The Hobbes Node Virtualization Layer: Lessons Learned and Path Forward

March 23, 2017
SOS 21

Kevin Pedretti
Center for Computing Research
Sandia National Laboratories

SAND2017-3130 PE
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
- Lessons Learned + Path Forward
Why Virtualization in Large-Scale HPC?

- 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
    - 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)
Lightweight Kernel Drivers Still Valid

- Lots of new hardware + software challenges to tackle
  - Heterogeneous cores and memory, node-local NVRAM, complex on-chip networks, power management, ...
  - LWK is 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 huge battle with Linux “community”
  - Examples: mmunotify, huge pages, OOM killer, page coloring, XPMEM
  - Vendors ship “special sauce” Linux kernel patches, not upstreamable
What is 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.
What is the Hobbes Node Virtualization Layer? (NVL)

### Unique Aspects of Hobbes NVL

- Ability to run **native** and **virtual** OS/R stacks side by side
- Cross OS/R stack composition mechanisms
- Performance isolation design goal
Applying MPP Partition Model to the Node

Rolf Riesen, SOS 1, 1997
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
- Lessons Learned + Path Forward
Hobbes Node Virtualization Layer Architecture
Enables Multiple Native + Virtual OS/R Stacks to Run Concurrently

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

Linux and LWK running side by side as Co-kernels
Hobbes NVL Glue: 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]
Pisces Resource Management

- 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

<table>
<thead>
<tr>
<th>Operations</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booting a Kitten co-kernel</td>
<td>265.98</td>
</tr>
<tr>
<td>Adding a single CPU core</td>
<td>33.74</td>
</tr>
<tr>
<td>Adding a 128MB memory block</td>
<td>82.66</td>
</tr>
<tr>
<td>Adding an Ethernet NIC</td>
<td>118.98</td>
</tr>
</tbody>
</table>

Fast Pisces Management Operations

[Ouyang et al., HPDC’15]
Pisces Increases Performance and Reduces Variability

**Performance Isolation for Hardware and System Software**

8 Nodes:

- **Socket 0**
  - Linux OS/R
  - Hadoop ML Benchmark

- **Socket 1**
  - Kitten OS/R
  - Mantevo Mini-app

**Graphs:**

- **HPCCG**
  - CDF (%)
  - Runtime (seconds)
  - Hobbes (Co-VMM), Native, Linux/KVM
  - [Ouyang et al., HPDC’15]

- **miniFE**
  - CDF (%)
  - Runtime (seconds)
  - Hobbes (Co-VMM), Native, Linux/KVM

[www.prognosticlab.org/pisces](http://www.prognosticlab.org/pisces)
Hobbes NVL Glue: Leviathan

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

Node Information Service

State of all resources tracked in in-memory NoSQL database

Enclave Lifecycle Management

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

User-Level Resource Management

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

Inter-Enclave Communication

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

# ./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 `xmem` 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

- **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

**Diagram**

- **App Component A**
  - Enclave
- **XEMEM Memory Mapping**
- **DTK Driver Enclave**
- **XEMEM Memory Mapping**
- **App Component B**
  - Enclave

*ADIOS: [Kocoloski et al., ROSS’15]*
*XASM: [Evans et al., ROSS’16]*
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
- Lessons Learned + Path Forward
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 PingPong Latency

- Cray Linux [Aries]
- Kitten Co-Kernel [Aries]
- Cray Linux [Gemini]
- Kitten Co-Kernel [Gemini]

OS Bypass = ~ Identical Latency
Kitten vs. Cray Linux

IMB 2017 Benchmark
MPI PingPong Bandwidth

GBytes Per Second

Message Size (Bytes)

Cray Linux [Aries]
Kitten Co-Kernel [Aries]
Cray Linux [Gemini]
Kitten Co-Kernel [Gemini]

Aries Is OS Bypass For Large Messages

Gemini Not OS Bypass

IMB 2017 Benchmark
MPI Collectives, 32 Nodes

**MPI Reduce**

![Graph showing MPI Reduce performance for different message sizes and platforms.]

**MPI Allreduce**

![Graph showing MPI Allreduce performance for different message sizes and platforms.]

**MPI Alltoall**

![Graph showing MPI Alltoall performance for different message sizes and platforms.]

**MPI Bcast**

![Graph showing MPI Bcast performance for different message sizes and platforms.]
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
- Lessons Learned + Path Forward
1. Performance isolation is not just about hardware, system software activities matter too

Hobbes Provides Excellent Performance Isolation

Linux Baseline, No Competing Workload

Linux, With Competing Workload

Hobbes Kitten Co-Kernel, With Competing Workload

[Ouyang et al., HPDC’15]
2. Networks that don’t have built-in virtualization support are a pain

- Needed way to share high-speed NIC between enclaves
  - HPC hardware generally lacks SR-IOV support, but is "sort of" self virtualizing in that it maps the NIC into multiple processes
- Had to develop system call forwarding layer, part of Leviathan
  - Built on XEMEM, command queues, and cross enclave signals
  - Depends on control plane being slow path, data plane being OS bypass
  - Handles drivers that do evil things, like use ioctl() to map memory

Thanks to McKernel for this approach
3. Vendors are still interested in lightweight kernels (just not ours)

- Intel developing mOS multi-kernel (Linux + LWK)
- RIKEN + Fujitsu developing McKernel multi-kernel for Post K
- Cloud community doing a ton of OS/R work
  - Reducing tail latencies through Linux patches and config tuning
  - Unikernels – sort of like lightweight kernels for cloud workloads
- Hobbes NVL-like infrastructure provides path to breaking free from the “locked down vendor OS/R stack”
  - More than two (as many OS/R stacks as you want, native or virtual)
  - Generalized interfaces and mechanisms for composition
  - Supports new use cases that require virtualization

First time Sandia LWK on a Cray since Red Storm
4. **Hardware performance is becoming more variable**

- Many sources of variability
  - Opportunistic frequency scaling (Turbo)
  - Power capping, power budget shifting between CPU, Memory, GPU, ...
  - Thermal throttling
  - Manufacturing part-to-part differences

---

**Single-node HPL Variability Across 100 Nodes**

**P-State Sweep**
*(Fixed Performance, Variable Power)*

**Power Cap Sweep**
*(Fixed Power, Variable Performance)*

---

**Mutrino HPL 1-32 GFLOPS vs Avg-Power**

- Hitting the Power Cap
- Each point is a node, avg 5 trials with error bars
5. Users really want containers

- Docker wasn’t really around when we started Hobbes
- Now all the rage, users eager to try it out
- Good application packaging and delivery vehicle
  - Mostly solves user-level software dependency problems
  - Doesn’t address use cases that require full virtual machines
- HPC specific adaptations, NERSC Shifter, LBL Singularity
- Challenges
  - How to compose across containers for HPC workflows
  - Achieving performance isolation between containers
  - Security, portability across HPC systems, and forward compatibility

Hobbes Infrastructure Could Support Containers
Path Forward

- **High-Level Project Outcomes**
  - Generalized system software infrastructure for running multiple OS/R stacks on a node and building cross-stack compositions
  - Demonstrated excellent performance isolation between enclaves
  - Demonstrated how to integrate with a vendor’s existing OS/R stack

- **Hobbes is over, but some work continuing**
  - Larger scale experiments, LWK evaluations
  - Analytics + Data Science on HPC systems

- **We were a bit ahead of the game with Hobbes**
  - Users still figuring out what they need for workflows + composition
  - Apps we were trying to work with weren’t really ready for composition
  - Need to better define how components expose and share information, essential for effective composition
Acknowledgements

- Ron Brightwell, Noah Evans, Kurt Ferreira, Brian Gaines, Jay Lofstead, Shyamali Mukherjee (Sandia)
- Jack Lange, Brian Kocoloski, Jiannan Ouyang (U. Pittsburgh)
- Patrick Bridges, Oscar Mondragon (U. New Mexico)
- Peter Dinda, Kyle Hale (Northwestern)
- Mike Lang (Los Alamos)
- Barney McCabe, David Bernholdt, Hasan Abbasi, Geoffroy Vallee, Thomas Naughton, Stuart Slattery (Oak Ridge)
- Jai Dayal (Georgia Tech)
Extra Slides
Example above shows three enclaves, two native and one virtual machine.
Each application component runs in its own enclave, which is a partition of the compute node’s resources (CPUs, memory, NICs).
Approach leads to excellent performance isolation across enclaves.
XEMEM allows user level memory to be shared across enclaves, useful tool for application composition.

HPDC’15: “Achieving Performance Isolation with Lightweight Co-Kernels”
HPDC’15: “XEMEM: Efficient Shared Memory for Composed Applications”
Application 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**
Hobbes NVL Has Multiple Levels of Virtualization

- Existing Hypervisors typically support one level, strict isolation
- NVL couples LWK “native” runtime with guest OS/R stacks

Application

Virtual Linux Runtime
Native LWK Environment

Application

Guest OS/R Stack
Optimized for NVL

Application

Guest OS/R Stack
No NVL Optimizations

Application

Guest OS/R Stack
NVL HW Emulation

Application

“Native” Guest OS/R
Guest OS modified to cooperate with NVL, Runs on Bare Metal

Hobbes NVL (Node Virtualization Layer)
Hobbes NVL Provides Composition Mechanisms

Inter-OS/R Stack Memory Mapping / Sharing (e.g., XPMEM)
Efficient Inter-OS/R Stack Networking (e.g., Portals4, Nessie)
Memory Snapshots (e.g., Multi-buffer)
I/O-based Composition (e.g., ADIOS)

Simulation App
Virtual Linux Runtime
Native LWK Environment

Analysis App
Guest OS/R Stack
Optimized for NVL

Key/Value Store
Guest OS/R Stack
No NVL Optimizations

Burst Buffer Proxy
Guest OS/R Stack
NVL HW Emulation

Simulation App
“Native” Guest OS/R
Guest OS modified to cooperate with NVL, Runs on Bare Metal
Leviathan Enclave Launch

# ./hobbes create_enclave cray_kitten_enclave.xml kitten-enclave-a
Launching Enclave (/dev/pisces-enclave0) on CPU 2

# ./hobbes create_enclave cray_kitten_enclave.xml kitten-enclave-b
Launching Enclave (/dev/pisces-enclave1) on CPU 4

# ./hobbes create_enclave cray_kitten_enclave.xml kitten-enclave-c
Launching Enclave (/dev/pisces-enclave2) on CPU 6

# ./hobbes create_enclave cray_kitten_enclave.xml kitten-enclave-d
Launching Enclave (/dev/pisces-enclave3) on CPU 29

# ./hobbes list_enclaves
5 Active Enclaves:

<table>
<thead>
<tr>
<th>ID</th>
<th>Enclave name</th>
<th>Type</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>master</td>
<td>MASTER_ENCLAVE</td>
<td>Running</td>
</tr>
<tr>
<td>1</td>
<td>kitten-enclave-a</td>
<td>PISCES_ENCLAVE</td>
<td>Running</td>
</tr>
<tr>
<td>2</td>
<td>kitten-enclave-b</td>
<td>PISCES_ENCLAVE</td>
<td>Running</td>
</tr>
<tr>
<td>3</td>
<td>kitten-enclave-c</td>
<td>PISCES_ENCLAVE</td>
<td>Running</td>
</tr>
<tr>
<td>4</td>
<td>kitten-enclave-d</td>
<td>PISCES_ENCLAVE</td>
<td>Running</td>
</tr>
</tbody>
</table>
Hobbes Composition Mechanisms

- **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

Works across enclave boundaries

- Linux to Linux
- Linux to Kitten
- Kitten to Kitten
- Native—Native, Native—VM, VM—VM

[Evans et al., ROSS’16]