SAND2015-1864PE

ASC Panel on Runtime System Topics

ECI Runtime Systems Workshop
March 11-13, 2015 – Rockville, MD

Kevin Pedretti
Sandia National Laboratories
ktpedre@sandia.gov

Sandia National Laboratories

Exceptional service in the national interest

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. SAND NO. 2011-XXXXP

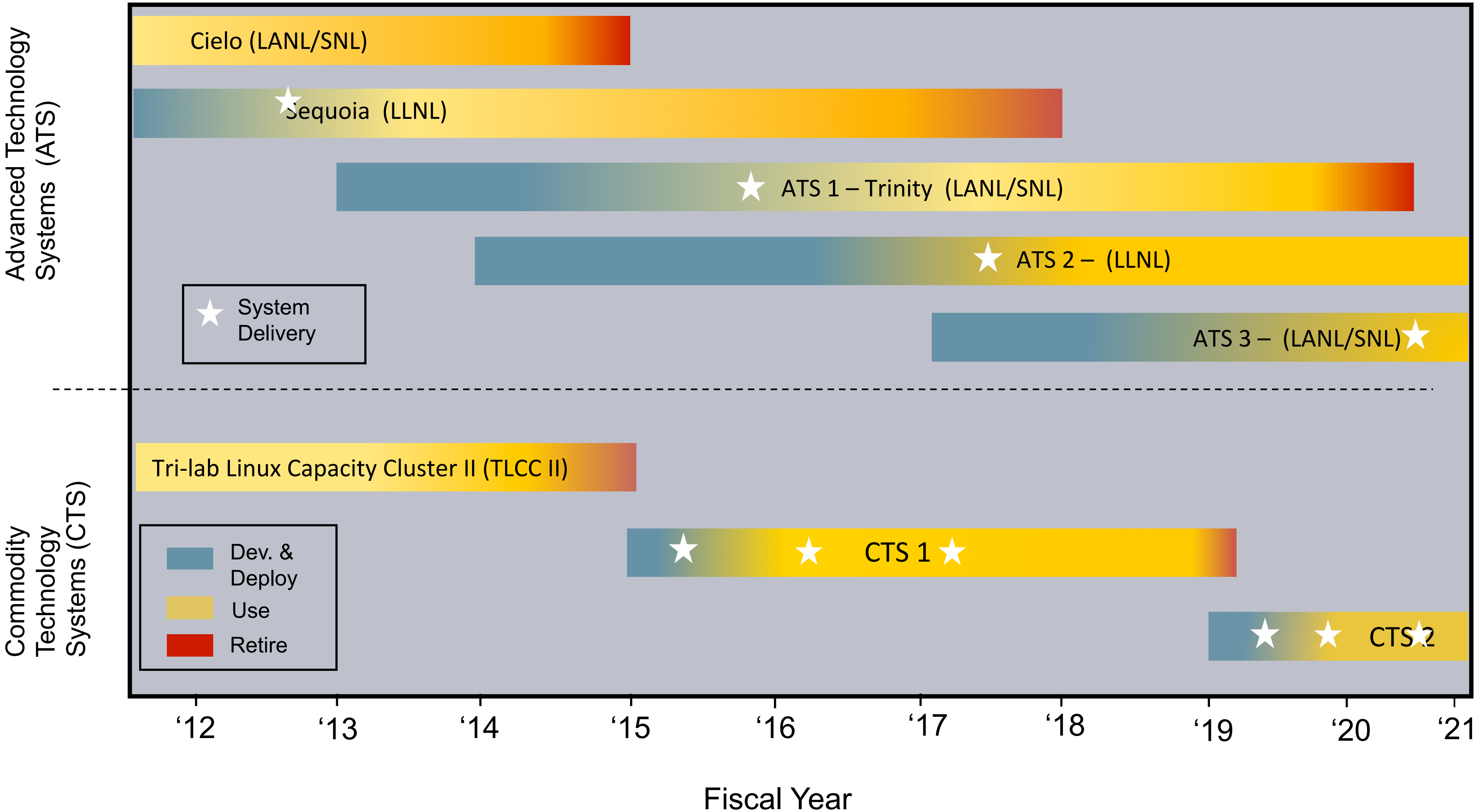
Outline

- Introduction
- Runtime system related activities at Sandia
  - ATDM
    - Qthreads / Kokkos
    - DHARMA
  - Power API / Trinity NRE
- Address workshop questions and issues
  - Sys Arch / Asynchrony
  - Sys Arch / Relationship between OS and Runtime
  - Evaluation

ASC Computing Strategy

- Approach: Two classes of systems
  - Advanced Technology: First of a kind systems that identify and foster technical capabilities and features that are beneficial to ASC applications
  - Commodity Technology: Robust, cost-effective systems to meet the day-to-day simulation workload needs of the program
- Investment Principles
  - Maintain continuity of production
  - Ensure that the needs of the current and future stockpile are met
  - Balance investments in system cost-performance types with computational requirements
  - Partner with industry to introduce new high-end technology constrained by life-cycle costs
  - Acquire right-sized platforms to meet the mission needs

ASC Platform Timeline



Outline

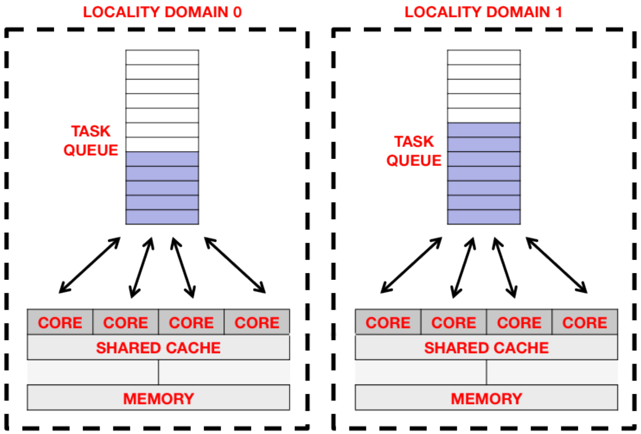
- Introduction
- Runtime system related activities at Sandia
  - ATDM
    - Qthreads / Kokkos
    - DHARMA
  - Power API / Trinity NRE
- Address workshop questions and issues
  - Sys Arch / Asynchrony
  - Sys Arch / Relationship between OS and Runtime
  - Evaluation

Advanced Technology Development and Mitigation (ATDM)

- ATDM is a new Tri-lab ASC program element addressing challenges of next generation platforms, leading path to useful exascale
  - Massive concurrency, fat nodes
  - Heterogeneous architectures, parallelism, performance, …
  - Multi-level memory hierarchies
  - Data movement: in-situ/transit analysis, workflows
- SNL effort focusing on applications important to ASC
  - Building from ground up over task-based programming model
  - Building supporting RTS and software infrastructure
- SNL runtime system activities
  - Kokkos – on-node parallelism, data parallel, data virt (PI: Carter Edwards)
  - DHARMA – distributed asynchronous many-task RTS (PI: Janine Bennett)
  - Qthreads being used to add tasking to Kokkos (LDRD)

Qthreads: Lightweight On-node Thread Runtime

SNL contacts: Dylan Stark, Stephen Olivier



- **Model:**
  - Somebody (app/runtime/compiler/…) exposes massive numbers of lightweight tasks (qthreads)
  - The qthreads dynamic runtime system manages the scheduling of tasks for locality and performance
- **Capabilities:**
  - Supports loop-based and task-based parallelism
  - Full/empty bit primitives for lightweight synchronization (emulates Tera/Cray MTA/XMT)
  - Locality-aware load balancing of tasks to support NUMA and complex cache hierarchies
  - Easy to embed in higher-level runtimes, C API with no special compiler support
- **Usage:**
  - Research: locality-based scheduling, dynamic concurrency throttling, task parallel over decomposition, incorporating task parallelism into Kokkos
  - OpenMP over Qthreads (using ROSE/XOMP and Intel frontends)
  - Default tasking layer in Chapel

Kokkos Task Parallel API (LDRD)

Existing SNL Technologies: Kokkos & Qthreads

Kokkos C++ API for efficient manycore data-vector parallelism	Qthreads multithreading library for scalable task parallelism

Development of New Capabilities

Extend Kokkos API for task parallelism and graph processing	Extend Qthreads for nested data parallelism, Phi, GPU tasks

Goal: Unified Task-Data-Vector Manycore API

Performance portable C++ API for CSE and graph applications

ATDM DHARMA project: Distributed asyncHronous Adaptive Resilient Management of Applications

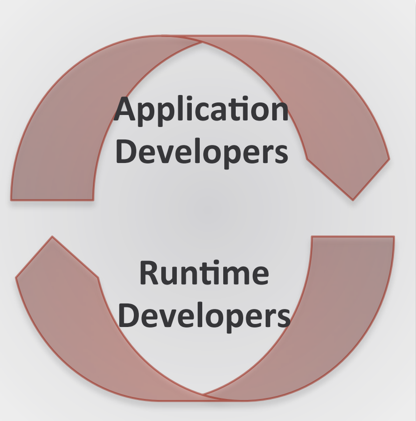
Sandia National Laboratories

Janine Bennett (PI), Jeremiah Wilke (Chief Architect), Robert Clay (PM), Ken Franko, Hemanth Kolla, Paul Lin, Greg Sjaardema, Nicole Slattengren, Keita Teranishi

- **Project Mission:** Assess & address fundamental challenges imposed by the need for performant, portable, scalable, fault-tolerant programming models at extreme-scale

FY15 ASC Level 2 MIlestone	Assess rich feature sets/usability/performance of existing Asynchronous Many-Task (AMT) runtimes in context of ASC workloads
Level 2 & DHARMA runtime	Research in <u>programmability</u>, <u>dynamic load-balancing</u>, and <u>fault-tolerance</u> of AMT runtimes

- **Current Activities:**
  - Implementing miniAero in Charm++, Legion, Unitah; Evaluate performance, programmability, mutability
  - Held coding bootcamps at U. Utah, Stanford, SNL/CA
  - Build-out of DHARMA v1.0 AMT runtime, transparently handle fail-stop node crashes



PowerAPI: Portable Power Management

- Need portable way to measure and control power
  - Today there are several power interfaces, every system is different
  - This makes it harder to write runtimes, tools, apps, …
- Power API fills this gap, input from community and vendors (FY14 L2 milestone)
  - Covers broad spectrum of use cases, from platform-level, to resource manager, to runtime system, to OS, to applications
  - Will be implemented for upcoming Trinity system
  - Expect to be there on future DOE/NNSA ATS systems
  - Will evolve over time

SANDIA REPORT

SAND2014-17061
Unlimited Release
Printed August 2014

High Performance Computing - Power Application Programming Interface Specification Version 1.0

James H. Laros III, David DeBonis, Ryan Grant, Suzanne M. Kelly, Michael Levenhagen, Stephen Olivier, Kevin Pedretti

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

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.

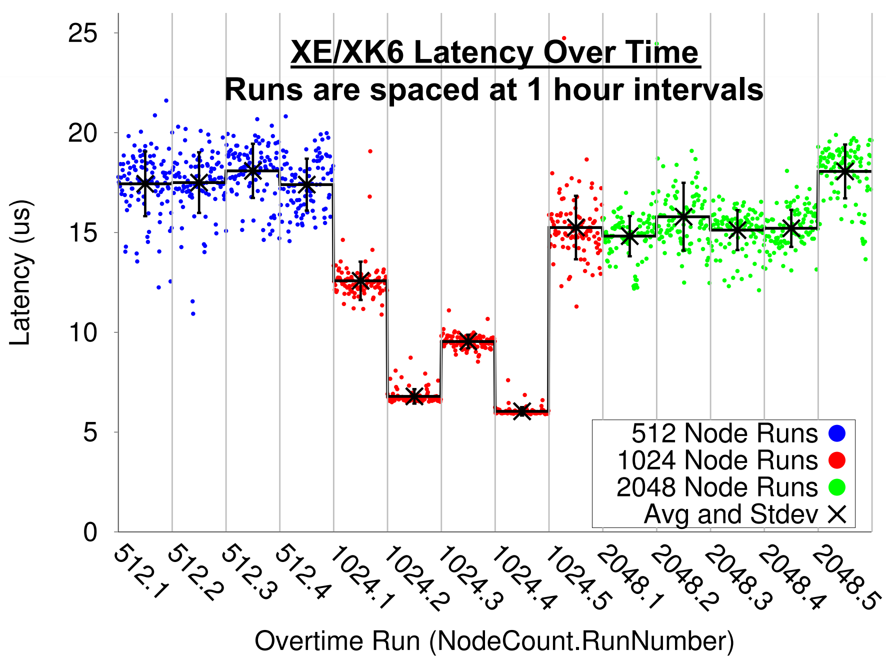
Sandia National Laboratories

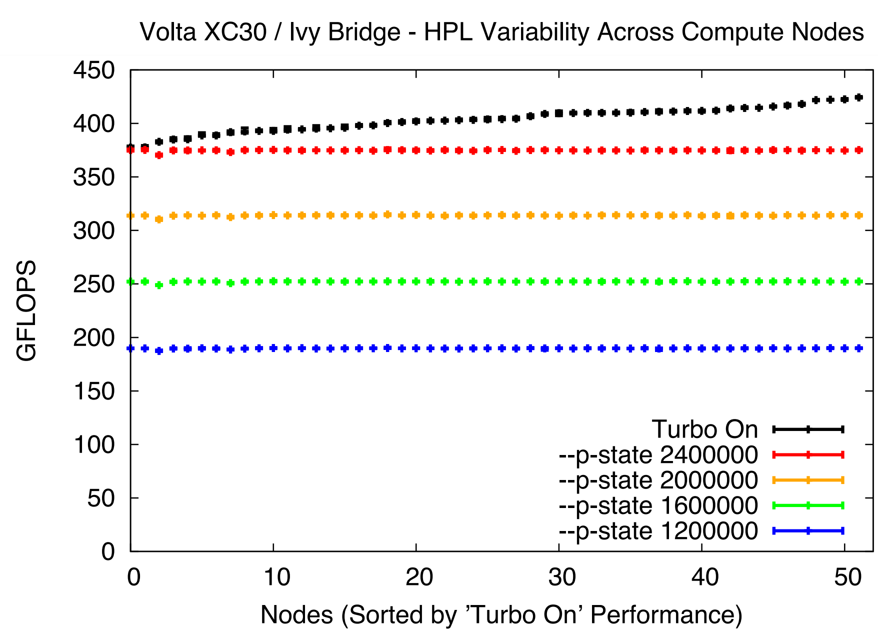
Outline

- Introduction
- Runtime system related activities at Sandia
  - ATDM
    - Qthreads / Kokkos
    - DHARMA
  - Power API / Trinity NRE
- Address workshop questions and issues
  - Sys Arch / Asynchrony
  - Sys Arch / Relationship between OS and Runtime
  - Evaluation

Sys Arch: Asynchrony

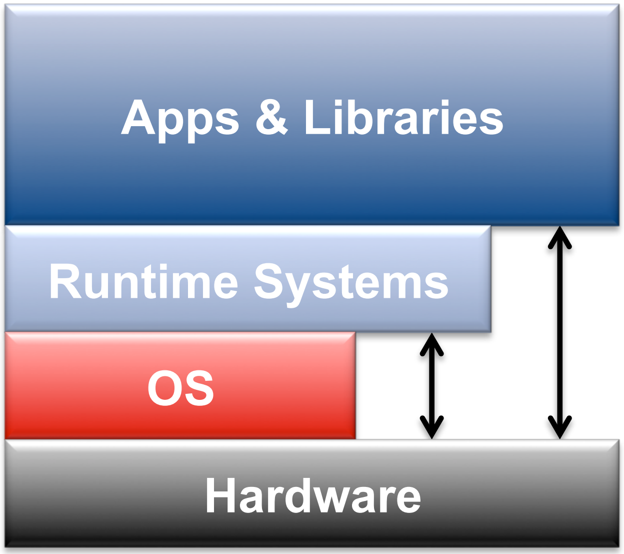
- Should this really be "Hardware Performance Variability"?
  - Equal work doesn't take equal time
  - True today, expect to get worse
- Different types of variability
  - Classic "OS Noise" – probabilistic nature, affects BSP apps
  - Manufacturing variability – fairly static, some parts better than others
  - Thermal throttling – based on environmental factors
  - Contention for shared resources – unpredictable if free for all access
  - Runtime-induced variability – non-determistic scheduling



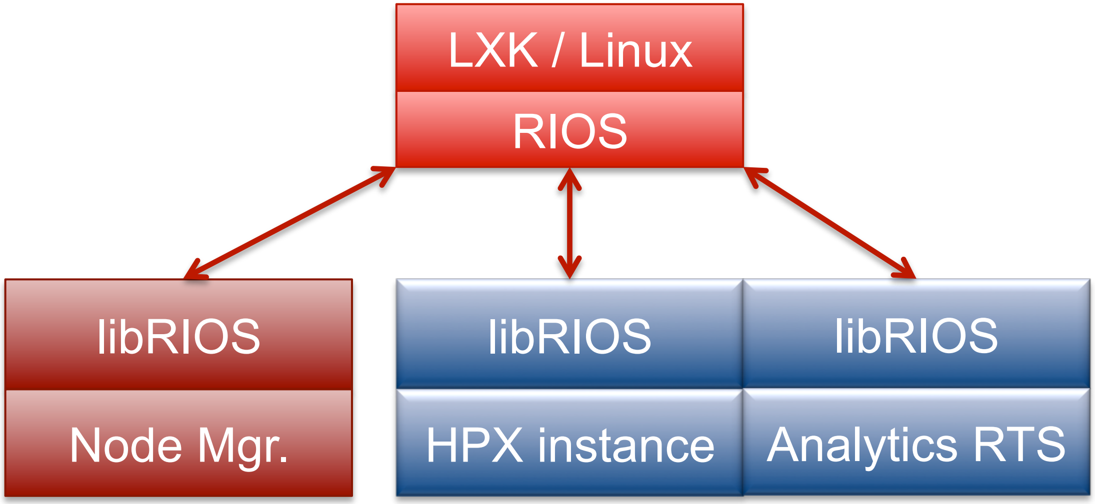


Sys Arch: OS and Runtime Relationship

- Compute node OS kernel
  - Gates access to privileged hardware
  - Provide two-way linkage between higher-level "OS" and local runtime(s) instances
    - Here's your new power budget
    - I could use more power if you have it
  - Final enforcer if runtime doesn't obey
  - Resource negotiation and coordination



Compute Node System Software Stack, OS Bypass

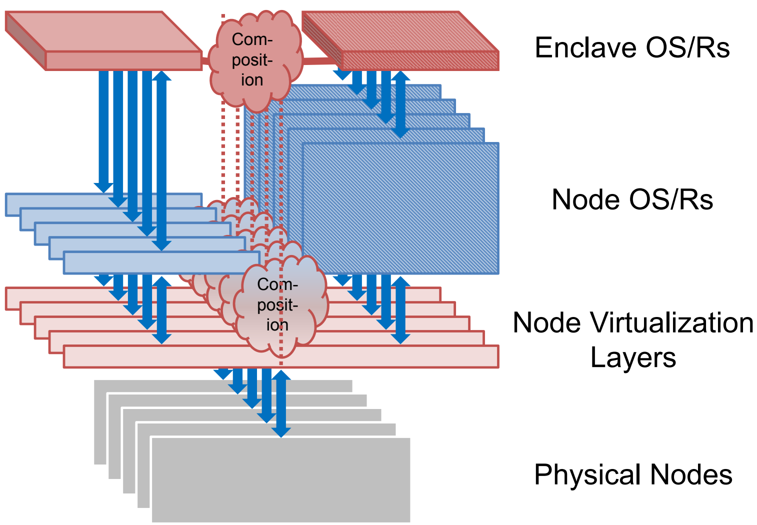


RIOS = Runtime Interface to Operating System

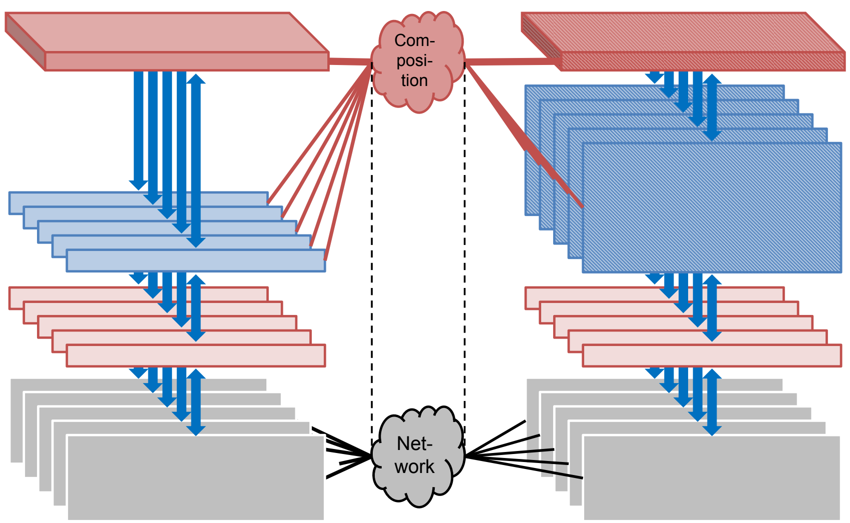
Funded by ASCR X-stack, XPRESS Project

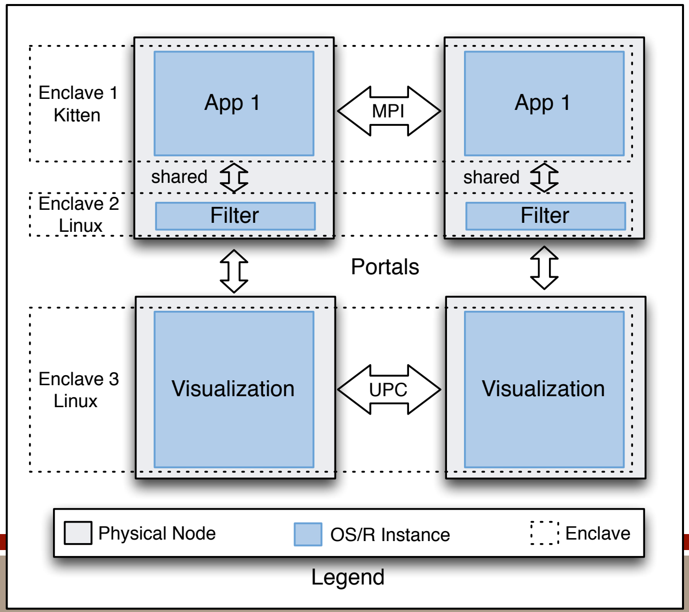
Hobbes: Composition Examples (ASCR)

Intra-node Composition



Inter-node Composition





Example Use Cases:

- Coupling CTH + Paraview/Catalyst on same node
  - CTH has few OS/R requirements
  - Paraview/Catalyst has some “full-OS” dependencies
  - Like previous in-transit case, but co-located like in-situ
- Coupling high fidelity simulation and low fidelity model
  - Useful for combustion and fusion examples
  - Tight coupling or loose coupling, elastic enclaves
- CASL multiphysics coupling, massive collisions
- LAMMPS and SmartPointer Analysis Pipeline
- Goldrush-style cycle stealing for analysis

Evaluation, Things that are Important

- Testing at scale
- Evaluating real applications
- Interoperability / Composability
- Stability of performance / run to run repeatability
  - Error bars are important
  - Compare runs in dedicated mode vs. production
- Ability to tolerate hardware performance variability
  - Run on mixture of slow and fast nodes
  - Test static configuration and dynamically changing configurations

Acknowledgements

- Janine Bennett
- David Bernholdt
- Ron Brightwell
- Doug Doerfler
- Ryan Grant
- Sue Kelly
- Brian Kocoloski
- Jack Lange
- Jim Laros
- Ron Oldfield
- Stephen Olivier
- Dylan Stark