Connect webinar – 6th April 2010

Virtualisation in theory and practice

Colin Butcher

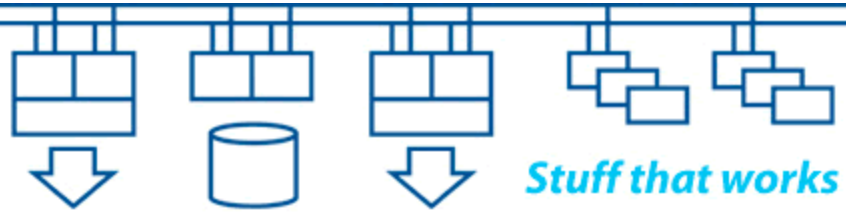
www.xdelta.co.uk

www.downloads.xdelta.co.uk/2010/2010_04_06-virtualisation-colin_butcher.pdf

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

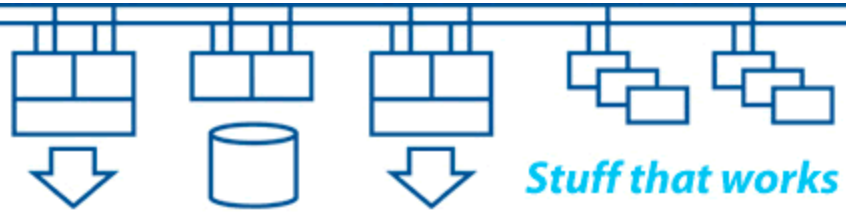
Humpty Dumpty, Through the Looking Glass,
by Lewis Carroll

Virtual = .NOT. Physical



Part 1:

Thinking about the architectural design of systems.



"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"

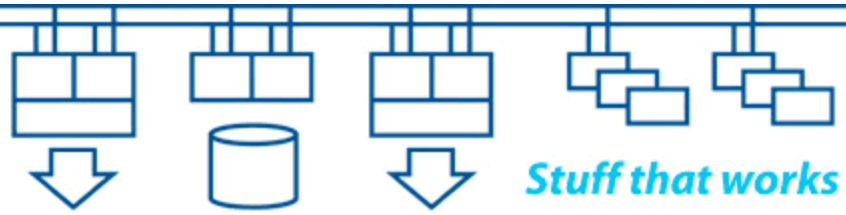
We need to strike a balance that is appropriate for the kinds of systems we're building.

	Before	Now	After
Environment			
System			
Component			

See www.triz.co.uk for a lot more information

- Big decisions which have long-term implications and constraints
- Small decisions which seem big at the time
- There will be requirements and constraints you don't yet understand or know about

So, aim to have a consistent and effective architectural structure with well defined interfaces and the flexibility to accommodate change when needed.



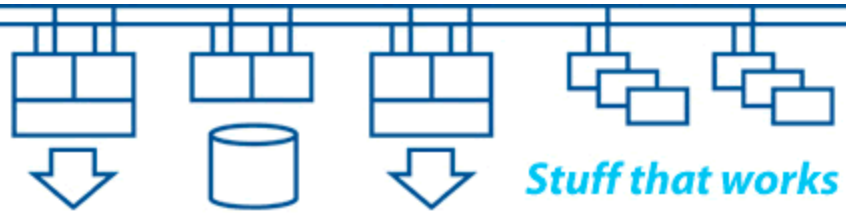
Part 2:

Understanding systems and how they behave.

Systems store data, process data and exchange data with other systems and users:

- CPUs do processing
- Memory holds data and instructions
- Storage subsystems let us store and retrieve data
- Data networks let us communicate

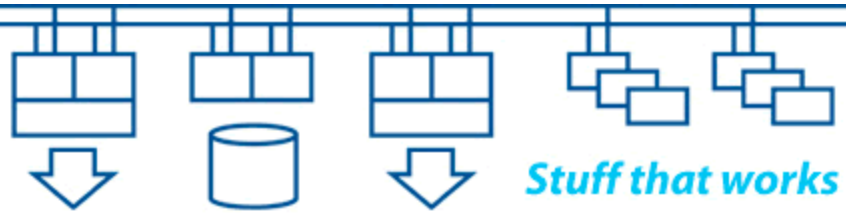
- Bandwidth – determines throughput
  - It's not just "speed", it's throughput in terms of "units of stuff per second"
- Latency – determines response time
  - Determines how much "stuff" is in transit through the system at any given instant
  - "Stuff in transit" is the data at risk if there is a failure
- 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



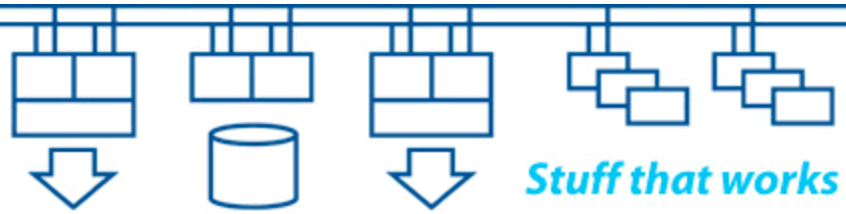
- Understand how your workload could break down into parallel streams of execution:
  - Some will be capable of being split into many small elements with little interaction
  - Some will require very high levels of interconnectivity and interaction
  - Some will require high-throughput single-stream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

- Increasing capacity of the overall system:
  - "Scale up" or "vertical scaling" refers to increasing the capacity by adding resources to a machine (CPU, memory, IO).
  - "Scale out" or "horizontal scaling" refers to increasing the capacity by adding more machines.
- Understand the need for synchronisation and serialisation of access to shared data structures

- Minimise contention for resources and data structures
- Minimise "wait states" (caches, parallelism)
- Maximise "User Mode", minimise the other modes:
  - The fastest IO is the IO you don't do
  - The fastest code is the code you don't execute
- Software scalability – do as much as possible once only, do little as possible every time.



- Trade memory for performance by caching data and instructions in memory ready for use
- Multi-core processors have shared on-chip cache, so can outperform single-core processors for certain types of workload
- Size systems to handle peaks in workload, not average throughput
- Don't make it go faster, prevent it from going slower!



- Ability to make use of parallelism in hardware
- Granularity of data structures
- Synchronisation techniques
- Serialisation techniques
- Scalability techniques
- Compilers
- Application design

- Designing and writing very good code requires very good programmers

Part 3:

Familiar virtualisation techniques:

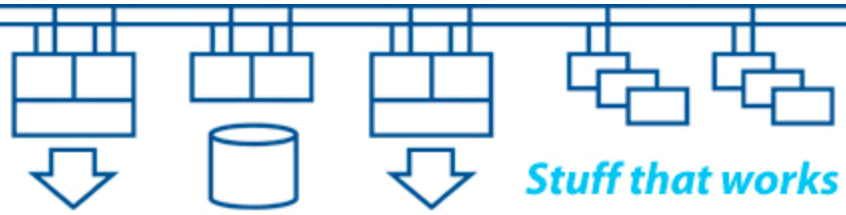
- Memory management: Virtual memory
- Data networks: VLANS
- Storage subsystems: SAN zones, LUNs (Vdisks)

- Each process can use the full address space capability of the machine
- Process address space is ‘mapped’ into physical memory
- Physical memory usage per process is controlled by the operating system
- Per process memory that exceeds the permitted physical memory quotas is ‘parked’ in the page file ready for use

VLANs are used to segment a data network:

- Implemented within core switches
- Connectivity between VLANs
- VLAN tagging of packets (802.1Q)
- VLAN tagging of packets out of NICs
- QoS (Quality of Service) and bandwidth reservation

- Converged ethernet as a common backbone in the data centre



Storage array controllers:

- The array "hides" the behaviour of the discs from you
- The array "levels" the storage to provide best throughput for the access pattern to the entire array (or disk group)
- The array controller caches much of the data

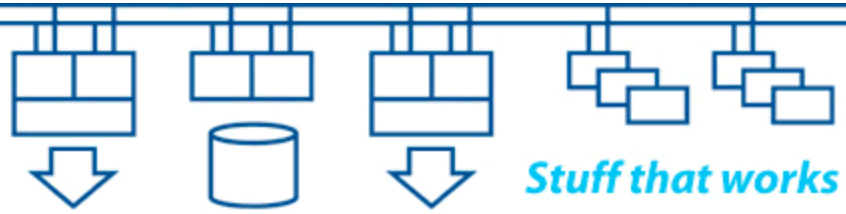
- The only real performance issues are bandwidth to and from the array controller pair and contention for access to the storage array

Zones (and VSANs) are used to segment a SAN fabric:

- Implemented within SAN switches (directors)
- Connectivity between zones and VSANs
- Converged ethernet as a common backbone in the data centre

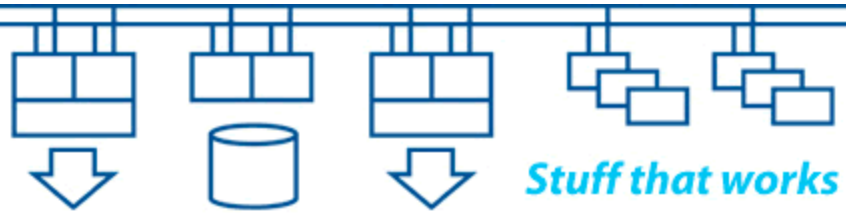
Part 4:

Virtualisation of systems:



What is "virtualisation" in its current context?

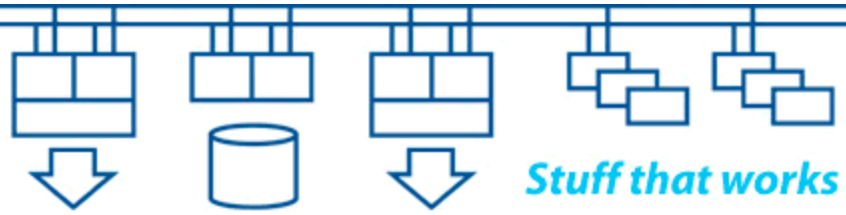
- It's another way of hiding the underlying hardware so that we can use it without having to think about it.
- If we virtualise the processing element (CPU) then we can share out one or more physical CPUs amongst a number of "virtual machine instances" of operating systems.



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

IO is the most difficult to deal with in a virtual environment

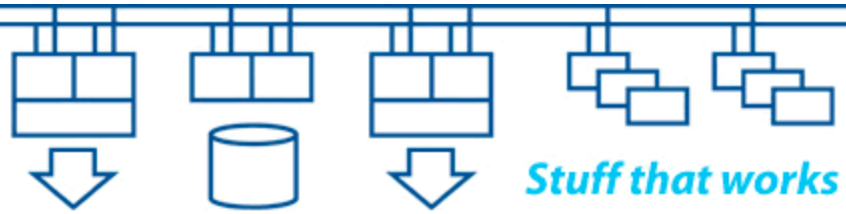


- CPU virtualisation provides a set of guest machine environments for a native machine running under a 'hypervisor' on the base hardware (eg: HPUX based hypervisor on Integrity supports HPUX and OpenVMS Integrity guests – all use the Itanium instruction set)
- The 'hypervisor' allocates physical machine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

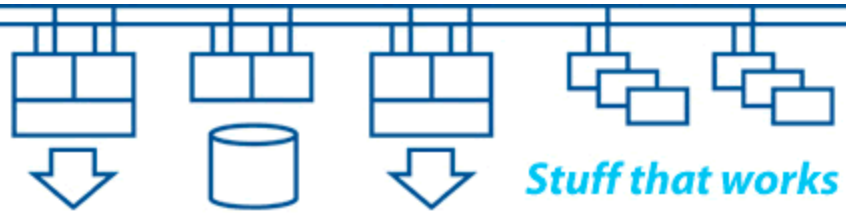


What can CPU virtualisation do for us?

- Improved utilisation of hardware resources

  It can let us move "workload units" around a set of hardware resources to provide improved utilisation of the available hardware.

  It can let us start a virtual system only when we need access to it. That could be an emulated system for intermittent access to historic data or software.



What can CPU virtualisation do for us?

- Improved high availability and disaster tolerance

  If that hardware is at different physical locations then it can also help to provide some level of high availability (within a site) and disaster tolerance (across multiple sites), provided that the data is replicated between the different locations and that the communications paths are available.

What can CPU virtualisation do for us?

- Parallelisation of processing

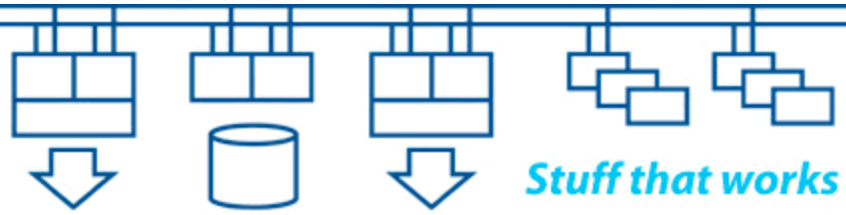
  It's another way to achieve parallelisation of processing without having to invest time in writing code that works well on a multi-processor system.

  However, we still need to think about concurrent access to data from multiple "virtual machine instances" and providing access to the multiple "virtual machine instances".

What can CPU virtualisation do for us?

- Multiple run-time environments

  We can create many "virtual machine instances" where each has different operating system and layered product versions, then only run the ones we need at any given point in time.



What can CPU virtualisation do for us?

- Minimise hardware dependencies

The host system (hypervisor) can present the physical devices in a consistent and generic manner to the guest instances. The actual hardware type can be 'hidden' from the operating system running in the virtual machine environment.

- Need to "map" IO devices in the virtual system to the physical devices
- How do we identify the endpoints? Think about things like ethernet MAC addresses and SAS WWIDs
- Need to minimise latency

- Device driver in virtual system communicates with host system (hypervisor) devices
- Host system (hypervisor) device driver communicates with physical devices
- Paravirtualisation allows a virtual system to directly and exclusively use a physical device

Part 5:

Hardware infrastructure:

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

- 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

- Availability, reliability and complexity
- Monitoring and management tools

- Maximum achievable throughput (bandwidth)
- Effects of latency and jitter
- Realistic testing for scale and performance

- Software deployment techniques
- Planning downtime for maintenance on shared-use hardware with many "virtual machine instances"
- Licensing policies and "pay per use"

With a blade infrastructure you're trading a relatively high initial investment in the platform / infrastructure for a smaller cost of incremental growth.

Going virtual for one or two systems on a new blade platform may not be the most cost effective solution, but it could be a long-term win.

Thank you for your participation

Discussion!

www.xdelta.co.uk