A case for HPC with Virtual Machines
Virtualization

Ability to run multiple OS instances (split address space) in one single machine

Resources like CPU, memory, and I/O devices are shared.

- Significant overhead only for I/O sharing.

Management Efficiency
Benefits for HPC

Ease of management
- Template based provisioning.
- VM migration and checkpoint-restart

Customized OS
- Light-weight, highly customized.

System security
- Can allow users to manage/alter kernel level services.
Design points

VMM Bypass

- Guest module in Guest dom does all privileged actions for initial setup
- Guest OS and user processes can now send/receive directly.

Improve the VM image management

- Customized kernels/OSes for HPC applications
- fast and scalable distribution schemes

Figure 4: VMM-Bypass I/O
Framework for VM Based Clusters

Front-end
- Users submit jobs and customized VM images

Physical resources
- Compute nodes with one VM per core

Management Module
- Match request with available resources.
- Distribute VM image instantiate the VMs

VM Image manager
- Database of VM images
- Images created by admin or submitted by users

Storage

Figure 5: Framework for Cluster with Virtual Machine Environment
Reducing VM image distribution cost

Small image
- Built for specific tasks, highly customized OS images

Fast and scalable distribution
- Binomial tree distribution

VM image caching
- Cache recently used VM images on local storage
Evaluation

Figure 7: MPI latency test

Figure 8: MPI bandwidth test

Figure 12: NAS Parallel Benchmarks (16 processes, class B)

Figure 13: HPL on 2, 4, 8 and 16 processes
Evaluation (contd...)

<table>
<thead>
<tr>
<th></th>
<th>Dom0</th>
<th>Xen</th>
<th>DomUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>3.6</td>
<td>1.9</td>
<td>94.5</td>
</tr>
<tr>
<td>SP</td>
<td>0.3</td>
<td>0.1</td>
<td>99.6</td>
</tr>
<tr>
<td>BT</td>
<td>0.4</td>
<td>0.2</td>
<td>99.4</td>
</tr>
<tr>
<td>EP</td>
<td>0.6</td>
<td>0.3</td>
<td>99.3</td>
</tr>
<tr>
<td>CG</td>
<td>0.6</td>
<td>0.3</td>
<td>99.0</td>
</tr>
<tr>
<td>LU</td>
<td>0.6</td>
<td>0.3</td>
<td>99.0</td>
</tr>
<tr>
<td>FT</td>
<td>1.6</td>
<td>0.5</td>
<td>97.9</td>
</tr>
<tr>
<td>MG</td>
<td>1.8</td>
<td>1.0</td>
<td>97.3</td>
</tr>
</tbody>
</table>

Table 2: Distribution of execution time for NAS

<table>
<thead>
<tr>
<th>Scheme</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binomial tree</td>
<td>1.3s</td>
<td>2.8s</td>
<td>3.7s</td>
<td>5.0s</td>
</tr>
<tr>
<td>NFS</td>
<td>4.1s</td>
<td>6.2s</td>
<td>12.1s</td>
<td>16.1s</td>
</tr>
</tbody>
</table>

Table 4: VM image distribution time
VM Migration using RDMA
Migration

- Helper processes on Dom0 do the transfer
- Multiple iterations in the Pre-copy stage
- Small downtime for final iteration
- Page translation tables need to be de-sensitized (mfn -> pfn)
Motivation

Obvious benefit of high bandwidth

Memory is accessed directly

• Reduces context switches between VM and Dom0

Reduces burden on target host.
Design points

Efficient migration over RDMA

• Overlap RDMA transfer of normal pages with page table translation.
• Dynamically choose between RDMA read and RDMA write based on load factors

Memory Registration

• Helper proc in Dom0 can’t register VM’s memory
  • Use direct data transfer using H/W address from the mfn entries.
• Translated pages sent using send-receive
Design Points (contd …)

Page clustering to send bigger chunks

• Reorder table by mfn value.
• Use RDMA read with scatter or RDMA write with gather.

QoS to throttle network utilization

• XEN’s adaptive usage is conservative
• Instead use maximum bandwidth first and throttle to lower if needed.
**Evaluation**

Fig. 6. Total migration time

Fig. 8. Migration downtime

Fig. 10. SPEC CINT 2000 (1 CPU)

Fig. 11. Impact of migration on applications in a non-migrating VM

Fig. 13. Adaptive rate limit control
Evaluation (contd...)