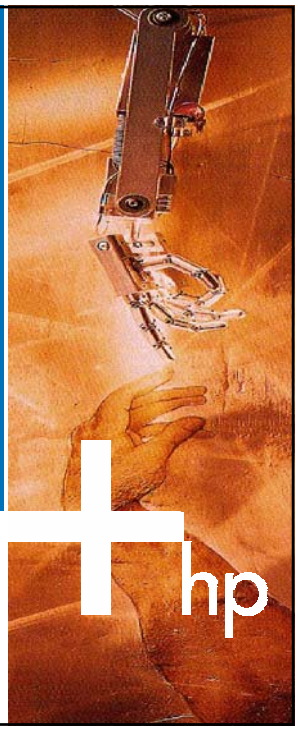
OpenVMS Moving Custom Code



Thomas Siebold, Senior Software Consultant
Business Critical Systems
Transition Engineering and Consulting Group

thomas.siebold@hp.com

Topics

- Porting Overview
- Compiler
- Binary Translation
- Software Development
- Application Considerations
- Middelware
- ISVs
- Next steps...

2

Porting Overview

Porting to OpenVMS I64

- Porting applications to I64 is easy
- Usually all that is required is to recompile/ relink and requalify the application.
  - Privileged code may require more effort
  - Porting 100,000 lines of C code did not require even one change

HOWEVER

4

Porting to OpenVMS I64

- MANY!!! Things have changed in the O/S
  - Different primitives
  - Different default floating point standard
  - New compilers
  - New image format
  - New calling standard
  - No console/PAL code

Most changes are transparent but these changes might affect your application

5

Compiler

Compiler Version Mapping Alpha vs. Itanium(r)

Compiler	*Alpha*	*Itanium*
Basic	V1.5	V1.6
Bliss	V1.10-030	V1.1 (-04x)
Cobol	V8.2-1286	V8.2 (-13xx)
Fortran 77	--	na (Alpha only)
Fortran 90	V7.5	V8.0
C	V6.5	V7.1
C++	V6.5	V7.1
Java	1.4.2	1.4.2 (-2)
Macro-32	V4.1-18	V1.0
Macro-64	V1.2	na (Alpha only)
Pascal	V5.8A	V5.9

7

Use latest compilers

- Porting to Itanium requires to use the latest compiler versions
  - Some issues may show up due to changes and even bugfixes to the compilers.
- Recommendation:
  - build application on Alpha using the latest version of the compilers to uncover any hidden bugs/changes
- Result:
  - Easier move to new platform

8

Itanium® Compiler Plans (1 of 3)

- C
  - CPQ C
    - Itanium® architecture implementations of OpenVMS CPQ C V6.5 compiler
    - Use for recompile/relink/requalify
    - GEM backend code generator
  - C Dialect Support in C++ Compiler
    - Will include some features from CPQ C but may require source code changes
    - Compiler for moving forward
    - Intel® backend code generator
    - Will be made this available with a future release of OpenVMS

9

Itanium® Compiler Plans (2 of 3)

- C++
  - Based on the same front end compiler technology as Compaq C++
  - Use for recompile/relink/requalify
  - Intel® backend code generator
- COBOL, BASIC, PASCAL, BLISS
  - Itanium® architecture implementations of the current OpenVMS compilers
  - GEM backend code generator
- Java
  - Itanium® architecture implementation of J2SE V1.4.2

10

Itanium® Compiler Plans (3 of 3)

- FORTRAN
  - Itanium® architecture implementation of the current OpenVMS Fortran 90 compiler
  - GEM backend code generator
  - Our plan is to replace GEM with the Intel® backend code generator in a future release in order to take advantage of enhancements in processor chip technology
- IMACRO
  - Compiles ported VAX Macro-32 code for Itanium® architecture
  - Itanium® architecture equivalent of AMACRO
- ADA
  - Ada-95 compiler from Ada Core Technology (GNAT)
  - HP Ada (Ada-83) compiler will not be ported

11

Binary Translation

Options for Applications without Source Code or Development Expertise

- Software Emulation
  - Emulator (Charon-VAX)
- Binary Translation
  - VEST Translator and TIE
  - AEST Translator and TIE
- Rewrite Applications / Purchase Replacement Applications (not covered here)

13

Software Emulation

- Images run on new platform
- Code interpretation (= emulation) at run time
- Positive
  - Nearly effortless
  - fast
- Negative
  - Some loss of performance
  - Maintenance & Extension difficult

14

Binary Translation

- Images are translated
- „Compilation“ of runable image
- No emulation
- VAX/VMS -> Alpha/VMS -> Itanium/VMS
- Fast
- Good performance
- Programming language independant
- Shareable images do work

15

Binary Translation

- Input: VAX/VMS Image
- Output: Alpha/VMS Image
- OR:
- Input: Alpha/VMS Image or translated VAX/VMS Image
- Output: Itanium/VMS Image

16

DECmigrate (VAX to Alpha)

- Translate images for which source code is not available
  - VAX Environment Software Translator (VEST) translates VAX binary image file into a native Alpha image
  - Translated images run under the Translated Image Environment (TIE) on Alpha
  - Alpha images contain native Alpha instructions
- Updated release by June '02
- Released and Supported by HP

17

VEST – Current Restrictions

- Will translate valid VAX/VMS image
  - Image(s) must be linked on OpenVMS V6.2 (removed in future)
- Restrictions:
  - Currently only up to V6.2 (removed in future)
  - Only user- mode apps
  - Non privileged instructions
  - No self-modifying code
  - No sys. Memory space reference
  - No user-written system services
  - No drivers

18

OpenVMS AEST Binary Translator

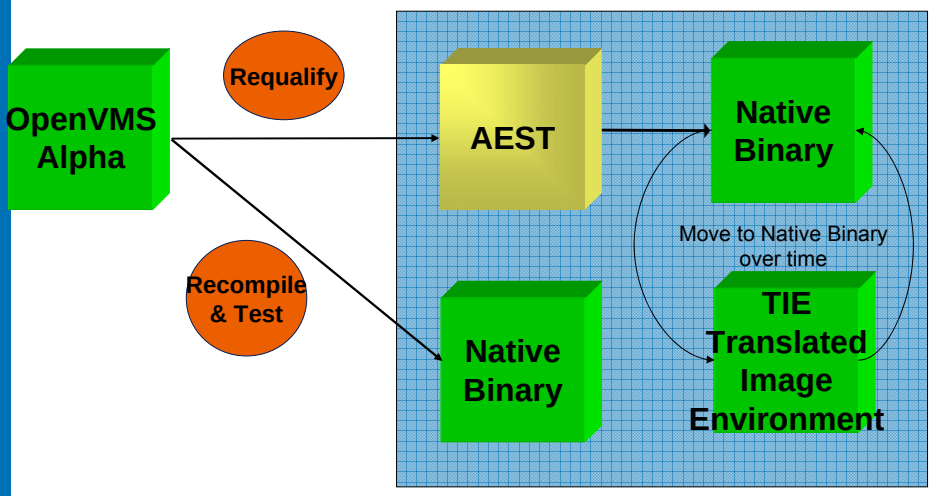
- Will translate Alpha OpenVMS binary images and libraries linked under all OpenVMS versions from 6.2 to current version
- Will translate a VESTed image that was translated by DECmigrate from a VAX binary image
- Will translate images written in C, C++, FORTRAN, or COBOL
  - Will not translate applications 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

19

TMS1

Alpha Environment Software Translation

OpenVMS I64



20

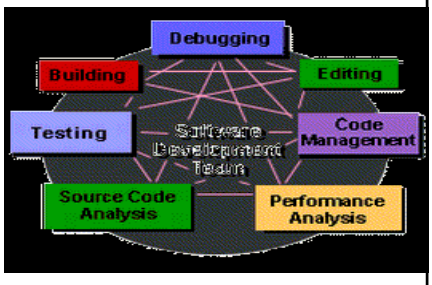
Folie 20

TMS1 This needs to be verified. With each run of an image the image becomes 'more and more' translated, e.g. native code generated.
Thomas Siebold; 2.2.2005

Software Development

Development Tools

- OpenVMS Debugger
- DECset Toolset:
  - Code Management System
  - Digital Test Manager
  - Language Sensitive Editor
  - Source Code Analyzer
  - Module Management System
  - Program Coverage Analyzer



22

Software Development

NetBeans?

- Sun-Sponsored Open-Source Integrated Development Environment
- 100% Java – runs anywhere there's a JVM
- Feature-rich, drag-n-drop GUI creation, JSPs, Web services
- Extensible
- Supports other languages (C/C++, XML, HTML, Fortran*, Cobol*, Pascal*)
  - Support for CMS
  - CVS client
  - Ant (multiple platform builds from one build definition file)

23

Software Development

"NetBeans"?

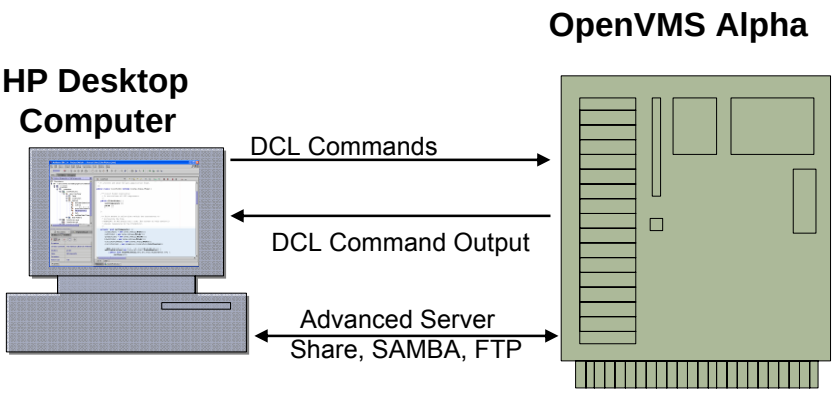
- Used to have Enterprise Toolkit for Visual Studio (V6), now replaced by Netbeans

"Distributed NetBeans" ?

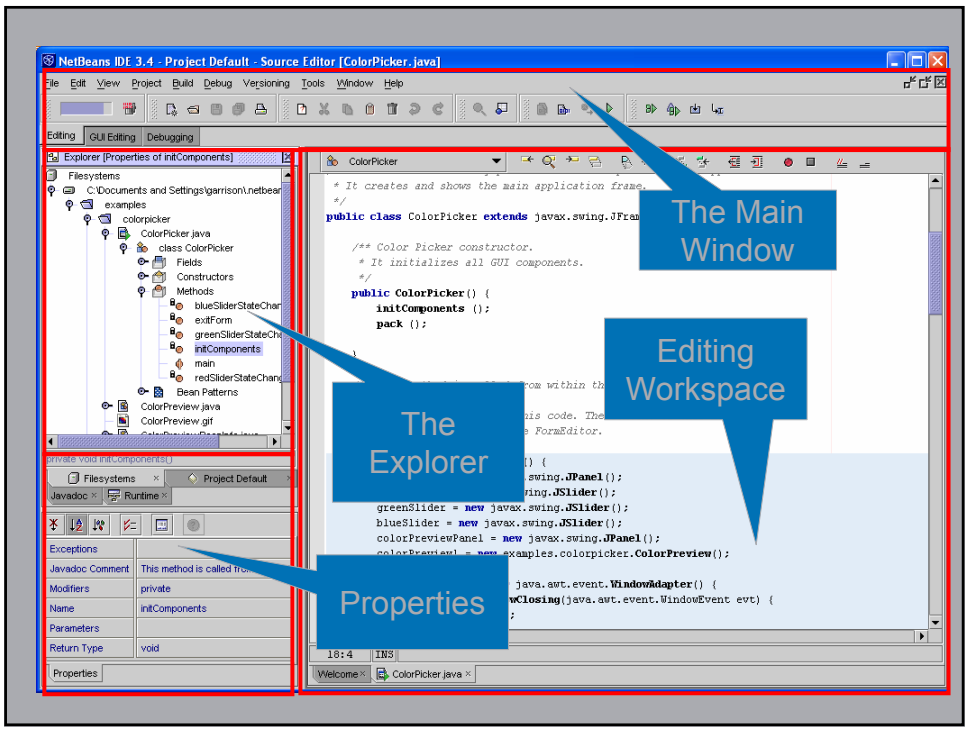
- Allows any desktop (Windows, Linux, HP-UX, etc.) to be used to do remote OpenVMS development
- NetBeans runs on the desktop
- Provides remote compilation (Java, C/C++,…), error navigation, remote execution, and eventually debugging
- Also provides remote Ant (*"Make without the wrinkles")* operations
- Remote CMS operations

24

Software Development
Distributed NetBeans



25



Application Considerations

Architecture specific build procedures

- ARCH_TYPE
  - Returns 1 on VAX, 2 on Alpha, 3 on Integrity
  - Supported on OpenVMS Version 5.5 or later
- ARCH_NAME
  - Returns text string "VAX" on VAX, "Alpha" on Alpha, “IA64” on Integrity systems
  - Supported on OpenVMS Version 5.5 or later.
- HW_MODEL
  - Returns an integer that identifies a particular hardware model
  - Values >= 1024 identify Alpha systems
  - Values = 4096 identify Integrity systems

28

procedures

```
$! Determine architecture type
$ type_symbol = f$getsyi("arch_type")
$ if type_symbol .eq. 1 then goto ON_VAX
$ if type_symbol .eq. 2 then goto ON_ALPHA
$ if type_symbol .eq. 3 then goto ON_INTEGRITY
$ ON_VAX:
$ !Do VAX-specific processing
$ exit
$ ON_ALPHA:
$ !Do Alpha-specific processing
$ exit
$ ON_INTEGRITY:
$ !Do INTEGRITY-specific processing
$ exit
```

29

Conditionalized code

- This is the first (and easiest) step to take
  - Usually, IA64 should take what use to be the Alpha code path.
    - In some cases, IA64 specific code path should be added

```
#include <stdio.h>
#include <arch_defs>
void main()
{
#ifdef __vax
        printf("This is the VAX codepath");
#endif
#ifdef __alpha
        printf("This is not the VAX codepath");
}
```

30

Conditionalized code – example

```
IPL31> type arch_test.c

#include <stdio.h>
#include <arch_defs>
void main()
{
#ifdef __vax
        printf("This will be printed on VAX\n");
#endif
#ifdef ALPHA
        printf("This will be printed on Alpha\n");
#endif
#ifdef __ia64
        printf("This will be printed on IA64\n");
#endif
#ifndef __vax
        printf("This program is not running on VAX");
#endif
}
```

31

Conditionalized code

Executed on IA64 system

```
IPL31> write sys$output f$getsyi("arch_name")
IA64
IPL31> r arch_test
This will be printed on IA64
This program is not running on VAX
IPL31>
```

Executed on Alpha system

```
MIKAXP> write sys$output f$getsyi("arch_name")
Alpha
MIKAXP> r arch_test
This will be printed on Alpha
This program is not running on VAX
```

32

Conditionalized code
Sample Fortran 90 program

COM file

```
$!
$! Note: F90 not available on VAX
$!
$ if f$getsyi("ARCH_NAME") .EQS.
  "IA64"
$ then
$      f90/define=IA64 archdef_for
$ else
$ if f$getsyi("ARCH_NAME") .EQS.
  "Alpha"
$ then
$       f90/define=ALPHA
  archdef_for
$ endif
$ endif
$ endif
$ link archdef_for
```

Language file

```
 program archdef
        implicit none
!DEC$ IF DEFINED (VAX)
        type *, 'Running on VAX hardware'
!DEC$ ELSEIF DEFINED (ALPHA)
        type *,'Running on Alpha hardware'
!DEC$ ELSEIF DEFINED (IA64)
        type *,'Running on Integrity hardware'
!DEC$ ENDIF
        end
```

33

Conditionalized code
Sample Basic program

COM file

```
$!! if you VAX or Alpha system is older,
   ARCH_NAME may not be accepted
$!!   by f$getsyi... ARCH_TYPE (1-VAX,
   2=Alpha, 3=IA64) will be...
$ open/write out
   sys$disk:[]archdef.basic_include
$ write out "%LET %ARCH_TYPE =
   ",f$getsyi("arch_type")
$ close out
$ purge sys$disk:[]archdef.basic_include
$ basic archdef_bas
$ link archdef_bas
$ exit
```

Language file

```
 !
 !
    %INCLUDE "sys$disk:[]archdef.basic_include"
    program archdef_bas

    %IF (%ARCH_TYPE = 1)
        %THEN Print "Running on VAX"
        %ELSE %IF (%ARCH_TYPE = 2)
            %THEN Print "Running on Alpha"
            %ELSE  %IF (%ARCH_TYPE = 3)
              %THEN Print "Running on
  Integrity"
            %END %IF
        %END %IF
    %END %IF

end program
```

34

Conditionalized code Sample Cobol program

COM file

```
$!
$ if f$getsyi("ARCH_NAME") .EQS. "IA64"
$ then
$       cobol/conditional=I archdef_cob
$ else
$ if f$getsyi("ARCH_NAME") .EQS. "VAX"
$ then
$       cobol/conditional=V archdef_cob
$ else
$ if f$getsyi("ARCH_NAME") .EQS. "Alpha"
$ then
$       cobol/conditional=A archdef_cob
$ endif
$ endif
$ endif
```

Language file

```
identification division.
program-id. HW.
environment division.
data division.
procedure division.
p1.     display "Hello World".
\A      display "Running on Alpha".
\V      display "Running on VAX".
\I      display "Running on Integrity".

        stop run.
```

35

Conditionalized code Sample Pascal program

COM file

```
$ pascal archdef_pas
$ link archdef_pas
```

Language file

```
program example(output);

       %if %arch_name = "Alpha"
       %then
        var handle : integer := 0;
       %elif %arch_name = "IA64"
       %then
        var handle : integer64 := 0;
       %elif %arch_name = "VAX"
       %then
        var handle : integer := 0;
       %endif

       begin
       writeln('Program running on ',%system_name,
        ' ',%arch_name,
        ' ',%system_version);

       %if %arch_name = "Alpha"
       %then
        writeln('Running on Alpha');
       %elif %arch_name = "IA64"
       %then
        writeln('Running on Integrity');
       %elif %arch_name = "VAX"
       %then
        writeln('Running on VAX');
       %endif
      end.
```

36

Example – Moving from F77 to F90

- When using double precision float (REAL*8) for doing direct assignment (a=5.3)
  - F77 uses double precision
  - F90 uses single precision.
  - The result is slightly further away from the real 5.3 value.
- A computation will produce a different result with no error signaled.
- Possible solutions:
  - Fix the coding bug, as the assignment is wrong.
    - Change the assignment to a=5.3D0 or a=5.3_8
    - 5.3D0 works for both F77 and F90
  - Compile using the /ASSUME=FP_CONSTANT switch

37

IEEE floating- point

- This is one of the biggest porting issues.
- Itanium supports only IEEE floating-point in hardware
- On IA64 - IEEE floating-point is the default floating point format for the compilers.
  - VAX floating point formats will be supported when specified as a switch to the compilers
  - The compilers generate code to call conversion routines (performance hit).

38

```
AXP> ty wait.c
#include <stdio.h>
main()
{
float wait=7.0;

        printf("Waiting 7 seconds\n");
        lib$wait(&wait,0,0);
        printf("I'm done wainting..ciao...\n");

        return 0;
}
```

Executed on Alpha:

```
AXP> cc wait
AXP> link wait
AXP> r wait
Waiting 7 seconds
I'm done wainting..ciao...
```

39

Executed on I64:

```
I64> cc wait
I64> link wait
I64> r wait
Waiting 7 seconds
%SYSTEM-F-FLTINV, floating invalid operation, PC=FFFFFFFF82142760, PS=0000001B
%TRACE-F-TRACEBACK, symbolic stack dump follows
image     module    routine              line      rel PC           abs PC
LIBRTL                                        000000000016C752 FFFFFFFF82142752
LIBRTL                                        000000000020F430 FFFFFFFF821E5430
WAIT                                          0000000000010250 0000000000010250
WAIT                                          0000000000010180 0000000000010180
                                              0000000000000000 FFFFFFFF80B1A030
                                              0000000000000000 000000007AE1BEE0
```

The default floating point format used by LIB$WAIT is F_FLOAT, which does not match the default floating point format used on I64 (S_FLOAT)

40

Here is a modified version that will work on both platforms, using the native floating point formats

```
I64> ty wait_common.c
#include <stdio.h>
#include <arch_defs>
#include <libwaitdef>
main()
{
float wait=7.0;
#ifdef __alpha
        int mask = LIB$K_VAX_F;
#endif
#ifdef __ia64
        int mask = LIB$K_IEEE_S;
#endif
        printf("Waiting 7 seconds\n");
        lib$wait(&wait,0,&mask);
        printf("I'm done wainting..ciao...\n");

        return 0;
}
```

41

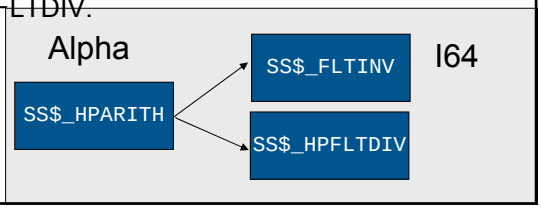
IMACRO

- On I64 the calling standard changed
  - We now use Intel's calling standard
  - IA64 only preserves register R4-R7 across routine calls
  - Alpha preserves R2-R15
  - Register numbering scheme has changed too
  - High-level languageprorams (like C,Bliss) this is not visible.
  - High-level languages might trash a register IMACRO assumed to be preserved (and vice versa)
  - Please reference the IMACRO porting guide for more details

42

Condition Handlers & SS$_HPARITH

- OpenVMS Alpha:
  - SS$_HPARITH is signaled for a number of arithmetic error conditions
- OpenVMS I64:
  - SS$_HPARITH is never signaled for arithmetic error conditions
  - the more specialized SS$_FLTINV and SS$_FLTDIV are used
- Requirement:
  - Update condition handlers to detect the more specialized error codes
  - To keep common code extend it for to also consider SS$_FLTINV and SS$_FLTDIV.



Quotas and Process settings

- OpenVMS I64 images are much larger, sometimes 3x-4x!
- Start with 5x your Alpha settings and adjust
  - BYTLM, FILLM, WSDEF, WSQUO, WSEXTENT, PGFLQUOT

44

If this is not enough….. there is more……

- We adopted Intel's calling standard. Code with knowledge about the calling standards will have to change
  - Stack/frame walking – the code will need to be modified to use the new LIB$*_INVO_* routines
  - Home grown stack switching/threading – the code will need to be ported to use Kernel Processes
- We adopted the ELF/DWARF formats. Code with knowledge about image format and debug format will have to change
  - Calling LIB$FIND_IMAGE_SYMBOL and friends does not count. The LIB$ routines were modified to support the new formats

45

Alignment faults

- Once the port of the application has been completed, you might want to look at alignment faults
  - Alignment faults are expensive on Alpha but are 100 times more expensive on IA64
  - The DEBUG SET MODULE/ALL command used to take 90 seconds. After fixing some alignment faults, it now takes 2 seconds.
  - DCL procedures take approx. 10% less time to execute after fixing alignment faults in DCL.
  - You may detect alignment faults using FLT extension in SDA or using SET BREAK/ALIGN option in the debugger
  - Some alignment faults are easy to fix, some are very hard and some

    are close to impossible.

46

FLT Alignment Fault Tracing

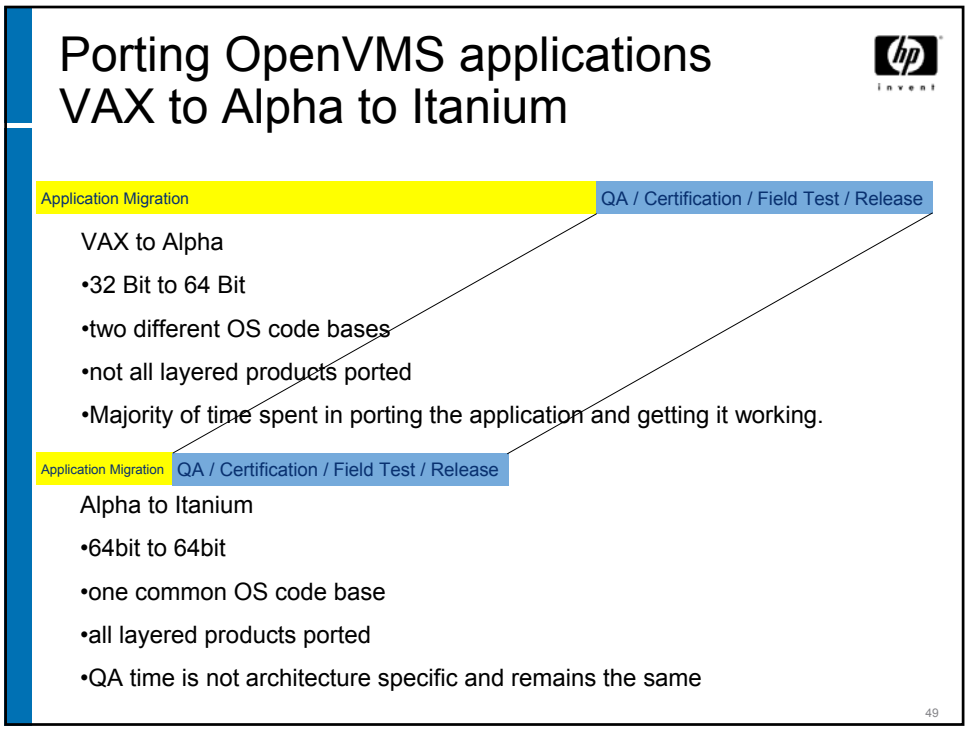
- Ideal is no alignment faults at all !
  - Poor code and unaligned data structures do exist
- Alignment fault summary…
  - `SDA> FLT START TRACE`
  - `SDA> FLT SHOW TRACE /SUMMARY`
  - `flt_summary.txt`
- Alignment fault trace...
  - `SDA> FLT START TRACE`
  - `SDA> FLT SHOW TRACE`
  - `flt_trace.txt`

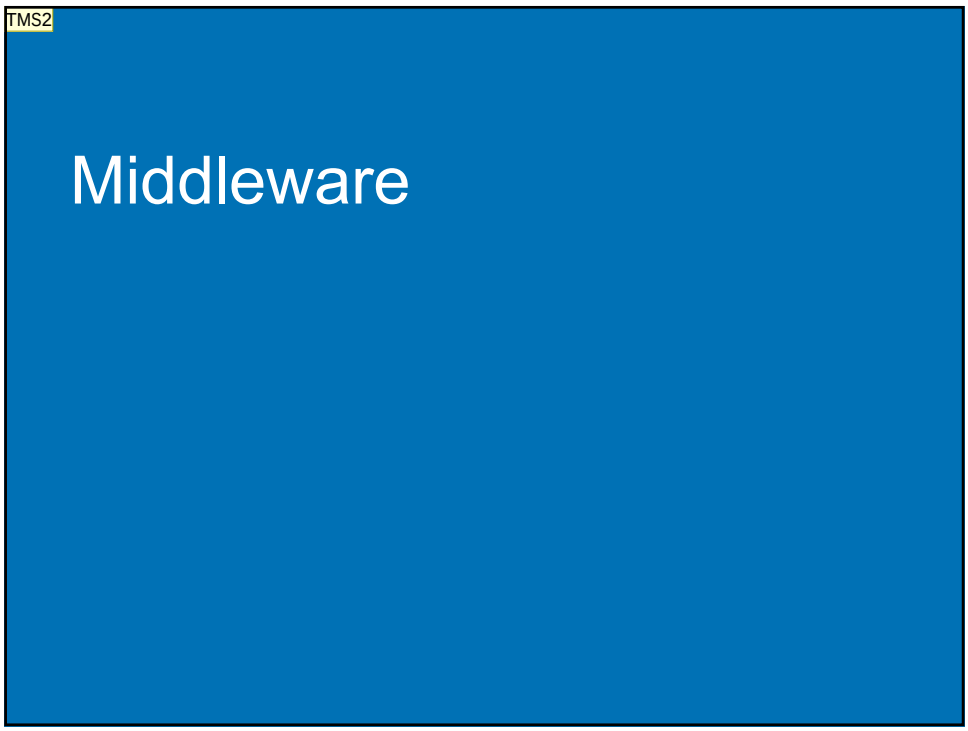
47

Cost of alignment faults

Relative cost of access	Aligned	Unaligned, compiler knows	Unaligned, compiler doesn't know
OpenVMS VAX	1	1	1
OpenVMS Alpha	1	4	10
OpenVMS I64	1	10	100

48


Porting OpenVMS applications
VAX to Alpha to Itanium
Application Migration
QA / Certification / Field Test / Release
VAX to Alpha
•32 Bit to 64 Bit
•two different OS code bases
•not all layered products ported
•Majority of time spent in porting the application and getting it working.
Application Migration
QA / Certification / Field Test / Release
Alpha to Itanium
•64bit to 64bit
•one common OS code base
•all layered products ported
•QA time is not architecture specific and remains the same
49

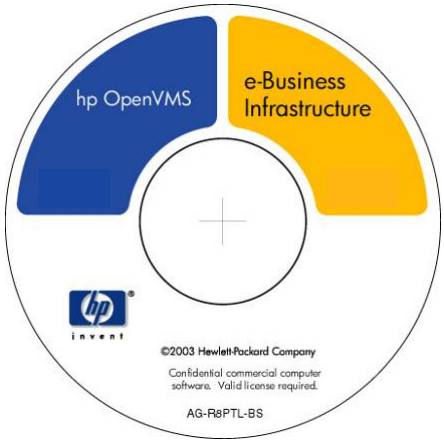

TMS2
Middleware

Folie 50

TMS2 How much do I need to talk about here? These seem to many slides, following from here!?
Thomas Siebold; 3.2.2005

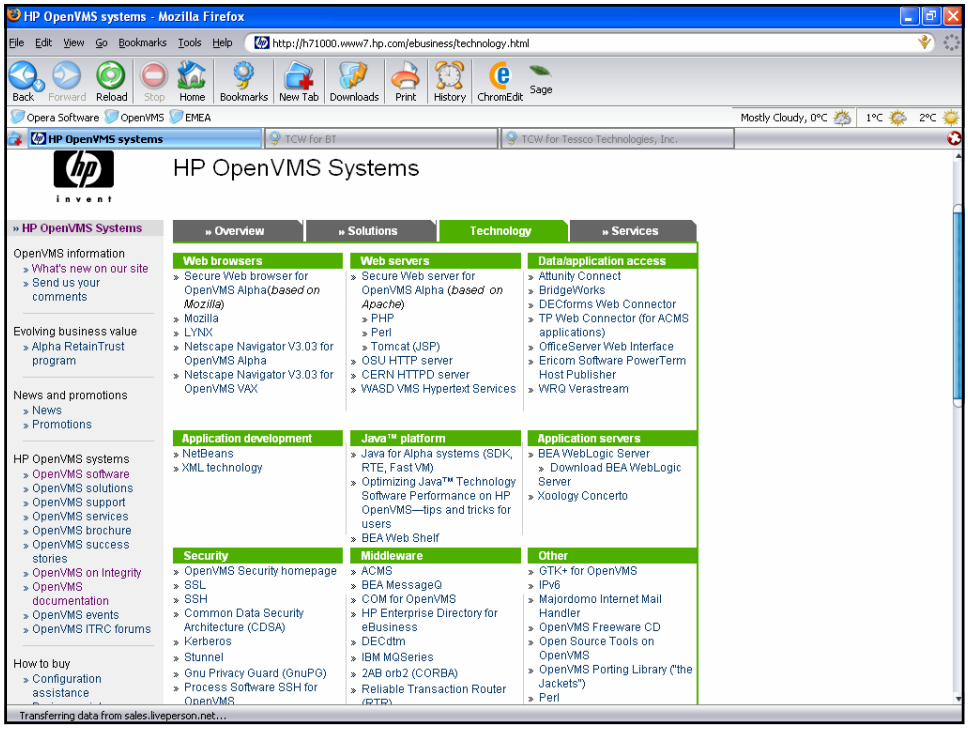
e-Business Infrastructure Packaging for OpenVMS

- The key e-Business, integration, and Internet technologies are packaged with OpenVMS on the e-Business Infrastructure CD



- Secure Web Server (based on Apache) including PHP, Perl, and JSP (Tomcat) support
- Secure Web Browser (based on Mozilla)
- Software Development Kit (SDK) for the Java™ platform
- Reliable Transaction Router
- Enterprise Directory (LDAP)
- COM
- BridgeWorks
- NetBeans
- Simple Object Access Protocol (SOAP) Toolkit (based on Apache Axis)
- UDDI Client Toolkit

51



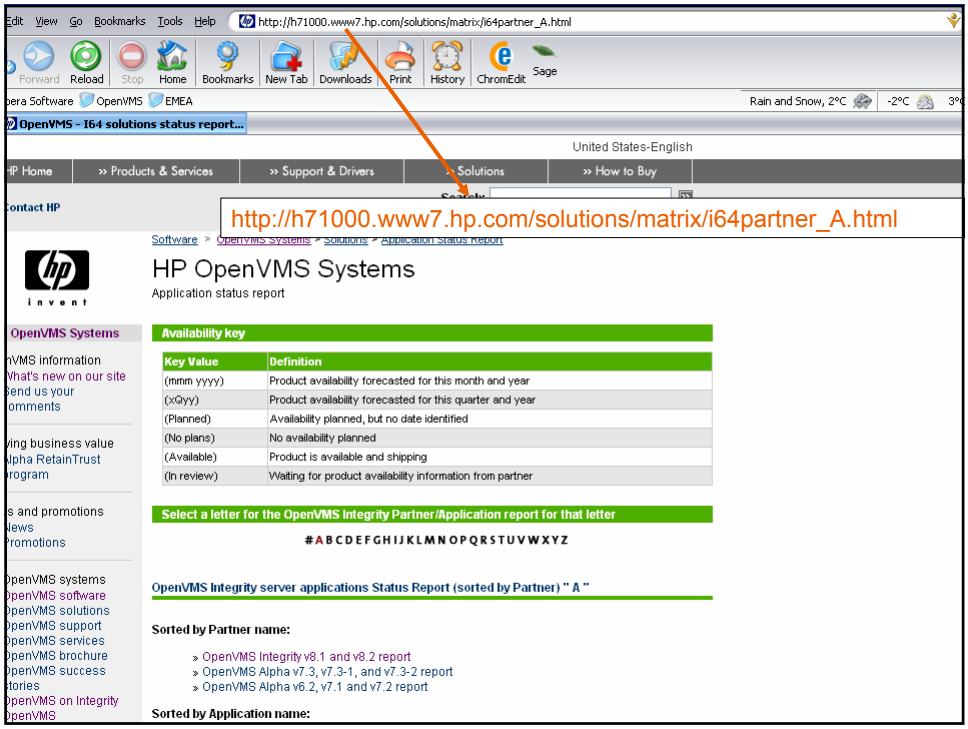
ISVs

OpenVMS ISV momentum: Cross-section of leading ISVs porting to Integrity Servers

Over 800 applications and service offerings from 370 partners currently planned, more every week.

250 apps and services declared ready - 80% of service offers available today!

54


http://h71000.www7.hp.com/solutions/matrix/i64partner_A.html
HP OpenVMS Systems
Application status report
Availability key
Key Value
Definition
(mmm yyyy)
Product availability forecasted for this month and year
(xQyy)
Product availability forecasted for this quarter and year
(Planned)
Availability planned, but no date identified
(No plans)
No availability planned
(Available)
Product is available and shipping
(In review)
Waiting for product availability information from partner
Select a letter for the OpenVMS Integrity Partner/Application report for that letter
#ABCDEFGHIJKLMNOPQRSTUVWXYZ
OpenVMS Integrity server applications Status Report (sorted by Partner) " A "
Sorted by Partner name:
» OpenVMS Integrity v8.1 and v8.2 report
» OpenVMS Alpha v7.3, v7.3-1, and v7.3-2 report
» OpenVMS Alpha v6.2, v7.1 and v7.2 report
Sorted by Application name:

Next Steps....

How do I start?

- How do I start porting my application?
  - There are several approaches:
    - Re-examine the application for potential “hot spots”
    - Compile/link and see what’s broken
    - Compile and examine new messages
- There is no right approach, take the one that you feel most comfortable with

57

The ultimate porting tool

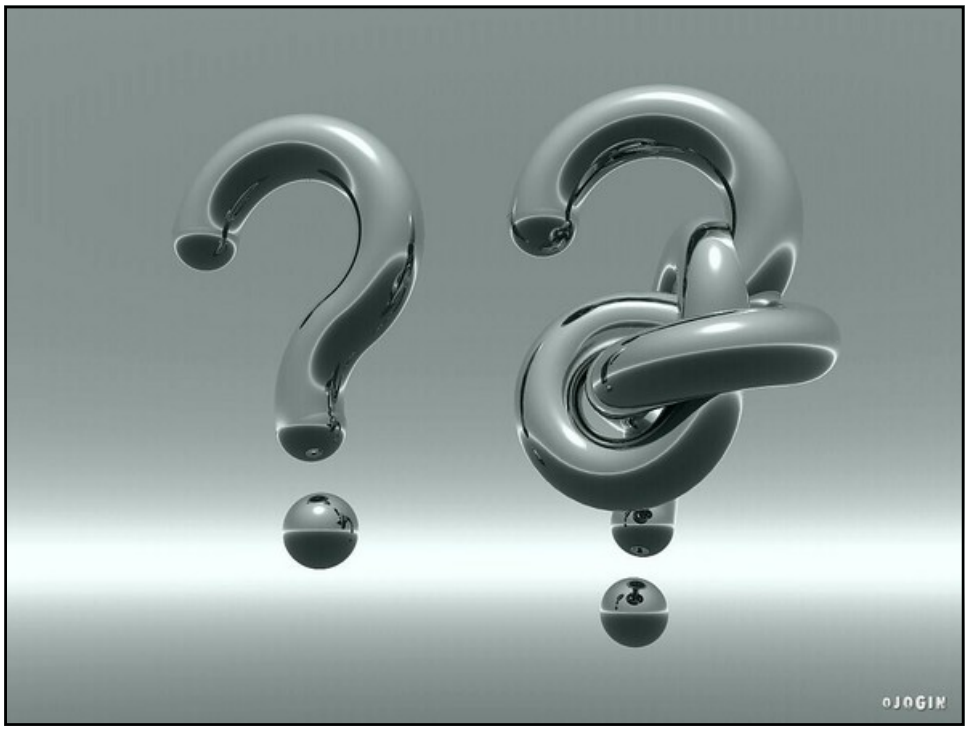
Cook books can be rather useful,
But you have to trust them.
And mistakes happen.

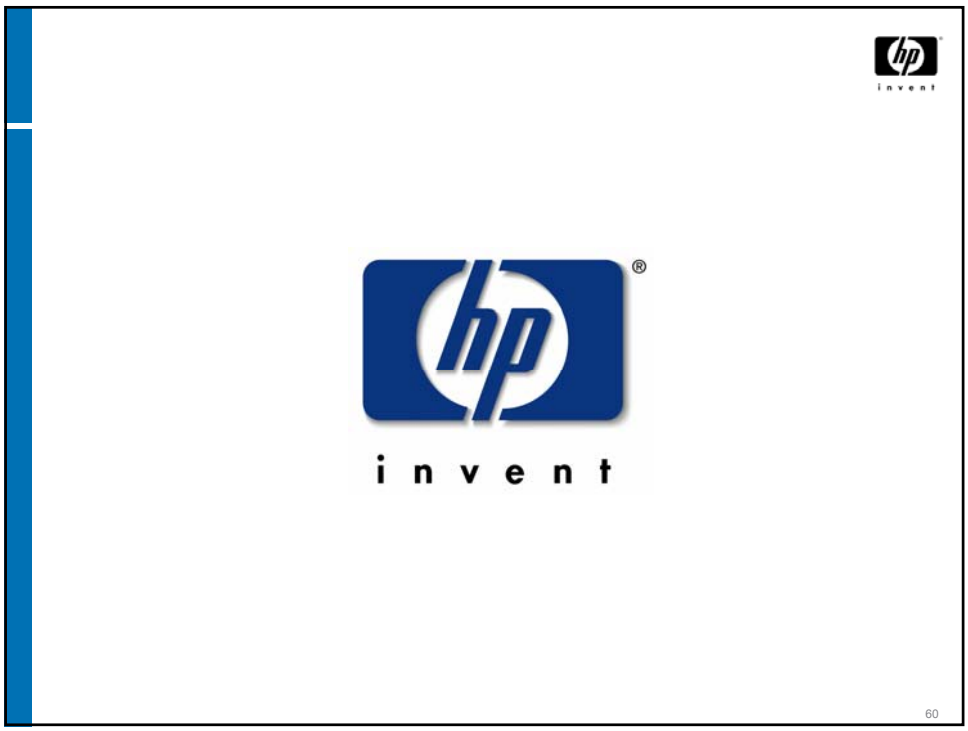
1 kg of salmon
10 kg of salt
5 g of pepper …

The absolutely best porting tool is easy to use …

Nothing beats understanding!

58


©JOGIN


invent
invent
60