Hands-on: BullX DLC *Froggy*
NPB-MZ-MPI / BT

VI-HPS Team

Tutorial exercise objectives

- 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
  - …

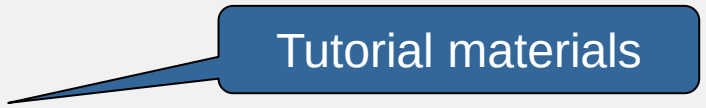
Access to Froggy

```
# Connect to the CIMENT access host
% ssh –X -C login@access-rr-ciment.imag.fr
# Then, from the CIMENT access host, go to froggy:
% ssh –X –C froggy.ujf-grenoble.fr
```

- Logging on to Froggy

/home/$USER
/scratch/$USER

/home/PROJECTS/pr-vi-hps-tw18/
/scratch/PROJECTS/pr-vi-hps-tw18/



- File systems
  - Use scratch for the tutorial

- More extensive Documentation:
  - https://ciment.ujf-grenoble.fr/wiki/index.php/Froggy_quickstart

Compiling

- Loading development modules with the default Intel compiler with Intel MPI

```
% source /applis/site/env.bash
% module load intel-devel
```

  - mpiicc (mpicc –cc=icc)
  - mpiicpc (mpicc –cxx=icpc)
  - mpiifort (mpif77 -f77=ifort), etc.

- For alternative GCC compiler with OpenMPI

```
% source /applis/site/env.bash
% module load gnu-devel
```

  - mpicc, mpicxx, mpif77, etc

Job submission and start

```
% chmod ugo+x ./jobscript.oar
% oarsub –S ./jobscript.oar
```

- Submit job with oarsub
  - Has to be an executable script

```
#OAR -l /nodes=2/core=16,walltime=0:15:00
#OAR -t devel
#OAR --project vi-hps-tw18
#OAR --stdout run_%jobid%
#OAR --stderr run_%jobid%

. /applis/site/env.bash
module load intel-devel

export OMP_NUM_THREADS=4
export I_MPI_FABRICS=shm:dapl
export I_MPI_PERHOST=`echo "16/$OMP_NUM_THREADS" | bc`
export I_MPI_HYDRA_BOOTSTRAP_EXEC=oarsh
export I_MPI_HYDRA_HOST_FILE=$OAR_NODE_FILE

mpiexec.hydra -genvall -n 8 ./bt-mz_C.8
```

Project id

Environment, choice of mpi

Mpiexec.hydra with 8x4

- Minimal hybrid job script for Intel MPI

```
% oarstat -u $USER
% oardel <jobid>
```

- View job queue
- Cancel job

```
% oarsub -I -l /nodes=4/core=16 --project vi-hps-tw18
```

- Interactive session

Local Installation (*Froggy* BullX DLC)

- Setup preferred program environment compilers
  - Default set Intel Compilers with Intel MPI
  - GCC+OpenMPI and Intel + BullxMPI also available

```
% source /applis/site/env.bash
% module load intel-devel
% module use /home/PROJECTS/pr-vi-hps-tw18/opt/mf
```

- Copy tutorial sources to your working directory, ideally on a parallel file system (scratch: /scratch/$USER)

```
% cd /scratch/$USER
% tar zxvf /home/PROJECTS/pr-vi-hps-tw18/tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

NPB-MZ-MPI Suite

- 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/  jobscript/  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 is ready to “make” one or more of the benchmarks and install them into a (tool-specific) “bin” subdirectory

Building an NPB-MZ-MPI Benchmark

```
% 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=C NPROCS=8                           *
 ***********************************************************
```

- Type “make” for instructions

Building an NPB-MZ-MPI Benchmark

```
% make bt-mz CLASS=C NPROCS=8
make[1]: Entering directory `BT-MZ'
make[2]: Entering directory `sys'
cc  -o setparams setparams.c -lm
make[2]: Leaving directory `sys'
../sys/setparams bt-mz 8 C
make[2]: Entering directory `../BT-MZ'
mpif77 -f77=ifort  -c  -O3 -openmp       bt.f
                                  […]
mpif77 -f77=ifort  -c  -O3 -openmp       mpi_setup.f
cd ../common;  mpif77 -f77=ifort  -c  -O3 -openmp        print_results.f
cd ../common;  mpif77 -f77=ifort  -c  -O3 -openmp        timers.f
mpif77 -f77=ifort  -O3 -openmp   -o ../bin/bt-mz_B.8 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
make[2]: Leaving directory `BT-MZ'
Built executable ../bin/bt-mz_C.8
make[1]: Leaving directory `BT-MZ'
```

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the number of MPI processes: NPROCS=**8**
  - the benchmark class (S, W, A, B, C, D, E): CLASS=**C**

Shortcut: % **make suite**

NPB-MZ-MPI / BT (Block Tridiagonal Solver)

- What does it do?
  - Solves a discretized version of the 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
  - 8 processes each with 8 threads should be reasonable for 4 compute nodes of Froggy
  - bt-mz_B.8 should run in around 7-8 seconds
  - bt-mz_C.8 should take around 30 seconds

NPB-MZ-MPI / BT Reference Execution

```
% cd bin
% cp ../jobscript/froggy/run.oar .
% less run.oar
% oarsub –S ./run.oar
% cat run_<job_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:     8
 Total number of threads:       32  (  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 = 28.78
```

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

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

Tutorial Exercise Steps

- Edit config/make.def to adjust build configuration
  - Modify specification of compiler/linker: MPIF77
- Make clean and build new tool-specific executable

```
% make clean
% make bt-mz CLASS=C NPROCS=8
Built executable ../bin.$(TOOL)/bt-mz_C.8
```

- Change to the directory containing the new executable before running it with the desired tool configuration

```
% cd bin.$(TOOL)
% cp ../jobscript/froggy/$(TOOL).oar  .
% oarsub –S ./$(TOOL).oar
```

NPB-MZ-MPI / BT: config/make.def

```
#                SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.
#
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
# Configured for generic MPI with INTEL compiler
#---------------------------------------------------------------------------
#OPENMP  = -fopenmp      # GCC compiler
OPENMP = -openmp         # Intel compiler

...
#---------------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------------
MPIF77 = mpiifort # Intel compiler

# Alternative variant to perform instrumentation
#MPIF77 = scorep --user  mpiifort

# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP) mpiifort
...
```

Default (no instrumentation)

Hint: uncomment a compiler wrapper to do instrumentation