VEE’16 Panel: Sweet Spots and Limits for Virtualization

April 3, 2016

Kevin Pedretti
Center for Computing Research
Sandia National Laboratories
Large-Scale HPC Systems in General

- Distributed-memory MIMD
- Message passing used between nodes (MPI)
- Custom interconnect, leverage commodity processors and memory
  - Network point-to-point latency \( \sim 1 \) us, bandwidth \( \sim 10 \) GBytes/s and growing
- Several node types, system software for each type specialized to task
- Big systems typically \( \$100M - \$200M \) procurements

**Compute Partition**
- \( > 10K \) Nodes
- Runs Physics Simulations, Fortran/C/C++, MPI/OpenMP, Batch Scheduled

**Login Partition**
- 10’s of Nodes
- Runs Full Linux Distribution (SUSE, RedHat)

**I/O Nodes**
- 100’s of Nodes
- Runs Full Linux Distribution, Optimized for Storage I/O

**Runs Slimmed-Down OS, Nodes Typically Diskless (LWK, Customized Linux)**
Why Virtualization in Large-Scale HPC?

- Support multiple system software stacks in same platform
  - Vendor’s stack good for physics simulations, bad for data analytics
  - Virtualization adds flexibility, deploy custom images on demand
  - Not just user-space containers, need ability to run different OS kernels
    - Special-purpose Lightweight Kernels: mOS, McKernel, Kitten
    - Large-scale emulation experiments, networks + systems
  - Leverage industry momentum, student mindshare

- Virtualization overhead can be very low
  - Don’t oversubscribe, space share nodes, pin everything, use large pages, physically contiguous virtual memory
  - Demonstrated < 5% overhead in practice on 4K nodes (VEE’11)

- Challenges
  - Deployment: getting virtualization into vendor’s stack
  - Networking: pass-through OK, but want to be able to share NIC between VMs, getting RDMA drivers in guest, migration
  - Complex nodes: heterogeneous memory, many-core, SMT, NUMA, ...
Hobbes: Multi-Stack Approach for Application Composition

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
- Leverage Kitten LWK and Palacios Hypervisor
- Use standard Linux host for bootstrap and enclave control
- Develop libhobbes for use by Apps/Tools/Services

Team:
- Kevin Pedretti, Jay Lofstead, Brian Gaines, Shyamali Mukherjee, Noah Evans (SNL)
- Jack Lange, Brian Kocoloski, Jiannan Ouyang (Pitt)
- Patrick Bridges, Oscar Mondragon (UNM)
- Peter Dinda, Kyle Hale (Northwestern)
- Mike Lang (LANL)
- David Bernholdt (Enclave lead), Hasan Abbasi (ORNL)
- Jai Dayal (GaTech)

Node Virtualization Layer (NVL)

<table>
<thead>
<tr>
<th>Application</th>
<th>Analytics</th>
<th>Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hobbes Runtime</td>
<td>XEMEM</td>
<td>ADIOS</td>
</tr>
<tr>
<td>Operating System</td>
<td>Vendor’s Linux OS (+ Pisces Driver)</td>
<td>Kitten Co-Kernel (Pisces)</td>
</tr>
</tbody>
</table>

Compute Node Hardware

Team:
- Kevin Pedretti, Jay Lofstead, Brian Gaines, Shyamali Mukherjee, Noah Evans (SNL)
- Jack Lange, Brian Kocoloski, Jiannan Ouyang (Pitt)
- Patrick Bridges, Oscar Mondragon (UNM)
- Peter Dinda, Kyle Hale (Northwestern)
- Mike Lang (LANL)
- David Bernholdt (Enclave lead), Hasan Abbasi (ORNL)
- Jai Dayal (GaTech)

http://github.com/hobbesosr/nvl

HPDC’15
Thank You
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
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

![Diagram showing different levels of virtualization]

- Application
  - Virtual Linux Runtime
  - Native LWK Environment
  - Optimized for NVL
  - Guest OS/R Stack
  - Guest OS/R Stack
  - No NVL Optimizations
  - NVL HW Emulation
  - “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

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

Hobbes NVL (Node Virtualization Layer)