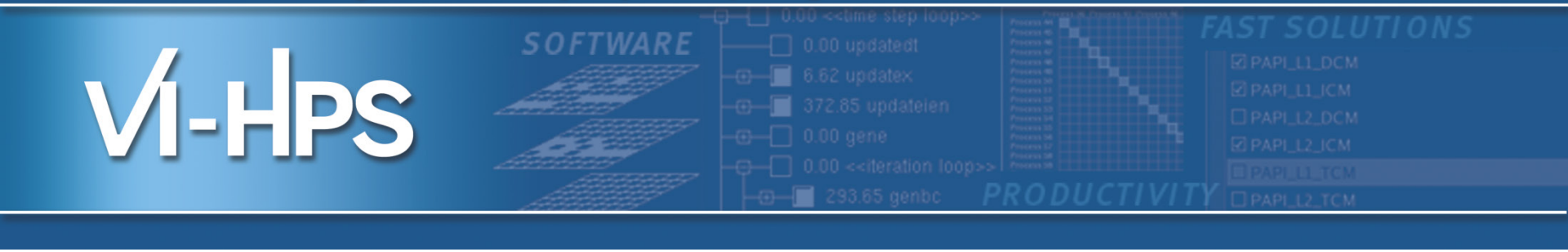

13th VI-HPS Tuning Workshop

Barcelona Supercomputing Center

10-14 February 2014

- Presenters
  - Wolfgang Frings, Marc Schlütter (JSC)
  - Robert Dietrich, Tobias Hilbrich (TUD)
  - Tim Cramer, Joachim Protze, Felix Münchhalfen (RWTH)
  - Michael Firbach, Isaías Comprés (TUM)
  - Andres Charif-Rubial, Jean-Baptiste Besnard (UVSQ)
  - Judit Giménez, Juan González, Germán Llort, Harald Servat (BSC)

- Monday, 10th February
  - 13:30 Registration
  - 14:00 Welcome
    - Introduction to VI-HPS & overview of tools
    - Introduction to parallel performance engineering
    - Introduction to parallel performance modeling
    - Parallel file I/O bottlenecks and solutions
  - 15:00 (break)
  - 15:30 Lab setup
    - MareNostrum-III hardware and software environment
    - Building and running NPB-MZ-MPI/BT-MZ & CGPOP
  - 17:30 (adjourn)

Outline of the remaining week

- Tuesday, 11th February
  - 09:00 – 13:00 **Paraver** & **Dimemas**
- Wednesday, 12th February
  - 09:00 – 10:45 **Score-P** & **CUBE**
  - 11:15 – 13:00 **Score-P** & **Scalasca**
- Thursday, 13th February
  - 09:00 – 10:45 **Advanced Score-P & Vampir**
  - 11:15 – 13:00 **Periscope**
- Friday, 14th February
  - 09:00 – 13:00 **MUST**, **MAQAO**

- Hands-on exercises part of each tool presentation every morning session.
- Hands-on coaching to apply tools to analyse & tune your own codes each afternoon to 17:30

Prepare to analyse your own application codes

- Ensure your application codes build and run to completion with appropriate datasets
  - Initial configuration should ideally run in less than 15 minutes with 1-4 compute nodes
    - to facilitate rapid turnaround and quick experimentation
  - Larger/longer scalability are also interesting
    - turnaround may be limited due to busyness of batch queues
  - Compare your application performance on other systems
    - VI-HPS tools already installed on a number of HPC systems
      - If not, ask your sysadmin to install them (or install a personal copy yourself)

Disclaimer

Tools will ***not*** automatically make you, your applications or computer systems more *productive*.

However, they can help you understand ***how*** your parallel code executes and ***when/where*** it's necessary to work on correctness and performance issues.

DON'T PANIC!

The workshop presenters are here to assist you.

Workshop accounts in MareNostrum-III

- nct00001
  - Teacher's account
- nct010[01-30]
  - Student's account

Workshop system (hardware)

System	**MareNostrum-III**
Domain	bsc.es
Login nodes	mn[1-3].bsc.es
Vendor	IBM
Network	Infiniband FDR10 fat-tree
Processors	SandyBridge-EP E5-2670
Frequency	2.60GHz (turbo up to 3.30GHz)
Compute nodes	2100 (25 racks * 84 compute nodes)
Chips per node	2
Cores per chip	8
Threads per core	1
Memory per node	32 GBytes (4GB OS / 28 GB user)

Workshop system (software)

System	**MareNostrum-III**		
Domain	bsc.es		
Filesystem	GPFS		
Parallel filesystem	/gpfs/scratch/nct00/${USER}		
Compilers	Intel compiler-suite v13.0.1	GNU compiler-suite 4.7.2	
OpenMP flag	-openmp	-fopenmp	
MPI	OpenMPI 1.5.4	Intel MPI 4.1.1	MVAPICH 1.8.1
C compiler	mpicc	mpicc	mpicc
C++ compiler	mpiCC	mpicxx	mpicxx
F77 compiler	mpif77	mpif77	mpif77
F90 compiler	mpif90	mpif90	mpif90

Available modules

- To switch between compilers
  - module load intel (default, v13.0.1)
  - module load gcc (base 4.3.4, module version 4.7.2)
- To switch between MPI implementations
  - module load openmpi (default, v1.5.4)
  - module load impi (v4.1.1)
  - module load mvapich2 (v1.8.1)
- To set gnu compilers as backend compilers for MPI
  - module load gnu
- To use the performance tools
  - module load bsctools
  - module load unite

Submitting jobs

- Job submit

  bsub < my_job.lsf

- List jobs

  bjobs [-w][-X][-l job_id]

- Job cancel

  bkill <job_id>

Example of job (MPI)

```
#!/bin/bash

#BSUB -n 48                  → # of MPI processes
#BSUB -R"span[ptile=16]"     → Span, 16 MPI processes per node
#BSUB -x                     → Exclusive use of the nodes assigned
#BSUB -oo output_%J.out      → JOB standard output
#BSUB -eo output_%J.err      → JOB standard error
#BSUB -J cgpop               → Job name
#BSUB -W 02:00               → Wall clock time (HH:MM)

mpirun --bind-to-core ./cgpop
```

Example of job (hybrid MPI + OpenMP)

```
#!/bin/bash

#BSUB -n 64                  → # of MPI processes
#BSUB -R"span[ptile=2]"      → Span, 2 MPI processes per node
#BSUB -x                     → Exclusive use of the nodes assigned
#BSUB -oo output_%J.out      → JOB standard output
#BSUB -eo output_%J.err      → JOB standard error
#BSUB -J hybrid              → Job name
#BSUB -W 02:00               → Wall clock time (HH:MM)

# 4 MPI processes per node / 16 cpus available (4 threads per MPI process):
export OMP_NUM_THREADS=8
mpirun --bind-to-core --npersocket 1 --cpus-per-proc 8 ./bt-mz.B.64
```

Allocation summary

- Full nodes

  `--bind-to-core`

- Not-full nodes

  `--bind-to-core --npersocket X`

  `(where X balances the processes among the two sockets)`

- Hybrid (either full or not)

  `--bind-to-core --npersocket X --cpus-per-proc 8`

  `(where X balances the processes among the two sockets)`

Additional logistics (for your stomach)