OpenVMS best practices

Designing and building OpenVMS systems and clusters

Connect Germany 2021 online conference

Colin Butcher CEng FBCS CITP

Introduction

My business has been leading systems and infrastructure projects, many of them involving VMS based disaster-tolerant systems. It’s been technically complex and demanding work with a high level of responsibility.

As a designer or trouble-shooter, it’s essential to understand how everything that makes up the complete system functions and interacts.

The intention is to help other people who get involved with VMS systems and clusters. I hope it’s useful.

Agenda

1. **Planning ahead**

2. Hardware platforms
3. Network infrastructure
4. Storage infrastructure
5. Installation and setup
6. Boot, startup and shutdown
7. Hints and tips
8. Multi-site clusters

Design goals

- Availability is generally more important than performance

- Never corrupt or lose data

- Build in capability for:
  - Remote operation (lights-out)
  - Hardware maintenance
  - Patching and upgrades
  - Stand-alone backup and restore
  - Migration and hardware replacement

Design principles

- Design for change, not steady-state
- 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)
- Think long-term (e.g.: company mergers)

Naming conventions

- Choose your naming conventions very carefully – they are the hardest thing to change later
- Don’t tie nodenames to physical locations. Physical server names should be different to nodenames.
- Choose disk 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

Agenda

1. Planning ahead

2. **Hardware platforms**

3. Network infrastructure
4. Storage infrastructure
5. Installation and setup
6. Boot, startup and shutdown
7. Hints and tips
8. Multi-site clusters

OpenVMS versions (Alpha and Integrity)

- Alpha:
  - HPE V8.4
  - VSI V8.4-2L1 (for all generations of Alpha hardware)
  - VSI V8.4-2L2 (for EV6 and later – and emulators)

- Integrity:
  - HPE V8.4 (up to -i2 servers, NOT –i4 and –i6 servers)
  - VSI V8.4-1H1 and later (supports –i4 and –i6 servers)
  - VSI V8.4-2L3 is latest release

Integrity Server firmware

- Older hardware (prior to –i2) will require USB flash drive media for all firmware components

- Newer hardware (-i2 onwards) can use HPSUM firmware bundles for the base system, loaded over the network from Windows / Linux via the ILO

- Some firmware components will not load from the HPSUM bundle, but require USB flash drive media (SAS controller, LOM controller, HDDs, etc.)

Making use of CPU

- Hyperthreading – very workload dependent, “threads off” is a good starting point
- Fastpath IO devices and distributed interrupt handling
- Introduce parallelism into your code flow and batch jobs where possible
- Dedicated CPU for lock manager (local locking)
- Compression and encryption, eg: SCS compression, BACKUP compression
- QUANTUM, workload dependent – many SYSGEN parameter defaults changed in V8.2
- Power management

Making use of memory

- NUMA – “mostly UMA” is a good starting point
- Use memory (XFC, resident images, DECram etc.)
- Revisit working set sizes - WSMAX and process quotas Use RMS global buffers
- Revisit RMS system defaults
- GH regions – map lots of memory with a small number of page table entries
- INSTALL /RESIDENT and GH region size
- 64bit P2 space and memory reservations
- DECram and shadowing to fast disc

Making use of storage IO

- FC bandwidth is important – what else are you sharing your storage bandwidth with? Why?
- Rotational latency no longer matters, nor does balancing the IO load to the spindles
- Array controller cache size and volume characteristics (cluster factor, extend quantity, volume expansion etc.)
- HBVS – only shadow what you really need to
- HBVS – many shadow sets let you control how rapidly shadow copying proceeds during recovery
- HBVS mini-copy and mini-merge policies
- HBVS block count to match array controller cache size

Making use of network IO

- Network bandwidth and latency matter:
  - Use jumbo frames
  - Split the traffic across multiple NICs
  - What else are you sharing your bandwidth with ?
  - Are the switches over-subscribed ?
  - Need to avoid retransmits
- Multiple path networks:
  - Packets may not arrive in the order in which they were sent
- Beware bus speed limitations in servers

Agenda

1. Planning ahead
2. Hardware platforms

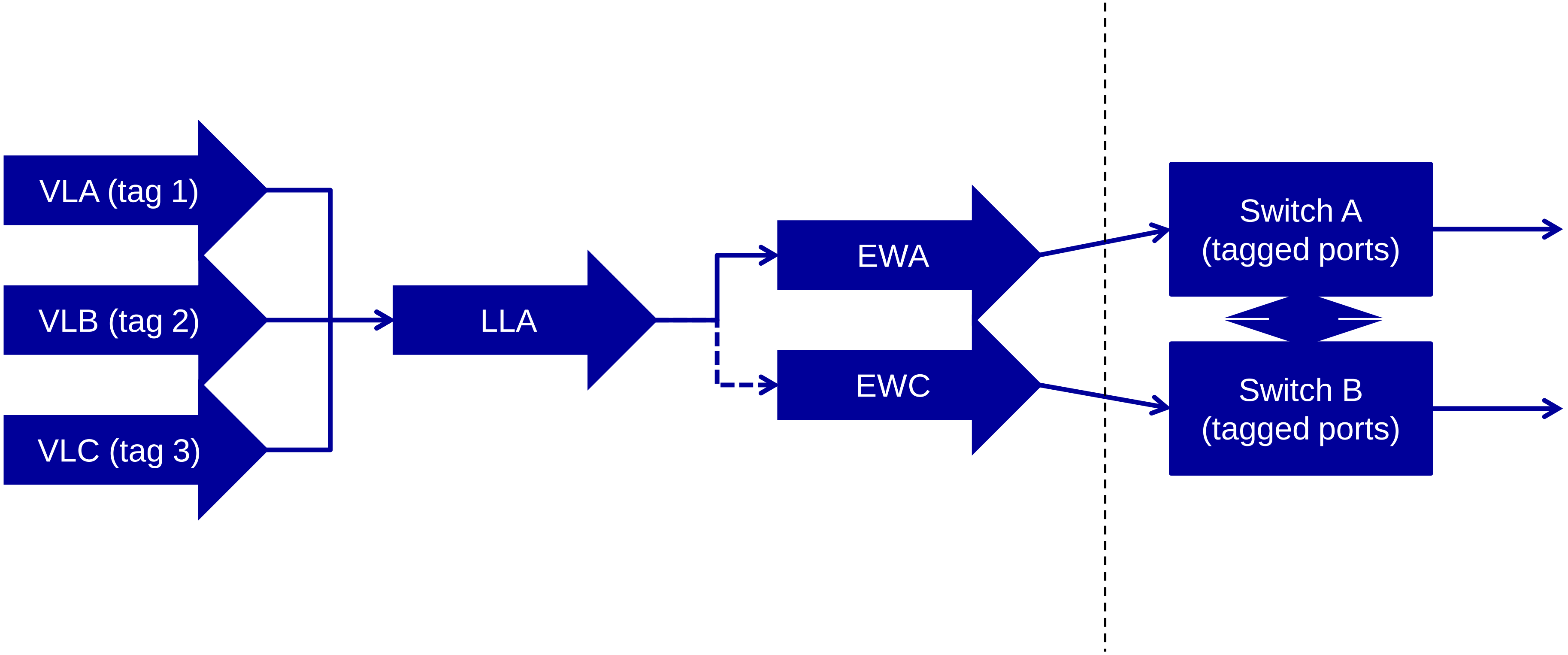
3. **Network infrastructure**

4. Storage infrastructure
5. Installation and setup
6. Boot, startup and shutdown
7. Hints and tips
8. Multi-site clusters

Network connectivity

- Multiple protocols: SCS, TCPIP, DECnet, AMDS
- 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, do not configure protocols on physical NICs.
- 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 (eg: DECdns, DTSS)

OpenVMS networking: VLAN / LAN failover



Network switch configuration

- Split traffic flows into different VLANs:
  - ILO and management
  - Cluster interconnects
  - Data feeds
  - External access

- Tagged ports using VLdriver, or untagged ports ?

- Failover sets using LAN failover

Network infrastructure – multiple paths

- Cisco: etherchannel
- HP / Aruba: Procurve meshing
- Extreme: EAPS ring

- VMS: LAN failover
- Other operating systems: NIC teaming, bonds, LACP

Agenda

1. Planning ahead
2. Hardware platforms
3. Network infrastructure

4. **Storage infrastructure**

5. Installation and setup
6. Boot, startup and shutdown
7. Hints and tips
8. Multi-site clusters

Fibrechannel switch configuration

- OpenVMS only supports fibrechannel switched fabric
- Zoning connects HBA ports with storage array ports:
  - Single initiator, multiple target zoning
  - Soft zoning by WWID, not by port number
  - Zone by WWPN, not WWNN
- Zone all nodes with all storage arrays
- Consistent use of LUN IDs and UUIDs

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)

- Zoning – single initiator, multiple target, use WWPN

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

Fibrechannel storage array configuration

- 3PAR: virtual volumes, exported to HBAs
- EVA: vdisks, presented to HBAs

- Multipath devices
- Consistent use of LUN IDs and UUIDs

- VMS does not support “thin provisioning”.
- VMS “erase on delete” can return blocks to pool if absolutely necessary, with performance overhead

- Volume expansion

Array configuration

- Use RAID 0+1 (EVA vRAID1) for best performance
- Use double sparing, single disk 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

Example naming convention – FC disks

- $1$DGA<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> = disk, matches DSA<nn> shadow set name

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

Example disk naming – DSA and FC disks

- DSA10 ($1$DGA1010, 2010, 1110, 2110) – common disk

- DSA11 - system disk, site A
- DSA12 - alternate system disk, site A
- DSA13 - system disk, site B
- DSA14 - alternate system disk, site B
- DSA15 - system disk, site C
- DSA16 - alternate system disk, site C

- DSA21 … DSA39 – data (small shadow sets)
- etc.

Agenda

1. Planning ahead
2. Hardware platforms
3. Network infrastructure
4. Storage infrastructure

5. **Installation and setup**

6. Boot, startup and shutdown
7. Hints and tips
8. Multi-site clusters

Example server (ILO) naming convention

S<nn2>DC<n3>, where:

“S” = Server (the physical machine and ILO)

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

“DC” = “data centre” (site)

<n3> = 1 … 9 (site number)

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)

Quorum and voting

- Is application “cluster aware” or rapid failover ?
- What do you want to happen when a site fails ?
- Avoid quorum disk if possible
- Availability manager / DTCS quorum adjustment
- <Ctrl-C> quorum adjustment on Integrity

SCS at layer 2 or “clusters over IP” ?

- SCS at layer 2 needs extended LAN between sites. This may not be possible.

- Some network switches (Cisco) can drop layer 2 multicast packets when congested

- SCS starts on all NICs, so disable SCS on non-cluster connection NICs early in boot (SYCONFIG)

- IPCI (IP cluster interconnect) at layer 3

Shadowing

- Many shadow sets for performance with multi-path disks
- Small shadow sets to minimise copy/merge time (especially common disk)
- Enough arrays per site to always have local source
- Only mount system disks on nodes booted from that disk
- System disk at a site is shadowed to other sites
- Use mini-copy and mini-merge for performance

Bootable environments

- External storage (SAN) - separates bootable system and data from server hardware
- Local storage (RAID) for node specific and temporary purposes

- Normal operation: boot from SAN, shadowed system discs, alternate system discs for mixed version running (upgrades etc.)
- Local “maintenance boot” system with full capabilities

- Integrity servers:
  - Local USB flash drive “installation boot”, can then reconfigure local storage controller and reinstall from pre-built system backup
  - Local USB flash drive for firmware updates (FAT32 format)

Disc layout

- SAN devices, common, shadowed:
  - System disc
  - Common disc
  - [alternate system disc]

- Local RAID devices, not shadowed:
  - Page/swap/dump disc(s)
  - T4 data disc

- Always move the common files and the page / swap / dump files off the system disc

Local RAID storage

- Usually faster than SAN storage (lower write latency)

- RAID 0+1 or RAID 6
- RAID 6 may require a licence key for the RAID controller

- Local page and swap files
- Local dump files (dump off system disc – DOSD)

- Local T4 data files
- Local staging area for backups (eg: backup to local storage, compressed, then encrypt, then copy elsewhere)

Local RAID controllers

- Battery-backed or flash-backed cache
- Set up as RAID 0+1 or RAID 6
- RAID 6 may need a licence key, depends on hardware
- Partition into logical units using all discs
- Bias read/write cache ratio in favour of write (80% / 20%). VMS has lots of read caches (XFC, RMS)

Local RAID controllers - example

- Booted from USB flash drive

```
$ mcr msa$util
set controller pka
add unit 0 /part=0/disk=(1,2,5,6)/adg/size=50gb     ! DKA0 LMB system
add unit 1 /part=1/disk=(1,2,5,6)/adg/size=10gb     ! DKA1 install media
add unit 2 /part=2/disk=(1,2,5,6)/adg/size=100gb    ! DKA2 T4 data
add unit 3 /part=3/disk=(1,2,5,6)/adg/size=500gb    ! DKA3 pgswpdmp
add unit 4 /part=4/disk=(1,2,5,6)/adg               ! DKA4 Staging area
!
set globals/write=80/read=20                        ! Write / read cache
exit
```

Scanning SAN fabric pre-boot (Integrity)

- Fibrechannel HBAs can be configured to scan entire fabric looking for devices

- May be faster to only scan devices that appear in boot and dump lists

- Choose fibre scan option in ILO by setting scan level in HBAs

Agenda

1. Planning ahead
2. Hardware platforms
3. Network infrastructure
4. Storage infrastructure
5. Installation and setup

6. **Boot, startup and shutdown**

7. Hints and tips
8. Multi-site clusters

Booting

- Requires firmware support for SAN (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 disk READ ONLY
- Delete root <SYS0> to avoid unexpected booting with unconfigured hardware

OpenVMS boot and startup

- Power-up sequence
- Console view of hardware and devices
- Boot process
- Base system startup
- Device configuration
- LAN devices, pseudo-devices and IPCI (if needed)
- Cluster formation
- STARTUP.COM and site-specific SY*.COM files
- Network transports startup
- Layered products startup
- Application startup

Power-up sequence

- Hardware tests: CPUs, memory, devices, etc.
- Can disable tests for speed of power-up
- Auto-boot – disable in HA/DT environments
- Alpha console:
    - serial port (or MBM on last Alphas)
        - >>> prompt, some commands are machine specific
        - partitioning and galaxy
        - wwidmgr for FC devices
- Integrity console:
    - ILO
        - EFI / UEFI menu interface
        - Make EFI shell the first option (safe)

Finding devices to boot from

- Alpha console:
  - wwidmgr for FC devices
  - boot* environment variables (bootdef_dev etc.)

- Integrity console:
  - Decoding the view from EFI shell (FS<nn> devices)
  - FS<nn>: \efi\vms\ EFI scripts

- Boot flags to specify system root etc.

Boot loader

- Alpha:
  - APB on target system disk / root

- Integrity:
  - IPB on target system disk / root
  - EFI shell sees a FAT partition
  - FS<nn>:\efi\vms\vms_loader.efi is the initial loader
  - VMS sees the ODS2/5 file system
  - SYS$EFI.SYS is a container file within the VMS file system
  - EFI$CP is an undocumented tool to access SYS$EFI.SYS

OpenVMS early stages of boot

- Minimalist drivers for boot (and dump)
- Reads system parameter file
- If flag set, enters SYSBOOT> to make parameter changes
- Obtains paths to all system disks and dump device (all system disk shadow set members need to be in the list)
- Configures LAN devices (LL, VL, IPCI), before clustering
- Loads MSCP for disk serving, before clustering
- Forms / joins cluster
- Activates local page files (if present)

- Hands over to SYS$SYSTEM:STARTUP.COM

OpenVMS post-boot startup

- SYS$SYSTEM:STARTUP.COM
- Startup database and phases (SYSMAN STARTUP)
- Devices (SYS$DEVICES.DAT and SYSMAN IO EXCLUDE)
- SY*.COM files (site specific, system platform actions):
  - SYCONFIG
  - SYLOGICALS
  - SYPAGSWPFILES
  - SYSTARTUP_VMS

- Application startup actions separated from system platform actions (disks, queues, databases, service addresses, etc.)

SYS$SYSTEM:STARTUP.COM

- STARTUP_P2 sysgen parameter, or
- SYSMAN STARTUP SET OPTIONS

- Enable log output to SYS$SPECIFIC:STARTUP.LOG

- Enable information output re startup stages (checkpoints)

- SYSMAN STARTUP SHOW OPTIONS

STARTUP database

- SYSMAN STARTUP SHOW FILE /FULL
- Some layered products add themselves to the startup database
- Personal preference – call layered product and application startup from SYSTARTUP_VMS.COM

SYCONFIG.COM

- Called very early in startup, also called by AUTOGEN, so check if running in startup mode

- Typical uses:
  - Disable SCS on unused paths with SCACP STOP LAN <dev>
  - Set RMS defaults
  - Configure Audit Server (eg: disable)
  - Configure DECdtm transaction journaling (eg: disable)
  - Configure DECwindows (eg: disable workstation components)
  - Configure DECnet-Plus (eg: prevent DECdns and DTSS starting on unused interfaces)

SYLOGICALS.COM

- Called after SYCONFIG

- Typical uses:
  - Set mini-copy / mini-merge policies
  - Check / mount system disk shadow set members
  - Mount common disk shadow set members
  - Define logical names for files on common disk (UAF etc.)

- Keep this for system platform logical names etc.

SYPAGSWPFILES.COM

- Called after SYLOGICALS

- Typical uses:
  - Mount page / swap / dump disks (eg: local SAS RAID)
  - Create directory structures and page / swap / dump files if needed (;32767 prevents modification)
  - Delete page / swap / dump files from system disk if they still exist (rename, then delete after reboot)
  - Install page / swap files

SYSTARTUP_VMS.COM

- Called at end of startup sequence

- Typical uses:
  - Mount additional shadow sets
  - Start network protocols
  - Start queue manager
  - Start layered products
  - Start 3rd party products
  - Call application startup sequence

Application startup after system startup

- Separated from system startup sequence
- Called by SYSTARTUP_VMS.COM as final action
- Easy to control application startup on boot with SYSGEN user defined system parameters, so no need to edit SYSTARTUP_VMS etc.
- Calls <APPNAME>_STARTUP.COM

Application startup sequence

- Top level command file <APPNAME>_STARTUP.COM
- Calls a set of command files for actions, eg:
  - <APPNAME>_DISK_MOUNTS.COM
  - <APPNAME>_LOGICALS.COM
  - <APPNAME>_BATCH_QUEUES.COM
  - <APPNAME>_PRINT_QUEUES.COM
  - <APPNAME>_ENABLE.COM

Agenda

1. Planning ahead
2. Hardware platforms
3. Network infrastructure
4. Storage infrastructure
5. Installation and setup
6. Boot, startup and shutdown

7. **Hints and tips**

8. Multi-site clusters

Monitoring and alerting – Why it matters

- Monitoring is essential in a high availability system with no single points of failure.

- HA and DT systems do their best to survive failures.

- The second failure kills you if you don't detect the first!

- You need to know what's going on, even if it's nothing

- You need good information to understand what happened when something fails

Monitoring and alerting

- Application availability, depends on:
  - All related physical equipment
  - Cluster interconnects
  - Storage volumes
  - Network connections
  - Performance behaviour
  - Queues, databases, 3rd party products, etc. …

- It’s the combination of data from different sources that enables you to understand what’s happening
- Implement periodic tests and checks

Console logging and monitoring

- Console logging can alert you to a developing situation or a transient problem.

- Consider how much OPCOM output you want – if logging all consoles, maybe restrict OPCOM output to be per-node

- Find a set of tools that works for you, for example:
  - DTCS includes IAM Consoles
  - Cockpit Manager
  - TDI Consoleworks
  - Networking Dynamics

DTCS for OpenVMS (now with DXC)

- Per-node software with:
  - Control of multi-site shadow set formation on boot, supporting up to 6-way shadowing
  - Rule based monitoring of cluster member nodes

- Windows management station (typically one per site) with:
  - Rule based monitoring:
    - storage arrays (WEBES, SNMP etc.)
    - Storage infrastructure (SNMP)
    - Network infrastructure (SNMP)
    - reachability (PING etc.)
  - Console access and logging via ILO
  - Alerts and notifications (email etc.)

Availability manager

- Windows management stations (typically one per site)
- Gives “real time” view of nodes in management group(s)
- Uses AMDS protocol (layer 2 – use LL or VL device)
- Interacts with OpenVMS driver at high IPL
- Permits modification of running system:
  - Quotas
  - Dynamic parameters
  - Quorum

Backups and data copies

- Shadow sets – up to six members
- Drop and reintroduce members for backup and data copy
- Mini-copy and mini-merge
- Use alternate members in rotation for data copy

- Use storage array snapshots or use shadowing ?

- Array based operations can reduce need for operating system based IO, but require careful synchronisation with other operations
- Beware SSH algorithm incompatibility

House keeping

- Disk space usage / free space
- File fragmentation
- Disk fragmentation
- Shadow set consistency
- Log files
- Audit files
- Backup / archive records
- Etc. …

- Don’t let log and audit files become too big and unwieldy
- Clean up files on a cycle time that makes sense in the context of your operational regime

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

BOOT_OPTIONS – BOOT and DUMP lists

- Use ANALYZE/SHADOW to check if all shadowset members are consistent
- Ensure that all members of a shadowed system disc appear in the boot and dump lists
- Need enough paths to each device for minimalist boot/dump drivers to write errorlog buffers
- Can prune the list to make it less complicated

Avoiding data corruption

- When adding a new IA64 node to a cluster, mount target system disc READ ONLY before using BOOT_OPTIONS to set up boot paths

- Use SET DEVICE /NOAVAILABLE to prevent access to discs and other devices if <u>not</u> booted as a cluster member but with discs visible to booted node (eg: stand-alone backup or local maintenance boot)

Devices

- Fibrechannel tape:
  - SYSMAN IO FIND_WWID and IO CREATE_WWID
  - See SYS$DEVICES.DAT

- Exclude unused devices:
  - SYSMAN IO SHOW EXCLUDE
  - SYSMAN IO SET EXCLUDE = “<list>”

- LANCP DEFINE LL<nn> / VL<nn> etc.

SYSMAN IO EXCLUDE list

- Disable unused devices to prevent allocating memory resources for the data structures and buffers. Especially important on big blades for unused ethernet devices.

- Also useful for security, eg: prevent USB device use.

- To disable the USB subsystem drivers:

```
      mc sysman io set exclude=(UH*,EH*,DNA*,USB*,UCM*)
```

Fastpath

- Try to keep all interrupt handling on the same socket as the primary CPU (0)
- Think about what happens if hyperthreads are enabled
- Same device types on the same CPU (core)

- Example (-i4 or –i6, threads = off):
  - CPU 0 = primary
  - CPU 1 = ERA, ERB, BG
  - CPU 2 = EIA, EIB, EIC, EID, PE
  - CPU 3 = FGA, FGB, FGC, FGD
  - CPU 4 = PKA, PKB, PKC, PKD

Lock Manager

- Dedicated CPU for lock manager may be useful
- Try to keep it on the same socket as the primary CPU (0)
- Also consider fastpath CPU settings
- Think about what happens if hyperthreads are enabled

TCPI/IP stack

- TCPIP (BG device) on the same CPU as the network interfaces
- Keepalives can solve a lot of dropped connection problems
- SYSCONFIGTAB mechanism and “stanza file”:

```
socket:
        somaxconn=65535
        sominconn=65535
inet:
        tcbhashsize=16384
        tcbhashnum=16
        tcp_keepalive_default=1
        tcp_keepidle=1200
```

Memory reservations

- Useful to pre-allocate memory regions
- Allows placement of regions in specific RADs on a NUMA machine
- Large memory machines can benefit from bigger XFC
- Large memory machines can benefit from RAMdisc
- Some applications / databases use global sections, which can be sized and placed in RADs appropriately

External authentication

- Allows integration with a Windows AD or similar
- Set up ACMELDAP so that VMS can reach the AD for authentication
- Kits and configuration file templates in SYS$UPDATE:
- Map VMS usernames to AD entry fields
- Set “EXTAUTH” flag in UAF

DECwindows

- Do you need DECwindows at all ?

- Prevent using ILO as graphics device for DECwindows:

  mc sysman io set exclude=(GHA*)

  Note: device name is hardware platform specific

- Disable workstation components on startup

Performance engineering

- Without good data you cannot do good performance work

- Avoid guesswork - run T4 all the time
- If needed, use T4 “expert mode” and SDA extensions

- A faster machine just waits more quickly!
- Don’t make it go faster, stop it going slower
- The fastest IO is the IO you don’t do
- The fastest code is the code you don’t execute
- The idle loop is anything but idle

Performance monitoring

- Free T4 for long-term monitoring and trend analysis
- Implement your own modules to instrument your application

- Availability Manager (AMDS)

- More comprehensive performance monitors, eg:
  - TDC
  - Perfdat

- Find a set of tools that works for you.

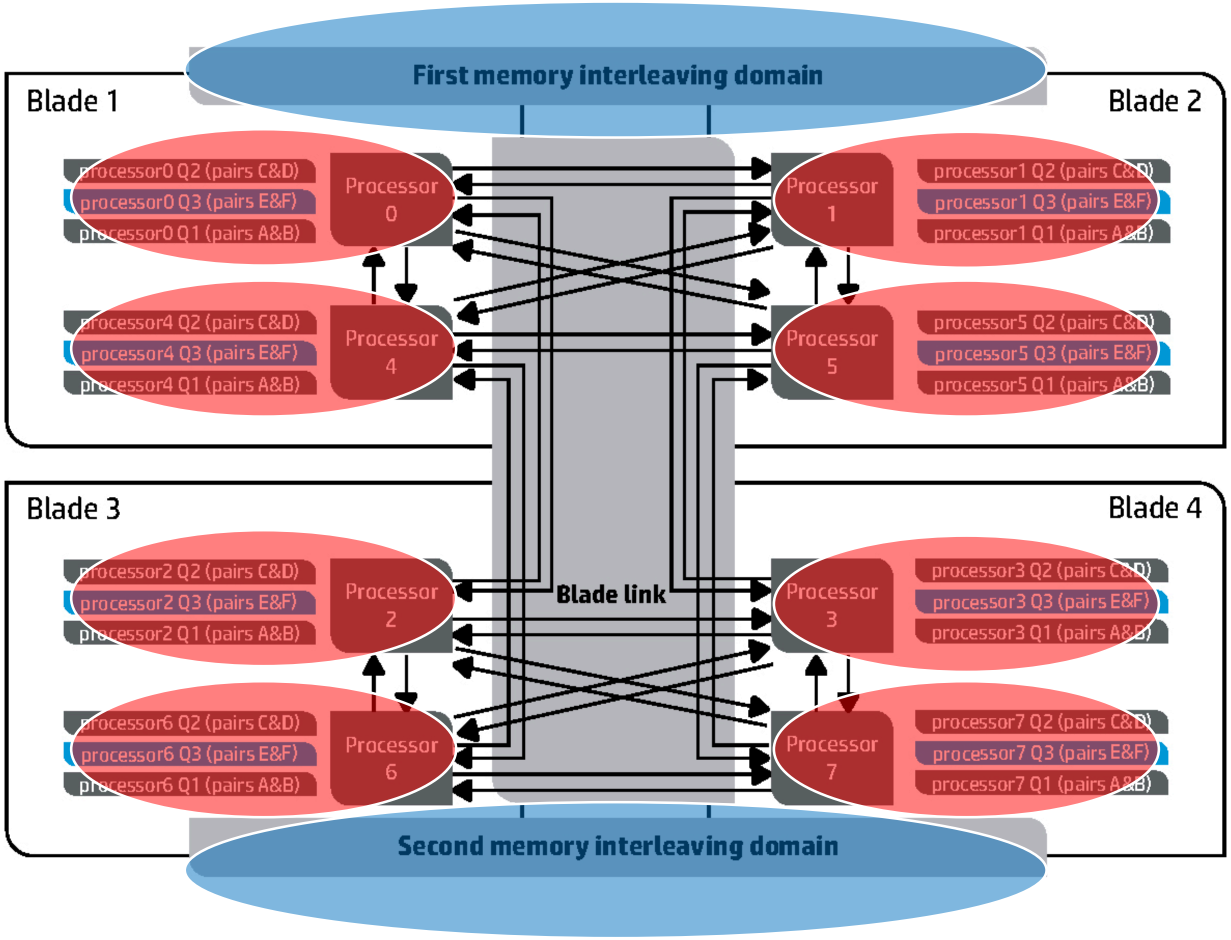
Contention for resources

- Many processes running in parallel can create contention for access to data files and shared data structures
- Look for high MP SYNC (or disguised MP SYNC which may show up as high INTERRUPT or KERNEL modes)
- Fast systems with a high level of parallelism can create conditions where a lot of things ”bunch up” and the overall effect is to slow the whole system down
- Look closely at the workload and flow of activities through the system

NUMA (non-uniform memory access)

- OpenVMS uses large shared memory regions:
  - XFC (50 % available memory by default)
  - RMS global buffers
  - Global sections (especially database caches)
  - Memory disc driver (MD devices)
- Set memory interleave behaviour with “memconfig” at EFI shell (requires reboot)
- Ensure physical memory installation is correct!
- Useful starting point for OpenVMS is “mostly UMA”

Memory architecture – bl890c-i4


First memory interleaving domain
Blade 1
Blade 2
processor0 Q2 (pairs C&D)
processor0 Q3 (pairs E&F)
processor0 Q1 (pairs A&B)
Processor 0
Processor 1
processor1 Q2 (pairs C&D)
processor1 Q3 (pairs E&F)
processor1 Q1 (pairs A&B)
processor4 Q2 (pairs C&D)
processor4 Q3 (pairs E&F)
processor4 Q1 (pairs A&B)
Processor 4
Processor 5
processor5 Q2 (pairs C&D)
processor5 Q3 (pairs E&F)
processor5 Q1 (pairs A&B)
Blade 3
Blade 4
processor2 Q2 (pairs C&D)
processor2 Q3 (pairs E&F)
processor2 Q1 (pairs A&B)
Processor 2
Blade link
Processor 3
processor3 Q2 (pairs C&D)
processor3 Q3 (pairs E&F)
processor3 Q1 (pairs A&B)
processor6 Q2 (pairs C&D)
processor6 Q3 (pairs E&F)
processor6 Q1 (pairs A&B)
Processor 6
Processor 7
processor7 Q2 (pairs C&D)
processor7 Q3 (pairs E&F)
processor7 Q1 (pairs A&B)
Second memory interleaving domain

Hyperthreading

- Hyperthreading is extremely workload dependent

- In general the OpenVMS scheduler does a better job

- Enable / disable hyperthreading with “cpuconfig” at EFI shell (requires reboot)

- “CPU” count will appear to double when enabled
  - V8.4-2 supports 64 “scheduling units”
  - V8.4-1H1 supports 32 “scheduling units”
  - Can “STOP/CPU <nn>” to selectively shut down co-threads

Agenda

1. Planning ahead
2. Hardware platforms
3. Network infrastructure
4. Storage infrastructure
5. Installation and setup
6. Boot, startup and shutdown
7. Hints and tips

8. **Multi-site clusters**

Inter-site links - storage

- MSCP over PEdriver:
  - Poor for performance, useful fallback
  - Requires cluster member nodes at remote site
- FC extension (DWDM over dark fibre):
  - Best for performance
  - Set inter-switch link port characteristics
- FC over IP:
  - Need low latency and low jitter IP network
  - Use “fast write” in FCIP devices

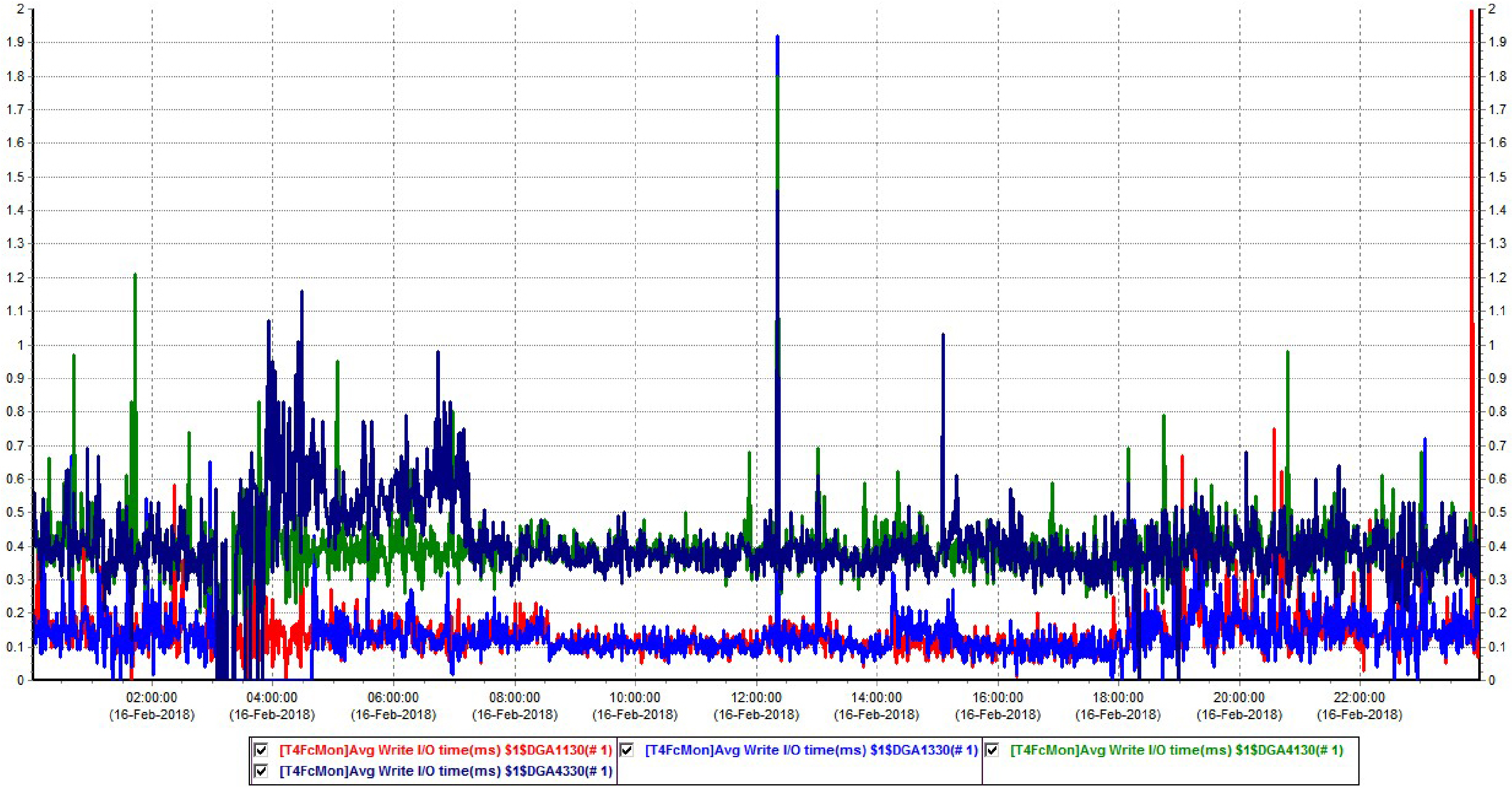
Multi-site cluster – storage

- MSCP over PEdriver:
  - No direct path to MSCP served devices at remote sites
  - Greater probability of triggering shadow merge
- FC extension and FC over IP:
  - Multi-path fibrechannel devices
  - Direct path to devices from all nodes
  - Set the SITE value to bias reads from local site

Multi-site cluster – storage performance

- Reads should be biased from local storage
- Shadowing writes are synchronous, so remote shadow set members are the limiting factor.
- Write latency is a combination of:
    - Local IO write latency (system to local storage array)
    - ISL endpoint latency (FC switches etc.)
    - Distance latency
- IO write rate: 1ms = 1000 IOs per second, 2ms = 500, …

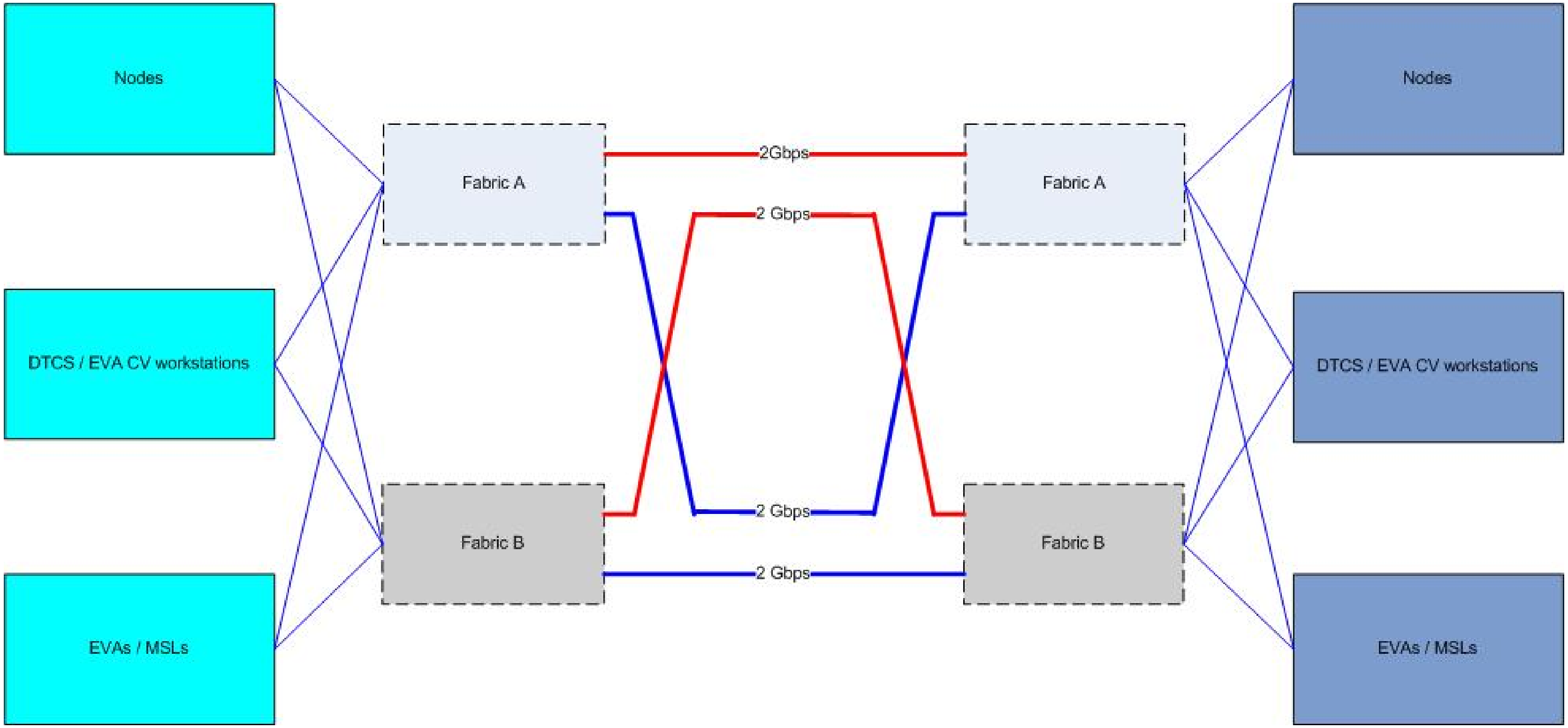
Multi-site shadow set (4 members)



Inter-site storage (SAN) links

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

Example SAN connectivity



Inter-site links – data networking (1)

- 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

Inter-site links – data networking (2)

- LAN extension – layer 2 (DWDM over dark fibre):
  - Best for performance, lowest latency
  - Passes all layer 2 protocols, so use SCS, AMDS, etc.

- LAN extension – layer 2 (MPLS pseudo-wire):
  - Variable latency, depends on topology and contention
  - Passes all layer 2 protocols, so use SCS, AMDS, etc.

- TCP/IP – layer 3:
  - Variable latency, depends on topology and contention
  - IP only, use IPCI, no AMDS or other layer 2 protocols

Extended layer 2 LANs

- DWDM over dark fibre
- MPLS

- Traffic separation with VLAN 802.1Q tags

- Use QoS to control traffic flows

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

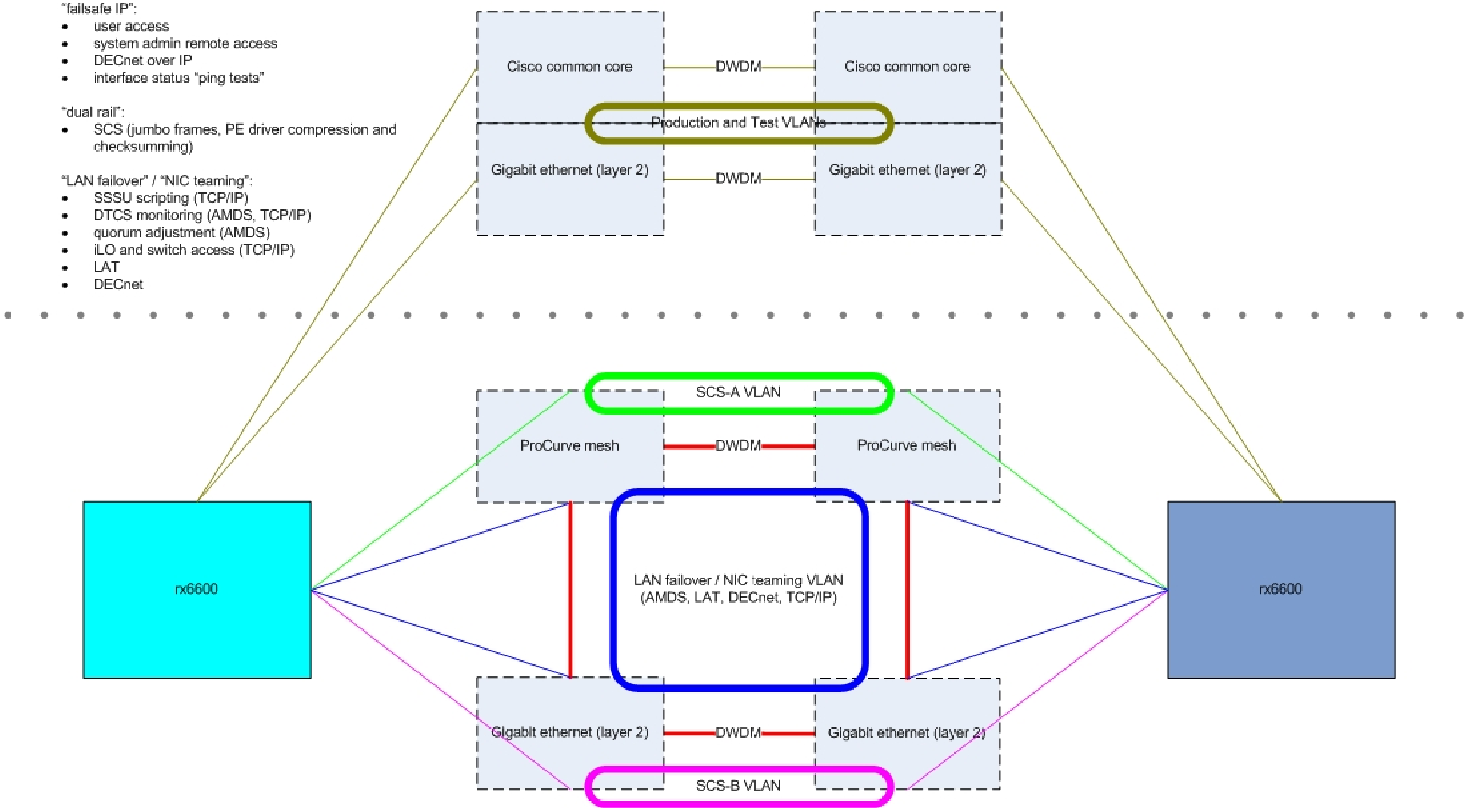
Multi-site cluster – data networking

- LANCP to configure and manage LAN devices
- LLDRIVER for LAN failover
- VLDRIVER for VLANs

- PEDRIVER implements channels and virtual circuits
- SCACP to manage PEDRIVER
- Disable SCS on unused LAN devices

- Enable jumbo frames if network infrastructure permits

Example data network connectivity



"failsafe IP":

- user access
- system admin remote access
- DECnet over IP
- interface status "ping tests"

"dual rail":

- SCS (jumbo frames, PE driver compression and checksumming)

"LAN failover" / "NIC teaming":

- SSSU scripting (TCP/IP)
- DTCS monitoring (AMDS, TCP/IP)
- quorum adjustment (AMDS)
- iLO and switch access (TCP/IP)
- LAT
- DECnet

Cisco common core
DWDM
Cisco common core
Production and Test VLANs
Gigabit ethernet (layer 2)
DWDM
Gigabit ethernet (layer 2)
SCS-A VLAN
ProCurve mesh
DWDM
ProCurve mesh
rx6600
LAN failover / NIC teaming VLAN
(AMDS, LAT, DECnet, TCP/IP)
rx6600
Gigabit ethernet (layer 2)
DWDM
Gigabit ethernet (layer 2)
SCS-B VLAN

Multi-site cluster – cluster interconnects

- How many channels should we have?
- Depends on network infrastructure

- Single LAN failover device – simple, single channel, multiple NICs, performance and failover behaviour depends on LLDRIVER

- Multiple LAN devices – more NICs (or VL devices), multiple channels, better performance and failover behaviour

- VLANs – tagged (VLDRIVER) or untagged?

Best practices for OpenVMS clusters

The design, build, operation and support of OpenVMS systems and clusters

Connect Germany 2021 online conference

Colin Butcher CEng FBCS CITP