Device Virtualization Architecture

Jake Oshins
Architect
Windows Virtualization
Microsoft Corporation
Goals

Participants will leave with an understanding of:

- How Microsoft intends to enable efficient I/O virtualization
- How others’ I/O solutions interact with Microsoft’s virtualization systems
- Which I/O virtualization strategies will be available with Windows Server virtualization and which must wait
Agenda

- General strategies for I/O virtualization
- Technical overview of Virtual Device Framework
- Technical overview of VMBus
Device Emulation

- Virtual machine "sees" real hardware devices
- Each access to the "device" involves an intercept, sent to the parent virtual machine
  - Performance is sub-optimal
- Compatibility with existing software can be perfect
- Microsoft provides emulations
  - The hardware that is emulated is from ~1997, providing in-box compatibility with old OSes
- Requires a "monitor" partition that contains software for emulating the devices
- Physical devices can be shared among multiple guests
I/O Enlightenment

- Uses abstract protocols to describe I/O
- Useful protocols already exist
  - SCSI, iSCSI
  - RNDIS
  - RDP
- New device stack implementations in the secondary guests can be written that use these abstract protocols
- Protocol servers exist in a primary guest (parent), which is the partition that controls the physical devices
- Multiple secondary guests can share the services of a single hardware device
- Doesn’t require an emulator
- Doesn’t require a monitor partition
Device Assignment

- Guest OSes control their devices directly
  - Parent OS gives up control of these devices
- Ownership of a device is exclusive
- Performance can match that of a non-virtualized machine
- Interdependence of partitions can be minimized
- Strong isolation of partitions can be achieved
Windows Virtualization Will Provide

- Device emulation
  - Provides migration path for Microsoft Virtual Server users
  - ~1997 era virtual motherboard
  - Good for compatibility with old OSes

- I/O enlightenment
  - Storage
  - Networking
  - Video
  - USB
Agenda

- General strategies for I/O virtualization
- Overview of Virtual Device Framework
- Technical overview of VMBus
Virtual Device (VDev)
- A software module that provides a point of configuration and control over an I/O path for a partition

Virtualization Service Provider (VSP)
- A server component (in a parent or other partition) that handles I/O requests
  - Can pass I/O requests on to native services like a file system
  - Can pass I/O requests directly to physical devices
  - Can be in either kernel- or user-mode

Virtualization Service Consumer (VSC)
- A client component (in a child partition) which serves as the bottom of an I/O stack within that partition
  - Sends requests to a VSP

VMBus
- A system for sending requests and data between virtual machines
Virtual Devices (VDevs)

- Come in two varieties
  - Core: Device emulators
    - Written by Microsoft
  - Plug-in: Enlightened I/O
    - Written by Microsoft and industry

- Management is through WMI
- Packaged as COM objects
  - Run within the VM Worker Process
- Often work in conjunction with a VSP
Virtualization Service Providers (VSPs)

- Communicate with a VDev for configuration and state management
- Can exist in user- or kernel-mode
  - COM object
  - Service
  - Driver
- Use VMBus to communicate with a VSC in the child partition
Example VSP/VSC Design
Agenda

- General strategies for I/O virtualization
- Technical overview of Virtual Device Framework
- Technical overview of VMBus
VMBus – What Is It?

- A protocol for transferring data through a ring buffer
  - A means of mapping a ring buffer into multiple partitions
  - A definition for the format of the ring buffer
  - A means of signaling that a ring buffer has gone non-empty
- A protocol for offering/discovering services
- A protocol for managing guest physical addresses
- A protocol for enumerating WDM device objects that represent a data channel
- A bus driver which implements all of those protocols
- A data transfer library which can be linked into a user-mode service or application
- A data transfer library which can be linked into a kernel-mode driver
**VMBus Definitions**

- **Endpoint**
  - A module that reads or writes data through VMBus

- **Channel**
  - Two endpoints – one server, one client
  - Two ring buffers

- **Transfer Page**
  - Pre-allocated page of memory that is mapped into both endpoints’ partitions
  - Not part of a ring buffer
  - Used as a target for DMA or for other operations that may take a “long” time to complete
**VMBus Definitions**

- **Guest Physical Address Descriptor List (GPADL)**
  - Memory descriptor list that can be passed to another partition
  - Allows a device to do DMA to or from a child partition directly

- **Pipe**
  - A default channel protocol that allows a client to use ReadFile or WriteFile to send data between partitions
  - Serves as the basis for cross-partition Remote Procedure Call
How Is Data Moved Between Partitions?

- Commands are placed in ring buffers
- Small data is placed in ring buffers
- Larger data is placed in pre-arranged pages shared between partitions
  - Described by commands in ring buffers
- Largest data is mapped into another partition without copying
  - Described by GPADLs placed in ring buffers
Hypervisor Involvement

- **When is it necessary?**
  - Channel setup
  - Signaling another partition
    - Modeled as a hardware interrupt

- **When is it not necessary?**
  - When placing packets in a ring buffer
  - When removing packets from a ring buffer
  - When reading or writing Transfer Pages
  - When translating guest memory maps
Guest Physical Address Space
GPADLs

- Allow transactions to refer to guest buffers
  - No data copying required
- Built within the Virtualization Stack in the parent partition
- Allows I/O to be handled without switching into and out of the hypervisor
- Allows child partitions’ VSCs to use their own physical addresses in requests to VSPs
- Allows VSPs easy access to translations
  - Particularly if VSP is a driver in kernel-mode
  - Typical transaction can involve no hypercalls
Request Packet Structure

Parent Partition

Child Partition

1

2

3 Application Buffers

Header GPADL – Describes Application Buffers Protocol – Device Specific
What Does Traffic Look Like?

- VMBus underlying protocol is very simple
  - Packets are sent asynchronously
    - Primitives exist to allow synchronization
  - Packets have very little structure
    - Packet may reference Transfer Pages
    - Packet may reference a GPADL
- Other protocols must be defined by the users of the channel
Request Packet Flow

Parent Partition

VSP

Child Partition

VSC
Request Packet Flow

Parent Partition

VSP

Interrupt through Hypervisor

Child Partition

VSC
Request Packet Flow

Parent Partition

VSP

Child Partition

VSC
Data Flow

Parent Partition

VSP

Child Partition

VSC
Interrupt Management

- Can be sent between partitions to signal VSP or VSC code to start running
  - Avoids software polling
- Cost of an interrupt is a hypercall and maybe a partition context switch
- Only necessary when VSP/VSC wouldn’t already be running
  - When ring buffer was previously empty
  - When ring buffer was previously full
- Multiple channels’ interrupts can be coalesced
- VMBus can track latency requirements
  - Allows requests to be batched
Bus Driver

- VMBus acts as a bus driver
- It can form the bottom of a device stack
- VSCs can be instantiated on top of VMBus

(Names of components not finalized)
Call To Action

- Please attend the following session on Virtual Networking and Storage
- Participate in future Windows Server virtualization Beta programs