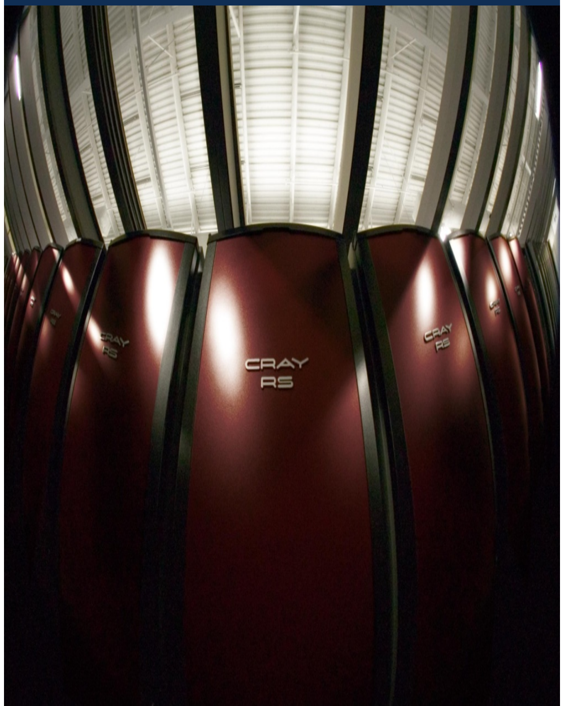
SAND2017-7168PE



Center for Computing Research

Embracing Diversity: OS Support for Integrating High-Performance Computing and Data Analytics

Ron Brightwell and Kevin Pedretti

Scalable System Software Department

July 10, 2017

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

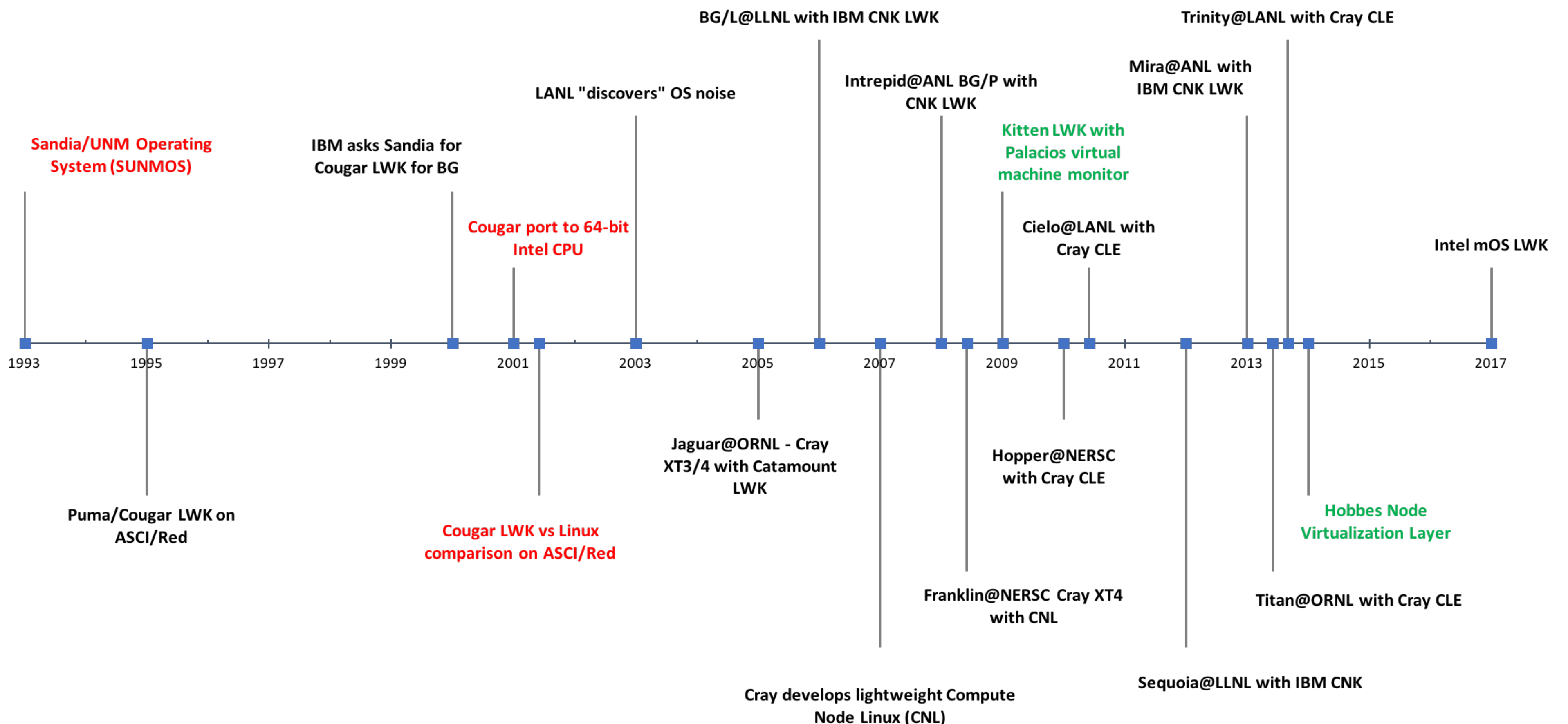
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- Background and Motivation
- Hobbes Node Virtualization Layer (NVL)
- NVL Components
  - Operating Systems: Linux, Kitten, and Palacios
  - Glue: XEMEM, Pisces, Leviathan
  - Composition: ADIOS, XASM, XEMEM
- Hobbes on Cray XC
- Future Directions

Impact of Sandia’s Lightweight Kernel (LWK) R&D

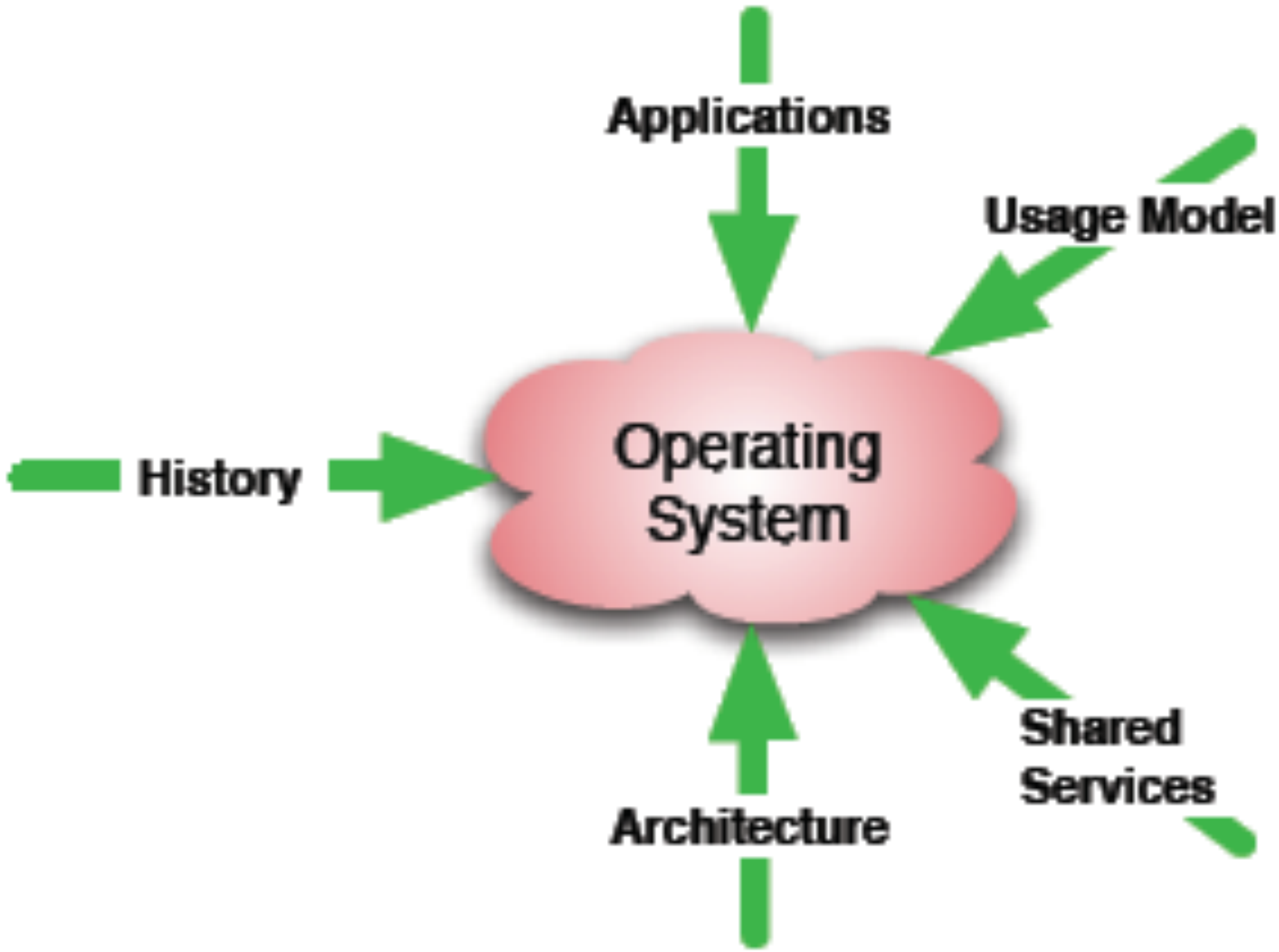
- Sandia is the only DOE lab to partner with vendors to deploy its LWK OS technology in production
  - SUNMOS LWK on Intel Paragon
  - Cougar LWK on Intel ASCI/Red
  - Catamount LWK on Cray Red Storm
- Other vendors have adopted the Sandia LWK model
  - IBM’s Compute Node Kernel for BG/{L,P,Q}
  - Cray’s lightweight Linux Environment (CLE)
- LWK model has been shown to be critical to performance and scalability on distributed memory machines
- Every DOE large-scale HPC machine in the past 25 years has deployed a lightweight OS



DOE/MICS funded activity

DOE/ASCR funded activity

Factors Influencing OS Design



Multiphysics Example

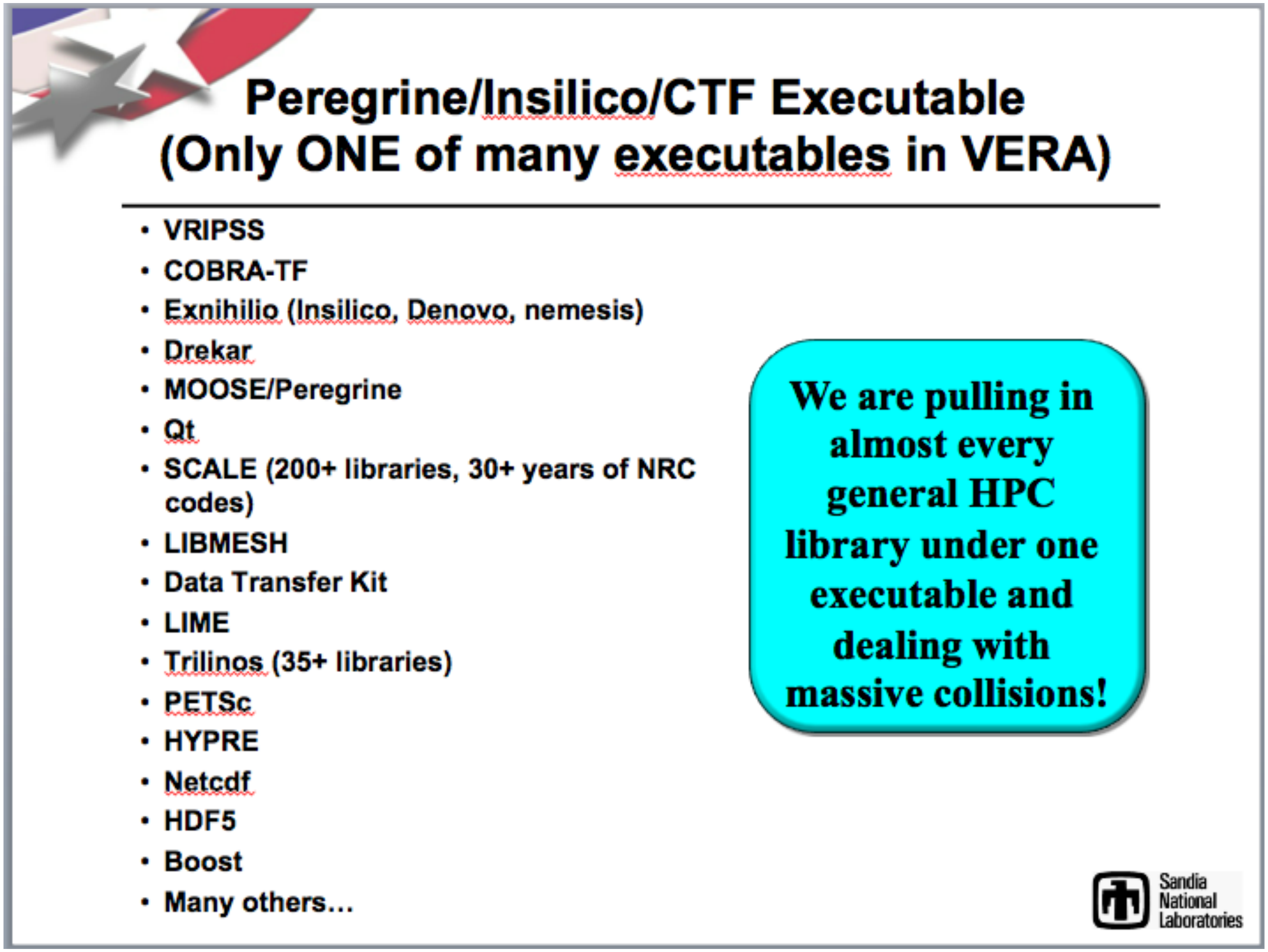
Technical Discussion on CASL: Why is Multiphysics Coupling Difficult?

- The most complex software engineering project I have been involved with
  - Fortran, C, C++, Java, Python, Perl, …
  - 21 git repositories
  - VERA is composed of 350+ software engineering packages, 12 TPLs
- Multiscale physics: Thermal hydraulics (CFD, Subchannel), Neutron transport (SN, MOC), materials models, crack propagation, multiphase boiling, …
- Multiple discretizations and solution algorithms
  - Steady-state, transient (explicit, operator split, implicit), pseudo-transient, continuation, eigensolvers, etc…
  - CVFEM, FE, DGFEM, DAE network models, …
  - Stability and Conservation are critical
- Code use different units, coordinate systems, dimensions, pin axis alignment
- Software engineering quality of individual codes: app → library = disaster!

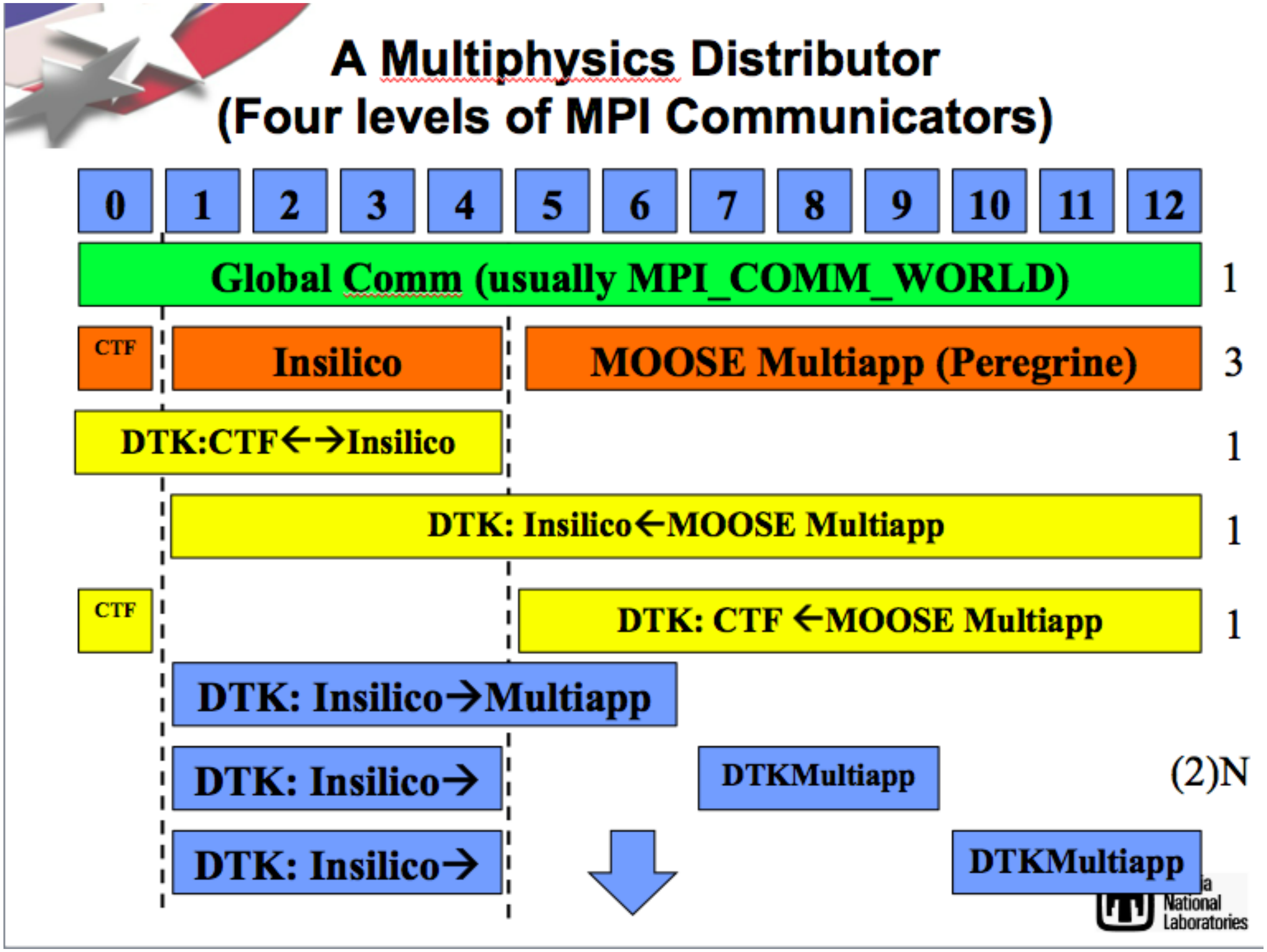
Code integrations require a strong combination of skills in physics simulation, numerical algorithms and software engineering

Sandia National Laboratories

Multiphysics Example (cont’d)



Multiphysics Example (concl'd)



Applications and Usage Models are Diverging

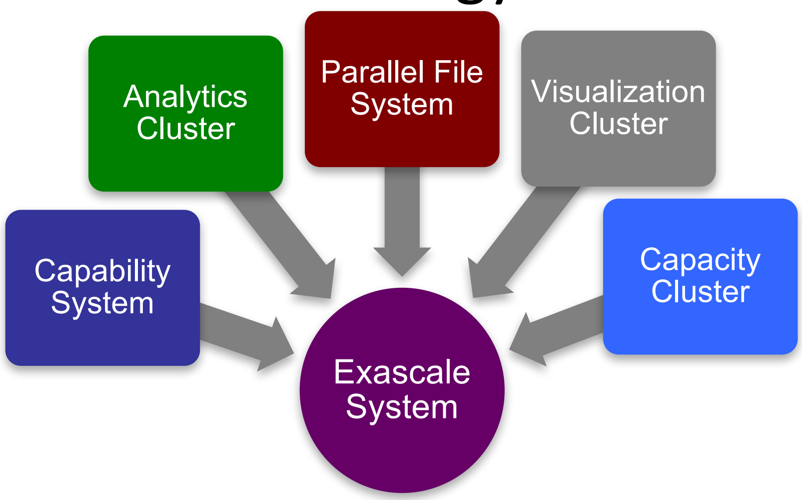
- Application composition becoming more important
  - Ensemble calculations for uncertainty quantification
  - Multi-{material, physics, scale} simulations
  - In-situ analysis and graph analytics
  - Performance and correctness analysis tools
- Applications may be composed of multiple programming models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

Applications Workflows are Evolving

- More compositional approach, where overall application is a composition of coupled simulation, analysis, and tool components
- Each component may have different OS and Runtime (OS/R) requirements, in general there is no "one-size-fits-all" solution
- Co-locating application components can be used to reduce data movement, but may introduce cross component performance interference
  - Need system software infrastructure for application composition
  - Need to maintain performance isolation
  - Need to provide cross-component data sharing capabilities
  - Need to fit into vendor's production system software stack

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



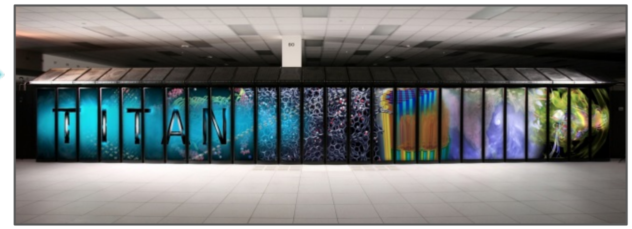
Merging of HPC and data analytics

Future architectures will need to combine HPC and big data analytics into a single box

Apollo: Urika-GD
Graph Analytics

Helios: Urika-XA
BDAS
(Hadoop, Spark)

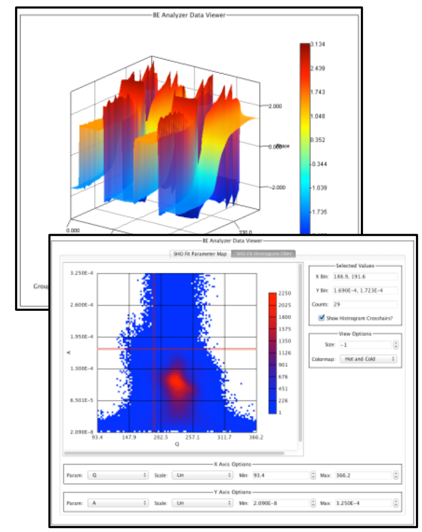
CADES Pods
Compute & Storage



OLCF's Titan
Cray XK7

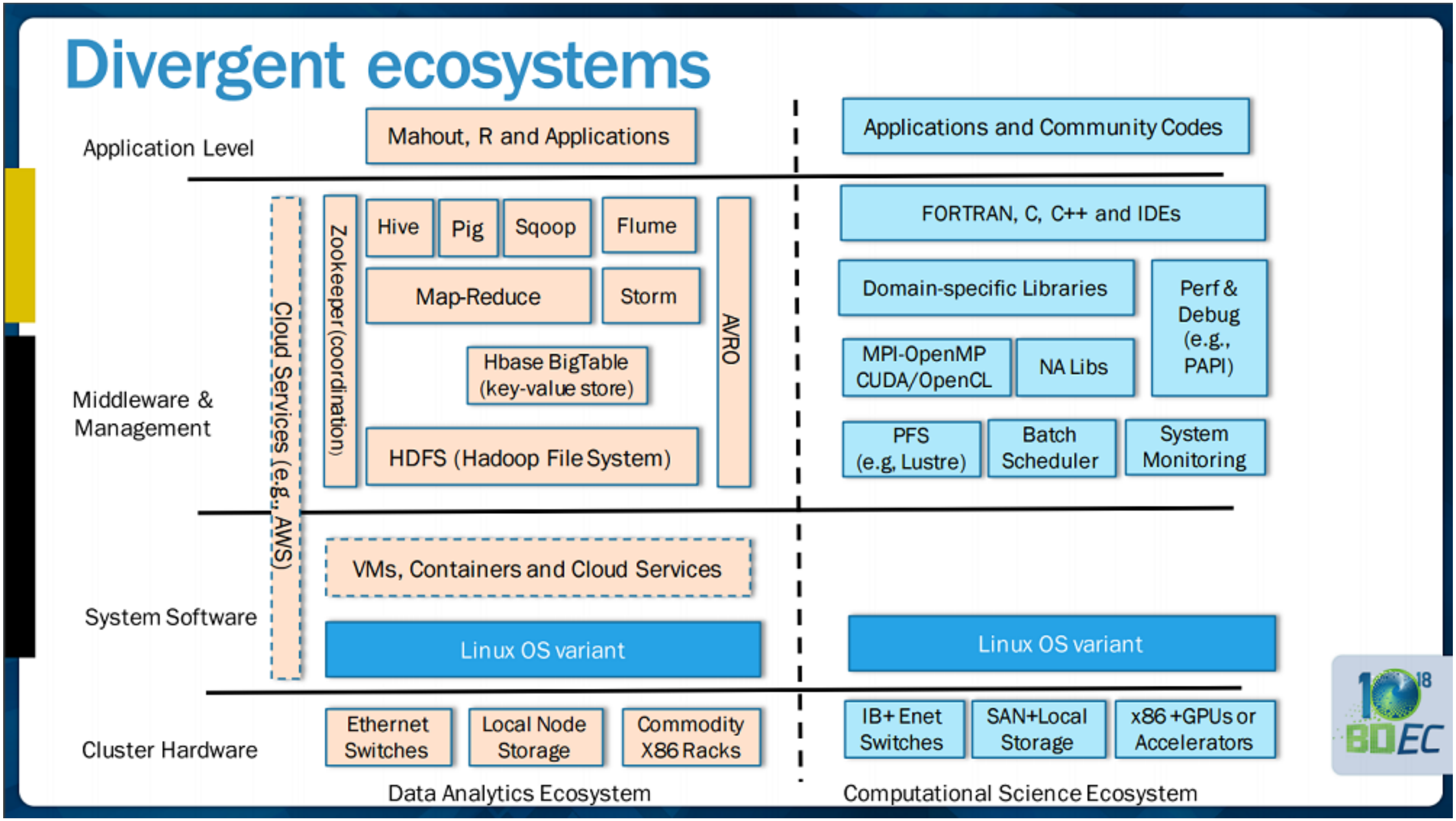
Metis
Cray XK7

beam

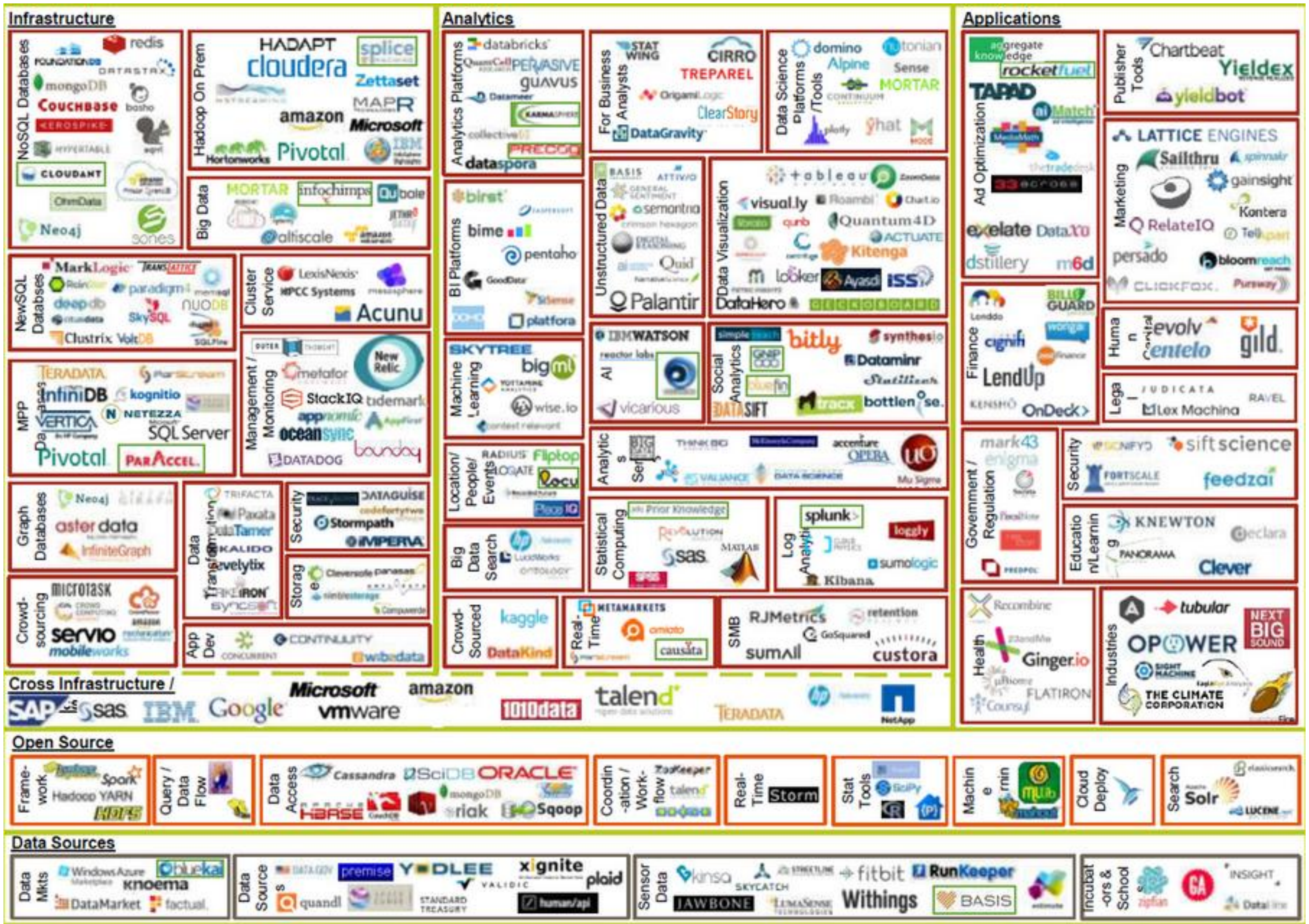


BEAM's "BE Analyzer" tool displaying interactive 2D and 3D views of analyzed multi-dimensional data generated at ORNL's Center for Nanophase Materials Sciences (CNMS)

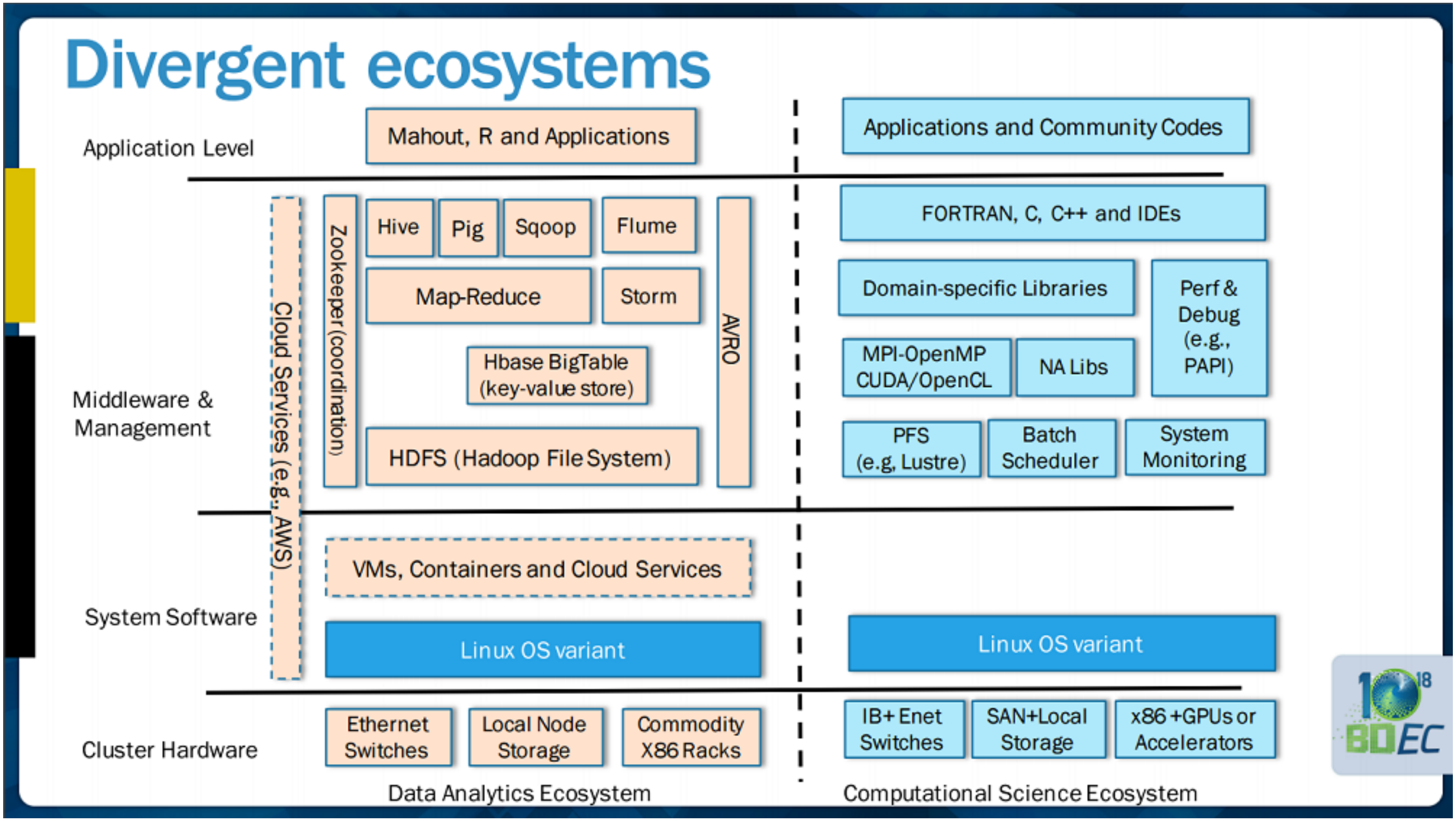
How Do We Bring the Two Worlds of HPC and Big Data Together?



“Big Data” Environment



So, How Do We Bring the Two Worlds of HPC and Big Data Together?



We Don't

We Should Embrace Divergence

- Functional partitioning based on software stack will continue
  - Service nodes, I/O nodes, network nodes, compute nodes, etc.
  - Nodes are becoming too big to be smallest unit of allocation
- Provide infrastructure to manage diverse software stacks
  - Node-level partitioning of resources with different stacks
  - Support for improved resource isolation
  - Mechanisms that provide sharing to reduce data movement
- Enable applications and workflows to define their own software environment

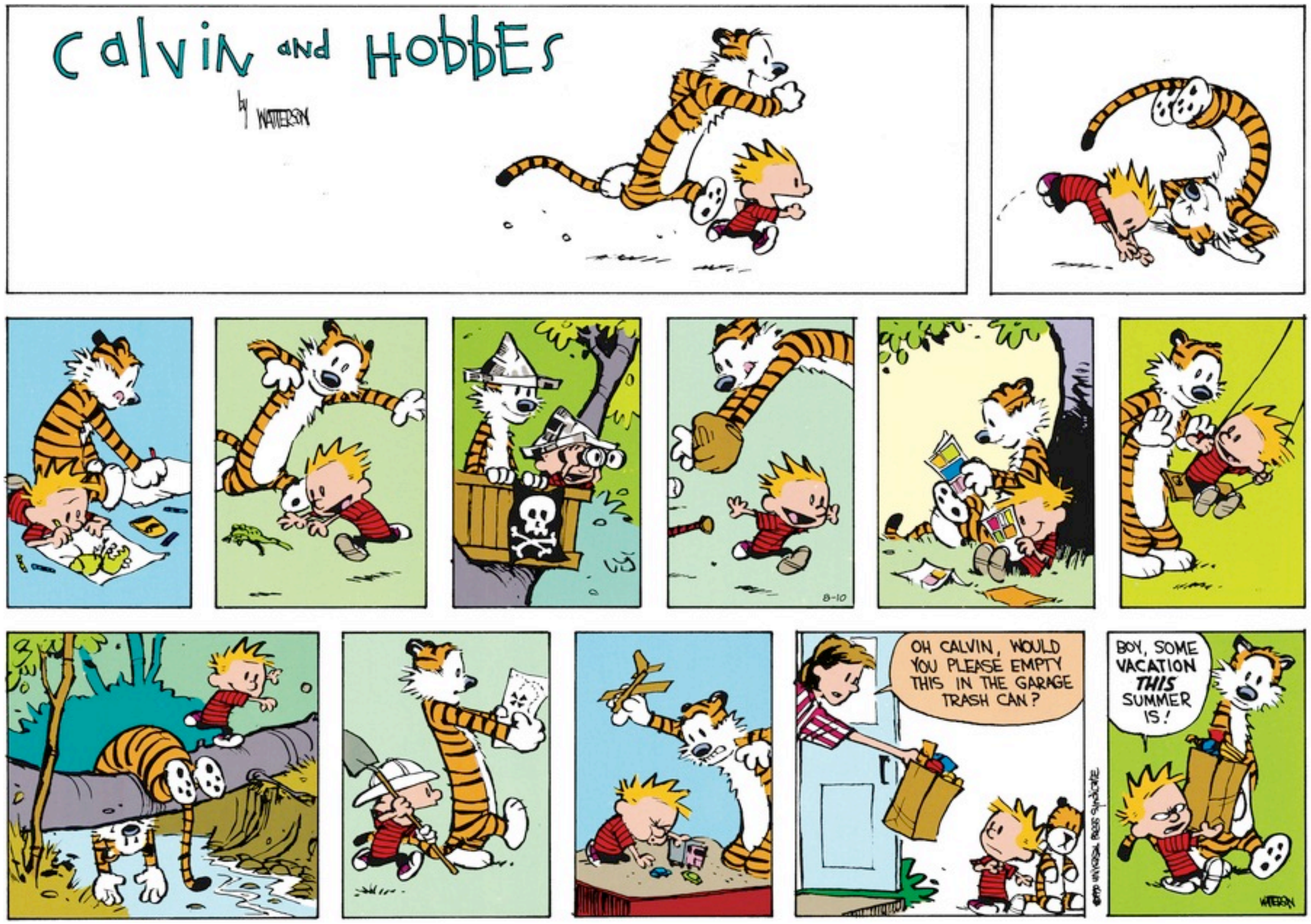
Hobbes Project

- US DOE/ASCR project in OS/R Program started in 2013
- Develop 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
  - Evaluate performance and resource management issues for supporting multiple software stacks simultaneously
  - 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 (partitions)
- Leverage Kitten lightweight kernel and Palacios lightweight virtual machine monitor
- 11 partner institutions – 4 DOE labs, 7 universities

Composition Examples

- SNAP + Analytics
  - "SNAP calculates synonymous and non-synonymous substitution rates based on a set of codon-aligned nucleotide sequences." (HIV related)
  - Proxy app from LANL used for example
- GTC-P + Analytics
  - Fusion simulation testing/proxy app used to test new hardware and algorithm integration into the PIC model. (PPPL)
  - Analytics generate statistics on particles (histograms), sorts, and filters on bounding boxes
- LAMMPS + Analytics
  - Full, production molecular dynamics application from Sandia
  - Analytics look for crack formation by calculating atomic spacing in output data to change simulation from coarse to fine grained.

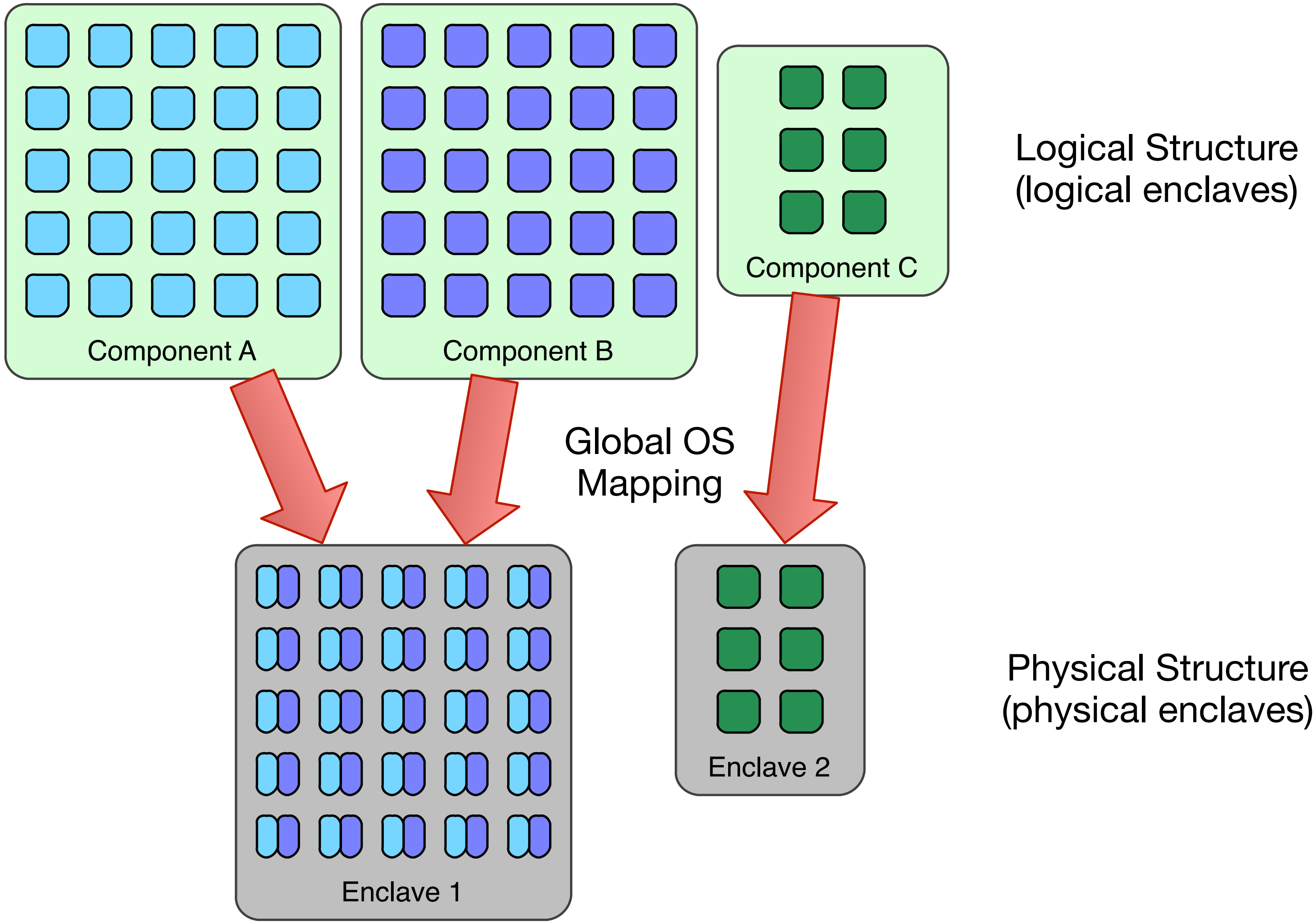
About the Name….



Or Possibly…

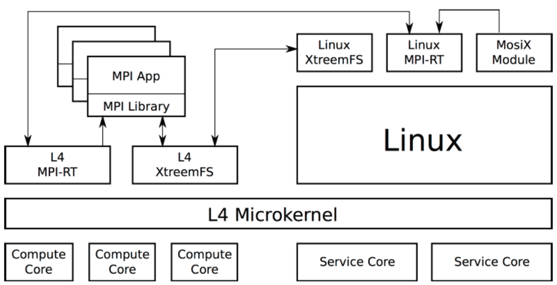
- HPC
- OS
- Building
- Blocks for
- Extreme-scale
- Systems

Application Composition in Hobbes

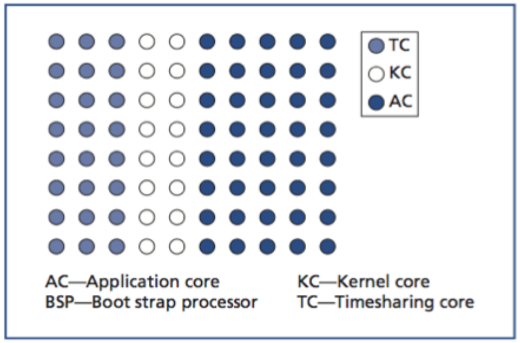


"Combined OS" Approach is Not New

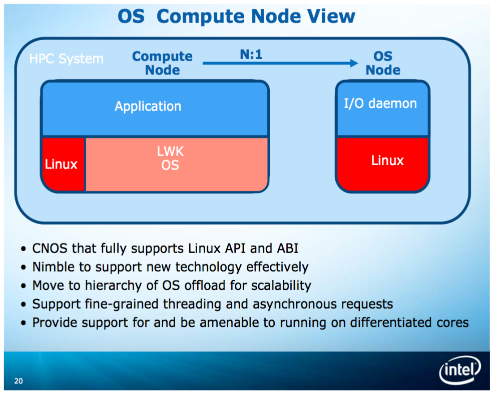
TU Dresden L4Linux (2010)



IBM/Bell Labs NIX (2012)



Intel mOS (2013)



IBM FusedOS (2011)

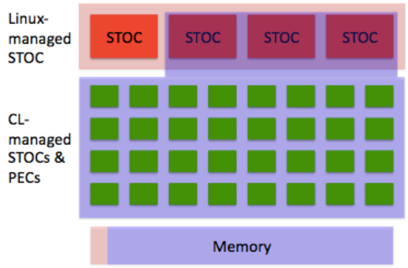


Fig. 3. Partitioning of cores and memory for HPC applications.

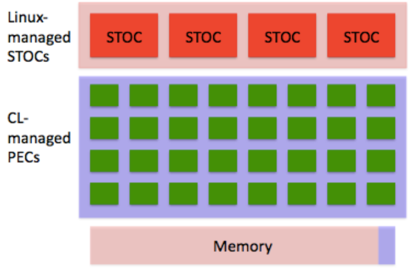
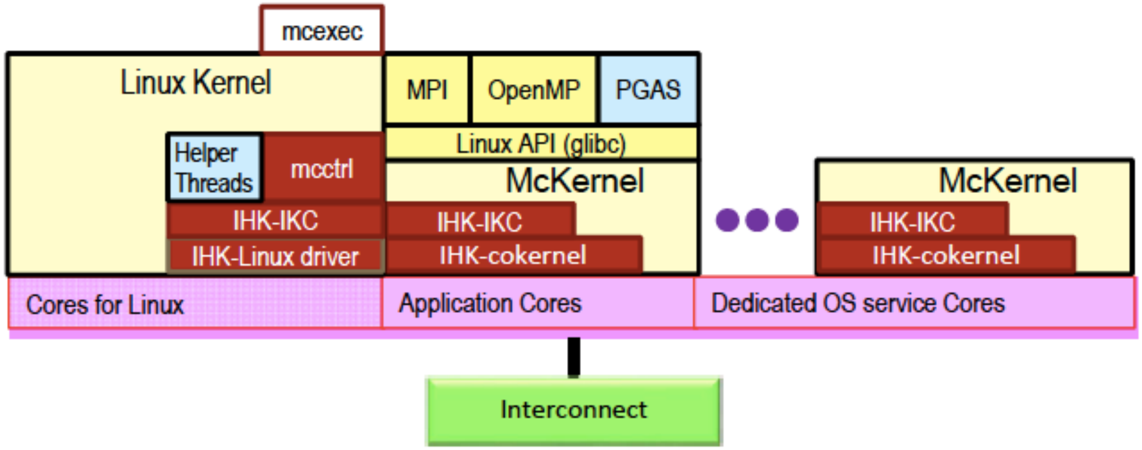


Fig. 4. Partitioning of cores and memory for Linux applications.

MAHOS (2013)



Outline

- Hobbes Node Virtualization Layer (NVL)
- NVL Components
  - Operating Systems: Linux, Kitten, and Palacios
  - Glue: XEMEM, Pisces, Leviathan
  - Composition: ADIOS, XASM, XEMEM
- Hobbes on Cray XC
- Future Direction

Why Specialized Operating Systems in HPC?


Kitten Lightweight Kernel

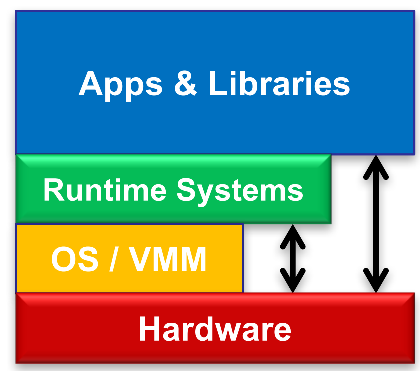
- Lots of new hardware + software challenges to tackle
  - Heterogeneous cores and memory, node-local NVRAM, complex on-chip networks, power management, …
  - Lightweight kernels are a good vehicle for exploring solutions
- Still can't separate OS from architecture
  - BlueGene used embedded cores with weak MMU/TLB -> Linux had issues
  - GPUs don't run an OS, but do have a 20M+ SLOC driver stack + firmware
  - D.E. Shaw Anton, Cray MTA/XMT, … so different it is very hard to run a general purpose OS, need custom system software development
  - New hardware capabilities, like heterogeneous cores and memory, and non-cache-coherent core groups, break traditional OS assumptions
- Ability to do HPC-specific things, without doing battle with Linux "community"
  - Examples: mmunotify, huge pages, OOM killer, page coloring, XPMEM
  - Vendors ship "special sauce" Linux kernel patches, not upstreamable

Why Virtualization in HPC?

Palacios Hypervisor

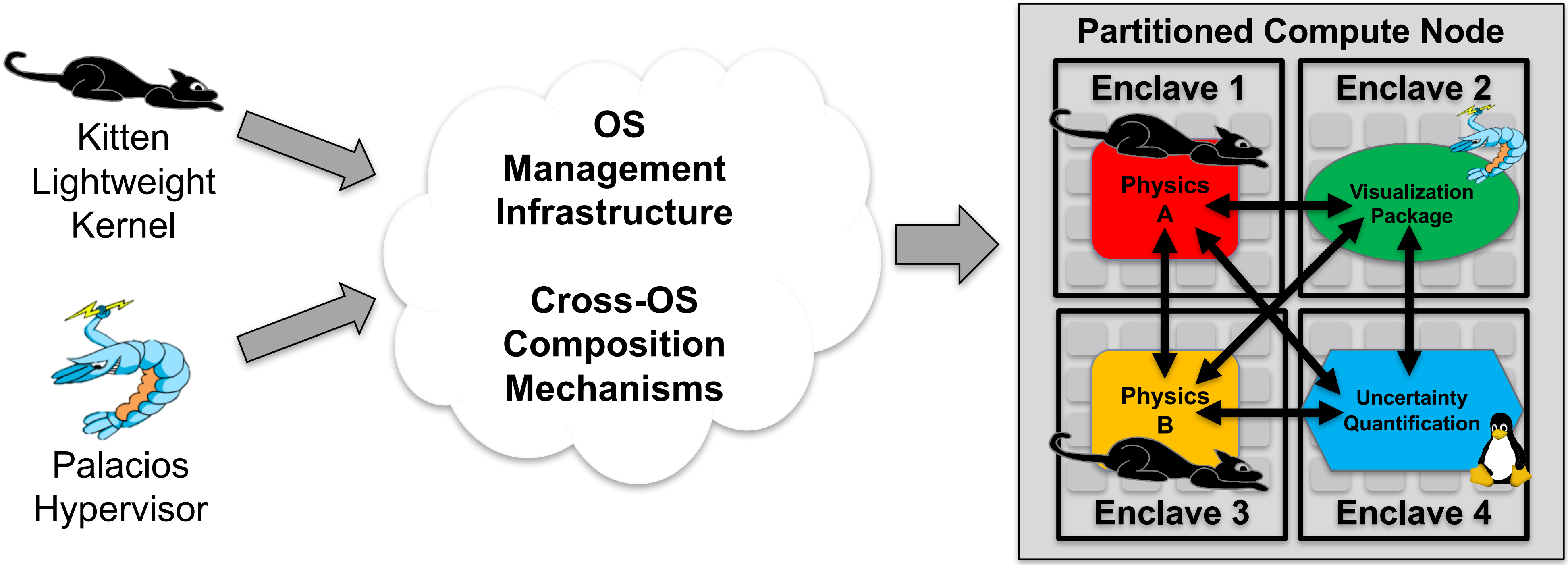
- Support multiple system software stacks in same platform
  - Vendor's stack good for physics simulations, data science difficult
  - Virtualization adds flexibility, deploy custom images on demand
  - Not just user-space containers, need ability to run different OS kernels
    - New Linux kernel versions, replace vendor's old kernel
    - Special-purpose OS/R stacks: mOS, McKernel, Kitten, FFMK/L4, Argo, …
    - Large-scale emulation experiments, networks + systems
  - Leverage industry momentum, student mindshare

- Virtualization overhead can be very low
  - Use hardware support, don't oversubscribe, space share, use large pages, physically contiguous virtual memory
  - Demonstrated < 5% overhead in practice on 4K nodes (VEE'11)



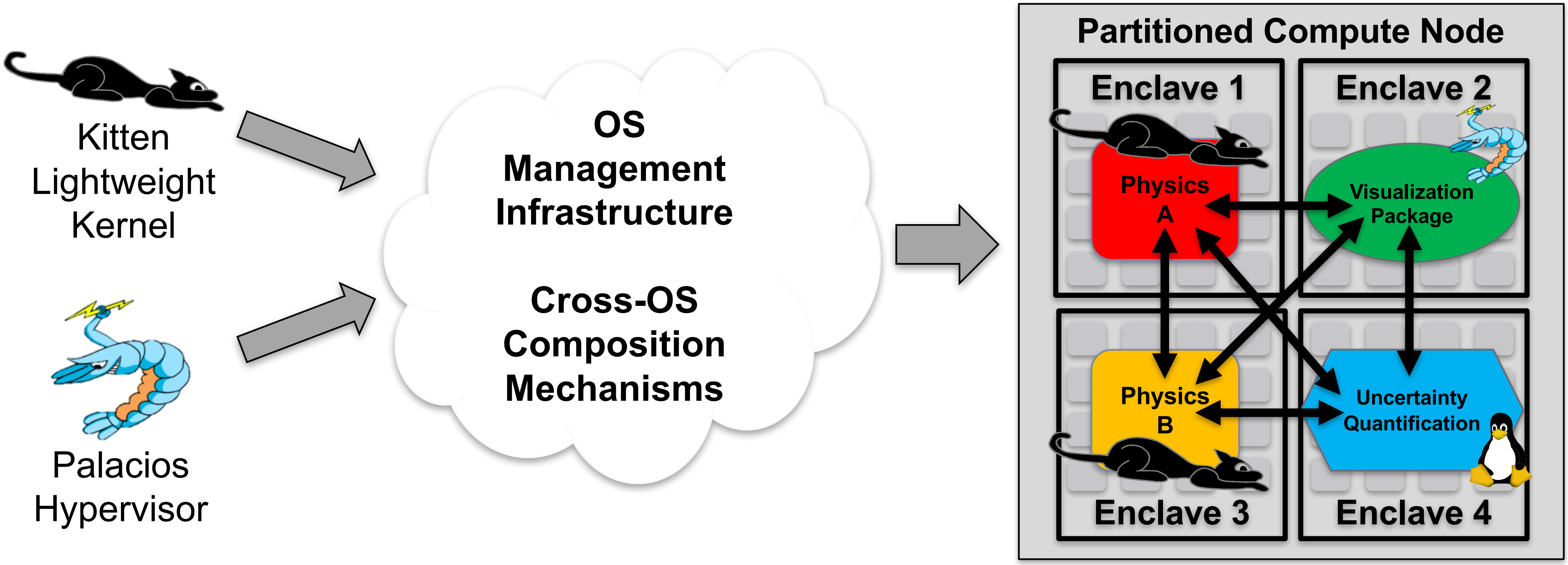
Compute Node System Software Stack, OS Bypass

The Hobbes
Node Virtualization Layer (NVL)



Generalized system software infrastructure for partitioning a compute node's resources (CPUs, memory, disk, NICs) into **space-shared enclaves**, launching **multiple OS/R instances** one per enclave, and portable interfaces for **selectively relaxing isolation** for cross-enclave composition

The Hobbes Node Virtualization Layer (NVL)

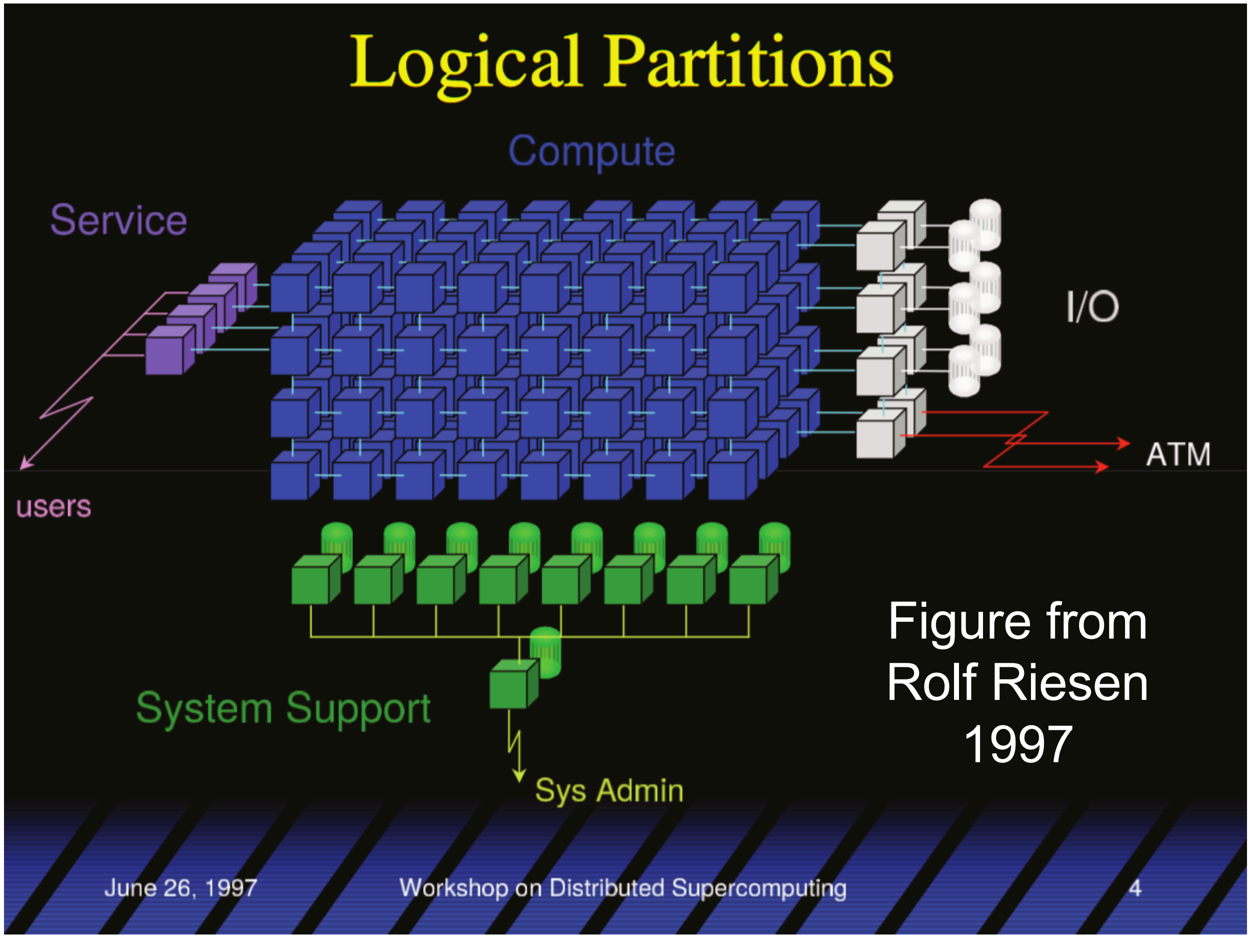


Unique Aspects of Hobbes NVL

- Run native and virtual OS/R stacks side by side
- Performance isolation at hardware **and** system software levels
- Cross OS/R stack composition mechanisms

Applying Massively Parallel Processor Partition Model to the Node

Compute Node

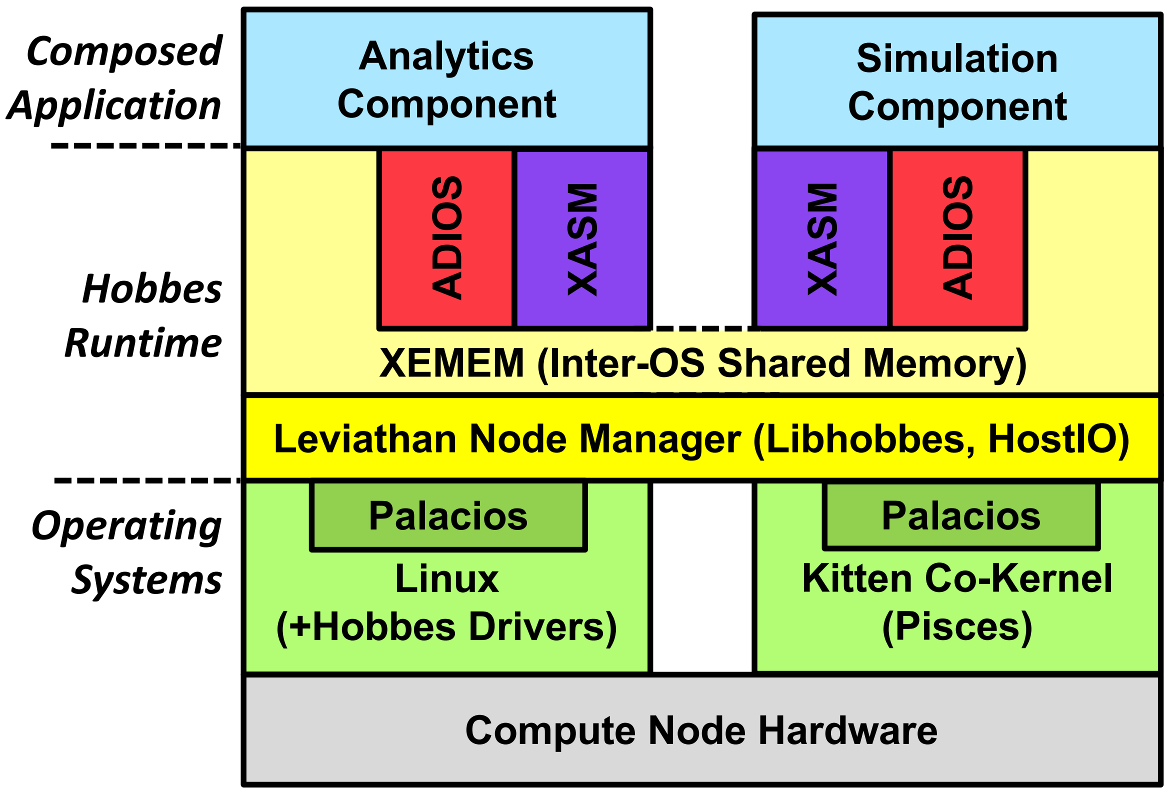


Outline

- Hobbes Node Virtualization Layer (NVL)
- NVL Components
  - Operating Systems: Linux, Kitten, and Palacios
  - Glue: XEMEM, Pisces, Leviathan
  - Composition: ADIOS, XASM, XEMEM
- Hobbes on Cray XC
- Future Direction

Hobbes Node Virtualization Layer Architecture

Enables Multiple Native + Virtual OS/R Stacks to Run Concurrently



Linux and LWK running side by side as Co-kernels

Key Ideas

- No one-size-fits-all OS/R
- Partition node-level resources into "enclaves"
- Run (potentially) different OS/R stack in each enclave

Challenges

- Performance isolation
- Composition mechanisms

Approach

- Build a real, working system
- Integrate with vendor's infrastructure + extend

Hobbes NVL Operating Systems

- Host Linux
  - Vendor supplied and supported
  - Extent with Hobbes kernel drivers

github.com/hobbesosr/kitten

- Kitten Lightweight Kernel
  - SUNMOS (1993), Cougar (1997), Catamount (2004), Kitten (2008-)
  - Linux ABI + API compatible user space, compile on Linux run on Kitten
  - Runs standalone or as part of Hobbes OS/R

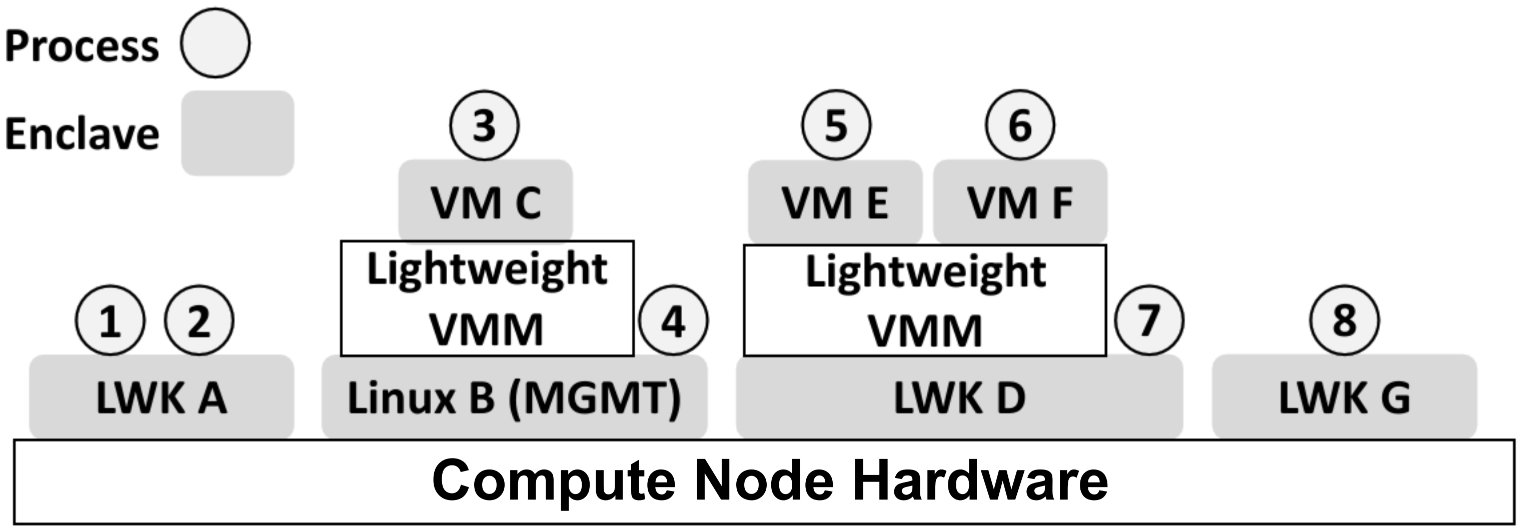
www.prognosticlab.org
www.v3vee.org

- Palacios Virtual Machine Monitor
  - OS independent, easily embeddable design
  - Lightweight resource management policies
  - Relies on x86 arch virtualization extensions
  - Demonstrated < 5% overhead for HPC workloads on 4K nodes (VEE'11)

Hobbes NVL Glue: XEMEM

www.prognosticlab.org/xemem

Enables Shared Memory Between Any Process in Any Enclave



- Maintains simplicity of single OS programming
- Processes need no knowledge of enclave topology
- Challenges Addressed: **Unique Naming** and **Discoverability**

[Kocoloski et al., HPDC'15]

XEMEM Interfaces

www.prognosticlab.org/xemem

- API backwards compatible with Cray/SGI XPMEM API
- XEMEM adds synchronization to the XPMEM API (wait and signal)

Function	Operation
xpmem_make	Export address region as shared memory. Returns *segid*.
xpmem_remove	Remove an exported region associated with a *segid*.
xpmem_get	Request access to shared memory region associated with a *segid*. Returns permission grant.
xpmem_release	Release permission grant.
xpmem_attach	Map a region of shared memory associated with a *segid*.
xpmem_detach	Unmap a region of shared memory.

[Kocoloski et al., HPDC'15]

Pisces Resource Management

www.prognosticlab.org/pisces

- Enables multiple native OS/R stacks to run concurrently
- Resources hot-removed from host Linux and given to Pisces
- Kitten modified to be Pisces-aware, access assigned resources only
- Minimal kernel-to-kernel communication, via IPIs and shared mem

Operations	Latency (ms)
Booting a Kitten co-kernel	265.98
Adding a single CPU core	33.74
Adding a 128MB memory block	82.66
Adding an Ethernet NIC	118.98

Fast Pisces Management Operations

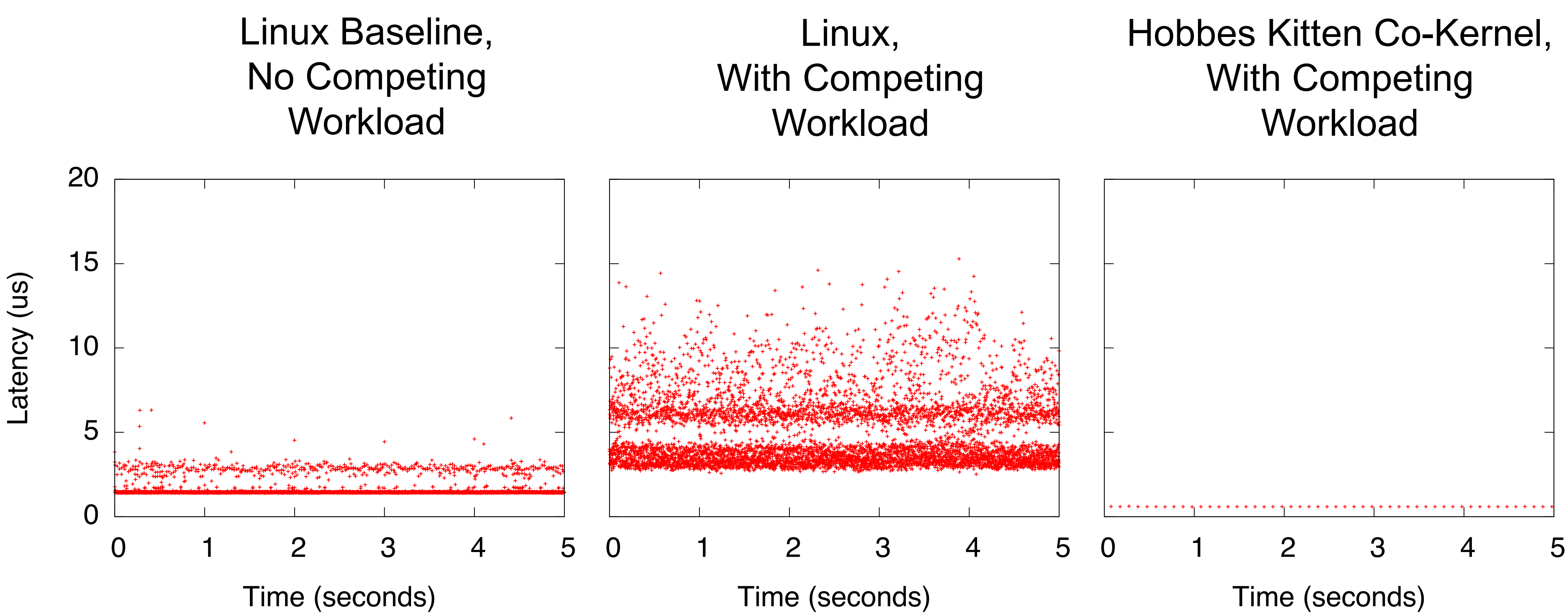
[Ouyang et al., HPDC'15]

Pisces Provides Excellent Performance Isolation

www.prognosticlab.org/pisces

Hardware is not the only shared resource, system software also matters

Two socket node
Socket A: Selfish OS Noise Benchmark
Socket B: Nothing or Linux Kernel Build



[Ouyang et al., HPDC'15]

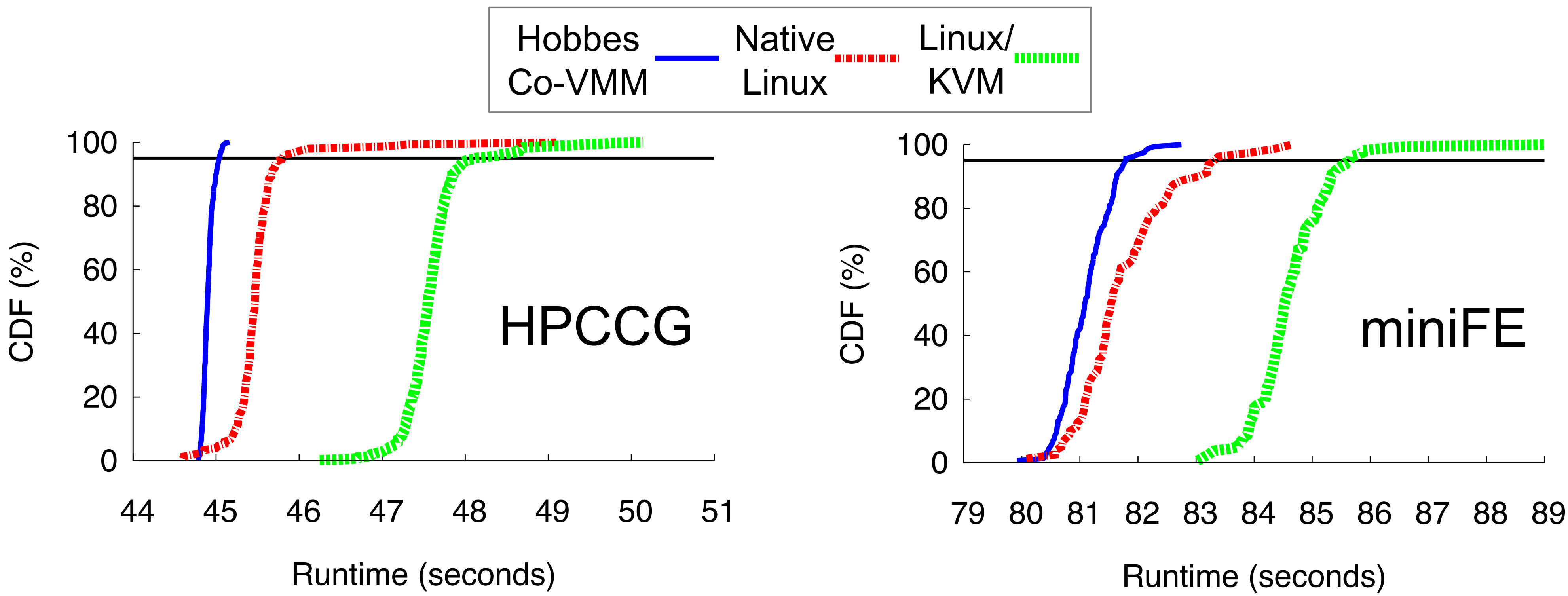
Pisces Increases Performance and Reduces Variability

www.prognosticlab.org/pisces

Performance Isolation for Hardware and System Software

8 Nodes:

Socket 0	Socket 1
Linux OS/R	Kitten OS/R
Hadoop ML Benchmark	**Mantevo Mini-app**

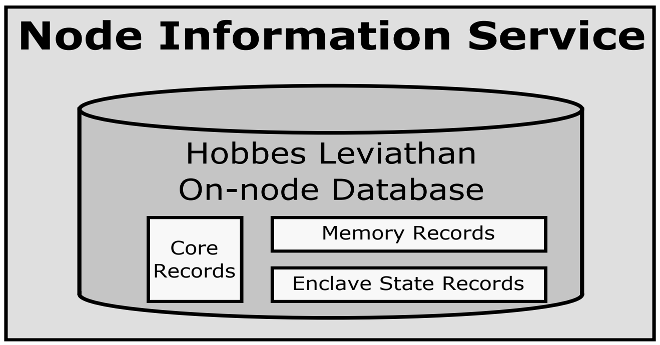


[Ouyang et al., HPDC'15]

Hobbes NVL Glue: Leviathan

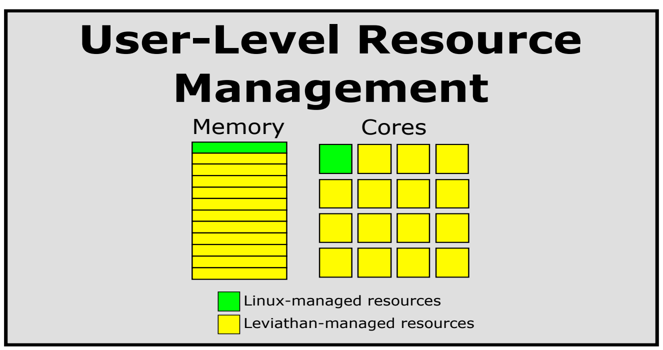
www.prognosticlab.org/leviathan

Generalized interfaces for managing and configuring multiple OS/R enclaves running on the same compute node; OS/R agnostic

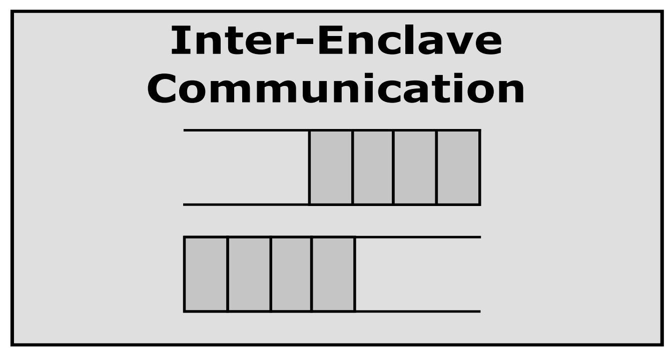


State of all resources tracked in in-memory NoSQL database

The Leviathan Hobbes shell provides commands to form enclaves and launch applications



User-level has explicit control of physical resources managed by Leviathan



Built-in services for command queues, discovery, global IDs, and generic host I/O

Hobbes NVL Glue: Leviathan

www.prognosticlab.org/leviathan

Entity: Any piece of software that can manage a raw piece of hardware

Resource: Any piece of hardware that is functionally isolatable

In-Memory Resource Database

Memory Table

Rsrc ID	*Hobbes Entity*	Phys ID	Alloc'd
M0	E0	0x100000	Yes
M1	E1	0x200000	Yes
M2	N/A	0x300000	No
M3	A0	0x400000	Yes
M4	A1	0x500000	Yes

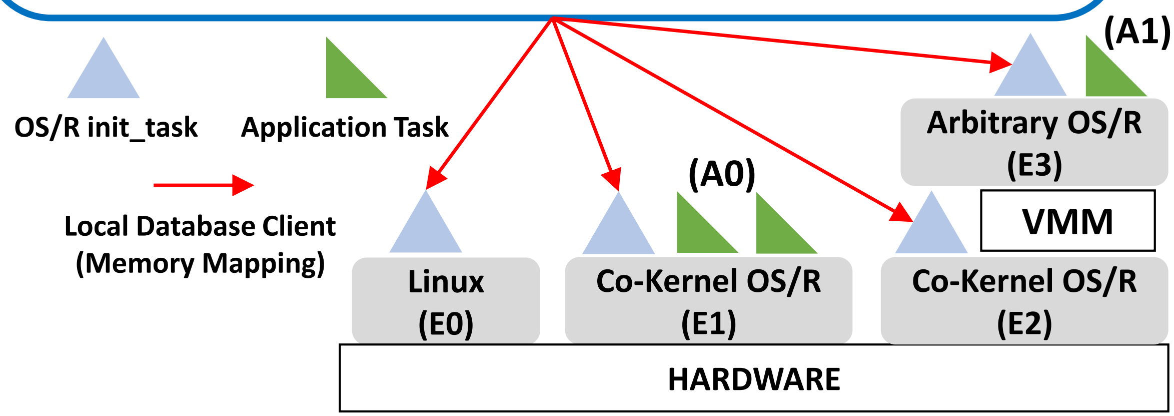
Core Table

Rsrc ID	*Hobbes Entity*	Phys ID	Alloc'd
C0	E0	Apic 0	Yes
C1	N/A	Apic 2	No
C2	E1	Apic 4	Yes
C3	E1	Apic 6	Yes
C4	E2	Apic 8	Yes

Device Table

Application Table

...



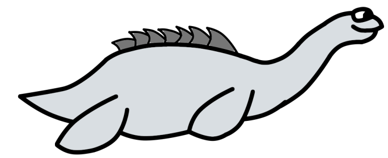
To integrate a new OS/R with Leviathan, OS must be modified to be able to map abstract resource ID handles to entities.

This minimally requires OS support for:

- Hotplug/unplug
- PCI
- XEMEM

Plus a user-level control daemon

Leviathan Hobbes Shell



www.prognosticlab.org/leviathan

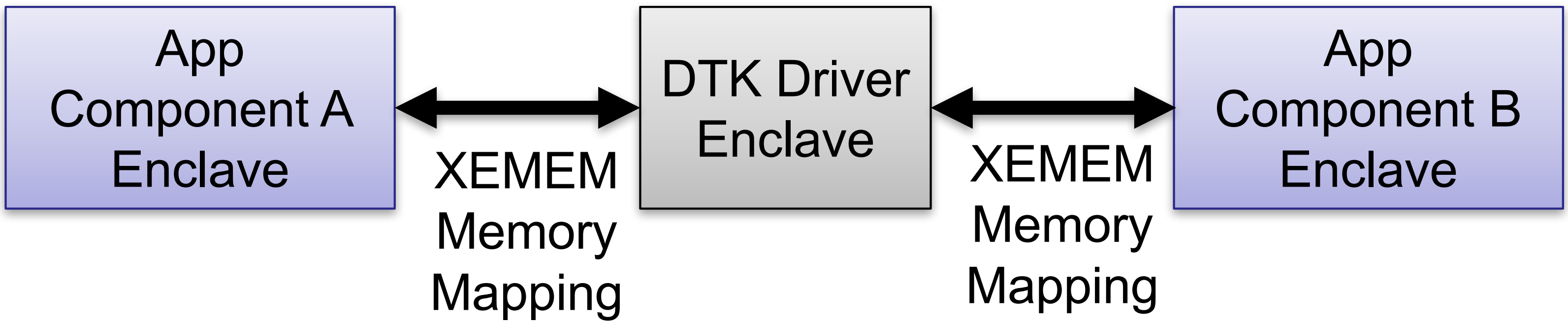
```
# ./hobbes
Hobbes Runtime Shell 0.1
Report Bugs to <jacklange@cs.pitt.edu>
Usage: hobbes <command> [args...]
Commands:
    create_enclave      -- Create Native Enclave
    destroy_enclave     -- Destroy Native Enclave
    create_vm           -- Create VM Enclave
    destroy_vm          -- Destroy VM Enclave
    ping_enclave        -- Ping an enclave
    list_enclaves       -- List all running enclaves
    list_segments       -- List all exported xemem segments
    launch_app          -- Launch an application in an enclave
    list_apps           -- List all applications
    dump_cmd_queue      -- Dump the command queue state for an enclave
    cat_file            -- 'cat' a file on an arbitrary enclave
    cat_into_file       -- 'cat' to a file on an arbitrary enclave
    list_memory         -- List the status of system memory
    list_cpus           -- List the status of local CPUs
    list_pci            -- List the status of PCI devices
    assign_memory       -- Assign memory to an Enclave
    assign_cpus         -- Assign CPUs to an Enclave
    assign_pci          -- Assign PCI device to an Enclave
    remove_pci          -- Remove PCI device from an Enclave
    console             -- Attach to an Enclave Console
```

Hobbes shell similar in concept to numactl

Hobbes Composition Mechanisms

ADIOS: [Kocoloski et al., ROSS'15]
XASM: [Evans et al., ROSS'16]

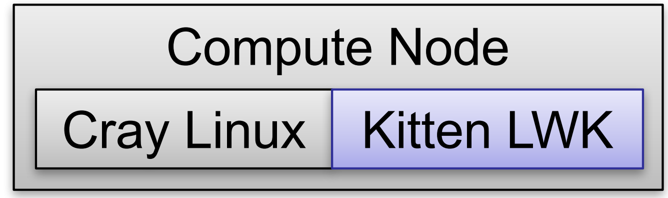
- XEMEM transport for ADIOS
  - ADIOS: High performance middleware enabling flexible data movement
  - Many applications already using it
- XASM – Cross Enclave Asynchronous Shared Memory
  - Adds copy-on-write semantics to XEMEM memory mappings
  - Producer can export a snapshot and then continue immediately
- Data Transfer Kit (DTK) modified to use Hobbes XEMEM
  - Each component runs in a separate enclave
  - Driver enclave uses XEMEM to access each component's memory



Outline

- Hobbes Node Virtualization Layer (NVL)
- NVL Components
  - Operating Systems: Linux, Kitten, and Palacios
  - Glue: XEMEM, Pisces, Leviathan
  - Composition: ADIOS, XASM, XEMEM
- Hobbes on Cray XC
- Future Direction

Hobbes on Cray XC



1. Load Hobbes drivers on each compute node

```
rmmod xpmem                # Unload Cray xpmem
insmod petos.ko            # Load Hobbes PetOS support module
insmod xpmem.ko ns=1       # Load Hobbes XEMEM /w nameserver
insmod pisces.ko           # Load Hobbes Pisces framework
```

2. Start Hobbes daemon on each compute node

```
lnx_init --cpulist=0,16 ${@:1} &
```

3. Use Hobbes shell to load Kitten enclave on each compute node

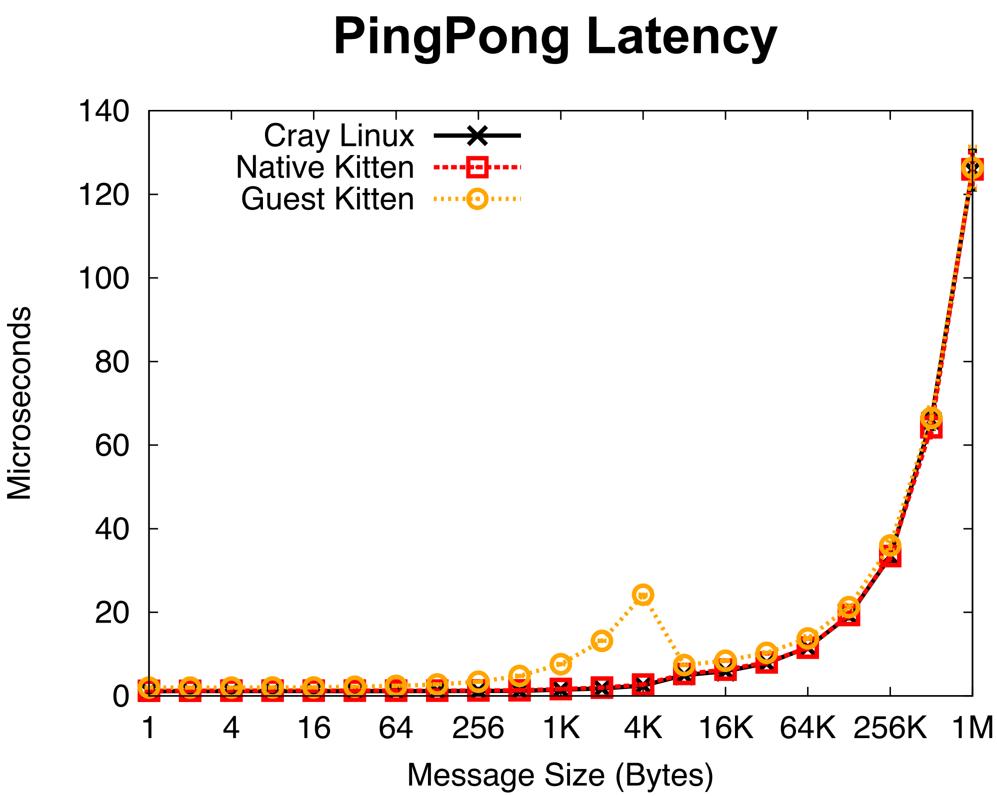
```
hobbes create_enclave kitten_enclave.xml kitten-enclave-0
```

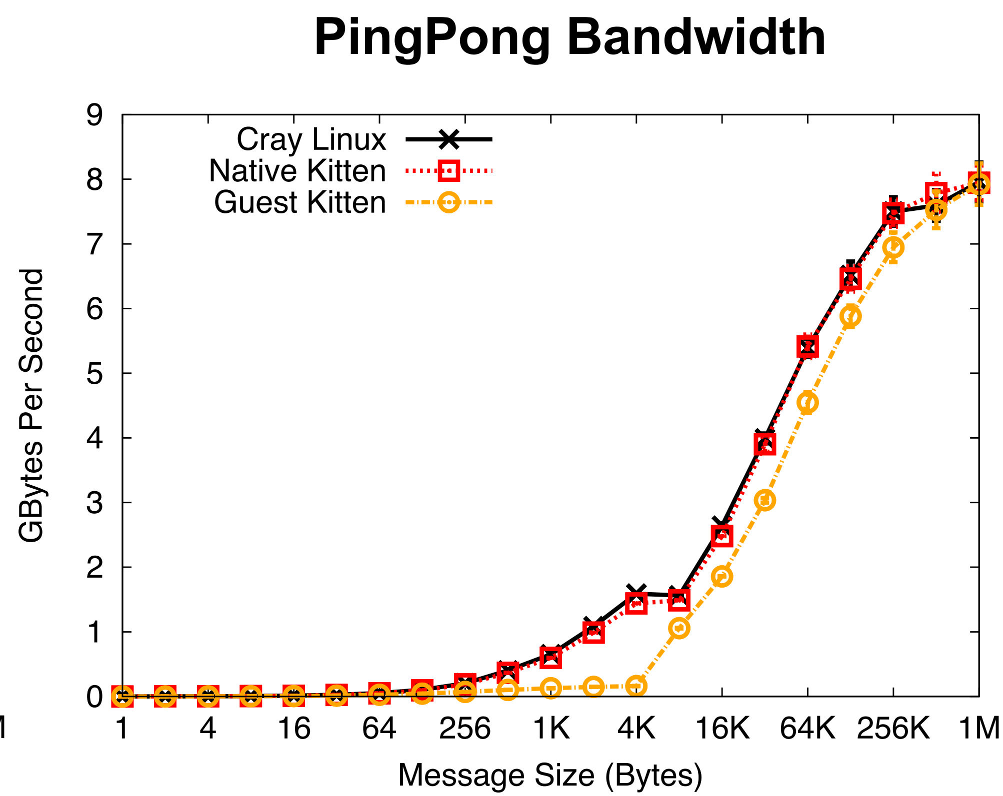
4. Build app like normal, using Cray's normal toolchain
5. Use Hobbes shell with aprun to launch application on Kitten

```
aprun –N 1 –n 32 ./hobbes launch_app kitten-enclave-0 \
    IMB-MPI1.cray_mpich
```

MPI Point to Point Performance

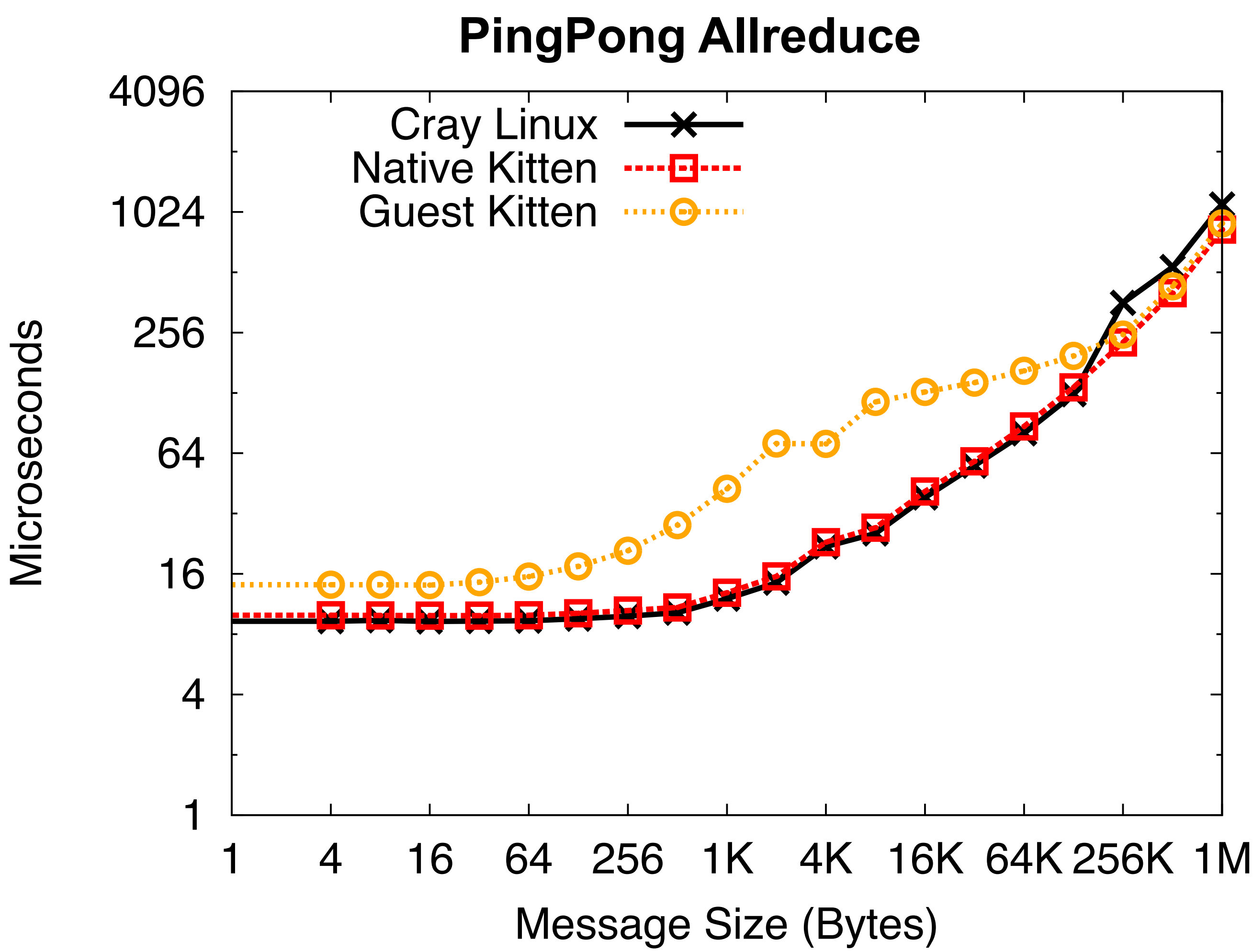
IMB 2017 Benchmark, built with standard Cray toolchain, MPI over Aries
Same binary used for all environments





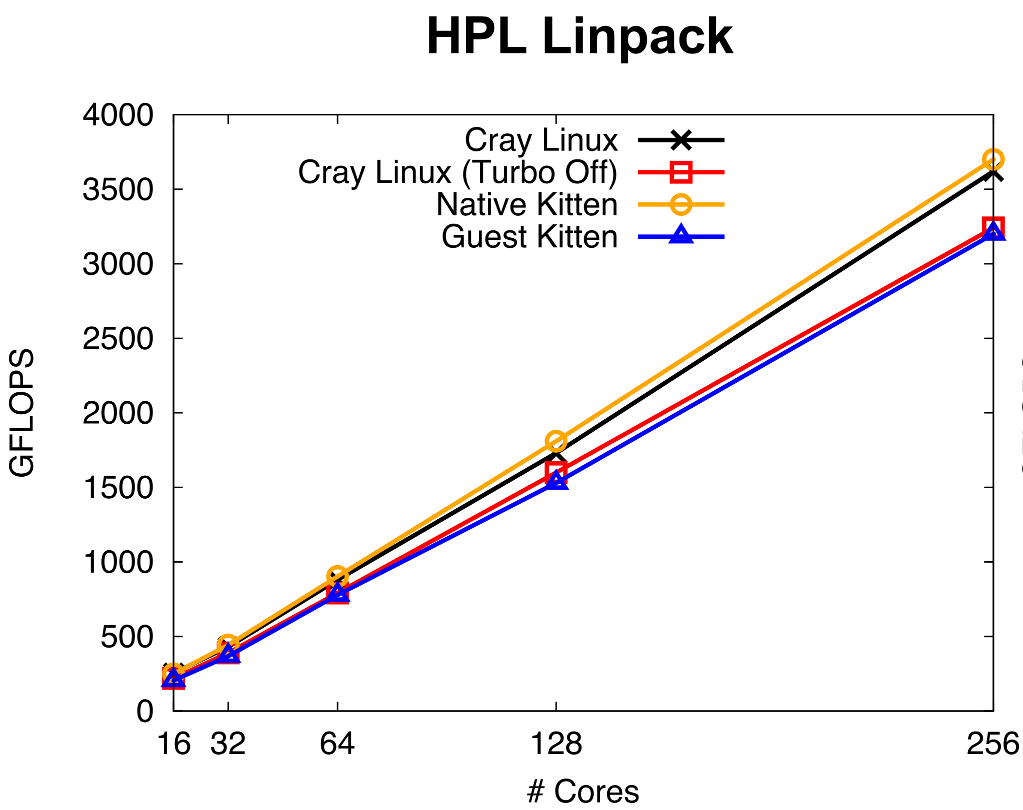
MPI Allreduce on 32 Nodes

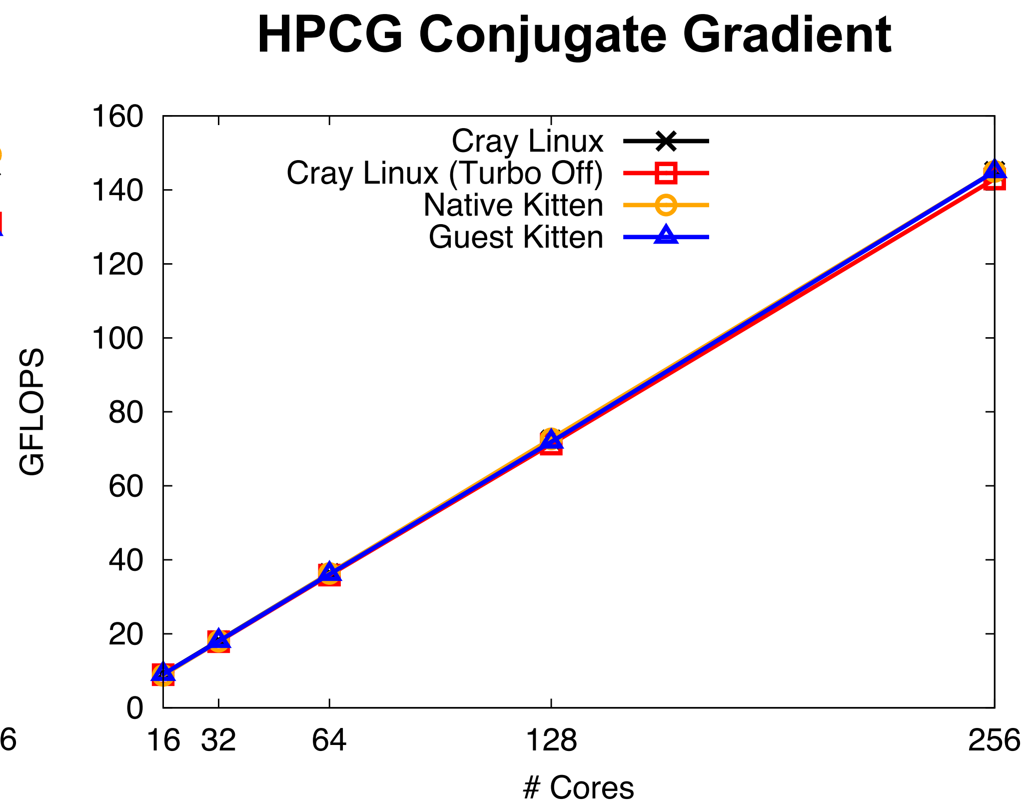
IMB 2017 Benchmark, built with standard Cray toolchain, MPI over Aries
Same binary used for all environments



Top 500 Benchmarks on 32 Nodes

IMB 2017 Benchmark, built with standard Cray toolchain, MPI over Aries
Same binary used for all environments



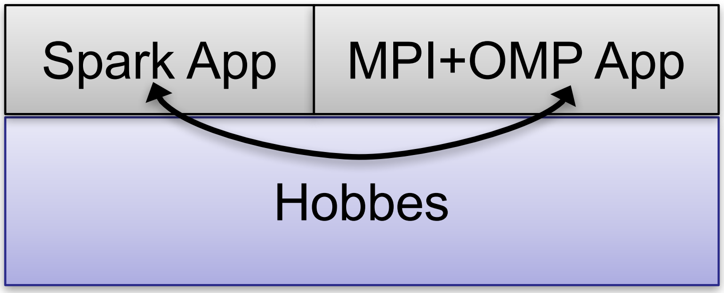


Outline

- Hobbes Node Virtualization Layer (NVL)
- NVL Components
  - Operating Systems: Linux, Kitten, and Palacios
  - Glue: XEMEM, Pisces, Leviathan
  - Composition: ADIOS, XASM, XEMEM
- Hobbes on Cray XC
- Future Direction

Composing HPC with Data-centric Computing

- Many working on supporting HPC or Data-centric in isolation
- Few working on HPC+Data composition
  - Like MPI+X, the "+" is a key challenge
  - Need effective ways to share data structures, ideally with no copying
- Hobbes infrastructure provides a good starting point
  - Provides explicit resource partitioning with sharing + multiple OS/Rs
  - Must find compelling use case drivers, engage with users from start
  - Explore space of loose-coupling of separate peer programs vs. tight-coupling into an integrated runtime system



Acknowledgments

- University of Pittsburgh
  - Jack Lange, Brian Kocoloski
- Oak Ridge National Laboratory
  - Barney Maccabe, David Bernholdt, Geoffroy Vallee, Thomas Naughton, Stuart Slattery
- University of New Mexico
  - Patrick Bridges
- Northwestern University
  - Peter Dinda
- Los Alamos National Laboratory
  - Mike Lang
- Sandia
  - Noah Evans
  - Shyamali Mukherjee