

Netherlands OpenVMS Technical Update

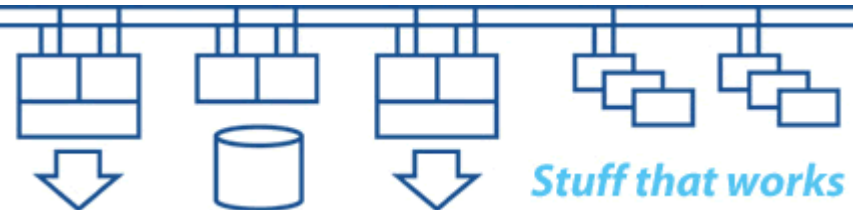
Migrating to Integrity

Colin Butcher

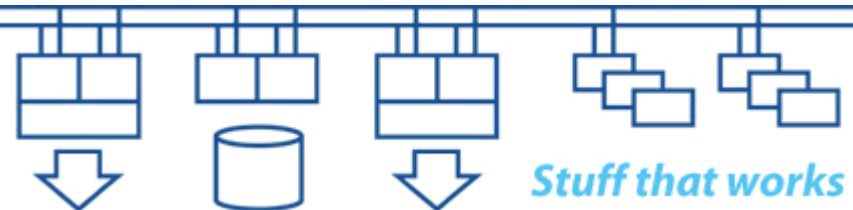
Copyright © Colin Butcher, XDelta Limited, October 2007

Web: <http://www.xdelta.co.uk>

E-mail: vms@xdelta.co.uk



- **Naming conventions**
- **Hardware platforms**
- **Console subsystems**
- **Blade systems**
- **Overall system design issues**
- **Software design and implementation issues**
- **Emulation and Binary Translation**
- **Migration and Porting guidelines**

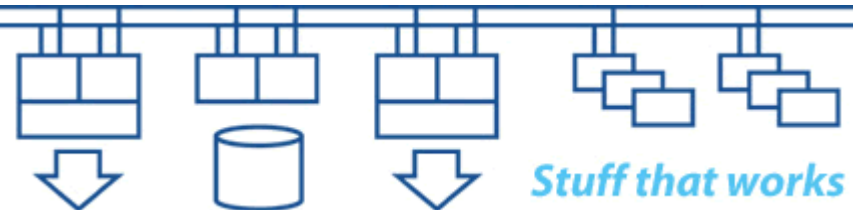


VAX and Alpha are used interchangeably to refer to system architectures (the instruction set), CPU / microprocessor implementations (eg: nVAX, EV7) and complete system platforms (eg: VAX 7000, AlphaServer GS1280):

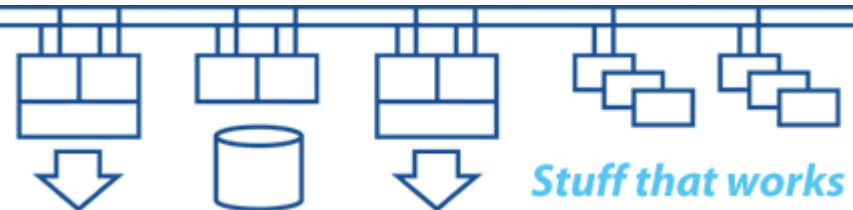
- **VAX VMS was used up to V5.5-2H4**
- **OpenVMS VAX and OpenVMS Alpha are used from V6.0 onwards with the introduction of Alpha**

Intel implementations are different:

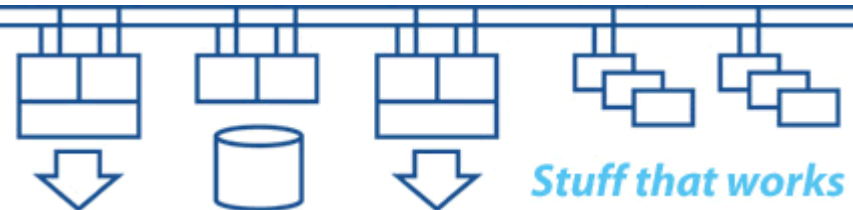
- **Itanium Architecture is the system architecture**
- **Itanium is the Intel microprocessor family (hence IPF)**
- **Integrity Server is the HP platform**
- **OpenVMS for AlphaServers and Integrity Servers is the HP operating system name**



- **64bit EPIC. Designed when memory is plentiful and cheap**
- **Itanium Processor Family architecture relies on compiler output to generate an efficient instruction and data flow through the CPU**
- **Instructions are packaged in “bundles” of up to three instructions per bundle – which are then processed entirely in parallel**
- **Needs synchronisation and serialisation issues to be carefully considered and coded**
- **Data alignment is important**
- **Floating point is IEEE format by default**

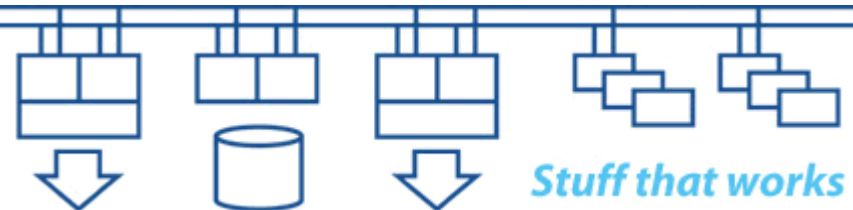


- **No “VAX like” or “Alpha like” console**
- **Has multiple consoles:**
 - **Management Processor (MP) and Baseboard (BMC)**
 - **Serial and Web (ILO) interfaces**
 - **Both attempt to be consistent across the entire range (only real difference is partitioned systems)**
- **Uses Extensible Firmware Interface (EFI) rather than BIOS**
- **PCI bus based (3.3volt only)**
- **PCI Express devices**
- **SAS (Serial Attached SCSI) devices**
- **Multi-core processors and HyperThreading**

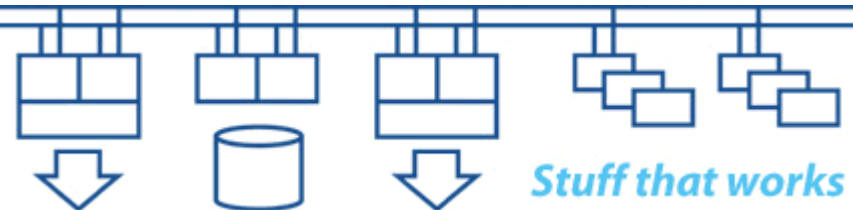


- **What is Partitioning?**
- **What is a Cell?**
- **What is a Socket?**
- **What is a Core?**
- **What is HyperThreading?**

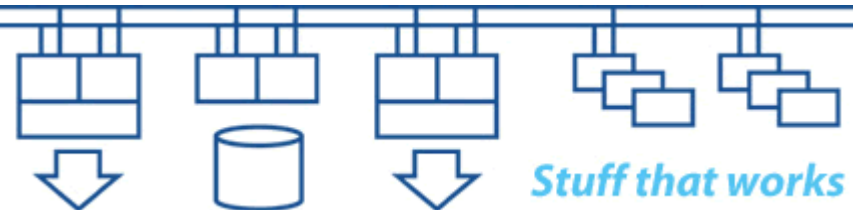
- **What does the operating system see?**
- **What are the performance implications?**
- **How is it licensed?**



- **Used to be small, slow and expensive, so tried hard to minimise memory usage**
- **Now plentiful, fast and relatively cheap**
- **Can use memory to gain performance**
- **Caches and synchronisation**
- **Error detection and correction**
- **“Locality” to CPUs**
- **Memory interconnects to CPUs**

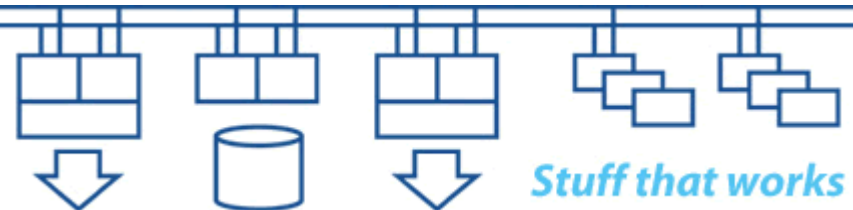


- **Provides interface with outside world (physical, electrical, logical ...)**
- **Connects to storage subsystems**
- **Access times (memory, cache etc.)**
- **“Locality” to CPUs**
- **IO devices operation (interrupts, registers)**
- **IO interconnects to CPU and memory subsystems**
- **Disc subsystems (e.g.: EVA) depend on bandwidth in the fibrechannel path**

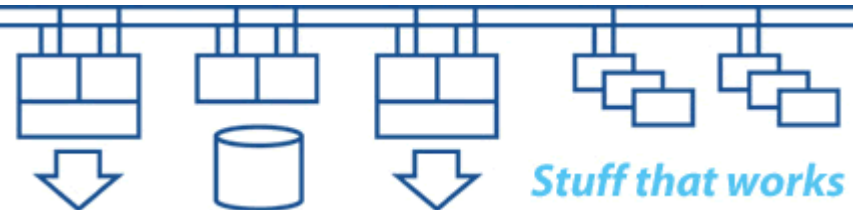


- **MP serial console for “Management Processor”**
 - **Runs with base box level power, even with system off**
 - **Local, remote (modem) and network connectivity**
 - **Console configuration (terminal type, etc.)**
 - **<ctrl-H>, <ctrl-B>**
 - **Network configuration (hostname, IP address, etc.)**
 - **Multiple console sessions (one writer, many readers)**
 - **Provides ability to copy files over network with TFTP (firmware updates etc.)**
 - **Access to local USB devices for firmware updates**

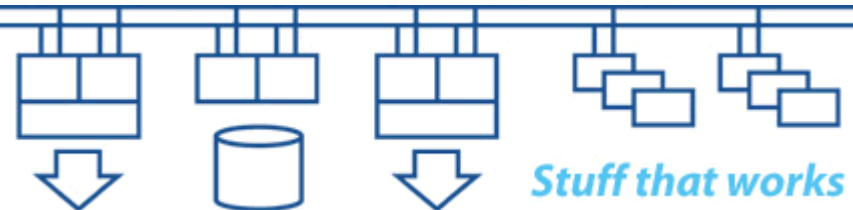
Note: Set your terminal emulator up for delete / <ctrl-H> in VT100+ mode



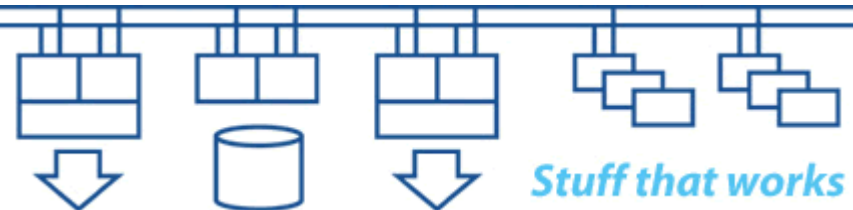
- **MP ILO (web) console for “Management Processor”**
 - **Runs with base box level power, even with system off**
 - **Access MP through web browser**
 - **Private MP LAN connection**
 - **Initial configuration of MP LAN via MP serial console (“LC” and “LS” commands at “CM” level)**
 - **VT terminal emulation**
 - **Redirects from HTTP to HTTPS**
 - **Currently not remotely accessible through a NAT firewall / router**



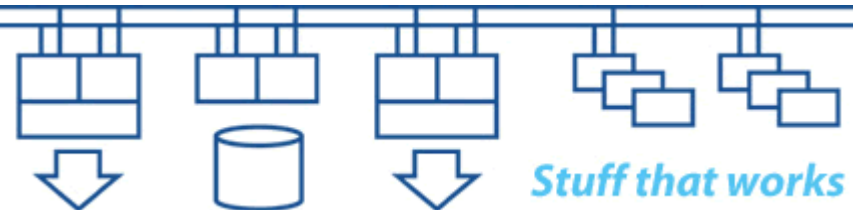
- **EFI (extensible firmware interface):**
 - Mini operating system
 - FAT formatted file system for EFI scripts and other files (eg: firmware updates)
 - OpenVMS currently presents a FAT16 partition to EFI
 - Boot menu and defaults
 - Environment variables are sticky (VMS_FLAGS etc.)
 - VMS_SHOW.EFI – shows device names in VMS format
 - VMS_LOADER.EFI - finds and loads IPB.EXE
 - Data passed to boot loader, eg: efi> vms_loader -fl 0,0
 - BOOT_OPTIONS.COM for FibreChannel devices on OpenVMS installation DVD
- **IPB.EXE (boot loader) understands the OpenVMS file structure, EFI does not**



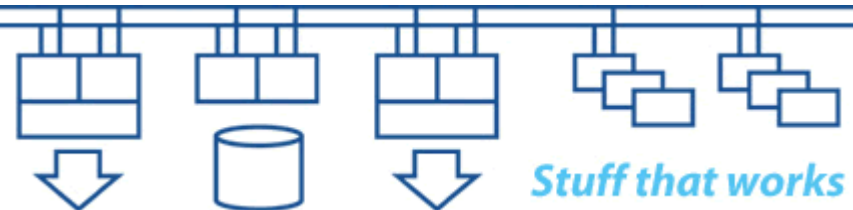
- **EFI boot loader from FAT partition (hidden as a container file in the system disc file structure)**
- **Boot flags passed through environment variables**
- **Reads executive into memory**
- **Reads system parameters (SYSBOOT> if flags set) and initialises fixed data structures**
- **Passes control to executive**
- **Structures created with SET BOOTBLOCK command**
- **2048 byte boot block for IDE/ATAPI optical media**
- **512 byte boot block for SCSI optical media**



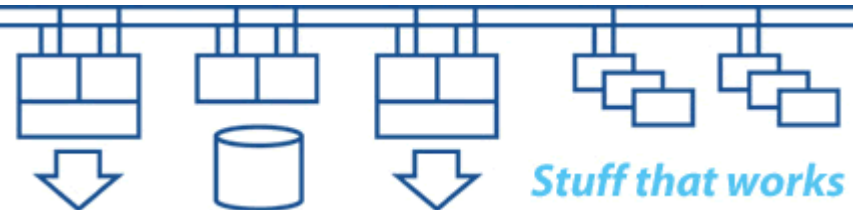
- **Itanium Processor Family architecture uses ACPI (Advanced Configuration and Power Interface) for device detection by firmware**
- **SYSMAN IO AUTO queries ACPI data to find devices and set up OpenVMS device drivers to communicate with the hardware**
- **Devices appear as a set of CSRs (Control and Status Registers) in physical memory - the IO space**
- **Devices have Interrupt Vectors which connect a device interrupt request to the device driver Interrupt Service Routine**



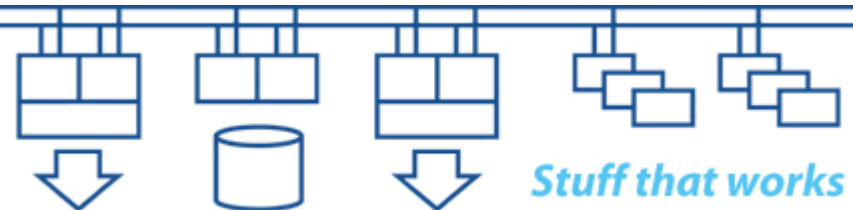
- **Think DEChub900 for systems!**
- **Common chassis / backplane / power / cooling etc.**
- **Modular components for processors, storage and network switching**
- **Virtual interconnects capable of re-assignment**



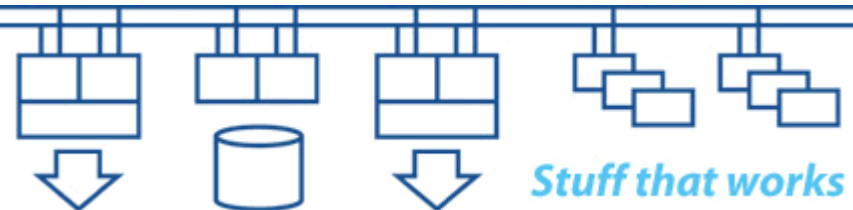
- “low end” DS15/25 equivalents e.g.: rx2660/rx3660
- “mid-range” GS80/ES47 equivalents: e.g.: rx6600/rx7640
- “high-end” GS1280 equivalents: e.g.: rx8640/SuperDome
- “blades” OpenVMS on the c-class (BL860C)
- Consistent console interface across all platforms (except no partitioning on low-end systems)
- See www.hp.com/go/integrity
- See Golden Eggs at <http://goldeneggs.spyderbyte.com> by Matti Patari



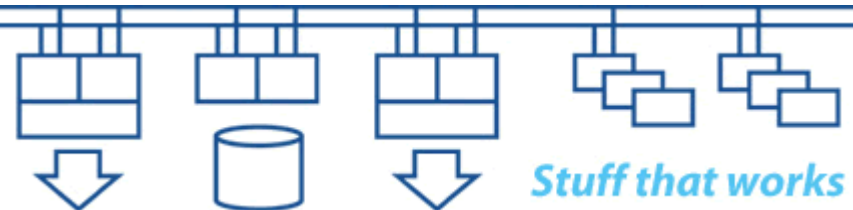
- **Compilers are the key to performance**
- **Exception handling (LIB\$SIGNAL etc.)**
- **Data alignment (unexpected alignment faults)**
- **Floating Point format (IEEE by default)**
- **Implicit assumptions**
- **Debugging**
- **ELF & DWARF formats**
- **Integrity calling standard and register usage**



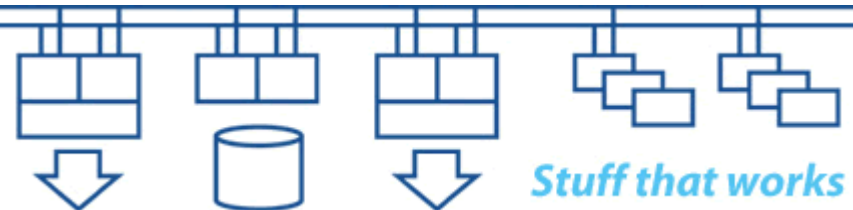
- **Currently a mixture of Intel standard compilers and GEM back-end compilers, depending on the language**
- **Defaults are likely to be different on Integrity, so explicitly set the command line switches to get the behaviour you need**
- **Check the SPDs for the compilers you need**
- **Test compile some of your code early on – get to the latest versions first if you can before you change platform**
- **DEC ADA'83 not available – use Gnat ADA Pro**
- **Standard C is Standard C**
- **Fortran DEC extensions are (mostly) supported**



- **Unexpected alignment faults on Integrity are expensive and affect the entire system performance, so eliminate them if you can. Use the appropriate compiler switches.**
- **MONITOR ALIGN will show if you have issues**
- **Home in on the problem areas using the SDA extension:**
- **Alignment fault trace (with SDA)...**
 - SDA> FLT START TRACE**
 - SDA> FLT SHOW TRACE /SUMMARY**
 - flt_summary.txt**
- **Alignment fault trace (with SDA)...**
 - SDA> FLT START TRACE [/CALL]**
 - SDA> FLT SHOW TRACE**
 - flt_trace.txt**

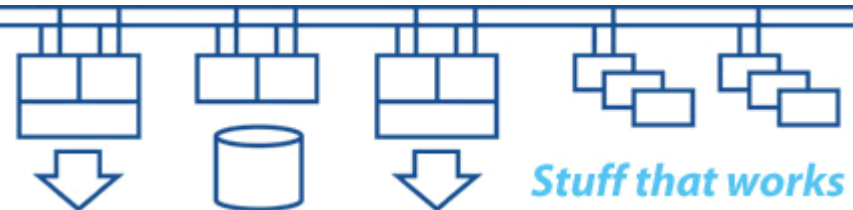


- **Exception handling (LIB\$SIGNAL etc.) is a more expensive mechanism than it was on Alpha**
- **Consider alternatives if your code makes extensive use of exception handling as part of it's normal flow of control**
- **Increase stack space when using threads**

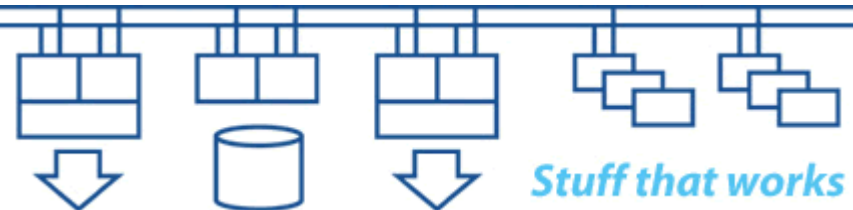


- **Integrity default floating point format is IEEE**
- **VAX floating point is handled by converting to IEEE format, doing the processing, then converting the result back from IEEE format to VAX format, hence a small performance penalty**
- **Alphas do both VAX and IEEE, but the default is VAX**
- **Results with IEEE are ‘slightly different’, not by much**
- **Consider testing for a small difference, not equal**

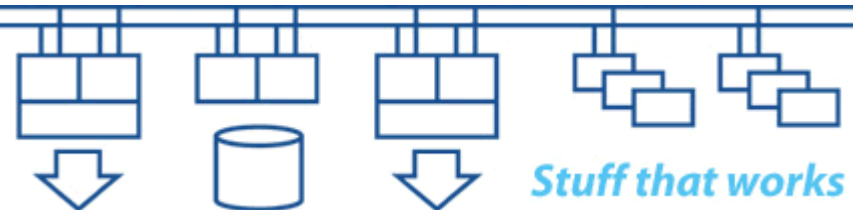
- **Beware LIB\$WAIT(<real>)!**
- **Beware reading in data files / writing out data files and moving them between different systems**
- **What standard do you use for moving data?**



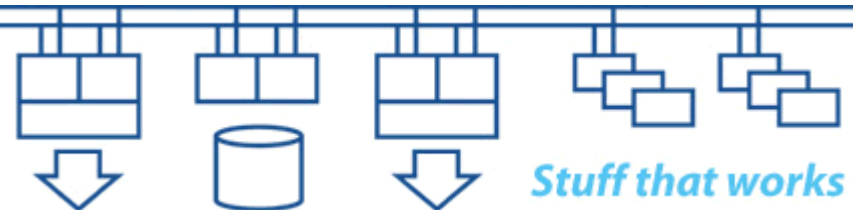
- **Check for code making assumptions about hardware:**
 - “if not Alpha” is definitely a problem!
 - “if not VAX” should be less of a problem
 - “if VAX” / “if Alpha” / “if IA64” is safe
- **Check for code making assumptions about page sizes:**
 - Memory page sizes are variable on Integrity (and Alpha)
 - VAX is 512 bytes (same as disc block size)
 - Alpha is 8192 bytes
 - Integrity is currently 8192 bytes



- **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**

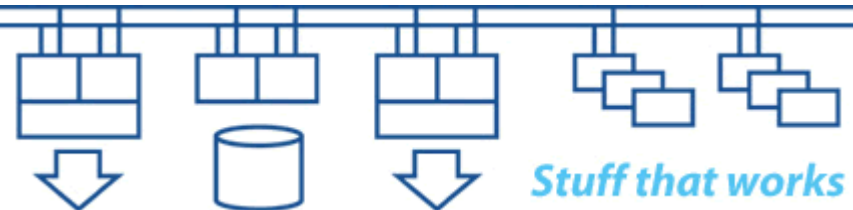


- **Make a thorough audit and analysis of what you're doing at the moment**
- **Consider taking advantage of new features and new ways to do things**
- **There may be no direct equivalents**
- **Direct 'bug for bug' port or re-implement application?**
- **Are you going to have to continue to support non-migrated systems as well?**
- **System infrastructure implications – storage subsystems (eg: SANs) and network connectivity (eg: multiple NICs)**

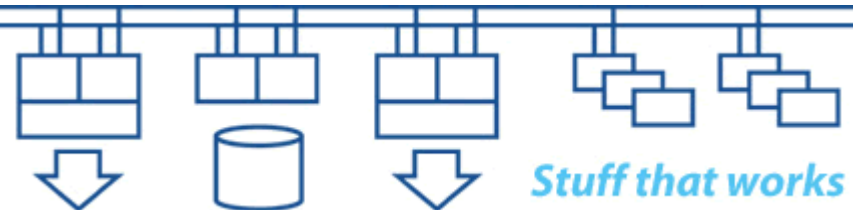


- **Emulators can let you run existing binary code on an emulated hardware platform hosted on a replacement system platform (typically a Windows PC)**
- **Emulators are entirely dependent on the underlying system platform for their overall behaviour**
- **Emulators require IO support to be created specifically**
- **Emulators can give you long-term complications – carefully consider the long-term cost of ownership and the implications for supporting critical applications over time**

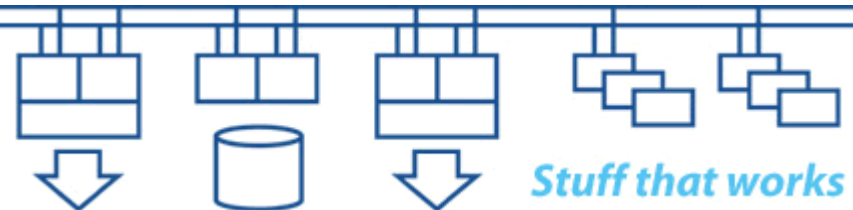
- **Binary translators will convert executable images “on the fly” into an executable image for the target processor, even VAX to Alpha to Integrity**



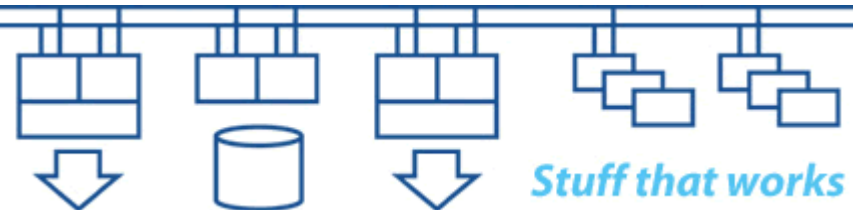
- **Hardware constraints, eg: real-time IO devices**
- **Software constraints, eg: tied to VAX hardware**
- **Operating system version constraints**
- **Layered product version constraints**
- **Network constraints, eg: X.25, PathWorks**
- **Performance constraints, eg: interrupt latency**
- **It may be surprisingly easy – recompile, relink, test thoroughly and then deploy**



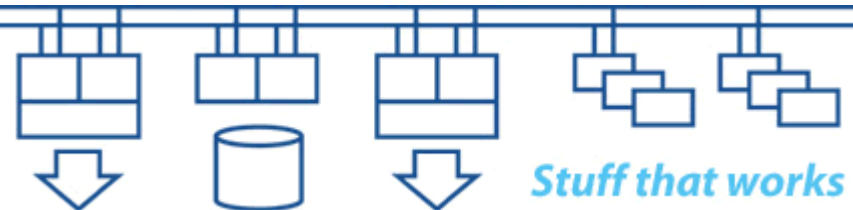
- **Tidy up application code (eg: synchronisation & serialisation, hard-coded values, run-time library calls, system service calls, implicit assumptions, etc. ...)**
- **Think about data alignment**
- **Think about floating point data types**
- **DCL command files and systems management changes**
- **Convert from VMSINSTAL to PCSI?**
- **Either:**
 - **Move via Alpha as an intermediate step**
- **Or:**
 - **Move directly to Integrity**



- **Hardware constraints, eg: IO devices**
- **Software constraints, eg: tied to Alpha hardware**
- **Operating system version constraints**
- **Layered product version constraints**
- **Network constraints, eg: PathWorks**
- **Performance constraints, eg: big multiprocessor systems**
- **Functionality constraints, eg: Galaxy soft partitioning**
- **It may be surprisingly easy – recompile, relink, test thoroughly and then deploy**

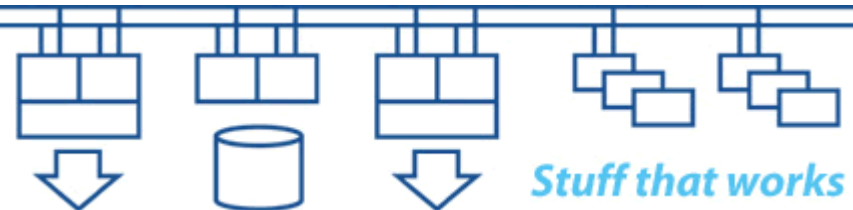


- **Tidy up application code (eg: synchronisation & serialisation, hard-coded values, run-time library calls, system service calls, implicit assumptions, etc. ...)**
- **Think about data alignment**
- **Think about floating point data types**
- **DCL command files and systems management changes**
- **Convert from VMSINSTAL to PCSI?**
- **Move to Alpha current (eg: OpenVMS Alpha V8.3)**
- **Move from Alpha current to Integrity (eg: OpenVMS Alpha V8.3 to OpenVMS Integrity V8.3)**



A lot of information is on the web:

- **Porting workshops**
- **HP DSPP (<http://www.hp.com/dspp>)**
- **OpenVMS Technical Journals**
- **OpenVMS documentation**
- **OpenVMS V8.3-1H1 Integrity Servers**
- **Transition tools:**
<http://h71000.www7.hp.com/openvms/ovmsproducts.html>
- **White papers:**
<http://h71000.www7.hp.com/openvms/whitepapers/index.html>
- **FAQ:**
<http://h71000.www7.hp.com/openvms/integrity/faqs.html>



Thank you for your participation

Q & A

