Mission-critical systems with OpenVMS

OpenVMS Bootcamp 2017

Session 233

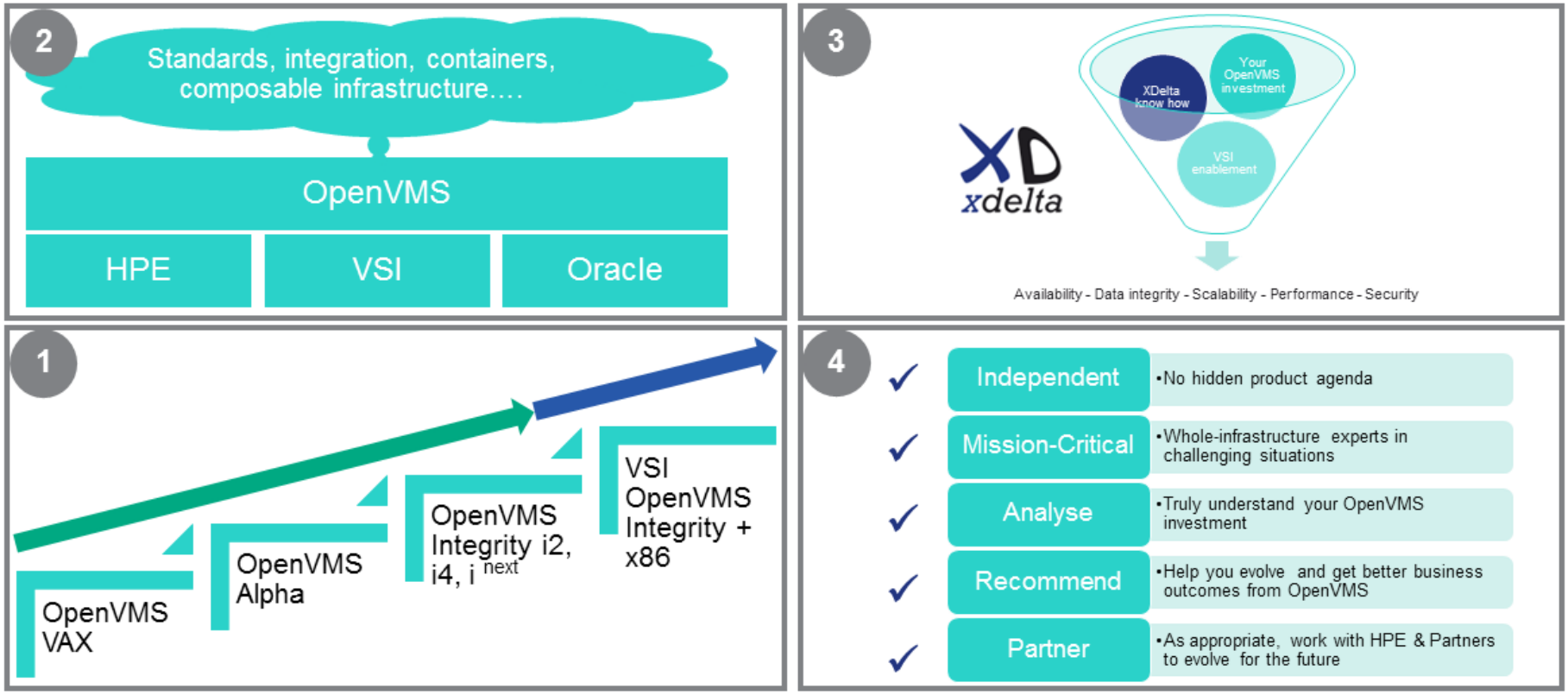
Colin Butcher CEng FBCS CITP
Technical director, XDelta Limited
www.xdelta.co.uk

XDelta: Who we are

vms Software

- VSI Professional Services Alliance member
- Independent consulting engineers since 1996:
  - UK based with international reach
  - Delivering OpenVMS based systems for 30+ years
- Technical leadership for business-critical systems
  - Design, planning and implementation
  - Mentoring and skills transfer
  - Systems engineering background
- Gartner (2009):
  - Identified XDelta as one of few companies world-wide capable of OpenVMS platform migration projects

XDelta - a trusted advisor to advance your critical OpenVMS application infrastructure



Hewlett Packard Enterprise

Agenda

- Design principles
- Network and storage connectivity
- Storage layout, shadowing, booting
- Log file management
- Backup / restore
- Performance
- Monitoring and management
- Example – replacement of all hardware without loss of service

Systems engineering

It's a multi-disciplinary and holistic approach to creating something to meet a specific purpose.

From the NASA Systems Engineering Handbook, June 1995:

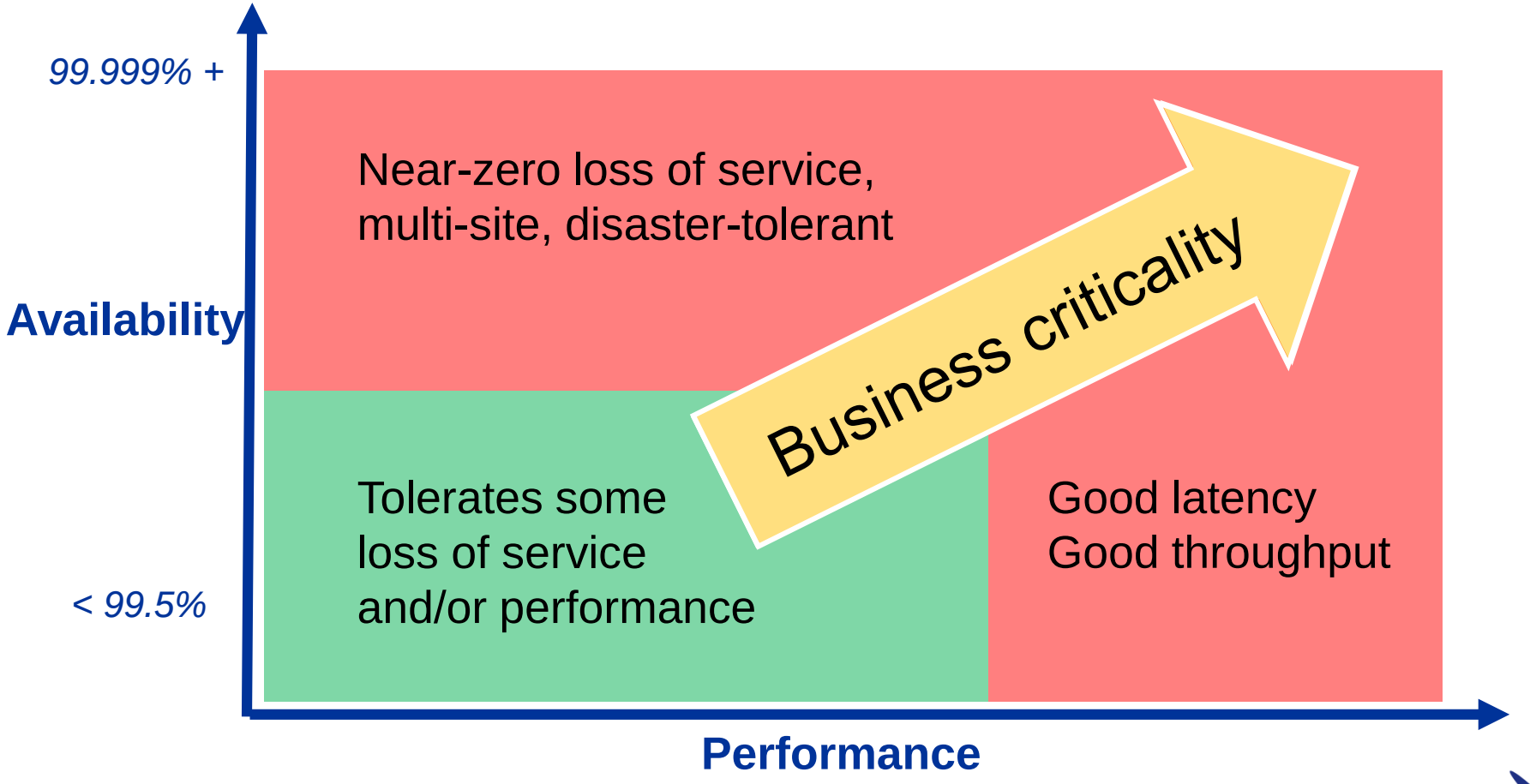
"[Systems engineering] is a field that draws from many engineering disciplines and other intellectual domains. The boundaries are not always clear, and there are many interesting intellectual offshoots."

What are mission-critical systems ?

Systems that are relied on to get something done, without data loss or corruption and without stopping working when they need to work.

Usually there is some kind of severe penalty for systems failure, be it financial, or legal, or business-threatening, or life-threatening.

How critical are your systems ?



Terminology

- Availability:
  - Probability of system being available for use when needed
  - Function of MTBF (reliability) and MTTR (repair time)
- Disaster tolerance:
  - Surviving major site outages without loss of service
- High availability:
  - Surviving failures at a site without loss of service
- Disaster recovery:
  - Restarting systems after loss of service (however brief), typically from another location (DR site)

Design goals

- Design for change, not steady-state
- Think long-term
- Operational safety – minimise risk of errors and disruption
- Understand the purpose and the target environment
- Build in logging and information gathering
- Adapt to changing requirements (performance, scalability)
- How to test it?
- How to transition into service?
- How to transition to follow-on system?

The design process

- Understand the requirements
- Start with the “ideal design”
- Understand any constraints you have to deal with
- Think ahead to minimise problems later
- Build the whole system “on paper”
- Have a well-structured overall architecture
- Understand the details, complexities and interactions
- Remain flexible and adapt to change

Design decisions

- All design decisions are compromises and require you to exercise judgement
  - 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
- Make careful assumptions as needed to get started
- Establish meaningful naming conventions
- Document your design and decisions

The survivability matrix

Cause of Outage	Planned (Maintenance)	Unplanned (Failure)
Hardware	?	?
Operating System	?	?
Network	?	?
Application Software	?	?
Data	?	?
Environment	?	?
People	?	?

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. The systems have to have sufficient capacity and performance to deal with the workload in an acceptable period of time under normal, failure and recovery conditions.

Risk and failure analysis

Risk is a combination of probability of occurrence and worst-case effects for a given failure scenario.

Allow for failure – success is only one of many possible outcomes.

Designing for availability

- Which parts of the system are mission-critical ?
- Which parts of the system are safety-critical ?
- What kind of failure do we prefer ?
- What state transitions occur ?
- Build in the ability to make changes when in service
- Protect the data
- Build “proof of concept” systems and simulators

The risk continuum

- What is the probability of a situation occurring ?
- What is the impact if that situation occurs ?
- What are the long-term consequences ?

- Most projects handle medium risk well enough
- Many projects over-specify to cater for low risk issues
- Some projects under-specify and fail to cater for high risk issues

- We need to identify critical components / people
- We need to identify critical stages during the project

Risk identification and assessment

- Can we identify specific scenarios of interest ?
- Can we test all the conditions ?
- What happens to our data ?

- How can we start to identify what the risks might be ?
- How can we look for single points of failure ?
- How can we look for modes of failure ?
- How can we analyse how failures will ripple through ?

Testing, transition and operation

A mission-critical system hardly ever fails, so we need the people responsible for its operation to have a good understanding and ‘feel’ for the way it works.

Most failures result from a combination of several things and are thus complex and difficult to understand and resolve.

Failure and recovery

- We need to know how the system behaves
- We need to know the ‘warning signs’ of incipient failure
- We need to know how to return the system to its normal operational state without data loss or data corruption

- We need to regularly rehearse and test our procedures and plans to ensure that we stay current

- We must have a representative offline test environment

Testing

- Understand the requirements and acceptance criteria
- How do we generate a typical workload ?
- How do we generate representative data sets ?

- Test under normal, failure and recovery conditions
- Don’t just confirm that the system behaves as expected
- Must test for scalability as well as functionality

Transition into service / out of service

- Minimise risk of data loss
- Minimise risk of loss of service
- Migrate user connectivity
- Migrate live data + historic data

- How can we split transition into manageable steps ?
- Is anything a one-way step ?
- How much can we do in advance ?
- How could we revert to the original system ?

Finding and fixing problems

- How can we spot a problem early on, eg: data corruption ?
- What evidence can we look at ?
- Can we recreate the problem in a test environment ?

- Time synchronisation across the whole system is essential

- Continual monitoring and event logging is essential

- Knowledge of the whole system is essential

Naming conventions

- Choose your naming conventions very carefully – they are the hardest thing to change later
- Don’t tie nodenames to physical locations
- Choose disc device IDs that identify meaningful things (e.g.: environment, site, array and purpose)
- Choose network addresses and hostnames that identify meaningful things and make sense in your context

Example node naming convention

<n1><nn2>DC<n3>, where:

<n1> = “P” (Production), or
“T” (Test), or
“D” (Development)

<nn2> = 01 … 99 (node number within site)

DC = “data centre” (site)

<n3> = 1 … 9 (site number)

Example naming convention – FC discs

- 1DGA<n1><n2><nn3>

  <n1> = site / environment (1 … 3 = Production A, B, C;
  4 … 6 = Test A, B, C;
  7 … 9 = Development A, B, C)

  <n2> = array within site (1 … 9)

  <nn3> = disc, matches DSA<nn> shadow set name

- 1DGA<nnnnn> = common (RO) discs

Note: UUID must be unique across the fabric, ALLOCLASS = 1

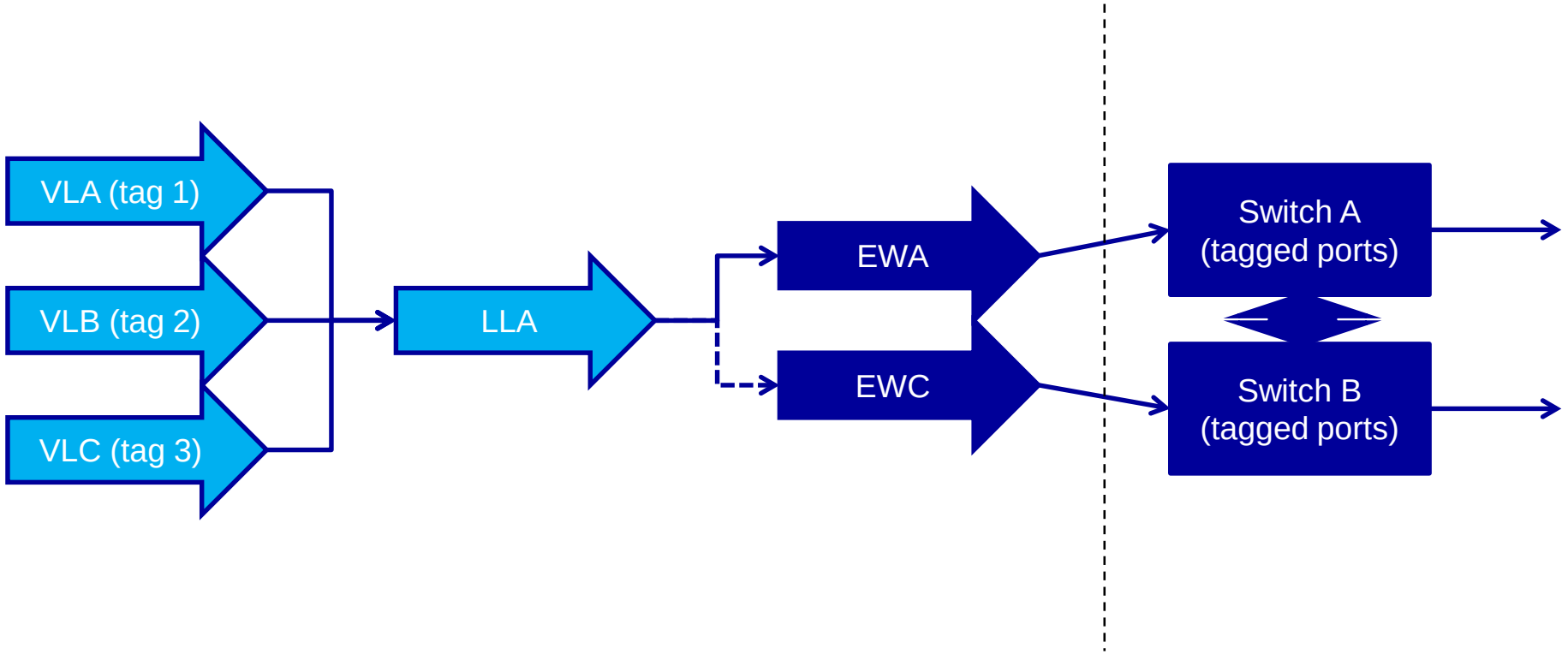
Network connectivity (1)

- Multiple protocols: SCS, TCPIP, DECnet, AMDS, etc.
- Use LAN failover with multiple NICs for hardware resilience
- Use VLAN tagging and/or LAN failover sets to separate traffic flows
- VL / LL devices map to physical NICs, avoid configuring protocols on physical NICs.
- Understand how protocols behave with multiple paths

Network connectivity (2)

- Use “service addresses” to separate data flows
- Use QoS in data network for different data flow types
- Use SCACP to control which port(s) SCS runs on
- Use LATCP to control which port(s) LAT runs on
- Disable unused protocols on NICs (eg: DECdns, DTSS)

OpenVMS networking: connectivity



Inter-site data network links

- Extended layer 2 or routed layer 3 ?
- SCS at layer 2 or “clusters over IP” ?
- Preference is to use extended layer 2 with QoS on specific VLANs to control latency and bandwidth
- LAT is a useful protocol to test connectivity paths at layer 2
- AMDS (Availability Manager) is a layer 2 protocol
- Avoid MSCP serving, especially with shadow sets

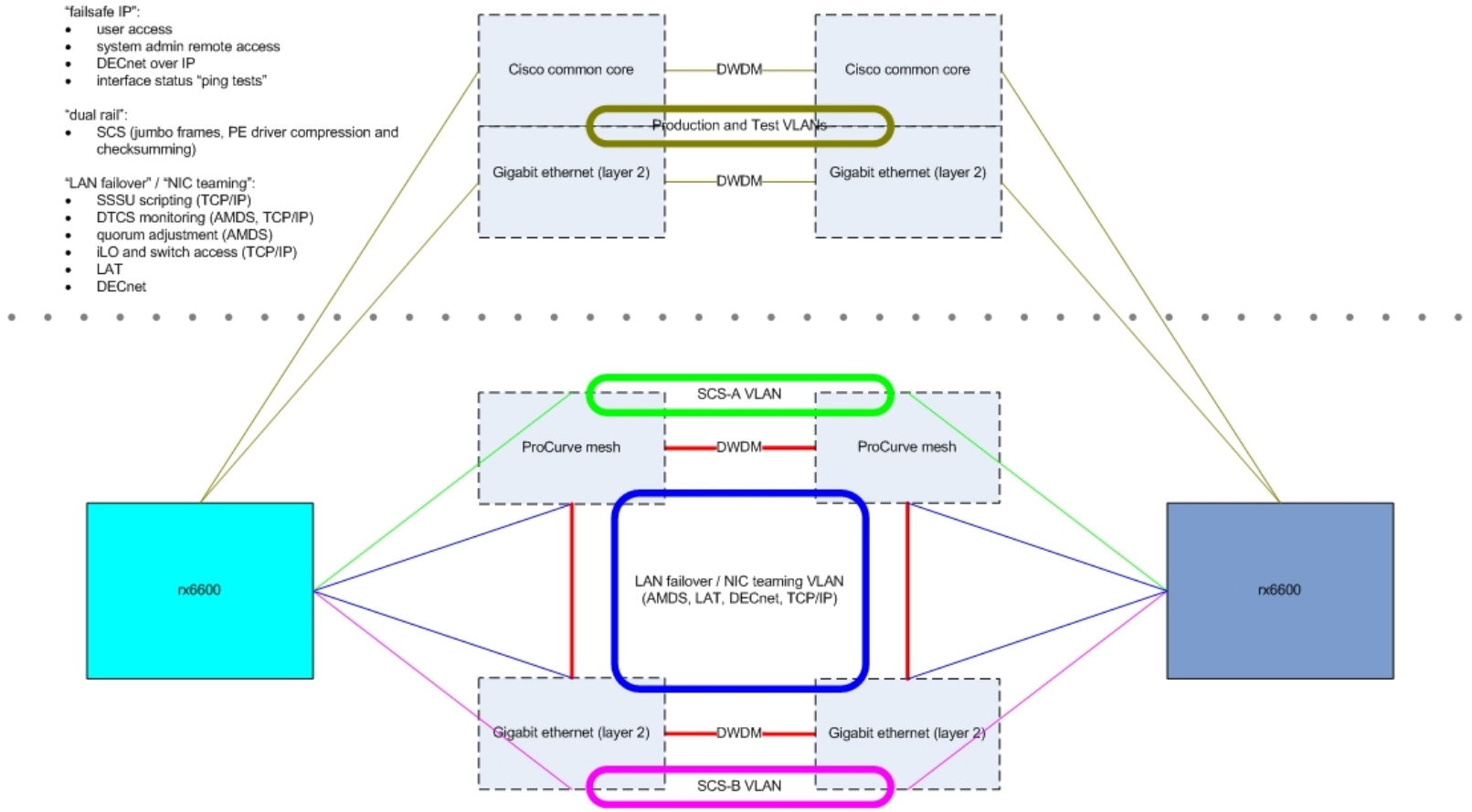
Extended layer 2 LANs

- DWDM over dark fibre
- MPLS
- MPLS “pseudo-wires” over private MPLS network

- Traffic separation with VLAN tagging (802.1Q)
- Use QoS to control traffic flows

- Switches have manufacturer specific features:
  - HP Procurve has “meshing”
  - Cisco has “etherchannel”
  - Extreme has “EAPS ring”

Example data network connectivity



Storage connectivity

- Fibrechannel uses WWIDs:
  - WWN = World Wide Name
  - WWNN = World Wide Node Name (points to entire array or tape drive or multi-port HBA)
  - WWPN = World Wide Port Name (point to specific port in array controller or tape drive or HBA)

- Storage element (LUNs) presentation to HBA
- OpenVMS uses UUID to set device name

- OpenVMS multi-path FC devices

SAN fabrics, HBAs and storage devices

- Zoning:
  - Soft zoning by WWPN, not WWNN
  - Single initiator, multiple target
  - Consider locking port type and speed

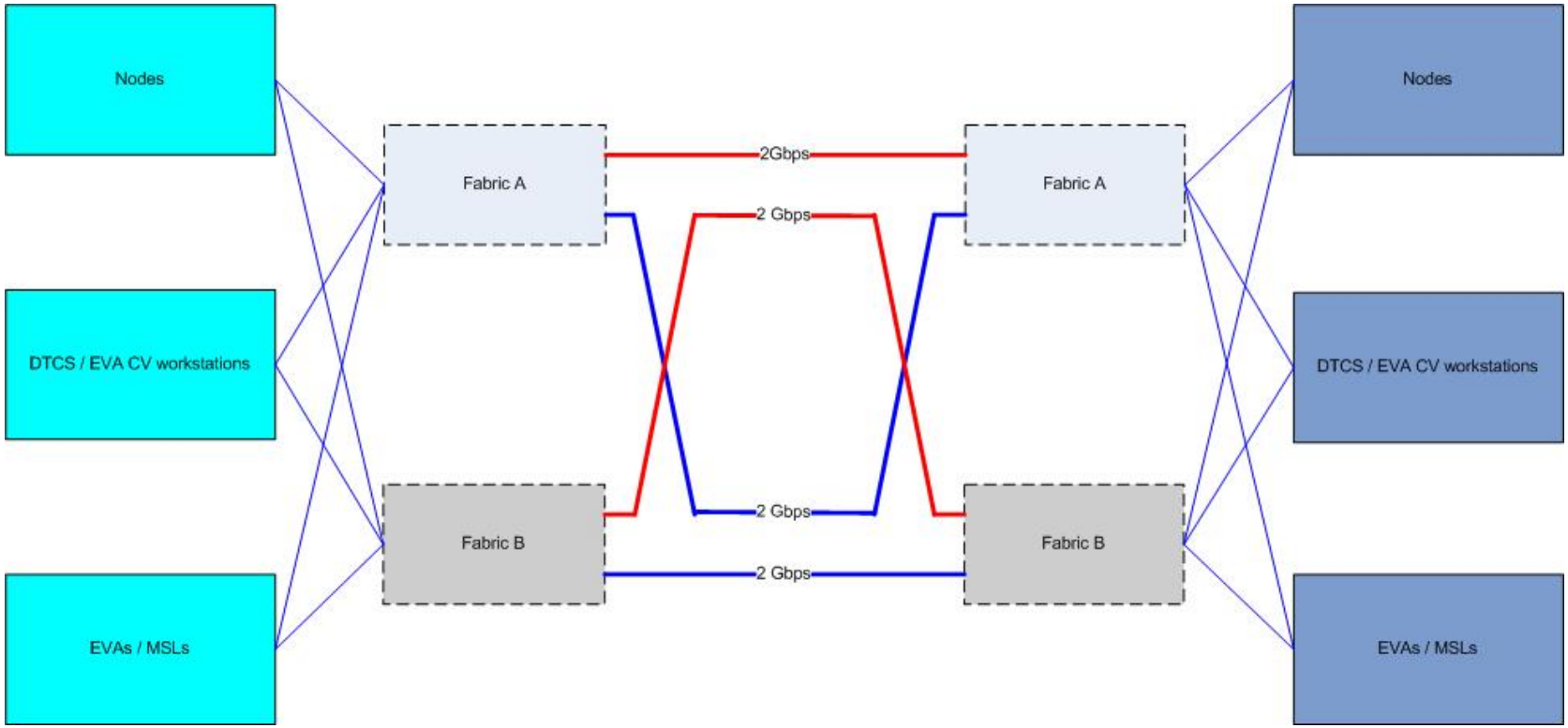
- Presentation:
  - “export” VV in 3PAR, “present” Vdisk in EVA
  - Set LUN ID
  - Needs UUID (determines DG device name)

- Be consistent with LUN, DG<nnnn>, etc. allocation

Inter-site storage (SAN) links

- Use direct path fibrechannel with SAN extension
- Dual fabric SAN
- Avoid path switching on ISL failure by dual-path connection per fabric with one connection over each ISL per fabric
- Enable MSCP as an alternate path mechanism
- Use mini-copy and mini-merge
- Avoid cross-site booting
- Only mount site-specific discs at their site, even if shadowed to all sites (eg: per-site shadowed system discs)

Example SAN connectivity



Shadowing

- Many shadow sets for performance with multi-path discs
- Small shadow sets to minimise copy/merge time
- Enough arrays per site to always have local source
- Set “site” values to bias reads from site-local storage
- Only mount system discs on nodes booted from that disc
- System disc at a site is shadowed to other sites
- Use minicopy and minimerge for performance
- Set shadow copy buffer size to multiple of 16 (112)

Array configuration (3PAR, EVA)

- Use RAID 0+1 (EVA vRAID1) for best performance
- Use double sparing, single disc group (EVA)
- Snaps are only a short-term point in time temporary entity – they can hurt array controller performance
- Clones have better performance, but require more space
- Consider explicit path specification and explicit controller preference for preferred path configuration
- Use SSD for best performance

Booting

- Requires firmware support for HBA and array
- Boot drivers are lightweight
- View from EFI shell is extremely hard to interpret
- Use BOOT_OPTIONS.COM to configure boot paths, or use efi$bcfg.exe directly (see command line help)
- When adding a node to an existing cluster, ALWAYS mount the target system disc READ ONLY
- Delete root <SYS0> to avoid unexpected booting with unconfigured hardware

Note: “deep scan” of fibre by HBAs can take ages
Note: memory tests on large machines can take ages

Quorum nodes

- Avoid quorum disc if possible
- Physical quorum node (IA64 or Alpha)
- HP VM based quorum node (IA64)
- Alpha V8.4 quorum node on emulator in a virtual machine

Quorum and voting

- Is application “cluster aware” or rapid failover ?
- What do you want to happen when a site fails ?
- Votes and expected_votes
- Availability manager / DTCS quorum adjustment
- <Ctrl-P> quorum adjustment on Integrity

Log file management

- Fragmentation is a problem worth avoiding
- Use LD containers: write log files to the LD device, then simply move containers to archive.
- Block net$server.log (and others) by creating an empty ;32767 version
- Avoid too many files in a directory – use search lists

Backup & restore, archive

- Backup – be able to restore quickly if needed
- Archive – remove stale data and retain for reference

- FC tape libraries, drive based encryption
- Off-site copies, VTL (virtual tape library)

- Backup strategy & process:
  - Must be consistent in time
  - Applications must be quiescent
  - Drop shadow set member (or stall IO to member)
  - Use array based clones and snaps

Hardware maintenance and replacement

- Keep firmware up to date – plan sequence to avoid disruption
- FC devices with same UUID but different WWPNs will show up as the SAME device but with extra paths
- Keep systems modular with minimal configuration per node
- Alternate system discs for upgrades / replacements
- Record ILO configurations

Techniques for rolling upgrades

- Primary and alternate system discs per site (or per node)
- Page / swap / dump files off system discs
- Common disc(s) for system-wide files etc. (UAF, rightslist, queue manager databases, etc.)
- Keep everything tidy and know where everything is
- Make copies in the storage arrays before you start
- Know how to back out and revert to where you started from

Planning the changes

- Permissible downtime usually controls the planning
- Risk of disruption usually controls the sequencing
- Never do too many things at once
- Only do one thing at once for critical actions
- Identify any one-way steps as early as possible
- Stay current enough to be supported

Getting things done safely

- Shift workload around to deliver continuous service
- Write everything down in painful detail as a checklist, with anticipated timings
- Rehearse as much as you can
- Do as much as you can in advance
- Make sure everything is tested and ready to go
- Tell everyone what's going on and monitor everything
- Never proceed until you know things are safe
- Don't be afraid to call it off and back out
- Never work alone

Project planning and management

Some useful adages:

“Proper Planning and Preparation Prevents Piss Poor Performance”

“Time spent in reconnaissance is never wasted”

“No battle plan survives first contact with the enemy”

Planning and implementation

- Estimating and planning are key
- You cannot know everything up front
- Make effective assumptions to get started
- Beware assuming that everything will go well
- Cumulative discrepancies add up very quickly
- How will you monitor progress ?
- Checklists are essential, especially under pressure

"More software projects have gone awry for lack of calendar time than all other causes combined."

"The mythical man-month" – Brooks

Management

- Concentrate on quality of information and decision making
- Be thoughtful, not reactive - do not rush to respond
- Regular briefings, in person, phones off, no e-mail
- Need people to be committed
- Never have one person working on their own
- Find good people, guide them and trust them
- Good administrative support lets people focus on their work
- Good management enables people to get things done

Leadership and people

"Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity."

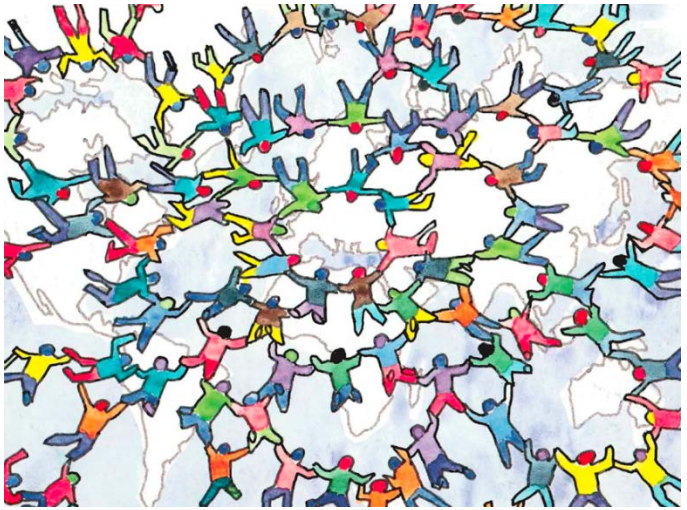
General George Patton

"The best executive is the one who has sense enough to pick good people to do what he wants done, and self-restraint enough to keep from meddling with them while they do it."

Theodore Roosevelt

People and behaviours

- Build groups of excellent people who work well together
- Choose people who are willing to share information and help each other
- Give them the support and help they need so that they aren't distracted by trivia
- Confidence good; arrogance bad
- We're all in this together!



Summary

Most of what we deliver has no physical reality. Those involved need good conceptual skills and the ability to communicate ideas clearly. All those involved need to have a sufficiently similar conceptual model, which relies on good communication.

Most projects go wrong through mis-match of expectations and lack of understanding.

Team size is limited by the necessary level of communication and the skills of those involved.

Procurement and responsibility

- Procurement – do enough work up front:
  - Understand what is technically feasible
  - Understand what is strictly necessary
  - Clearly establish the scope
  - Define clear objectives
  - Define acceptance criteria

- Avoid split responsibility and be absolutely clear where the "duty of care" lies

Design, leadership and management

- Design and implementation:
  - Have clear objectives. Think ahead as far as you can. Have a well-structured systems architecture. Understand the constraints. Focus on the core functions. Implement them as well as is possible.
- Project leadership:
  - Ensure that everyone involved maintains a consistent understanding of the project. Plan ahead as best you can.
- Budget and Schedule:
  - They have to be appropriate for the problems you're trying to deal with. Don't set them first!

Key success factors

- Strong team of good people
- Collaboration and willingness to share information
- Build a “proof of concept” early on and learn from it
- Minimise complexity
- Structured design
- Clearly defined interfaces
- Document your decisions (what, how, why)
- Make life as easy as you can for those who come after you

Mission-critical systems with OpenVMS

OpenVMS Bootcamp 2017

Session 233

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