UKCMG Conference, 16th – 17th May 2010

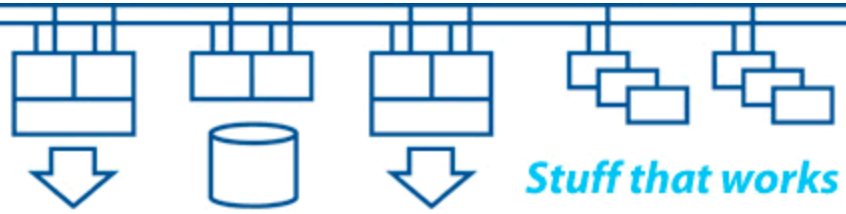
I/O in a virtual environment

A step-by-step journey through the system from the application to the outside world

Colin Butcher

www.xdelta.co.uk

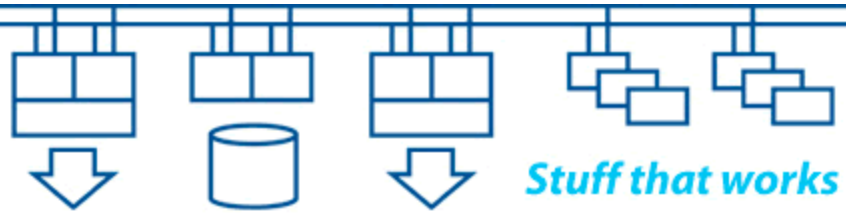
ukcmg THE INDEPENDENT IT USER FORUM



"When I use a word, it means just what I choose it to mean - neither more nor less"

Humpty Dumpty, Through the Looking Glass,
by Lewis Carroll

Virtual = .NOT. Physical



Part 1:

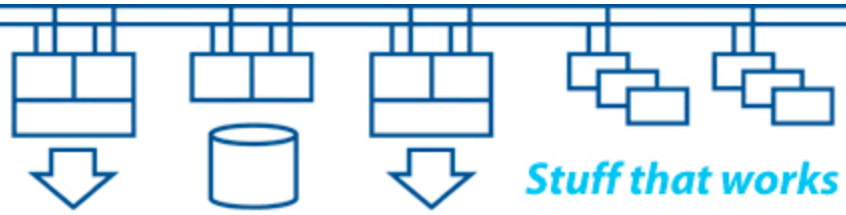
Some fundamental principles:

- Abstraction layers
- Performance characteristics
- Parallelism
- Scalability

"All problems in computing can be solved by introducing another layer of abstraction"

"Most problems in computing are caused by too many layers of complexity"

We need to strike a balance that is appropriate for the kinds of systems we're building



- Bandwidth – determines throughput
  - It's not just "speed", it's throughput in terms of "units of stuff per second"
- Latency – determines response time
  - Determines how much "stuff" is in transit through the system at any given instant
  - "Stuff in transit" is the data at risk if there is a failure
- Jitter ("div latency" or variation of latency with respect to time) – determines predictability of response
  - Understanding jitter is important for establishing timeout values
  - Latency fluctuations can cause system failures under peak load

- Understand how your workload could break down into parallel streams of execution:
  - Some will be capable of being split into many elements with little interaction
  - Some will require very high levels of interconnectivity and interaction
  - Some will require high-throughput single-stream processing

- Increasing the capacity of the overall system:
  - "Scale up" or "vertical scaling" refers to increasing capacity by adding more resources to a machine or buying a bigger machine (CPU count, memory, I/O adapters, etc.)
  - "Scale out" or "horizontal scaling" refers to increasing capacity by adding more machines (eg: blades)
  - It depends on how your workloads break down into parallel streams of execution and on what level of availability you need to achieve

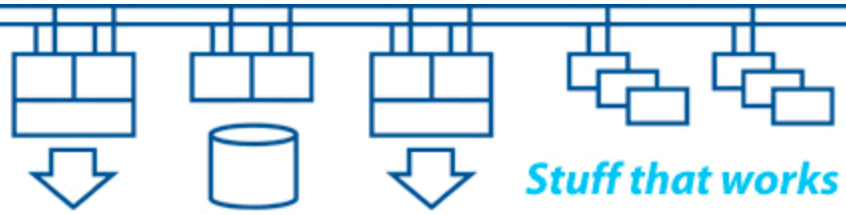
- Capacity in a virtual environment:
  - To the "hypervisor", each and every virtual machine is a workload needing physical hardware resources
  - Within a virtual machine, each application (and the operating system overhead) is a workload
  - What level of interaction is there between the virtual machines that run your applications ?
  - What happens when your host hardware runs out of resources or when virtual machines move to another hardware host ?

Part 2:

Familiar virtualisation techniques:

- Data networks: VLANS
- Storage arrays: Virtual discs (LUNs)
- Storage subsystems: SAN zones; Storage arrays

- VLANs are used to segment a data network:
  - Implemented within core network switches
  - VLAN tagging of packets (802.1Q)
  - Extended LANs spanning multiple sites
  - "Converged ethernet" as a common backbone in the data centre

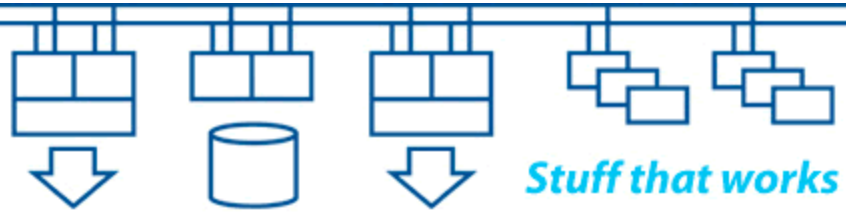


- Storage arrays:
  - The array controller "hides" the behaviour of the physical discs and ensures that writes occur to more than one disc
  - The array controller caches much of the data, using battery backed write-back cache techniques
  - The only real performance issues are bandwidth to and from the array controller pair and contention for access to the storage array

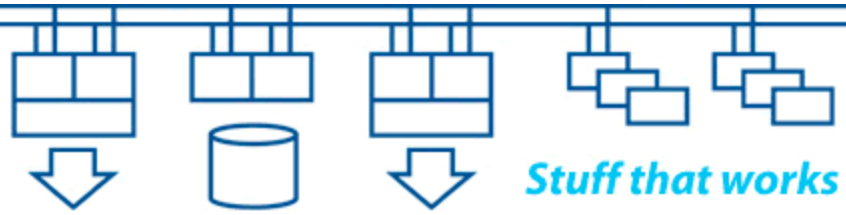
- Zones and VSANs are used to segment a SAN fabric:
  - Implemented within SAN switches (directors)
  - Zoning and VSANs – unlike data networks, nothing connects by default
  - Extended SAN fabrics spanning multiple sites
  - "Converged ethernet" as a common backbone in the data centre

Part 3:

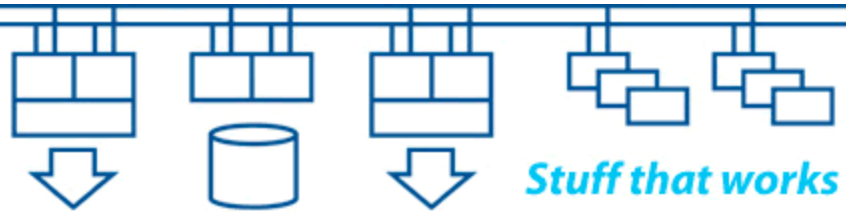
Virtualisation of systems:



- What is "virtualisation" in its current context?
  - It's another way of hiding the underlying hardware so that we can use it without having to think about it.
  - If we virtualise the processing element (CPU) then we can share out one or more physical CPUs amongst a number of "virtual machine instances" of operating systems.



- What can CPU virtualisation do for us?
  - Improved utilisation of hardware resources
  - Improved high availability and disaster tolerance
  - Parallelisation of processing
  - Multiple run-time environments
  - Minimise hardware dependencies

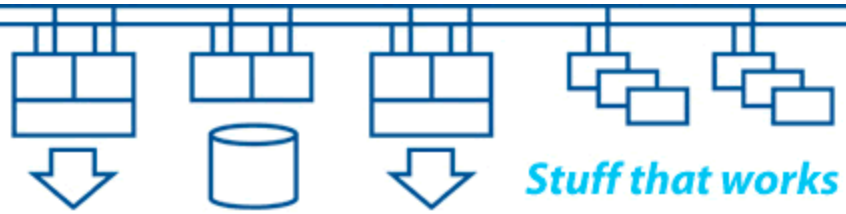


- A CPU cannot exist in isolation - we need:
  - CPU – processing capability
  - Memory – stores data and instructions ready for use
  - IO – moves data into / out of memory and communicates with other systems
  - Console access

- CPU virtualisation provides a set of guest machine environments for a native machine running under a 'hypervisor' on the base hardware (eg: HPUX based hypervisor on Integrity supports HPUX and OpenVMS Integrity guests – all use the Itanium instruction set)

- The 'hypervisor' allocates physical machine resources to the guest machine environments

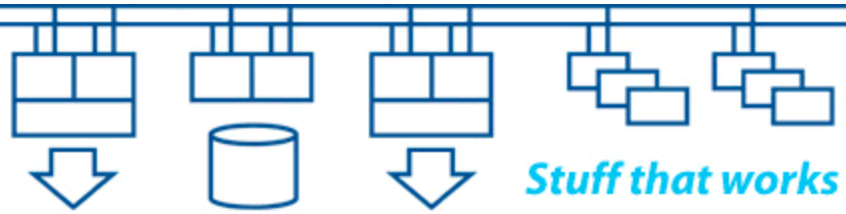
- The booted operating system running in a guest machine environment co-operates with the hypervisor at device driver level with purpose-written drivers



- Emulators provide a machine environment for a non-native machine (eg: SIMH will provide an emulated 32bit VAX hardware and instruction set on a range of non-VAX host platforms)

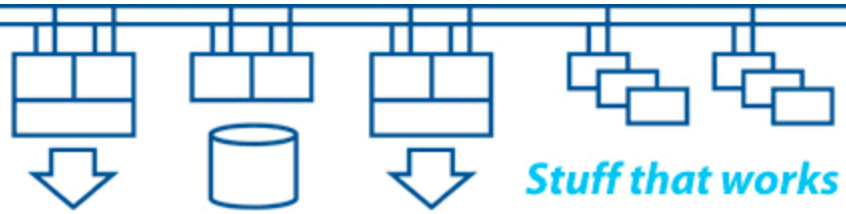
- The emulated machine environment behaves as much like the real machine as possible

- No changes are made to the operating system and device drivers of the system running in the emulator – all the hard work is done in the emulator



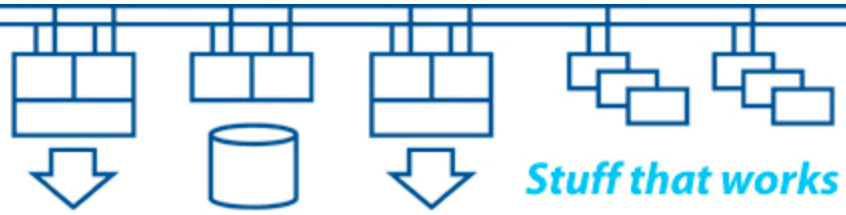
- The VM is a multi-threaded application running under the control of the hypervisor
- CPU scheduling in the guest OS maps to threads scheduled by the hypervisor
- Physical memory in the guest OS maps to virtual memory in the hypervisor – lots of physical memory in the hardware platform is a good thing

- Devices presented to the operating system running inside the VM have to 'map through' to devices managed by the hypervisor:
- Storage devices typically map to container files, not physical hardware
- Network devices typically map to 802.1Q tagged VLANs, not physical hardware



Part 4:

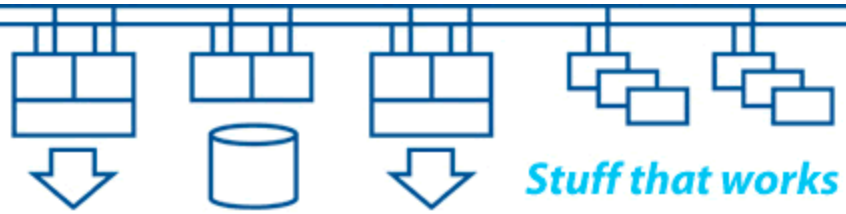
IO flow through a virtual system:



Typical I/O flow overview

- Application to OS kernel I/O services
- OS kernel I/O services to OS device driver
- OS device driver to VM side of presented device

- Hypervisor side of presented device to hypervisor device representation
- Hypervisor device representation to storage container file or network VLAN
- Storage container file I/O to physical storage device or network VLAN tagged packets to physical NIC



- How do we identify the endpoints ?
- Need to minimise latency
- Physical I/O typically takes a few milliseconds – we can execute a lot of CPU instruction cycles in a few milliseconds (1 GHz = 1 nanosecond, thus 1 million instruction cycles per millisecond)!
- The two main factors affecting performance are "wait states" and contention for resources

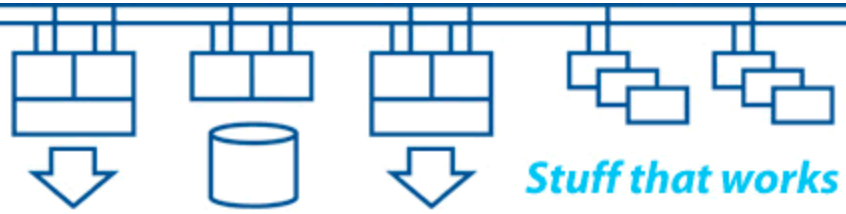
IO flow in a virtual system (2)

- Device drivers in virtual machines communicate with host system (hypervisor) device representations
- Host system (hypervisor) device representations communicate with physical devices
- Paravirtualisation allows a virtual system to directly and exclusively use an external (outside the VM) physical device, bypassing the hypervisor as far as possible

- Device drivers in virtual machines communicate with host system (hypervisor) device representations
- Host system (hypervisor) device representations communicate with physical devices
- Paravirtualisation allows a virtual system to directly and exclusively use an external (outside the VM) physical device, bypassing the hypervisor as far as possible

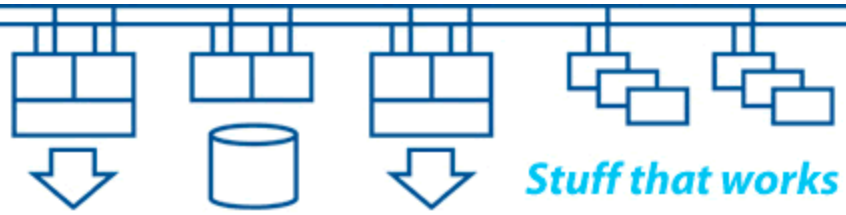
Part 5:

Hardware infrastructure:



Non-blade hardware platforms

- Big high-end multiprocessor systems are needed for certain workloads. They provide a "scale-up" solution.
- Stand-alone systems are needed in highly secure or mission-critical / safety-critical environments
- Small environments do not need the cost and complexity of blade chassis infrastructures



- Blade technology brings virtualisation of the system infrastructure (chassis components):
  - Virtual connections from processing components over backplane channels – think about how WWIDs and MAC addresses are presented
  - Modular systems provide great flexibility of configuration and interchangability of components
  - Blades are largely a "scale-out" solution

- The end to end I/O path is extremely complex, especially in a blade environment
- Think about how the various applications interact and thus which VMs to place on which hardware platforms
- Think about how to perform maintenance and support activities with minimal disruption to service
- Think about how to architect and monitor the entire infrastructure

Thank you for your participation

Discussion!

www.xdelta.co.uk