Opportunities for Leveraging OS Virtualization in High-End Supercomputing

MASVDC’10:
Workshop on Micro Architectural Support for Virtualization, Data Center Computing, and Clouds

December 5, 2010

Kevin Pedretti
Sandia National Labs
Albuquerque, NM
ktpedre@sandia.gov

Patrick Bridges
Univ. of New Mexico
Albuquerque, NM
bridges@cs.unm.edu
Outline

• Introduction
• Previous Work
• High-End HPC Virtualization Use Cases
• Results
• Conclusion
Apples and Oranges, But…
No Doubt Mainstream Virtualization Seeing Explosive Growth

Sources: SC web sites, news articles, blog posts
Virtualization in HPC?

• “Every problem in computer science can be solved with another level of abstraction” ;-)  
• “No virtualization in HPC”  
  – Well, we (usually) have virtual memory  
  – Virtualization is potentially disruptive  
    • Clayton M. Christensen's keynote at SC’10  
    • Won’t/Can’t attack established HPC initially, may sneak up over time

Vendors steadily decreasing virtualization overhead and adding capabilities
Virtualization in High-End HPC?

- Compelling use cases not necessarily dependent on achieving absolute highest performance
  - Increase flexibility, app-specific OS/runtime
  - Enable new capabilities not present today
  - Modest overheads tolerable
- Well known techniques such as VMM-bypass and large paging mitigate overheads

Our results show virtualization overhead is low, typically less than 5%
Outline

• Introduction
• Previous Work
• High-End HPC Virtualization Use Cases
• Results
• Conclusion
Previous Work: Motivation and I/O Optimization

• Motivation for migrating HPC workloads to VMs
  (ICS’06: Huang, Liu, Abali, Panda)
  – Ease of management (live migration, checkpoint)
  – Ability to run custom tailored OS (LWK)
  – Exposing privileged ops to user (kernel modules)

• High-performance I/O
  – VMM-bypass (USENIX’06: Liu, Huang, Abali, Panda)
  – Migrating VMM-bypass VMs (VEE’07: Huang, Liu, Koop, Abali, Panda)
  – PGAS applications in Xen VMs
    (Cluster’07: Scarpazza, Mullaney, Villa, Petrini, Tipparaju, Brown, Nieplocha)
Previous Work:
Resiliency and Overhead Reduction

• Proactive VM migration to improve resiliency
  (ICS’07: Nagarajan, Mueller, Engelmann, Scott)
  (FGCS-Mar10: Scott, Vallee, Naughton, Tikotekar, Engelmann, Ong)
  – Migrate away from nodes with observed deteriorating health
  – Reactive checkpoint frequency can be reduced if MTTI improved

• Nested paging to reduce VM exits
  – AMD nested paging, Intel EPT
  – 2-D nested page table caching scheme
    (ASPLOS’08: Bhargava, Serebrin, Spadini, Manne)
  – NPT structure does not have to match native
    (CAL-Jan10: Hoang, Bae, Lange, Zhang, Dinda, Joseph)
Previous Work:
Cloud and VM Scalability

• **Using public clouds for HPC**
  – Migrating workloads and performance measurements
    (SC’08: Deelman, Singh, Livny, Berriman, Good)
    (GC’09: Hill, Humphrey)
  – Amazon’s EC2 HPC instances with 10GigE + GPUs

• **Scalability of MPI apps in VM on Cray XT**
  (IPDPS’10: Lange, Pedretti, Hudson, Dinda, Cui, Xia, Bridges, Gocke, Jaconette, Levenhagen, Brightwell)
  – Micro-benchmarks and real applications
  – Up to ~6K nodes, more on way
Outline

- Introduction
- Previous Work
- High-End HPC Virtualization Use Cases
- Results
- Conclusion
Enhancing Lightweight OS Flexibility

• Original motivation
• LWK provides high perf. native environment
• VMM allows full-featured guest OS (e.g., Red Hat Linux) to be loaded on-demand
  – Perl, python, matlab, …
  – COTS databases, simulators, …
  – You name it
• Approach also applies to lightweight Linux distributions like CLE (Cray Linux Env.)

Kitten available from: http://code.google.com/p/kitten/
Palacios available from: http://v3vee.org/
Tool for Exascale OS Research

• Obtaining dedicated time on supercomputer to test prototype OS is HARD
• VM capability would partially mitigate
  – Test prototype “X-stack” at scale, expose effects that only occur at scale
  – Rapid turnaround for debug iterations
  – VM is convenient instrumentation layer
• Support HW/SW co-design efforts
  – Prototype new HW/SW interfaces and capabilities
  – Tie to architectural simulator
Enable New Capabilities

• Internet-scale simulation
  – Run commodity OSes and software
  – Multiple virtual nodes per physical node

• Migration based on VMM-level runtime monitoring
  – Better map application onto network topology
  – Migrate memory pages among NUMA nodes
  – Make up for all VMM overhead and more (?)

• Provide backwards compatibility
  – Support legacy software on future exascale systems
  – Provide incremental path to native environment
Outline

• Introduction
• Previous Work
• High-End HPC Virtualization Use Cases

• Results

• Conclusion
# Test Platform

| **Processor** | Intel X5570 2.93 GHz quad-core  
|              | 2 sockets, 8 cores total  
|              | 2 NUMA nodes  
|              | Theoretical Peak: **94 GFLOPS** |
| **Memory**   | 24 GB DDR3-1333  
|              | Three 4 GB DIMMs per socket  
|              | Theoretical Peak: **64 GB/s** |
| **BIOS Configuration** | Hyper-Threading Disabled  
|                   | Turbo-Boost Disabled  
|                   | Maximum Performance |
| **Software**  | Linux 2.6.36.7 with **KVM**  
|              | Guest image identical to host  
|              | kvm-clock para-virtualized clock, plus ntp daemon  
|              | **NUMA topology exposed to guest**  
|              | libhugetlbfs for large paging |
Benchmarks

• Compute overhead
  – Linpack (HPCC HPL)

• Memory overhead
  – OpenMP STREAM
  – GUPs (HPCC MPIRandomAccess)

• MPI
  – PingPong (IMB PingPong)
    Intra-node only, via shared mem (MPICH2 Nemesis)
HPL Linpack
No Compute Virtualization Overhead

![Graph showing Gflops vs. # Cores for different configurations: Native 2M, Native 4K, Guest 2M/2M, Guest 4K/2M, Guest 2M/4K, and Guest 4K/4K. The lines are nearly overlapping, indicating minimal virtualization overhead.]
OpenMP STREAM

Little Memory BW Virtualization Overhead

![Graph](image.png)
MPI Random Access
2.5% to 40% Overhead Depending on Config
MPI PingPong
Latency in Guest More Variable
Bandwidth Essentially Identical

Variability possibly due to
timekeeping inaccuracy in guest
VMM-Bypass MPI Latency on Cray XT4
Avoiding Interrupt Virtualization Important

![Graph showing latency vs. message size for different configurations: Native, Guest, Native, Accel Portals, Guest, Accel Portals. The graph illustrates the impact of interrupt virtualization overhead on latency.]
Conclusions

• Virtualization support continuously improving
• Significant previous HPC virtualization work
• Compelling use cases for high-end HPC
  – Increase flexibility
  – Enable new capabilities
• Results show low virtualization overhead
  – NUMA and VCPU pinning important in all cases
  – Large paging important for random access
Acknowledgements

• **Funding**
  – DOE ASCR X-stack (current)
  – DOE ASC (current)
  – Sandia LDRD (past)

• **Collaborators**
  – Peter Dinda, Northwestern Univ.
  – Jack Lange, Univ. of Pittsburgh
  – Geoffroy Vallee, ORNL