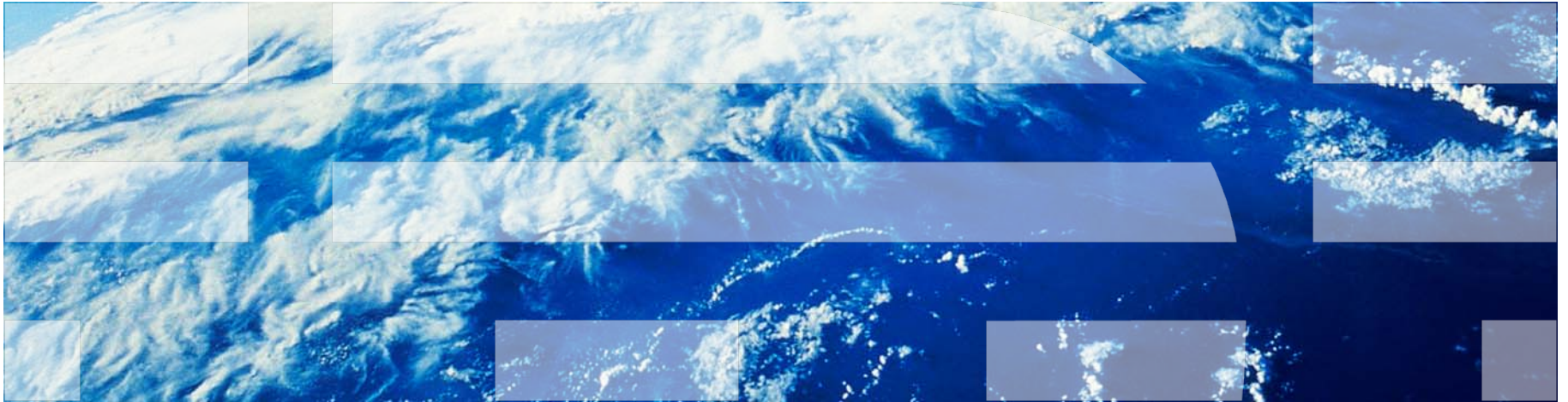


Cloud Computing Workshop 3/2010

Day 1 – Virtualization



March 2010

Agenda – Day 1 - Virtualization

- **Virtualization Technologies**

- Server Virtualization (Hypervisors)
- Storage Virtualization
- Network Virtualization

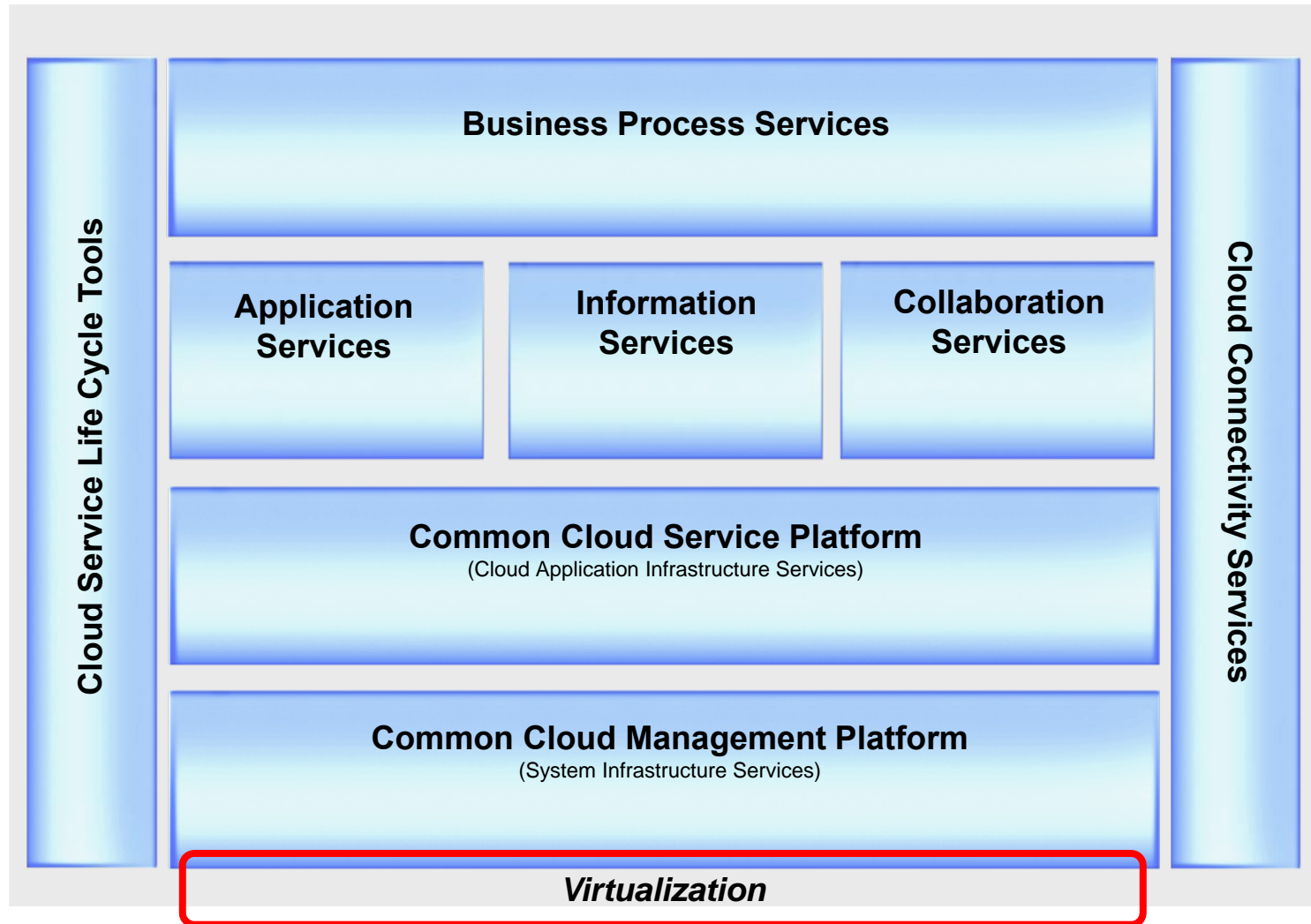
- **Workload / Image Deployment**

- Virtual Machine Image
- Capture and Deploy

- **Role of Standards**

- System Virtualization Partitioning and Clustering (SVPC)
- Open Virtualization Format (OVF)
- Cloud related work at DMTF

Location of Virtualization in the Cloud Stack

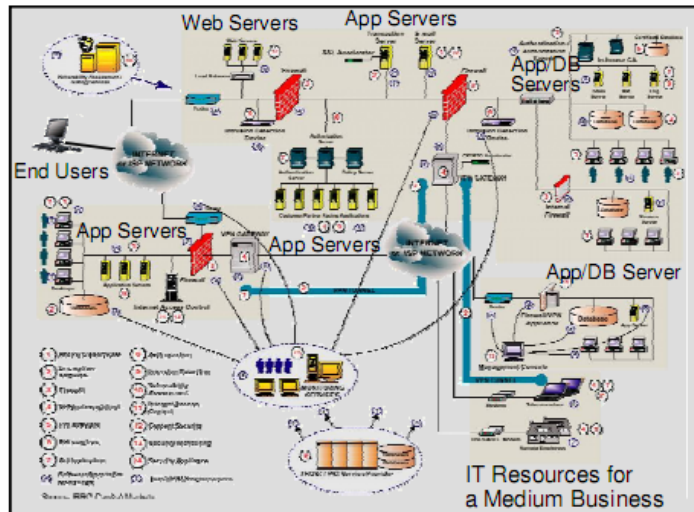


Virtualization Trends and Relevance to Cloud

- **Virtualization is an important building block for cloud environments**
 - Increases flexibility of IT resources
 - Creation of virtual resources
 - Provides for adaptation to varying workloads
- **Trend: Server virtualization will become comprehensive and pervasive, and become integrated with the hardware itself**
 - Hypervisors in the firmware
 - Hypervisors with integrated virtual switches
 - I/O adapters with built-in I/O virtualization
- **Customers prefer to deal with applications and business processes, not with operating systems and hypervisors**
- **Cloud software will need to bridge between:**
 - **IT Resource perspective:** Servers, operating systems, networks, storage (all virtualized)
 - **Customer perspective:** Applications, business processes
- **Trend: Over time, virtualization will be done higher up in the software stack**
 - e.g. at the middleware level, or higher up

Vision: Virtualization will Transform IT

IT Without Virtualization

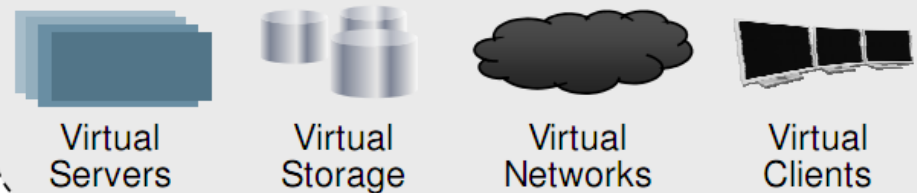


- **Rigid configurations**
- **Fixed resources per server**
- **Low server utilization**
- **Wasted energy and floor space**
- **HW changes impact SW assets**
- **Servers managed individually**

Future Virtualized IT

Virtual Environment

- Virtual resources are easier to deploy, grow, move, ...
- Virtual resources, configurations, and workloads are decoupled and insulated from physical environment

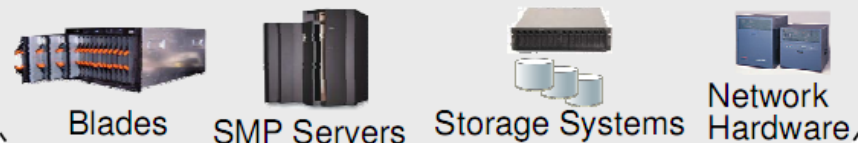


Virtualization

Decouples Virtual and Physical Environments

Physical Environment

- Physical resource changes can be made without impact to running IT workloads
- Improved HW utilization and energy efficiency

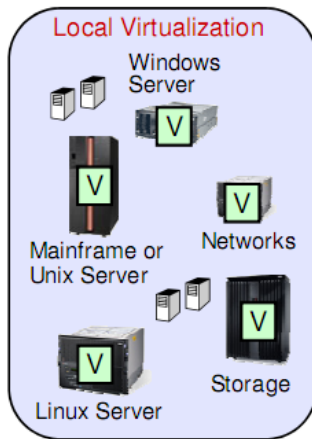


Progressive Stages in IT Virtualization

Virtualization has major benefits
but introduces more complexity

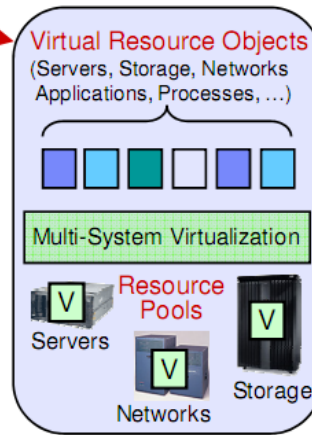


Consolidations



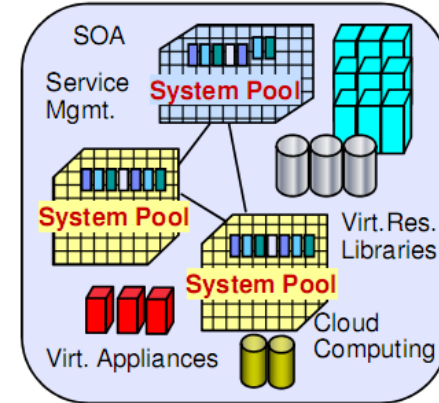
- Better hardware utilization
- Improved IT agility
- Lower power consumption

Objects and Pools



- Better SW investment protection
- Simplified HA solutions
- Improved resource optimization
- Ready-to-run packaged software

Integration for Simplification



- Reduced scale-out complexity
- Integrated autonomic mgmt
- Dynamic energy optimization
- Greater resilience / security

Continuing Advances

- **Exploitation of virtualization**
- **Automation, optimization, ...**

Continuing Advances (I/O, network, resilience, performance, ...)

- **Hardware resources become self-virtualizing**
- **Application virtualization at OS and MW levels**

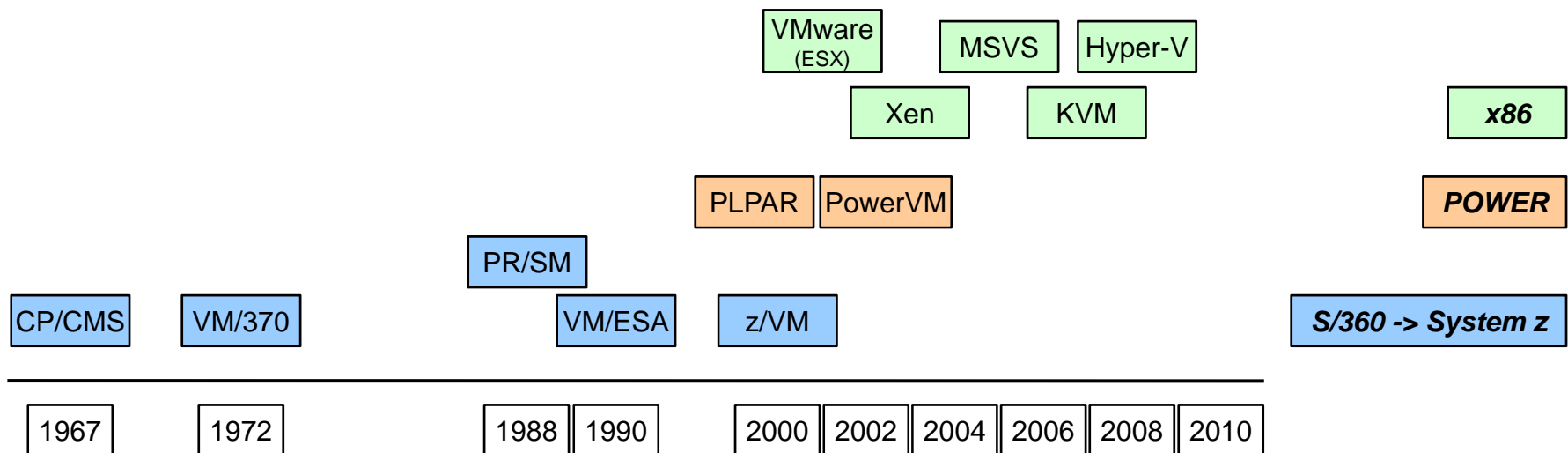
History of Server Virtualization at IBM

▪ 43 years of experience virtualizing our servers

- Virtualization was originally developed to make better use of critical hardware
- IBM runs Linux as a first-class virtualized OS across our entire hardware portfolio
- IBM is still innovating in virtualization

Year of introduction for some hypervisors on IBM servers :

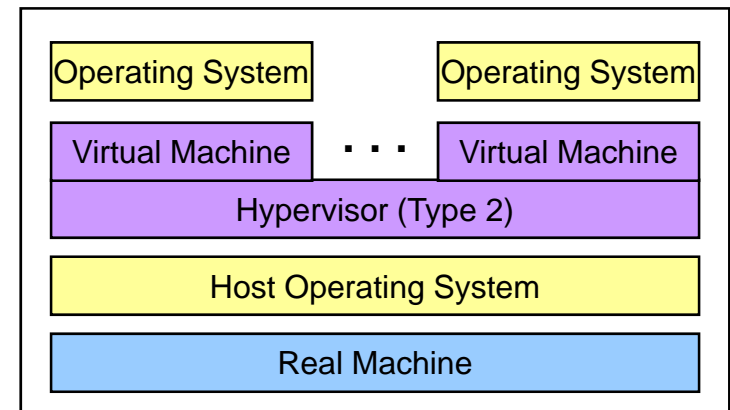
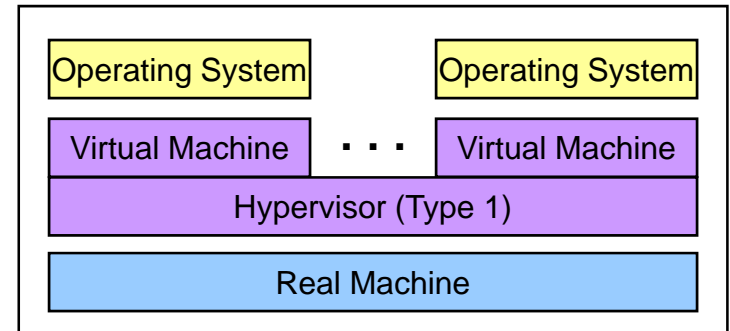
(hypervisor = virtualizer for computer systems)



More hypervisors: http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines

Hypervisor Types

- **Original definition:**
 - Goldberg: "Architectural Principles for Virtual Computer Systems" (1973), [link](#)
- **Type 1:**
 - Hypervisor runs on underlying „real machine“ includes device drivers for real machine
 - „Native“, „Bare metal“
- **Type 2:**
 - Operating system runs on underlying „real machine“ includes device drivers for real machine -> „Host Operating System“
 - Hypervisor runs on / in host operating system:
 - As one user space process per hypervisor
 - As one user space process per virtual machine
 - Hypervisor may require extending the host OS with kernel modules
 - „Hosted“
- **Alternative choices of terminology:**
 - Real / Virtual
 - Host / Guest
 - In any case, it is always relative to the hypervisor -> Hypervisors can be stacked, e.g. z/VM in z LPAR
- **Type 1,2 depends on point of view:**
 - What if Host Operating System and Type 2 Hypervisor are bundled into one SW package ?
 - What if Type 1 hypervisor internally uses a stripped down operating system as its lowest layer ?
 - What if portions of a Type 2 Hypervisor run as kernel modules of the Host Operating System ?
 - Depends on point of view:
 - Internal technical componentry
 - SW packaging
 - General usage of Host OS is possible
 - „Hypervisor ABC is Type X“ -> from what point of view ?



Virtualization Methods

▪ Full Virtualization:

- Also known as: “virtual machines”, “HW emulation”
- Virtual machine provides all interfaces to the guest OS that the real machine provides
- Upside: Guest OS does not need to be aware (i.e. modified) to run virtualized
- Downside: Low level HW specific stuff needs to be virtualized (e.g. HW bus interrupts on x86)

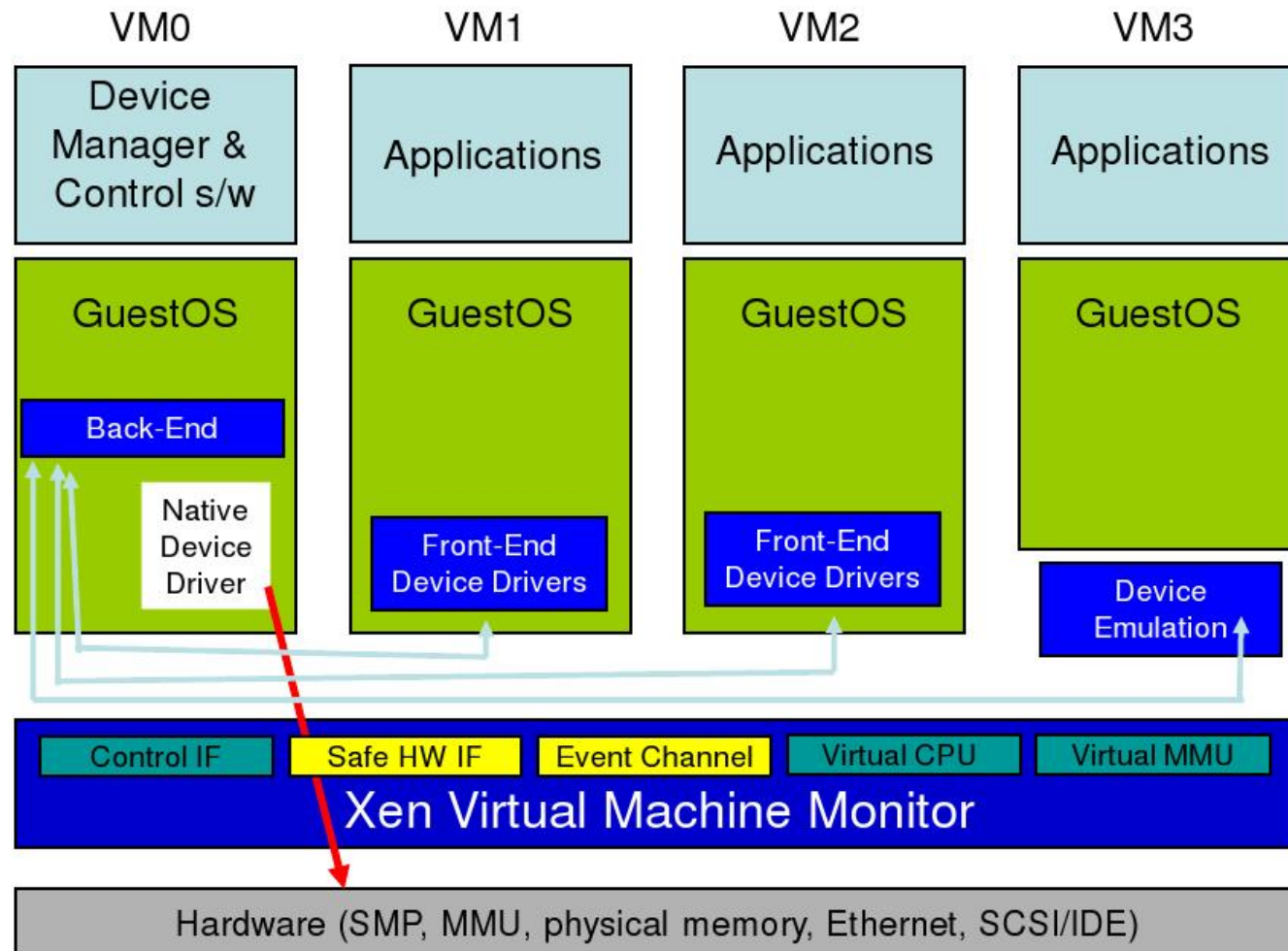
▪ Paravirtualization:

- Virtual machine replaces some real machine interfaces to the guest OS with more efficient (higher level) interfaces
- Multiple paravirtualization interfaces exist
- Downside: Guest OS needs to support the particular paravirtualization interface
- Upside: Better efficiency, compared to full virtualization

▪ OS level virtualization:

- Also known as „containers“, „workload partitions (WPAR)“
- „**System containers**“: Host OS provides virtual machines that can run a guest OS (Type 2 hypervisor)
- „**Application containers**“: Application hosting environment with some virtualization and isolation (to neighbors), no real virtual machine capable of running a guest OS

Example: Xen I/O Paravirtualization (Ian Pratt, Usenix '08)



Virtualized Server Resources - CPU

▪ CPU virtualization:

- Multiple kinds:
 - A fixed portion of a CPU of the real machine is dedicated to a CPU of a virtual machine
 - Corner case: Complete CPU is dedicated
 - One CPU of the real machine powers multiple CPUs of virtual machines

▪ CPU overcommitment:

- Definition: Total number of CPUs visible in all virtual machines is / can be larger than number of CPUs visible in real machine
- This does not increase the overall CPU *capacity* provided to virtual machines, relative to what is provided by the real machine
 - In fact, there is a CPU virtualization overhead

▪ Dynamic CPU assignment:

- Ability to dynamically adjust CPU capacity assigned to virtual machines to needs
 - Driven by CPU need of guest OS
 - Governed by business importance („workload management“)
- Important for CPU needs that vary over time
- „Hypervisor ABC supports CPU overcommitment“
 - > How dynamic is the CPU assignment and how can it be managed ?

Virtualized Server Resources - Memory

▪ Memory virtualization:

- Multiple kinds:
 - Ranges of memory of the real machine are dedicated to a virtual machine
May be mapped to “clean” addresses (e.g. starting at address 0, as seen in virtual machine)
 - Memory of the real machine is dynamically assigned to virtual machines
Usually with swapping/paging to handle memory overcommitment

▪ Memory overcommitment:

- Definition: Total memory addressable in all virtual machines can be larger than memory addressable in real machine
- Requires a swapping / paging mechanism of some sort
- This increases the overall memory capacity provided by the real machine
 - at the price of swap/page penalties
- Important for memory needs that vary over time
- Support for virtual address translation or a “virtual memory manager” does not necessarily mean the memory can be overcommitted
- “Hypervisor ABC supports memory virtualization”
 - > Does it support memory overcommitment, how much and how efficient ?

Small Hypervisor Typology

Hypervisor	Hyperv. Type (general Host OS usage)	Virtualiz. Method	CPU overcomm.	Memory overcomm.	HW Platform
IBM PR/SM (LPAR)	Type 1	Full ⁽⁴⁾	yes	no	IBM System z
IBM z/VM	Type 1	Full ⁽⁴⁾	yes	yes	IBM System z
IBM PowerVM (LPAR)	Type 1	Full	yes	no	IBM POWER (System i and System p)
VMware ESX Server	Type 1	Full+Para	yes	yes	x86
VMware Server (formerly „GSX Server“)	Type 2	Full	yes	yes	x86
Microsoft Hyper-V	Type 1 ⁽¹⁾	Full+Para	yes	no	x86
Microsoft Virtual Server	Type 2	Full	yes	yes	x86
Xen	Type 1 ⁽²⁾	Full+Para	yes	no	x86
KVM	Type 2 ⁽³⁾	Full+Para	yes	yes	x86, ...

(1) Hyper-V sometimes qualified as both Type-1 and Type-2, due to two options of SW packaging

(2) Xen sometimes qualified as Type 2, due to internal componentry

(3) KVM sometimes qualified as Type 1, due to presence of KVM kernel modules that extend Linux to get hypervisor abilities

(4) z/VM and PR/SM sometimes qualified as Para due to hypervisor specific DIAG instructions

Virtualized Server Resources – I/O

- **Multiple approaches:**
 - Adapter port sharing between virtual systems
 - Adapter virtualization
 - Integrated virtual switches
 - PCI: Single and multi root I/O virtualization
- **Most network adapters will have an integrated MAC-level switch**
- **Multiple DMA queues, one for each virtual function**
- **Opportunity to add additional functions to the device**
 - Most I/O virtualization devices are programmable
 - Many run an embedded Linux kernel or system executive
 - IEEE VLAN support in the adapter
 - Service level and workload management
- **Operating systems and hypervisors will both exploit I/O virtualization devices**

„Pure“ Virtual Resources

- **Resources that exist without an underlying resource of the same kind in the real machine**
- **Virtual switch**
 - Pure virtual resource for the hypervisor
 - Not directly visible as a resource by the virtual machine
 - Virtual machine connects to virtual switch through a virtual network port
 - e.g. z/VM VSWITCH, Guest LAN
- **Virtual network port**
 - Also known as “Virtual NIC” or “vNIC” (NIC = Network Interface Card)
 - For connecting to networks via network adapters or virtual switches
- **Virtual disk/drive**
 - E.g. backed with memory of the real machine, or file of the real machine

Virtualized Network Resources

- **Network connectivity elements**
 - Adapters / Ports
 - Switches

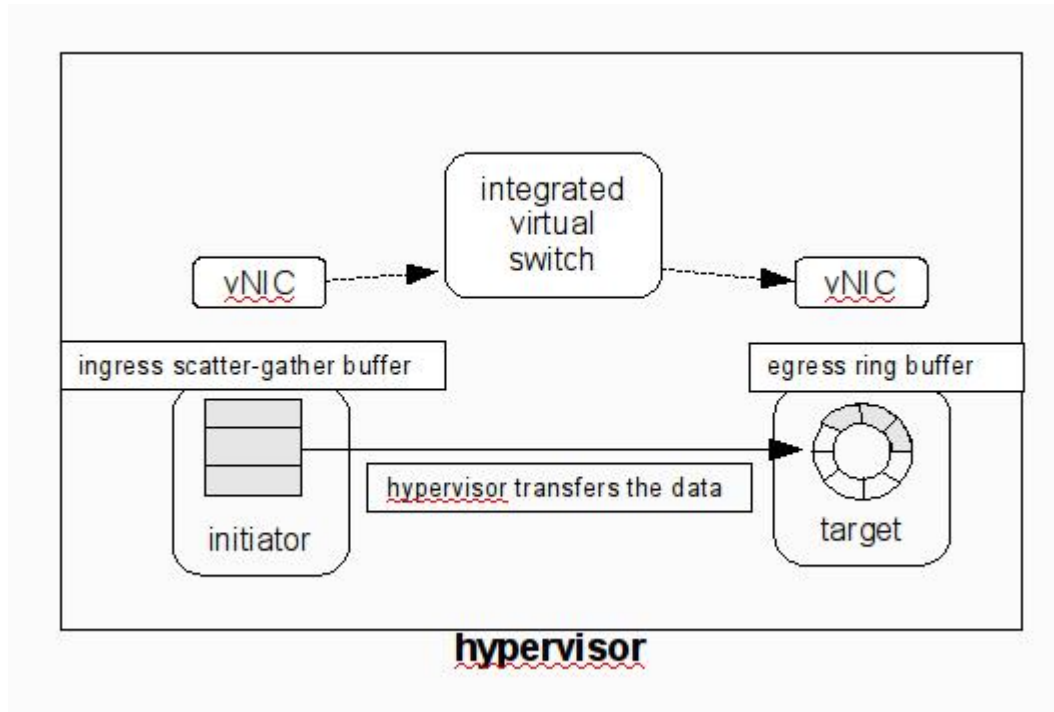
- **Network**
 - IEEE VLAN
 - VPNs

- **Lots of flexibility, cloud tools need to simplify based on:**
 - Application needs (example: Needs Layer 2)
 - Best practices
 - Customer guidelines

- **Trend to QoS**
 - Bandwidth
 - Error rate
 - ...

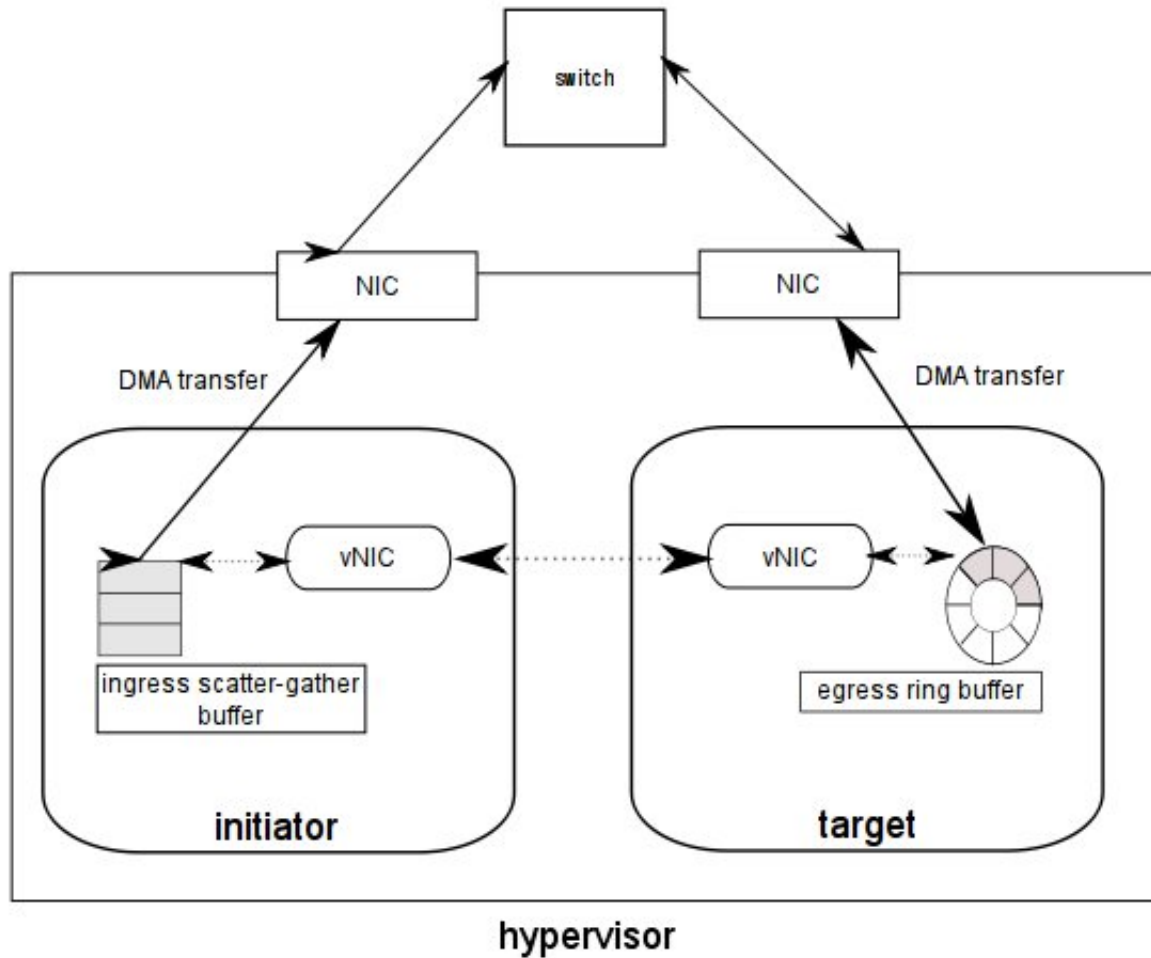
-> more details in “Network Considerations” session

Integrated Virtual Switching



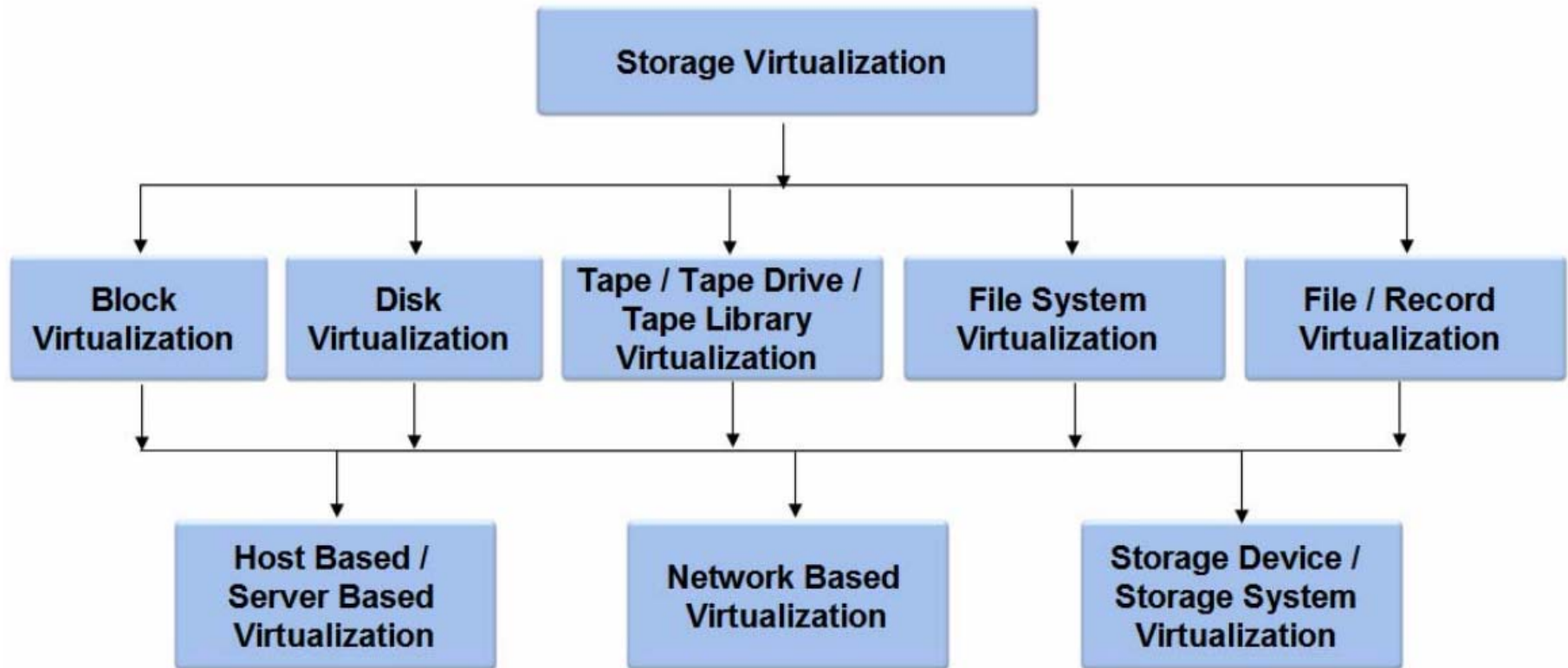
- All hypervisors have an integrated virtual switch
- VMware's definition of a formal switch module and API has changed the way virtual switching is viewed by hypervisor developers
- Expectation moving forward is that virtual and physical switches will support same management commands and syntax
- Concept of distributed virtual switching

Integrated Physical Switching



- Let I/O virtualization hardware do the switching, even within a single physical host
- Offload host CPU
- Needs a good general-purpose (RAM to RAM) DMA capability
- Needs a better mechanism for direct processing of device interrupts within guest

Storage Virtualization



Source: Storage Networking Industry Association, SNIA

-> more details in „Storage in the Cloud“ session

Agenda – Day 1 - Virtualization

- **Virtualization Technologies**

- Server Virtualization (Hypervisors)
- Storage Virtualization
- Network Virtualization

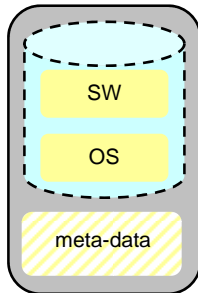
- **Workload / Image Deployment**

- Virtual Machine Image
- Capture and Deploy

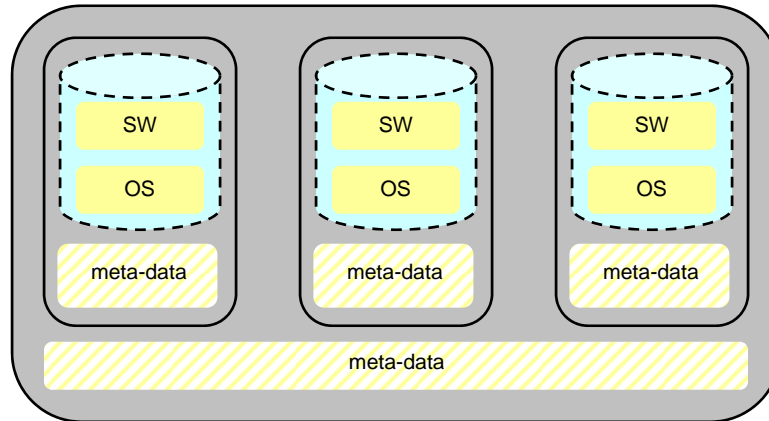
- **Role of Standards**

- System Virtualization Partitioning and Clustering (SVPC)
- Open Virtualization Format (OVF)
- Cloud related work at DMTF

What is a Virtual Machine Image?



Virtual Machine Image



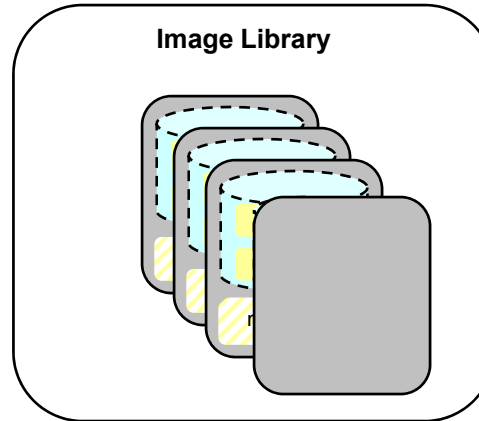
Virtual Machine Image Composition

- **A Virtual Machine Image is a self-contained package containing....**
 - **Meta-data describing the required server resources needed to run the image**
 - Number of CPUs (dedicated vs. shared)
 - Memory requirements
 - IO and network requirements
 - **Meta-data describing goals and constraints for the image**
 - Performance and availability goals
 - Placement constraints (i.e. security isolation)
 - **Meta-data describing configuration variables**
 - OS Configuration parameter – IP Address, etc.
 - Application Configuration parameters
 - **One or more disk images containing OS, middleware and other application software**
- **May be composition of multiple related virtual machine images**
 - Virtual machine images making up a multi-tiered distributed application workload.
 - Includes additional meta-data scoped to the entire composition.

Deploying and Capturing Virtual Machine Images

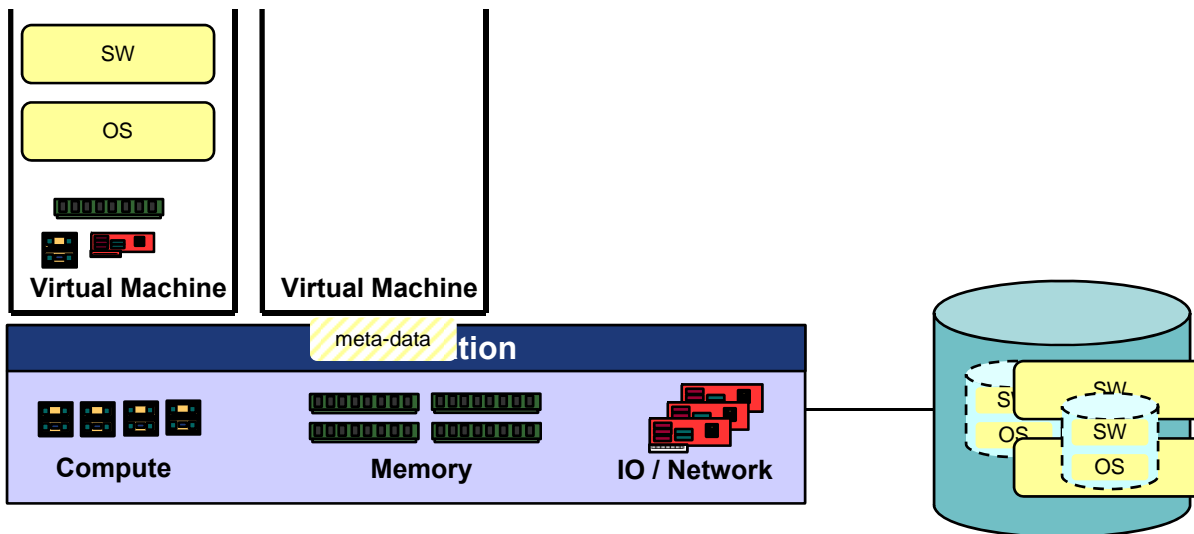
Deploy a Virtual Machine Image

- The Image meta-data is used to create VM container, allocating the required platform resources.
- The bootable image is copied and/or made accessible to the selected host system.
- The virtual machine is started from the bootable disk image.



Capture a Virtual Machine Image

- The bootable disk image is copied into the image library.
- Image meta-data describing the Virtual Machine container is captured and included as part of the image.



Agenda – Day 1 - Virtualization

- **Virtualization Technologies**

- Server Virtualization (Hypervisors)
- Storage Virtualization
- Network Virtualization

- **Workload / Image Deployment**

- Virtual Machine Image
- Capture and Deploy

- **Role of Standards**

- Open Virtualization Format (OVF)
- System Virtualization Partitioning and Clustering (SVPC)
- Cloud related work at DMTF

OVF Standard

- **Defined by SVPC WG at DMTF**

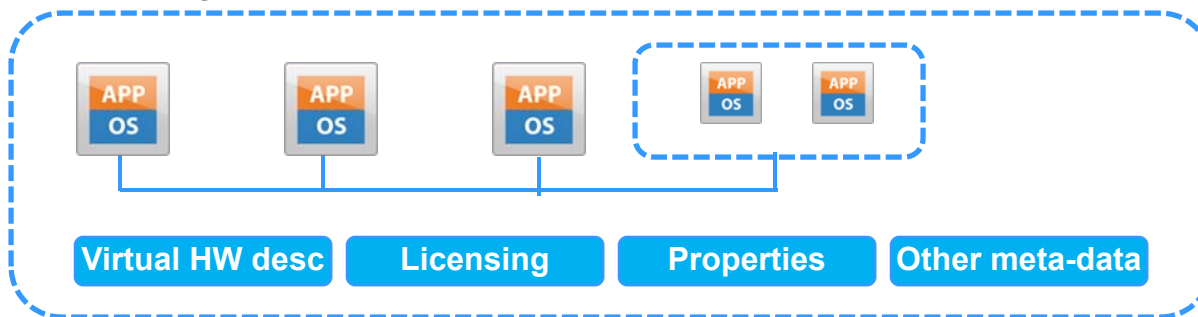
- SVPC = System Virtualization, Partitioning and Clustering
- DMTF = Distributed Management Task Force, www.dmtf.org
 - A standards development organisation
 - >3000 individual members from >100 companies across the industry
- Major virtualization players are participating in SVPC WG

- **OVF Standard**

- Describes a format for virtual machine images
 - “OVF Descriptor”, “OVF Envelope”
 - Extensible
 - Vendor neutral
- Describes a format for customization information used during deployment
 - “OVF Environment”
- OVF 1.1 has reached DMTF Standard status:
 - DSP0243 Open Virtualization Format
 - DSP8023 XML Schema for OVF Envelope
 - DSP8027 XML Schema for OVF Environment
- DMTF Standards on http://www.dmtf.org/standards/published_documents/

Open Virtualization Format

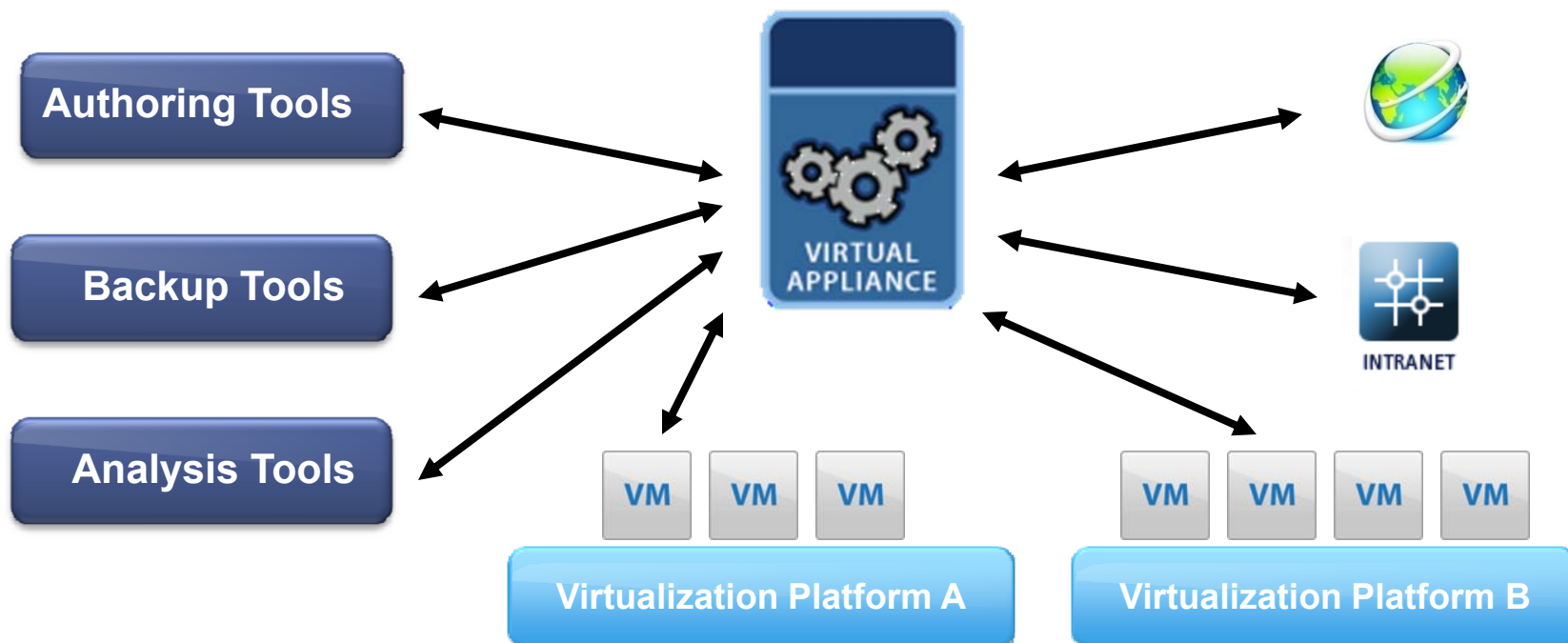
- A **distribution format** for virtual appliances
 - Provides a complete description of a single VM or complex multi-VM software solution package



- Vendor and platform independent
 - Interoperable across virtualization platforms
- Extensible
 - Facilitate value-added features by DMTF or third-parties
- DMTF specification
 - Version 0.9 technology submission September 2007
 - Version 1.0 preliminary September 2008
 - Version 1.0 final February 2009
 - Version 1.1 in approval process, work-in-progress available
 - Version 2.0 work has started

Large Scale VM distribution

- Provisioning is a critical function
 - How do different products and vendors exchange VMs?
 - How do different products interoperate on meta-data?
 - Suitability of data for distribution?

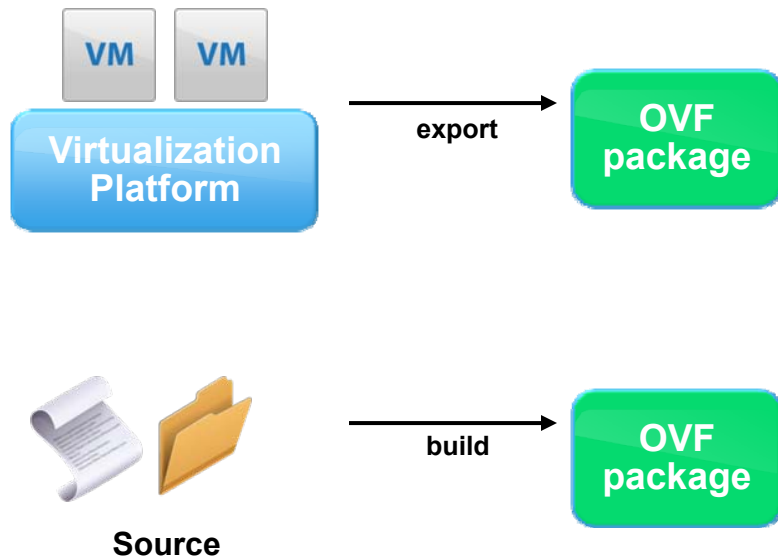


OVF Workflow

Author → Distribute → Deploy

- **Author Role:** To create a complete software package that makes life simple for the deployer
- **Deployer Role:** Deploy and customize the package for a particular instance
- Author must express and deployer must understand:
 - Compatibility and optimization requirements
 - Resource requirements
 - Entitlements requirements
 - Customization options
 - Package integrity

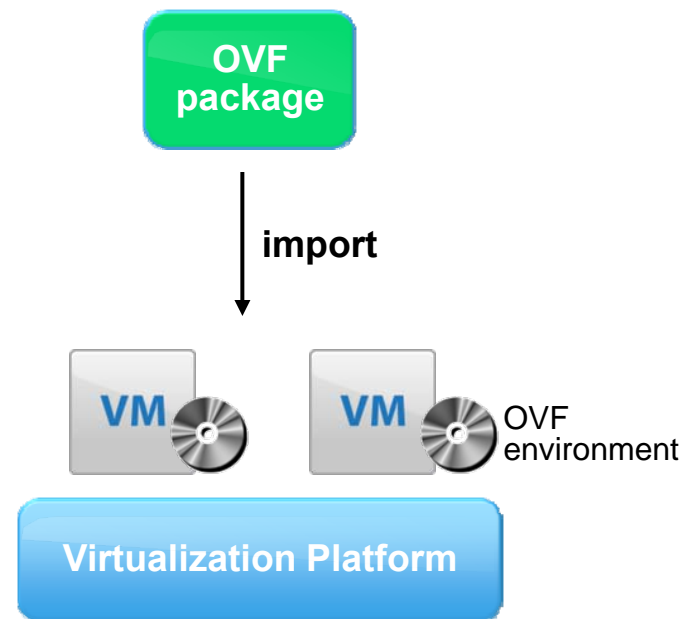
OVF Authoring



- A virtual machine can be exported into OVF or built from source
- The author can describe “portable hardware” for maximum interoperability or optimum performance
- The virtual machine images can be compressed to facilitate distribution
- The author can sign the package to guarantee its integrity

OVF Deployment

- Validation
 - Verify licensing, security, integrity
 - Resource requirements and placement
 - Application properties (such as IP addresses and passwords)
- Conversion
 - Convert virtual disks to run-time format
- Installation
 - Provide runtime environment for application (customization and localization)



The OVF Specification

- A Packaging Format
 - How to bundle files and do digital signing
- OVF Descriptor
 - An XML file describing the software in an OVF package
 - Organized as an envelope with an extensible set of sections
- Core Sections
 - 10 core sections for describing virtual hardware, EULA, Product information, etc.
- OVF Environment
 - An XML document available to the guest software which enables it to adapt to the deployment environment

Packaging – Guest Software

- Software is installed in one or more virtual disks
- Any public specified virtual disk format is supported
- For example,
 - Created using a virtual appliance development kit
 - Exported from existing virtualization platform setup



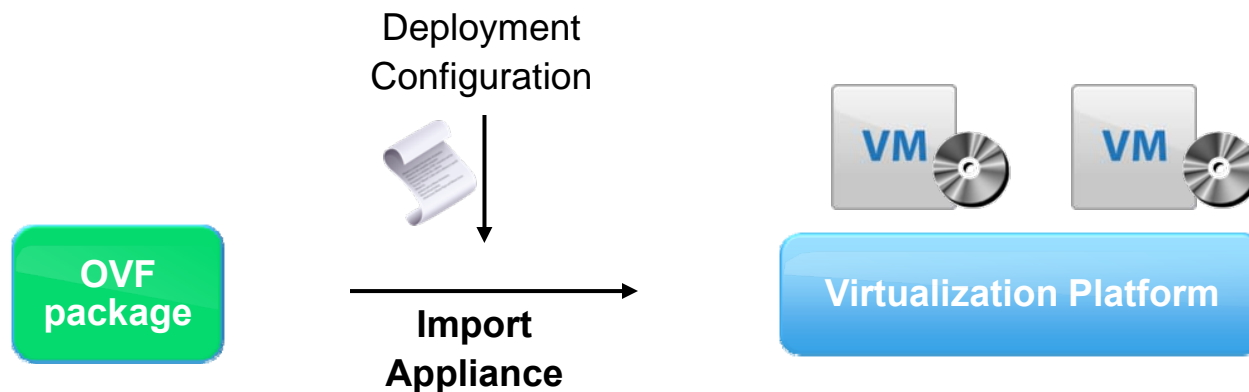
web.xxx



db.xxx

OVF environment

- How do the key/value pairs get delivered to the guest software
- How does a multi-tiered application know the IP addresses of its peers?
- Common **OVF Environment** is available to each VM via dynamically generated ISO image



Extensibility

- Extensibility model
 - Inspired by Open Content model in XML Schema 1.1, but described using Schema 1.0
 - Enables forwards and backwards compatibility
- Extension points
 - Allows new sections to be defined
 - Allows defined elements to extended with additional content at the end
 - Allows additional attributes on existing types
 - All extensions must be in non-OVF namespaces
- **ovf:required** attribute
 - Determines whether a consumer is required to interpret an extension

Conformance

- Three defined conformance levels
 - **Conformance level 1**
 - Uses only OVF sections and elements that are described in the specification
 - **Conformance level 2**
 - Uses extensions not defined in the OVF specification but all extensions are optional (`ovf:required="false"`)
 - **Conformance level 3**
 - Uses extensions not defined in the OVF specification that are required (`ovf:required="true"`), where all extensions are defined with publicly available and unencumbered XML Schema. Documentation may be inclusive in the schema or available as a separate document.

OVF 1.1 Specification

- Minor release
 - Clarification of manifest formats
 - Clarification of certificate formats
 - Clarification of OVA package formats
 - Clarification of “iso” transport format and access
 - Optional support for UI hiding for password properties
 - Declared ovf:transport attribute as global in schemas
 - Minor errata

OVF 2.0 Specification

- Expect technical work to complete by mid-2010
- Cloud input may result in additional specifications being developed
- Initial features under consideration
 - Device boot order configuration
 - Express BIOS boot order of virtual system
 - Contracts/Policy
 - Ability to abstractly specify what a virtual system expects/requires in the environment to be deployed and exhibit correct runtime behavior
 - Example: Dependency on firewall with specific configuration
 - Parameter extensions
 - Example: Storage configuration, network properties
 - Embedded composition of multiple OVF packages
 - Possible today, but cumbersome and problematic with signed packages
 - OVF Launcher/Deployer/Modifier requirements
 - Part of OVF spec or a separate spec?
 - Need to clarify the boundary between deployment and operational mgmt
 - Library of OVF packages

OVF Summary

- **Complete description of Virtual Machine**
Supports multiple disk formats and virtual hardware platforms.
- **Supports single VM and multi-VM configurations**
Models multi-tier services consisting of multiple VMs, including composed services.
- **Optimized for distribution**
Supports verification, signing, versioning, and handling of licensing terms.
- **Optimized for simple, automated user experience**
Supports validation of entire package, provides descriptive information to user, and includes application level meta-data.
- **Vendor and platform independent**
Does not rely on specific host platform, virtualization platform, or guest operating system properties.

OVF Summary (2)

- **Extensible**

Allows vendor specific meta-data and provides an extensible hardware description model supporting future advances in virtualization technology.

- **Localizable**

Supports user visible descriptions in multiple locales.

- **Open standard**

A collaboration of key vendors

DMTF System Virtualization Management Standards

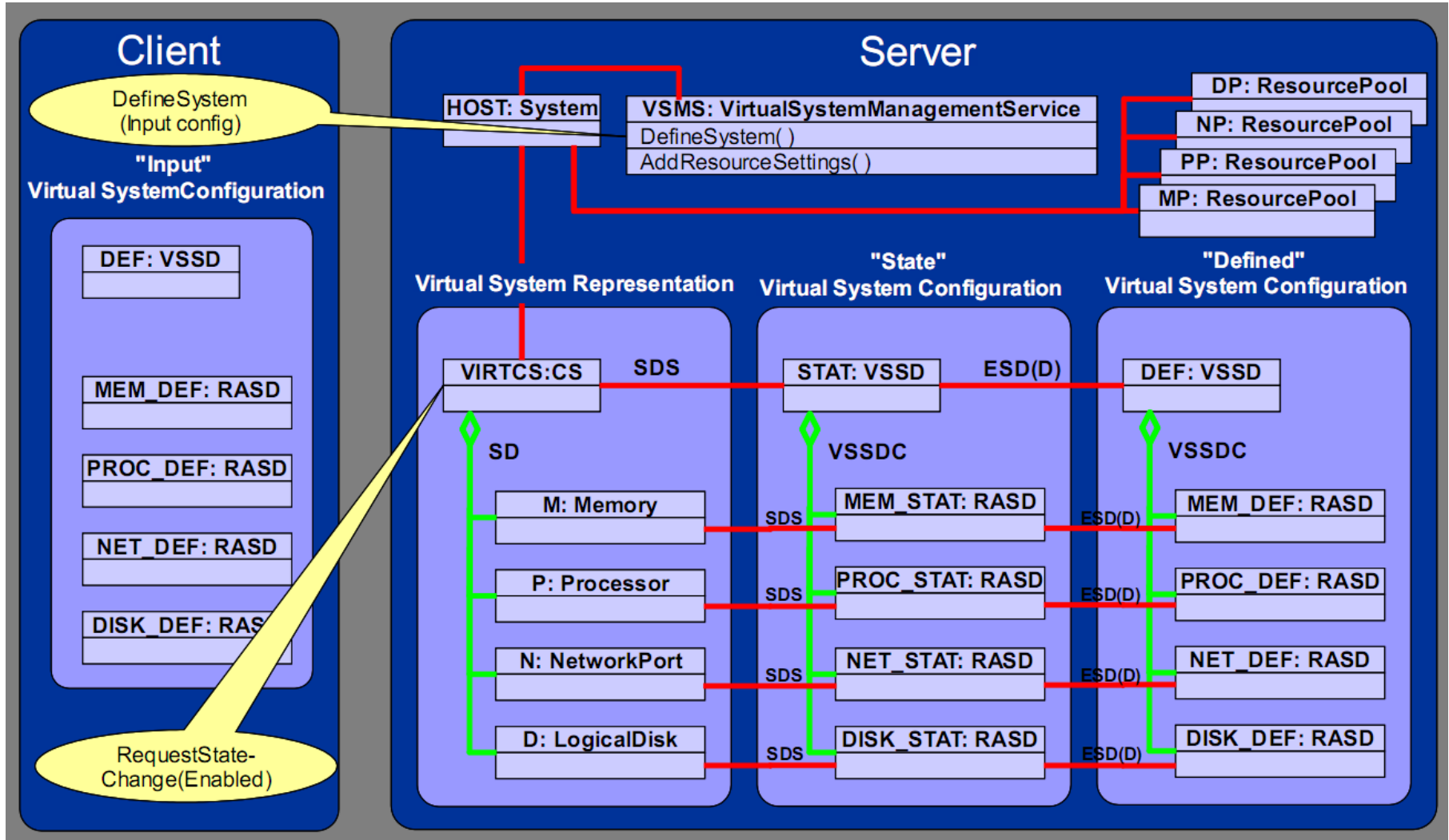
- **Defined by SVPC WG at DMTF**

- SVPC = System Virtualization, Partitioning and Clustering
- DMTF = Distributed Management Task Force, www.dmtf.org
 - A standards development organisation
 - >3000 individual members from >100 companies across the industry
- Major virtualization players are participating in SVPC WG

- **“SVPC” standards**

- A set of *management profiles*, based on CIM (Common Information Model)
- SVPC 1.0 covers management of hypervisors and virtual systems
 - Inspection / discovery of hypervisors and virtual systems
 - State management of virtual systems
 - Hypervisor resource pool inspection and management
 - Creation & destruction of virtual systems
 - Addition, removal and modification of virtual resources
 - Virtualization specific functionality like snapshots and live guest migration
 - ...
- Most SVPC 1.0 standards have reached DMTF Standard status:
 - DSP1042 System Virtualization Profile
 - DSP1057 Virtual System Profile
 - DSP1081 Virtual System Migration Profile
 - DSP1044 Processor Resource Virtualization Profile
 - DSP1045 Memory Resource Virtualization Profile
 - DSP1050 Ethernet Resource Virtualization Profile
 - DSP1097 Virtual Ethernet Switch Profile
 - DSP1047 Storage Resource Virtualization Profile
 - DSP1049 Storage Adapter Resource Virtualization Profile
 - DSP1059 Generic Device Resource Virtualization Profile
 - ...
- DMTF Standards on http://www.dmtf.org/standards/published_documents/
- Work in Progress on <http://www.dmtf.org/standards/wip/>

SVPC Standards Implementation on a Server



Connection between OVF and SVPC

- **SVPC standards describe**

- Virtual systems: “Virtual System Setting Data” (VSSD)
- Virtual resources: “Resource Allocation Setting Data” (RASD)
- Management methods such as DefineSystem() and AddResourceSettings() get VSSD and RASD objects as input, that describe the system and its resources

- **OVF standard**

- Use VSSD and RASD format to describe requirements on target virtual system

-> Synergy effect: Target platforms supporting the SVPC standard can easily deploy OVF images

Cloud related work in DMTF

- **DMTF has started a “Cloud Incubator”**

- Leadership Board:

- AMD, CA, Cisco, Citrix, EMC, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Novell, Rackspace, RedHat, Savvis, SunGard, Sun Microsystems, and VMware

- Scope of the work:

- Charter: “The scope of this activity is focused on mainly cloud resource management aspects of Infrastructure as a Service (IaaS) with some work touching on Platform as a Service (PaaS) including SLAs, QoS, utilization, provisioning and accounting and billing.”

- Published “Interoperable Clouds” White Paper in 11/2009

- http://www.dmtf.org/about/cloud-incubator/DSP_IS0101_1.0.0.pdf

- More information on:

- <http://www.dmtf.org/about/cloud-incubator/>

Summary – Day 1 - Virtualization

▪ Virtualization Technologies

- Server Virtualization (Hypervisors)
- Storage Virtualization
- Network Virtualization

▪ Workload / Image Deployment

- Virtual Machine Image
- Capture and Deploy

▪ Role of Standards

- Open Virtualization Format (OVF)
- System Virtualization Partitioning and Clustering (SVPC)
- Cloud related work at DMTF