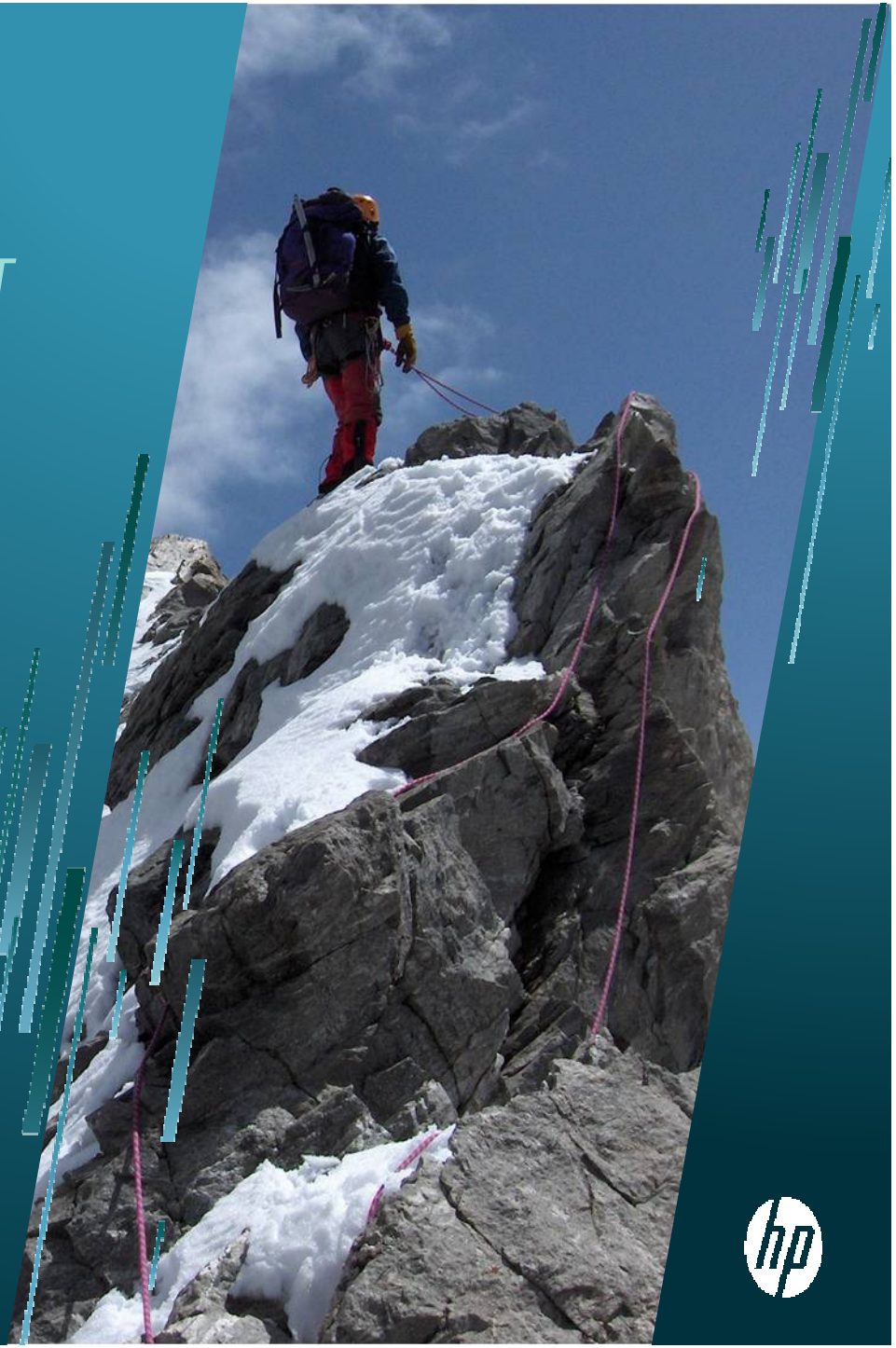
# APPLICATION IMPROVEMENT AND OPTIMIZATION

## PAST EXPERIENCES



Sorin Cheran & Eric Mena

# Agenda

- Compiler issues
  - elsA :repeatability problem when using intel compiler
  - Actipole: convergence and performance issues on different mathematical libraries

- How to improve your application: "The Actran Usecase"

# elsA – Problem Description

- When compiling with the Intel Compiler elsA would:

P1. Deliver different results ( repeatability issue): 2 consecutive runs with the same options and on the same testcase would produce different results.

P2. Optimization differences:  Compiling with "-O2" option would produce different results than when compiling with the "-O1" option on the same testcase.

# elsA – People Involved and time frame

- People involved:


Time Frame

- Problem signaled: May 2010
- Workaround proposed and issue closed: August 2010 ( delayed due to holidays of different people involved).

# elsA – Resolution approaches.

- Two approaches have been followed in parallel

- 1. Intel with HP - Intel Compiler team was looking for bugs/issue in their mkl library by running the testcases that could not produce repeatable results.

- 2. HP – has been working in decomposing the elsA code and identifying the module that was responsible for the repeatability issue and for the sensitivity to optimization flags.

# elsA Solution

- ...10.000 tests later we have a Solution

- BUG: implementation of pow() and cbrt()  in Intel libraries

- Proposed Workaround:
  - Changes to source code
    - *Modifying files like: ./Def/Global/DefFortranGlobal.h, Tur/Trp/TurCompEARSMgdmaeF, ./Geo/Grid/GeoCompCellDimF to prevent the compiler to use CBRT() call instead of POW(). The CBRT() call in Intel's math library seems to have a severe precision problem*
  - Changes to Make_intelIA32em.mk
    - *add "fp-model precise" option*
    - *include the libm.a in the library path before Intel libs.*

# elsA – Lessons Learnt

1. Problem gets solved faster if singled to the right people.

2. Always involve the users

3. Ask for as many brainstorming sessions as needed to try to find as many paths possible.

4. Always escalate to the ISV

5. Patience ☺

6. Follow later with the ISV to check updates of BUGS in new releases.

# Actipole: Problem Description

P1. Convergence problems when using the Intel compiler and the MKL library ( not on all the usecases)

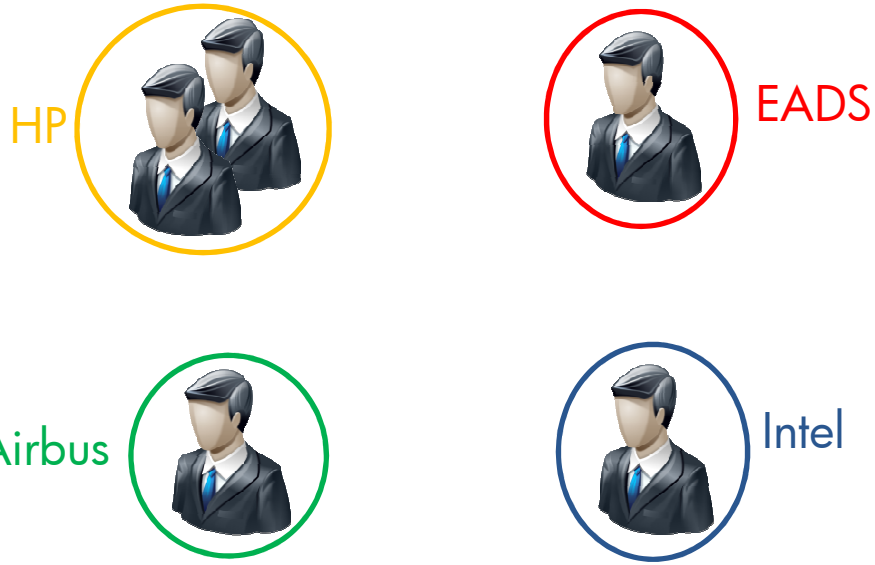
Alternative1: Use Intel Compiler plus blas/lapack library

P2. Performance problems when Intel Compiler and Blas/Lapack.

The code is much slower when running with miniblas and lapack compared the results provided by the MKL binary when it works.

# Actipole – People Involved and time frame

People involved:




## Time Frame

- Problem signaled: May 2010
- Workaround proposed and issue closed: August 2010 ( delayed due to holidays of different people involved).

# Actipole– Resolution approaches.

- Two approaches have been followed in parallel

*P1. Intel with HP* – Philippe ran tests using MKL to tackle the convergence issue.

**Environment**
- Intel Compiler 11.1.059
- FULL MKL.
- MUMPS Support
- vectorization workaround
- Flags: -O3 -axSSE4.2, SSE4.1,SSSE3, SSE3,SSE2

Tests with Platform MPI 7.1 and Intel MPI

2. HP - Eric ran tests to tackle the performance issue when using blas/lapack.

**Environment**
- Intel Compiler 11.1.059
- Blas/lapack
- MUMPS Support
- vectorization workaround
- Flags: -O3 -axSSE4.2, SSE4.1, SSSE3, SSE3,SSE2

Tests with HPMPI 2.3, PLatform MPI 7.1, 8.0

# Actipole - Solution

- HP ( Eric) managed to find the correct configuration ( Compiler/libraries/flags/MPI ) to solve the performance issue when using blas/lapack.
- We discovered that changing the MPI flavour solves the convergence(MKL) /performance issue (blas/lapack). See sample below.

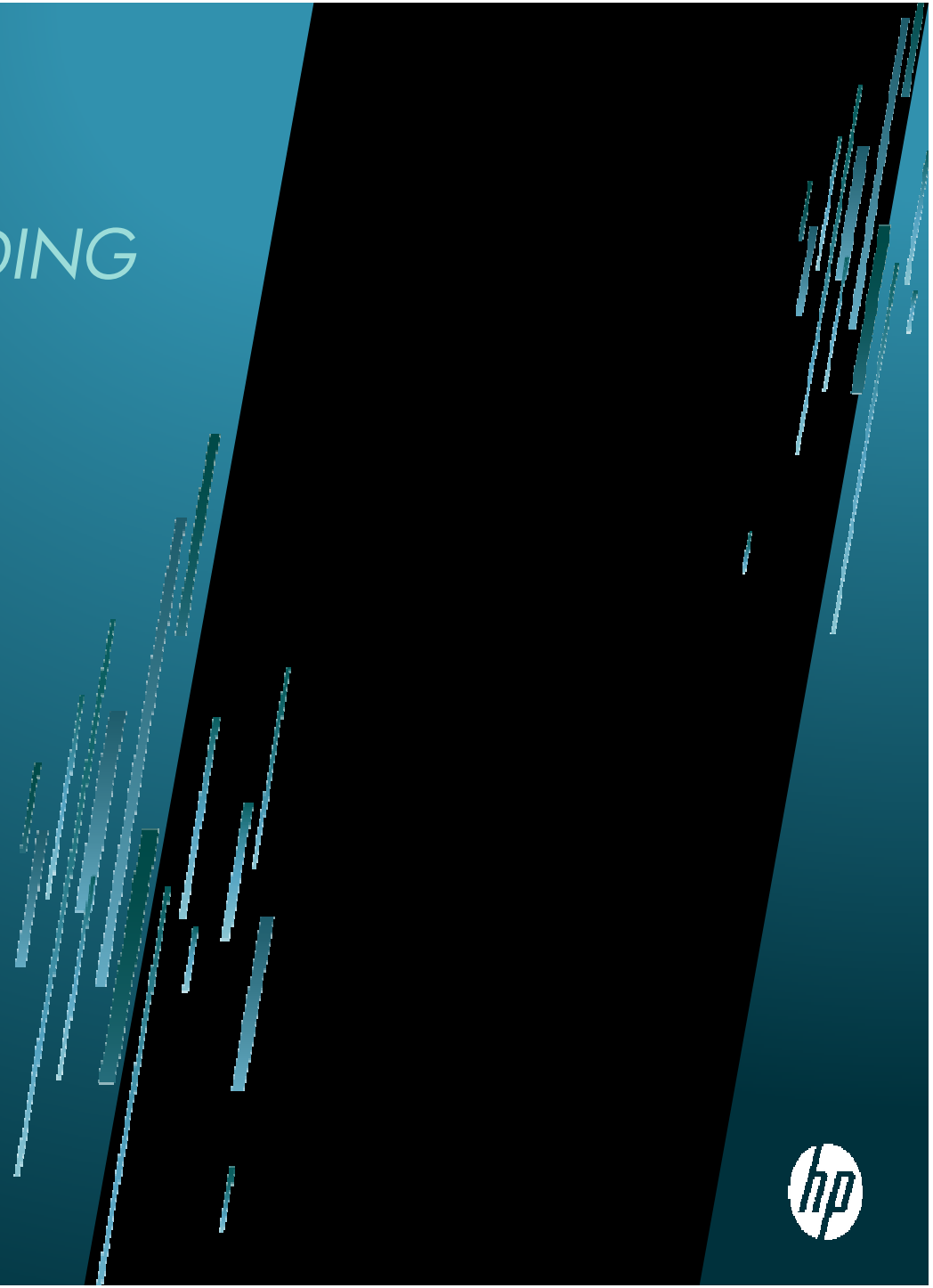
| Binary\cores | 100 | 104 | 128 | 256 | 400 | 512 |
|---|---|---|---|---|---|---|
| | | | | | | |
| HP_IntelMPI | N/R | N/R | 4206 | 4194 | N/R | 6031 |
| HP_IntelMPI_Seq | N/R | N/R | 4010 | 5796 | 4481 | 6157 |
| HP_PMPI_8.0 (India) | N/R | 5567 | N/R | 5667 | N/R | N/R |
| **HP_PMPI_8.0 (Hamburg)** | **5010** | **4567** | **4127** | **3707** | **3567** | **3172** |
| HP_PMPI_8.0_Seq | N/R | N/R | N/R | **3662** | 4668 | 3230 |
| HP_PMPI_7.1_Seq | 10078 | N/R | 3832 | 3595 | 3614 | 6788 |

- W

# Actipole – Lessons Learnt

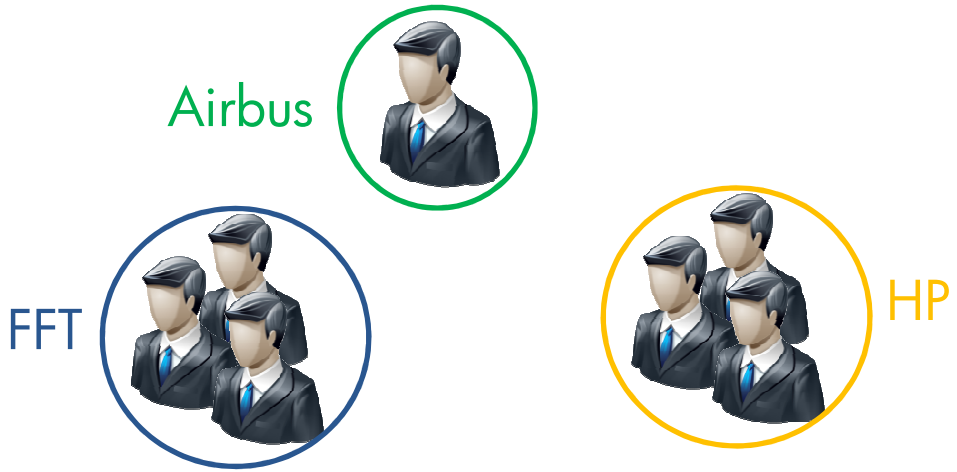
1. Problem gets solved faster if singled to the right people
2. Always involve the users from different groups
3. Ask for as many brainstorming sessions as needed to try to find as many paths possible.
4. Always escalate to the ISV
5. Patience ☺

# BINDING AND THREADING THE ACTRAN USECASE



# Actran

- People Involved




- Usecase
  - Actran 11.2 / TM_3D/ USECASE running on 50/25 nodes each processes requesting 35000MB
- Decision
  - Test using different amount of threads per process
  - Test using different binding techniques.

# Actran - Threads

- Tests were done on POD2 and POD3
- Two types of options submitted using  openMPI mpirun
  - Conventional
    - -display-map --mca mpi_warn_on_fork 0 --mca btl_openib_want_fork_support 0 --mca btl openib,sm,self
  - Extended
    - --display-map --mca mpi_warn_on_fork 0 --mca btl_openib_want_fork_support 0 --mca btl openib,sm,self --mca btl_openib_receive_queues P,32768,128,96,64 --mca btl_openib_max_send_size 32768 --mca btl_openib_eager_limit 32768 -mca btl_openib_rndv_eager_limit 32768

- btl_openib_receive_queues P,32768,128,96,64  - Amount and size of PER-PEER Receive queues

-btl_openib_max_send_size 32768  - Maximum size of a send fragment of an mpi message

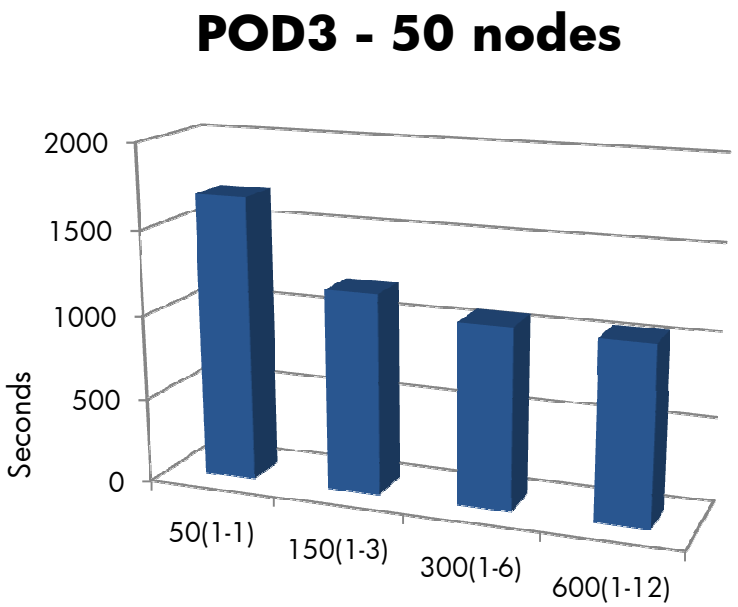
-btl_openib_eager_limit  and rndv_eager_limit 32768 – Eager limits, size of short/small messages.

# Actran – Threads POD3

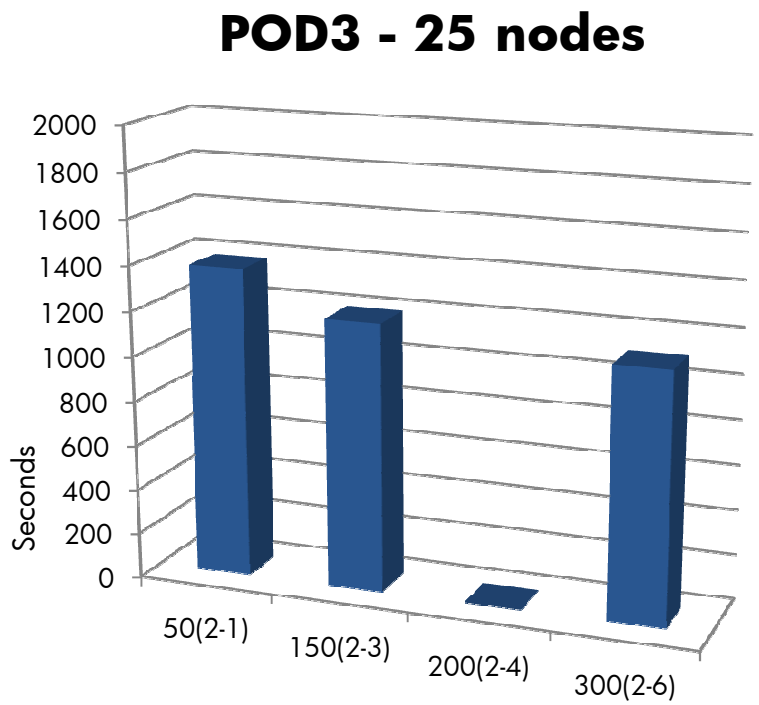
Each jobs was launched with 50 mpi processes.

| Test | MPI OPTIONS | Nodes used | Total Cores Available | MPI PROCESSES/ node | # THREADS/MPI Process | USED CORES/node | Total Threads (=Total Used Cores) | RESULTS (seconds) |
|---|---|---|---|---|---|---|---|---|
| POD3/1 | Conventional | 50 | 600 | 1 | 1 | 1 | 50 | 1683 |
| POD3/2 | Conventional | 50 | 600 | 1 | 3 | 3 | 150 | 1185 |
| POD3/3 | Conventional | 50 | 600 | 1 | 6 | 6 | 300 | 1067 |
| POD3/4 | Conventional | 50 | 600 | 1 | 12 | 12 | 600 | 1047 |
| POD3/5 | Conventional | 25 | 300 | 2 | 1 | 2 | 50 | FAILED |
| POD3/5bis | Extended | 25 | 300 | 2 | 1 | 2 | 50 | 1379 |
| POD3/6 | Conventional | 25 | 300 | 2 | 3 | 6 | 150 | FAILED |
| POD3/6 | Extended | 25 | 300 | 2 | 3 | 6 | 150 | 1192 |
| POD3/7 | Conventional | 25 | 300 | 2 | 6 | 12 | 300 | FAILED |
| POD3/7 | Extended | 25 | 300 | 2 | 6 | 12 | 300 | 1106 |

# Actran – Threads POD2 vs POD3




Cores used ( MPI Processes - Threads per Process)




Cores used ( MPI Processes - Threads per Process)

# Actran - Binding

Open MPI 1.4/1.6 supports the following binding switches to mpirun:

- --bind-to-none: Do not bind processes. **(Default)**
- --bind-to-core: Bind each MPI process to a core.
- --bind-to-socket: Bind each MPI process to a processor socket.
- -rankfile /path/to/rankfile - the user specifies a host node and *slot list* binding for each MPI process in your job

# Actran – Binding

| Test | Nodes used | MPI PROCESSES/node | Threads/MPI process | USED CORES/node | RESULTS (seconds) | Comments |
|---|---|---|---|---|---|---|
| POD3/7 | 25 | 2 | 6 | 12 | 1054 | NoBindings |
| POD3/7 | 25 | 2 | 6 | 12 | 1064 | NoBindings |
| POD3/7binding1 | 25 | 2 | 6 | 12 | 1072 | **bind-to-socket** |
| POD3/7binding1 | 25 | 2 | 6 | 12 | 1072 | **bind-to-socket** |
| POD3/7binding2 | 25 | 2 | 6 | 12 | 1053 | **cpus-per-proc 6 - -bind-to-socket** |
| POD3/7binding2 | 25 | 2 | 6 | 12 | 1073 | **cpus-per-proc 6 --bind-to-socket** |
| POD4/7rankfiles | 25 | 2 | 6 | 12 | 901 | **rankfiles** |
| POD4/7rankfiles | 25 | 2 | 6 | 12 | 918 | **rankfiles** |

Using Rankfiles drops the computations with about 15%. But further tests on other usecases are needed

# Actran – Conclusions

The tests that we have run for Actran led to the following conclusions:

1. Some of the Openmpi 1.4 and 1.5 options are not fully qualified ( by OPENMPI) therefore not stable. Are in 1.6
2. Using rankfiles on POD4 proved to provide a speed up 14-15% in the solving time. But further tests are needed and the “HP build_rankfile” script has to be correctly inserted in the LSF submit script.

*THANK YOU*