OpenVMS I64 Technical Update

Session 3840

Gaitan D'Antoni
OpenVMS Technical Architect
Hewlett-Packard
gaitan.dantoni@hp.com

Agenda

- OpenVMS Roadmap / directions
- Porting applications to OpenVMS I64
- OpenVMS Calling Standard

HP OpenVMS – Itanium® port status exceeding expectations

OpenVMS v8.2 Production Release
(AlphaServer & Integrity) 2004H2

OpenVMS v8.1 Evaluation Release
December 2003

1st & 2nd ISV Applications Ported
August 2003

OpenVMS v8.0 in DSPP labs
August 2003

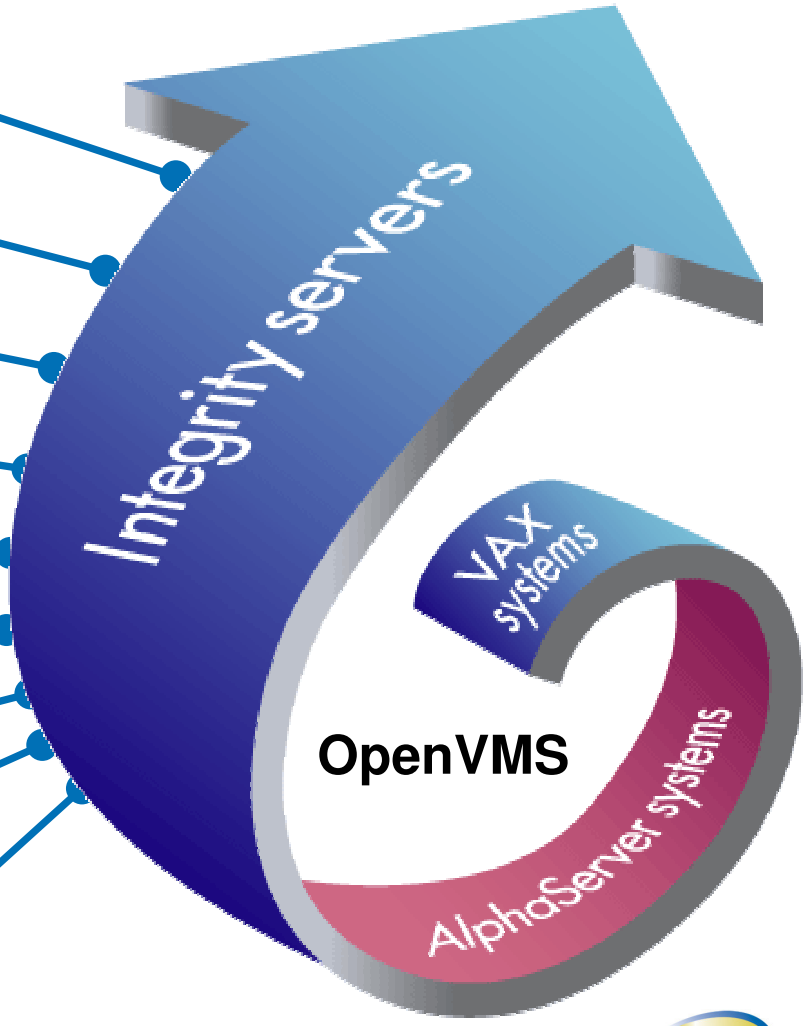
4 Processor System Boot
July 2, 2003

OpenVMS v8.0 Evaluation Release
June 30, 2003

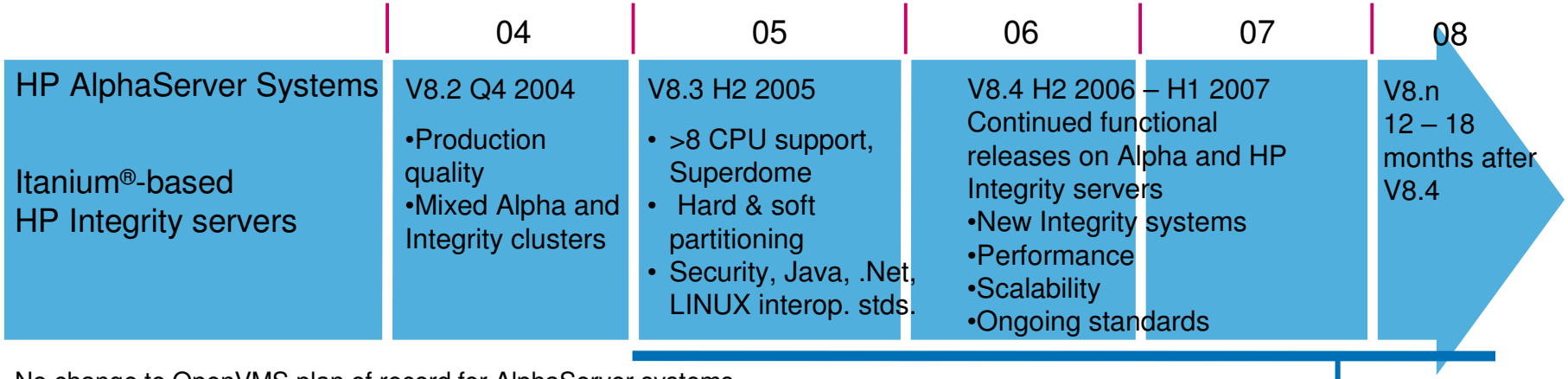
1st Application Port & Mixed Cluster
May 15, 2003

Boot to rx 2600 server
March 17, 2003

1st boot to Itanium® system
January 31, 2003



HP OpenVMS roadmap



No change to OpenVMS plan of record for AlphaServer systems
- Continued enhancements, full support for EV7z systems
- Sales of AlphaServer systems at least until 2006, with support at least until 2011

Integrity Server support
- Production release for HP Integrity servers in H2/2004, V8.1 evaluation release available now
- OpenVMS V8.2 production release will support the rx1600, rx2600, and rx4640 Integrity servers

Focus areas:
- Adaptive Enterprise: Workload Manager, Pay Per Use, Utility Data Center
- OpenView support for OpenVMS: Network Node Manager, Data Protector,Storage Area Manager, OVO Agent available today; looking to add database SPI's, Performance Agent, upgrade to OVO v8
- Continued enhancements in performance and scaling, disaster tolerance, security and standards
- Support for current and next generation storage architectures
- Continued J2EE and .Net support
- Mixed Alpha and Itanium cluster support with shared fibre channel storage today, full 96 node support in 2005.

Services
- Full set of tools/services to support ISVs and customer transition to Itanium-based HP Integrity servers
- Investment protection through Alpha RetainTrust Program

ISV support
- Over 600 applications committed for porting to OpenVMS I64
- FastTrack program to assist with ISV support

Porting Applications to OpenVMS I64

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

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 1.4.2-2
  - Pascal V5.8
    - V5.9 Planned for mid-2004

OpenVMS on Integrity Servers Compiler Plans

- 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

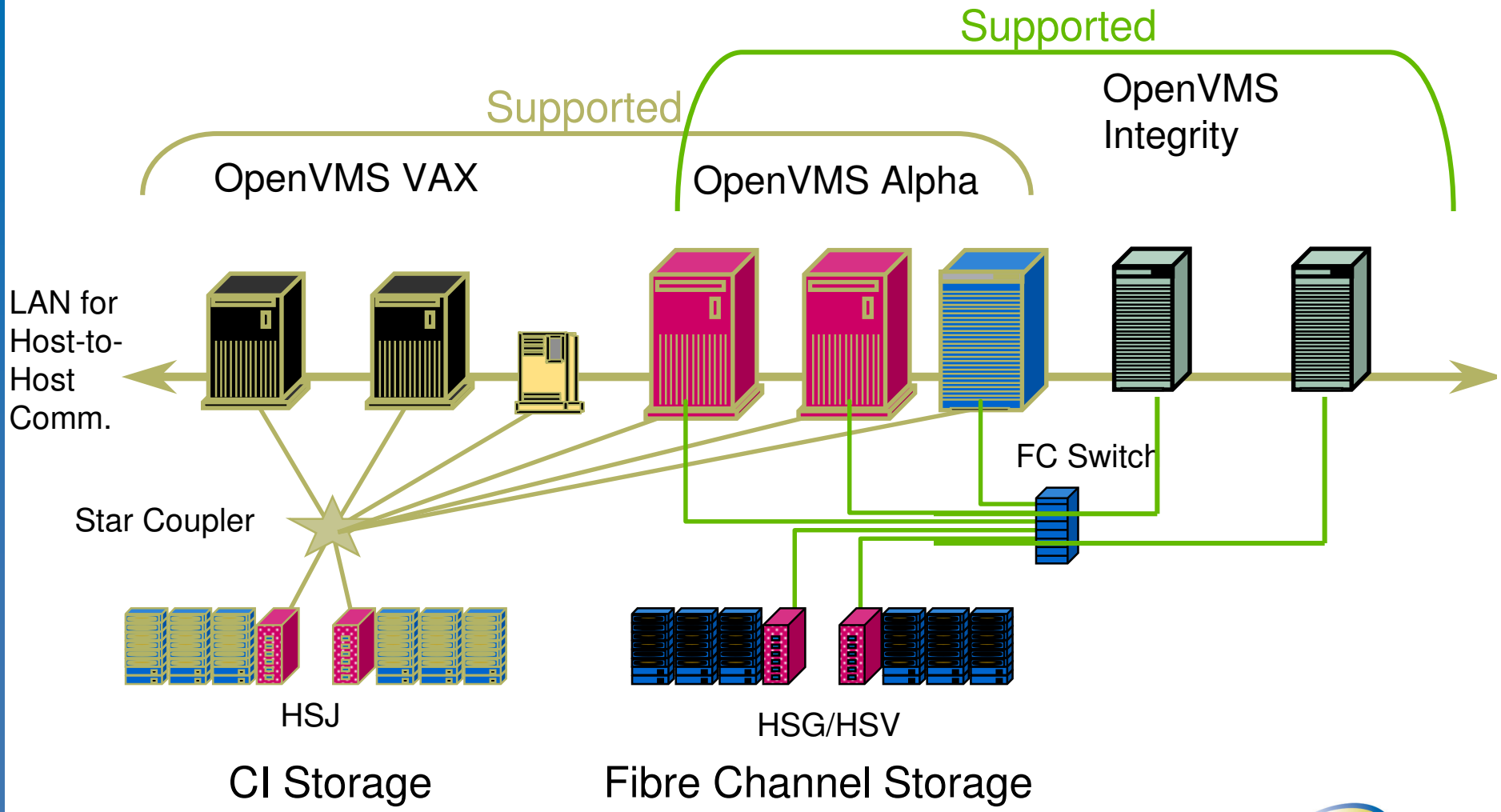
OpenVMS on Integrity Servers Compiler Plans

- 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 provide an Ada-95 compiler
  - We will not port the existing Ada-83 compiler

Binary Translator

- Will translate Alpha OpenVMS binary images and libraries linked under all OpenVMS versions from 6.2 to current version
- Will translate a VESTed image that was translated by DECmigrate from a VAX binary image
- Will translate images written in C, C++, FORTRAN, or COBOL
  - 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 sys. Memory space reference
  - No user-written system services

Continuing evolution of OpenVMS Clusters



NOTE: Support for VAX and Integrity mixed environment is not currently planned.

Infrastructure changes

- Some system level data structures have been changed in OpenVMS V8.2 (Alpha and I64)
- Benefits
  - We're laying the foundation for scalability and performance improvements in future releases of OpenVMS
- The OpenVMS Philosophy
  - Try to never break non-privileged images
    - An image linked on early versions of OpenVMS Alpha should run on current versions of OpenVMS Alpha
  - Only make changes that impact privileged images with the release of a "major" version
    - V6 to V7 introduced 64-Bit Support

Infrastructure changes

- Impact to applications
  - Non-privileged applications are not affected
  - 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”.
  - Applications that access the modified data structures in non-standard ways may need to be modified
    - Examples: hard-coded data structure sizes and assumptions about the relative locations of fields within a data structure

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 frame pointer (FP)
    - Multiple stacks
    - only 4 preserved registers across calls
    - Register numbers you're familiar with will change
  - All OpenVMS provided tools will "know" about these changes
  - 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 will “know” about these changes
  - User written code that “knows” the object file format may have to change
  - We will be publishing these specifications in the near future

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 will be the default
  - HP will update 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$_HPARITH (high performance arithmetic trap) is replaced by SS$_FLTINV (floating point invalid) and SS$_FLTDIV (floating divide by zero)
  - 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
- We will be providing a new Porting Guide with details

Major Porting Considerations

Improperly declared functions and data

- Function declarations that point to objects that are not functions may work on Alpha but these declarations will not work on I64
- This problem may manifest itself in many ways
  - From our experience 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

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're ported to OpenVMS I64
- /FLOAT=IEEE_FLOAT and IEEE_MODE= DENORM_RESULTS are the floating point defaults
- Consult each compiler's Release Notes for problems and restrictions with the Field Test versions of the compilers

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.

Performance

- As we are doing with the GS1280, we are committed to setting accurate performance expectations for OpenVMS on Integrity Servers.
- Throughout 2004 we will expand our testing to provide additional and more accurate performance information.
- Our tests will include low level component tests as well as high level workload tests.
- We will provide results comparing similarly configured systems (number of CPUs, amount of memory, CPU speed, disk speed and capacity)
  - ES45 / rx4640

The HP OpenVMS Itanium® Calling Standard

What's the Problem?

An example:

- On VAX and Alpha
  - R0 = function return value
- On Itanium® architecture
  - R0 = zero

What's a Calling Standard?

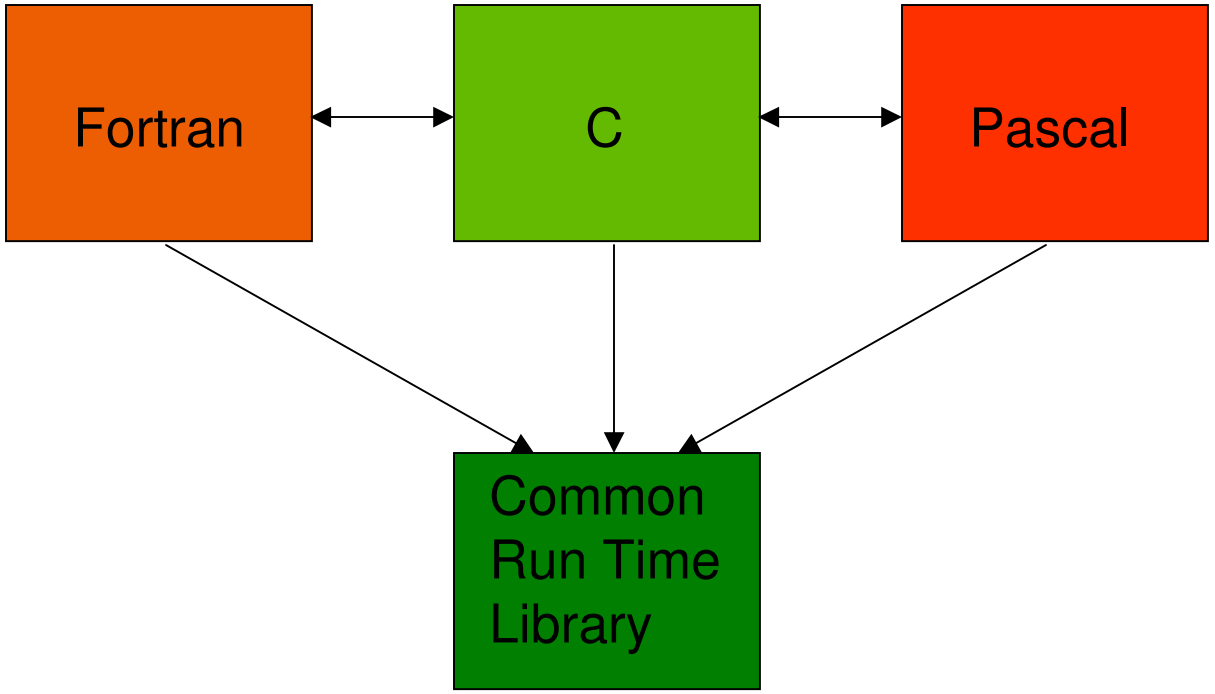
- In C

  Z = func (X, Y);

- Binary representation of subroutine linkage:
  - Where are my arguments?
  - Where's my execution environment?
  - How do I get back to my caller?
  - How do exceptions get handled?

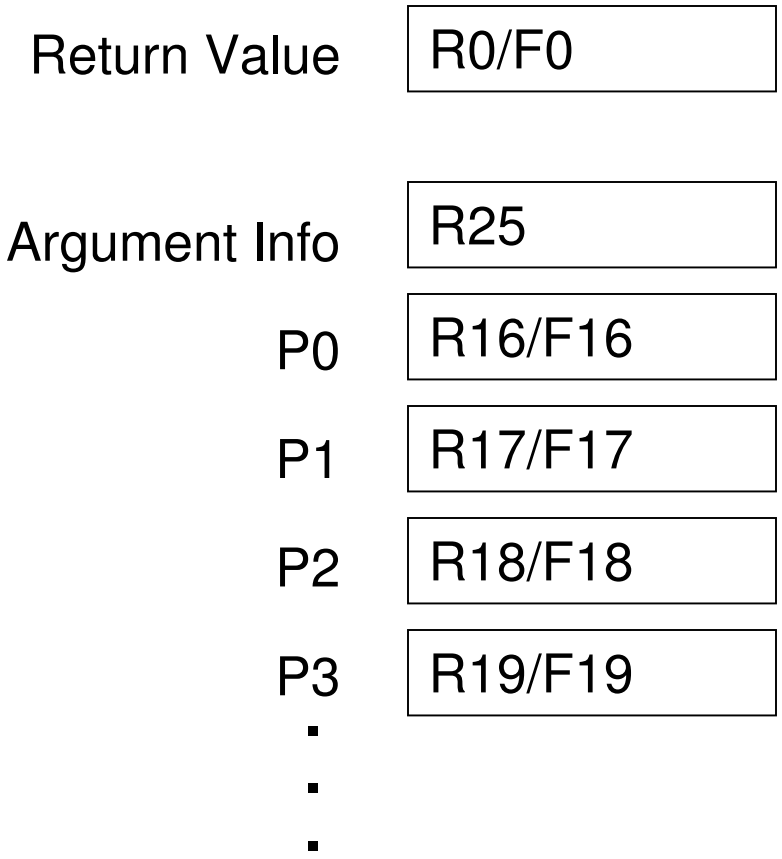
Why Have a Calling Standard?

- Multi-language interoperable programming environment



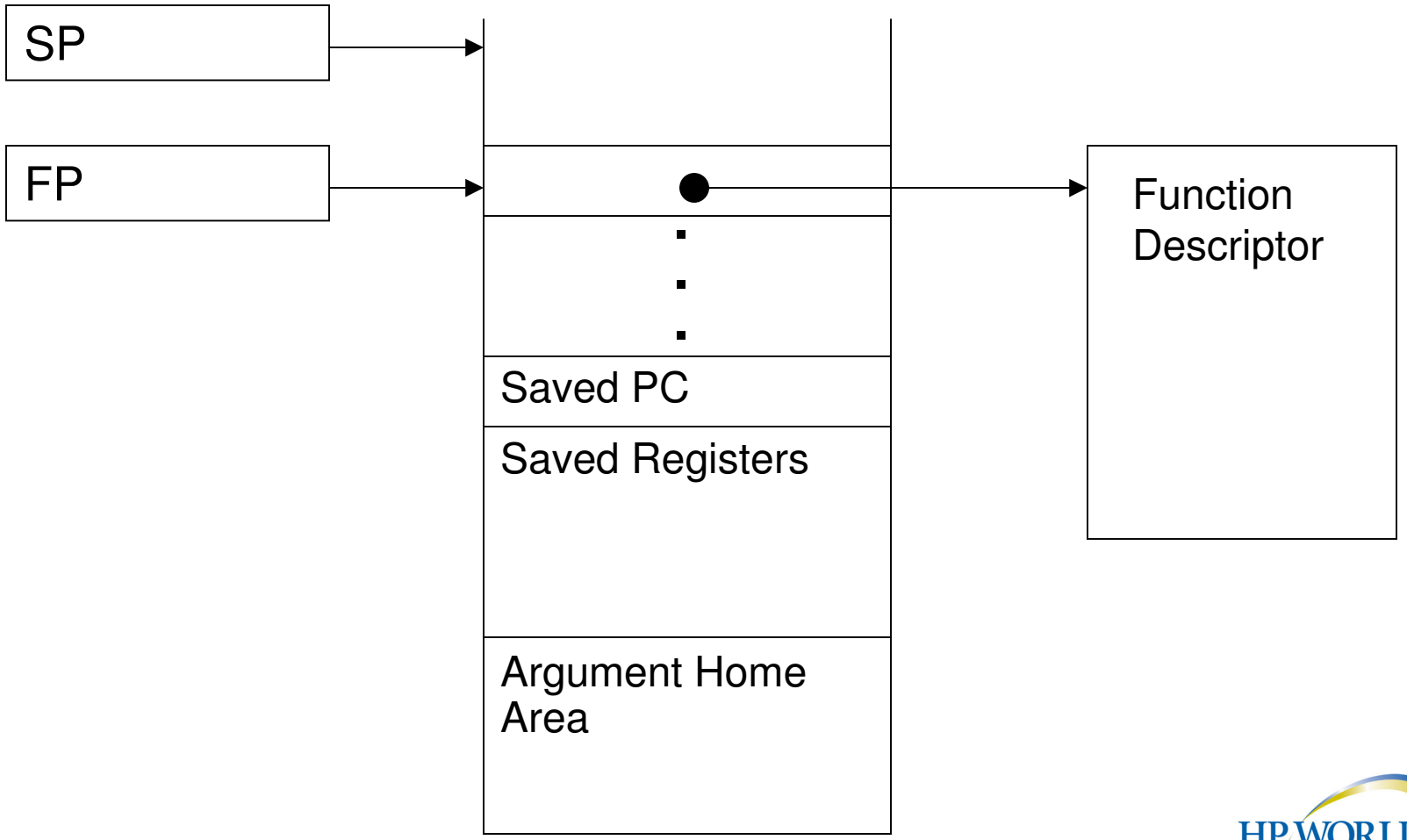
On Alpha

- Alpha – Argument List & Return Value



On Alpha

- Alpha – Call Frame and Function Descriptor



The Challenge

- Intel® defined Itanium® Calling Conventions
- Comparable to Alpha calling standard, but lacking
  - Argument count / information
  - VAX/Alpha floating point datatypes
  - Support for translated images
  - Definition of invocation context handle

Our Approach

- Adopt the mainstream Itanium® standards
  - Intel® calling standard
  - ELF object / image file format
  - DWARF debug information format
- Extend / specialize as needed to support existing VMS features

Benefits

- Intel® has accepted our extensions
- It will be easier to use industry standard tools
  - Object file post-processors / analyzers
  - Linux linker

The Future

- OpenVMS Itanium® Calling Standard
- Based on industry standard Itanium® Calling Conventions
- Extended for OpenVMS
  - Argument count / information register
  - VAX/Alpha floating point formats
  - Translated image support
  - Additional definitions

Argument List & Return Value

CALLEE's view of registers

Return Value	R8-9/F8-9
Argument Info	R25
P0	R32/F8
P1	R33/F9
P2	R34/F10
P3	R35/F11

Compiler Support for Migration

- Register Mapping
  - VAX Macro compiler maps Alpha register numbers to their Itanium® architecture equivalents
  - Bliss compiler supports user switchable mapping
  - C compiler will automatically map registers used in the #pragma linkage directives to their corresponding I64 registers

Impact on Applications

- Mostly none
- Except for
  - Explicit register use
  - Knowledge of stack and exception frame format

For further Information about OpenVMS on Integrity Servers

- OpenVMS on the Itanium® Architecture Web Sites
  - General OpenVMS on Integrity Servers **http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/resources.html**
  - Layered products schedules **http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/openvms_move.html**
  - Layered products plans (products that either will not be ported or are under review) **http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/openvms_plans.html**
  - OpenVMS Partner plans **http://www.hp.com/products1/evolution/alpha_retaintrust/openvms/partners.html**

Q & A

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional