Operating Systems – the Code We Love to Hate

(Operating and Runtime Systems:

Status, Challenges and Futures)

Fred Johnson

Senior Technical Manager for Computer Science
Office of Advanced Scientific Computing Research
DOE Office of Science

The Office of Science

- Supports basic research that underpins DOE missions.
- Constructs and operates large scientific facilities for the U.S. scientific community.
  - Accelerators, synchrotron light sources, neutron sources, etc.
- Six Offices
  - Basic Energy Sciences
  - Biological and Environmental Research
  - Fusion Energy Sciences
  - High Energy
  - Nuclear Physics
  - Advanced Scientific Computing Research

Simulation Capability Needs -- FY2005 Timeframe

Application	Simulation Need	Sustained Computational Capability Needed (Tflops)	Significance
Climate Science	Calculate chemical balances in atmosphere, including clouds, rivers, and vegetation.	> 50	Provides U.S. policymakers with leadership data to support policy decisions. Properly represent and predict extreme weather conditions in changing climate.
Magnetic Fusion Energy	Optimize balance between self-heating of plasma and heat leakage caused by electromagnetic turbulence.	> 50	Underpins U.S. decisions about future international fusion collaborations. Integrated simulations of burning plasma crucial for quantifying prospects for commercial fusion.
Combustion Science	Understand interactions between combustion and turbulent fluctuations in burning fluid.	> 50	Understand detonation dynamics (e.g. engine knock) in combustion systems. Solve the "soot " problem in diesel engines.
Environmental Molecular Science	Reliably predict chemical and physical properties of radioactive substances.	> 100	Develop innovative technologies to remediate contaminated soils and groundwater.
Astrophysics	Realistically simulate the explosion of a supernova for first time.	>> 100	Measure size and age of Universe and rate of expansion of Universe. Gain insight into inertial fusion processes.

CS research program

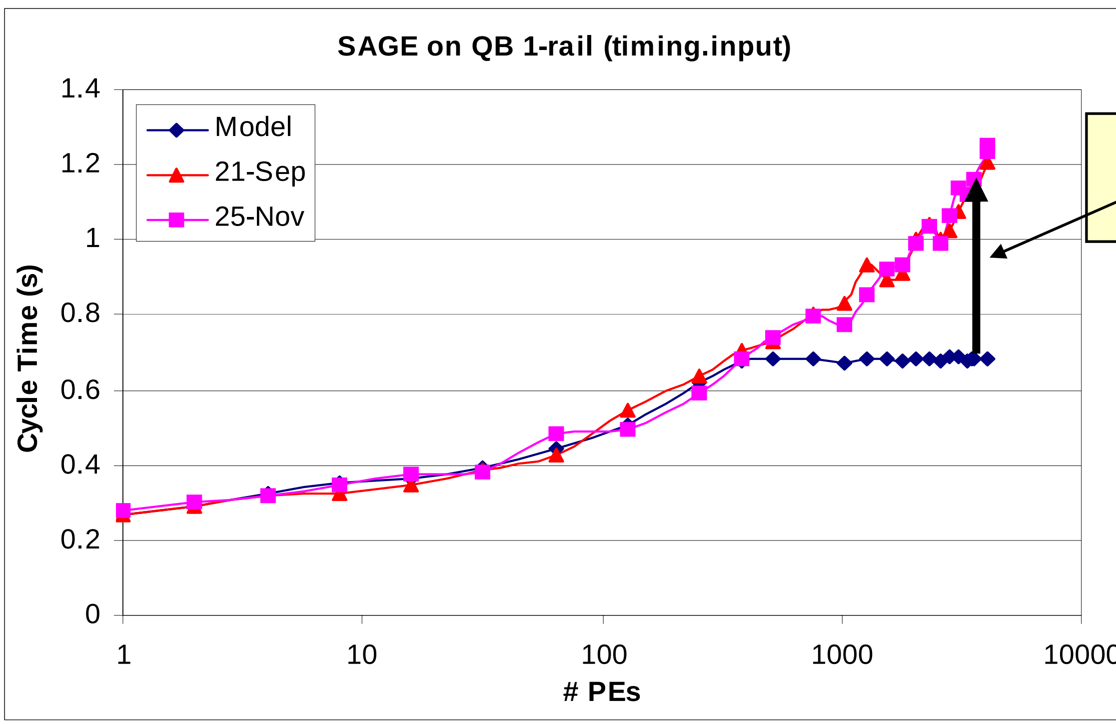
- Operating Systems
  - FAST-OS
  - Scalable System Software ISIC
  - Science Appliance
- Programming Models
  - MPI/ROMIO/PVFS II
  - Runtime libraries
    - MPI, Global Arrays (ARMCI), UPC (GASNet)
- Performance tools and analysis
- Program Development
- Data management and visualization

Outline

- Motivation – so what?
- OSes and Architectures
- Current State of Affairs
- Research directions
- Final points

OS Noise/Jitter

- Latest two sets of measurements are consistent (~70% longer than model)
- Fabrizio Petrini, Darren J. Kerbyson, Scott Pakin, "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q", in Proc. SuperComputing, Phoenix, November 2003.



There is a difference <u>why</u> ?

Lower is better!

OS/Runtime Thread Placement

- G. Jost et al --
  - *What Multilevel Parallel Programs Do When You Are Not Watching: A Performance Analysis case Study Comparing MPI/OpenMP, MLP, and Nested OpenMP*, WOMPAT 2004.
  - http://www.tlc2.uh.edu/wompat2004/Presentations/jost.pdf
- Multi-zone NAS Parallel Benchmarks
  - Coarse grain parallelism between zones
  - Fine grain loop-level parallelism in solver routines
- MLP 30% – 50% faster in some cases. Why?

MPI/MLP Differences

- MPI:
  - Initially not designed for NUMA architectures or mixing of threads and processes
  - API does not provide support for memory/thread placement
  - Vendor-specific APIs to control thread and memory placement
- MLP:
  - Designed for NUMA architectures and hybrid programming
  - API includes system call to bind threads to CPUs
  - Binds threads to CPUs (*Pinning*) to improve performance of hybrid codes

Results

- Thread binding and data placement:
  - Bad: Initial process assignment to CPUs is sequential – subsequent thread placement separates threads and data

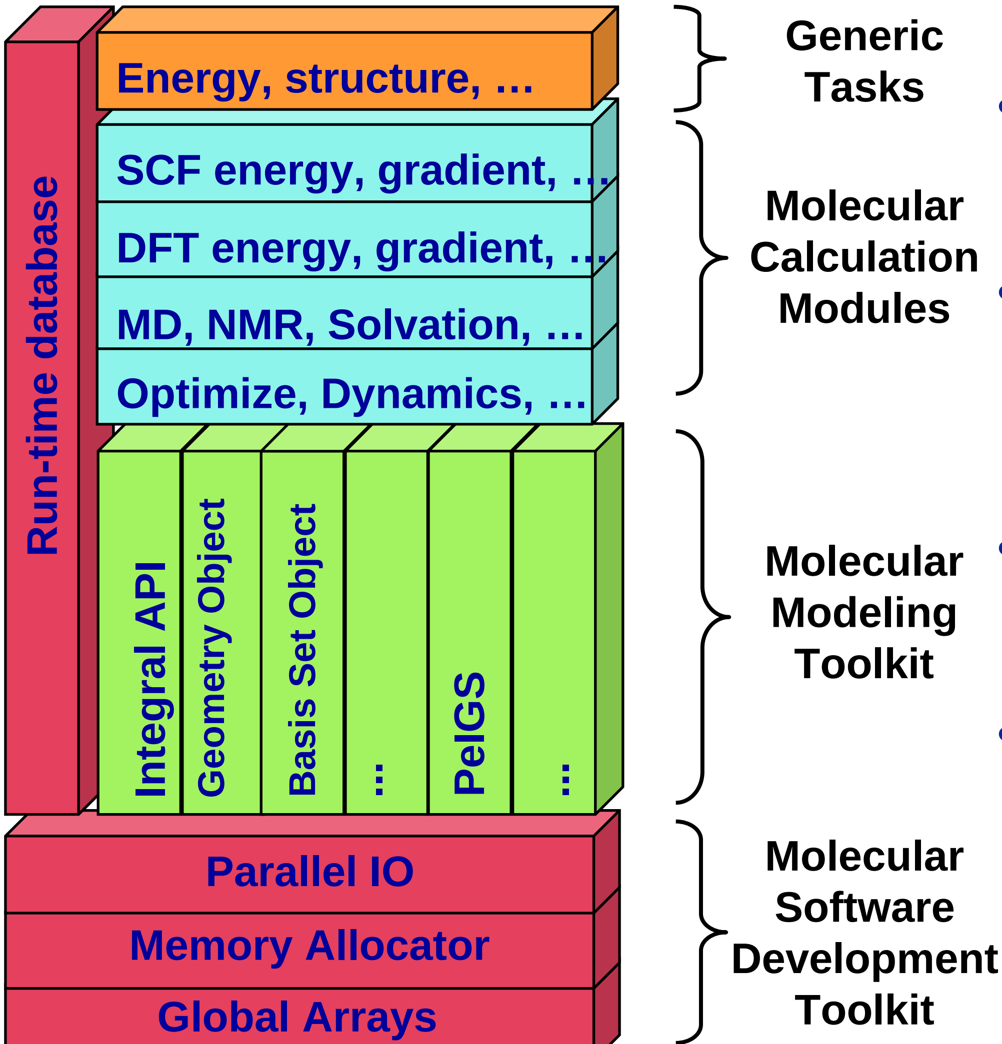
P1	P2								

  - Good: Initial process assignment to CPUs allows room for multiple threads/data of a logical process to be “close to each other”

P1						P2			

- ***“The use of detailed analysis techniques helped to determine that initially observed performance differences were not due to the programming models themselves but rather to other factors.”***

NWChem Architecture



- Object-oriented design
  - **abstraction, data hiding, handles, APIs**
- Parallel programming model
  - **non-uniform memory access, global arrays, MPI**
- Infrastructure
  - **GA, Parallel I/O, RTDB, MA, ...**
- Program modules
  - **communication only through the database; persistence for easy restart**

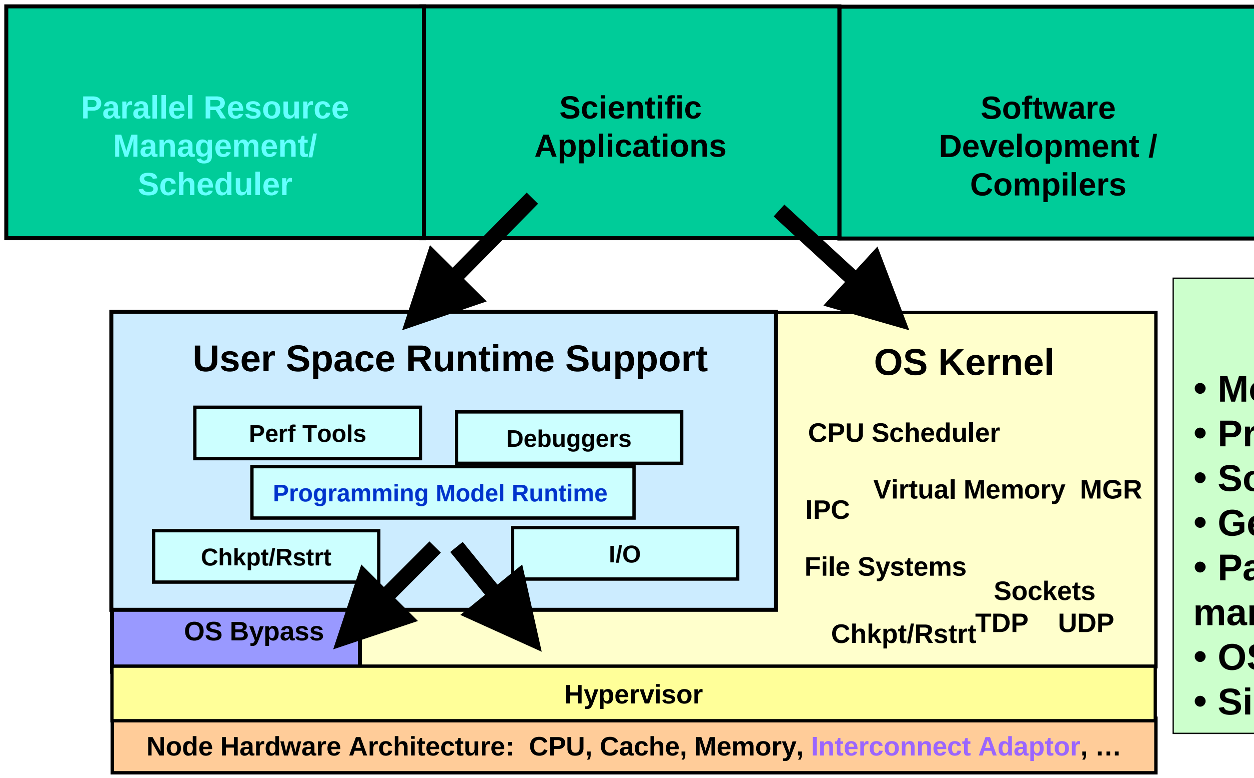
Outline

- Motivation – so what?
- OSes and Architectures
- Current State of Affairs
- Research directions
- Final points

The 100,000 foot view

- **Full-featured OS**
  - Full set of shared services
  - Complexity is a barrier to understanding new hw and sw
  - Subject to “rogue/unexpected” effects
  - Unix, Linux
- **Lightweight OS**
  - Small set of shared services
  - Puma/Cougar/Catamount – Red, Red Strom
  - CNK – BlueGene/L
- **Open Source/proprietary**
- **Interaction of system architecture and OS**
  - Clusters, CCNuma, MPP, Distributed/shared memory, bandwidth …
  - Taxonomy is still unsettled
    - High-Performance Computing: Clusters, Constellations, MPPs, and Future Directions, Dongarra, Sterling, Simon, Strohmaier, CISE, March/April 2005.

Birds eye view of a typical software stack on a compute node

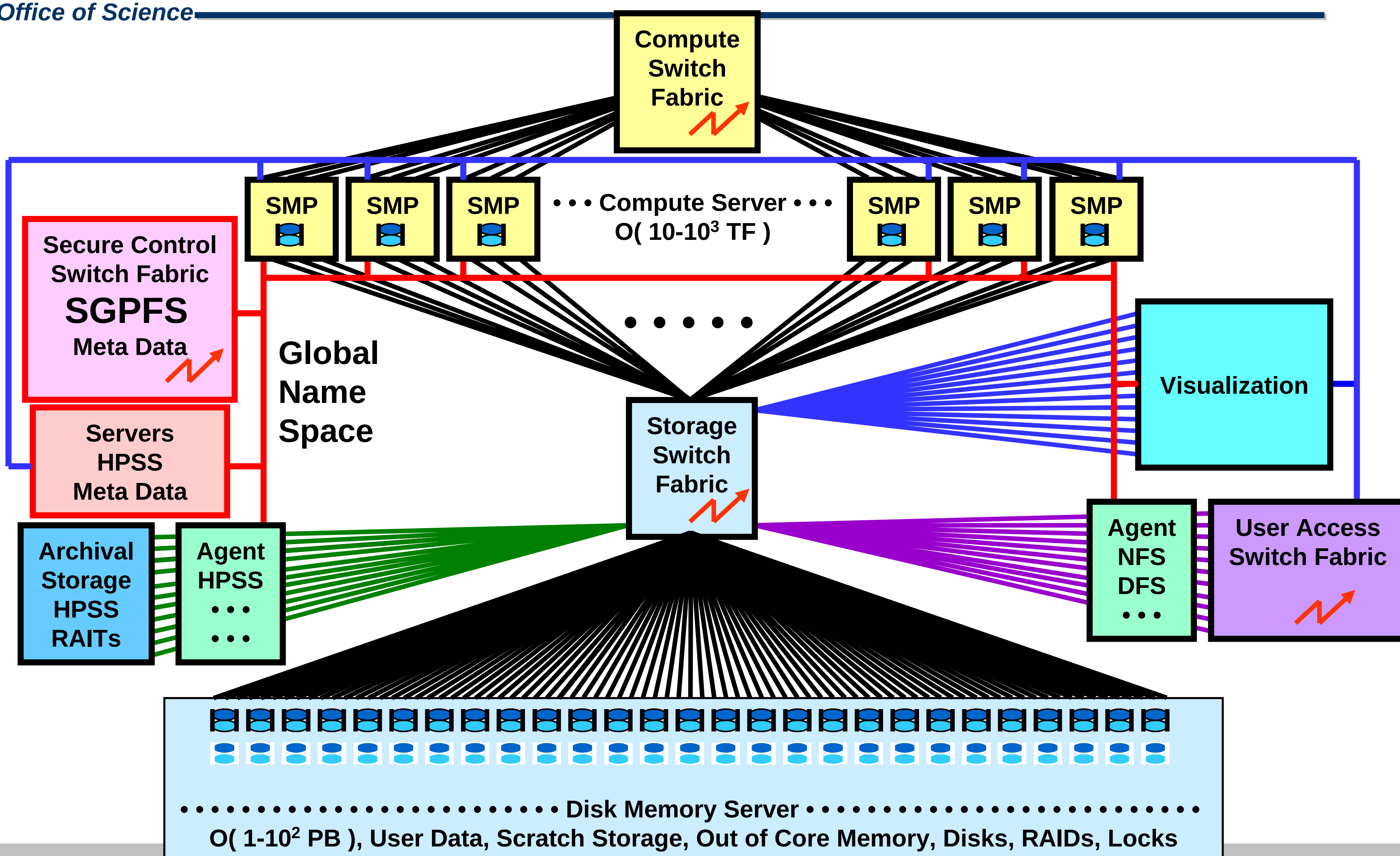


HPC System Software Elements

Kernel Characteristics

- Monolithic
- Privileged operations for protection
- Software tunable
- General purpose algorithms
- Parallel knowledge in resource management and runtime
- OS bypass for performance
- Single system image?

A High-End Cluster

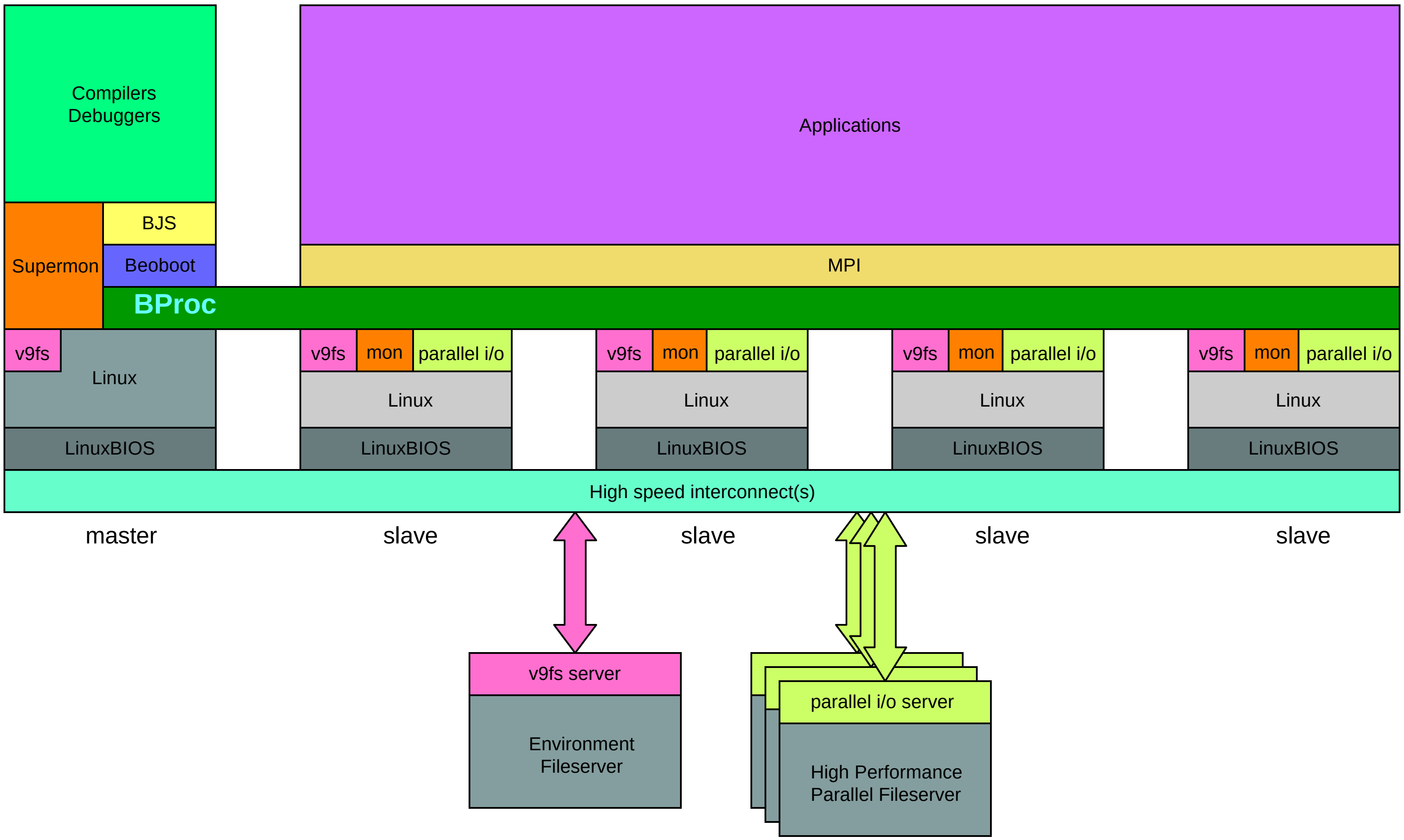


Clustering software

- **Classic**
  - Lots of local disks with full root file systems and full OS on every node
  - All the nodes are really, really independent (and visible)
- **Clustermatic**
  - Clean, reliable BIOS that boots in seconds (LinuxBIOS)
  - Boot directly into Linux so you can use HPC network to boot
  - See entire cluster process state from one node (bproc)
  - Fast, low overhead monitoring (Supermon)
  - No NFS root -- root is local ramdisk

LANL SCIENCE APPLIANCE

Scalable Cluster Computing using *Clustermatic*

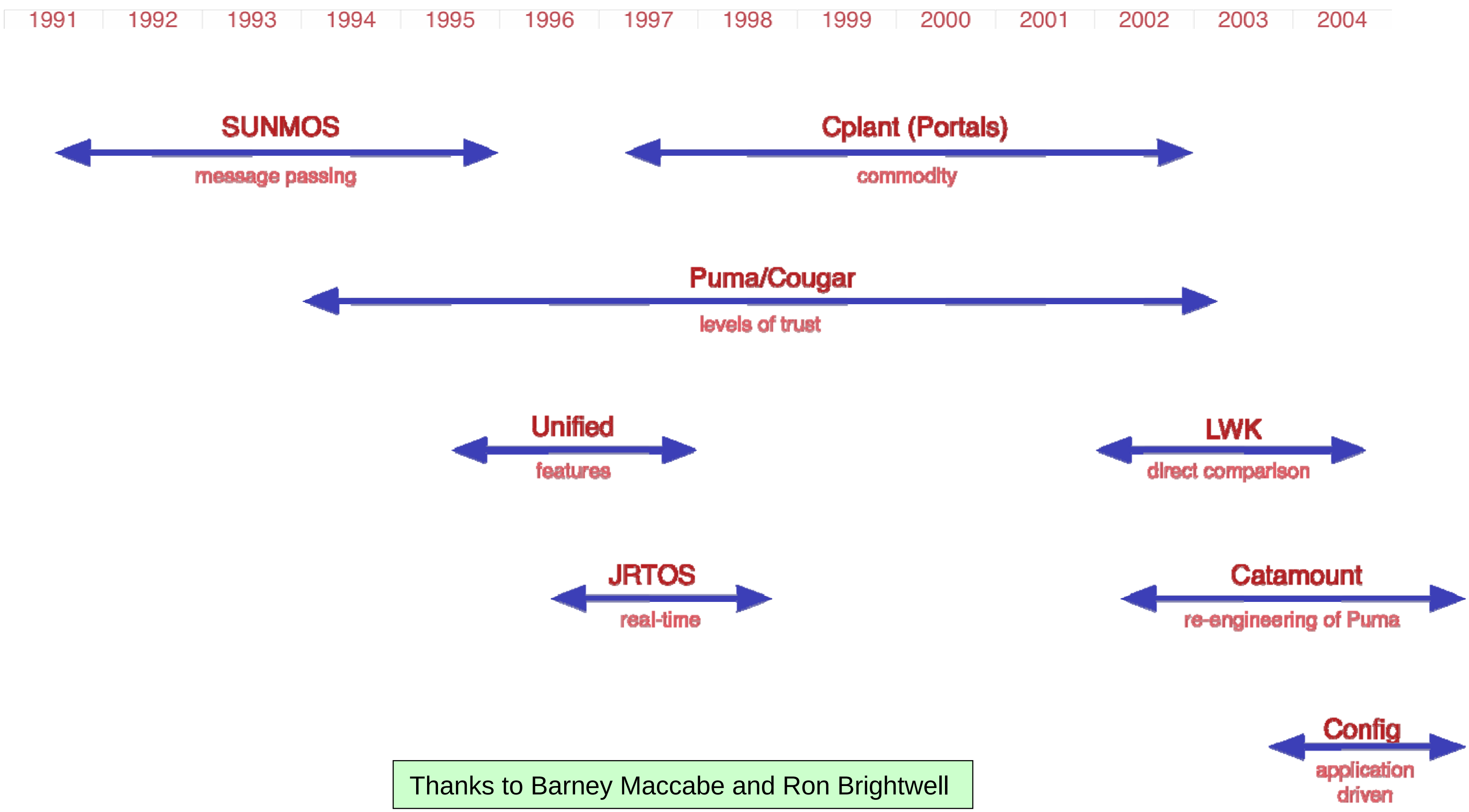


Thanks to Ron Minnich

LWK approach

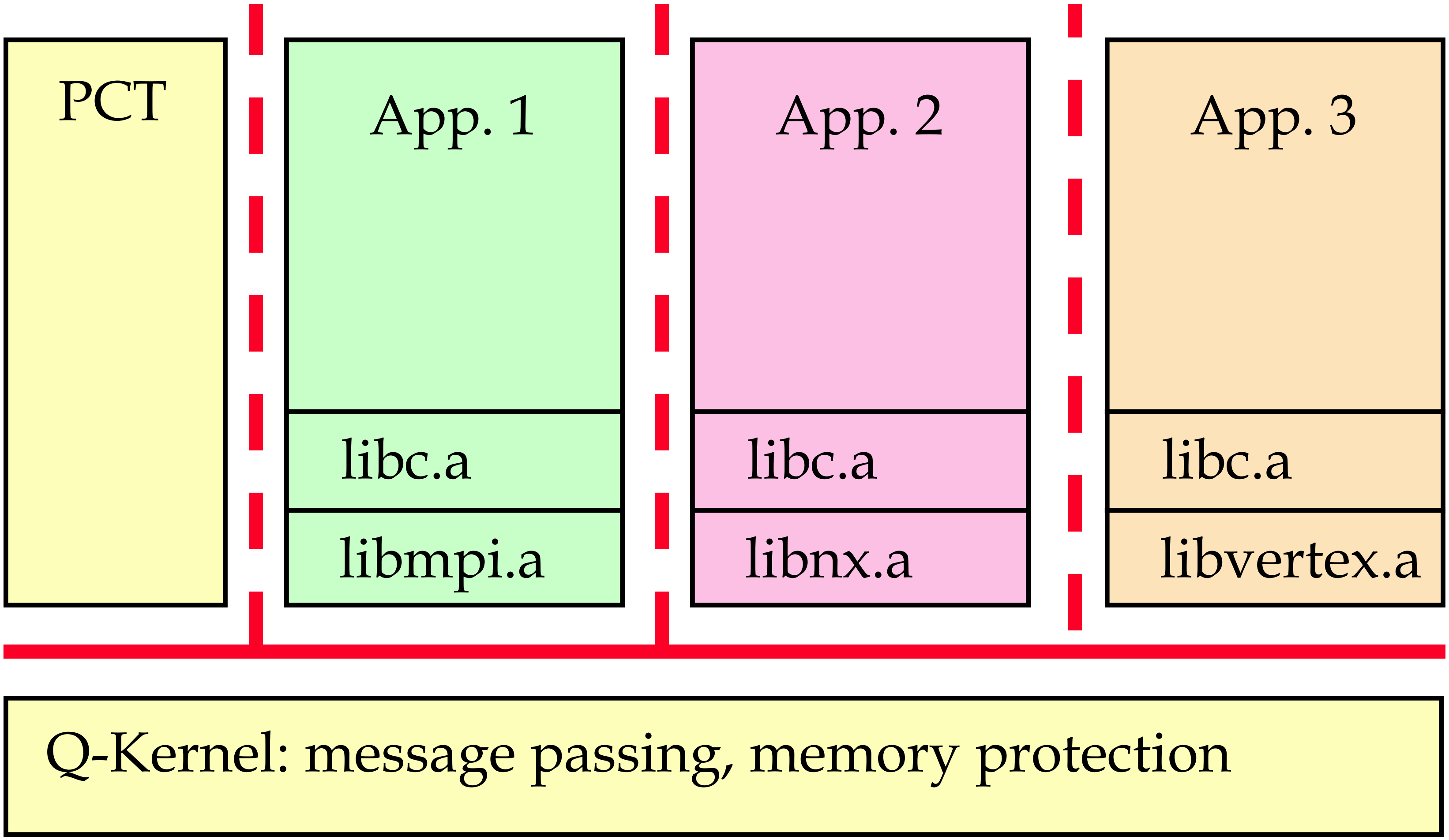
- Separate policy decision from policy enforcement
- Move resource management as close to application as possible
- Protect applications from each other
- Get out of the way

History of Sandia/UNM Lightweight Kernels



Thanks to Barney Maccabe and Ron Brightwell

LWK Structure



LWK Ingredients

- Quintessential Kernel (Qk)
  - Policy enforcer
  - Initializes hardware
  - Handles interrupts and exceptions
  - Maintain hardware virtual addressing
  - No virtual memory paging
  - Static size
  - Small size
  - Non-blocking
  - Few, well defined entry points

- Process Control Thread
  - Runs in user space
  - More privileges than user applications
  - Policy maker
  - Process loading
  - Process scheduling
  - Virtual address space management
  - Changes behavior of OS without changing the kernel

- Portals
  - Basic building blocks for any high-level message passing system
  - All structures are in user space
  - Avoids costly memory copies
  - Avoids costly context switches to user mode (up call)

Key Ideas

- Kernel is small and reliable
  - Protection
- Kernel has static size
  - No structures depend on how many processes are running
  - All message passing structures are in user space
- Resource management pushed out of the kernel to the process and the runtime system
- Services pushed out of the kernel to the PCT and the runtime system

BGL/RS LWK System Call Comparison

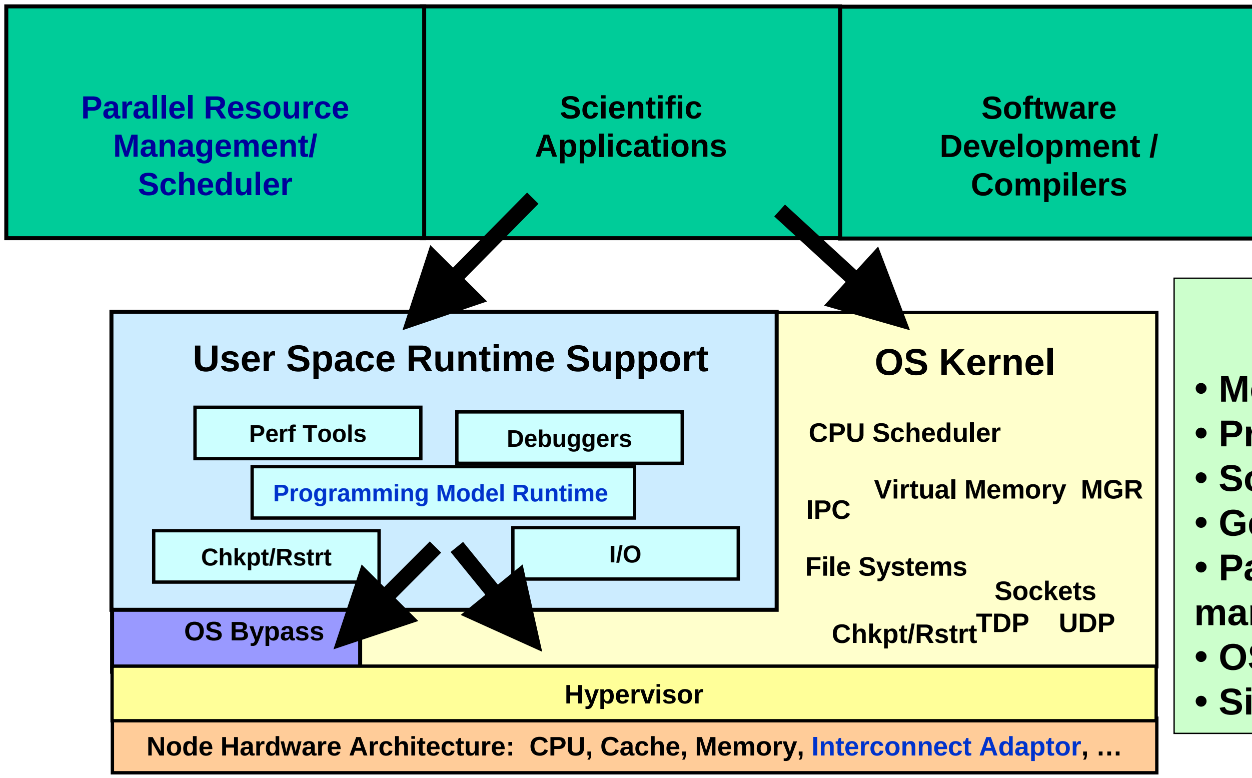
	BGL - ship	BGL - lwk	BGL - not	Total RS
RS - ship	26	4	6	36
RS - lwk	3	6	7	16
RS - not	5	0	45	50
RS - ???	11	6	81	98
Total BGL	45	16	139	200

Source: Mary Zosel, LLNL

U.S. Department of Energy

Office of Science

Sidebar: resource management software



HPC System Software Elements

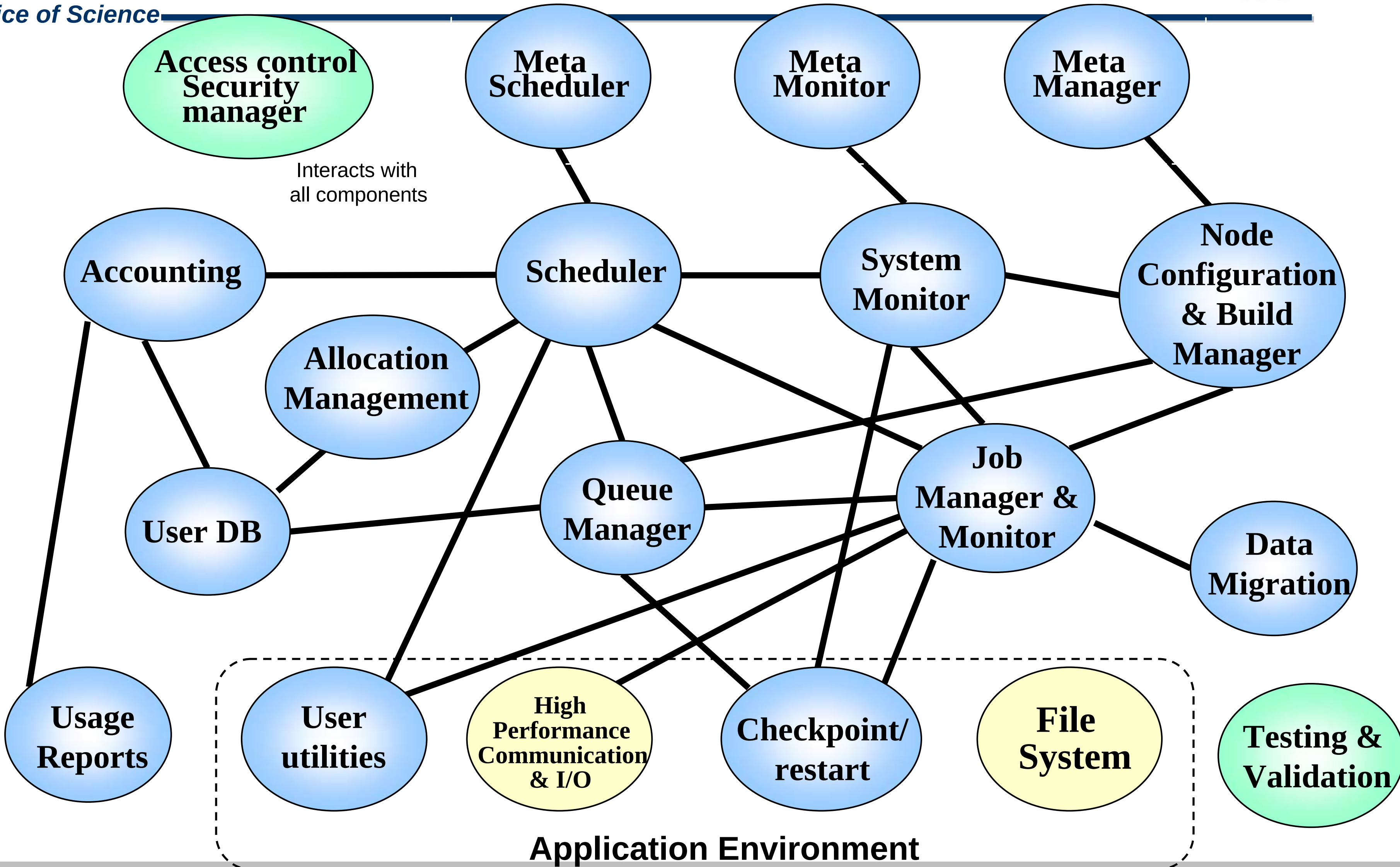
Characteristics

- Monolithic kernel
- Privileged operations for protection
- Software tunable
- General purpose algorithms
- Parallel knowledge in resource management and runtime
- OS bypass for performance
- Single system image?

Current status of system/resource management software for large machines

- Both proprietary and open-source systems
  - Machine-specific, PBS, LSF, POE, SLURM, COOAE (Collections Of Odds And Ends), …
- Many are monolithic “resource management systems,” combining multiple functions
  - Job queuing, scheduling, process management, node monitoring, job monitoring, accounting, configuration management, etc.
- A few established separate components exist
  - Maui scheduler
  - Qbank accounting system
- Many home-grown, local pieces of software
- Scalability often a weak point

System Software Architecture



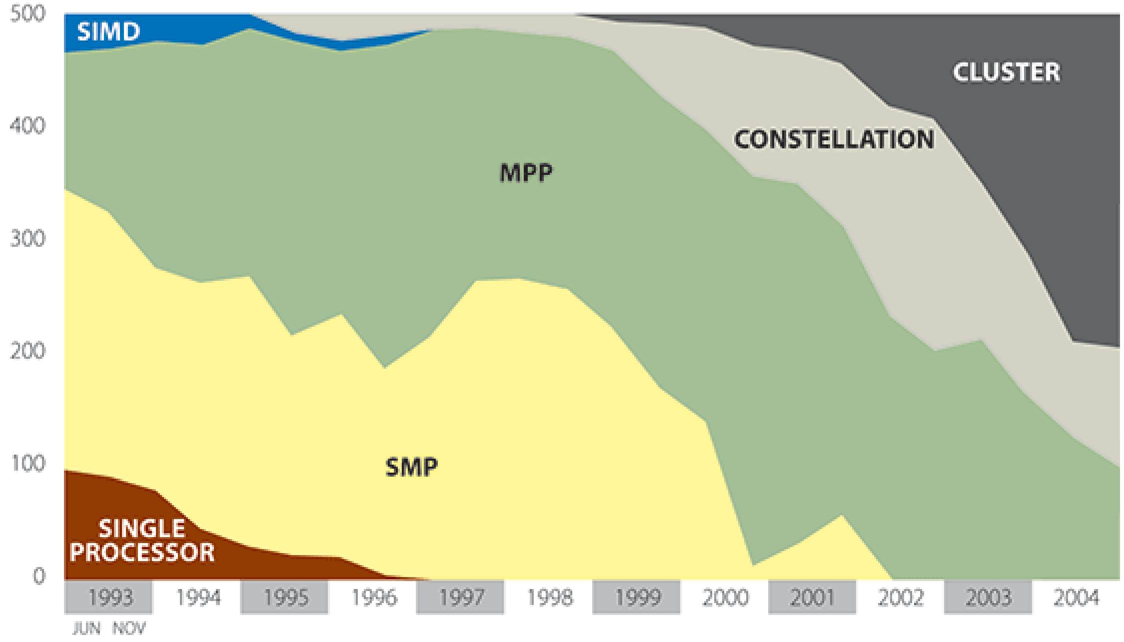
Outline

- Motivation – so what?
- OSes and Architectures
- Current State of Affairs
- Research directions
- Final points

Who's doing what – Top500 11/2004

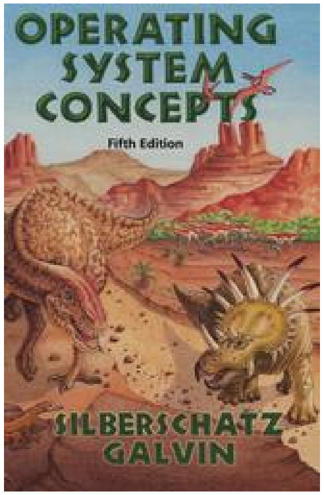
- Full Unix
  - NEC -- Earth Simulator (3)
  - IBM -- White/Seaborg (20, 21)
  - HP/Compaq -- Q (6)
- Lightweight Kernel
  - IBM -- BlueGene/L (1)
  - Intel -- ASCI Red (78)
- Linux
  - SGI -- Columbia (2)
  - HP -- PNNL (16)
  - MISC in Top 25 -- LLNL, LANL, NCSA, ARL, UCSD
- (Lots and lots of Linuxes)

Top500 Trends

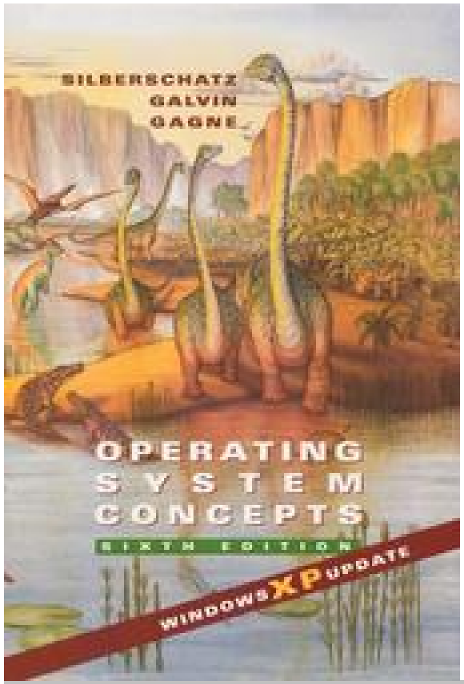


Estimated fraction of Linux in Top500: 60%

The Death of High-end OS Research



- Large effort required
  - 100’s of person years
- Unix, Linux pervasive
- Mach
  - picking a winner too early
- Services and standards
  - Users want a rich set of services
  - “To be a viable computer system, one must honor a huge list of large, and often changing standards: TCP/IP, HTTP, HTML, XML, CORBA, Unicode, POSIX, NFS, SMB, MIME, POP, IMAP, X, … A huge amount of work, but if you don’t honor the standards, you’re marginalized.”*



*** Rob Pike, Systems Software research is Irrelevant, August 2000. http://www.cs.bell-labs.com/cm/cs/who/rob/utah2000.pdf**

Death (continued)

- Hardware access
  - OS developers rarely get access to large systems (they want to break them)
  - OSF only had access to 32-node systems
- Research vs production quality
  - OS development focuses on features, not implementations
  - OS becomes more complex due to poor implementations
- Linux
  - Structure: 1,000's of lines of code know the socket structure
  - Acceptance metric: performance on servers and desktops
  - Culture: Linux hackers rarely acknowledge OS research

Sidebar: OS Statistics

- Windows 3.1 (1990) ~~ 3M lines of code
- Windows NT (1995) ~~ 4M LOC
- Windows NT 5.0 (2000) ~~ 20M LOC
- Windows XP (2002) ~~ 40M LOC
- Red Hat 7.1 (2001) ~~ 30M LOC
  - 8000 person years and cost >$1B if developed by proprietary means (COCOMO model)
- Linux kernel (2.4.2) ~~ 2.4M LOC
- 1.4M lines in kernel (~60%) are drivers

Source: More than a Gigabuck: Estimating GNU/Linux's Size, www.dwheeler.com/sloc

Outline

- Motivation – so what?
- OSes and Architectures
- Current State of Affairs
- Research directions
- Final points

OS Research Challenges

- Architecture
  - Support for architectural innovation is essential
    - Multiprocessor cores, PIM systems, smart caches
  - Current OSes can stifle architecture research
    - Linux page table abstraction is x86
- Enable multiple management strategies
  - Resource constrained applications
  - OS/application management mismatch
    - Application re-invent resource management
- Specialized HEC needs
  - New programming models
  - I/O services
  - Scalability
  - Fault tolerance

Details

- **OS structure**
  - Global/local OS split, OS/runtime split
  - Adaptability, composibility
  - Extending lightweight approaches
  - Protection boundaries and virtualization
  - Scalability – what needs to scale
- **Fault tolerance**
  - Checkpoint/restart (system, application, compiler support)
  - Run through failure, other forms of application fault tolerance
  - Migration
- **APIs/Interfaces**
  - Application/runtime, Runtime/OS, OS/compiler, architecture
  - Tool interfaces
  - Environment information
- **Hardware support for OS/runtime**
  - Protection, network reliability, collective operations, atomic memory operations, transactional memory
- **Application requirements**
- **Metrics**
- **Testbeds**

Forum to Address Scalable Technology for runtime and Operating Systems (FAST-OS)

- **Drivers:**
  - Challenges of petascale systems
  - No effective high-end OS research
    - several advanced architecture programs
    - advanced programming model activities (HPCS: Chapel, Fortress, X-10)
    - are doomed without OS/runtime innovation
  - Scaling efficiency is essential for success at the petascale
    - Must be addressed at all levels: architecture, operating system, runtime system, application and algorithms
  - Address both operating and runtime systems:
    - OS is primarily about protection of shared resources (memory, cpu, …)
    - RT is primarily about application support environment
- **Long term:**
  - Build high-end OS/Runtime community, including: vendors, academics, and labs
  - 2010 timeframe: petaflops and beyond

U.S. Department of Energy

Office of Science

FAST-OS Activities

▪ February 2002:	(Wimps) Bodega Bay
▪ July 2002:	(Fast-OS) Chicago
▪ March 2003:	(SOS) Durango
▪ June 2003:	(HECRTF) DC
▪ June 2003:	(SCaLeS) DC
▪ July 2003:	(Fast-OS) DC
▪ March 2004:	Research Announcement
▪ 2^{nd} Half 2004:	Awards

- 10 teams, about $21M invested over 3 years
- Part of HEC-URA, additional funding support from DARPA and NSA

FAST-OS Award Summary

FAST-OS Activity	Lead Org/ Coord. PI	Lab/Univ Partners	Industry Partners	Effort
Colony	LLNL Terry Jones	UIUC	IBM	Virtualization on minimal Linux with SSI services
Config	SNL Ron Brightwell	UNM, Caltech		Combine micro services to build app specific OS
DAiSES	UTEP Pat Teller	Wisconsin	IBM	Adaptation of OS based on Kperfmon & Kerninst
K42	LBNL Paul Hargrove	Toronto, UNM	IBM	Enhance applicability of K42 for HEC OS research
MOLAR	ORNL Stephen Scott	LaTech, OSU, NCSU, UNM	Cray	Modules to config and adapt Linux + RAS & fSM
SSI	ORNL Scott Studham	Rice	HP, CFS, SGI, Intel	OpenSSI, Intersection of big (SMP) and small (node) kernels
Right-Weight	LANL Ron Minnich		Bell Labs	Build application specific Linux/Plan 9 kernels
Scalable FT	PNNL Jarek Nieplocha	LANL, UIUC	Quadrics, Intel	Implicit, explicit, incremental checkpointing & resilience
SmartApps	Texas A&M Lawrence Rauchwerger	LLNL	IBM	Vertical integration between SmartApps and K42
ZeptoOS	ANL Pete Beckman	Oregon		Ultralight Linux, collective runtime, measure & FT

U.S. Department of Energy

Office of Science

Areas of Emphasis

- **Virtualization**
  - lightweight mechanisms for virtual resources
  - better balance for large set of small entities
- **Adaptability**
  - apps go through different phases
  - configurability versus adaptability
- **Usage model & system management**
  - dedicated, space shared, time shared
  - single system may have multiple OSes
- **Metrics & Measurement**
  - adaptation requires measurement
  - what and how to measure
  - HPC Challenge
- **OS Noise**
  - controlling asynchronous activities undertaken by the OS/runtime

- **Fault handling**
  - Checkpoint/restart -- implicit (system); explicit (application)
  - Run through failure
  - Prediction and migration
- **Common API**
  - Defining/supporting/using
- **Single System Image**
  - managing complexity at scale
  - system view, application view
- **Collective Runtime**
  - defining collective operations to support runtime
- **I/O**
  - compute node -- I/O system interface
  - I/O offload

U.S. Department of Energy

Office of Science

	Virtualization	Adaptability	Usage Models	Metrics	Fault Handling	Common API	SSI	Collective RT	I/O	OS Noise
Colony	H		M		H	M	H		M	H
Config	H	M	H					M	M	M
DAiSES		H		H		M				
K42		H		H		H	M			M
MOLAR		H	H	H	H			M		M
Peta-Scale SSI			H		H		H		H	H
Rightweight		M		H			M		M	H
Scalable FT					H			M	H	M
SmartApps	M	H		H		M				
ZeptoOS			H	H	H			H		H

H	High
M	Medium

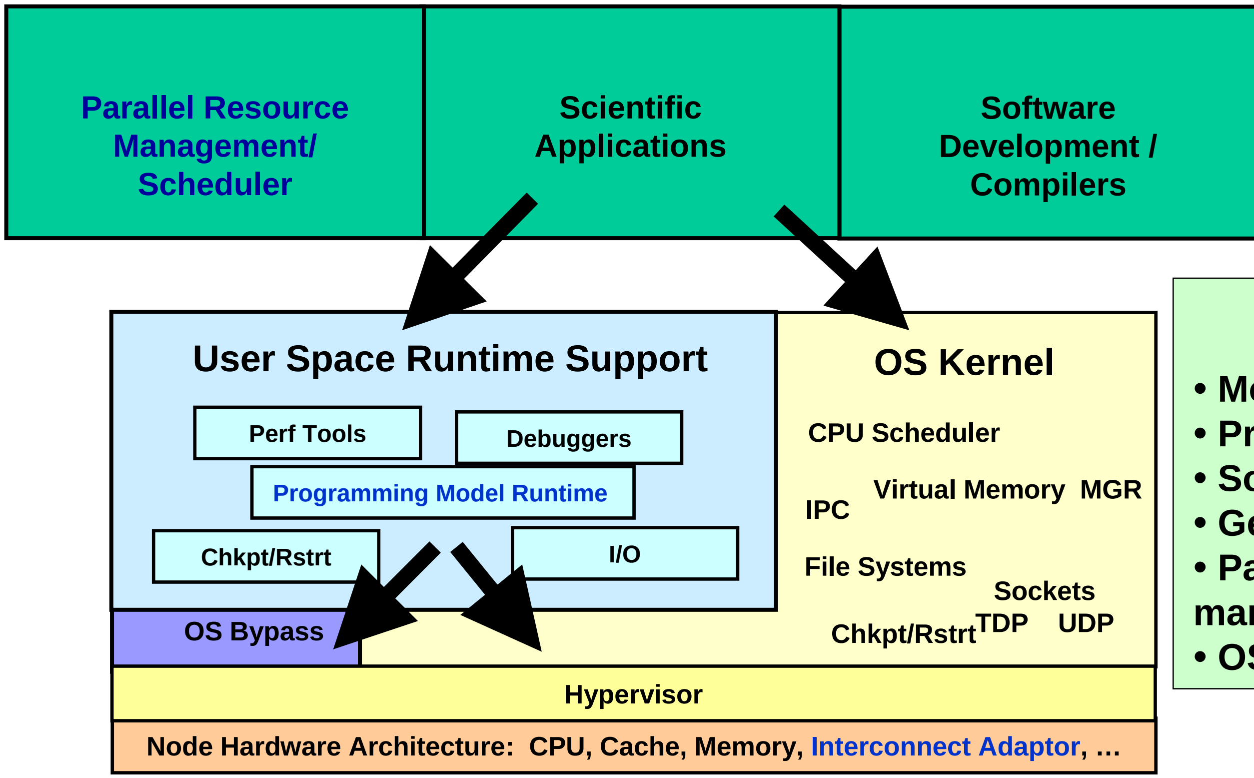
FAST-OS Statistics

- OSes (4):
  - Linux (6.5), K-42 (2), Custom (1), Plan 9 (.5)
- Labs (7):
  - ANL, LANL, ORNL, LBNL, LLNL, PNNL, SNL
- Universities (13):
  - Caltech, Louisiana Tech, NCSU, Rice, Ohio State, Texas A&M, Toronto, UIUC, UTEP, UNM, U of Chicago, U of Oregon, U of Wisconsin
- Industry (8):
  - Bell Labs, Cray, HP, IBM, Intel, CFS (Lustre), Quadrics, SGI

Outline

- Final points
  - Virtualization/hypervisor
    - Status, future
  - Research frameworks
    - K-42, Plan9
  - I/O, file systems
  - Engaging the open source community

Birds Eye View of the Software Stack



HPC System Software Elements

Characteristics

- Monolithic kernel
- Privileged operations for protection
- Software tunable
- General purpose algorithms
- Parallel knowledge in resource management and runtime
- OS bypass for performance

U.S. Department of Energy

Office of Science

Virtualization – hypervisers/virtual machines

- Full virtualization: VMware, VirtualPC
  - Run multiple unmodified guest OSes
  - Hard to efficiently virtualize x86
- Para-virtualization: UML, Xen
  - Run multiple guest OSes ported to special arch
  - Arch Xen/x86 is very close to normal x86

Xen is being used by the FastOS PetaSSI project to simulate SSI on larger clusters

Single System Image with process migration

OpenSSI	OpenSSI	OpenSSI	OpenSSI
XenLinux Linux 2.6.10 Lustre	XenLinux Linux 2.6.10 Lustre	XenLinux Linux 2.6.10 Lustre	XenLinux Linux 2.6.10 Lustre

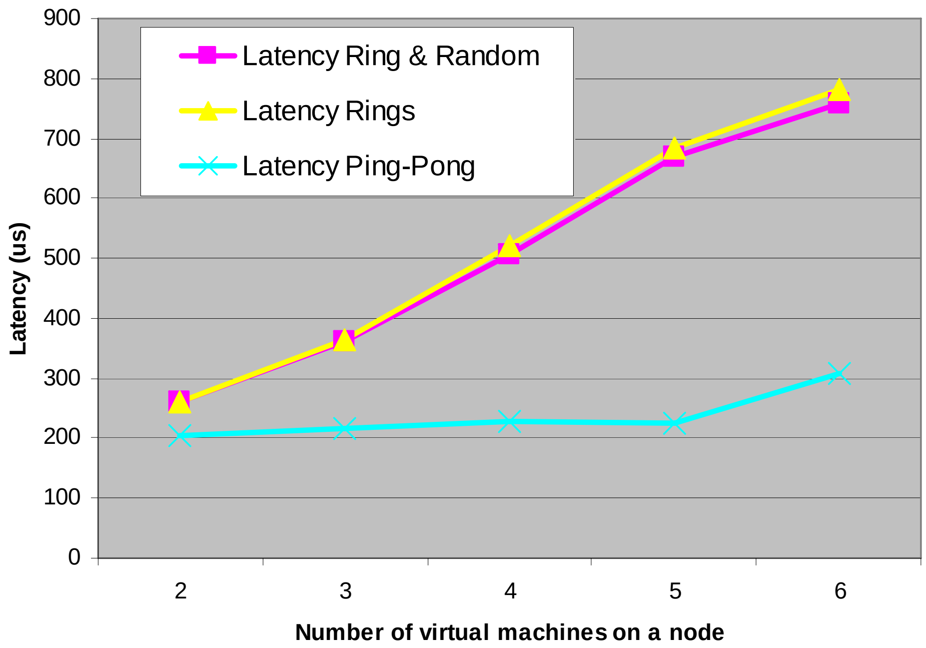
Xen Virtual Machine Monitor

Hardware (SMP, MMU, physical memory, Ethernet, SCSI/IDE)

Thanks to Scott Studham

Virtualization and HPC

- For latency tolerant applications it is possible to run multiple virtual machines on a single physical node to simulate large node counts.
- Xen has little overhead when GuestOS's are not contending for resources. This may provide a path to support multiple OS's on a single HPC system.



Overhead to build a Linux kernel on a GuestOS:

Xen:	3%
VMWare:	27%
User Mode Linux:	103%

Hypervisor Status HW support – IBM Power, Intel, AMD
SW support – Xen support in Linux 2.6 kernel

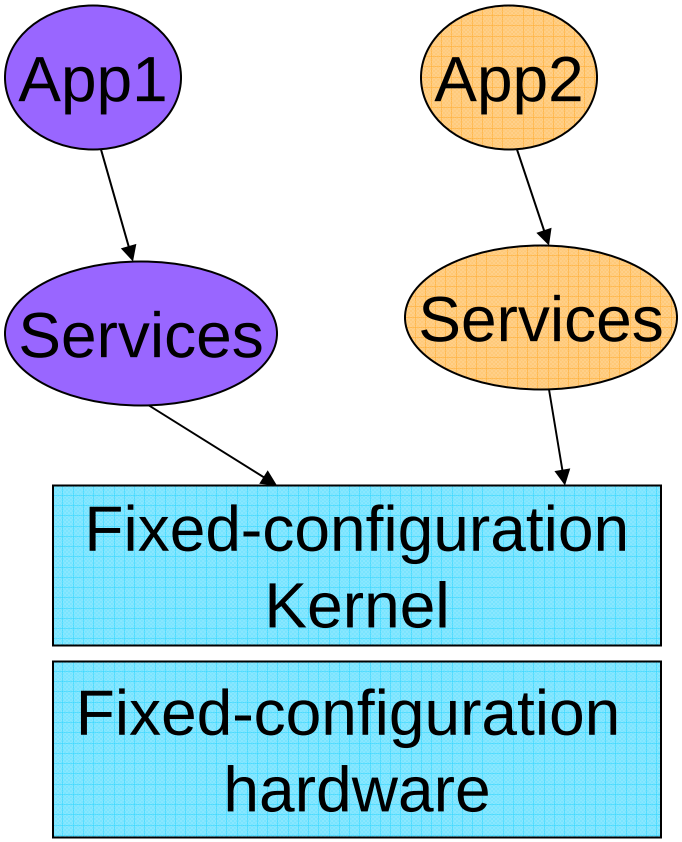
K-42

- Framework for OS research
- Full Linux compatibility – API and ABI
- Most OS functionality in user-level library
- Object-oriented design at all levels
  - Policies/implementations of every physical/virtual resource instance can be customizable to individual application needs
  - Enables dynamic adaptation to changing application needs (hot swapping)

K-42 (cont)

- Research activity at Watson Lab led by Orran Krieger
- Research OS for PERCS, the IBM HPCS project
- K(itchawan)-42

Plan 9 from Bell Labs (or Outer Space)



- Not like anything you've seen
  - Not a mini-, micro-, nano-kernel
- Core OS is fixed-configuration set of "devices"
  - Means "anything that *has* to be is in the OS"
  - E.g. Memory,TCP/IP stack, Net hardware, etc.
- Everything else is a "Server"
  - File systems, windowing systems, etc.

Thanks to Ron Minnich

Features

- What has to be in the OS goes in the OS
- Everything else is optional
  - If you need something you pay for it
  - If not, not
- Options are configured per-process-group
  - The name for this is “private name spaces”
  - There is no root user
  - There are no integer UIDs
  - There need not be a central “UID store”
- 38 system calls
  - Linux is at 300 and counting

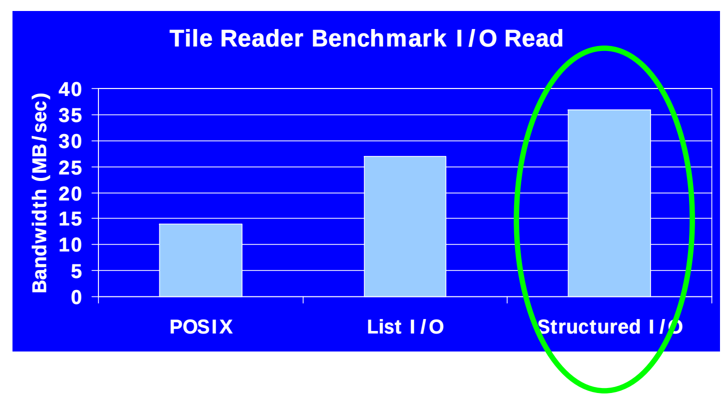
I/O software stacks

- I/O components layered to provide needed functionality
- Layers insulate apps from eccentric low-level details
  - HLLs provide interfaces and models appropriate for domains
  - Middleware provides a common interface and model for accessing underlying storage
  - PFS manages the hardware and provides raw bandwidth
- Maintain (most of) I/O performance
  - Some high-level library (HLL) features do cost performance
  - Opportunities for optimizations at all levels
- Parallel file system challenges:
  - Scaling effective I/O rate
  - Scaling metadata/management operation rate
  - Providing fault tolerance in large scale systems

Thanks to Rob Ross

Interfaces again

- POSIX I/O APIs aren't descriptive enough
  - Don't enable the description of noncontiguous regions in both memory and file
- POSIX consistency semantics are too great a burden
  - Require too much additional communication and synchronization, not really required by many HPC applications
  - Will never reach peak I/O with POSIX at scale, only penalize the stubborn apps
- Alternative: use more relaxed semantics at the FS layer as the default, build on top of that



Breaking down cultural barriers

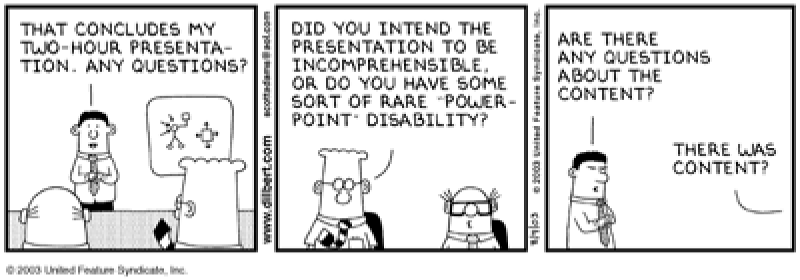
- ASC PathForward project goals
  - Accelerate IB support for HPC needs
    - Scalability, bw, latency, robustness, …
  - Effective community engagement
- Open IB Alliance
- Achieved major milestone with OpenIB driver and stack being accepted into 2.6 Linux kernel at kernel.org

Acknowledgements

- Thanks to all of the participants in FAST-OS and to the many others who have contributed to research in high-end operating systems, programming models and system software (and to this talk!)

http://www.cs.unm.edu/~fastos

FAST-OS PI Meeting/Workshop June 8-10, Washington, DC


THAT CONCLUDES MY TWO-HOUR PRESENTATION. ANY QUESTIONS?
DID YOU INTEND THE PRESENTATION TO BE INCOMPREHENSIBLE, OR DO YOU HAVE SOME SORT OF RARE "POWER-POINT" DISABILITY?
ARE THERE ANY QUESTIONS ABOUT THE CONTENT?
THERE WAS CONTENT?
www.dilbert.com
© 2003 United Feature Syndicate, Inc.