

Hobbes: OS and Runtime Support for Application Composition

Ron Brightwell
Coordinating PI



Sandia
National
Laboratories

*Exceptional
service
in the
national
interest*

OSR PI Meeting
May 23, 2016



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.

Hobbes Team

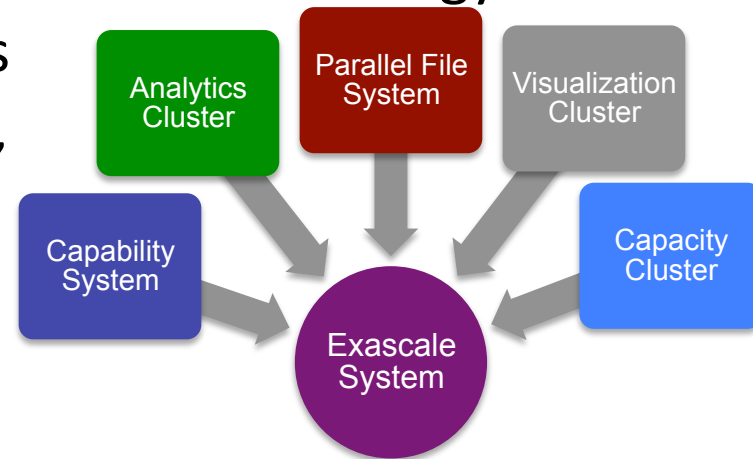
Institution	Person	Role
Georgia Institute of Technology	Ada Gavrilovska	PI
Indiana University	Thomas Sterling	PI
Los Alamos National Lab	Mike Lang	PI
Lawrence Berkeley National Lab	Costin Iancu	PI
North Carolina State University	Frank Mueller	PI
Northwestern University	Peter Dinda	PI
Oak Ridge National Laboratory	David Bernholdt	PI
Oak Ridge National Laboratory	Arthur B. Maccabe	Chief Scientist
Sandia National Laboratories	Ron Brightwell	Coordinating PI
University of Arizona	David Lowenthal	PI
University of California – Berkeley	Eric Brewer	PI
University of New Mexico	Patrick Bridges	PI
University of Pittsburgh	Jack Lange	PI

Project Goals

- Deliver prototype OS/R environment for R&D in extreme-scale scientific computing
- Focus on application composition as a fundamental driver
 - Develop necessary OS/R interfaces and system services required to support resource isolation and sharing
 - Support complex simulation and analysis workflows
- Provide a lightweight OS/R environment with flexibility to build custom runtimes
 - Compose applications from a collection of enclaves
- Leverage Kitten lightweight kernel and Palacios lightweight virtual machine monitor
 - Enable high-risk high-impact research in virtualization, energy/power, scheduling, and resilience

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



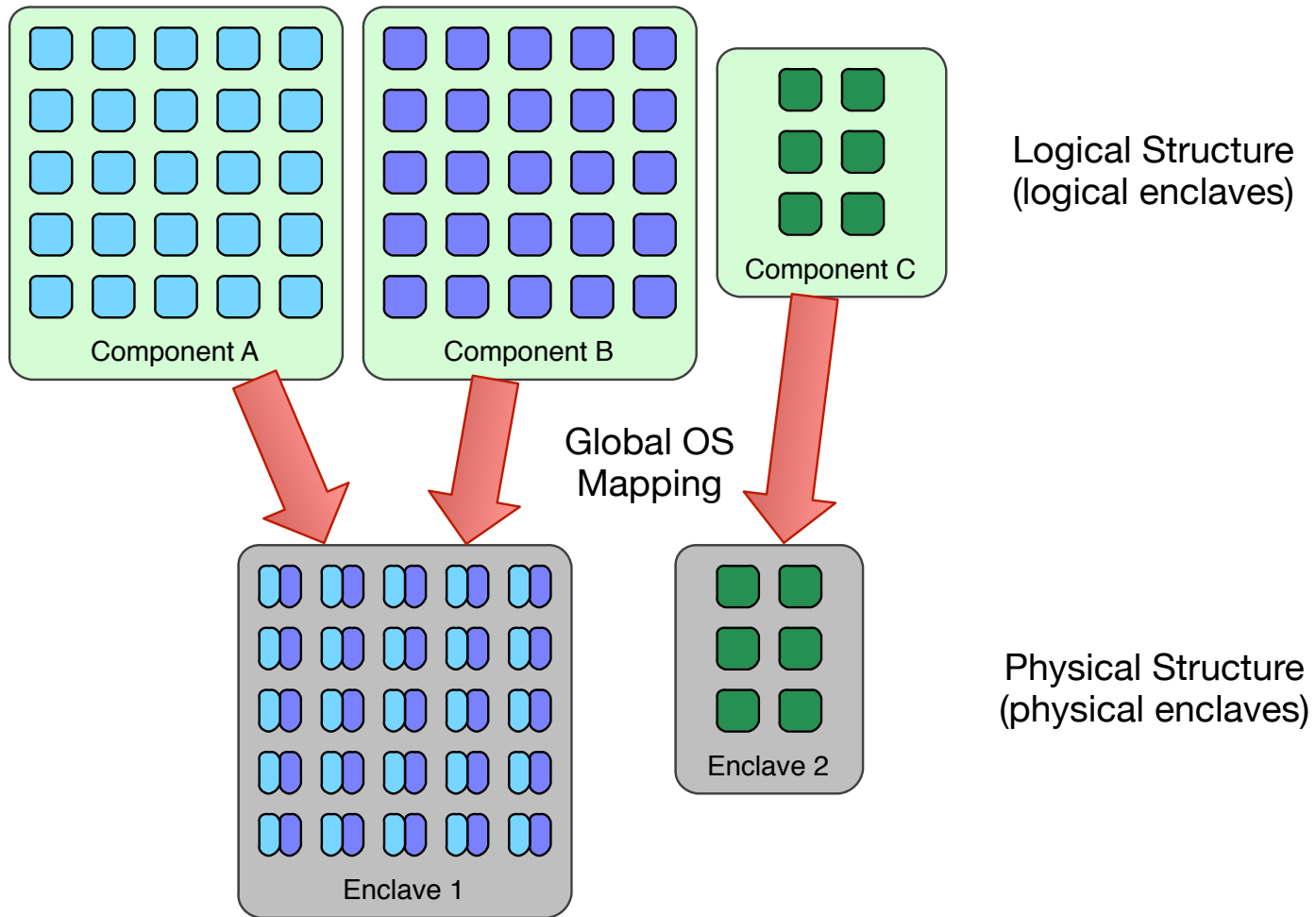
Applications and Usage Models are Diverging

- 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

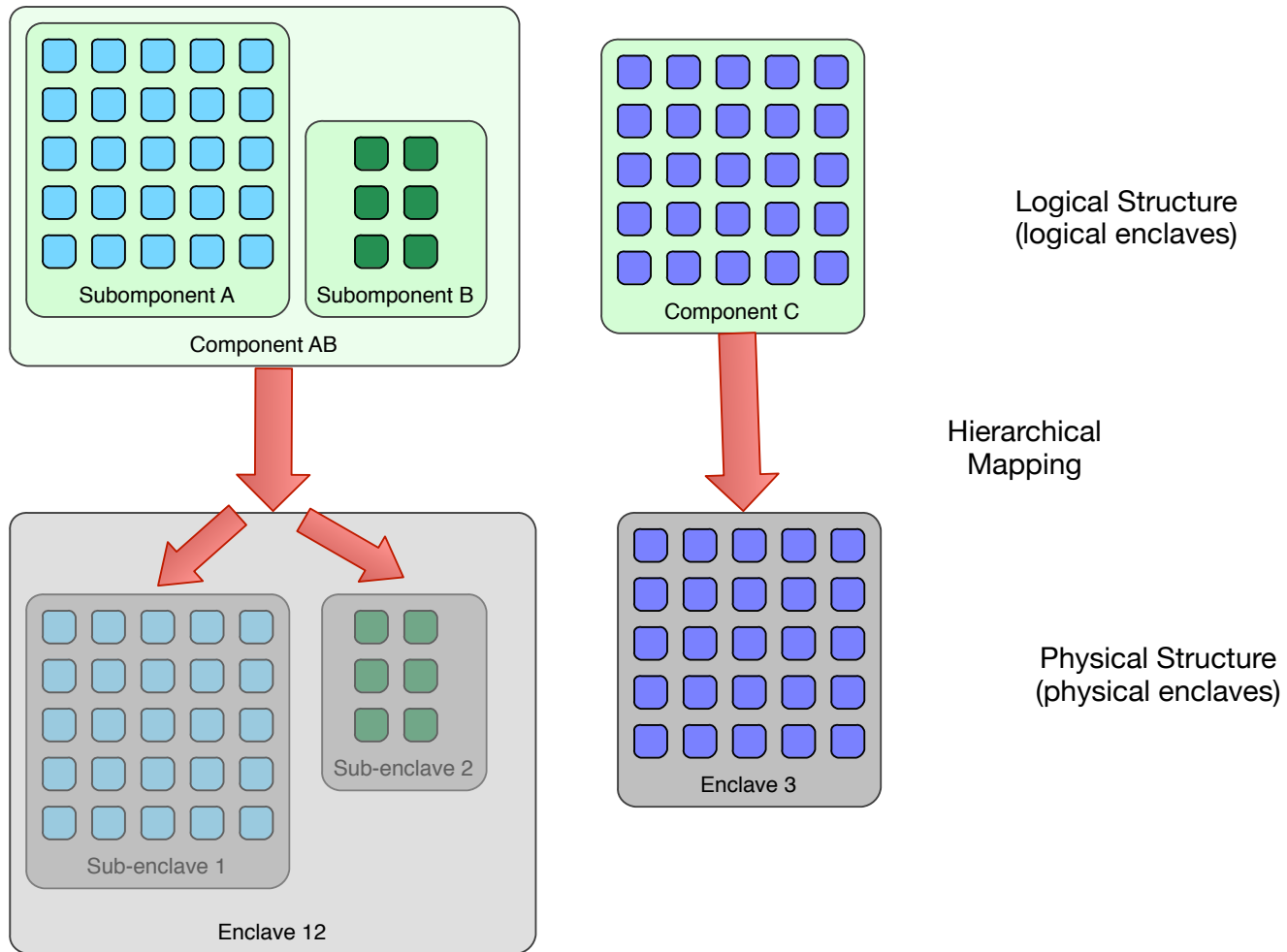
ENCLAVE COMPOSITION



Composition in Hobbes

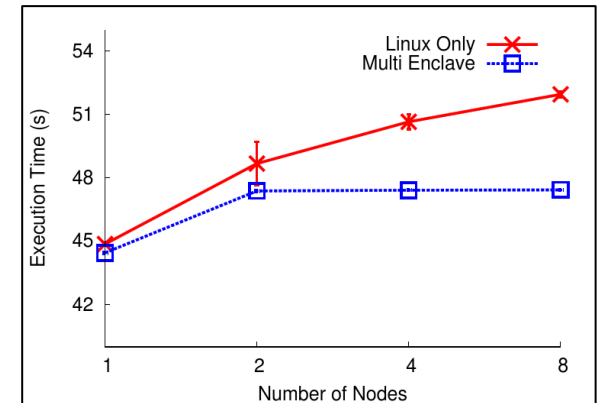


Composition in Argo



Enabling Multi-OS/R Stack Application Composition

- Problem
 - HPC applications evolving to more compositional approach, overall application is a composition of coupled simulation, analysis, and tool components
 - Each component may have different OS/R requirements, no “one-size-fits-all” OS/R stack
- Solution
 - Partition node-level resources into “enclaves”, run different OS/R instance in each enclave
Pisces Co-kernel Architecture: <http://www.prognosticlab.org/pisces/>
 - Provide tools for creating and managing enclaves, launching applications into enclaves
Leviathan Node Manager: <http://www.prognosticlab.org/leviathan/>
 - Provide mechanisms for cross-enclave application composition and synchronization
XEMEM Shared Memory: <http://www.prognosticlab.org/xemem/>
- Recent results
 - Demonstrated Multi-OS/R approach provides excellent performance isolation; better than native performance possible
 - Demonstrated drop in compatibility with both commodity and Cray Linux environments
- Impact
 - Application components with differing OS/R requirements can be composed together efficiently within a compute node, minimizing off-node data movement
 - Compatible with unmodified vendor provided OS/R environments, simplifies deployment



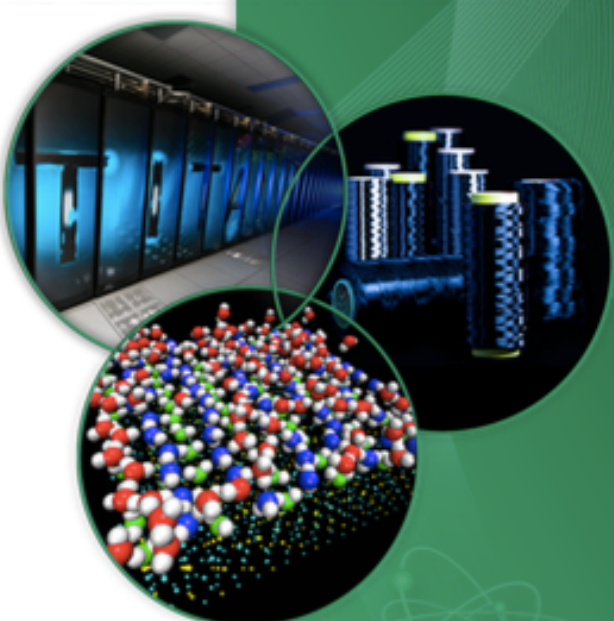
In-situ Simulation + Analytics composition in single Linux OS vs. Multiple Enclaves

Multiphysics Simulation Composition Demo


**Application
Composition with
Hobbes Enclaves**


Composition through process-level enclaves

Geoffroy Vallee (ORNL)
Thomas Naughton (ORNL)
Stuart Slattery (ORNL)
Damien Lebrun-Grandie (ORNL)
David Bernholdt (ORNL)



The image features three circular inset images. The top-left inset shows a long row of server racks in a data center. The top-right inset shows a close-up of several cylindrical battery cells. The bottom inset shows a 3D molecular simulation with atoms represented by red, white, and blue spheres.

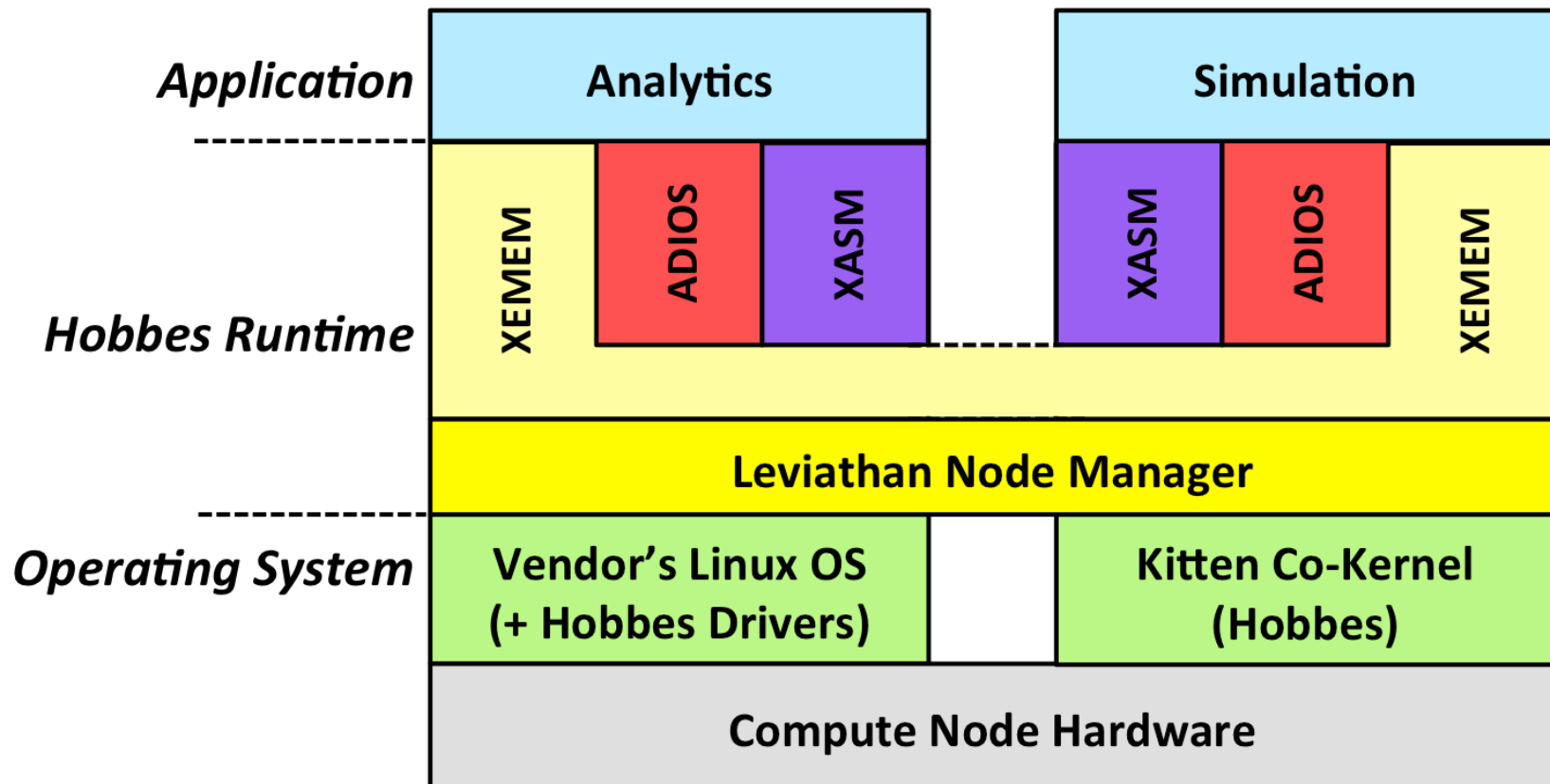
 U.S. DEPARTMENT OF ENERGY

 OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT BATTELLE FOR THE U.S. DEPARTMENT OF ENERGY

NODE VIRTUALIZATION LAYER

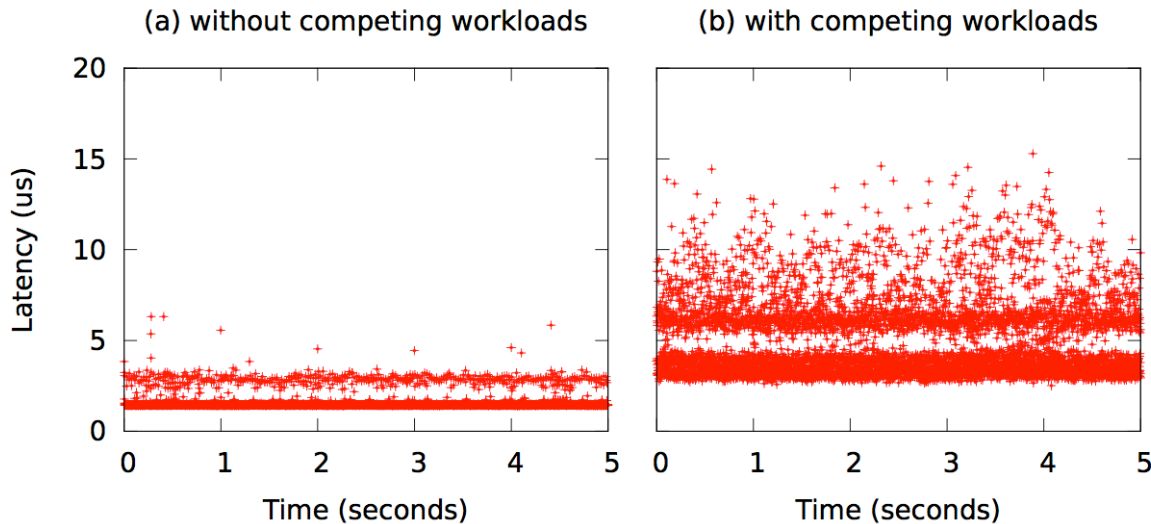


Node Virtualization Layer Architecture

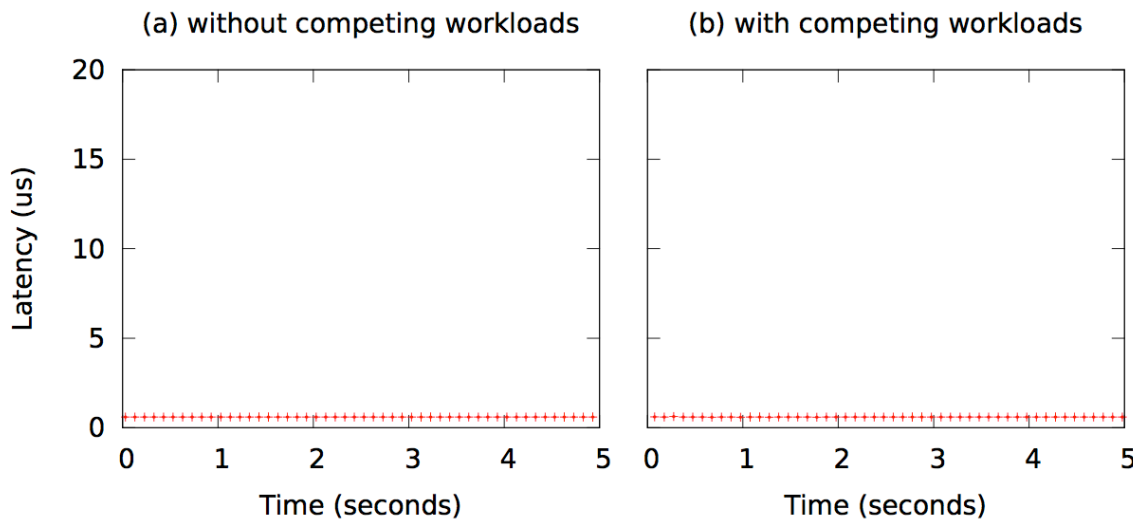


NVL Provides Excellent Inter-Enclave Isolation

Native Linux, Single Linux Image



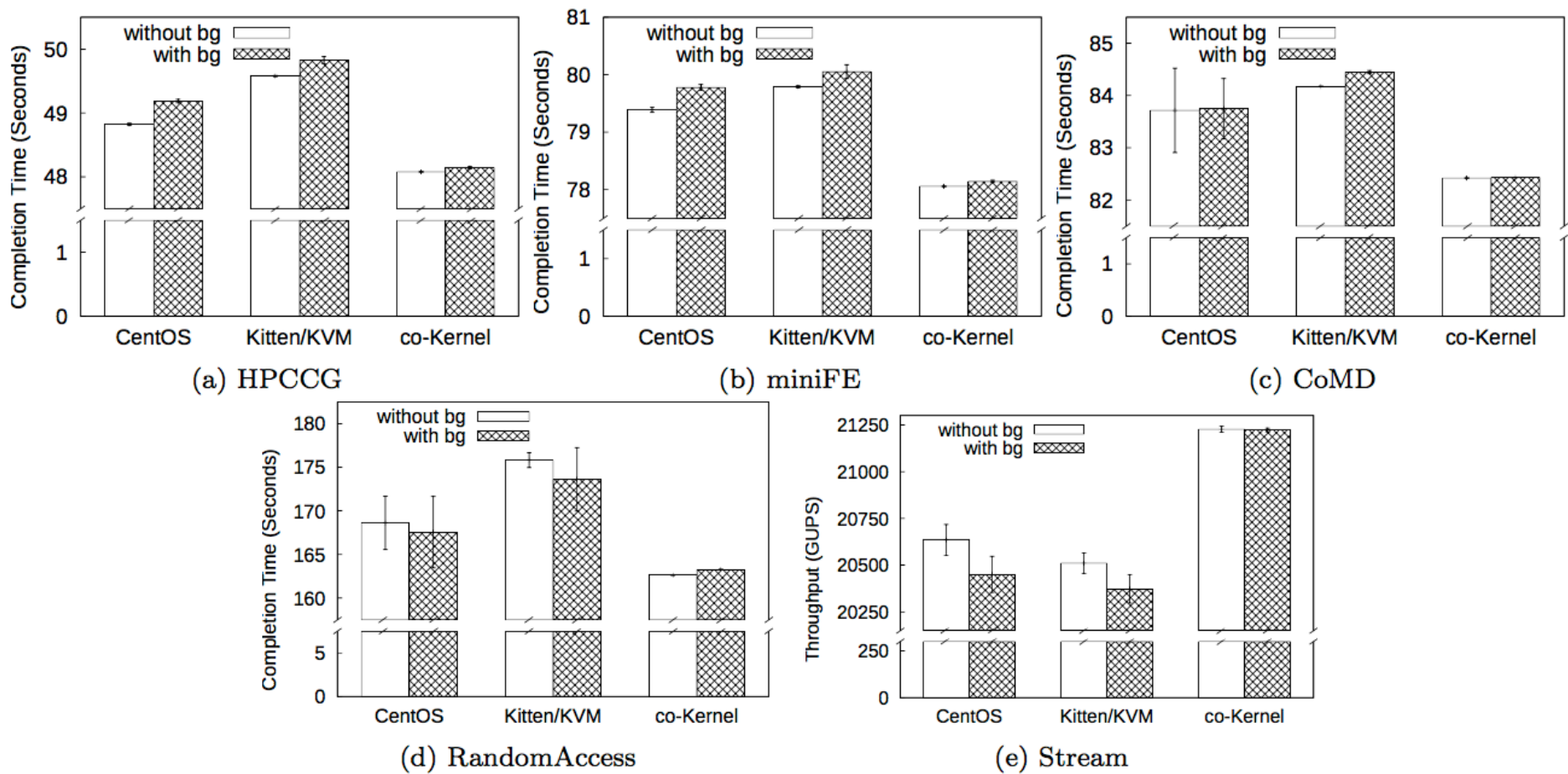
Kitten Enclave with Same Competing Workload



- Co-kernel Architecture nearly eliminates OS-induced interference
- When using single Linux OS (top), competing workload induces noise on other processes, even when they are pinned to disjoint cores and memory
- Isolating processes in a separate Kitten enclave (bottom) eliminates this interference

Excellent Isolation of NVL/Co-Kernel Arch Leads to Increased Performance and Reduced Run-to-run Variability

Comparison of Mini-app/Benchmark Performance With and Without a Competing Background Workload (Kernel Compile)

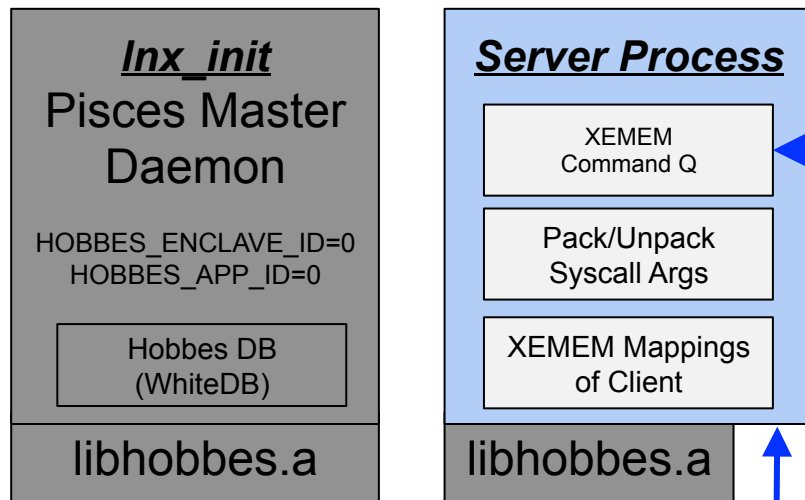


Hobbes NVL Networking Support

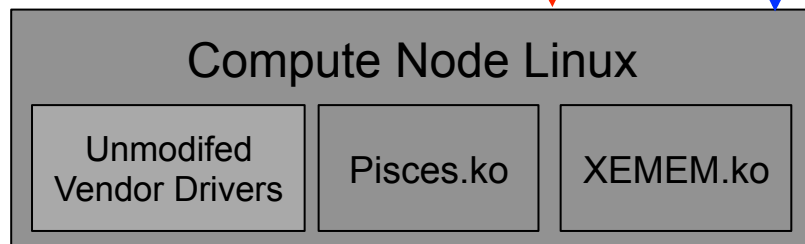
- Problem
 - Single network interface per compute node
 - Must be shared by multiple virtual machines / enclaves
 - HPC NICs usually not designed to be easily virtualized
- Approach
 - Rely on vendor-supplied driver stack running in host (boot) OS/R
 - Rely on HPC NICs to use OS bypass for fast-path operations
 - Use libhobbes.a infrastructure to proxy network setup commands
 - Proxy network-related system using XEMEM command queues
 - Map Kitten memory to Host Linux using XEMEM 1-to-1 mappings
- Issues
 - Some user-kernel interfaces rely heavily on IOCTL's, must know semantics
 - Some NICs require system calls in fast path, hurts proxied performance

Hobbes NVL Proxy Architecture

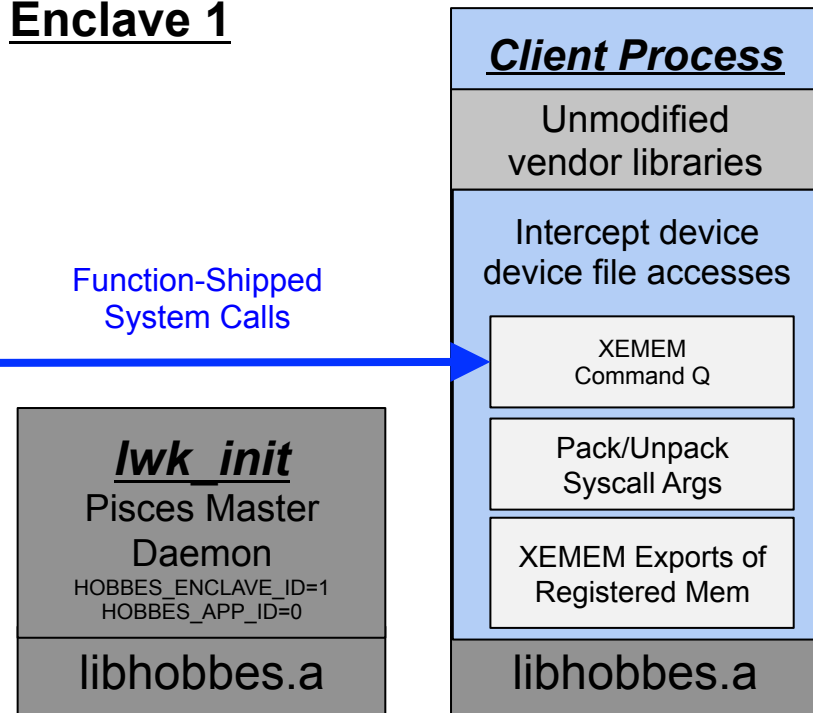
Enclave 0



`xemem_create()`
`xemem_attach()` Re-issued device file accesses



Enclave 1



Function-Shipped System Calls

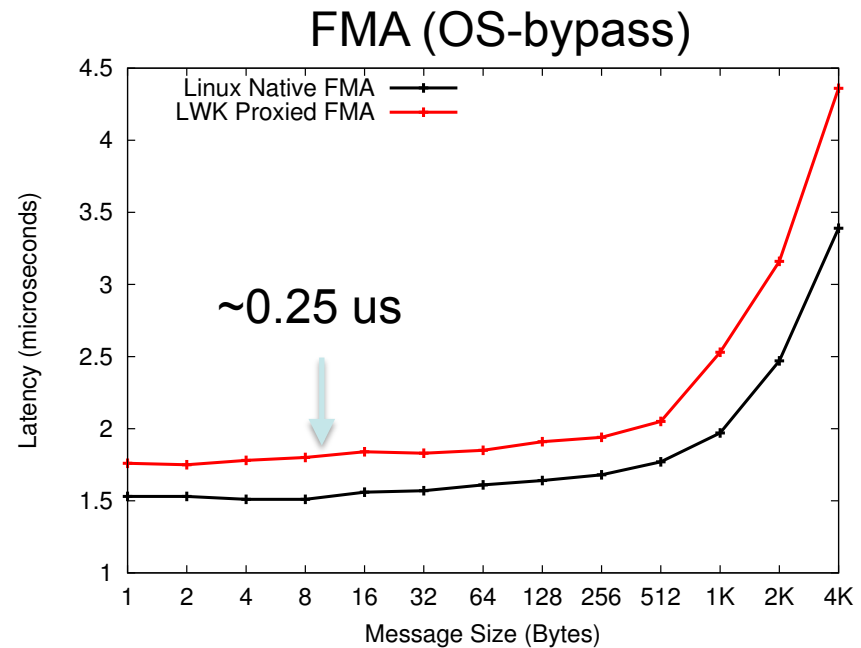
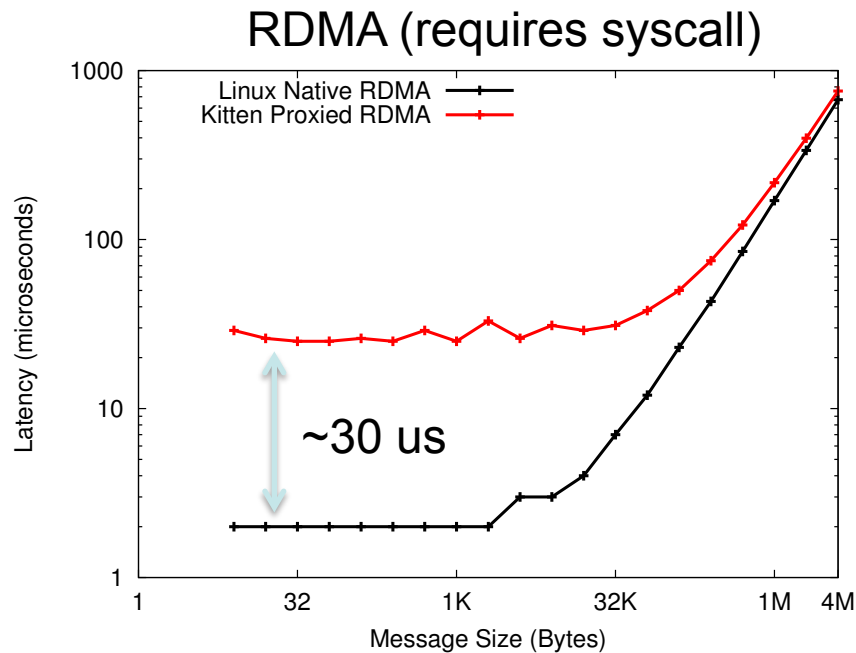
`xemem_create()`
`xemem_attach()`



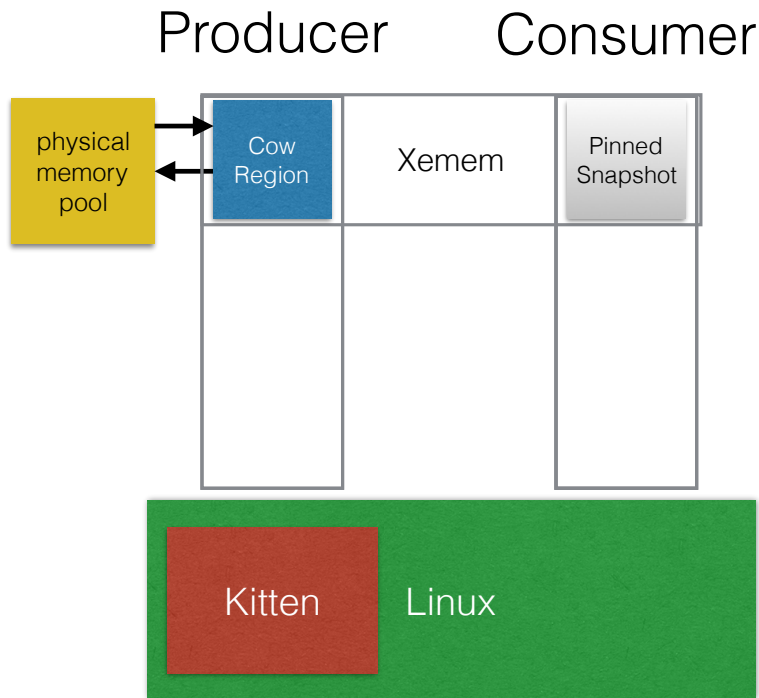
Hobbes NVL Cray XE6 Virtualization

- Gemini network > 6 years old, not fully OS-bypass
 - Nonetheless, motivated to support since that is the HW available
 - Also useful platform for developing NVL proxy architecture
- Results mostly as expected
 - High overhead for non-OS bypass network operations (~30 us per call)
 - Little overhead for OS-bypass network operations, likely due to NUMA issues

PingPong Results



Hobbes XASM: Cross-Enclave Asynchronous Shared Memory



- Mechanism for composition
 - Producer exports a memory snapshot
 - Consumer attaches to the snapshot
 - Copy-on-Write used to allow both to continue asynchronously
- Works across enclave boundaries
 - Linux to Linux
 - Linux to Kitten
 - Kitten to Kitten
 - Native—Native, Native—VM, VM—VM
- Built on top of Hobbes infrastructure

SCHEDULING

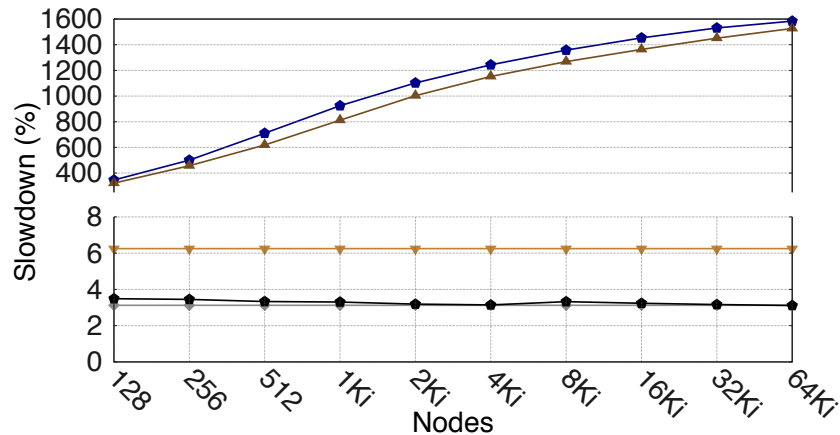


Scheduling Research

- Focus: Resource allocation for HPC application composition
- General Approach
 - Analyzing, quantifying, and controlling application/analytics composition interactions
 - Quantifying ability of modern OS techniques to mitigate negative interactions
 - Examining composition resource allocation issues in modern accelerator-based architectures
- Participating Institutions
 - UNM (Bridges)
 - ORNL (Jones)
 - GT (Schwan, Gavrilovska)
 - Sandia (Ferreira, Widener)
- Recent Highlights
 - Multiple SC submissions on research results
 - Demonstration of cooperative CPU/GPU scheduling
 - Evaluation of Titan time synchronization
 - Probabilistic modeling, empirical simulation of application/analytics interactions
 - Multiple students graduating

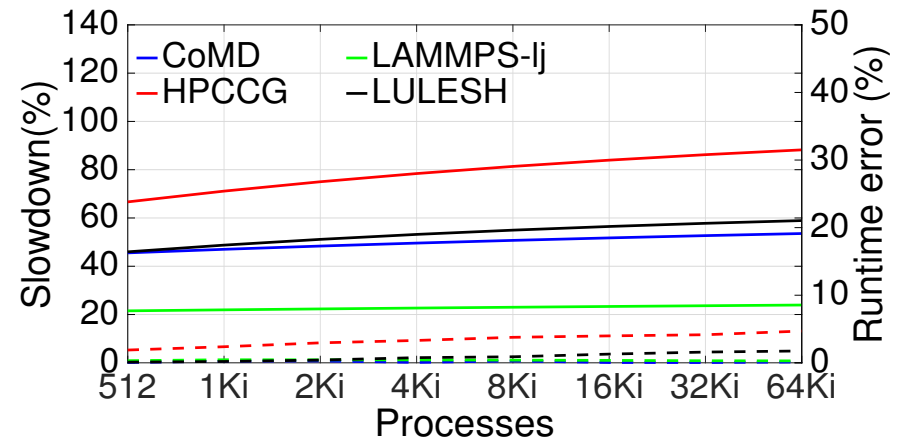
Simulation and Modeling

Application/Analytics Interaction



- Unsynchronized analytics tasks can significantly impact simulation performance

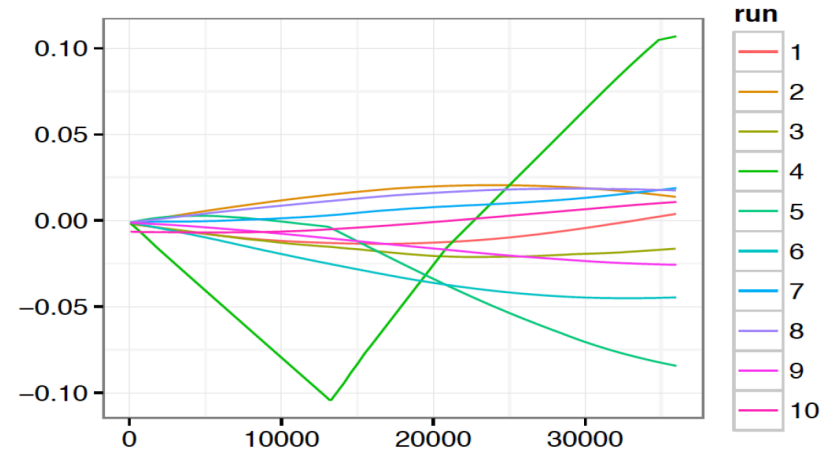
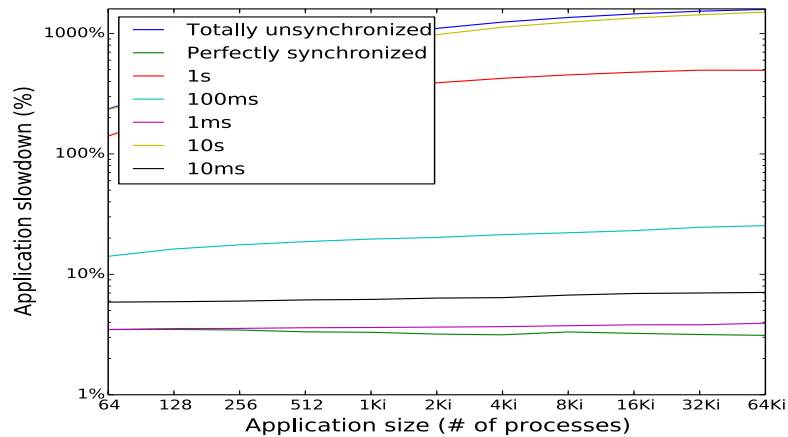
Mondragon et al. CCGrid 2016.



- Extreme value-based model can accurately predict impact of analytics interference on HPC application

Mondragon et al. In submission.

Synchronization-Based Mitigation



- Loose synchronization provided by MPI collective algorithms can mitigate analytics interference

Levy et al. In submission.

- Current NTP clocks on Titan provide only rough basis for synchronizing distributed actions

Jones et al. In preparation.

Scheduling Analytics for Modern Hardware and Runtimes

- Landrush: Coordinated scheduling and allocation of CPU and GPU Resources
 - Co-schedule analytics and simulation on GPUs
 - Reduces performance impact of analytics by 80%
 - Goswami et al. CCGrid 2016, IPDRM 2016.
- Coordinated OS-Runtime Scheduling
 - Integration of OCR with Hobbes Node OS
 - Progress-based scheduling based on runtime-based task queues and phases
 - Experimental evaluations with Cholesky, CoMD, FFT, Smith-Watterman, etc.
 - “HINTS: a progress-based scheduler for co-located runtimes in HPC”, In preparation.

GLOBAL INFORMATION BUS

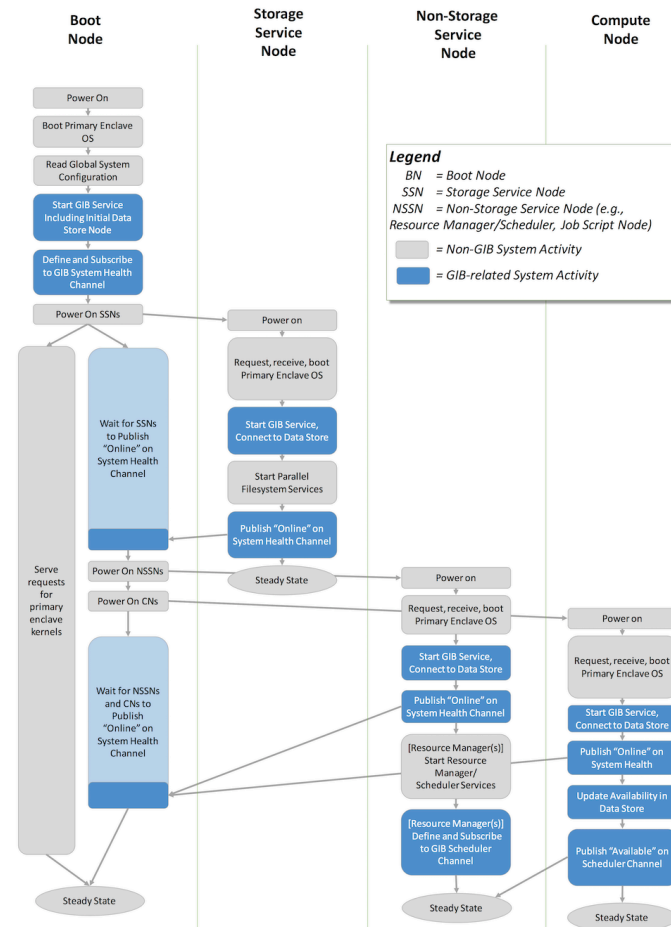


Global Information Bus

- Definition of use cases
 - Draft document in preparation
 - System boot, job launch, job monitoring, response to job termination, system shutdown
 - Useful for determining needed GIB publish/subscribe channels, which data stored in GIB data store
- Continued work on proof of concept implementation
 - In support of API, use case definition and refinement
 - BEACON (old version) for publish/subscribe
 - Riak for distributed data store
 - Integration with Hobbes Leviathan intra-node management and control service

GIB Use Case: System Boot

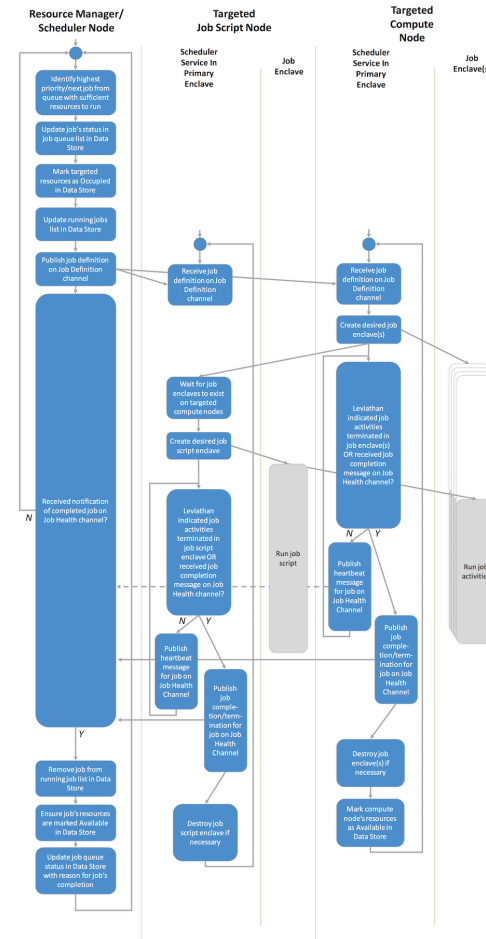
- Proposed boot sequence for establishing GIB data store, system health monitoring publish/subscribe channel, and resource management availability channel
- Potential optimizations if large per-node NVRAM is available
- See poster for more detail



Proposed boot procedure highlighting GIB-related activities

GIB Use Case: Job Management

- GIB involved in:
 - Job launch
 - Job monitoring
 - Response to job completion (graceful or otherwise)
- Assumptions made about job's use of separate enclaves and Leviathan's ability to detect and notify scheduler daemon on job enclave termination/failure
- See poster for more detail



Proposed Job launch/health monitor/termination detection highlighting GIB-related activities

ENERGY



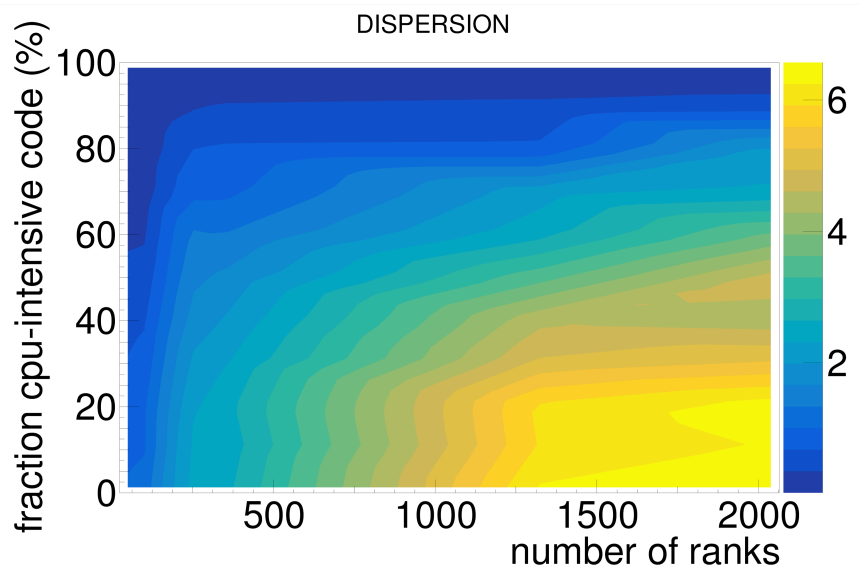
Motivation

- **Energy efficiency is a first order design concern**
Hardware <-> System Software <-> Applications
- **Long history in HPC using DVFS to reduce energy usage without performance loss BUT ...**
 - All techniques are quantitative (predict performance)
- **Recent applications are dynamic, unpredictable**
 - Changes in programming models
 - Changes in hardware

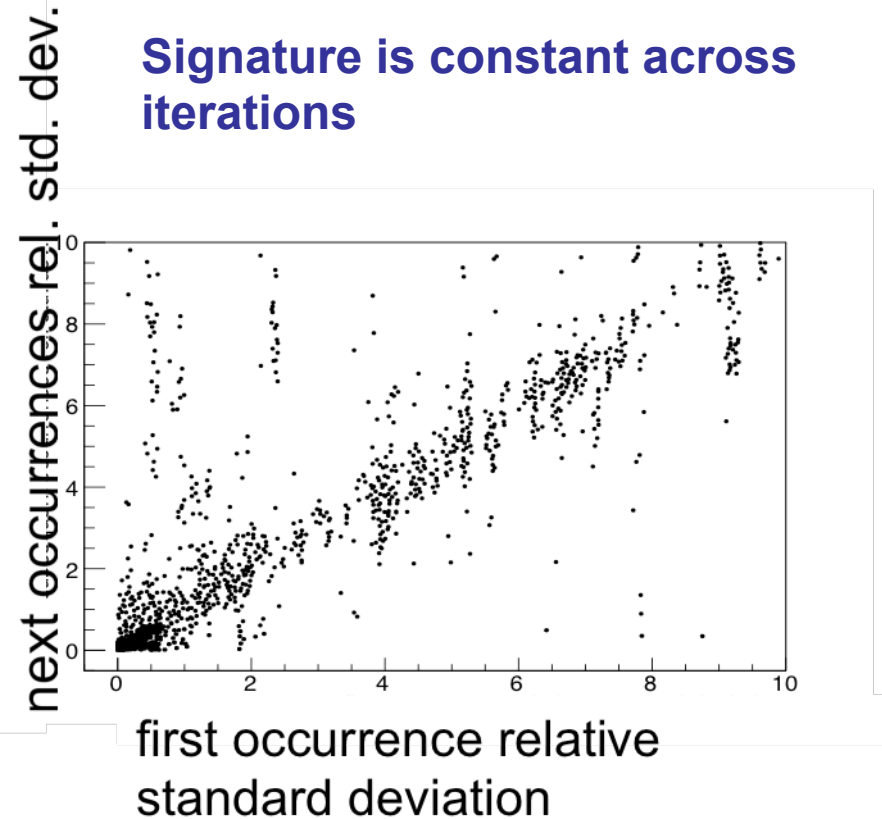
Our Qualitative Approach

Variability is caused by contended resources

Has recognizable signature
Dispersion + Skewness

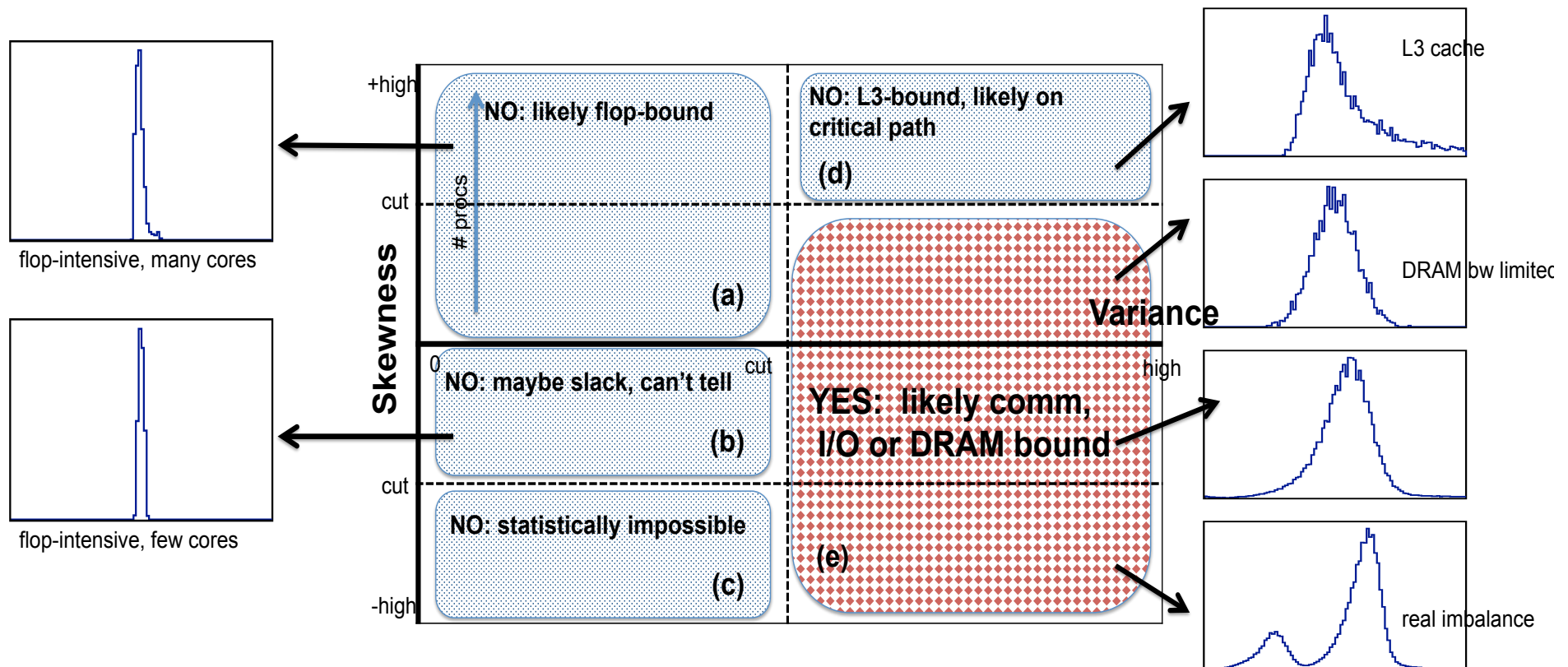


Signature is constant across
iterations



**Can't predict per rank magnitude,
but can predict distribution of timings across ranks**

Predicting DVFS Potential

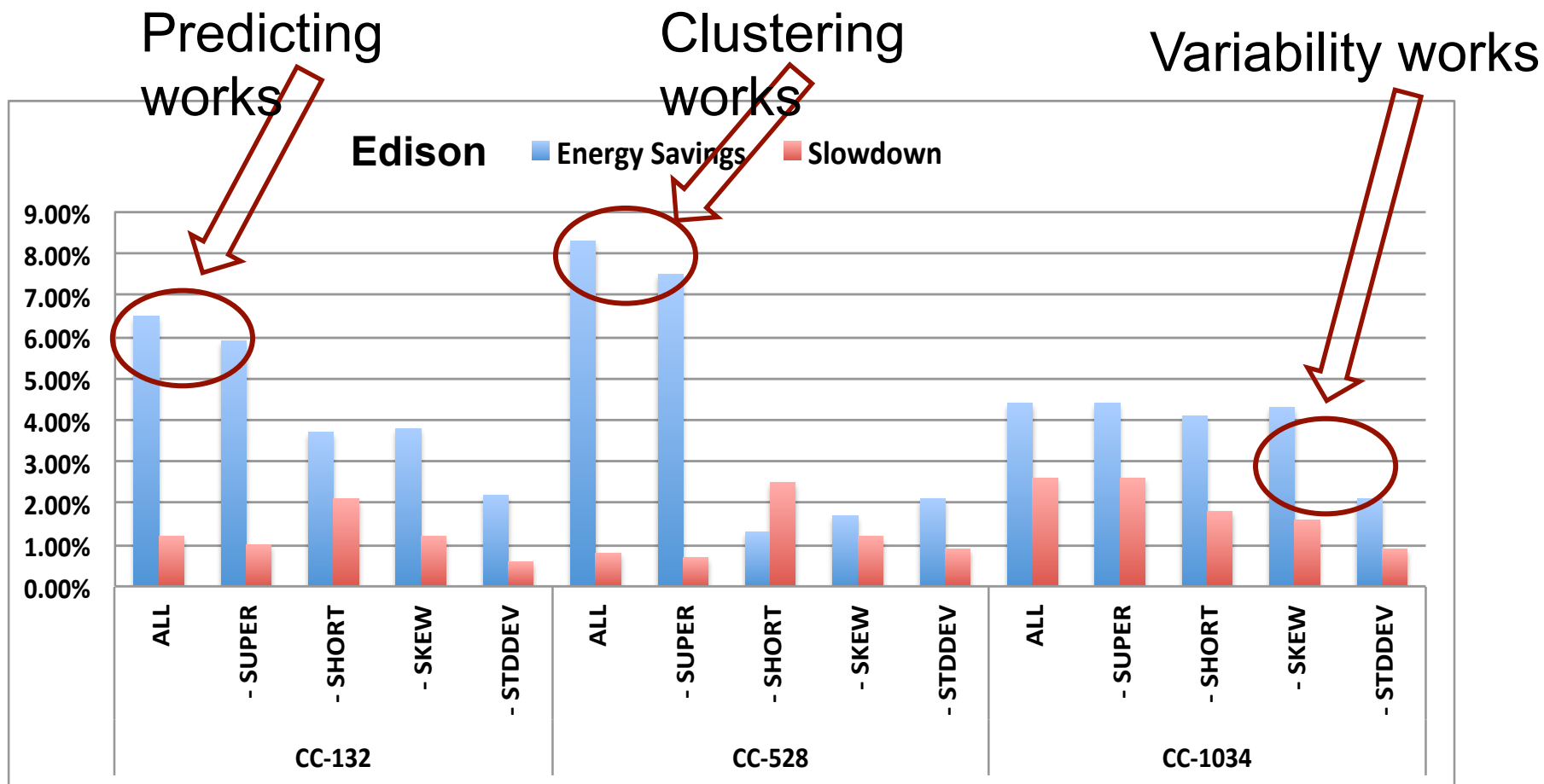


Combination of dispersion and skewness identifies execution on shared constrained resource

DVFS Control Algorithm

- **Programming model/language independent**
 - Context sensitive
 - Online and offline/trace
- **Partition execution into regions (barriers)**
- **Learn by choosing one target GLOBAL frequency, then vote**
- **Region clustering to hide DVFS control latency**
 - Candidates + short + bridge

Some NWChem Results



720 Haswell cores - Shepard

One-sided: 12% energy savings, no slowdown

Two-sided: 20% energy savings, < 1% slowdown

DEBUGGING



Enhanced Debugger for HPX-5 Runtime System

- **Goal**

Create new GDB commands that enable developers to dynamically inspect and modify HPX-5 constructs.

- **Objectives**

- Access and Modify data in **GAS**.
 - hpx print <hpx_address>
- Display and Track currently existing **Parcels**.
 - info parcels
- Investigate the status of **LCOs**.
 - info lco <hpx_address>

- **Status**

GAS	Finished	<input checked="" type="checkbox"/>
Parcels	Finished	<input checked="" type="checkbox"/>
LCOs	In Progress	<input type="checkbox"/>

```
Breakpoint 1, _stencil_action (args=<op
179      hpx_addr_t new_grid_addr = hp
(gdb) p neighbors
$8 = {131952, 131440, 131688, 131704}
(gdb) hpx print 131440 double
$9 = 0.25
(gdb) █
```

```
-----
Scheduler Parcels:
-----
Total number of worker threads: 1
-----
Worker 0 info:
-----
Current parcel address: 0x7f22e575a800
Current action: heat.c:_stencil
Parcel size: 16
Buffer address: 0x7f22e575a850 "\002"
-----
Parcels in Queue 0
-----
Parcel Address      Action
-----
0x7f22e53ff200     heat.c:_spawn_stencil
0x7f22e53ff180     heat.c:_spawn_stencil
0x7f22e53ff100     heat.c:_spawn_stencil
0x7f22e53ff080     heat.c:_spawn_stencil
0x7f22e53ff000     heat.c:_spawn_stencil
0x7f22ef514f80     heat.c:_spawn_stencil
0x7f22ef514f00     heat.c:_spawn_stencil
```

Debugging Support in Hobbes

- **Low-level debugging components**

- Support of *ptrace* system call in Kitten
- Intra-enclave debugger proxy
- Conventional external debugger under user control

- **Compiler techniques**

- Source-to-source transformation
- Based on the idea of Bertrand Meyer's *design by contract* (preconditions, postconditions, and invariants) and specialized error handlers
- Uses *#pragma debug* directives in C/C++ (or specially formatted comments in Fortran) to inject the user-defined conditions

- **Status**

Linux ptrace integration with LWK	In Progress	<input type="checkbox"/>
Process control using external <i>gdb</i>	To do	<input type="checkbox"/>
<i>#pragma debug</i> construct syntax	Finished	<input checked="" type="checkbox"/>
GCC S2S translation plugin	In Progress	<input type="checkbox"/>

DEMOS



Visualizing the OS



COVER FEATURE **OUTLOOK**

Outlook on Operating Systems

Dejan Milošević, Hewlett Packard Labs
Timothy Roscoe, ETH Zurich

Will OSs in 2025 still resemble the UNIX-like consensus of today, or will a very different design achieve widespread adoption?

Seventeen years ago, six renowned technologists attempted to predict the future of OSs for an article in the January 1999 issue of *IEEE Concurrency*.¹ With the benefit of hindsight, the results were decidedly mixed. These six experts correctly predicted the emergence of scale-out architectures, nonuniform memory access (NUMA) performance issues, and the increasing importance of OS security. But they failed to predict the dominance of Linux and open source software, the decline of proprietary UNIX variants, and the success of vertically integrated Mac OS X and iOS.

The reasons to believe that OS design won't change much going forward are well known and rehearsed: requirements for backward compatibility, the UNIX model's historical resilience and adaptability,² and so on. "If it ain't broke, don't fix it."

However, we argue that OSs will change radically. Our motivation for this argument is two-fold. The first has been called the "Innovator's Dilemma," after the book of the same name:³ a variety of interests, both commercial and open source, have invested substantially in current OS structures and view disruption with suspicion. We seek to counterbalance this view. The second is more technical: by following the argument that the OS will change, we can identify the most promising paths for OS research to follow—toward either a radically different model or the evolution of existing systems. In research, it's often better to overshoot (and then figure out what worked) than to undershoot.

Current trends in both computer hardware and application software strongly suggest that OSs will need to be designed differently in the future. Whether this means that Linux, Windows, and the like will be replaced by something else or simply evolve rapidly will be determined by a combination of various technical, business, and social factors beyond the control of OS technologists and researchers. Similarly, the change might come from incumbent vendors and open source communities, or from players in new markets with requirements that aren't satisfied by existing designs.

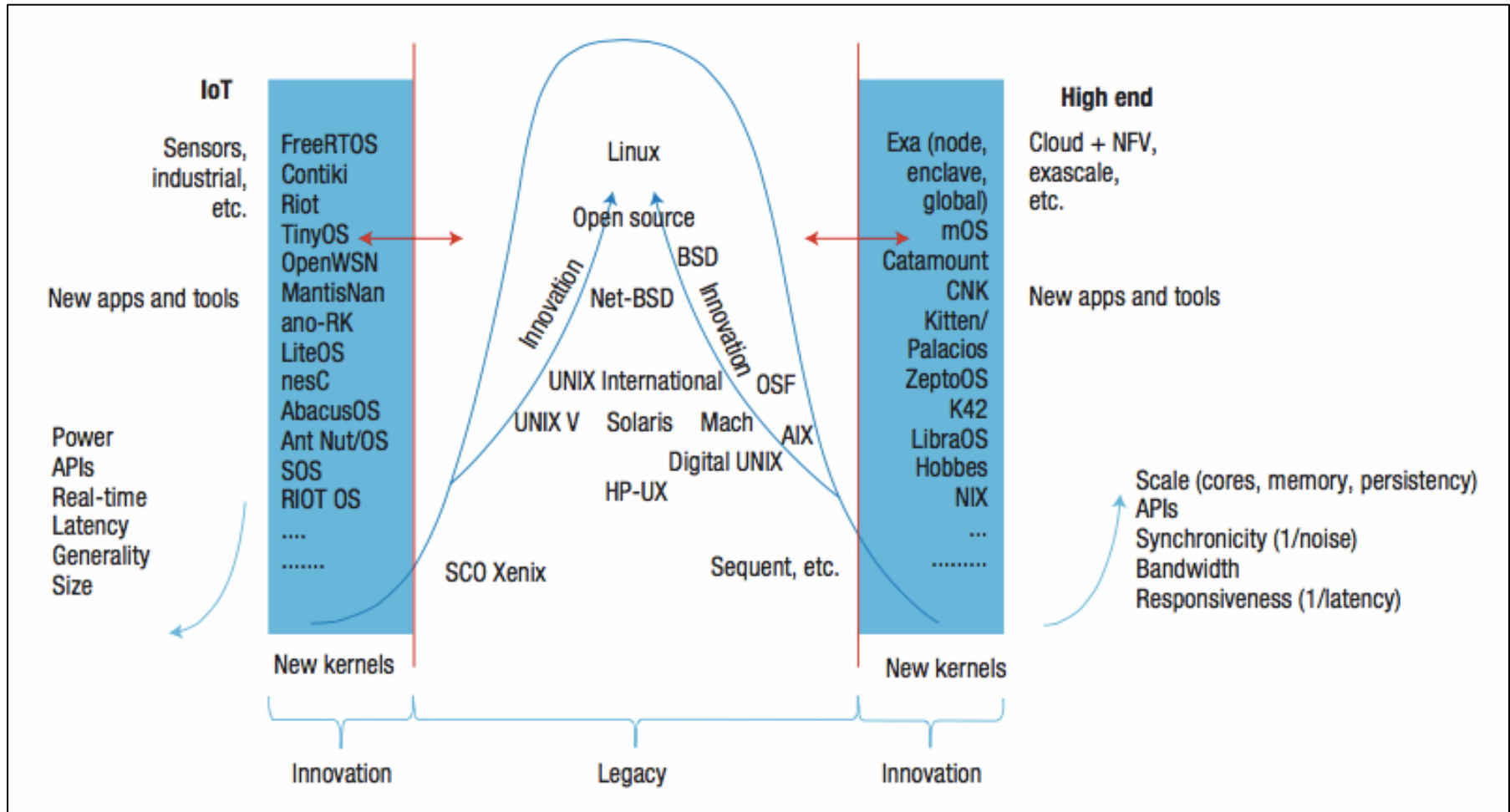
Ultimately, though, things are going to have to change.

HARDWARE TRENDS

Hardware is changing at the levels of individual devices, cores, boards, and complete computer systems, with deep implications for OS design.

32 **COMPUTER** PUBLISHED BY THE IEEE COMPUTER SOCIETY 0018-9219/15/01 0032 © 2016 IEEE

Opportunities for a New OS



Hobbes Demos

- Hobbes Support for Creating and Managing Enclaves
- Demonstration of a Multiphysics Simulation Composition Based on Hobbes Enclave and Data Transfer Toolkit
- Parallel Programming Debugging in Runtime System Environments
- Kitten on KNL Running LULESH on HPX-5

<http://xstack.sandia.gov/hobbes>

