HP OpenVMS European Technical Updates 2012

Practical experiences with OpenVMS on big blade servers (BL8x0c-i2)

Colin Butcher FBCS CEng

XDelta Limited

- Project overview
- Performance – general principles
- Porting to Integrity - summary
- Surrounding infrastructure and BL8x0c-i2 setup
- Installing and configuring OpenVMS on BL8x0c-i2
- Performance and system behaviour
- Q & A

Part 1 – Project Overview:

- AlphaServer GS1280s are close to being maxed out
- Want enough performance to cater for several years business growth
- Application is complex and has grown over time from microVAX to DS25s to GS1280s, now to BL890c-i2s
- Prefer to stay with OpenVMS
- Initial results with BL890c-i2 are encouraging

Why BL890c-i2 platform ?

- Used customer lab to test application on various hardware platforms with real code and real data
- Initially considered a cluster of rx2800-i2 nodes
- However, application scaling behaviour indicated that a high CPU count system would perform better
- Tested rx2800-i2, BL870c-i2 and BL890c-i2

- Final choice was BL890c-i2 and EVA P6500 storage

Part 2 – Principles of Performance:

- 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 time) – determines predictability of response
  - Understanding jitter is important for establishing timeout values
  - Latency fluctuations can cause system failures under peak load

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

- Understand scalability – do as much as possible once only, do little as possible every time

- Size systems to cope with peaks in workload
- Minimise contention for resources and access to data structures
- Understand the need for synchronisation and serialisation of access to data structures
- Understand implications of NUMA
- Don't make it go faster, stop it from going slower!
  - The fastest IO is the IO you don't do
  - The fastest code is the code you don't execute
  - Order workflow to minimise contention
  - Use memory to improve performance

Limits to performance

- Ability to make use of hardware parallelism
- IO bandwidth and latency
- Network bandwidth and latency
- Ability to make use of large memory machines
- Infrastructure design
- Application design
- Compilers

- Designing and building very good systems requires very good people

Part 3 – Porting to Integrity

- Integrity is very different from Alpha and VAX, so we should make appropriate changes where needed
- Setting the new systems and surrounding infrastructure up for good performance and reliability is one of the hidden aspects of porting applications
- We don’t want to spend our time managing performance and doing tuning exercises
- We don’t want to spend our time chasing problems

- Compilers are the key to performance
- Use the documented mechanisms provided by the operating system – read the release notes and new features, then use the current mechanisms
- Contention for resources on large systems
- Data alignment (unexpected alignment faults)
- Exception handling (LIB$SIGNAL etc.)
- Floating Point format (IEEE by default)
- Implicit assumptions (eg: uniprocessor; .not. Alpha)
- Debugging - ELF & DWARF formats
- Integrity calling standard and register usage

- Many processes running in parallel can create contention for access to files and data structures
- Image rundown and VA teardown is expensive (MMG spinlock usage)
- Many short-lived processes with big VA requirements (eg: RMS global buffers) can cause a lot of MP_SYNC
- Spinlock tracing (with SDA extension)
- 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

- Fixing up an unexpected alignment fault currently requires the MMG spinlock which affects the entire system performance - note that work is in progress to reduce or eliminate this (ask for the kit)
- Use the appropriate compiler switches if they are available (not all compilers have this, eg: BASIC)
- MONITOR ALIGN will show if you have issues
- Alignment fault tracing (with SDA extension)
- Spinlock tracing (with SDA extension)
- Look for high MP SYNC (or disguised MP SYNC which may show up as high INTERRUPT or KERNEL modes)

- Exception handling (LIB$SIGNAL etc.) is a more expensive mechanism than it was on Alpha
- Consider alternatives if your code makes extensive use of exception handling as part of it's normal flow of control
- Increase stack space when using threads

- Integrity default floating point format is IEEE
- VAX floating point is handled by converting to IEEE format, doing the processing, then converting the result back from IEEE format to VAX format, hence a small performance penalty
- Alphas do both VAX and IEEE, but the default is VAX
- Results with IEEE are ‘slightly different’, not by much
- Consider testing for a small difference, not equal
- Beware LIB$WAIT(<real>)!
- Beware reading in binary data files / writing out binary data files and moving them between different systems – consider re-writing files if you change FP format

- Check for code making assumptions about hardware:
  - “if not Alpha” is definitely a problem!
  - “if not VAX” should be less of a problem
  - “if VAX” / “if Alpha” / “if IA64” is safe

- Check for code (and DCL and system parameters) making assumptions about page sizes:
  - Memory page size can vary on Integrity depending on the CPU
  - Alpha is 8192 bytes
  - VAX is 512 bytes
  - Also think about storage array controller caches, RMS block counts, HBVS block counts, etc.

- Check for code making assumptions that the system is a uniprocessor machine:
  - Flag bits controlling access to an entire global section
  - Loops polling for flag bit status changes (spinlocks)
  - Data structures not protected from operations that may happen in parallel instead of sequentially
- Use the lock manager to serialise and synchronise access to data structures
- Minimise wait states by having appropriate granularity of access to data structures
- Take null locks out, then simply convert them as needed

Other aspects of porting

- Make a thorough audit and analysis of what you're doing at the moment
- Consider new features and new ways to do things
- There may be no direct equivalents
- Direct 'bug for bug' port or re-implement application ?
- Are you going to have to continue to support non-migrated systems as well ?
- You will need <u>all</u> of the source code !
- Test your complete software build process first
- How will you test the whole system ?

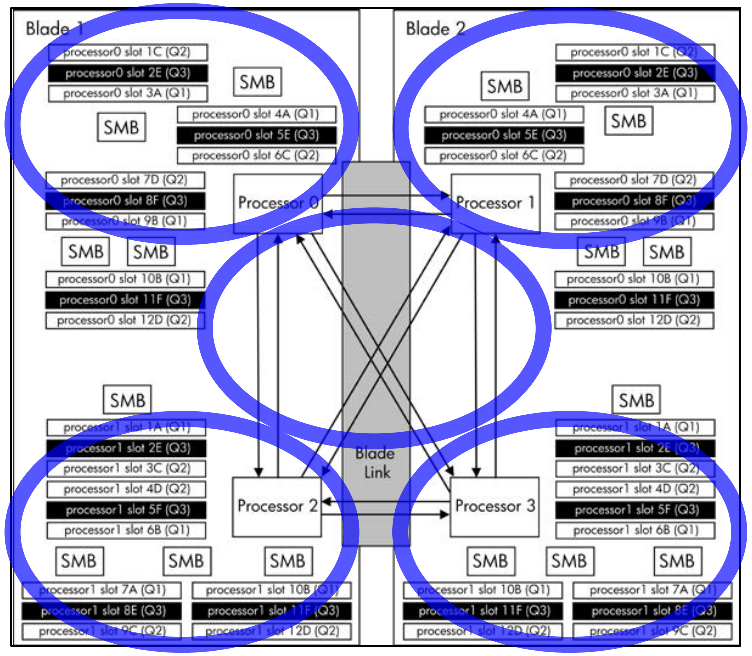
Part 4 – Systems Infrastructure:

Simplicity of configuration

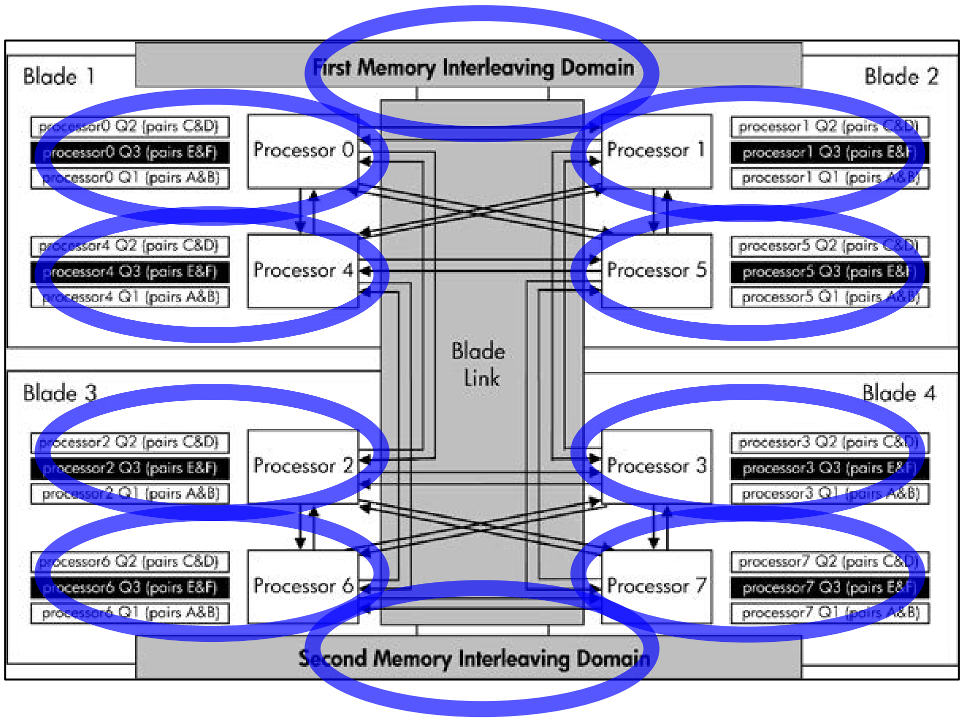
- Multiple instances of the complete system for high availability (failover between instances)
- An instance consists of the main OpenVMS core systems, plus surrounding Linux servers and EVA / 3PAR storage arrays
- Not a cluster - multiple independent nodes within an instance for availability and capacity (failover between nodes, workload distribution between nodes)

- Server / chassis configuration:
  - 2x c7000 with 1x BL890c-i2 + 1x BL870c-i2 (256GB) per chassis
  - Pass-through modules for 10GigE (16x 10GigE LOM NICs)
  - Embedded switches for 8GigFC (4x 8GigFC HBA ports)
  - local SAS storage for 1st embedded controller only (note: no BBWBC available, so consider SSDs)
  - 20+ Proliant DL380-G7 Linux (RHEL) servers
- Storage configuration:
  - EVA P6550s (OpenVMS servers)
  - 3PAR (currently Linux servers)

BL870c-i2 – 5 RADs



BL890c-i2 – 10 RADs



Diagrams from 4AA1-1126ENW.pdf (bl8x0ci2 memory subsystem)

Part 5 – Configuring the system:

- Power-up / boot / dump / restart time
- EFI shell as "default boot" ?
- Disable boot timeout ?
- Disable memory tests on power-up ?
- NUMA memory configuration and RADs
- Embedded SAS: no BBWBC, HBA and RAID modes
- EFI shell "map" and "alias" commands are useful
- EFI / MP / iLO configuration
- Remote monitoring / management
- Firmware update "load host" on network

- Where to put page / swap files ? Local or FC ?
- Where to put crash dumps ? Local or FC ?
- Where to put T4 data, log files, etc. ? Local or FC ?
- LD containers for log files etc. to avoid fragmentation and "house keeping"
- Compressed selective dumps with very large memory
- Unused devices in SYSMAN IO EXCLUDE list
- Flag FSnn: devices at EFI shell level with your own "marker file" (you can even write them from VMS by using the EFI partition tools – but be careful!)

Making use of CPU

- Introduce parallelism into your code and batch jobs where possible – understand the workflow
- Hyperthreads may be useful in some circumstances
- Fastpath IO devices and interrupt handling – set all devices of the same type to the same preferred CPU
- Dedicated CPU for lock manager (local locking)
- Dedicated CPU for TCPIP PPE (enable PPE, then set preferred CPU for BG device)
- Consider CPU affinity for key application processes
- Consider using /RAD=n for batch execution queues

- NUMA and RADs: “memconfig -mi” (EFI shell)
- Use memory (XFC, resident images, DECram etc.)
- Revisit working set sizes - WSMAX and process quotas (WSDEF / WSQUO / WSEXT / PGFLQUO + PQLs)
- Use RMS global buffers (maybe even 100%)
- Revisit RMS system defaults (multiples of 16 blocks)
- GH regions – map lots of memory with a small number of page table entries
- XFC memory reservations and RADs
- INSTALL /RESIDENT and GH region size
- DECram and RADs

Making use of network IO

- Pick IO ports (EW devices) to use
- LAN failover(LLdriver and LANCP)
- 802.1Q VLAN tagging (VLdriver and LANCP)
- Network protocols: SCS, TCP/IP, DECnet-Plus, LAT, MOP, AMDS, etc.
- Split protocols and data flows across adapters
- Revisit RMS block sizes for network IO
- Jumbo frame size 7500 (LANCP, see release notes)
- Fastpath preferred CPU
- Dedicated CPU for TCP/IP Packet Processing Engine

LAN failover and VLAN tagging

- Jumbo frames (LAN_FLAGS bit 6)
- Fast LAN transmit timeout (LAN_FLAGS bit 12)
- VLAN driver workaround (LAN_FLAGS bit 14)
- VL / LL / EW workaround (EW device buffers >= 128)

- We have 5x VLAN devices (VLA, VLB, VLC, VLD, VLE) for the different data flows, which map to 2x LAN failover devices (LLA, LLB) for throughput, which map to 4x NICs (LLA = EWC + EWK, LLB = EWH + EWP)

- Unused EW devices in SYSMAN IO EXCLUDE list
- Disable DTSS, DECdns on a per-NIC basis (logicals)

- EVA cache size and OpenVMS volume characteristics (cluster factor, extend quantity, dynamic volume expansion etc.) – 16 disc blocks = 8192 bytes = memory page size for Integrity (to date) and Alpha.
- Revisit RMS block sizes and shadow copy block count
- Consider SSDs for embedded SAS devices (no BBWBC on embedded SAS controllers)

Making use of FC storage

- FC bandwidth is important – what else are you sharing your storage bandwidth with? Why?
- Fastpath preferred CPU – pick same CPU for same devices, pick CPU closest to operating system data structures
- Explicit FC path selection and EVA preferred controller
- Place the data across many DG devices to get best overall IO throughput
- Use Vraid1 for all heavily used DG devices
- Typical disc IO response from lightly loaded EVA P6500 is 0.3 to 0.5 msec

Part 6 – Hints and tips:

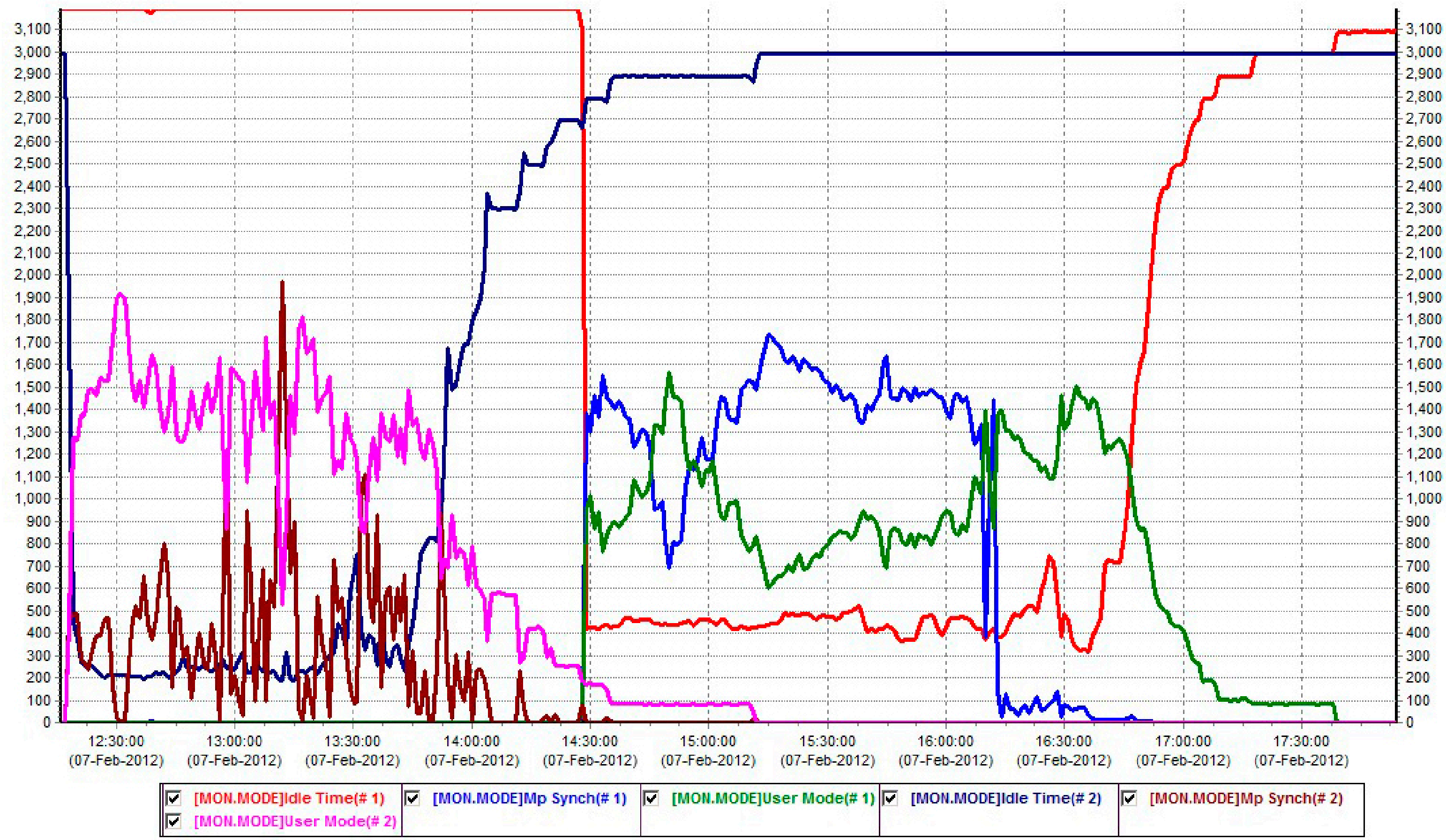
- Leave T4 running all the time. It is invaluable to have a record of system behaviour for comparison over time as the applications and workload change. Without data you're left guessing and making assumptions.

- For more detailed performance investigations, use other tools as needed.

- Use EVAperf (or equivalent tools) to monitor storage array behaviour

- Make sure your data is synchronised in time

Effect of dedicated CPU for LCKMGR



The mission-critical systems architects

Stuff that works

- Workflow ordering in extremely important – many improvements came for a better understanding of the application as a result of the porting work

- Large memory NUMA Integrity systems behave very differently to Alpha – it is easy to create contention issues between various parts of the application

- Look for MMG spinlock contention on image rundown with many short-lived processes and large working sets. Beware setting VHPT_SIZE big and spending a lot of time in MMG spinlock during VA teardown.

Part 7 – Summary:

Results

- BI890c-i2 based system is significantly faster than GS1280 based system
- Startup time from cold needs to be improved, as does time to reach EFI shell when needed
- Found several bugs, which have been fixed
- Whole system and application is now much better understood by development team
- Don't underestimate how long a porting project like this can actually take!

What worked well for us ?

- Dedicated CPU for lock manager
- Dedicated CPU for TCPIP PPE
- Explicit fastpath CPU selection (EW, FG devices)
- Use all that memory!
- Re-ordering the application workflow to reduce periods of contention made a big difference
- Going through the process of building the complete system from scratch flushed out a lot of previously hidden issues
- Involving the application developers, systems people and support people with them all working together has made a big difference

- HP Blade system documentation (c7000 chassis, IO mapping, pass-through modules, embedded FC switches, Flex-10, Virtual Connect, etc.)
- HP Integrity Blade documentation (bl8x0c-i2 maintenance and service guide, etc.)
- Firmware release notes etc.
- OpenVMS V8.4 operating system documentation
- Patch kit release notes etc.
- HP technical white papers (bl8x0c-i2 memory subsystem, bl8x0c-i2 technologies, bl8x0c-i2 scalable blades, OpenVMS NUMA programming, etc.)

- www.xdelta.co.uk/seminars – webinars and slide sets

Thank you for your participation.

If you have questions, please ask!

Colin Butcher, XDelta Limited

www.xdelta.co.uk