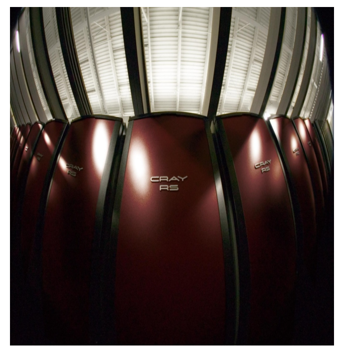
Exceptional service in the national interest

Sandia National Laboratories



Software Tools and Techniques for HPC, Clouds, and Server-Class SoCs

Ron Brightwell

R&D Manager, Scalable System Software Department

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Existing Approach for Runtime Systems

- Generalized abstractions and machine models that allow algorithm designers and application developers to create code that works reasonably well on a broad spectrum of systems
- Compilers, libraries, RTS, and OS work within the constraints of these abstractions to map the application to the underlying hardware as efficiently as possible
- Performance tools identify shortcomings in the mapping
- Refine the mapping on a per-platform basis
- Adjust the abstractions and models in response to evolving hardware
- Leverage RTS adaptivity within bounded set of resources and relatively fixed cost models

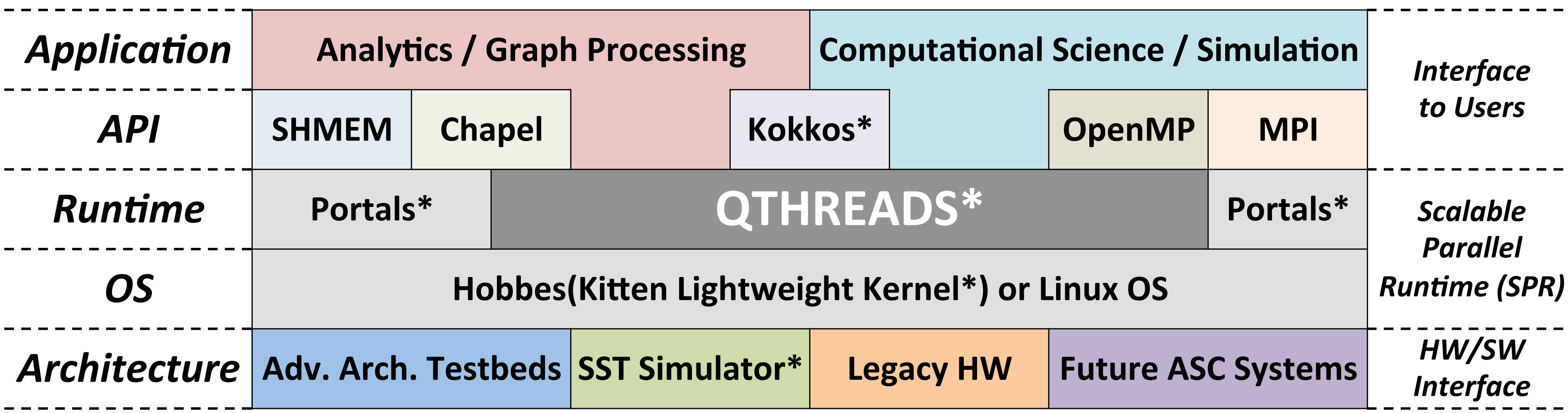
Vision for Exascale Runtime Systems

- Responsible for mapping the machine to the application
- Requires dynamic discovery
  - Determine the goals of the application
  - Develop knowledge on how well resources are being used
  - Make informed optimization decisions
  - Understand behavior in response to decisions
  - Adapt to constantly changing cost models
- Respond to elastic system and application resources
- Richer abstractions and models at the system level
- Improve the productivity of application and library developer as well as the scalability and efficiency of the system

Applications and Usage Models are Diverging

- Application composition becoming more important
  - Ensemble calculations for uncertainty quantification
  - Multi-{material, physics, scale} simulations
  - In-situ analysis and graph analytics
  - Performance and correctness analysis tools
- Applications may be composed of multiple programming models
- More complex workflows are driving need for advanced OS services and capability
  - "Workflow" overtaken "Co-Design" as most popular DOE buzzword ☺
- Desire to support "Big Data" applications
  - Significant software stack comes along with this
- Support for more interactive workloads
- Requirements are independent of programming model and hardware

Sandia Research System Software Stack



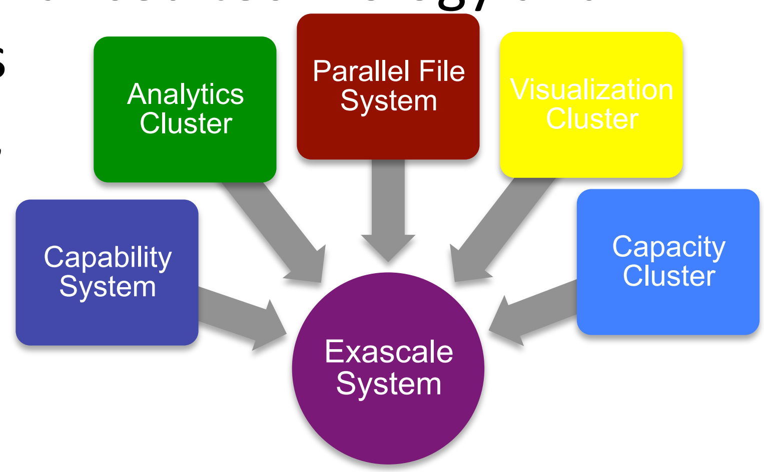
* Sandia-based software / API

Qthreads System Model

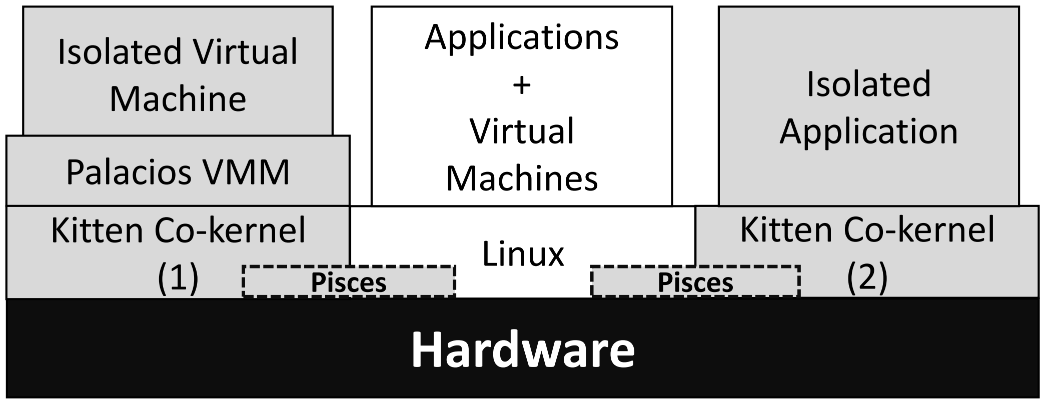
- The programmer exposes application parallelism as massive numbers of lightweight tasks (qthreads).
  - Problem-centric rather than processor-centric work decomposition to enhance productivity with transparent scaling
    - Both loop-based and task-based parallelism supported
  - Full/empty bit primitives for powerful, lightweight synchronization
    - Emulates behavior of Cray XMT (ThreadStorm) architecture
  - C API with no special compiler support required
- The run time system dynamically manages the scheduling of tasks for locality and scalable performance.
  - Heavyweight worker pthreads to execute the user's tasks
    - Worker pthreads pinned onto underlying hardware cores
    - Architecture-aware mapping of workers to hardware (e.g., NUMA or Phi)
  - Lightweight task switching

Systems Are Converging to Reduce Data Movement

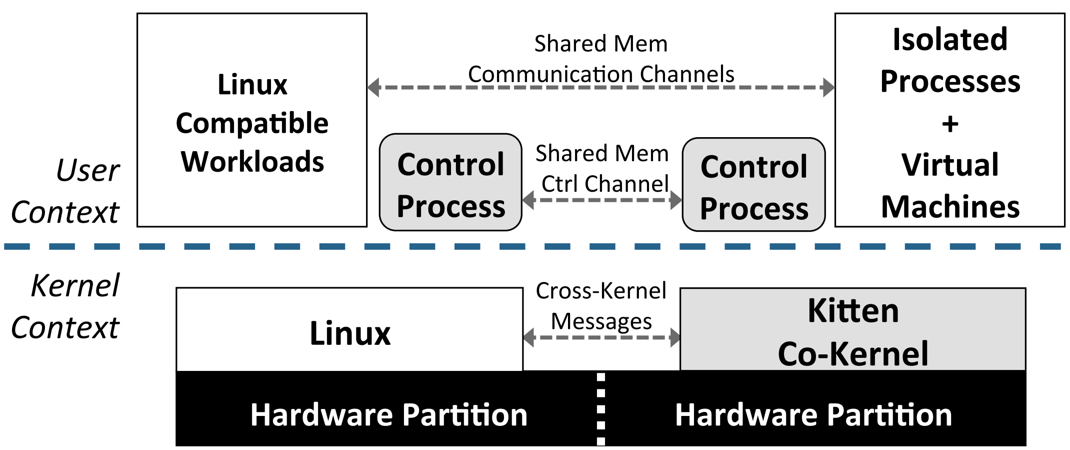
- External parallel file system is being subsumed
  - Near-term capability systems using NVRAM-based burst buffer
  - Future extreme-scale systems will continue to exploit persistent memory technologies
- In-situ and in-transit approaches for visualization and analysis
  - Can't afford to move data to separate systems for processing
  - GPUs and many-core processors are ideal for visualization and some analysis functions
- Less differentiation between advanced technology and commodity technology systems
  - On-chip integration of processing, memory, and network
  - Summit/Sierra using InfiniBand



Hobbes Node OS Architecture



1. Co-Kernel Architecture, Three Enclave Example



2. Cross-enclave communication used for enclave control and for cross-enclave app code coupling

- Co-kernels: Multiple OS kernels run side-by-side on same node in different enclaves
- Pisces infrastructure used to launch and manage enclaves and bind enclaves together
- XEMEM mechanism developed to enable cross-enclave memory sharing

Hardware Support for OS/Runtime and Interconnect

- Fast context switching of tasks
- Lightweight synchronization between tasks
- Fast task creation on network events
- Hardware queues (tasks and data)
- Isolation mechanisms (Qos)
  - Memory system partitioning
  - Network (Noc/NIC) partitioning
- Sharing mechanisms
  - Shared memory
  - Lightweight signaling
- Intra-node data movement (pt2pt, collective)
- Flexible memory translation capability (segments and pages)

Hardware Support for OS/Runtime and Interconnect (cont'd)

- Lightweight power management/control
- More sophisticated hw error management/control
- Performance information and correlation (memory, cores, NoC, NIC)
- Debugging support features
- Endpoint virtualization (translation)
- Parallelism in the NIC
- Hardware support for active messages
  - Hardware queues
  - Flow control
- Support for non-contiguous data (scatter/gather)