vBlades: Optimized Paravirtualization for the Itanium Processor Family

Usenix 3rd Virtual Machine Research and Technology Symposium (VM’04), May 6-7 2004

Dan Magenheimer (speaker) and Tom Christian
Hewlett-Packard Laboratories

© 2004 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice
Presentation Overview

• Virtualization overview
  – taxonomy
  – low-level VMM terminology
  – virtualizability

• Paravirtualization

• vBlades

• Conclusions
Taxonomy of VM Architectures

Virtual Machines (VMs)

Process VMs
- Multiprogrammed Systems
  - all modern OS’s
- High Level Language VMs
  - Java
  - .NET/CLR
- System VMs
  - different ISA

Classic OS VMs
- same ISA
  - VMware
  - MS Virtual Server
  - Xen
  - vBlades

Whole System VMs
- different ISA
  - Transmeta Crusoe
  - IBM Daisy

High-level vs low-level VM’s

<table>
<thead>
<tr>
<th>Linux app</th>
<th>Linux app</th>
<th>Java App</th>
<th>Java App</th>
<th>Linux</th>
<th>processor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>processor</td>
</tr>
</tbody>
</table>

VMM == virtual machine monitor
Low-level VMM: Hostless vs hosted

<table>
<thead>
<tr>
<th>Win32 App</th>
<th>Win32 App</th>
<th>Linux App</th>
<th>Linux App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Linux</td>
<td>VMM</td>
<td>processor</td>
</tr>
</tbody>
</table>

Type I (hostless)

<table>
<thead>
<tr>
<th>Win32 App</th>
<th>Win32 App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Windows</td>
</tr>
<tr>
<td>Linux app</td>
<td>Low-level VMM</td>
</tr>
<tr>
<td>Linux</td>
<td>processor</td>
</tr>
</tbody>
</table>

Type II (hosted)
What makes a processor virtualizable?

- PRIVILEGED instructions (trap if not PL0)
- SENSITIVE instructions (potentially disruptive of VMM or host OS)
- All processor instructions

If all sensitive instructions are privileged, the processor is “virtualizable”
Virtualizable processors
NOT “fully virtualizable”
• Intel x86 (IA-32)
• Intel Itanium family

Virtualizable
• IBM mainframes
• PowerPC
• future Intel processors (Vanderpool/Silervale)
Privilege/ring compression

- **PL3**: User
- **PL2**: unused
- **PL1**: unused
- **PL0**: Kernel

**Normal OS**

- **PL3**: User
- **PL2**: unused
- **PL1**: unused
- **PL0**: Kernel

**Hostless VMM w/ compression**

- **PL3**: User
- **PL2**: unused
- **PL1**: Kernel
- **PL0**: VMM

**“PL – 1” Hostless VMM on a fully virtualizable processor**
Consequences of non-full virtualizability

• Increased complexity
  – dynamically rewriting portions of guest OS
  – managing “shadow” page tables
  – avoiding “privilege leakage” (or not)
  – special drivers

• Significant performance impact
  – degradation of up to 30-50% or more
Alternative: Paravirtualization

- The VMM presents a machine abstraction that is similar – but not identical to – the physical machine.
- The guest operating system *must* be modified to accommodate the differing abstraction

Examples
- Denali (x86)
- Xen (x86)
- vBlades (Itanium)
Paravirtualization features

- Full multi-application commercial OS’s can be supported (Linux, Windows)
- Application modification is not necessary
Paravirtualization tradeoffs

• Disadvantages
  – Guest OS’s do not work “out of the box”

• Advantages
  – Near native performance
  – The guest OS can be made cognizant of the virtual environment (e.g. virtual vs real time, memory courtesy)
vBlades

- HP Labs research project
- Itanium-based hostless VMM
- Provides full virtualization
  - static translation for sensitive instructions
  - and provides API for paravirtualization
- Twelve simultaneous Linux’$s$ demonstrated
- Tuned to within 2% of native performance
- SMP support
- Network I/O model
- Guest OS migration support
vBlades contributions

- First Itanium VMM
- Optimized paravirtualization
- Transparent paravirtualization
- Metaphysical memory
Itanium virtualizability

- Itanium has three sensitive, non-privileged instructions – all rarely used
- Virtual memory system provides strong isolation:
  - Four full privilege levels
  - Region registers
  - Software-managed TLB
- Virtualization of Itanium register stack engine is painful, but possible
- Privilege leakage is expensive to avoid
Optimized paravirtualization

• Supporting both virtualization and paravirtualization provides valuable flexibility
  – “80/20” tuning rule applies
  – most startup code can remain unchanged
  – minimizing changes is a good thing
Transparent paravirtualization

- Allows common binary to run both on VMM and on bare metal.
- Performance impact is negligible.

- sensitive op
  - running on vBlades?
    - execute original OS code
    - execute paravirtualized code
  - done
Metaphysical* memory

- Presents illusion of physical memory to guest OS
- Eliminates need to paravirtualize physical memory code
- Supports oversubscription
- Simplifies live migration

* M-W: “a reality beyond what is available to the senses”
vBlades Conclusions

- Paravirtualization is a worthwhile research direction, perhaps even after industry-standard processors are fully virtualizable.
- Optimizing the tradeoffs of paravirtualization is possible.
- Creating a single OS binary that supports both paravirtualization and bare metal is possible and has a low impact.
- The VMM community needs to re-examine terminology for the middle-tier of virtual memory.