Systems performance engineering

HP Discover London 2015

Understanding availability and performance

Colin Butcher CEng FBCS CITP
Technical director, XDelta Limited
www.xdelta.co.uk

xdelta

Personal background

- Systems architect specialising in mission critical systems
- Engineering background (printing, power generation)
- Wide range of experience (aerospace, healthcare, finance, transport, power and energy)
- Strong interest in mentoring and teaching

xdelta

XDelta – what we do

- Lead mission-critical systems projects:
  - Strategic planning
  - Technical leadership
  - Project direction
- Minimise risk of disruption to business:
  - Design for change while in continuous operation
  - Prepare in advance for ease of transition
- Ensure long term success through skills transfer

xdelta

Agenda

- Concepts and principles
- Networks and storage infrastructure
- Hardware and operating systems
- Virtualisation
- Design, testing and trouble-shooting
- Discussion

xdelta

Availability and performance

A system that doesn't meet its performance requirements is a system that's not working properly, so it becomes unavailable.

Performance related failures are often transient and exceedingly difficult to fully understand and resolve.

Systems have to have sufficient capacity and inherent performance to deal with the workload within an acceptable period of time under normal, failure and recovery conditions.

What does success look like ?

- What is the purpose of the system ?
- What are the consequences of failing to perform ?
- What are our performance criteria ?
- How can we demonstrate that we've met them ?

xdelta

Part 1 - Principles

- Abstraction layers
- Performance characteristics
- Parallelism
- Scalability

Abstraction layers

"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.

xdelta

Using abstraction layers

- What looks like your dedicated resource is just a slice of a much bigger thing over which you may have little control:
  - What looks like a network isn't the whole network
  - What looks like a disc isn't a disc
  - What looks like memory isn't all of memory
  - What looks like CPUs aren't all the CPUs
- The operating system allocates and manages machine resources
- It's even more complicated in a virtualised environment

The “Hall of mirrors”

- You can’t see everything that’s going on
- The view is often distorted
- Hiding things makes it easier to deal with the bits you’re interested in
- Hiding things makes it much harder to understand what’s happening, especially with performance related problems

Performance characteristics

- Bandwidth – determines throughput
  - It's not just "speed", it's "units of stuff per second"
- Latency – determines response time
  - Determines how much data is in transit
  - "data in transit" is at risk if there is a failure
- "diff latency" (variation of latency with respect to time) or "jitter" - determines predictability of response
  - Important for establishing timeout values
  - Latency fluctuations will cause system failures under peak load

xdelta

Capacity and scalability

- Contention and saturation – running out of capacity
  - What else are we sharing our capacity with ?
  - Queuing theory

- Increasing the capacity of the overall system:
  - “Scale up” or “vertical scaling” – adding resources to a machine or buying a bigger machine (CPU count, memory, I/O adapters, etc.)
  - “Scale out” or “horizontal scaling” - adding more machines

Parallelism

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

xdelta

Current technology trends - parallelism

- Move from low core count, high clock rate processors to high core count, low clock rate processors
- Implicit assumption is that parallelism can be achieved
- Algorithm design is key
- Don’t leave it all to the compilers
- Serialisation, synchronisation and intercommunication

Part 2 – Networks and storage

- Data networks
- Storage arrays and SANs

xdelta

Data networks

- Ethernet switches
- Availability – avoid single points of failure
- Traffic segmentation - VLANs
- Traffic management – QoS
- Load balancers
- Firewalls

xdelta

Data networks - segmentation

- VLANs are used to segment a data network:
  - Implemented by using 802.1Q tagging of packets
  - Systems can behave as if they are switches and send tagged packets for multiple VLANs over the same NIC
  - Switch configurations generally map Layer 3 IP V4 subnets to Layer 2 VLANs and enforce IP routing between VLANs
  - Extended VLANs can span multiple sites
- “Software defined networking” - OpenStack

xdelta

Storage networks

- iSCSI – uses data networking (poor man's fibrechannel)
- Fibrechannel – designed for storage networking
- SANs
- Storage arrays

SAN zoning and VSANs

- Implemented within SAN switches
- Fibrechannel zoning and VSANs – unlike data networks, nothing connects by default
- Extended SAN fabrics spanning multiple sites
- Fibrechannel and iSCSI behave differently

Storage arrays

- Array controllers cache much of the data, using protected (battery or flash memory) write-back mirrored cache
- Array controller “hides” the behaviour of the physical discs and distributes the IO load across multiple spindles.
- Performance issues:
  - bandwidth to and from the array controller pair
  - contention by systems for access to the storage array
  - controller processing overheads (eg: RAID 0+1 v RAID 6)

Part 3 – Hardware, OS and software

- Server hardware (supercomputers to small servers)
- Client hardware (desktop, portable, mobile)
- Operating systems
- Applications

xdelta

Hardware platforms (1)

Blade technology brings virtualisation of the system infrastructure (chassis components):

- Virtual connections from processing components over backplane channels
- Modular systems provide great flexibility of configuration and interchangeability 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
- Current trend is high core count / large memory machines

NUMA (non-uniform memory access)

- Locality of memory and IO devices
- How to lay out data structures for equitable access ?
- How to exploit system for best performance ?

Operating systems

- Share out and arbitrate access to machine resources
- Protect users from each other
- Give the illusion that each user has the machine to themselves
- Java comes with its own run-time environment
- With virtual machines, the hypervisor is a “master” OS with the “guests” nested inside

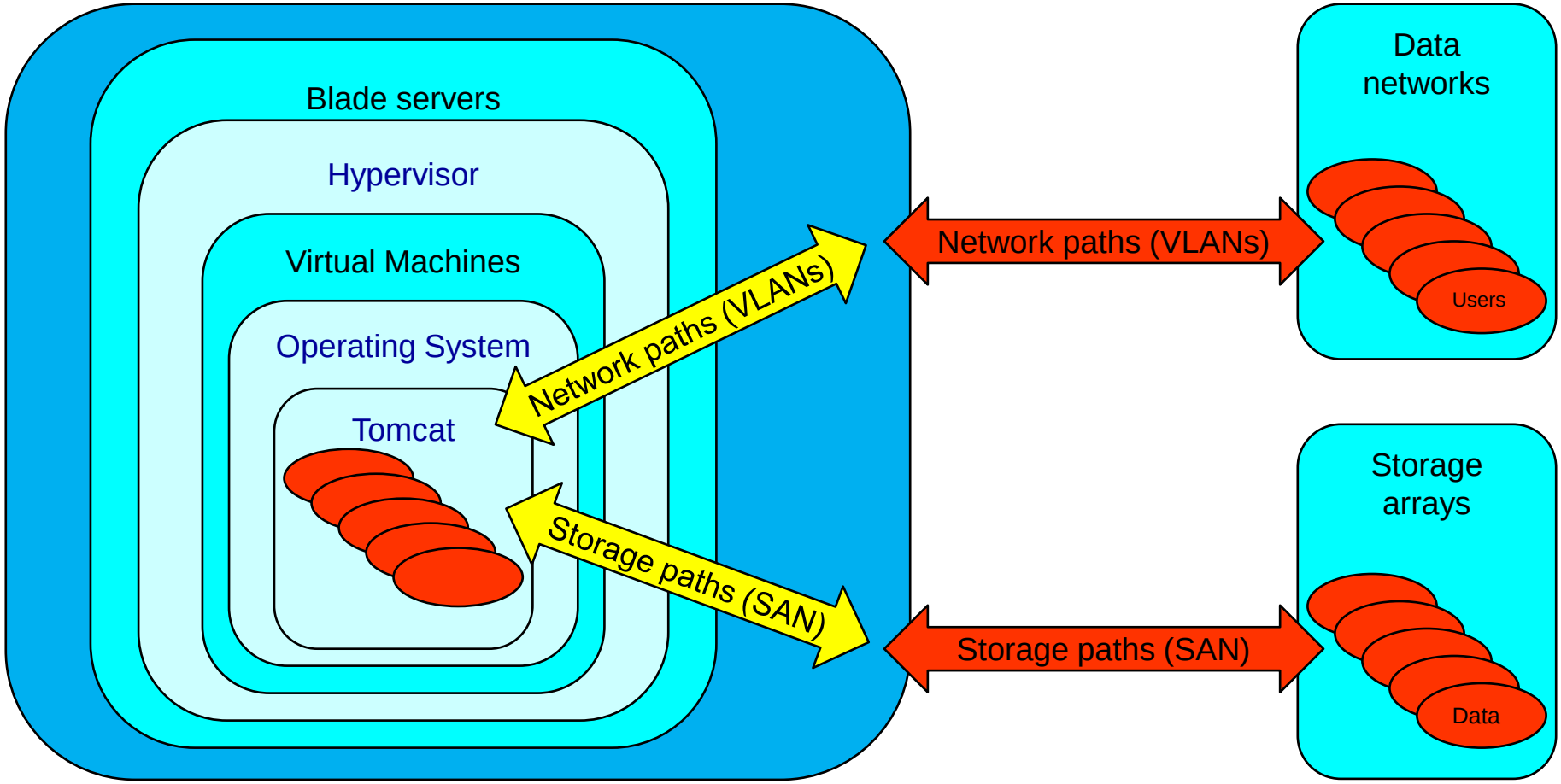
Part 4 - Virtualisation

"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

xdelta



xdelta

Workloads in a virtual world

- 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 ?

Virtual machine devices

- 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 using a “virtual switch” in the hypervisor that interconnects the virtual system NICs to each other and to the physical NICs visible to the hypervisor

IO flow in a virtual system

- Operating system device support for running under a hypervisor requires purpose written device drivers on the ‘inside’ of the virtual machine to communicate with the way the hypervisor represents a device to the ‘outside’ of the virtual machine

- Device drivers in virtual machines communicate with host system (hypervisor) device representations

- Host system (hypervisor) device representations communicate with physical devices

Part 5 – Design and trouble-shooting

- Design for performance
- Load testing
- “It’s slow” – what next ?
- Performance data and trend analysis
- Understand the whole system

xdelta

Designing for performance

- Size systems to cope with peaks in workload

- Have a “balanced” system
  - A faster CPU just waits more quickly
- Minimise “wait states”
  - Don’t make it go faster, stop it going slower
- Maximise “user mode”, minimise other modes:
  - The fastest code is the code you don’t execute
- Maximise “IO throughput”:
  - The fastest IO is the IO you don’t do

Designing and writing very good code requires very good programmers.

xdelta

Code paths and data flows

- How does your code scale up ?
- Separate out static data from dynamic data
- Minimise frequently executed code paths
- How to reduce impact on system and surrounding network and storage infrastructure ?

IO performance v CPU performance

- Physical I/O operations typically takes a few milliseconds to complete

- We can execute a lot of CPU instruction cycles in a few milliseconds (1 GHz = 1 nanosecond, thus 1 million instruction cycles per millisecond)!

Analysis tools and instrumentation

- Code analysis tools can help with finding interactions and heavily executed code paths
- Build instrumentation into your software, then you don't change its behaviour by adding temporary code
- Pay attention to state transitions and event timing
- Synchronisation and wait states are expensive

Testing

- How can we simulate realistic scenarios ?
- Test for scale, not just functionality
- Test to find out what really happens under load and under failure conditions
- Performance failures are usually transient, so how will you capture fine-grained enough data to capture a problem ?

“It’s slow” !

- What do they mean ?
  - Is it responding poorly ?
  - Is the responsiveness varying too much ?
  - Are batch jobs running slowly ?
  - How long are key activities taking ?
- Is the expectation unreasonable ?
- Is there anything wrong at all ?

xdelta

Performance data and trend analysis

- Without data for historical comparison, how do we know what's reasonable ?
- Without data, we're guessing
- Data needs to be synchronised in time across everything
- Don't jump to conclusions – correlation does not imply causation
- Most problems are combinations of several things

xdelta

Interactions and unexpected effects

- How can we segment a system to minimise performance impact and failure propagation ?
- How can we guarantee against data loss and corruption ?
- Recovery from failure usually has most impact, e.g: data replication back to a known good state
- How quickly do we need to recover ?

xdelta

Summary

- Most people focus on performance
- Performance and availability are inextricably linked
- Good holistic design is essential
- Performance can’t be added later
- Trouble-shooting and resolution requires good data and a thorough understanding of the whole system

xdelta

Systems performance engineering

HP Discover London 2015

Thank you for your participation

Colin Butcher CEng FBCS CITP
Technical director, XDelta Limited
www.xdelta.co.uk

xdelta