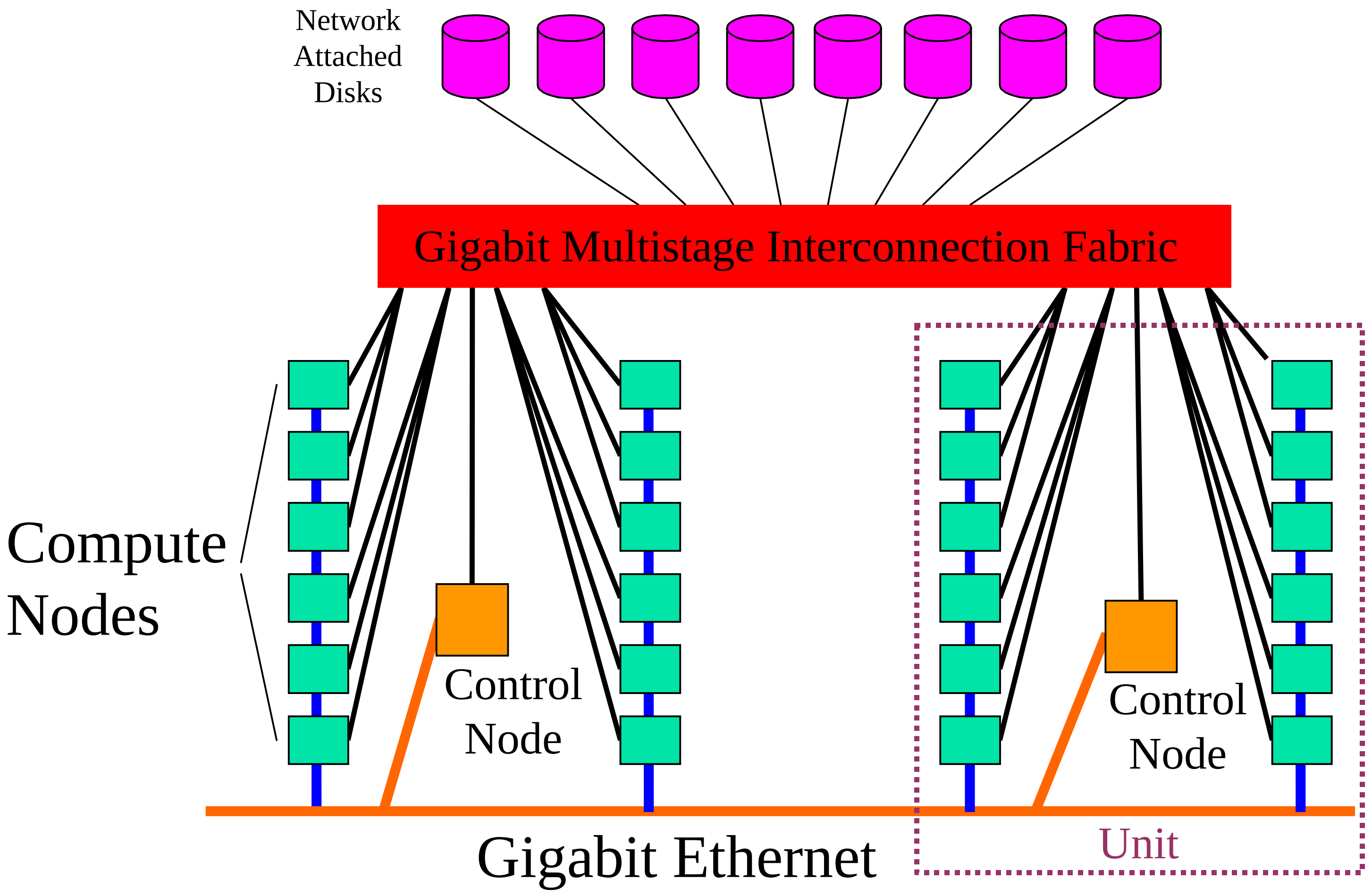
- Money is a factor
- Space is a factor
- User community is a factor

Typical clusters are...

- Few tens of nodes
  - Where 2 nodes is close enough.
- Made with standard hardware
  - From CompUSA, Office Depot, etc.
- Used by regular programmers
- In a closet somewhere

Overview:
Linux for Scientific Clusters

Large Linux Cluster Architecture



Linux Clusters are Hot Tech

- An IDC report on Technical Computing reported:
  - Of 201 industrial sites, 36% expected to purchase a cluster in the next year. Futhermore, Linux was the OS of choice, but NT was also in use
- Gartner Group predicts:
  - Cluster ratio for UNIX will be 50% in 2003 (current ratio is 20%)
- D.H.Brown: There are 4 areas where Linux been proven superior: ISPs, entry-level computer networks, special devices such as web appliances, and clustering
- www.google.com uses 4000+ Linux nodes for one of the world’s hottest search engines
- In Feb, Akamai had network of over 2000 Linux machines

But Wait... A Disclaimer:

Linux Clusters Cannot Replace All Supercomputers

- Aggregate memory bandwidth (STREAM) of NEC 16-processor is about 1800 times that of 600Mhz PIIIs
- Latency from one motherboard to another is still about 1000 times slower than to local memory
- LINPACK benchmark numbers are nearly meaningless (more on this later)
- There is still a lot of missing software

Why Linux?

- The reasons are practical, not technical.
  - Open source
  - Support for many processor families
  - Good environment for developing cluster infrastructure
  - Huge development effort means rapid improvement, support for new hardware.
  - Commercial applications
  - Talent pool
- The first three items are advantages over Windows/NT

Are Linux Clusters Ready?

- Clusters are relatively cheap
- People are relatively expensive
- A Linux cluster makes a **very** good *personal* supercomputer
- Today’s clusters are often weak as general purpose multi-user production machines
- Management of clusters is still very difficult
- Building such a cluster requires planning and understanding design tradeoffs
- We still hype them, making unrealistic expectations

Cluster Benchmarking
Lies, Damn Lies, and the Top500

Vendor Published Linpack, Latency, and Bandwidth numbers are worthless

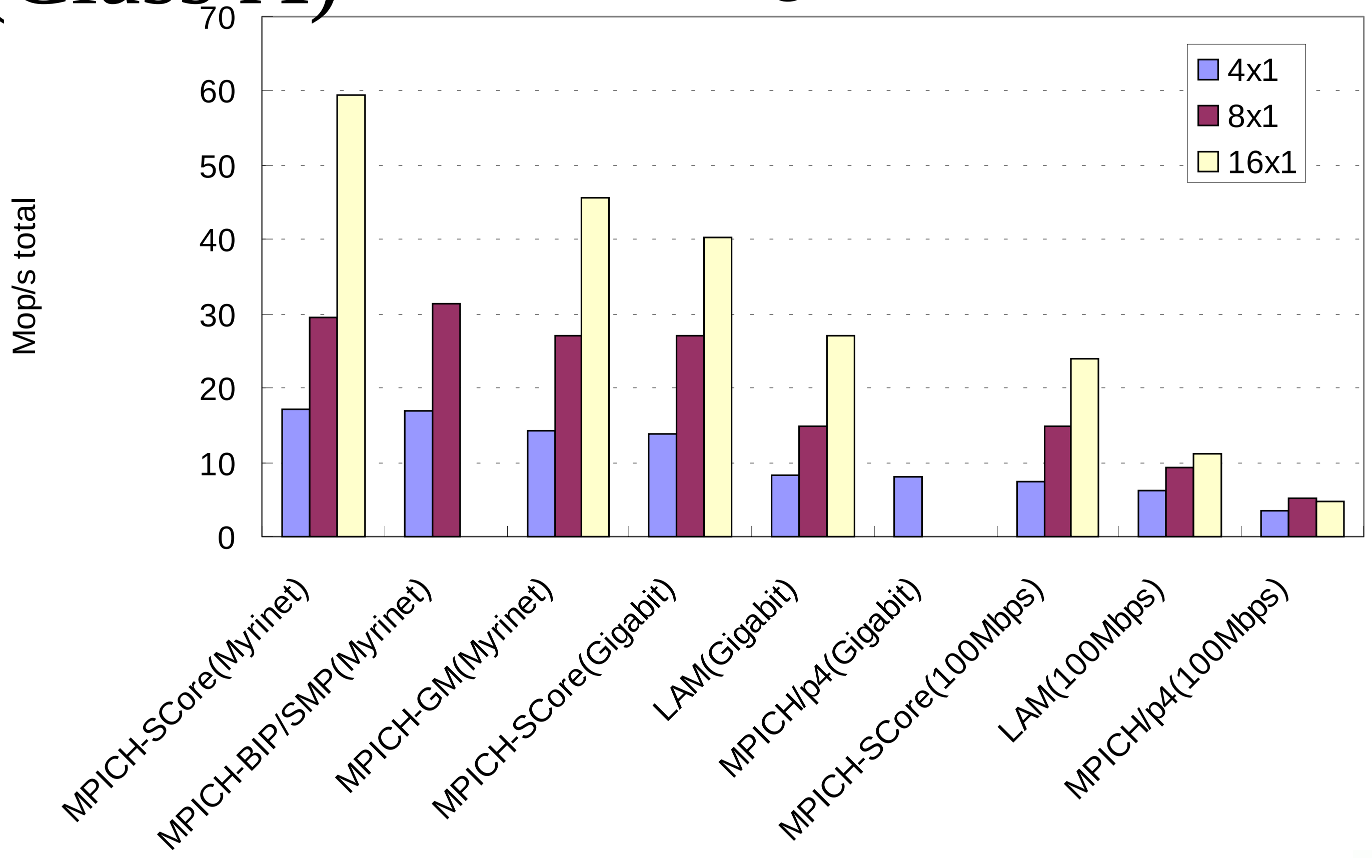
- Make MPI zero-byte messaging a special case (improves latency numbers)
- Convert multiply flops to addition, recount flops
- Hire a Linpack consultant to help you achieve “the number” the vendor promised
- “We unloaded the trucks, and 24hrs later, we calculated the size of the galaxy in acres.”
- For $15K and 3 rolls of duct tape I built a supercomputer in my cubicle....

RWCP Japan NAS Benchmarks

Courtesy Yutaka Ishikawa at RWCP

IS (Class A)

16 single Pentium-III, 500MHz

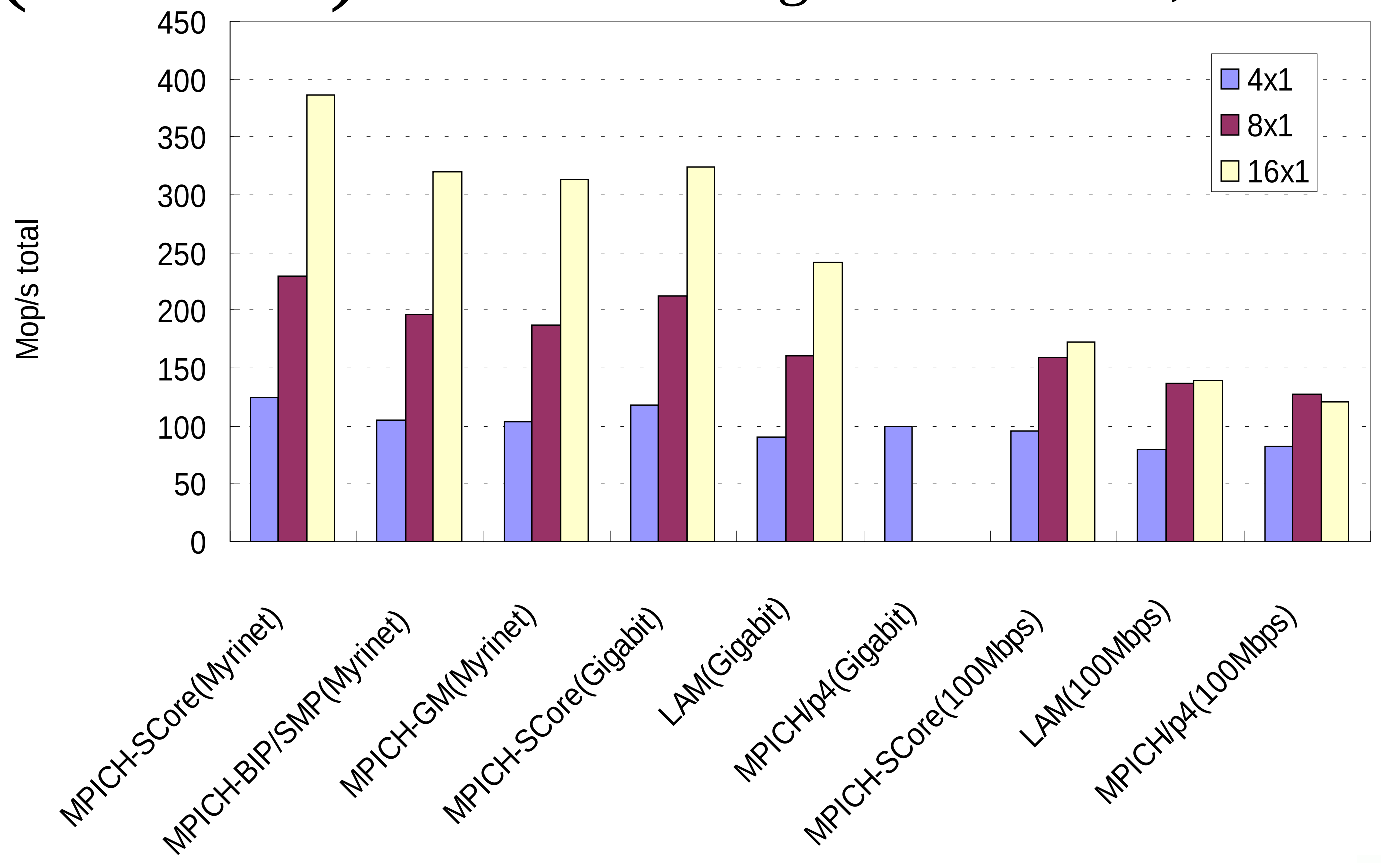


RWCP Japan NAS Benchmarks

Courtesy Yutaka Ishikawa at RWCP

CG (Class A)

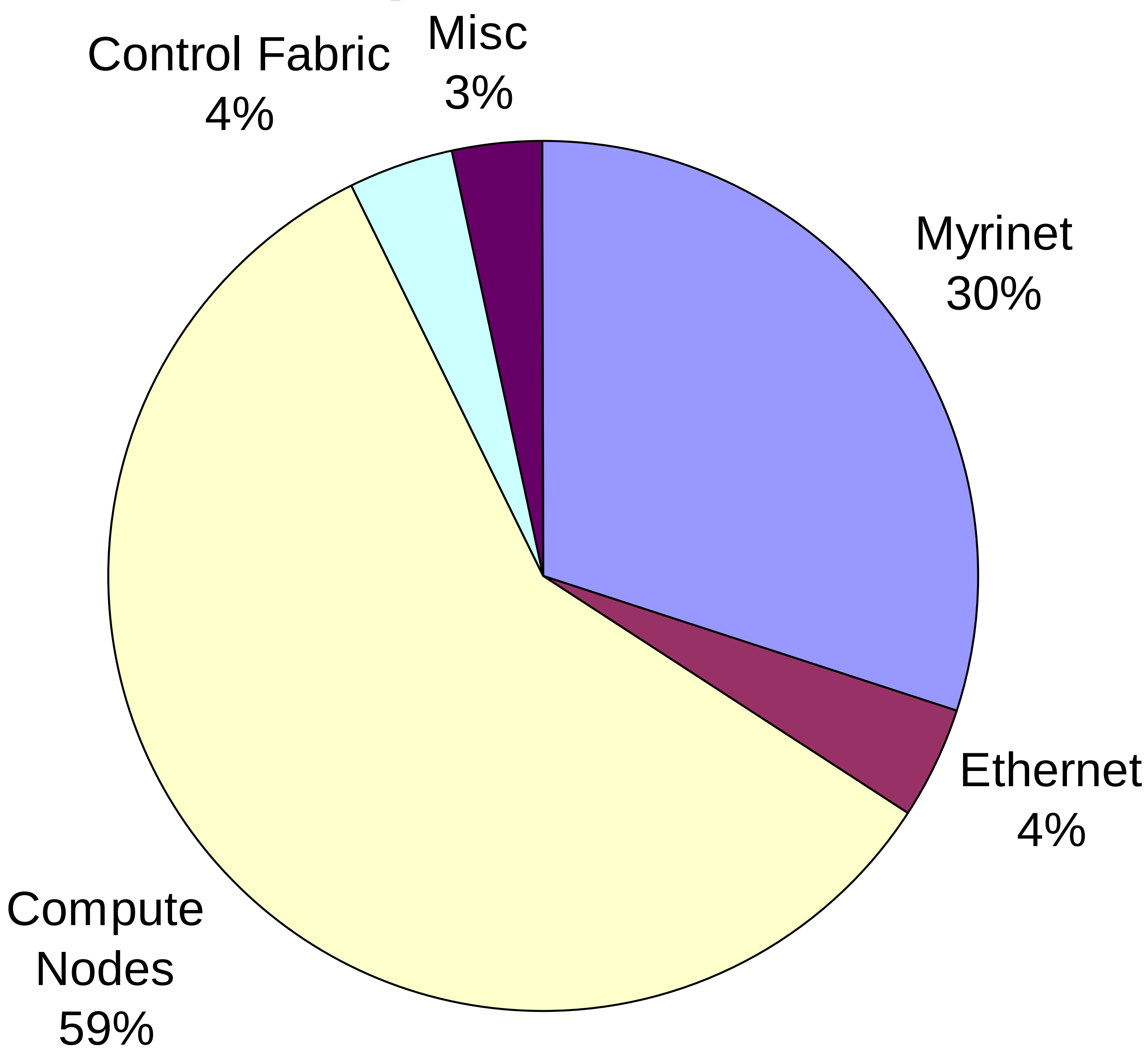
16 single Pentium-III, 500MHz



Honest Cluster Costs: Publish the Numbers!

- We need a *Consumer Reports* for clusters
- Linux is free, but what about everything else?
- How many sysadmins and programmers are required for support?
- What are the service and replacement costs?
- How much was hardware integration?
- How many users can you support and at what levels?
- How much was the hardware?

Example Cluster Costs



Thinking About Your Own Cluster?

Some Considerations...

Start with Solid Planning Write these Documents

- Overall System Diagram
- Life cycle & expectations
  - Machine lifetime (3 years?)
  - Delivered cycles (90% uptime?)
  - Funder's expectations...
- Purchase Timeline
- Installation Timeline
- Network Diagram
  - 1 diagram per network.
  - Show external connectivity.
- If you have them:
  - Serial Infrastructure.
  - Power Infrastructure.
- Racks Diagram
  - What goes in which rack where.
  - Cable management plan.
  - Floor plan for racks & cables.
- Power Planning
  - Which racks are wired on which circuit.
- Installation Guide
  - What goes in which rack where, in what order.
  - Cable location, labeling.
  - See MCS tools page for example.

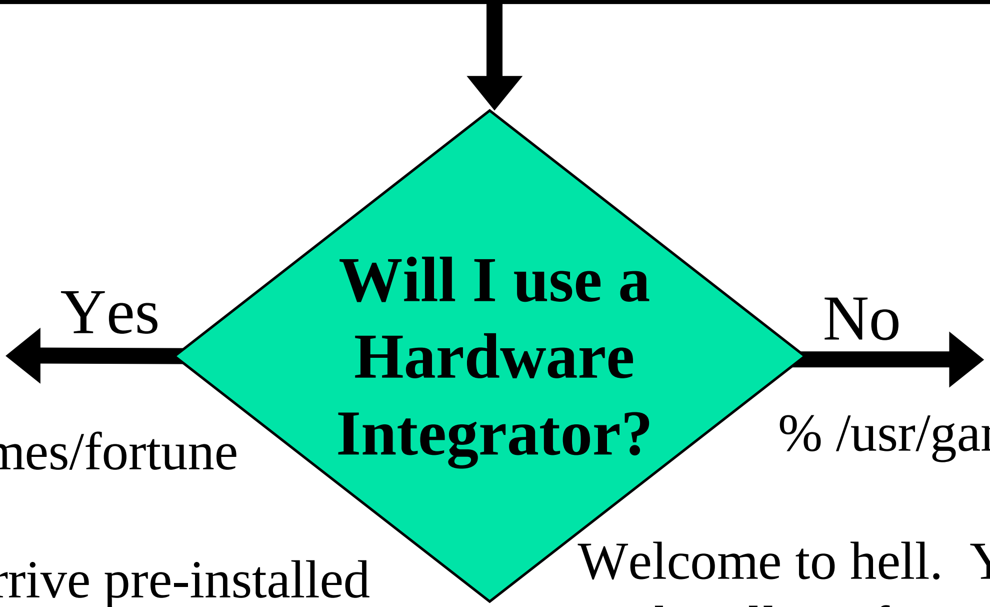
Important Considerations (1)

- User profile and infrastructure (!compute_nodes)
  - Home directory disk space, NFS servers
  - Compile servers
  - Scratch disk, visualization, tape archives, etc
- What is the physical space?
  - Raised floor?
  - Air cooling units, and their ratings
  - AC Power conditioning
  - How tall can racks be? Will they fit in the doors?
  - How much AC power is available, how many circuits? Plug types?
  - Environmental Safety
  - etc...

Important Considerations (2)

- How much do you have to spend?
  - Software licensing
  - Hardware
  - People time
    - Administrators
    - Developers (new tools)
    - Software Engineers (help port applications, tune, hold hands)
- THEN contact the vendors and request non-binding information (save everyone's time)
- Based on what the vendors can deliver, iterate again on proposals and design documents.

The First, Most Critical Choice:



% /usr/games/fortune

Your Cluster will arrive pre-installed with Linux, after posing for pictures, you will spend your time on the hard part, software integration and administration.

% /usr/games/fortune

Welcome to hell. Your gnashing of teeth will not fix your compatibility problems, and your new skills for attaching IDE cables is preparing you for a career at BestBuy. Remember, your time is free, and caffeine will help you get more done.

What Hardware Integrators can Provide

- A single, unified warranty. Failed nodes get sent to integrator.
- Integrated components, fewer surprises:
  - HW compatibility, physical form, heat, burn-in, etc.
- Preinstalled Linux, complete with appropriate drivers
- A small loaner system (or remote access) to test and benchmark the applications you really care about
- On-site installation
- Suggestions for system components
- Experience! ("we see power supplies die once a month on those systems")
- Coffee mugs, T-shirts, and free lunches

Talking to the HW Integrator

This is not a friendly place for the novice

- Remember that the integrator is assuming risk in exchange for money.
  - Integrating the warranties
  - Delivery of components in time for the cluster (e.g. memory price spike)
- Do your research, and know the prices for components.
- Only 1/10th of the detail demons:
  - Who installs what?
  - Acceptance tests
  - Who pays for return shipment of dead nodes during the warranty?
  - How will disputes be resolved?
  - Insist on uniformity (serviceability) where possible (Control node has same motherboard as compute nodes), etc.
  - Specify manufacturer part number for components to avoid confusion
  - Specify “automatic” rules for late delivery, etc

Thinking About a Linux Cluster?

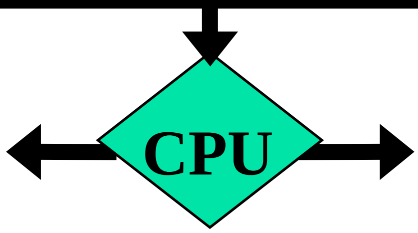
- Try not to build one yourself, unless your real interest is research into designing cluster system software and managing hardware
- Buy from an integrator:
  - IBM, HPTi, SGI, Compaq, VA Linux, Linux Networx, etc
- Learn about the details.... It is not yet safe to just ask for cluster to be delivered...
- Remember, Lies, Damn Lies, and Linpack: Test your app. on a cluster of similar architecture
- Ask about software.. (more later)

Some Advice: Cluster Architecture and Hardware

Basic Choices and Components

- Node type
  - Alpha/Pentium
  - uniprocessor/SMP
- Networking
  - TCP/IP connectivity
  - fast messaging (Myrinet, etc)
- Form factor
  - racks (yes!)
  - shelves (no)
  - unit size (1U/2U)
- Power Management
  - kludge
  - net-addressable controllers
- Basic Requirements
  - Space
  - Cooling
  - Power
- Console management:
  - BIOS/EMP/NVRAM
  - VGA
  - serial

The Heart of the System: Compute Nodes



	PIII / Xeon	Alpha 21264
■ Performance:	Xeon/500	21264/677
■ STREAM copy:	188 MB/Sec	1087 MB/Sec
■ SpecFP95:	15.1	48.4
■ Peak MFLOPS:	500	1354

- NAS results: http://www.nersc.gov/research/ftg/pcp/performance.html

■ Cost:		
■ Dual CPU	~6K (512K L2) ~13.5K (2mb L2)	~$15K (4mb L2)
■ Software:	lots	some

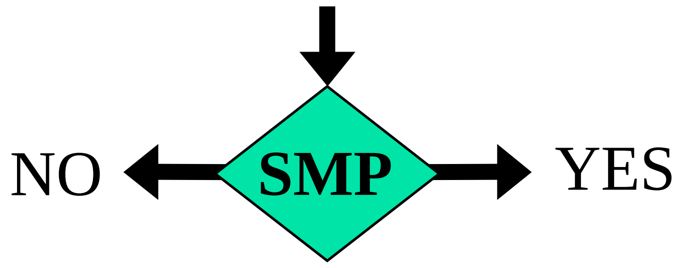
Compute Nodes: Other Considerations

- The motherboard implementation can be very very important:
  - PCI compatibility issues
  - Dual PCI
  - PCI performance (64 bit PCI, clock, etc)
  - SMP memory performance
  - Memory controller chipset, ECC, speed
  - BIOS / NVRAM / EMP / Power
- Integrated components:
  - 100BT, VGA
- Node style issues:
  - No local disk (net boot, NFS root)
  - No floppy, cdrom (net boot)
  - No VGA (serial console)

Yuk, this is all so messy!

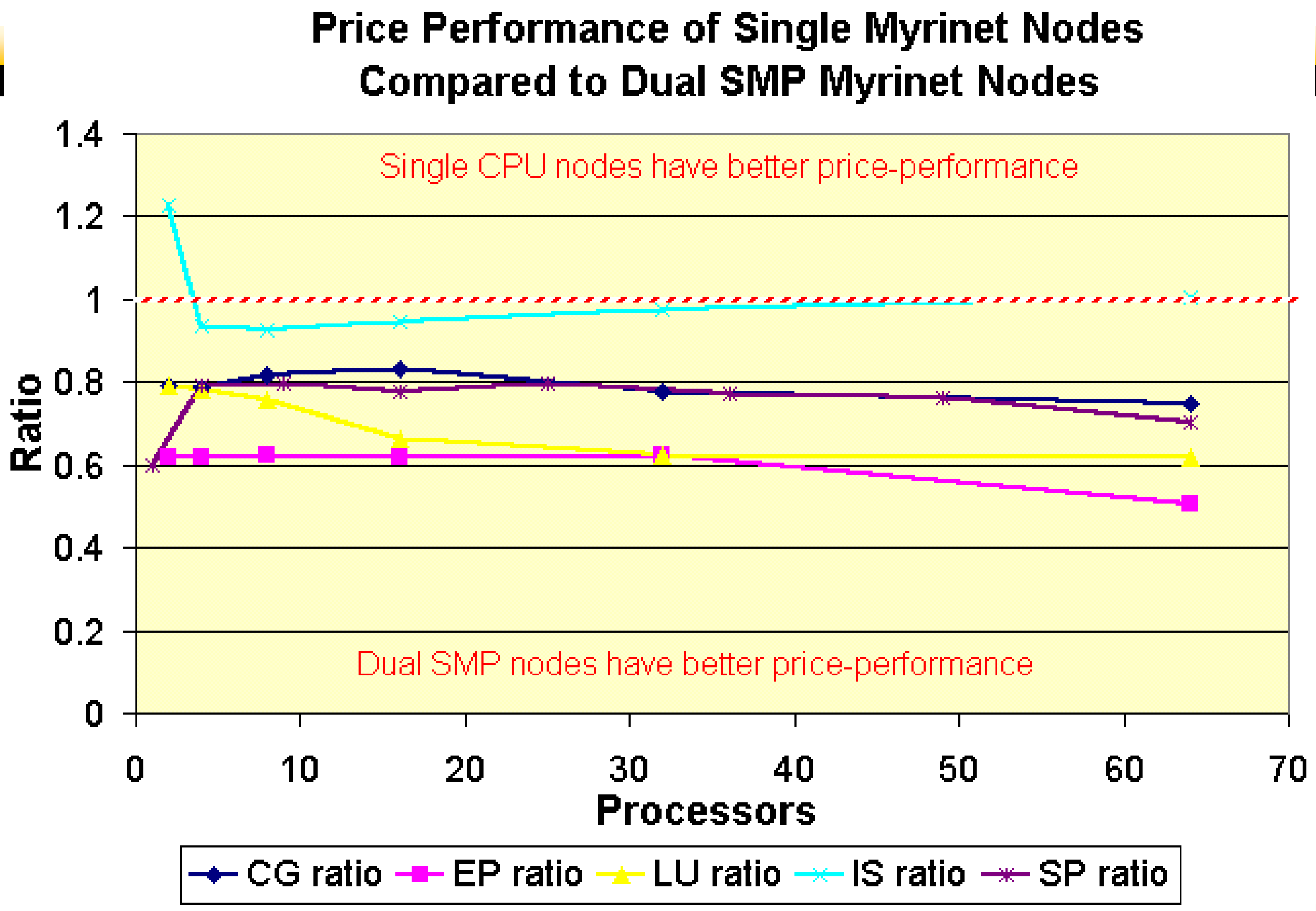
SMP or Uniprocessor Nodes

The obvious: Cost per CPU for a 2-way SMP cluster is less

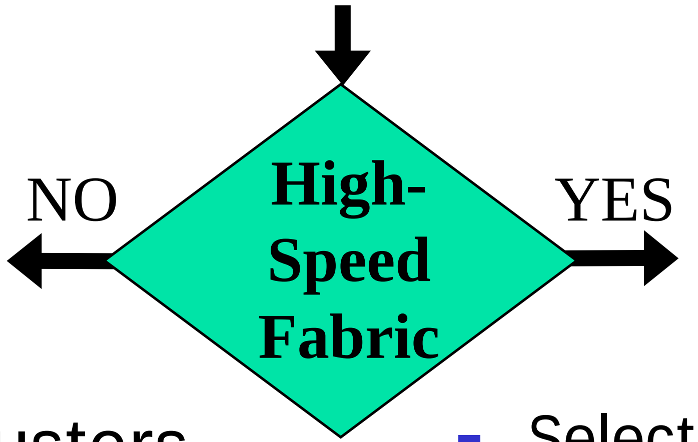


- Disadvantages to SMP:
  - Significantly complicates many layers of software
  - Overall memory bandwidth per CPU is usually reduced
  - Memory can be more expensive (high density usually required)
- Advantages to SMP:
  - Cost of fast interconnection fabric split over 2 CPUs
  - Compact form
  - Price/Performance can be better than uniprocessor nodes (next slide)

SMP Price/Performance



Networking



- Build small clusters
- Do mostly embarrassingly parallel computation

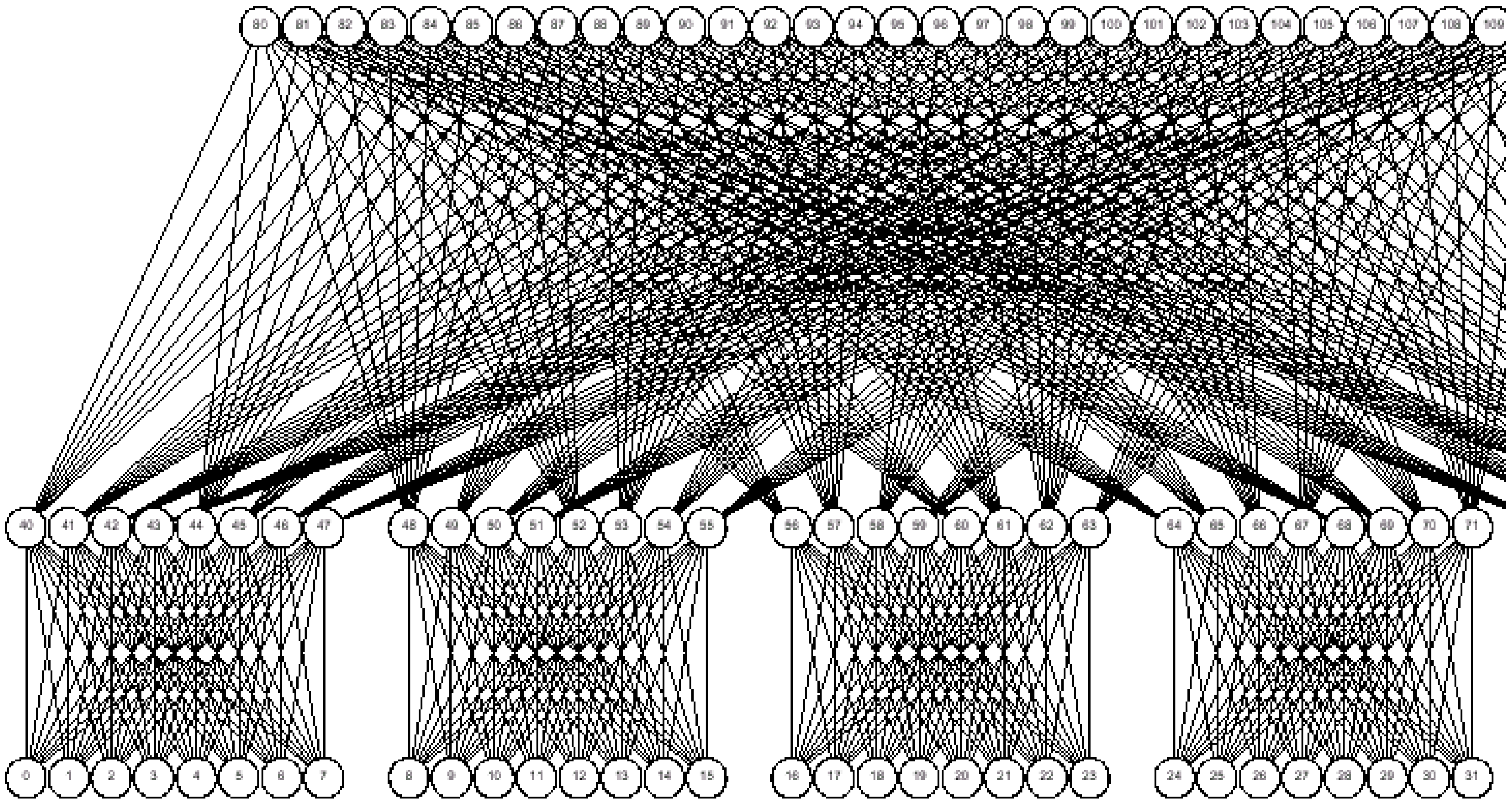
- Select scalable tech:
  - Myrinet, GigE, Giganet, Quadrics, etc
- Evaluate fast MPI
  - SMP?
- Open wallet, listen to giant sucking sound.

Myrinet 2000

- Copper media signaling of:
  - 2.5GbE, 2xFC, & 1x Infiniband
- ~9us GM latency
- 2+2 Gb/s links, Copper (HSSDC) and Fiber (850nm VCSEL)

Myrinet 2000 uses 2.5+2.5 GBaud -- 2.0+2.0 Gbit/s data rate after 8b/10b encoding

Common Myrinet Topology



Control Fabric (1)

- Power Management Options:
  - None. Physically walk to the machine room when you need to cycle a node.
    - Save money, burn more calories, live close....
  - Use a standard, commercial net-accessible AC controller
    - BayTech http://www.baytechdcd.com/
    - APC http://www.apc.com
    - Cost: ~$60/port
  - Compute nodes must have right "power button"
    - Bad: momentary contact button must be physically touched
    - Does On/Off switch retain state?
  - Other: Wake on LAN, etc

Control Fabric (2)

- Linux Console Management:
  - Shopping cart, bungee cord, monitor, and VGA cable
  - Very small clusters can use VGA switch
  - Serial console:
    - Physically connect all the serial ports into the control node via multi-port serial card
    - Comtrol’s Rocketport: www.comtrol.com
    - Cyclades: www.cyclades.com
    - Requires software infrastructure: conswatcher, chex
  - Emergency Management Port (EMP)
    - BIOS access, power/reset, NVRAM critical event log, sensor data
    - Software from VA Linux:
      - ftp//ftp.valinux.com/pub/software/vacm/

Concrete Example

- Scalable unit (32 compute nodes (64 CPUs), one Control Node)
  - 1 control node
    - Intel L440GX, PIII/500Mhz/512K, 256MB RAM, 18GB Quantum, CDROM, floppy, integrated 100BT
    - 2 32-port RocketPort serial cards
    - 4 serial port distribution panels
  - 32 compute nodes
    - Intel L440GX, Dual PIII/500Mhz/512K, 1 GBMB RAM, 9GB Quantum, floppy, integrated 100BT
    - Myrinet card
  - 2 racks
  - 1 Summit48 Enet switch with 2 GigE uplinks to PowerRail GigE switch
  - 6 BayTech power controllers (48 ports)
- Myrinet switch fabric to support all the nodes

The BIOS

- The BIOS controls what the machine does when it powers on.
- The Alpha BIOS is very powerful, very useful, and remotely accessible.
- The average PC BIOS is harder to work with than the average UNIX workstation BIOS:
  - No command line, No remote access.
  - No OS-level access, at least not for Linux. Yet.
- BIOS configuration settings are set at the factory.
  - You might be happy with what you get.
  - You might not.
- Options for fixing the BIOS:
  - Plug in a monitor and keyboard.
  - Cope.
  - Hope that any of several Linux development projects comes through.
  - Use one of the very few BIOSs with serial support.

Linux Networking for Clusters

Internal Cluster Networking

Not all IPC is equal

- Cluster services
  - YP/DNS
  - Monitoring
  - Heartbeat
  - Process startup/management
  - Batch system
- Filesystems
  - NFS
  - Parallel filesystem
- MPI communication

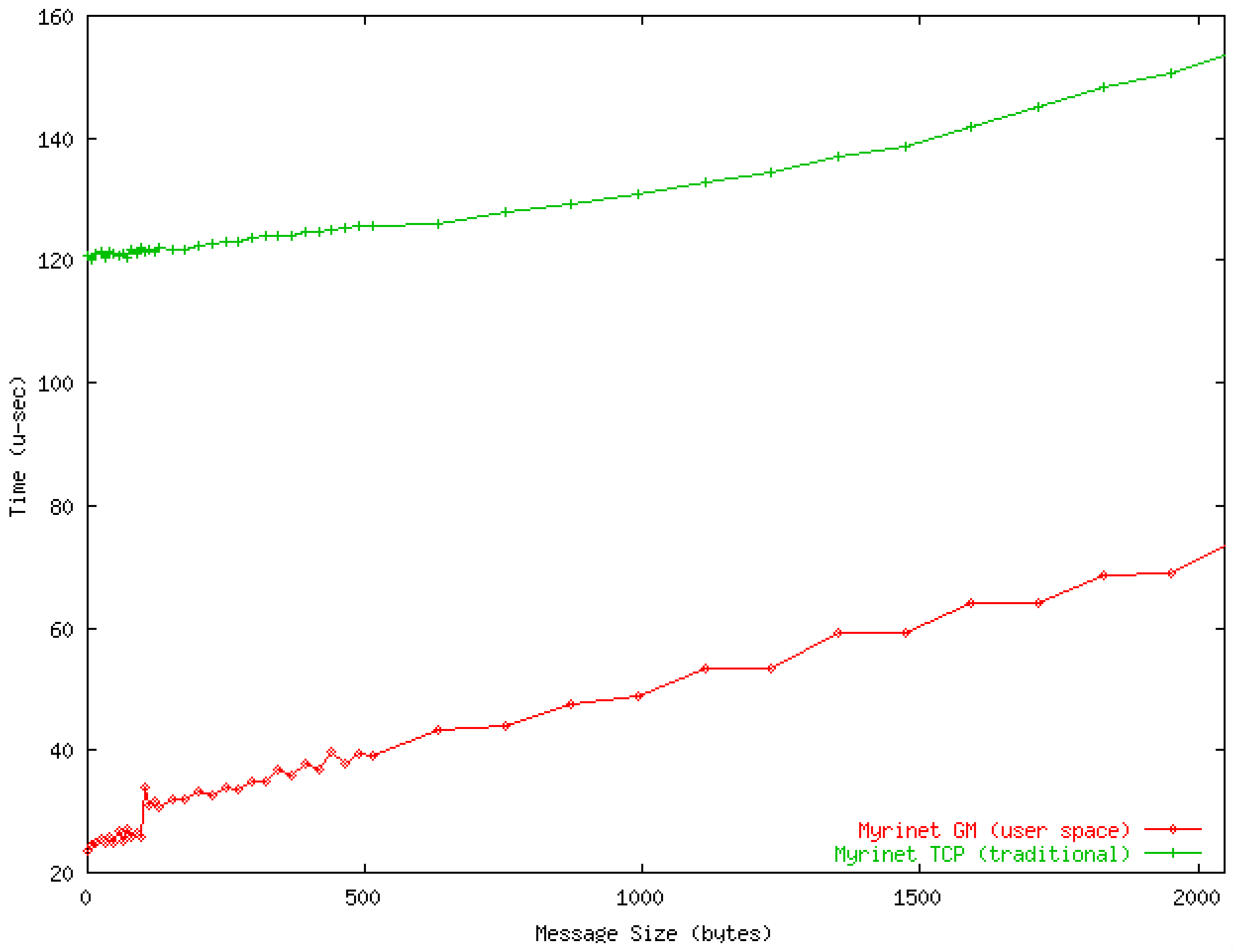
Communication Patterns

- Non-parallel computing
  - Usage occurs in spurts
  - Spurts from different machines happen at random times
  - Traffic pattern is typically client-server
- Parallel computing
  - Usage occurs in spurts
  - Spurts are usually synchronized (all machines talking at once)
  - Some traffic is client/server (e.g. yp/nfs) but bulk is many to many.

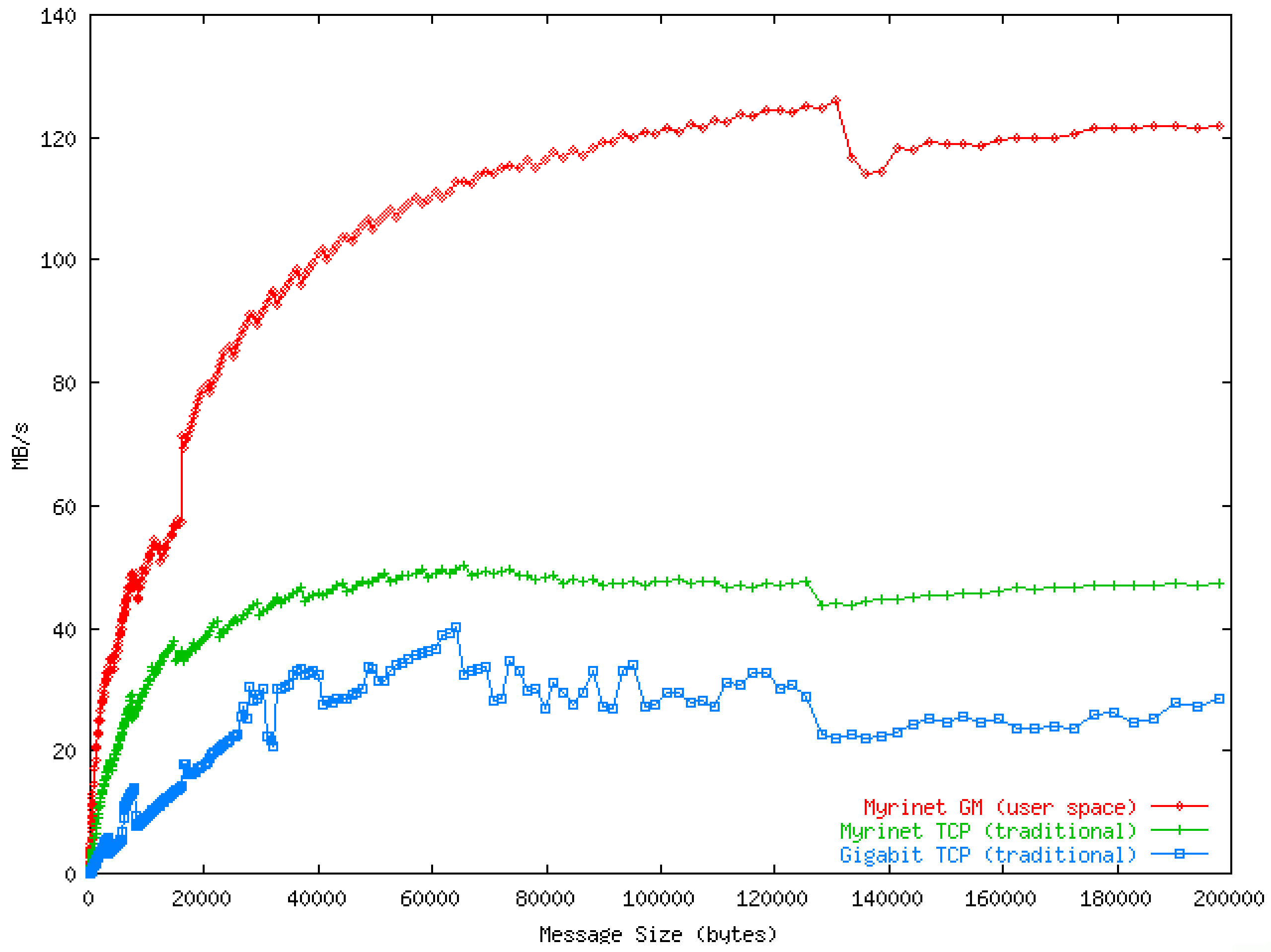
Network Performance

- Real performance is a combination of several important factors
  - NIC, media, duplex, switches, contention, hot-spots, etc.
  - Software (TCP/IP Stack, VIA, etc)
  - CPU Utilization (benchmarks can be very misleading here)
  - System parallelization, asynchronous data movement
- There are two important network usage patterns for clusters
  - Scientific message passing for parallel programs (MPI/PVM)
    - Users want:
      - 1) low latency, high bandwidth, no network hotspots
      - 2) Asynchronous data movement, low CPU cost
  - SAN/WAN (TCP/IP)
    - Fast disk, tape, grid, home directory NetApp access

MPI Latency -- User-Space v.s. Traditional



MPI BW -- User-Space vs. TCP



Measuring MPI Performance

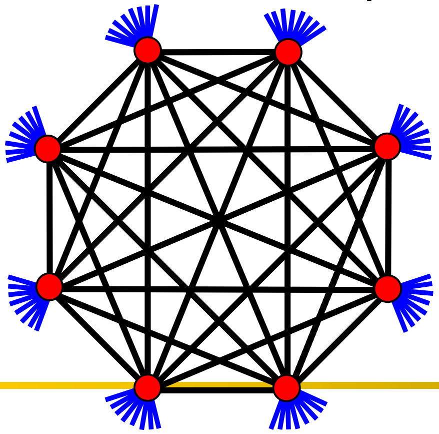
- Some Not-So-Standard Definitions:
  - **Latency**: The minimum time to get a zero-length message from A to B
    - Measurement Practices:
      - One-way latency
        - Rapid fire thousands of messages, get a single ACK
      - Half ping-pong
        - Do message ping-pongs, then divide by two
  - **Bandwidth**: The communication capacity (measured in bits/sec)
    - Measurement Practices:
      - Pre-posted Recv()
      - Report maximum for enormously large message
      - Subtract latency from timings

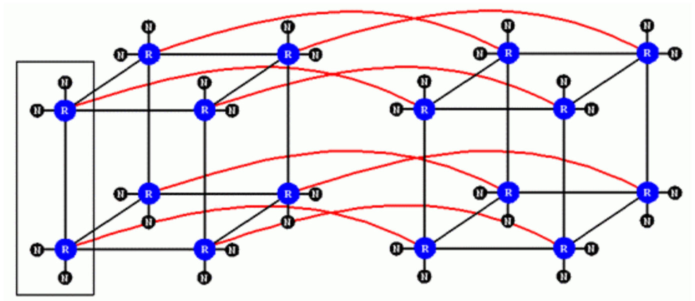
The Importance of Switching

- Parallelism in the network
  - Many-to-many can be done in parallel if network supports it. Need parallel network!
  - In worst case (e.g. FFT) N/2 processes sending/receiving to/from N/2 processes. Bisection bandwidth should be N*(bandwidth per stream)
  - On Fast Ethernet, switches also enable full duplex transmission
- Routing
  - MPI communication should go through switches, not nodes
  - Avoid IP routing by cluster nodes for internal traffic.
  - Fast protocols require switched-based routing
- Vendor numbers can be very deceiving

Bisection Bandwidth

- Bisection Bandwidth is used to report the connectivity of the entire cluster, in contrast to “latency” and “bandwidth” which are measured only between a pair of hosts.
- Theoretic Bisection Bandwidth is calculated by dividing the machine in half, using the partition of worst connectivity, and summing the bandwidth of the links between the halves.
- Interconnection topologies have “Full Bisection Bandwidth” when the number of links between any two halves of the machine is N/2.
- Bisection Bandwidth is important for programs that do global communication





Ugh! Number Overload! HELP!!! What does it all mean?

- The overall interconnection performance of your cluster is not easily characterized by two or three scalars.
- There are many ways to mislead with “latency” and “bandwidth” numbers for a given network. Run the tests yourself.
- Bisection Bandwidth is very important for some applications
- The only true measure:
  - How will my application run? How will my application scale?
- TCP/IP is generally a poor transport for MPI-style messaging
- “User-level” messaging or “OS Bypass” are techniques for improving the performance of messaging
- CPU Utilization and asynchronous data movement can be important, but remains largely unmeasured

“Legacy” Networks

- These networks are fine for most cluster services.
  - Scaling is quite limited. A single switch can have full bisection bandwidth, but beyond that generally not possible.
- Fast Ethernet
  - 100 Mb/s; switches available up to about 100; Switches essential to support full duplex.
  - Look for about $150 a port, total (including switch)
  - Gigabit uplink to servers is useful to avoid many-to-one problems.
- Gigabit ethernet
  - 1000 Mb/s; 30-40 MB/s in practice. Latency the same as Fast Ethernet. switches available up to about 64.
  - Has been very expensive. Now look for $1000 a port (including switch) with new copper-based NICs/switches.

User-space communication

- Time spent transferring data is time not spent computing.
  - TCP has many overheads.
  - For best performance, need a protected user-space communication system (kernel not involved in communication).
  - User-space communication requires special hardware support.
  - Until recently: Only Myrinet; many APIs
- **Virtual Interface Architecture**
  - New industry standard; used by Infinband
  - Single API for any network.

New Gigabit Networks

- Several new networks provide
  - Support for user-space communication
  - Ability to connect switches to build arbitrarily robust networks.
  - MPP-like performance
- **Myrinet**
  - Made by Myricom, Inc.
  - >100 MB/s; soon > 200 MB/s
  - The only choice for building very large networks (large switches, proven scalability)
  - Support user-space communication software is called “gm” -- latency not wonderful. There is a basic MPI over GM . Supports TCP simultaneously.
  - Programmable processor on NIC
  - Expect to pay ~$1500/node
  - VIA support coming soon.

New Gigabit Networks, Continued

- **Giganet**
  - Native VIA network. Does not support TCP (yet)
  - 100 MB/s; 8us latency;
  - Currently small switches only.
- **Servernet II**
  - Native VIA network. Does not support TCP (yet). (?)
  - Successor to Servernet I used in Tandem non-stop servers.
  - Available 1Q2000.
- **InfiniBand**
  - Will be usable as a network.
  - Many implications for cluster computing.

Why You Should Use Multiple Networks

- You need a high-performance network, but:
  - Parallel programs are very sensitive to perturbations (everything is an O(1) effect).
  - High performance networks tend to be fragile
  - High performance networks may not be available before routing is set up by software
- Redundancy in critical infrastructure is good. (E.g. have Ethernet as a backup even if Myrinet if your primary).
- Security -- next slides.

Private Networks

- Three sets of network addresses are reserved for anyone’s use. Just don’t let packets with these addresses out of your cluster!
  - 10.0.0.0 - 10.255.255.255
  - 172.16.0.0 - 172.31.255.255
  - 192.168.0.0 - 192.168.255.255
- Security. Complete trust inside a private network is possible.
  - Firewalling/packet filtering less complicated.
  - Some tools rely on complete (rsh-style) trust.
  - Many services do not need to be exposed outside the cluster
- Allocation of IP addresses is easier. Automatic assignment to nodes is easier.
- No interference from traffic external to the cluster.

But...

- Users want connections to the outside world directly from cluster nodes.
- Users want high performance

- Several solutions: dual interfaces in some nodes, masquerading, gateways, etc

Some Advice: Linux Software

Or: Reminders why you need an integrator

Why HPC Linux Clusters are Still Immature

- Compilers are good!
- Message passing
- Scheduler
- Debugger
- Performance tuning
- File System
- Parallel I/O
- System management

IPC For Parallel Programs

- MPI
  - Industry standard. Your code will work everywhere.
  - Allows scalable programs
  - Designed to be integrated into production environments
- PVM
  - Not quite a standard
  - Performance limitations
  - Designed for “personal parallel computers” – difficult to integrate into a production system as it includes an “operating system”
- Virtual Shared Memory
  - No standards.
  - Research area.
  - No data on production environments.

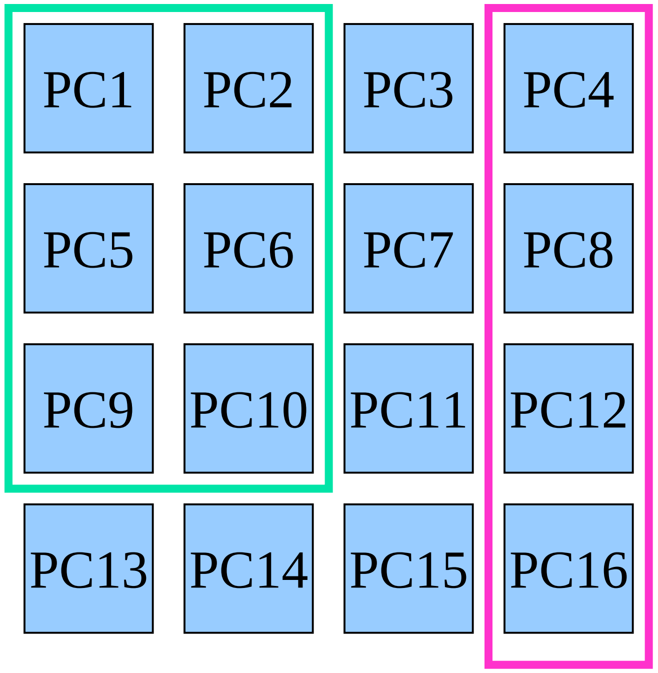
MPI Implementations

- MPICH from ANL/MSU
  - http://www.mcs.anl.gov/pub/mpi
  - Different transport layers available

- LAM from Notre Dame
  - http://www.mpi.nd.edu/lam
  - Designed for personal supercomputers. Some problems integrating a batch system and with an external process manager.
  - Faster and easier to use than MPICH for TCP-based communication and no batch integration.
  - Good support for code development.

Space Sharing

- Tightly coupled parallel programs must have dedicated compute resources.
  - Static load balancing in most parallel apps means loadave of 2 on one node of N reduces performance by at least 50%, not O(1/N).
  - Synchronization delays reduce performance further.
- Standard solution: space-sharing.



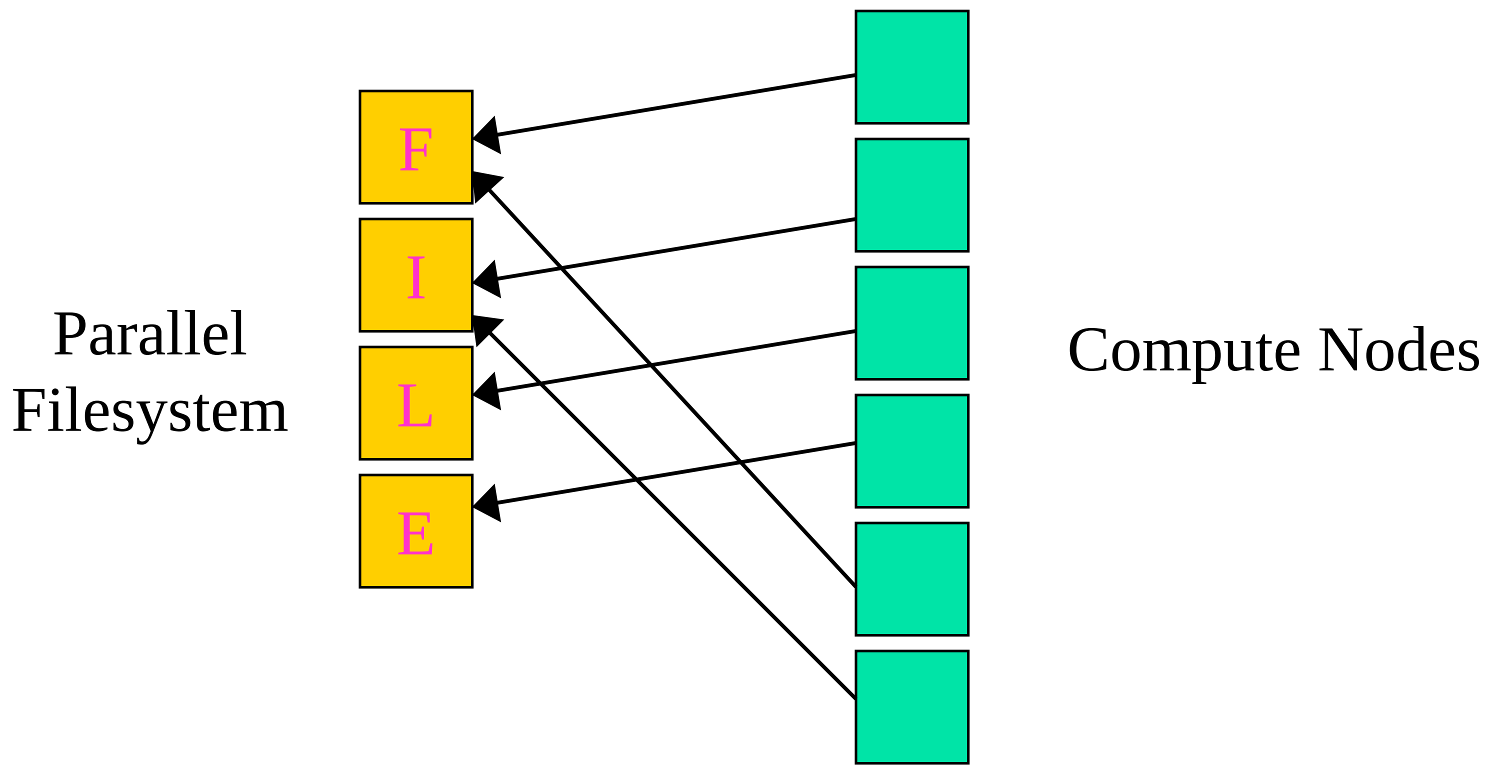
Things to Know About ch_p4 (1)

- Does not do space sharing.
  - Can only be fixed by a scheduler with “global” knowledge (PBS with hooks for example)
  - Decides where to run processes based on a hostfile in an obscure format
- By default starts up the first process on the “mpirun” node.
  - Not the right thing if you run from an interactive node.
- Orphaned processes
  - “kill –9” to the wrong process can leave others stranded, spinning
  - Something has to kill off those processes automatically (PBS prologue/epilogue)

Things to Know About ch_p4 (2)

- Slow startup
  - Based on rsh/ssh
  - Approx 1 task/sec
  - There is not yet a general purpose workaround for this.
- TCP disconnects under severe congestion. No known workaround.
- Relies on command line args in process startup. Size of command line grows with number of nodes. Large clusters need workaround.

Parallel I/O



MPI-IO and ROMIO

- Doing efficient parallel I/O is tricky.
- Need an efficient software interface *and* a parallel/scalable filesystem.
- MPI-2 I/O is the standard interface for cooperative parallel I/O
- The best/only implementation for clusters is ROMIO.
  - http://www.mcs.anl.gov/romio
  - Unfortunately there is not yet production level parallel filesystem to run it on top of!
  - Best bets are
    - NFS
    - PVFS: http://ece.clemson.edu/parl/pvfs/

Compilers

- Buy one

- C
- C++
- Fortran/90
- OpenMP (C and Fortran)

Compilers/x86

- C: gcc is fine
- C++
  - Newest gcc/egcs is good.
  - KCC (from Kuck and Associates -- www.kai.com -- has been necessary in some cases. Includes OpenMP support.
- Fortran
  - Portland group is the most widely used. Includes OpenMP support.
  - Other available from: Absoft, Fujutsu, NAG
  - g77 only as a last resort. (slower, limited syntax).

Compilers/Alpha

- C
  - gcc is ok.
  - Compaq C compiler is excellent (too bad Compaq is only a hardware company now...)
- C++
  - g++/egcs is the only option
  - Compaq compiler being ported, it may be done?
- Fortran
  - Compaq Fortran also good. Much faster code.
  - Don’t even think about g77.

http://www.unix.digital.com/linux/software.htm

Basic Math libraries

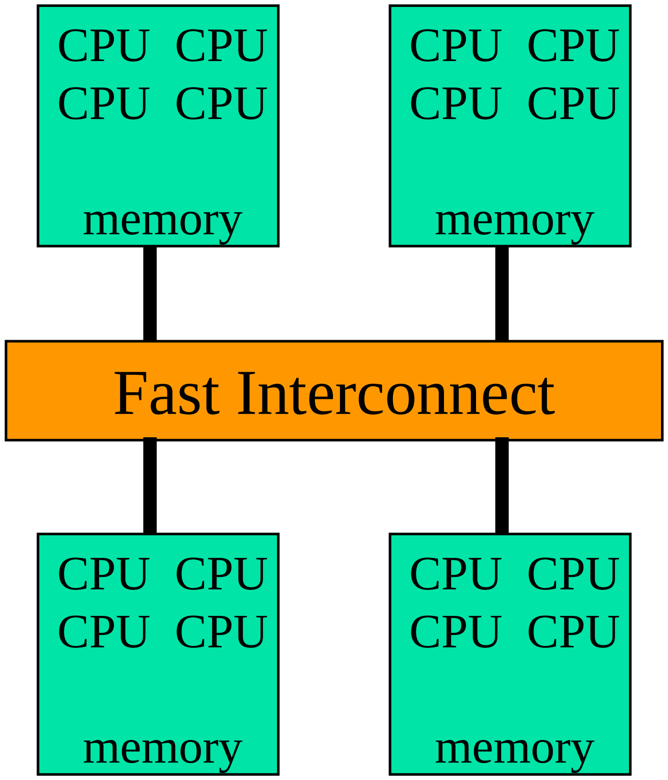
- Optimized BLAS for x86 available
  - http://www.cs.utk.edu/~ghenry/distrib/
- FFTs available in FFTW package
  - http://www.fftw.org
- Compaq math libraries for alpha
  - cpml: a drop-in replacement for libm.a
  - http://www.unix.digital.com/linux/cpml/index.html

Debuggers

- Slim pickings here.
- The best parallel debugger, but also very expensive, is Totalview.
  - http://www.etnus.com/products/totalview/index.html
- See also the Data Display Debugger. It’s not parallel, but it does have a nice interface.
  - http://www.cs.tu-bs.de/softech/ddd/

Programming Models for SMP Clusters
Warning... Religious Topic Ahead

- The Simplest Option:
- Ignore the architecture: MPI Everywhere
  - Advantages:
    - Very simple, very portable
    - Relatively easy to debug
  - Disadvantages:
    - Almost always extra memory copies on an already limited memory bandwidth node
    - Can have performance problems on cache-coherent shared memory architectures
    - Ignores the hardware support for shared memory



Threads & MPI

- Methodology:
  - 1) Begin with a standard MPI program, add thread parallelism by hand to sections of the ‘node code’. [David Bader (dbader@eece.unm.edu)]
  - 2) Redesign the code for two levels of parallelism, and then decompose across nodes for msg passing, and threads for shared memory
- Advantages:
  - Pthreads are fairly ubiquitous. The single box program does not require msg passing, and is tuned for SMP and cache effects
  - Dynamic parallelism and load balancing is easier in shared memory
  - Extra data copies can be avoided
- Disadvantages
  - Thread safe MPIs are rare
  - Code is much more complex, and required lots of work, hard to debug

OpenMP & MPI

- Very similar in nature to Threads & MPI, but uses program directives to inform compiler about possible parallelism
- Methodology:
  - Begin with a standard MPI program, add OpenMP directives to the “node code” loops, and ask the compiler to parallelism and synchronize them.
- Advantages:
  - Compiler vendors are strongly supporting OpenMP
  - In depth understanding of threads and thread models is not required
  - Compilers can help
  - Extra data copies could be removed
- Disadvantages:
  - Code is much more complex, and required lots of work, hard to debug
  - Mixing compiler directives and semi-explicit parallelism is messy
  - MPI Thread safety can be a problem

Use a Programming Framework

- Principle: The source code of the application should not become more complex to support more sophisticated run-time layers
- Let the underlying run-time support for the framework work out all the complexities of executing code for the machine.
- Examples:
  - POOMA/SMARTS (www.acl.lanl.gov/smarts)
  - OVERTURE
- Fundamental scientific programming abstractions are presented to and manipulated by user. Direct message passing is avoided.
- The framework moves data (message passing) and understands parallelism, and can use threads for SMP parallelism.

SMP Programming Summary:

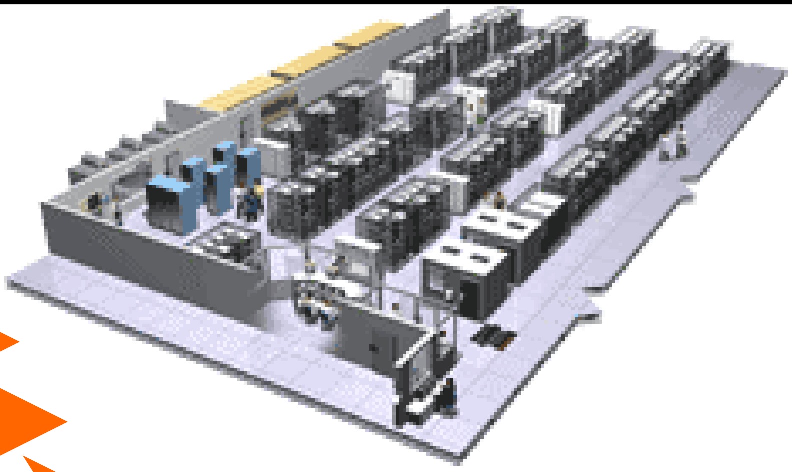
- Easy, simple, portable: MPI Everywhere
  - Unnecessary data movement and copies likely
- MPI + Threads
  - Requires more complex source code, but can deliver good SMP performance. Thread safety can be an issue
- MPI + OpenMP
  - Requires more complex source code, but can deliver good SMP performance. Special compiler required, mixing directives and explicit parallelism. Thread safety can be an issue.
- Parallel Programming Framework
  - Source code uses parallel programming abstractions, and is not made more complex for SMPs. RTS can improve execution behind the scenes. Requires a large investment in the framework and use.

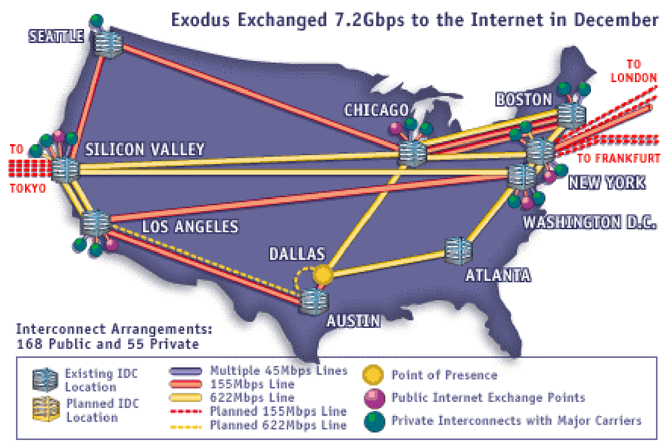
Linux and the Future of Clusters

Applications Service Providers are Making Some Desktop Software Obsolete

- Hotmail: Desktop mailers like Eudora and Outlook are nearly obsolete for thousands of users
- Mapquest: Buying street-map CDROMS is nearly obsolete
- OnMoney.com & Intuit.com: Manage finances & taxes from the web. Could desktop software become obsolete?
- When.com: Online calendar
- Traditional Business Software: payroll, travel expenses, HR, etc

Internet Trends: Centralized Internet-based Application Servers & non-PC access points

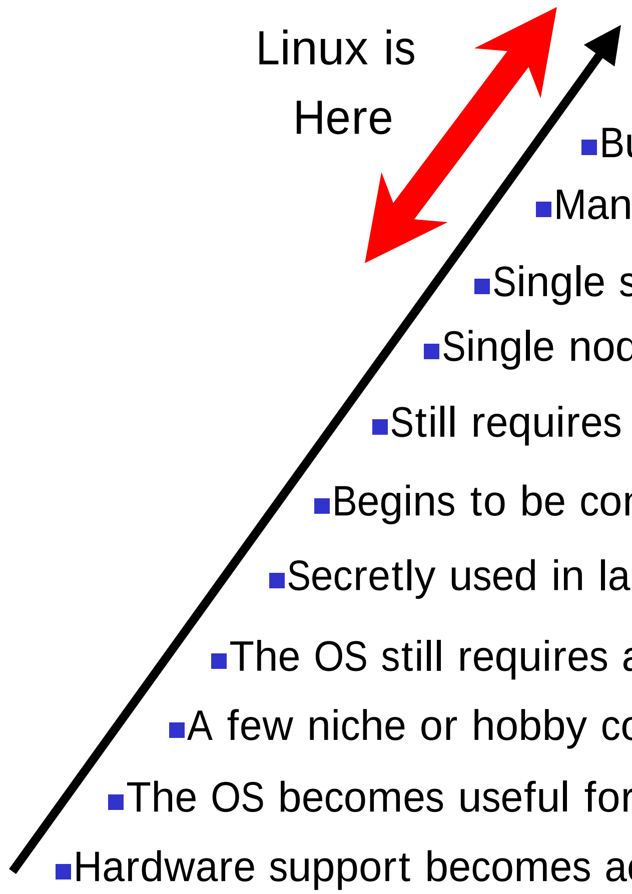




Observations of this Trend

- CPU and storage cycles are centralizing
- Scalability becomes critical
- Web protocols are not OS specific, the server can run Linux and serve a MS Windows desktop user
- Centralized servers must be fault tolerant
- Remote administration is critical
- Linux clusters are ideally suited for many of these tasks
- **The scientific computing community was the leader in large scalable clusters, but the Internet and dotcom industry has taken the lead in some areas**

Operating System Evolution



- Management costs dominate usability and scalability
- Businesses want to reduce TCO for large installations
- Many critical business applications emerge, installations proliferate
- Single systems good, but adhoc enterprise and clusters emerge
- Single node conf. and sys. mgt. is difficult and requires hacker glue
- Still requires specialized knowledge and attention
- Begins to be commercially viable solution for businesses
- Secretly used in large IT corporations by evangelists
- The OS still requires an expert, and is sensitive to configuration
- A few niche or hobby companies offer some tools and software
- The OS becomes useful for one or two real applications, i.e. email, web serving
- Hardware support becomes acceptable
- Programmers download 20 floppies, and tinker with the OS because it is fun

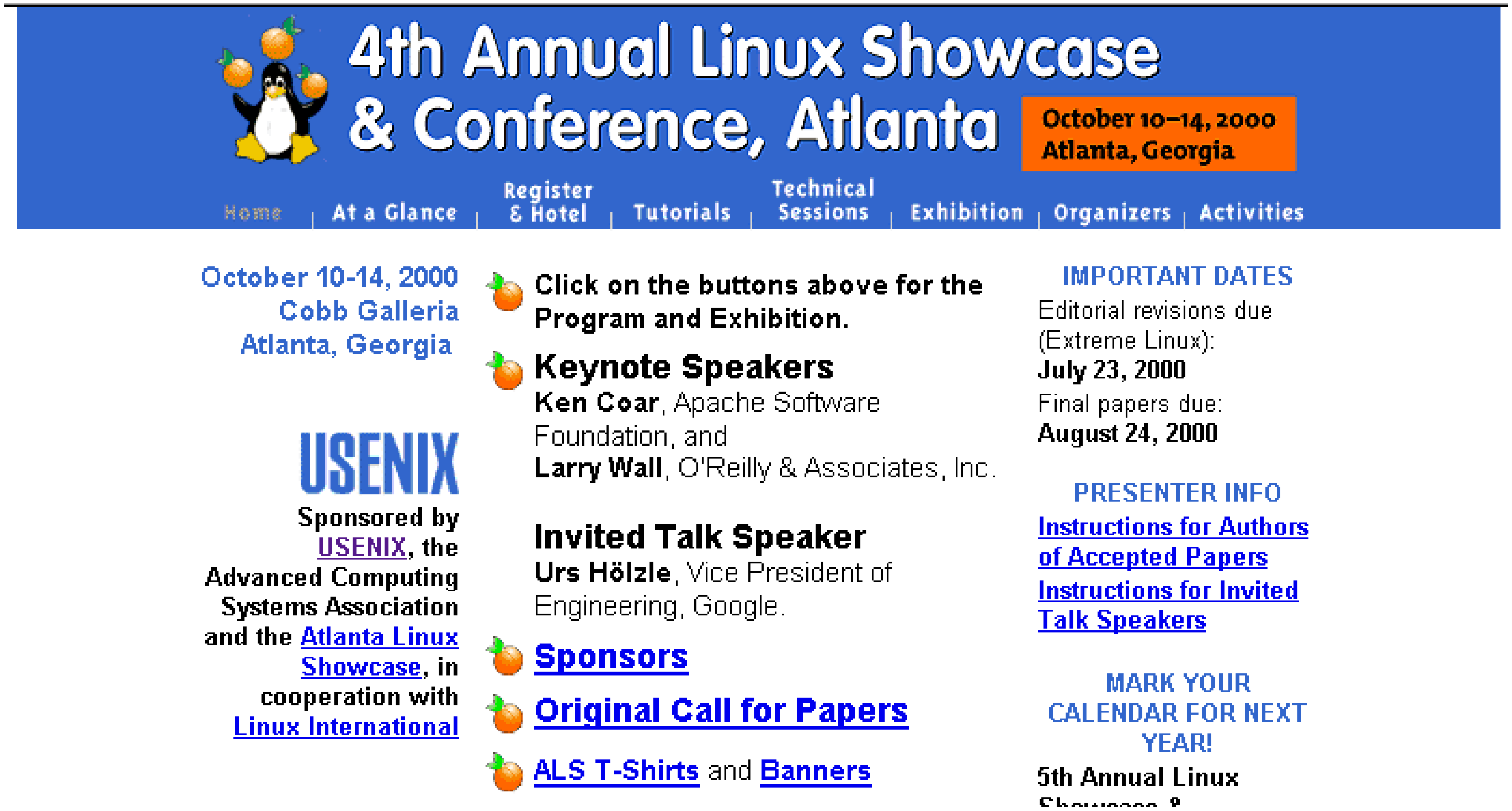
What Does The Future Hold?

- IBM, SGI, HP, Intel, Compaq, and others will continue shifting software and support to Linux
- Sun will continue to lag behind
- Internet infrastructure will continue shift towards Linux. Sites like Google & Akamai are simply the first in the trend
- Microsoft will be broken up, but continue to "Embrace and Extend" web protocols. With IE, MS will attempt to monopolize the Net
- Linux cluster technology will explode, leaving the HPC community to focus on HPC-specific software
  - MPI, PBS, parallel filesystems, etc
- Supercomputers will fall into three catagories:
  - Japanese vector machines, Linux clusters, and the Tera MTA
- Privacy concerns will create speed bump for ASPs

Extreme Linux Conference (Advertisement)

- Feb 1998: about 30 people
  - Maddog, Bob Young, Donald Becker, etc
- Linux Expo 98
- Linux Expo 99
- USENIX Tech: June 99
- ELDF: Feb 2000
- ALS: Oct 2000
  - www.usenix.org

Extreme Linux: www.linuxshowcase.org



October 10-14, 2000
Cobb Galleria
Atlanta, Georgia

Sponsored by USENIX, the Advanced Computing Systems Association and the Atlanta Linux Showcase, in cooperation with Linux International

Click on the buttons above for the Program and Exhibition.

Keynote Speakers
Ken Coar, Apache Software Foundation, and
Larry Wall, O'Reilly & Associates, Inc.

Invited Talk Speaker
Urs Hölzle, Vice President of Engineering, Google.

Sponsors

Original Call for Papers

ALS T-Shirts and Banners

IMPORTANT DATES
Editorial revisions due (Extreme Linux):
July 23, 2000
Final papers due:
August 24, 2000

PRESENTER INFO
Instructions for Authors of Accepted Papers
Instructions for Invited Talk Speakers

MARK YOUR CALENDAR FOR NEXT YEAR!
5th Annual Linux

Questions, Answers, and Discussion