Introduction to Virtualization

SHARE Seattle
Linux & VM Program

Romney White, IBM
romneyw@us.ibm.com
z Systems Architecture and Technology
Introduction to Virtualization
- Concept
- Server Virtualization Approaches
- Hypervisor Implementation Methods
- Why Virtualization Matters

Virtualization on z Systems
- Logical Partitions
- Virtual Machines
Virtualization Concept

Virtual Resources
- Proxies for real resources: same interfaces/functions, different attributes
- May be part of a physical resource or multiple physical resources

Virtualization
- Creates virtual resources and "maps" them to real resources
- Primarily accomplished with software or firmware

Resources
- Components with architecturally-defined interfaces/functions
- May be centralized or distributed - usually physical
- Examples: memory, disk drives, networks, servers

• Separates presentation of resources to users from actual resources
• Aggregates pools of resources for allocation to users as virtual resources
Server Virtualization Approaches

**Hardware Partitioning**
- Server is subdivided into fractions each of which can run an OS
- Adjustable partitions

**Physical partitioning**
- S/370™ SI-to-PP and PP-to-SI
- Sun Domains, HP nPartitions

**Logical partitioning**
- IBM eServer™ pSeries® LPAR
- HP vPartitions

**Bare-metal Hypervisor**
- Hypervisor provides fine-grained timesharing of all resources
- Hypervisor software/firmware runs directly on server
- z Systems LPAR and z/VM®
- POWER™ Hypervisor
- VMware ESX Server
- Xen Hypervisor

**Hosted Hypervisor**
- Hypervisor uses OS services to do timesharing of all resources
- Hypervisor software runs on a host operating system
- VMware GSX
- Microsoft® Virtual Server
- HP Integrity VM
- KVM

**Characteristics:**
- Bare-metal hypervisors offer high efficiency and availability
- Hosted hypervisors are useful for clients where host OS integration is important
- Hardware partitioning is less flexible than hypervisor-based solutions
# Hypervisor Implementation Methods

## Trap and Emulate

<table>
<thead>
<tr>
<th>Virt Mach</th>
<th>L</th>
<th>A</th>
<th>ST</th>
<th>PrivOp</th>
<th>L</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trap</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Hypervisor PrivOp emulation code</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Examples**: CP-67, VM/370
- **Benefits**: Runs unmodified OS
- **Issues**: Substantial overhead

## Translate, Trap, and Emulate

<table>
<thead>
<tr>
<th>Virt Mach</th>
<th>L</th>
<th>A</th>
<th>ST</th>
<th>TrapOp</th>
<th>L</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trap</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Hypervisor PrivOp emulation code</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Examples**: VMware, Microsoft VS
- **Benefits**: Runs unmodified, translated OS
- **Issues**: Substantial overhead

## Hypervisor Calls (“Paravirtualization”)

<table>
<thead>
<tr>
<th>Virt Mach</th>
<th>L</th>
<th>A</th>
<th>ST</th>
<th>Hcall</th>
<th>L</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Call</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Hypervisor service</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Examples**: POWER Hypervisor, Xen
- **Benefits**: High efficiency
- **Issues**: OS must be modified to issue Hcalls

## Direct Hardware Virtualization

<table>
<thead>
<tr>
<th>Virt Mach</th>
<th>L</th>
<th>A</th>
<th>ST</th>
<th>PrivOp</th>
<th>L</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exit</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Hypervisor service</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Examples**: z Systems LPAR, z/VM, Xen
- **Benefits**: High efficiency, runs unmodified OS
- **Issues**: Requires underlying hardware support
Virtualization can fill many roles and provide many benefits. In the final analysis, its potential benefits take three forms:

- **Help reduce hardware costs**
  - Help increase physical resource utilization
  - Small footprints

- **Can improve flexibility and responsiveness**
  - Virtual resources can be adjusted dynamically to meet new or changing needs and to optimize service level achievement
  - Virtualization is a key enabler of on demand operating environments such as cloud

- **Can reduce management costs**
  - Fewer physical servers to manage
  - Many common management tasks become much easier
Multi-dimensional virtualization technology

- z Systems provides logical (LPAR) and software (z/VM) partitioning
- PR/SM enables highly scalable virtual server hosting for LPAR and z/VM virtual machine environments
- IRD coordinates allocation of CPU and I/O resources among z/OS and non-z/OS® LPARs*

* Excluding non-shared resources like Integrated Facility for Linux processors
Logical CPU
- Program
- Instruction stream

Problem state
- Instructions

High-Frequency Control
- Instructions that require virtualization

Low-Frequency Control
- Instructions that require hypervisor virtualization

SIE: Start Interpretive “Instruction” Execution

Load, Store, Add, ...

Start Subchannel, Test Subchannel, ...

SIE Interception to hypervisor
- E.g., Modify Subchannel

LPAR hypervisor

Physical CPU Instruction Execution Unit

Hardware
- Instruction Interpretation Handling
- Virtualization Assists

Instruction Execution Controls
- Hardware or Firmware

Load
- Store
- Add
- ...

SSCH
- TSCH

LPAR CPU STATE Descriptor

z Systems Virtualization Technology
© 2015 IBM Corporation
LPAR Logical Processor Dispatching

LPAR hypervisor dynamically dispatches:

1. General-purpose logical processors on general-purpose physical processors
2. zAAP logical processors on zAAP physical processors
3. zIIP logical processors on zIIP physical processors
4. IFL logical processors to IFL physical processors
LPAR Memory Partitioning

A collection of up to \(2^{31}\)-or \(2^{64}\)-bytes virtual address spaces
Each virtual machine may have its own virtual address spaces mapped to this common logical partition address space

LPAR 1

\[ \text{z/VM Virtual Machine Physical Memory Space} \]

LPAR 2

\[ \text{z/OS or Linux High-Performance Logical Partition Physical Memory Spaces} \]

LPAR N

= the real partition memory pages associated with a virtual address space; that is, the sets of dynamically-allocated physical memory pages necessary to run a z/OS task or a Linux process

= the real partition memory pages associated with a virtual machine; that is, the sets of dynamically-allocated physical memory pages necessary to run a guest operating system in a virtual machine

z/OS and Linux exploit multiple such virtual address spaces
LPAR Multiple Logical Channel Subsystems

- Logical Channel Subsystem Image 1
- Logical Channel Subsystem Image 2
- Logical Channel Subsystem Image 3
- Logical Channel Subsystem Image 4

FICON Channel Path
- Transparently Shared by All Logical Partitions

OSA Ethernet Adapter
- Transparently Shared By Logical Partitions Configured to Channel Subsystem Image N

Switches, Network Links, etc.

z Systems Physical Channel Subsystem
The I/O infrastructure (adapters/channels, their transmission links, and attached I/O resources are shared by LPARs at native speeds (without hypervisor involvement)

- I/O requests, their associated data transfers, and I/O interruptions flow between each OS instance and the shared I/O components, just as if the I/O components were physically dedicated to a single OS instance
SIE: Start Interpretive “Instruction” Execution

- **Problem state Instructions**
- **High-Frequency Control Instructions that require virtualization**
- **Instruction Execution Controls**
- **Set Storage Key, Signal Processor, ...**
- **Logical CPU => Physical CPU Instruction Execution Unit**

**Virtual CPU**
- **Program Instruction stream**
- **Control Instructions that require hypervisor virtualization**
- **z/VM hypervisor**
- **z/VM CPU STATE Descriptor**

**Hardware**
- **Load, Store, Add, ...**
- **Hardware or Firmware**
- **Instruction Interpretation Handling**
- **Virtualization Assists**

**Instruction stream**
- **Problem state Instructions**
- **High-Frequency Control Instructions that require virtualization**
- **Instruction Execution Controls**
- **Set Storage Key, Signal Processor, ...**
- **Logical CPU => Physical CPU Instruction Execution Unit**

**z/VM Virtual CPU Dispatching and Execution Control**

E.g., Start Subchannel
z/VM Virtual Processor Dispatching

z/VM hypervisor dynamically dispatches:
1. General-purpose Virtual processors on general-purpose Logical processors
2. zAAP Virtual processors on zAAP or general-purpose Logical processors
3. zIIP Virtual processors on zIIP or general-purpose Logical processors
4. IFL Virtual processors to IFL or general-purpose Logical processors
z/VM Memory Virtualization

Virtual Machine 1

z/VM Virtual Machine
Virtual Memory Space

z/VM Hypervisor

Virtual Machine 2

z/OS or Linux Virtual
Machine Virtual Memory
Spaces

Virtual Machine N

Logical Partition Hypervisor

= the guest real memory pages associated with a virtual address space; that is, the sets of
dynamically-allocated host virtual memory pages necessary to run a z/OS task or a Linux
process

= the guest real memory pages associated with a virtual machine; that is, the sets of
dynamically-allocated host virtual memory pages necessary to run a guest operating system
in a virtual machine
**z/VM Disk Virtualization**

- **Legend:**
  - R/W = Read/Write
  - R/O = Read-Only

- **Minidisk Cache** (High-speed, in-memory disk cache)
- **Virtual Disk in Memory**
- **Minidisk**: real disk partitioning technology

- **TDISK**: dynamic temporary disk allocation pool

- **ESS 750, ESS 800, DS8000, DS6000**

- **Shared Application Code**

- **Linux 1**: R/W
- **Linux 2**: R/O
- **Linux 3**: R/W

- **Minidisk A**, **Minidisk B**, **Minidisk C**

- **Fullpack N**

- **TDISK 1**

- **TDISK space**
z/VM LAN and Switch Virtualization

Load Balancer Aggregator / Multiplexer

VM Switch Controller

z/VM Virtual Switch

Real Switch

LACP (Link Aggregation Control Protocol)

OSA

Port 1

Port 2

Port 3

Port 4
1. Send all Linux console output to a single CMS virtual machine

2. Use PROP and REXX to interrogate console messages

3. Issue hypervisor commands on behalf of Linux servers

1. Use the CP Monitor to automatically capture performance and resource consumption data for each Linux server

2. Use Performance Toolkit for VM to process Monitor data
Over 45 years of continual innovation in virtualization

- Refined to support modern business requirements
- Exploiting hardware technology for economical growth

IBM z Systems – a comprehensive and sophisticated suite of virtualization function