



## Hands-on exercise: NPB-MZ-MPI / BT

VI-HPS Team

- Familiarise with usage of VI-HPS tools
  - complementary tools' capabilities & interoperability
- Prepare to apply tools productively to *your* applications(s)
- Exercise is based on a small portable benchmark code
  - unlikely to have significant optimisation opportunities
- Optional (recommended) exercise extensions
  - analyse performance of alternative configurations
  - investigate effectiveness of system-specific compiler/MPI optimisations and/or placement/binding/affinity capabilities
  - investigate scalability and analyse scalability limiters
  - compare performance on different HPC platforms
  - ...

- VI-HPS tools accessible through the UNITE framework
  - Currently installed in a non-default location
    - Need to extend the module search path manually
  - Load the UNITE meta-module

```
% module use /gpfslocal/pub/vihps/UNITE/local
% module load UNITE
% module avail
----- /gpfslocal/pub/vihps/UNITE/modulefiles/tools -----
[...]
scalasca/2.0b3-intel13-intelmpi (default)
scorep/1.1.1-intel13-intel-papi-cuda-static
scorep/1.1.1-intel13-intelmpi-papi-static (default)
tau/2.22.2-intelmpi-intel13
vampir/8.1-dev
[...]
maqao/2.0.0
```

- VI-HPS-TW meta-module loads default set of VI-HPS tools (and corresponding compiler+MPI) for workshop

```
% module load VI-HPS-TW
VI-HPS-TW loaded
(load) intel version 13.0.1
(load) intelmpi version 4.0.3
scorep/1.1.1-intel13-intelmpi-papi-static loaded
cube4/4.1.6-gnu loaded
scalasca/2.0b3-intel13-intelmpi loaded
tau/2.22.2-intelmpi-intel13 loaded
vampir/8.1-dev loaded
% module list
1) UNITE/1.1
2) intel/13.0.1
3) intelmpi/4.0.3
4) cube4/4.1.6-gnu
5) scorep/1.1.1-intel13-intelmpi-papi-static
6) scalasca/2.0b3-intel13-intelmpi
7) tau/2.22.2-intelmpi-intel13
8) vampir/8.1-dev
9) VI-HPS-TW/intel13
```

- Module help provides location of tutorial sources

```
% module help VI-HPS-TW
[...]
VI-HPS-TW: VI-HPS Tuning Workshop environment
for intel/13.0.1 compilers and intelmpi/4.0.3 MPI

VI-HPS tutorial sources: $UNITE_ROOT/tutorial
VI-HPS sample experiments: $UNITE_ROOT/samples
[...]
```

- Copy tutorial sources to your working directory

```
% cp -r $UNITE_ROOT/tutorial .
% cd tutorial/NPB3.3-MZ-MPI
```

- (When available, generally advisable to use a parallel filesystem such as \$WORK)

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - Available from  
<http://www.nas.nasa.gov/Software/NPB>
  - 3 benchmarks in Fortran77
  - Configurable for various sizes & classes
- Move into the NPB3.3-MZ-MPI root directory

```
% ls
bin/    common/   jobsript/   Makefile  README.install  SP-MZ/
BT-MZ/   config/   LU-MZ/     README    README.tutorial sys/
```

- Subdirectories contain source code for each benchmark
  - plus additional configuration and common code
- The provided distribution has already been configured for the tutorial, such that it's ready to "make" one or more of the benchmarks and install them into a (tool-specific) "bin" subdirectory

- Type “make” for instructions

```
% make  
=====  
= NAS PARALLEL BENCHMARKS 3.3 =  
= MPI+OpenMP Multi-Zone Versions =  
= F77 =  
=====
```

To make a NAS multi-zone benchmark type

```
make <benchmark-name> CLASS=<class> NPROCS=<nprocs>
```

where <benchmark-name> is “bt-mz”, “lu-mz”, or “sp-mz”  
<class> is “S”, “W”, “A” through “F”  
<nprocs> is number of processes

[ . . . ]

```
*****  
* Custom build configuration is specified in config/make.def *  
* Suggested tutorial exercise configuration for HPC systems: *  
* make bt-mz CLASS=B NPROCS=4 *  
*****
```

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the number of MPI processes: **NPROCS=4**
  - the benchmark class (S, W, A, B, C, D, E): **CLASS=B**

```
% make bt-mz CLASS=B NPROCS=4
cd BT-MZ; make CLASS=B NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c
../sys/setparams bt-mz 4 B
mpiifort -c -O3 -openmp bt.f
[...]
cd ../common; mpiifort -c -O3 -openmp timers.f
mpiifort -O3 -openmp -o ./bin/bt-mz\_B.4 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
./common/print\_results.o ./common/timers.o
Built executable ./bin/bt-mz\_B.4
make: Leaving directory 'BT-MZ'
```

- What does it do?
  - Solves a discretized version of unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Performs 200 time-steps on a regular 3-dimensional grid
- Implemented in 20 or so Fortran77 source modules
- Uses MPI & OpenMP in combination
  - 4 processes with 4 threads each should be reasonable
  - bt-mz\_B.4 should run in around 13 seconds
  - bt-mz\_C.4 should take around 3-4x longer

- Copy jobscript and launch as a hybrid MPI+OpenMP application

```
% cd bin  
% cp ..../jobscript/mzs/run.ll .  
% less run.ll  
% llsubmit run.ll  
% cat NPB_B.<id>  
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark  
Number of zones: 8 x 8  
Iterations: 200 dt: 0.000300  
Number of active processes: 4  
Total number of threads: 16 ( 4.0 threads/process)  
  
Time step 1  
Time step 20  
[...]  
Time step 180  
Time step 200  
Verification Successful  
  
BT-MZ Benchmark Completed.  
Time in seconds = 12.74
```

Hint: save the benchmark output (or note the run time) to be able to refer to it later