

**BRUDEN** **OSSG**  
On-Shore Systems Group

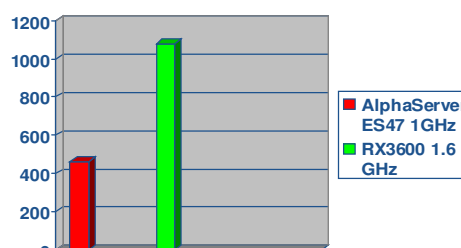


**Guy Peleg**  
Senior Member of the Technical Staff  
Director of EMEA Operations  
[guy.peleg@bruden.com](mailto:guy.peleg@bruden.com)

**BRUDEN** **OSSG**  
On-Shore Systems Group

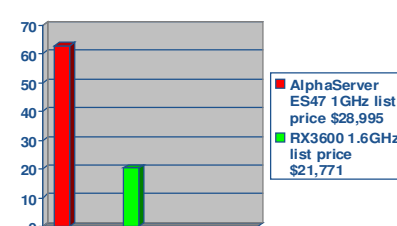
## Why Should I Port?

Application X measuring transactions per second



Server Model	Transactions per Second
AlphaServer ES47 1GHz	~480
RX3600 1.6GHz	~1100

Application X cost of 1 transaction in USD



Server Model	Cost of 1 Transaction (USD)
AlphaServer ES47 1GHz (list price \$28,995)	~65
RX3600 1.6GHz (list price \$21,771)	~22

2

**BRUDEN** **OSSG**  
On-Shore Systems Group

## CPU Performance Comparisons

Simple Integer Computations – single stream

Processor	Rating (single CPU)
rx3600 1.6GHz/9MB	~580
rx2620 1.6GHz/4MB	~550
rx2600 1.5GHz/6MB	~520
rx4640 1.3GHz/3MB	~480
ES45 1GHz	~450
GS1280 1.15GHz	~420

More is better  
(Small number of computations in test do not take full advantage of EPIC)

Floating Point Computations – single stream

Processor	Elapsed Time
rx6600 1.6GHz	~250
rx3600 1.6GHz	~300
GS1280 1.15GHz	~450
ES45 1.0GHz	~550
GS1280 1.15GHz	~600

Less is better

The Itanium processors are fast.

- Faster cores
- 128 general purpose and 128 floating point registers
- Large caches compared to Alpha EV7

Various SPEC benchmarks also show the Itanium processors outperforming Alpha processors

3

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Memory Bandwidth

Memory Bandwidth (small servers)

Processor	Memory Bandwidth (MB/sec)
rx4640 1.3GHz/3MB	~2000
rx2600 1.5GHz/6MB	~2500
rx2620 1.6GHz/4MB	~2500
rx3600 1.6GHz/9MB	~3200
ES45 1GHz	~1800

More is better

Memory Bandwidth (large servers)  
Computed via memory test program – single stream

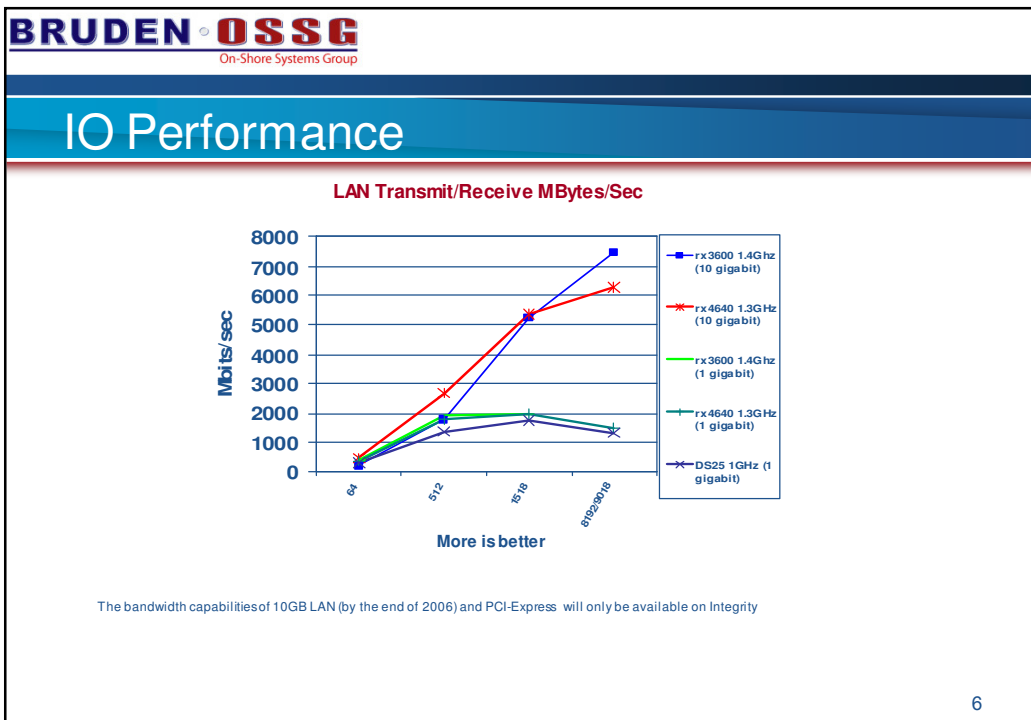
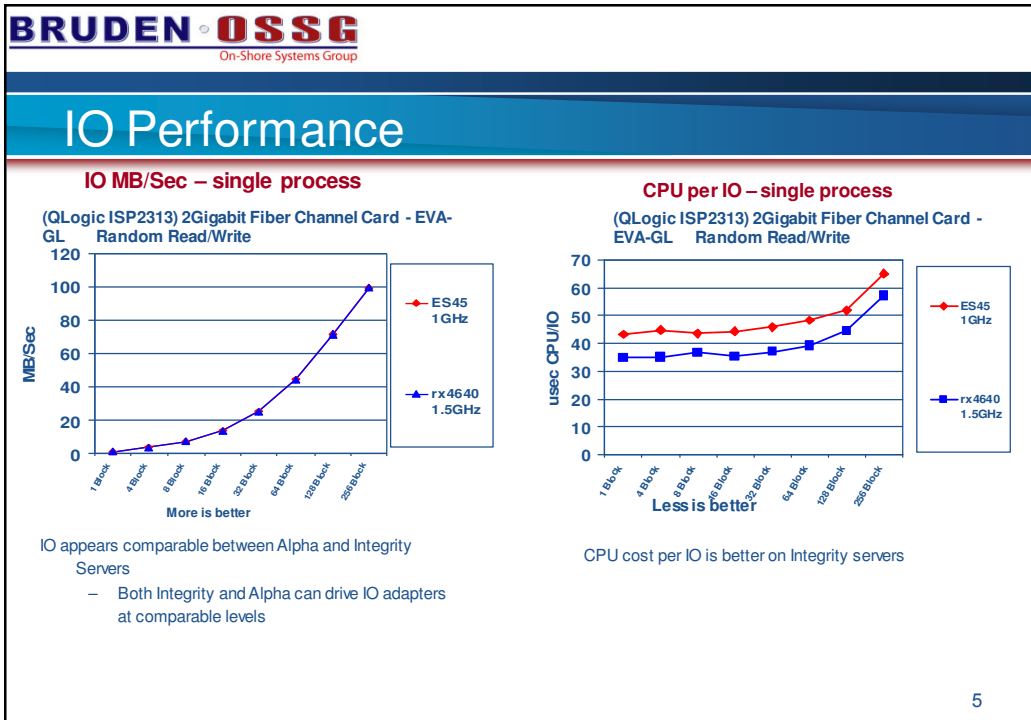
Configuration	Memory Bandwidth (MB/sec)
SuperDome 1.5GHz (sx1000) 1 Cell	~1600
SuperDome 1.5GHz (sx1000) 2 Cells	~1200
SuperDome 1.5GHz (sx1000) 4 Cells	~1200
rx7640 1.6GHz 1 Cell	~2700
rx7640 1.6GHz 2 Cell	~2300
rx8640 1.4GHz 4 Cell	~1800
GS1280 1.15GHz	~1700

More is better

The small Integrity Servers have very good memory bandwidth

- Applications which move memory around or heavily use caches or RAMdisks should perform well

4



**BRUDEN** On-Shore Systems Group **OSSG**

## Porting to OpenVMS I64

- Porting applications to I64 is easy
- Re - compile
- Re - link
- Test (if you must)

7

**BRUDEN** On-Shore Systems Group **OSSG**

## Porting to OpenVMS I64

- Major changes 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 some might affect your application

8

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Build tools

- Try building your application on Alpha, using the latest version of the compilers you are using to uncover any hidden bugs/changes. This will make the move to the new platform easier.
- You might get away with relying on uninitialized variables, but on I64 you will be severely punished

9

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Conditionalized code

- This is the first (and easiest) step to take
  - Usually, IA64 should take what used 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");
}

```

10

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Conditionalized code

- Macro

```
.IF DF ALPHA          .IF DF IA64
.ENDC                  .ENDC
```
- C / C++

```
#ifdef __alpha        #ifdef __ia64
#endif                #endif
```
- Bliss

```
%IF ALPHA %THEN%IF IA64 %THEN
%FI                    %FI
```

11

**BRUDEN** **OSSG**  
On-Shore Systems Group

## 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
#ifdef __vax
    printf("This program is not running on VAX");
#endif
}
```

12

**BRUDEN** On-Shore Systems Group **OSSG**

## 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
```


13

**BRUDEN** On-Shore Systems Group **OSSG**

## IEEE floating- point

- 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 supported when specified as a switch to the compilers*
  - *The compilers generate code to call conversion routines (performance hit).*


14



IEEE floating- point

- To use IEEE change the source to use S & T versions of the APIs.
  - Some functions (like sin, cos,...) already know how to handle IEEE and require no changes to the application.

15



IEEE floating- point

```

$ ty float_test.c
#include <stdio.h>
#include <libdtdef.h>
#include <dscscrip>
#include <ssdef>

// prototypes
int lib$cvtf_to_internal_time();
int sys$fao();
int lib$put_output();
int lib$signal();

static $DESCRIPTOR (fao_desc, "Converted delta time: !%T");

main () {
    float f1;
    unsigned long long int delta1;
    int status;
    char output[50]={0};
    struct dsc$descriptor_s outdesc;
    short int length;

    //init the descriptor
    outdesc.dsc$w_length = sizeof(output);
    outdesc.dsc$a_pointer = (char *)output;
    outdesc.dsc$b_class = DSCSK_CLASS_S;
    outdesc.dsc$b_dtype = DSCSK_DTYPE_T;


    f1 = 156.45;

    printf("This program converts %f seconds to delta time\n", f1);
    status = lib$cvtf_to_internal_time(&LIB$K_DELTA_SECONDS_F, &f1, &delta1);
    if (!(status&1))
        lib$signal(status);
    if ((sys$fao(fao_desc, &length, &outdesc, &delta1))&1)
        lib$put_output(&outdesc);
}

```

Convert seconds to delta time

16



IEEE floating- point

**Executed on Alpha:**

```

AXP> cc float_test
AXP> link float_test
AXP> r float_test
This program converts 156.449997 seconds to delta time
Converted delta time: 00:02:36.44

```


**Executed on IA64:**

```

I64> cc float_test
I64> link float_test
I64> r float_test
This program converts 156.449997 seconds to delta time
%LIB-F-IVTIME, invalid time passed in, or computed
%TRACE-F-TRACEBACK, symbolic stack dump follows
image      module      routine      line      rel PC      abs PC
FLOAT_TEST  FLOAT_TEST  main        4514      0000000000000174 0000000000030174
FLOAT_TEST  FLOAT_TEST  __main     0         0000000000000064 0000000000030064
                                0         FFFFFFFF8025DE94 FFFFFFFF8025DE94

```

17



IEEE floating- point

**Compiled again using the /FLOAT qualifier**

```

I64> cc float_test/float=g_float
I64> link float_test
I64> r float_test
This program converts 156.449997 seconds to delta time
Converted delta time: 00:02:36.44
IPL31>

```

**Note the program would fail on Alpha as well if compiled with ieee\_float**

```

AXP> cc float_test/float=ieee
AXP> link float_test
AXP> r float_test
This program converts 156.449997 seconds to delta time
%LIB-F-IVTIME, invalid time passed in, or computed
%TRACE-F-TRACEBACK, symbolic stack dump follows
image      module      routine      line      rel PC      abs PC
FLOAT_TEST  FLOAT_TEST  main        4514      0000000000000174 0000000000030174
FLOAT_TEST  FLOAT_TEST  __main     0         0000000000000064 0000000000030064
                                0         FFFFFFFF8025DE94 FFFFFFFF8025DE94

```

18



LIB\$WAIT

```

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 waiting..ciao...\n");

    return 0;
}

```


**Executed on Alpha:**

```

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

```

19



LIB\$WAIT

**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  LIBRTL      00000000016C752 FFFFFFFF82142752
LIBRTL  LIBRTL      00000000020F430 FFFFFFFF821E5430
WAIT    WAIT        000000000010250 000000000010250
WAIT    WAIT        000000000010180 000000000010180
WAIT    WAIT        000000000000000 FFFFFFFF80B1A030
WAIT    WAIT        000000000000000 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)*

20



## LIB\$WAIT

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 waiting..ciao...\n");

return 0;
}

```

21



## 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

22

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Example – Moving from F77 to F90

```
IPL31> type float.for
      REAL*8          TEST

      TEST = 5.3
      PRINT 100,TEST
100   FORMAT(F,' assigned as TEST = 5.3 ')

      TEST = 5.3D0
      PRINT 200,TEST
200   FORMAT(F,' assigned as TEST = 5.3D0')

      END

IPL31> for float
IPL31> link float
IPL31> r float
5.3000001907348633 assigned as TEST = 5.3
5.2999999999999998 assigned as TEST = 5.3D0
IPL31> for/assume=fp_const float
IPL31> link float
IPL31> r float
5.2999999999999998 assigned as TEST = 5.3
5.2999999999999998 assigned as TEST = 5.3D0
```

23

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Fortran compile time initialization

- Very large common blocks with fields initialized at compile time may result in excessively large object files and long compile times
- This problem does not exist on Alpha
- Perform data initialization at runtime or move the initialized data to a smaller common block to avoid the problem

24

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Improperly declared functions and data

- C function declarations that points to objects that are not functions, may work on Alpha but will fail to work on IA64
  - Also seen with Bliss and Fortran
- The problem may manifest itself in many ways
  - For whatever reason, the most common symptom is a call to routine CLI\$DCL\_PARSE that fails with CLI-E-INV TAB

```
external int my_cld();
    status = cli$dcl_parse(&cmdstr, my_cld,
                        lib$get_input, lib$get_input);
external int my_cld;
    status = cli$dcl_parse(&cmdstr, &my_cld,
                        lib$get_input, lib$get_input);
```

25

**BRUDEN** **OSSG**  
On-Shore Systems Group


## Improperly declared functions and data

- The Fortran variant of the fix would be
 

```
CDEC$ ATTRIBUTES EXTERN :: MY_CLD
```
- The Linker detects this condition
 

```
%LINK-I-DIFTYPE, symbol TEST_CLD of type OBJECT cannot be referenced as type FUNC
module: TEST
file: $1$DKC600:[IA64]TEST.OBJ;2
```


26



C++ Default Model

- The default value for the /MODEL qualifier is ARM on Alpha and ANSI on IA64
- /MODEL is ignored on IA64
  - ANSI is the only supported format
  - May require changes to existing code
    - See the C++ release notes for full details:
   
[http://h71000.www7.hp.com/commercial/cplus/ia64\\_doc/rni64.html](http://h71000.www7.hp.com/commercial/cplus/ia64_doc/rni64.html)
- Compiled with /MODEL=ARM string literals are of type "array of char"
- Compiled with /MODEL=ANSI string literals are of type "array of const char"

27



C++ Default Model

```

$ type error.cxx
#include <iostream.h>
double divide (double x, double y)
{
    if (y==0)
        throw "divide by 0";
    else
        return (x/y);
}
void main()
{
    try{
        cout<<"5/2="<<divide(5.0, 2.0) <<endl;
        cout<<"5/0="<<divide(5.0, 0.0) <<endl;
        cout<<"5/1="<<divide(5.0, 1.0) <<endl;
    }
    catch (char *msg){
        cout<<msg<<endl;
    }
    cout<<"end of main"<<endl;
}

```

Simple Exception Signaling  
in C++

28

**BRUDEN** **OSSG**  
On-Shore Systems Group

## C++ Default Model

**Executed on Alpha:**

```

AXP> cxx error
AXP> cxxlink error
AXP> r error
5/2=2.5
5/0=divide by 0
end of main

```

**Executed on IA64:**

```

I64> cxx error
I64> cxxlink error
I64> r error
5/2=2.5
%CXXL-F-TERMINATE, terminate() or unexpected() called
%TRACE-F-TRACEBACK, symbolic stack dump follows
image  module  routine  line  rel PC  abs PC
TRACEBACK - Exception occurred during traceback processing
CXXL$LANGRTL 0 0000000000054B90 FFFFF802090C0B90
TRACEBACK - Exception occurred during traceback processing
CXXL$LANGRTL 0 0000000000041190 FFFFF802090AD190
CXXL$LANGRTL 0 FFFFFFFF803DD150 FFFFFFFF803DD150
image  module  routine  line  rel PC  abs PC
TRACEBACK - Exception occurred during traceback processing
CXXL$LANGRTL 0 00000000000415B0 FFFFF802090AD5B0
TRACEBACK - Exception occurred during traceback processing
CXXL$LANGRTL 0 0000000000054280 FFFFF802090C0280
ERROR  ERROR  divide  #5 0000000000000160 0000000000101E0
ERROR  ERROR  main  #13 0000000000000320 0000000000103A0
ERROR  ERROR  ELF$TFRADR #1788 0000000000000730 0000000000107B0
0 FFFFFFFF80B6C630 FFFFFFFF80B6C630
DCL 0 000000000006BD20 000000007AE25D20
%TRACE I LINENUMBER. Leading '#' specifies a source file record number.

```

29

**BRUDEN** **OSSG**  
On-Shore Systems Group

## C++ Default Model

Here is a modified version that will work on both Platforms:

```


$ type error.cxx
#include <iostream.h>
double divide (double x, double y)
{
    if (y==0)
        throw "divide by 0";
    else
        return (x/y);
}
void main()
{
    try{
        cout<<"5/2="<<divide(5.0, 2.0) <<endl;
        cout<<"5/0="<<divide(5.0, 0.0) <<endl;
        cout<<"5/1="<<divide(5.0, 1.0) <<endl;
    }

    catch (const char *msg) {
        cout<<msg<<endl;
    }

    cout<<"end of main"<<endl;
}

```


30



## C++ Optimization

- Default optimization options are consistent with Alpha.
- Programs using floating point should try compiling with  
   /assume=(noaccuracy\_sensitive,nomatherno)
  - Result changing optimizations are not turned on by default
  - Similar optimization to Intel Linux compiler
- Try /opt=level=5
  - More aggressive optimization
  - Main differences are for floating point code but might benefit integer as well.

31



## Hardware Model

- The hardware model for all IA64 systems is set to 4096


```

I64> write sys$output f$getsyi("hw_name")
HP rx2600 (900MHz/1.5MB)
I64> write sys$output f$getsyi("hw_model")
4096

```

- Any code relying on the hardware model of the system has to change.
- SHOW MEMORY used to determine the page size based on the following algorithm:  
   if (hardware\_model >= 1024)  
     page\_size = 8192;  
   else page\_size = 512;
- SHOW MEMORY has been modified to use the SYI\$\_PAGE\_SIZE item code on VAX/Alpha and IA64.


32



## IMACRO

- On I64 the calling standard changed
  - VMS now use Intel's software conventions
  - IA64 only preserves register R4-R7 across routine calls (Alpha preserves R2-R15)
  - For programs written in high-level languages (like C, Bliss) this difference is not visible.
  - When MACRO-32 programs added you have to start worrying about how to pass register parameters between languages.
  - High-level languages might trash a register IMACRO assumed to be preserved (and vice versa)
  - Please reference the IMACRO porting guide for more details

33



## IMACRO - coding changes

- JSB\_ENTRYs that reference R16-R21 should be changed to .CALL\_ENTRYs that reference n(AP)


```

XFC_COMPARE_QIOS:
.JSB_ENTRY
MOVL    (R16), R0
MOVL    (R17), R1
EVAX_SUBQ CFS$Q_TOTALQIOS (R0), -
          CFS$Q_TOTALQIOS (R1), R0
RSB

XFC_COMPARE_QIOS:
.CALL_ENTRY
MOVL    @4(ap), R0
MOVL    @8(ap), R1
EVAX_SUBQ CFS$Q_TOTALQIOS (R0), -
          CFS$Q_TOTALQIOS (R1), R0
RET
          
```

- Code that is moving data in R16-R21 and then perform a JSB should be modified to use \$SETUP\_CALL64 and \$CALL64

34



## IMACRO - coding changes

- MACRO32 JSBs to any other language (Bliss/C) routines
  - If IMACRO can't determine the language of a target JSB, the following message will be generated:  
  

```
JSB      (R6)  ;ALLOCATE AND INSERT ENTRY IN SYMBOL TABLE
%IMAC-I-CODGENINF, (1) Indirect JSB with default linkage
```
- .USE\_LINKAGE directive with the LANGUAGE=OTHER option tells iMacro that the target routine is written in a language other than IMACRO. IMACRO will than save and restore R2,R3,R8-R15 around the JSB except for registers in the output mask.

35



## Lock Pages in Working Set

- 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

36

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Condition Handlers Use of SS\$\_HPARITH

On OpenVMS Alpha, SS\$\_HPARITH is signaled for a number of arithmetic error conditions. On OpenVMS I64, SS\$\_HPARITH is never signaled for arithmetic error conditions; instead, the more specialized SS\$\_FLTINV and SS\$\_FLTDIV error codes are signaled on OpenVMS I64.

Update condition handlers to detect these more specialized error codes. In order to keep code common for both architectures, wherever the code refers to SS\$\_HPARITH, extend it for OpenVMS I64 to also consider SS\$\_FLTINV and SS\$\_FLTDIV.

```

graph LR
    subgraph Alpha
        A[SS$_HPARITH]
    end
    subgraph I64
        B[SS$_FLTINV]
        C[SS$_HPFLTDIV]
    end
    A --> B
    A --> C
  
```

37

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Up yours!

Quotas and process settings

- OpenVMS I64 images are much larger, sometimes 3x-4x
- Ensure your pgflquo and bytlim are (at least) 4x-10x your Alpha settings.
  - \$ set default sys\$system
  - \$ run authorize
  - UAF> mod your\_account/pgflquo=nnnnnn
  - UAF> mod your\_account/bytlim=nnnnnn

38

**BRUDEN** On-Shore Systems Group **OSSG**

## THREADS

- THREADCP tool was not ported to OpenVMS I64
  - Relink to change threads related characteristics of an image
  - Use the new SET IMAGE command
  
- If your application increases the stack size for a thread from the default size, you should increase it more

HP recommends starting with an increase of three 8-Kb pages (24576 bytes).


39

**BRUDEN** On-Shore Systems Group **OSSG**

## There is more.....

- The topics covered so far are the most common issues seen by people porting their applications. Here is a list of less common (and much more complicated) issues.
- VMS adopted Intel's calling standard. Code with knowledge about the calling standard 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 KPs
- VMS 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

40




## Reading EXE and OBJ files

- Use ANALYZE/IMAGE vs. parsing the EXE file.
- We are looking at adding a callable interface into SHOW/SET image.

ANALYZE/IMAGE	DCL Symbol that is set	Sample Value
/SELECT=ARCHITECTURE	ANALYZE\$ARCHITECTURE	OpenVMS IA64
/SELECT=NAME	ANALYZE\$NAME	"DECC\$COMPILER"
/SELECT=IDENTIFICATION=IMAGE	ANALYZE\$IDENTIFICATION	"C T7.1-003"
/SELECT=IDENTIFICATION=LINKER	ANALYZE\$LINKER IDENTIFICATION	"Linker I02-08"
/SELECT=LINK TIME	ANALYZE\$LINK TIME	"6/29/2004 4:26:35 PM"
/SELECT=FILE TYPE	ANALYZE\$FILE TYPE	Image
/SELECT=IMAGE TYPE	ANALYZE\$IMAGE TYPE	Executable

41



## Infrastructure changes in OpenVMS V8.2

- HP 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 may need to be modified
    - Examples: hard-coded data structure sizes and assumptions about the relative locations of fields within a data structure

42

**BRUDEN** On-Shore Systems Group **OSSG**

## Alignment faults

- Once the port of the application has been completed, look for alignment fault
- Biggest “performance killer” on IA64
- Tools for detecting alignment faults
  - MONITOR ALIGN (V8.3)
  - FLT
  - DEBUG SET BREAK/UNALIGN

43

**BRUDEN** On-Shore Systems Group **OSSG**

## Wait a second....I don't have the sources...

- OpenVMS Migration Software for HP AlphaServer Systems to HP Integrity Servers (OMSAIS)
- Utility that translates Alpha executables and shareable images to I64
- Supports translation of images written in: C, C++, Fortran, COBOL, BLISS, MACRO-32, MACRO-64 and PASCAL

44

**BRUDEN** **OSSG**  
On-Shore Systems Group

## OMSAIS

- OMSAIS includes two components:
  - AEST (Alpha Environment Software Translator)
  - TIE (Translated Image Environment)
- TIE provides run-time support for translated images
  - Integrated into V8.2-1
  - Separate download for V8.2
- Free download available from:
 

<http://h71000.www7.hp.com/openvms/products/omsva/omsais.html>

45

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Hyperthreading with Stalls vs Hyperthreading with No Stalls

**Serial Execution with Stalls (no Hyperthreading)**

**Hyperthreading with Stalls**

**Serial Execution with No Stalls (no Hyperthreading)**

**Hyperthreading with No Stalls**

46

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Two Cores vs Hyperthreading (NoStalls)

**Serial Execution with No Stalls on Two Cores**

**Hyperthreading with No Stalls**

47

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Sample Run on Single Core

```

Session 1
$ set process/aff/perm/set=31
$ r tc_cf
cycle interval = 0.00000000250000 seconds
cycles 10282355676
time to execute == 25.70588919000000 seconds
Wall clock time: 49143ms
System CPU time: 25620ms
Session2
$ set process/aff/perm/set=31
$ r tc_cf
cycle interval = 0.00000000250000 seconds
cycles 10353364312
time to execute == 25.88341078000000 seconds
Wall clock time: 49225ms
System CPU time: 25900ms
$

```

48

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Sample Run on Single Core with CoThreads

```

Session 1
$ set process/aff/perm/set=31
$ r tc_cf
cycle interval = 0.00000000250000 seconds
cycles 14092713701
time to execute == 35.23178425250000 seconds
Wall clock time: 35322ms
System CPU time: 18640ms
$
Session 2
$ set process/aff/perm/set=63
$ run tc_cf
cycle interval = 0.00000000250000 seconds
cycles 14346088453
time to execute == 35.86522113250000 seconds
Wall clock time: 35950ms
System CPU time: 19260ms
$

```

49

**BRUDEN** **OSSG**  
On-Shore Systems Group

## Sample Run on Two Cores

```

Session 1 (Still affinitized to CPU 31)
$
$ r tc_cf
cycle interval = 0.00000000250000 seconds
cycles 9572697013
time to execute == 23.93174253250000 seconds
Wall clock time: 23997ms
System CPU time: 23920ms
$
Session 2 (Non-Co-Thread CPU)
$ set process/aff/perm/set=32/clear=63
$ stop/cpu 63
%SMP-I-CPUTRN, CPU #63 was removed from the active set.
$
$ r tc_cf
cycle interval = 0.00000000250000 seconds
cycles 9726287048
time to execute == 24.31571762000000 seconds
Wall clock time: 24398ms
System CPU time: 24290ms
$

```

50

**BRUDEN** On-Shore Systems Group **OSSG**

## Porting Services from BRUDEN-OSSG

- BRUDEN engineers are experts in OpenVMS performance and migration to Itanium
- BRUDEN experts were part of the team ported OpenVMS to Itanium
- BRUDEN successfully ported many mission critical customers throughout Europe
- BRUDEN can help you to get the most out of Montecito HyperThreading

51

**BRUDEN** On-Shore Systems Group **OSSG**

## Questions?

**BRUDEN-OSSG thanks you for attending this session.**

**See us at [www.BRUDENOSSG.com](http://www.BRUDENOSSG.com) for:**

- *Performance analysis*
  - *(Performance Results Or No Expense)*
- *Porting assistance*
- *Special OPS (OpenVMS Programming Services)*
- *Support*

52