Porting OpenVMS Applications to the Itanium Processor Family

Gaitan D'Antoni
OpenVMS Technical Architect

System and hardware support

Integrity Servers

- OpenVMS V8.2 will support entry-level to mid-range HP Integrity servers
  - Entry-level Integrity: rx1600—all CPU speed variants
  - Low-end Integrity server: rx2600—all CPU speed variants
  - Mid-range Integrity server: rx4640—all CPU speed variants

Alpha Servers

- Support for new EV7 chip speedup for AlphaServer ES47, ES80, and GS1280

Porting goals

- Provide an operating system environment, development tools, and documentation to make porting as easy as possible
  - Full port of the operating system, runtime libraries, development tools, and most layered products
  - Recompile, relink, requalify
- Use our experiences porting the operating system to make it easier for others to port their applications
  - Internal layered product groups, partners, and customers

Major changes to the base OS

No Alpha Console

- Booting
- Device discovery
- Interrupts
- TLB miss handler

No Alpha PALcode

- VAX queue instructions
- VAX registers
- IPL and mode change

Different primitives in CPU

- Register conventions
- Exception handling
- Atomic instructions
- Process context

Plus, we decided to change

- Calling standard
- Object language
- Image format

General development rules

- Object file and image file sizes are larger on OpenVMS I64 than on OpenVMS Alpha
- Alignment faults are more costly on I64 than on Alpha
- Applications should be built on OpenVMS Alpha using the latest versions of the compilers before they are ported to OpenVMS I64
- Consult each compiler's Release Notes for problems and restrictions with the current version of the compilers

Alpha compilers

- HP recommends that you build your applications on OpenVMS Alpha using the latest versions of the compilers prior to starting your port to OpenVMS I64
- Latest/next releases on Alpha Platform
  - C V6.5, C++ V6.5
  - Fortran V7.5 (F90)
  - Basic V1.5
  - COBOL V2.8
  - Java 1.4.2-3
  - Pascal V5.9

OpenVMS on Integrity Server Compilers

C

- Itanium architecture implementation of OpenVMS HP C V6.5 compiler

C++

- Based on the same front end compiler technology as HP C++
- This is not a port of HP C++ V6.5 but it will be able to compile most of the same source code as HP C++ V6.5

COBOL, BASIC, PASCAL, BLISS

- Itanium architecture implementations of the current OpenVMS compilers

OpenVMS on Integrity Servers Compiler Plans

FORTRAN

- Itanium architecture implementation of the current OpenVMS Fortran 90 compiler

Java

- Itanium architecture implementation of J2SE V1.4.2

IMACRO

- Compiles ported VAX Macro-32 code for Itanium architecture
- Itanium architecture equivalent of AMACRO

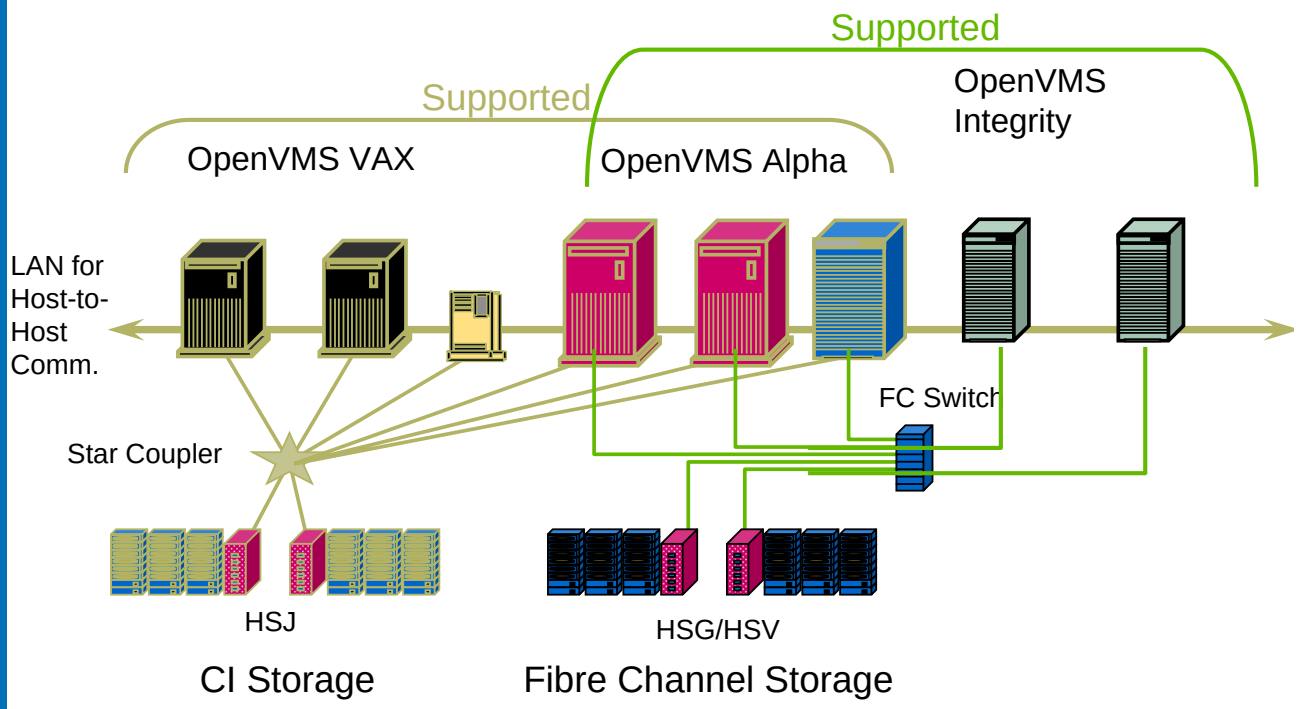
ADA

- We will be providing an Ada-95 compiler
- We did not port the existing Ada-83 compiler

Binary translator

- Can translate Alpha OpenVMS binary images and libraries linked under all OpenVMS versions from 6.2 to current version
- Can translate a VESTed image that was translated by DECmigrate from a VAX binary image
- Will not translate applications written BASIC, Pascal, PL/1, or Ada
- Restrictions
  - Alpha binary code
  - Only user-mode apps
  - No privileged instruction
  - No self-modifying code
  - No system memory space reference
  - No user-written system services

Continuing evolution of OpenVMS Clusters



NOTE: Support for VAX and Integrity mixed environment is not supported in a production environment.

Infrastructure changes in OpenVMS V8.2

- We made changes to some system level data structures in OpenVMS V8.2 (Alpha and I64)
- Benefits
  - Laying the foundation for scalability and performance improvements in future releases of OpenVMS
- Impact to applications
  - Non-privileged applications are not affected
  - Applications that access the modified data structures in non-standard ways might need to be modified
    - Examples: Hard-coded data structure sizes and assumptions about the relative locations of fields within a data structure

Infrastructure changes in OpenVMS V8.2

Impact to applications (continued)

- Some privileged applications (such as device drivers) will need to be recompiled and relinked
  - Privileged applications in this case are images linked against the system using the /SYSEXE qualifier and reference the changed data structures or related structures and routines
  - Attempting to execute or load such an image that has not been rebuilt will result in an error during image activation of SYSVERDIF—“System Version Mismatch”

Major porting considerations

New calling standard

- Publicly available today at http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/resources.html
- Intel calling standard with OpenVMS modifications
  - No FP
  - Multiple stacks
  - Only 4 preserved registers across calls
  - Register numbers you are familiar with will change
- All OpenVMS provided tools “know” about these changes
- Most user applications are not affected
- Your code that “knows” about the Alpha standard will almost certainly need to change

Major porting considerations

Object file format

- ELF/DWARF industry standards plus our extensions
  - ELF—Executable and Linkable Format, Itanium architecture object code, images, etc.
  - DWARF—Debugging and traceback information (embedded in ELF)
- All OpenVMS provided tools “know” about these changes
- Most user applications are not affected
- User written code that “knows” the object file format may have to change
- Specifications are available on the Web

Major porting considerations

Floating point data types

- Itanium architecture supports IEEE float only
- All compilers that currently support F, D, G, S, T, and X (S and T are native IEEE formats) will continue to do so on Itanium architecture
- IEEE is the default
- We have updated the appropriate Runtime Libraries to add IEEE interfaces where needed
- White paper with technical details about the differences between VAX Float and IEEE Float is available at http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/resources.html

Major porting considerations

- Source Code that May Need to Change
- Architecture Specific code
  - All Alpha assembler code must be rewritten
- Conditionalized code
  - Build command files
    - `$ if .not. Alpha  ! Assumes VAX`
  - Application source code
    - `#ifndef (alpha)  // Assumes VAX`
    - C asm code
- SYS$GOTO_UNWIND system service must be replaced by SYS$GOTO_UNWIND_64
  - OpenVMS I64 requires a 64-bit invocation context
  - SYS$GOTO_UNWIND_64 can be used on Alpha to maintain common source code

Major porting considerations

- Source code that may need to change
- SYS$LKWSET and SYS$LKWSET_64 system services runtime behavior has been modified
  - The entire image, not the specified range of pages, is locked
  - Consider using LIB$LOCK_IMAGE and LIB$UNLOCK_IMAGE for simplicity
- SS$_HPARITH is replaced by SS$_FLTINV and SS$_FLTDIV
  - To maintain common code use:

```
if ((sigargs[1] == SS$_HPARITH) || (sigargs[1]
== SS$_FLTINV) || (sigargs[1] == SS$_FLTDIV))
```

- Mechanism Array data structure has been changed
  - Standard calling interfaces have not changed
- The Porting Guide contains all of the details

Major porting considerations

Improperly declared functions and data

- C function declarations that points to objects that are not functions, may work on Alpha but these declarations will not work on IA64
  - Also seen with the Bliss compiler
- This problem may manifest itself in many ways
  - For whatever reason, the most common symptom is routine CLI$DCL_PARSE failing with CLI-E-INVTAB
  - In case of a failure the command table is usually defined as — int master_cmd();

    Change to extern master_cmd;

    and change the way the parameter is passed to cli$dcl_parse from master_cmd to &master_cmd

Linker

- New /BASE_ADDRESS qualifier
  - Replaces the CLUSTER=,[base-address] option on Alpha
- New SHORT_DATA option
  - Allows you to combine read-only and read-write short data sections into a single segment
  - Eliminates unused space (up to 65,535 bytes) between two (read-only and read-write) segments
- New alignments for PSECT_ATTRIBUTE option
  - Integer values of 5, 6, 7, and 8 representing the byte alignment indicated as a power of 2. e.g. 2**6 = 64-bit alignment
- It is highly recommended that data reduced object libraries are not referenced in Link command on Itanium architecture
  - It is necessary for the linker to expand the entire module in memory which can result in BYTLM or PGFLQUOTA issues

Alignment faults

Once the port of the application has been completed you should look at alignment faults

- Alignment faults are expensive on Alpha but are 100 times more expensive on Integrity Servers
- The DEBUG SET MODULE/ALL command used to take 90 seconds; after fixing some alignment faults, it now takes 2 seconds
- DCL procedures take approximately 10% less time to execute after fixing alignment faults in DCL
- You can detect alignment faults using the FLT extension in SDA or using SET BREAK/UNALIGN option in the debugger
- Some alignment faults are easy to fix, some are very hard and some are close to impossible

Real-life experiences

- HP/Intel Developer Forum
  - 5 events, 75 participants, 51 solutions ported
- FORTRAN—Switch from F77 to F90
- Update to latest versions of compilers
- MACRO-32—Pay attention to the porting guide
- Old development environments
  - Have not kept up with changes to compilers, RTLs, pthreads, etc.

For further information about OpenVMS on Integrity Servers

- General OpenVMS on Integrity Servers
  http://h71000.www7.hp.com/openvms/integrity/index.html
- Layered product rollout schedules
  http://h71000.www7.hp.com/openvms/os/swroll/index.html
- Layered products plans (products that either will not be ported or are under review)
  http://h71000.www7.hp.com/openvms/integrity/openvms_plans.html
- OpenVMS Partner plans
  http://h71000.www7.hp.com/openvms/integrity/partners.html
- OpenVMS on Integrity Servers Total Cost of Ownership white paper
  http://h71000.www7.hp.com/openvms/whitepapers/alinean_tco.pdf


invent