



System Software R&D at Sandia: To Red Storm and Beyond

Ron Brightwell

Scalable Computing Systems Department

Center for Computation, Computers, Information and Math

Sandia National Laboratories

rbrigh@sandia.gov

<http://www.sandia.gov/~rbrigh>

NCAR CISL Seminar

February 23, 2007



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.





Outline

- **Sandia's massively parallel systems**
- **Evolution of Sandia's system software and networking environment**
- **Top three problems Sandia is facing**
- **OS research and development activities**

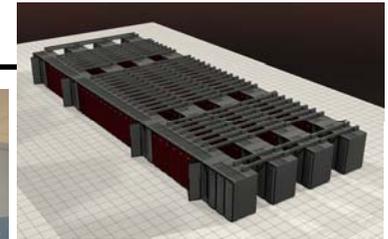


Sandia Systems



2004

1999



Red Storm

- 41 Tflops
- Custom interconnect
- Purpose built RAS
- Highly balanced and scalable
- Catamount lightweight kernel

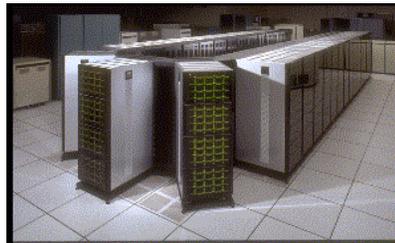
1997



Cplant

- Commodity-based supercomputer
- Hundreds of users
- Enhanced simulation capacity
- Linux-based OS licensed for commercialization

1993



Paragon

- Tens of users
- First periods processing MPP
- World record performance
- Routine 3D simulations
- SUNMOS lightweight kernel

1990



nCUBE2

- Sandia's first large MPP
- Achieved Gflops performance on applications



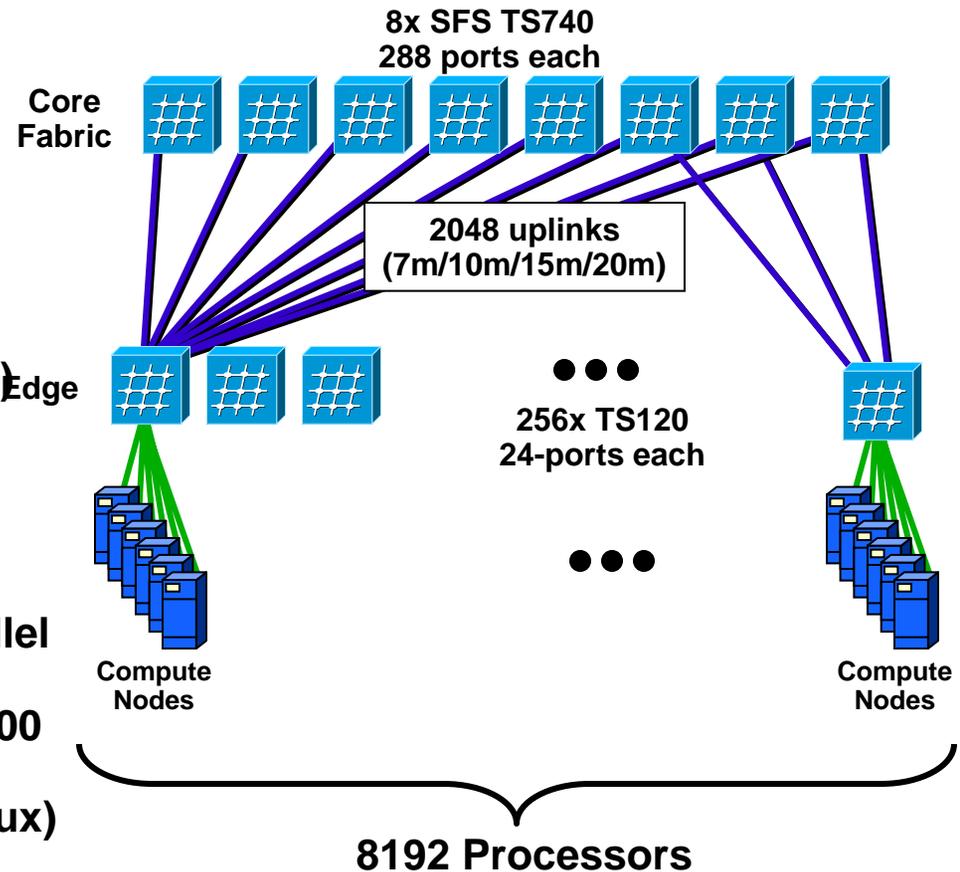
ASCI Red

- Production MPP
- Hundreds of users
- Red & Black partitions
- Improved interconnect
- High-fidelity coupled 3-D physics
- Puma/Cougar lightweight kernel



2005 - Thunderbird

- 60 TF peak
- Compute nodes
 - 4512 Dell Servers
 - Dual 3.6 GHz EM64T
 - 6 GB RAM
- Network
 - InfiniBand (Cisco (Topspin)) Edge
 - 50% Blocking Ratio
 - 8 TS-740s
 - 256 TS-120s
- Node count
 - Largest PC cluster for parallel in the world
 - #6 on November 2005 Top500 list
 - Open Fabrics software (Linux)

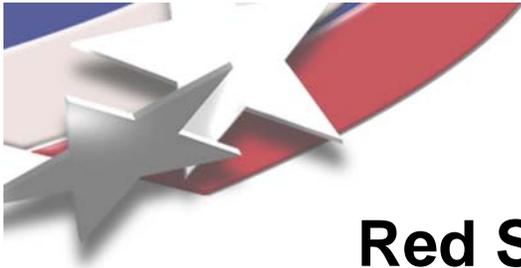




2007 – Upgraded Red Storm

- **12,980 2.4 GHz dual-core AMD Opteron CPUs**
 - 124.6 TF/s peak
- **SeaStar 2.1 network**
 - 2.1 GB/s one-way bandwidth
- **2 GB DDR 333 Memory**
- **#2 on November 2006 Top 500 list**
- **Catamount LWK with virtual node mode support**



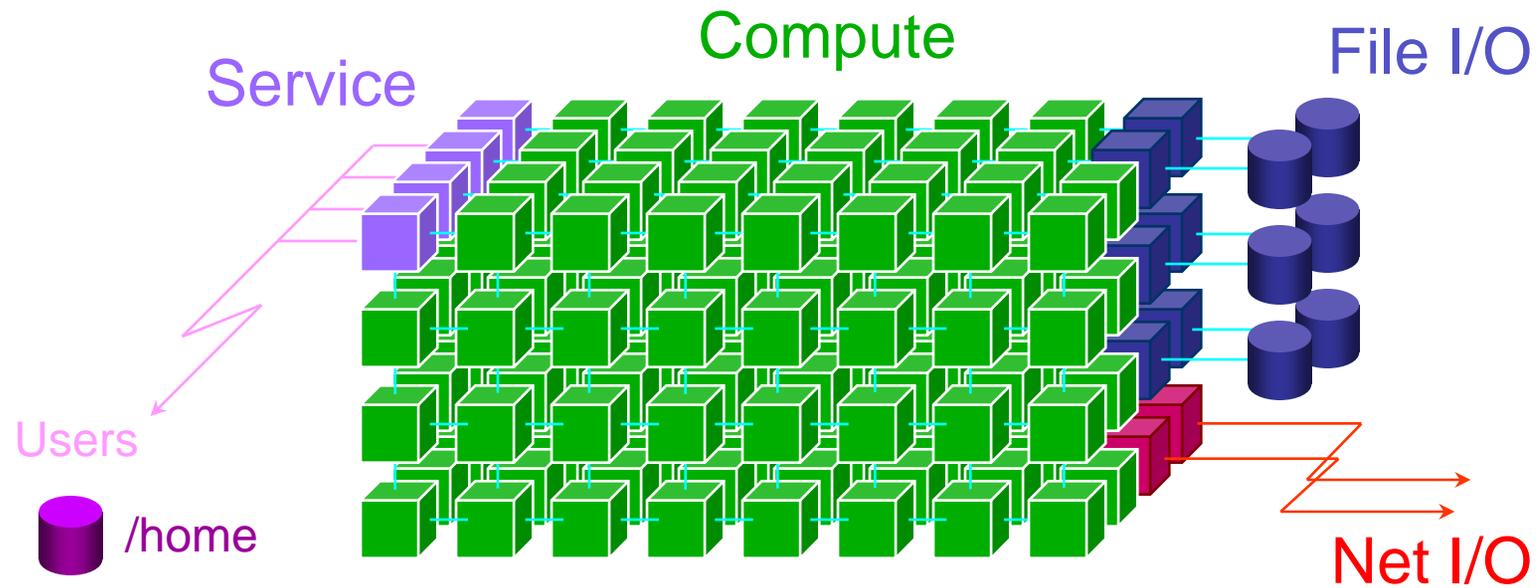


Red Storm is a Highly Balanced System

Machine	Peak Node (GFLOPS)	Peak BW (GB/s)	Ratio
IBM BG/L	5.6	0.35	0.0625
Cray Red Storm'07	9.6	4.8	0.5000
IBM Purple	48	8	0.1700
SGI Columbia	24	6.4	0.2700
Dell Thunderbird	14.4	2	0.1300
Cray Red Storm'04	4	4.8	1.2000
NEC Earth Simulator	64	12.3	0.1920
Mare Nostrum	17.6	0.5	0.0280
Thunder	22.4	1	0.0400



Conceptual Partition Model





Sandia System Software

- **Lightweight kernel (LWK) compute node operating and runtime system**
 - **Several instances of custom LWKs for several machines**
- **Portals high-performance network stack**
 - **Several generations and implementations for various custom and commodity networks**



LWK Design Goals

- Targeted at massively parallel environments comprised of thousands of processors with distributed memory and a tightly coupled network.
- Provide necessary support for scalable, performance-oriented scientific applications
- Offer a suitable development environment for parallel applications and libraries.
- Emphasize efficiency over functionality.
- Maximize the amount of resources (e.g. CPU, memory, and network bandwidth) allocated to the application.
- Seek to minimize time to completion for the application.
- Deterministic performance

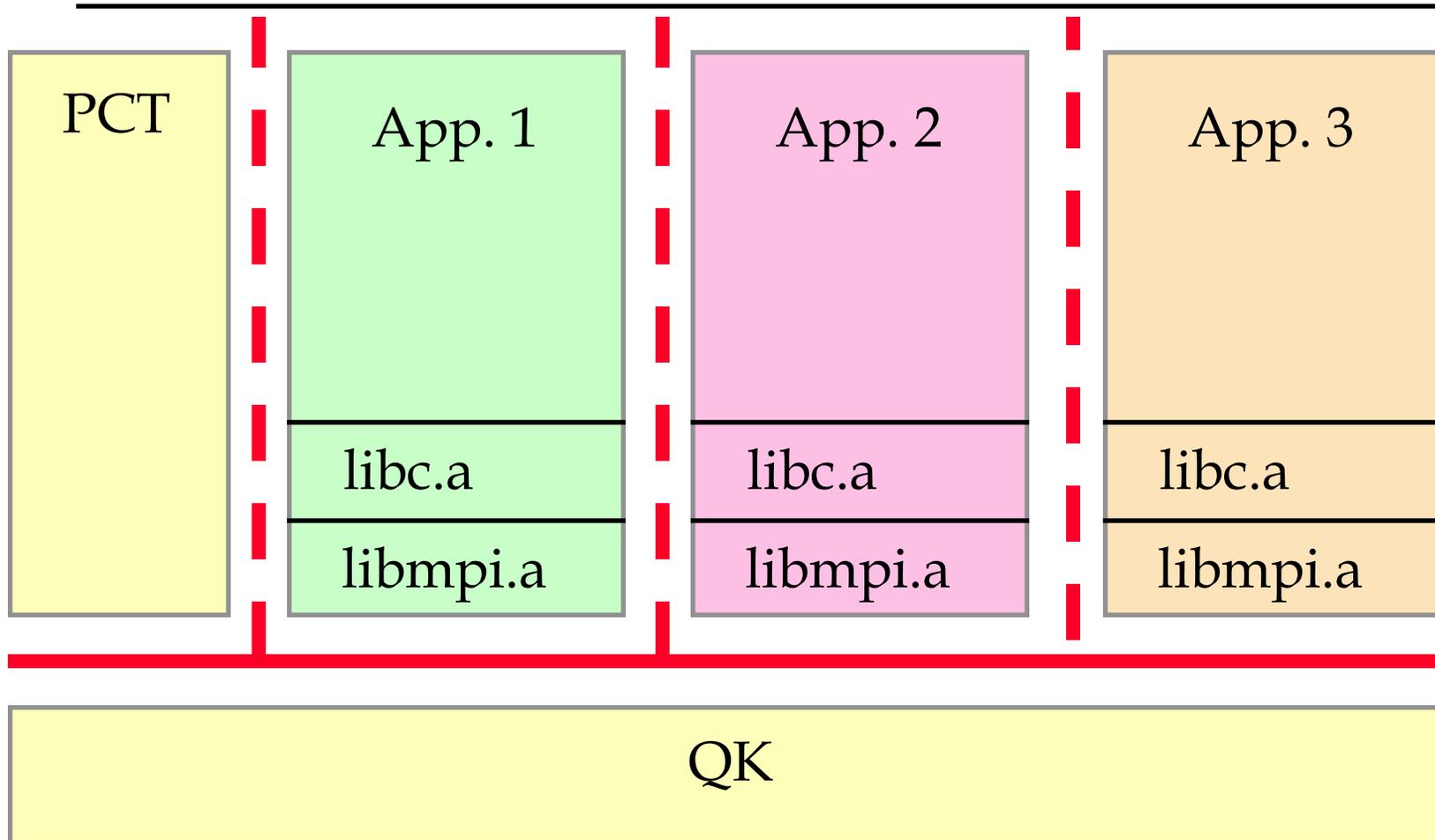


LWK Approach

- **Separate policy decision from policy enforcement**
- **Move resource management as close to application as possible**
- **Protect applications from each other**
- **Let user processes manage resources**
- **Get out of the way**

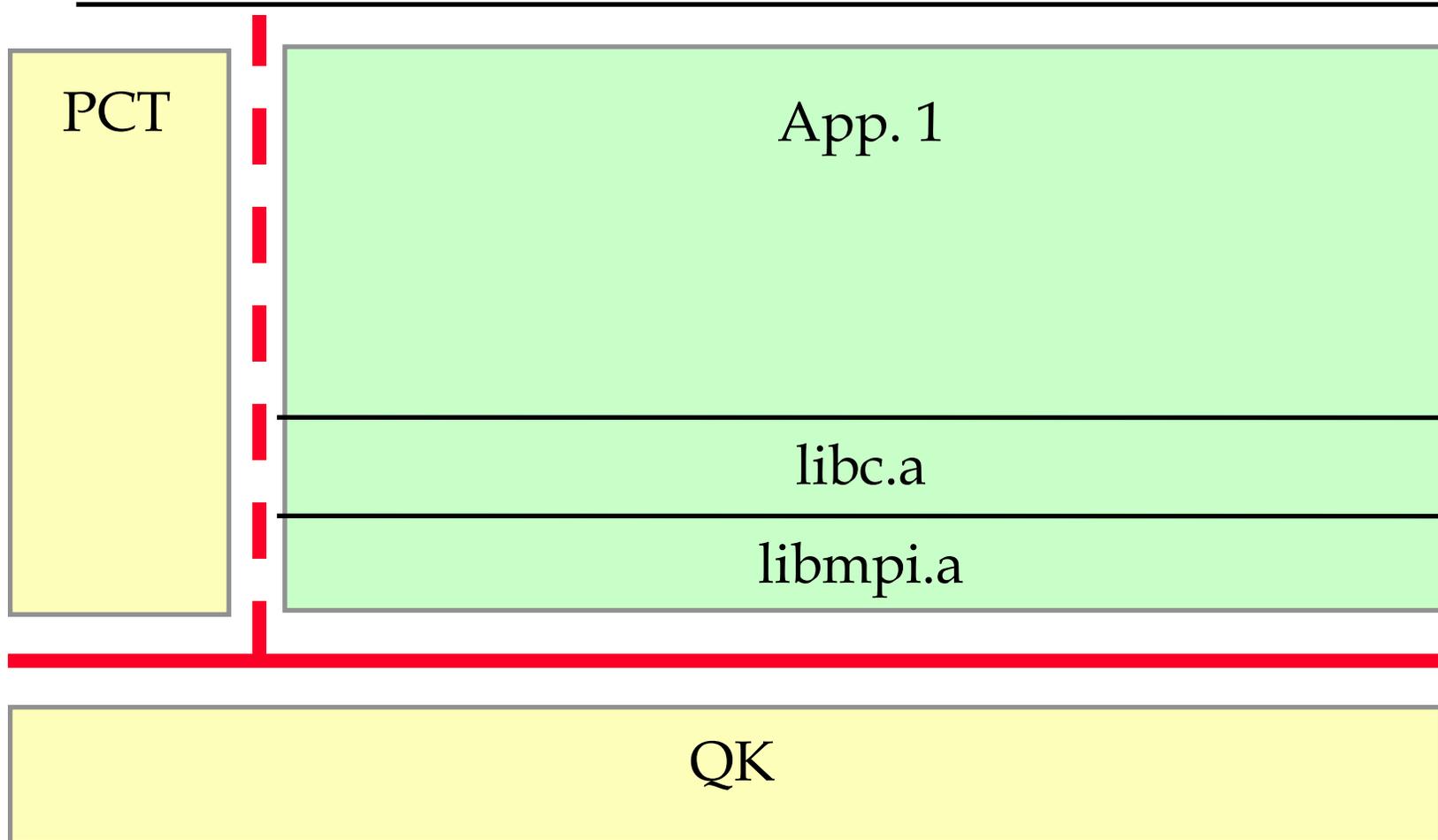


LWK General Structure





Typical Usage





Quintessential Kernel (QK)

- **Policy enforcer**
- **Initializes hardware**
- **Handles interrupts and exceptions**
- **Maintains hardware virtual addressing**
- **No virtual memory support**
- **Static size**
- **Non-blocking**
- **Small number of well-defined entry points**



Process Control Thread (PCT)

- **Runs in user space**
- **More privileged than user applications**
- **Policy maker**
 - **Process loading (with yod)**
 - **Process scheduling**
 - **Virtual address space management**
 - **Fault handling**
 - **Signals**



PCT (cont'd)

- **Customizable**
 - Singletasking or multitasking
 - Round robin or priority scheduling
 - High performance, debugging, or profiling version
- **Changes behavior of OS without changing the kernel**



Yod

- **Parallel job launcher**
- **Runs in the service partition**
- **Communicates via Portals**
- **Command line arguments for**
 - **Size of job**
 - **Processor mode**
 - **Stack size**
 - **Heap size**
 - **Sharing**
- **Services I/O and system call requests from compute node processes once job is running**



LWK Processor Modes

- Chosen at job launch time
- Heater mode (proc 0)
 - QK/PCT and application process on system CPU
- Message co-processor mode (proc 1)
 - QK/PCT on system CPU
 - Application process on second CPU
- Compute co-processor mode (proc 2)
 - QK/PCT and application process on system CPU
 - Application co-routines on second CPU
- Virtual node (VN) mode (proc 3)
 - QK/PCT and application process on system CPU
 - Second application process on second CPU

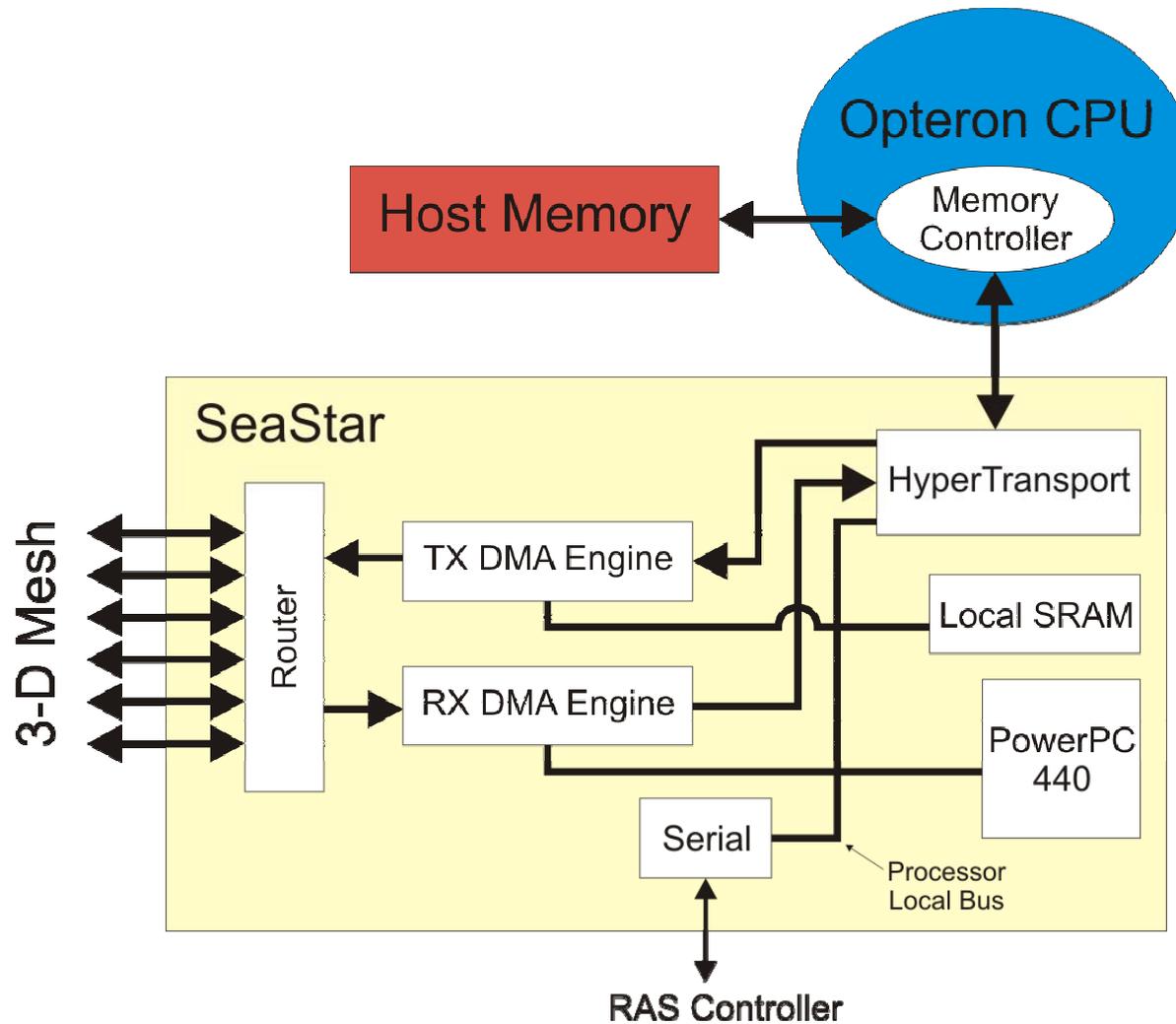


LWK Key Ideas

- **Protection**
 - Levels of trust
- **Kernel is small**
 - Very reliable
- **Kernel is static**
 - No structures depend on how many processes are running
- **Resource management pushed out to application processes and runtime system**
- **Services pushed out of kernel to PCT and runtime system**

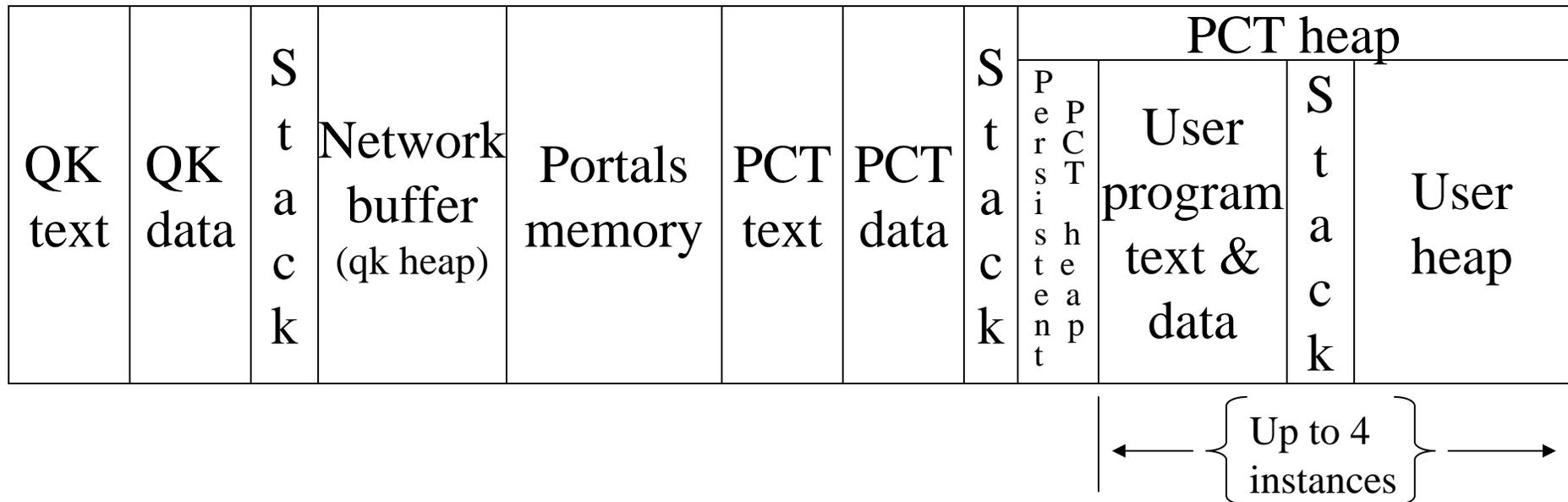


Red Storm/XT3 Node





Catamount LWK Physical Memory layout



Note: not to scale

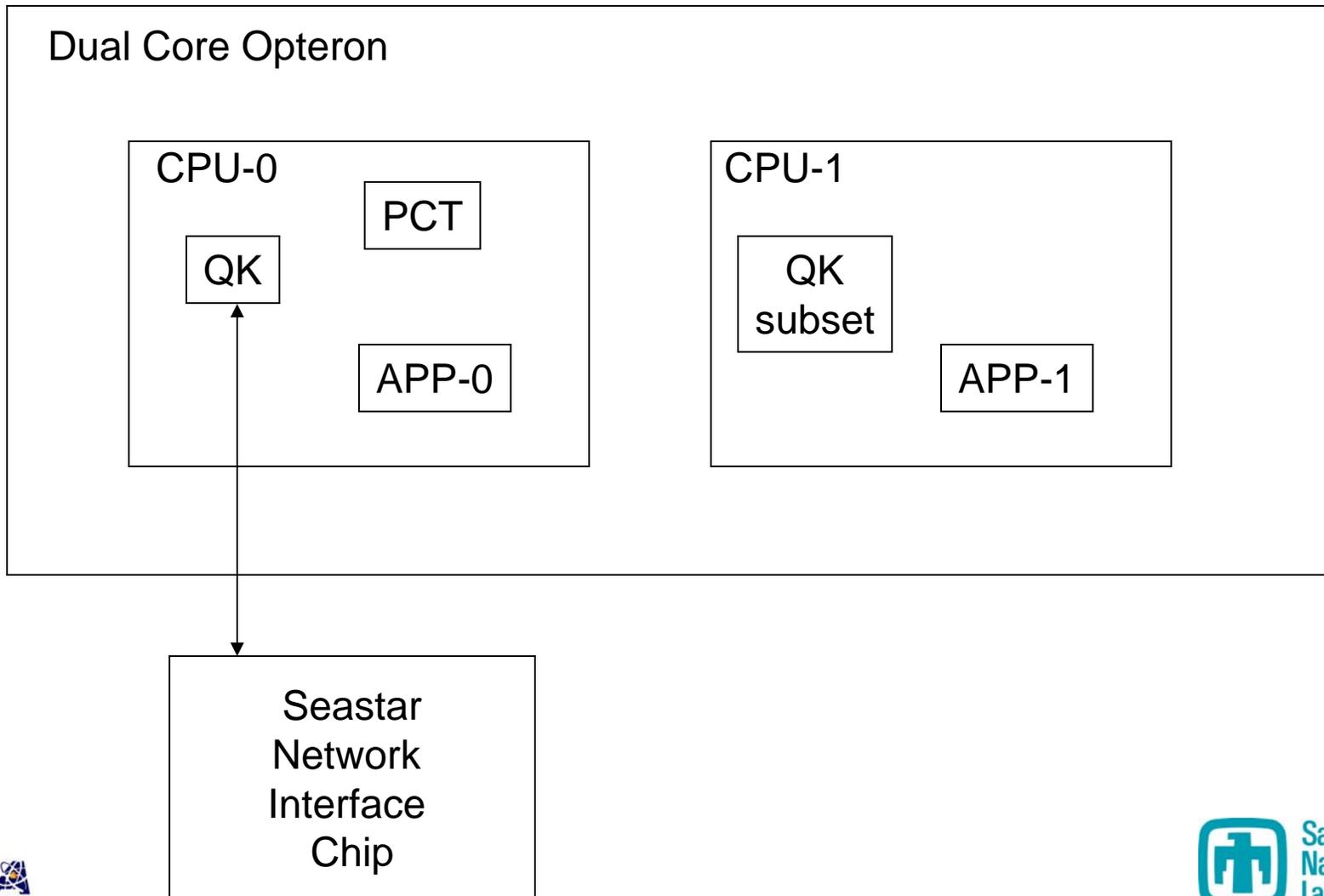


Catamount Dual Core Design

- **Virtual node mode**
- **Application perspective**
 - Twice as many nodes
 - Half the memory
 - No changes to existing applications
- **System perspective**
 - One copy of QK
 - One PCT
 - Network access done by CPU-0 QK only
 - Network requests from CPU-1 are proxied to CPU-0
- **Network perspective**
 - One Node Identifier
 - Two process Indices



Dual Core CPU Responsibility Assignments





Portals



Portals 0.0

- **SUNMOS (Sandia/UNM OS)**
 - Modeled on Vertex (the OS for the nCUBE)
 - Dynamic allocation for incoming messages
- **Experiments**
 - Multiple paths
 - Pre-posted receives
 - Use of a co-processor
- **nCUBE 2 and Intel Paragon**
 - Direct access to network FIFO's
 - Message co-processor (Paragon)



Portals 1.0

- **Moved all message reception structures to user space**
- **Kinds of portals**
 - **Kernel managed portals**
 - **Single block portals**
- **Never implemented**
- **Published 😊**



Portals 2.0

- **Puma/Cougar lightweight kernel**
- **Separate matching from memory descriptors**
- **Variety of memory descriptors**
 - **Kernel managed (dynamic)**
 - **Single block**
 - **Independent block**
 - **Combined block**
- **Intel TeraFLOPS (ASCI Red)**
 - **Direct access to message FIFO's**
 - **Message co-processor**



Issues with Portals 2.0

- **No API**
 - Data structures in user space
 - Protection boundaries have to be crossed to access data structures
 - Data structures have to be copied, manipulated, and copied back
 - Requires interrupts
- **Address validation/translation on the fly**
 - Incoming messages trigger address validation
 - Doesn't fit the Linux model of validating addresses on a system call for the currently running process



Portals 3.x

- **Operational API**
- **Unified memory descriptors**
- **Commodity processors and networks**
 - **Alphas, IA-32, IA-64, etc.**
 - **Linux OS with modules**
 - **Myrinet, Quadrics, etc.**
 - **DMA access to memory**
- **Fundamental change**
 - **NIC doesn't have logical address maps**
 - **NIC access to memory needs to be carefully managed**



Portals 3.3 Features

- **Best effort, in-order delivery**
- **One-sided operations**
 - **Put, Get, Atomic swap**
- **Supports zero-copy**
- **Supports OS-bypass**
- **Supports application offload**
 - **No polling or threads to move data**
 - **No host CPU overhead**
- **Well-defined transport failure semantics**
- **Unexpected operations are discarded**
- **Receive-side access control**
- **Runtime-system independent**



What Makes Portals Different?

- **Connectionless RDMA with matching**
- **Provides elementary building blocks for supporting higher-level protocols well**
 - MPI, RPC, Lustre, etc.
- **Allows structures to be placed in user-space, kernel-space, or NIC-space**
- **Receiver-managed offset allows for efficient and scalable buffering of MPI “unexpected” messages**
- **Supports multiple protocols within a process**
 - Needed for compute nodes where everything is a message



Portals Characteristics

- **Minimal library space**
 - Nothing depends on message size
 - All objects can be confirmed when created
- **Designed for library writers**
 - Not for application developers
 - **Low-level API**
 - We're happy to drop requests
 - Structures are complicated
 - Some functions (`Pt1MDUpdate()`) are not obvious
- **Designed to reflect underlying hardware**
 - NICs
 - Packets and failure
- **Provide the right amount of protection**



RDMA is the Wrong Model for MPI at Scale

- **Complexity of scalable connection management**
- **Lack of message matching ability leads to**
 - No progress without extra threads
 - Receive CPU overhead on every transfer
 - Severely limited ability to overlap computation with communication
- **Lack of scalable support for unexpected messages leads to**
 - Extra flow control in MPI library to manage unexpected message buffers
 - Inefficient use of application memory



Portals Was Designed for MPI at Scale

- **Connectionless**
- **Supports MPI matching semantics**
 - **Allows for offloading MPI matching to NIC**
 - **Very low CPU overhead for both small and large messages**
- **Maximizes overlap of computation and communication**
- **Provides scalable and efficient support for buffering unexpected messages**
 - **Does not require extra flow control in MPI**



Top Three Problems for Sandia



Top Three Problems For Sandia

- **Large multi-core processors**
 - **Maintaining system balance**
 - **Memory bandwidth**
 - **Network bandwidth**
 - **Programming model**
 - **MPI for intra-node communication won't work**
 - **Mixing MPI and OpenMP/threads has not worked well**
- **Parallel I/O and filesystems**
 - **Lustre works**
 - **Nothing seems to work well enough**



System Software R&D Projects

- **Short-term**
 - Enhancements to the LWK architecture to support large-scale multi-core processors
 - In-depth study of the impact of OS interference on applications
 - Explore and understand role of virtualization
- **Long-term**
 - Composable operating systems project



N-Way Lightweight Kernel Project

- **Catamount support for multi-core processors and multi-processor nodes**
- **Exploring enhancements to LWK to better support non-MPI programming models**
- **Identified candidate network stack architectures**
- **Created prototype NIC-based network stack for Red Storm and evaluated its performance**
- **Developed prototype LWK platform to examine commodity PC bootstrap requirements**



Two Tough Questions

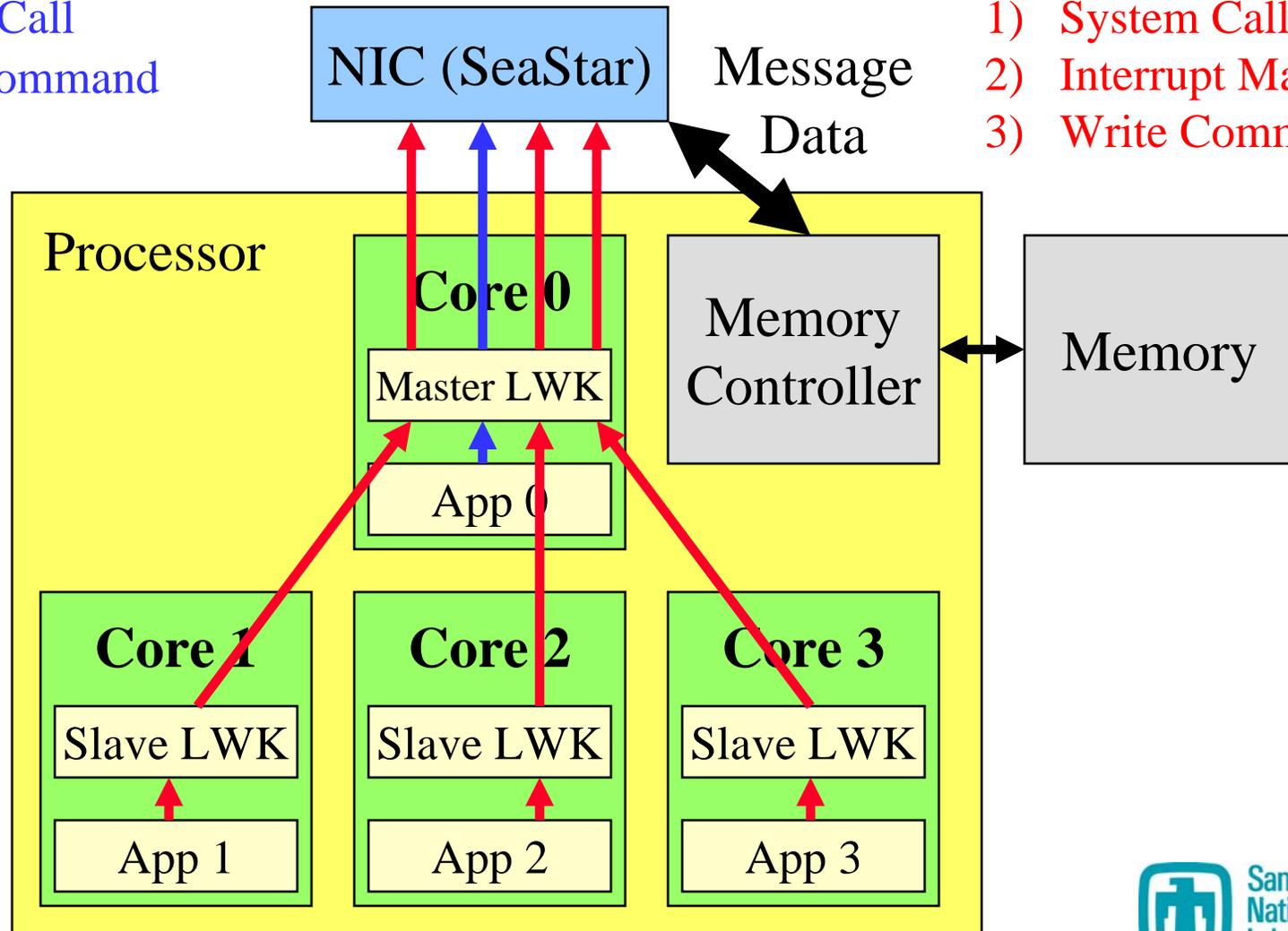
- **What is the programming model?**
 - Must support traditional MPI model
 - Other shared address space based models
- **What is the architecture of the network stack?**
 - Host-based vs. NIC-based
 - Intra-node communication path
- **Both questions are interrelated**



Today: Virtual Node Mode with a Asymmetric Host-based Network Stack

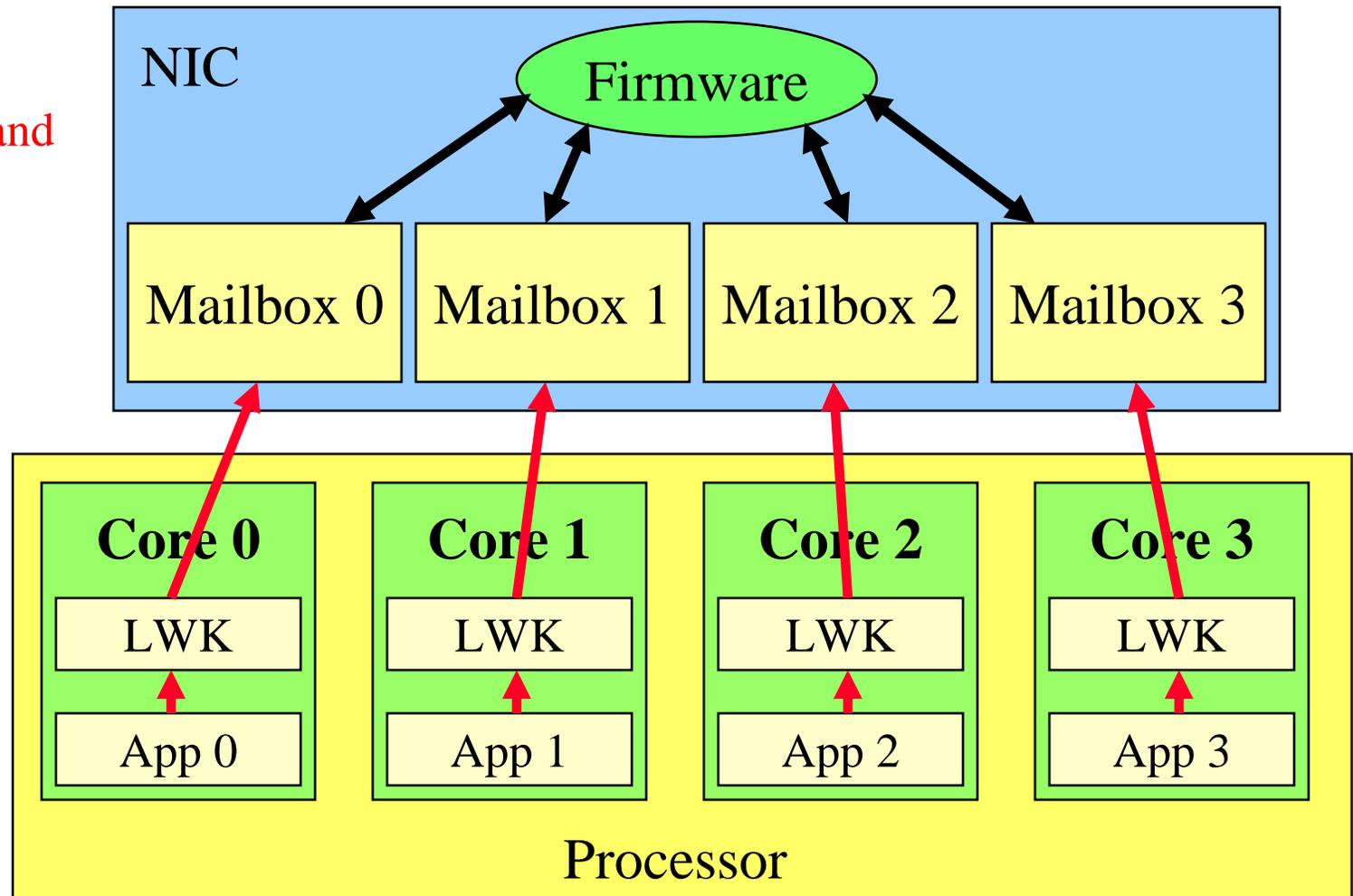
- 1) System Call
- 2) Write Command

- 1) System Call
- 2) Interrupt Master
- 3) Write Command



Virtual Node Mode with a Symmetric Host-based Network Stack

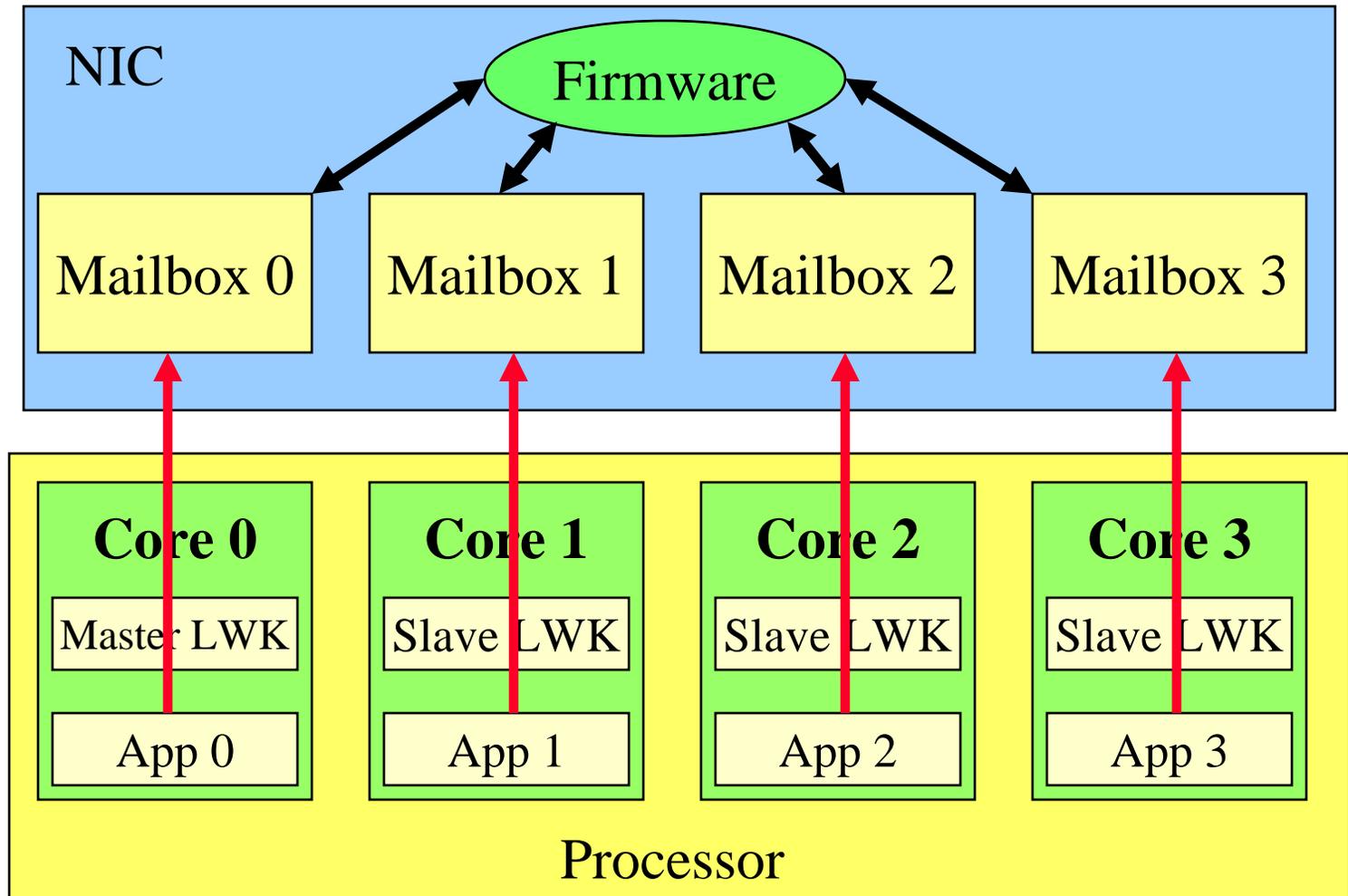
- 1) System Call
- 2) Write Command





Virtual Node Mode with a NIC-based Network Stack

1) Write Command





Prototype NIC-based Network Stack

- **Allowed characterization of**
 - Impact of interrupts on latency
 - Impact on throughput (Messages/second)
 - NIC vs. host CPU matching speed
 - Penalty of having multiple NIC mailboxes
 - NIC resource requirements of each CPU core
- **4 cores/SeaStar probably upper bound**
- **IEEE Micro paper accepted:**
 - “Cray’s SeaStar Interconnect: Balanced Bandwidth for Scalable Performance”, Ron Brightwell, Trammell Hudson, Kevin Pedretti, Keith Underwood, May/June 2006.



Upcoming Challenges

- **Support for non-MPI programming models**
 - **OpenMP + MPI**
 - **User managed threads**
 - **Global Arrays**
 - **SHMEM**
- **Current thinking is two new LWK capabilities are needed:**
 - **Support for shared address spaces**
 - **Support for dynamic process creation**

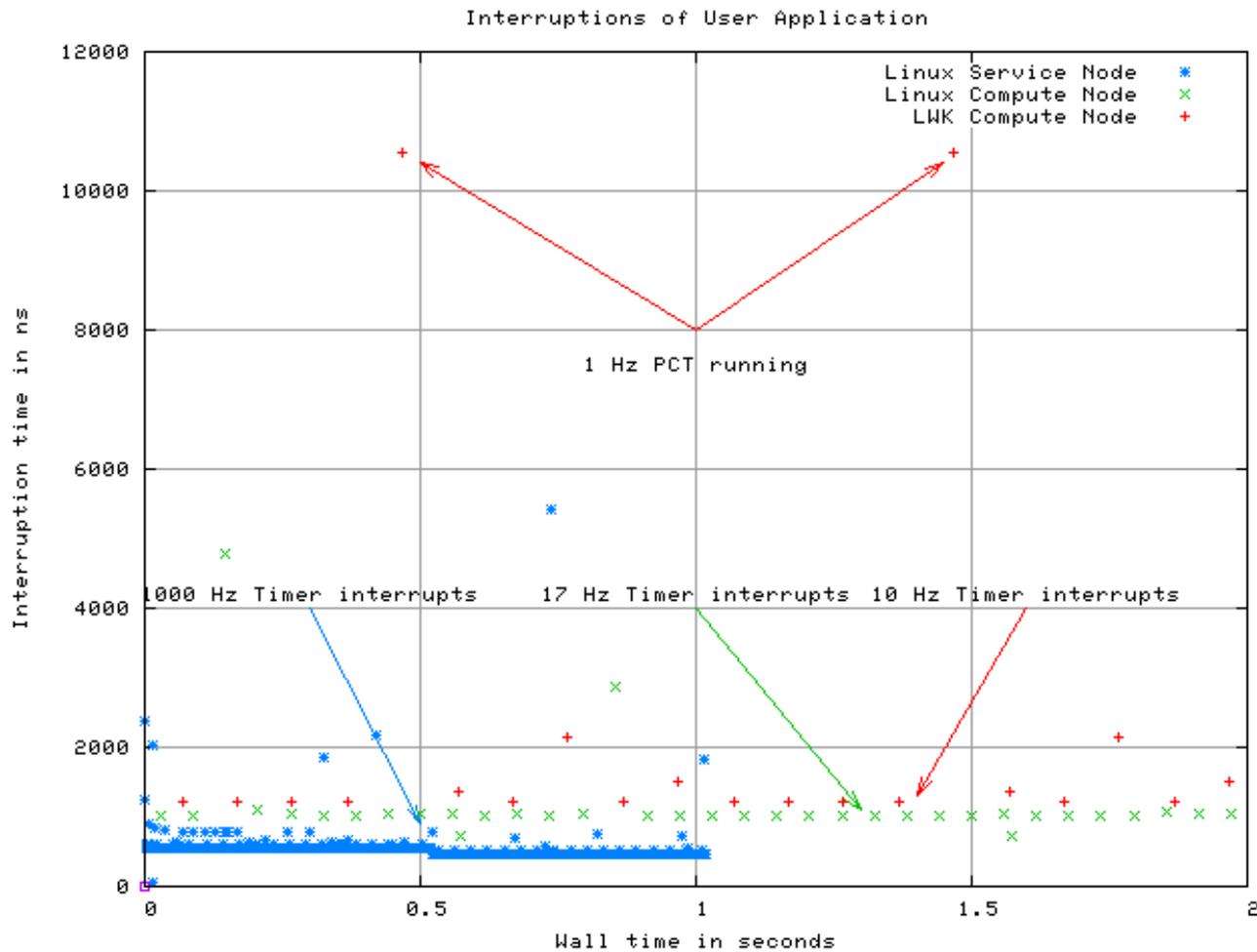


Quantifying the Impact of OS Interference

- **Still many unanswered questions in the HPC OS research community**
- **Popularity of the XT3 platform has increased the debate on the impact of OS “noise”**
- **Important question is the sensitivity of Sandia applications to interference rather than amount of interference in the OS**
 - **Currently pursuing a strategy that will allow for injecting interference into Catamount in a controlled fashion**
 - **Also plan to artificially degrade network performance to understand the impact of interference on unbalanced systems**



Selfish Benchmark



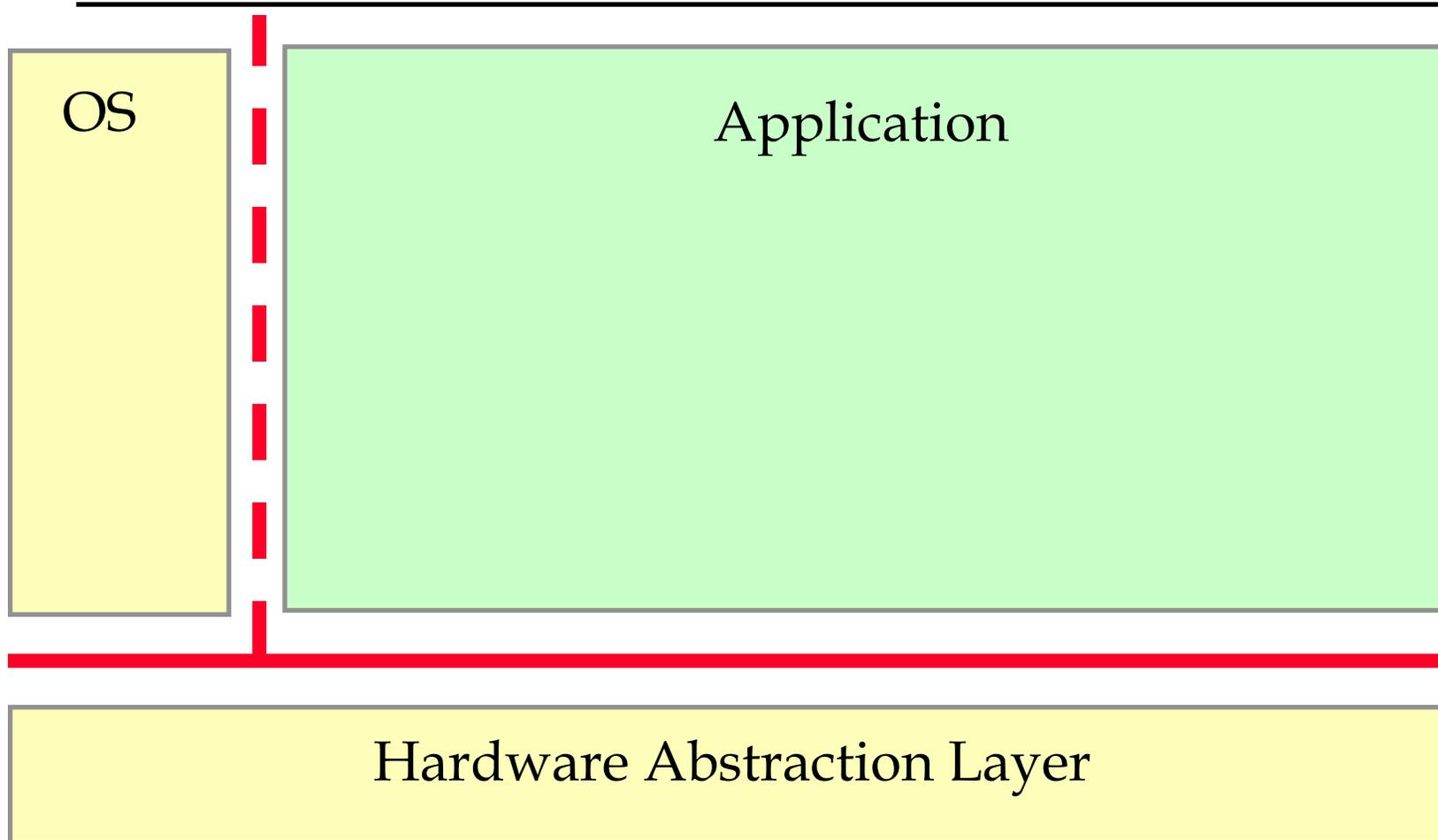


The Role of Virtualization

- **Exploring the Xen virtualization technology**
 - Ported OpenCatamount to Xen
 - Ported Xen to XT3 hardware
 - Quantifying the impact of Xen on performance
- **Evaluating the possibility of augmenting Catamount to support virtualization**
 - Current design of Catamount is very similar to a virtualization layer
 - Virtualization would support running native Catamount applications as well as Linux applications on XT3 compute nodes without rebooting



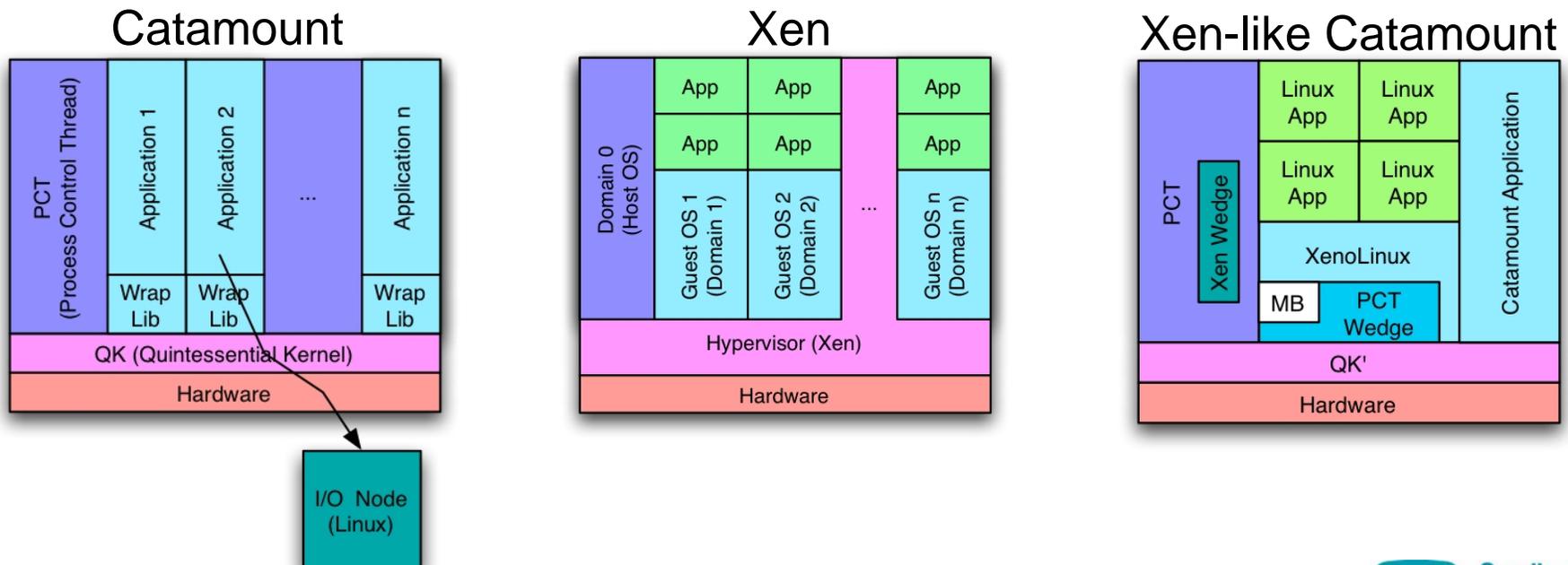
QK is Really a HAL





Catamount Already Looks Like a VMM

- Xen hypervisor interface support in Catamount
 - Catamount applications run without performance impact
 - Linux applications run on XenoLinux guest OS





Where We Don't Need Virtualization

- **Processor**
 - Can't leverage processor-specific features
 - i860 bus locking
- **Memory**
 - Linux already makes everything look like an x86
 - We already have enough problems with tracking memory usage
 - Applications always know better how to do resource allocation
- **Network**
 - No good way to provide isolation with network virtualization



Why We Might Need Virtualization

- **OS development**
 - Use VMM as hardware abstraction layer
 - No need to port to every new machine
 - Debugging
 - Easily capture entire state
 - Testbeds
- **OS comparison**
 - HAL makes direct OS performance comparison a little easier
 - Porting the OS isn't the issue – it was the network
- **Checkpoint/Restart/Migration**
 - For those who want this in the first place



Possible Benefits of Catamount Virtualization

- **Supports the Red Storm requirement for running a single job that spans both compute and service partitions**
 - Dynamically sizable service partition
- **May allow for running Accelerated Portals under Linux**
 - Provide a hook for XenLinux to acquire physically contiguous memory
- **Virtualizing network may help with fault tolerance**



Composable OS Project

- **Funded DOE/MICS FAST-OS program**
- **Collaboration with the University of New Mexico (Barney Maccabe) and LSU (Thomas Sterling)**

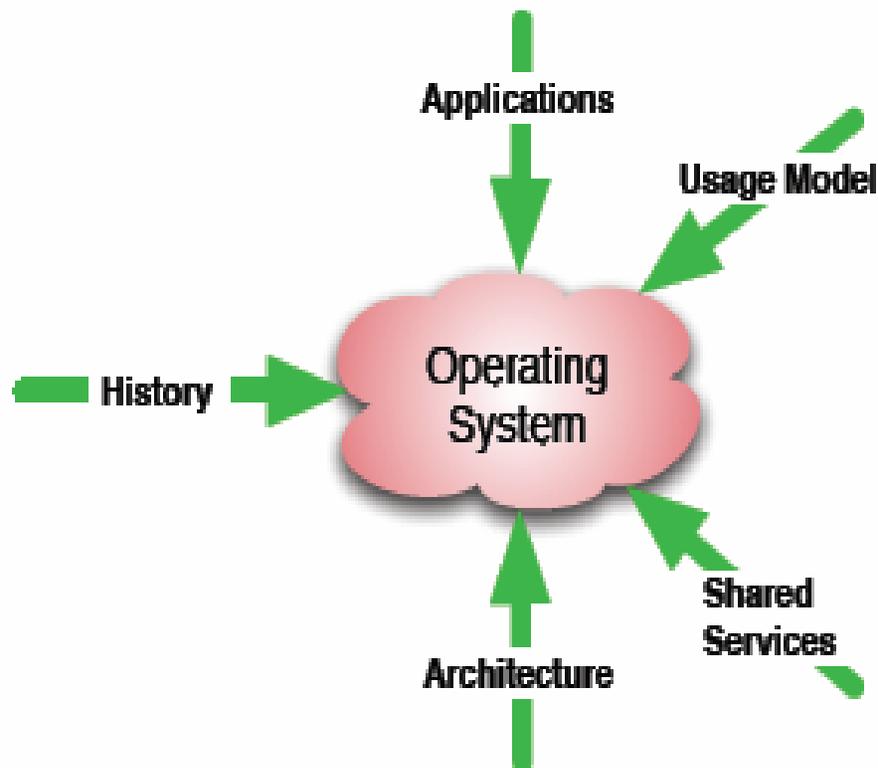


OS Issues

- **General-purpose operating systems**
 - **Generality comes at the cost of performance for *all* applications**
 - **Assume a generic architectural model**
 - **Difficult to expose novel features**
- **Lightweight operating systems**
 - **Limited functionality**
 - **Difficult to add new features**
 - **Designed to be used in the context of a specific usage model**
- **Operating system is an impediment to new architectures and programming models**



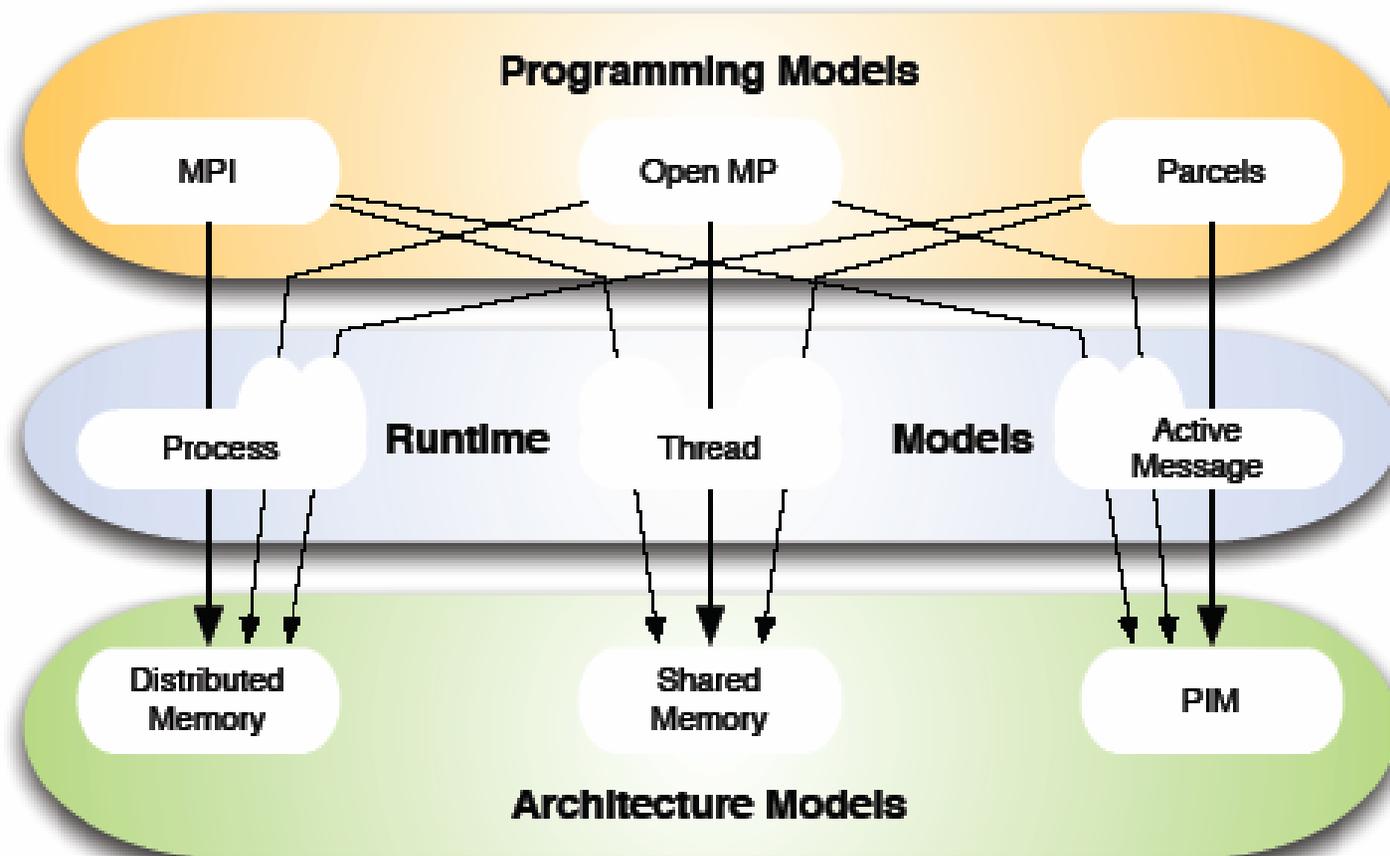
LWK Influences



- **Lightweight OS**
 - Small collection of apps
 - Single programming model
 - Single architecture
 - Single usage model
 - Small set of shared services
 - No history
- **Puma/Cougar**
 - MPI
 - Distributed memory
 - Space-shared
 - Parallel file system
 - Batch scheduler

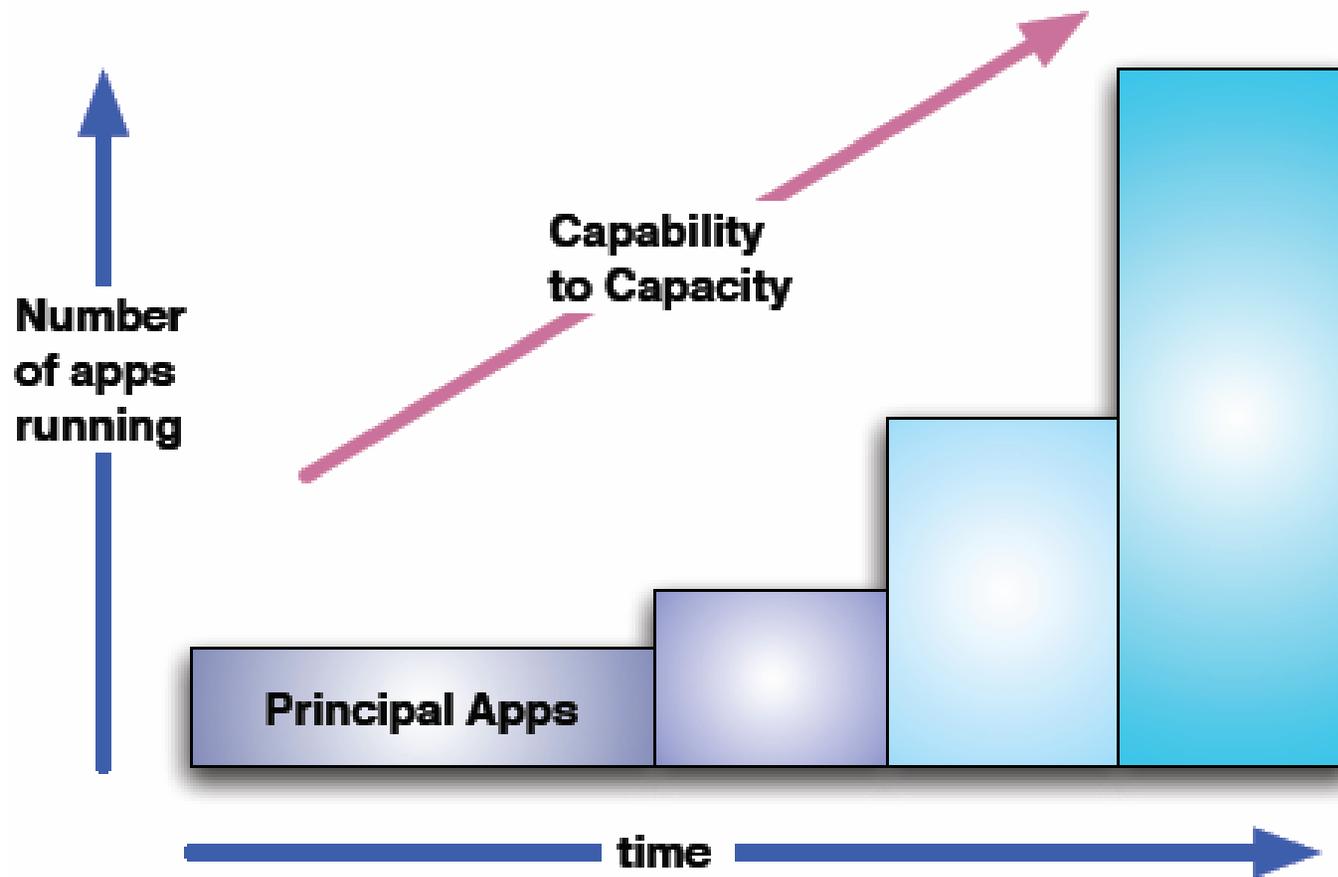


Programming Models





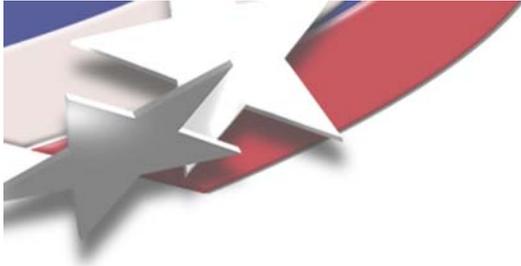
Usage Models





Current and Future System Demands

- **Architecture**
 - Modern ultrascale machines have widely varying system-level and node-level architectures
 - Future systems will have further hardware advances (e.g., multi-core chips, PIMs)
- **Programming model**
 - MPI, Thread, OpenMP, PGAS, ...
- **External services**
 - Parallel file systems, dynamic libraries, checkpoint/restart, ...
- **Usage model**
 - Single, large, long-running simulation
 - Parameter studies with thousands of single-processor, short-running jobs



Project Goals

- **Realize a new generation of scalable, efficient, reliable, easy to use operating systems for a broad range of future ultrascale high-end computing systems based on both conventional and advanced hardware architectures and in support of diverse, current and emerging parallel programming models.**
- **Devise and implement a prototype system that provides a framework for automatically configuring and building lightweight operating and runtime system based on the requirements presented by an application, system usage model, system architecture, and the combined needs for shared services.**

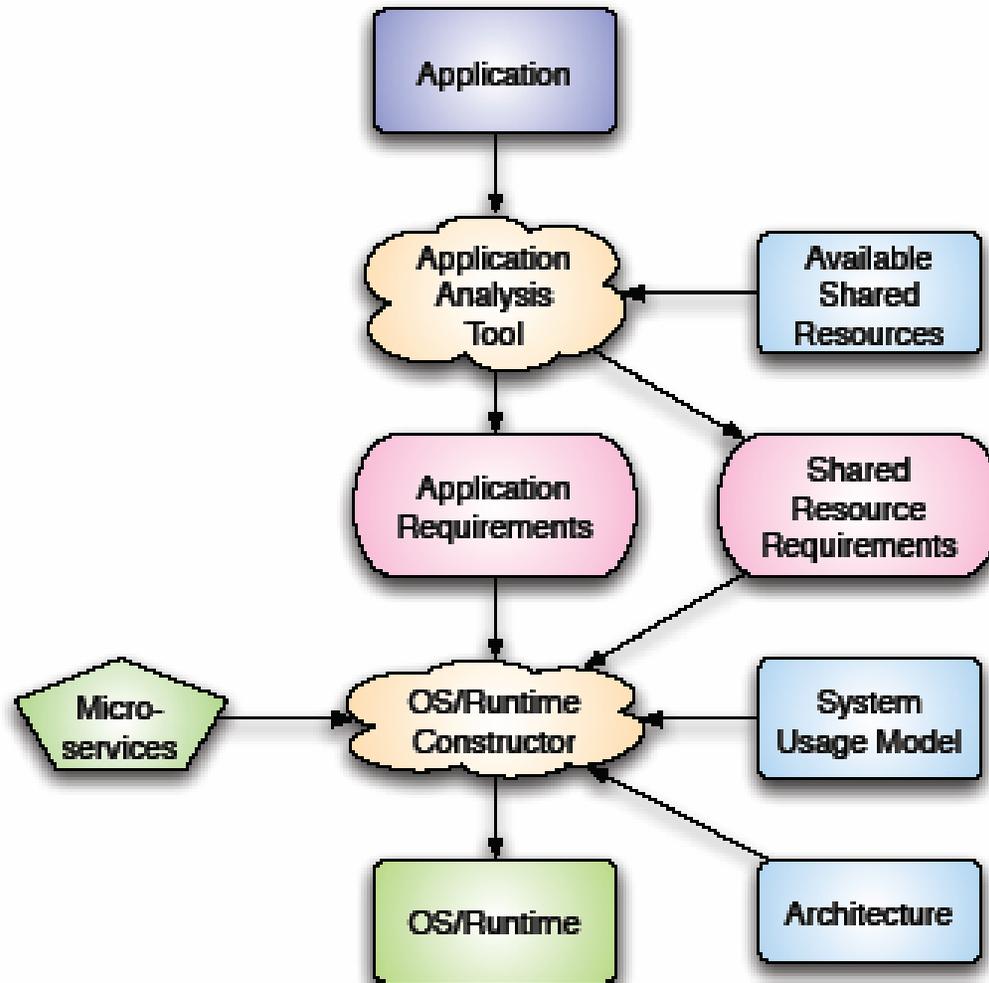


Approach

- **Define and build a collection of micro-services**
 - **Small components with well-defined interfaces**
 - **Implement an indivisible portion of service semantics**
 - **Fundamental elements of composition and re-use**
- **Combine micro-services specifically for an application and a target platform**
- **Develop tools to facilitate the synthesis of required micro-services**



Building Custom Operating/Runtime Systems





THINK Framework

- **Think Is Not a Kernel**
- **A component-based framework for operating systems development**
 - **Set of components (library) and tools (for compiling, linking, etc)**
- **Allow for building complete OS by assembling software components**
 - **No predefined kernel structure (e.g. monolithic/micro/exo kernels)**
 - **No predefined component size**
 - **Mutex as well as protocol stack are components**
 - **No predefined core functionalities**
 - **Scheduler, mmu, etc.**



Motivation and Benefits of THINK

- **Need to build OS *a la carte***
 - Don't want to pay the cost of unneeded functionalities
- **Mastering complexity of OS building**
 - Provide homogeneous vision of OS topology
 - Facilitate code reuse
 - Insure correct assembly of components
 - Facilitate integration of non-functional properties
 - QoS, real-time, security, etc.



More THINK Details

- **Uses low-level languages (ASM/C)**
- **Preliminary work targeted at embedded systems**
 - **ARM/Lego, RCX/DSP**
- **Aimed at reducing component overhead**
 - **Smaller components have higher overhead**
 - **Trade-off between code reuse, level of dedication and run-time overhead**
 - **Average of 2% more memory in the actual implementation**
- **Developed by France Telecom R&D**
 - **Available at <http://think.objectweb.org>**



Acknowledgments

- **Sandia Scalable Computing Systems**
 - Kevin Pedretti, Rolf Riesen, Keith Underwood
- **University of New Mexico**
 - Barney Maccabe, Patrick Bridges, Jean-Charles Tournier
- **Louisiana State University**
 - Thomas Sterling
- **OS Research**
 - Trammell Hudson