The Assessment of Benchmarks Executed on Bare-Metal and Using Para-Virtualisation

Mark Baker, Garry Smith and Ahmad Hasaan
SSE, University of Reading
Paravirtualization

• A full assessment of paravirtualization is important, because without knowledge about the various overheads, users can not understand whether using virtualization is a good idea or not.

• In this presentation we are very interested in assessing the overheads of running various benchmarks on bare-metal, and on paravirtualization, and then turning on monitoring and logging.

• The knowledge from assessing various benchmarks on these different systems will help a range of users understand the use of virtualization systems.
Virtualization Systems

• In this paper we assess the overheads of using Xen, VMware, KVM and Citrix.
• These different virtualization systems are used extensively by cloud-users.
• We are using various Netlib benchmarks, which have been developed by the University of Tennessee at Knoxville (UTK), and Oak Ridge National Laboratory (ORNL).
  – http://www.netlib.org/
• In order to assess these virtualization systems, we run the benchmarks on bare-metal, then on the paravirtualization, and finally we turn on monitoring and logging.
• The later is important as users are interested in Service Level Agreements (SLAs) used with the Cloud providers, and the use of logging is a means of assessing the services bought and used from commercial providers.
**Paravirtualization**

- Para-virtualization is an OS-assisted virtualization; where some modifications are made in the guest operating system to enable better performance.
- In this kind of virtualization, the guest operating system is aware of the fact that it is running on the virtualized hardware and not on the bare hardware.
- In Paravirtualization the device drivers in guest operating system coordinates with the device drivers of host operating system in reducing the performance overhead.
- The use of Para-virtualization is intended to avoid the bottleneck associated with slow hardware interrupts that exist when full virtualization is employed.
HPC Systems

• The project will use the Thamesblue supercomputer, the Hactar cluster and the IBM JS20 blade server available at University of Reading for benchmarking of different Virtualization systems.

• Thamesblue are:

• **IBM PowerPC Blade Centre JS21 Cluster**
  - CPUs - 2800 2.5 GHz processors
  - RAM - 20 TeraFlops of sustained performance
  - DISK - 5.6 Terabytes.
  - *Network connectivity* - Myrinet/Ethernet

• The Hactar cluster consists of one head node and four worker nodes:
  - **Dual-Core AMD Opteron(tm) Processor 1212**
  - CPUs - 2009.438 MHz
  - RAM - 4 Gigabytes
  - *Network Connectivity* - Ethernet

15/09/2010 mark.baker@computer.org
Hardware Ring used in Virtualization

- Ring 3: Applications
- Ring 2
- Ring 1
  - Ring 0: Most Privileged Zone
    - Guest Operating System
      - E.g. Linux or Windows
  - Hypervisor/Virtualization layer
  - Physical Hardware
    - [CPU, RAM, Disk, Network & Disk controllers]

Direct Instruction execution

15/09/2010 mark.baker@computer.org
Virtualization Systems Accessed.

<table>
<thead>
<tr>
<th>Type of System</th>
<th>Name of VM based systems</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Source Systems</td>
<td>XEN</td>
<td><a href="http://www.xen.org">http://www.xen.org</a></td>
</tr>
<tr>
<td></td>
<td>KVM</td>
<td><a href="http://www.linux-kvm.org">http://www.linux-kvm.org</a></td>
</tr>
<tr>
<td>Commercial Systems</td>
<td>VMware</td>
<td><a href="http://www.vmware.com">http://www.vmware.com</a></td>
</tr>
<tr>
<td></td>
<td>Citrix</td>
<td><a href="http://www.citrix.com">http://www.citrix.com</a></td>
</tr>
</tbody>
</table>

15/09/2010 mark.baker@computer.org
Comms1

- **COMMS1**, is a pingpong, benchmark that measures the basic communication properties of a message-passing MIMD computer.
  - A message of variable length, n, is sent from a master node to a slave node.
- The slave node receives the message into a Fortran data array, and immediately returns it to the master.
- Half the time for this message pingpong is recorded as the time, T, to send a message of length, N,
- In this case advantage can be taken of bidirectional links, and a greater bandwidth can be obtained than is possible with COMMS1.
- The COMMS1 benchmark computes all four of the above parameters, because each emphasizes a different aspect of performance.
- In the case that there are different modes of transmission for messages shorter or longer than a certain length, the benchmark can read in this breakpoint and perform a separate least-squares fit for the two regions.
PARKBench BENCHMARKs

PARKBench (PARallel Kernels and BENCHMARKs)

Logging & Monitoring OFF
- Baro-Metal Real time
- KVM Real time
- XEN Real time
- VMware Server Real time

Time per message (sec)

Message Size (Bytes)
Lapack

- **LAPACK** is written in Fortran90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.
- The associated matrix factorisations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations and estimating condition numbers.
- The original goal of the **LAPACK** project was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors.
- **LAPACK** addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops.

15/09/2010 mark.baker@computer.org
LAPACK -- Linear Algebra PACKage

LAPACK

15/09/2010 mark.baker@computer.org

LAPACK is written in Fortran90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

If you're uncertain of the LAPACK routine name to address your application's needs, check out the LAPACK Search Engine.

The original goal of the LAPACK project was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors. On these machines, LINPACK and EISPACK are inefficient because their memory access patterns disregard the multi-layered memory hierarchies of the machines, thereby spending too much time moving data instead of doing useful floating-point operations. LAPACK addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops. These block operations can be optimized for each architecture to account for the memory hierarchy, and so provide a transportable way to achieve high efficiency on diverse modern machines. We use the term "transportable" instead of "portable" because, for fastest possible performance, LAPACK requires that highly optimized block matrix operations be already
ScALAPACK

- The ScALAPACK library includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers.
- It is currently written in a Single-Program-Multiple-Data style using explicit message passing for inter-processor communication.
  - It assumes matrices are laid out in a two-dimensional block cyclic decomposition.
- ScALAPACK is designed for heterogeneous computing and is portable on any computer that supports MPI or PVM.
- ScALAPACK routines are based on block-partitioned algorithms in order to minimise the frequency of data movement between different levels of the memory hierarchy.
- The building blocks of the ScALAPACK library are distributed memory versions of the Level 1, 2 and 3 BLAS, and a set of Basic Linear Algebra Communication Subprograms (BLACS) for communication tasks that arise frequently in parallel linear algebra computations.
- One of the design goals of ScALAPACK was to have the ScALAPACK routines resemble their LAPACK equivalents as much as possible.
ScaLAPACK

The ScaLAPACK Project

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. ASC-9313958 and DOE Grant No. DE-FG03-94ER25219. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF) or the Department of Energy (DOE).

# The ScaLAPACK Project

This directory contains a number of different software packages involved in the ScaLAPACK project.

*********

PLEASE NOTE: ScaLAPACK, version 1.8.0, is now available!

*********

ScaLAPACK installer is now available!

PROTOTYPE codes are in scalapack/prototype directory.

The URL for the BLAS is http://www.netlib.org/blas/.
The URL for the BLACS is http://www.netlib.org/blacs/.
The URL for the LAPACK is http://www.netlib.org/lapack/.
The URL for PVM is http://www.netlib.org/pvm/.
The URL for MPI is http://www.netlib.org/mp/.
ScalAPACK

ScalAPACK LU Factorization Benchmarks

Logging & Monitoring ON

Bare-Metal Real time
KVM Real time
KEN Real time
VMware Server Real time

Time (sec)

Problem Sizes (N)

15/09/2010
mark.baker@computer.org
DL_POLY

• DL_POLY is a general purpose serial and parallel molecular dynamics simulation package developed at Daresbury Laboratory by W. Smith, T.R. Forester and I.T. Todorov.

• The original package was developed by the Molecular Simulation Group (now part of the Computational Chemistry Group, MSG) at Daresbury Laboratory under the auspices of the Engineering and Physical Sciences Research Council (EPSRC) for the EPSRC's Collaborative Computational Project for the Computer Simulation of Condensed Phases (CCP5).

• Later developments were also supported by the Natural Environment Research Council through the eMinerals project. The package is the property of the Central Laboratory of the Research Councils.
High Performance LAPACK (HPL) Benchmarks

DL_POLY

15/09/2010 mark.baker@computer.org
DL_POLY Molecular Simulation Package

1. Simulation of metallic aluminium at 300K temperature
2. Simulation of 15 peptide in 1247 water molecules
3. Simulation of a sodium-potassium disilicate glass

- Bare-Metal Real time
- KVM Real time
- KEN Real time
- VMware Server Real time

Real/Total Time (sec)

Molecular Simulation Benchmark Number
It has been found that PARKBench ping-pong communication benchmark performs best on the bare metal.

The throughput decrease is less on XEN virtualisation system than KVM and VMware.

The performance of virtualisation systems improves as the size of the message increases.

The performance trend between XEN, KVM and VMware remains consistent for smaller and larger message sizes.

The results also show that the logging and monitoring imposes less performance overhead on the virtualisation systems.

It is also interesting to note that for all the virtualisation systems, the overhead of logging and monitoring is not significant.
HPL

- The performance of HPL benchmarks on virtualisation systems is almost equivalent to the bare metal performance.
- For the problem size 2000, the time taken by XEN is 1.0057 times the time taken by bare-metal.
- Similarly the time taken by KVM is 1.011 times the time taken by bare metal and the time for VMware is 1.048 times the time taken by bare metal.
- The results 1.0057, 1.011 and 1.048 for XEN, KVM and VMware show that the virtualisation overhead on the performance of HPL application is not significant.
- The same trend has been observed for the problem sizes 1000, 3000 and 4000.
The DL_POLY molecular simulation benchmarks performance is slower on the KVM and VMware virtualisation system.

This is an interesting result because DL_POLY is also computational intensive application like HPL.

It is also interesting to note that the performance of DL_POLY on XEN virtualisation system is almost comparable with bare metal performance.

For the benchmark “Simulation of a sodium-potassium dislocates glass” the time taken by XEN is 1.0037 times the time on bare metal.

The time taken by the same benchmark on KVM is 1.3117 times the time on the bare metal and the time on VMware is 1.3192 times the time taken by the benchmark on the bare metal.

Similarly like in other benchmarks, the logging and monitoring affects KVM and VMware more than the XEN in DL_POLY benchmarks.

15/09/2010 mark.baker@computer.org