BCS Bristol Branch

Virtualisation in theory and practice

Colin Butcher

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

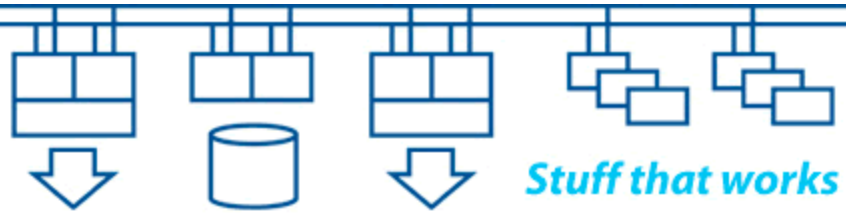
Humpty Dumpty, Through the Looking Glass, by Lewis Carroll.

Virtual = .NOT. Physical

"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			

Design decisions

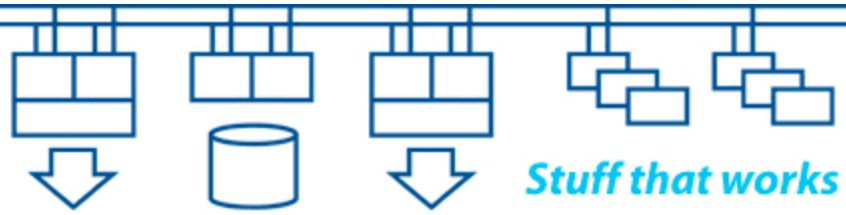
- Big decisions which have long-term implications and constraints
- Small decisions which seem big at the time
- There will be requirements and constraints you don't yet understand or know about

What does a system do?

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

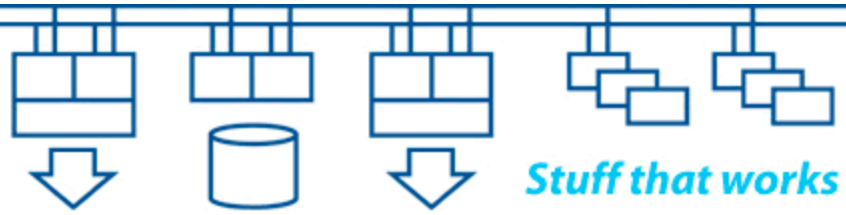
- Bandwidth – determines throughput
  - It's not just "speed", it's throughput in terms of "units of stuff per second"
- Latency – determines response time
  - Determines how much "stuff" is in transit through the system at any given instant
  - "Stuff in transit" is the data at risk if there is a failure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



- Size systems to cope with peaks in workload
- Minimise "wait states" (caches, parallelism)
- Minimise contention for resources and data structures
- Understand the need for synchronisation and serialisation of access to data structures
- 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

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



- Understand 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
- Eliminate wait states - 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

- Each process is capable of using the full address space capability of the machine
- Process address space is ‘mapped’ into physical memory – translation buffer hardware and page table entries are where the microprocessor and operating system meet
- Physical memory usage per process is controlled to maintain overall performance and prevent interactions
- Per process memory that exceeds the permitted physical memory quotas is ‘parked’ in the page file ready for use

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

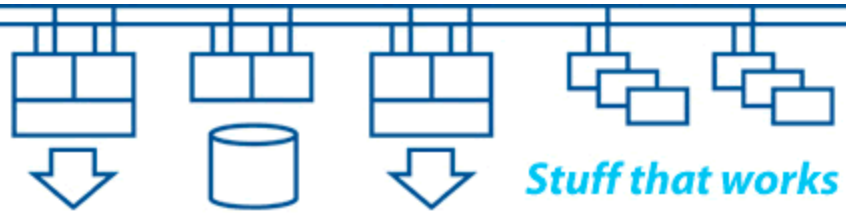
IO is the most difficult to deal with in a virtual environment

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.

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 issue is bandwidth to and from the array controller pair and what else is contending for that bandwidth

VSANs are used to segment a SAN fabric:

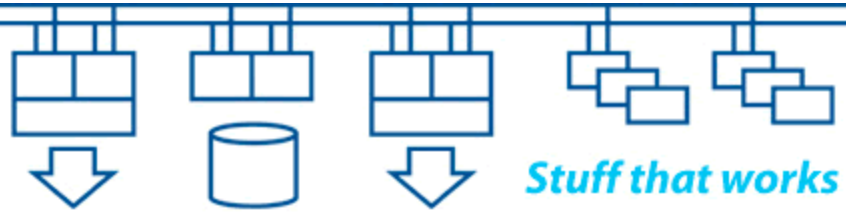
- Implemented within SAN switches (directors)
- Connectivity between VSANs
- Converged ethernet as a common backbone in the data centre

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

Hardware platforms (2)

- Big high-end multiprocessor systems are needed for certain workloads
- Stand-alone systems will still be needed in highly secure or mission-critical / safety-critical environments



- Maximum achievable throughput (bandwidth)
- Effects of latency and jitter
- Monitoring and management tools
- Software deployment techniques
- Availability, reliability and complexity
- Planning downtime for maintenance on shared-use hardware with many “virtual machine instances”
- Realistic testing for scale and performance
- 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!