Hands-on exercise (JUROPA): 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
  - …

Local Installation (juropa)

- Connect to JUROPA using trusted X11 forwarding

```
% ssh –YC juropa.fz-juelich.de
```

- Check/modify modules for MPI & compilers

```
% module list
Currently loaded modules:
1)parastation/mpi2-intel-5.0.026-1
2)intel/11.1.072
[…]
```

- Copy tutorial sources to your work directory

```
% cp -r ~train001/tutorial $WORK
% cd $WORK/tutorial/NPB3.3-MZ-MPI
```

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

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

```
% $WORK/tutorial/NPB3.3-MZ-MPI; 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's ready to “make” one or more of the benchmarks and install them into a (tool-specific) “bin” subdirectory

Building an NPB-MZ-MPI Benchmark

- 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                           *
 ***************************************************************
```

Building an NPB-MZ-MPI Benchmark

- 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
mpif77 -c  -O3 -openmp bt.f
 [...]
cd ../common;  mpif77 -c  -O3 -openmp timers.f
mpif77 –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'
```

Hint: for default configuration:
```
% make suite
```

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

- 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 for execution on a single compute node
  - bt-mz_B.4 should run in around 30 seconds
  - bt-mz_C.4 should take around 3-4x longer

NPB-MZ-MPI / BT Reference Execution

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

```
% cd bin
% cp ../jobscript/juropa/run.msub .
% less run.msub
% msub run.msub
% cat mzmpibt.<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 = 28.74
```

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

Local UNITE installation (juropa)

- VI-HPS tools accessible through the UNITE framework
  - Load the UNITE meta-module to see available tools
  - May have multiple versions and configurations

```
% module load UNITE
% module avail
----- /usr/local/UNITE/modulefiles/tools -----
 [...]
periscope/1.5
scalasca/1.4.3-parastation-intel-papi-sion(default)
scalasca/2.1-parastation-gnu
scalasca/2.1-parastation-intel
scorep/1.2.1-parastation-gnu-papi
scorep/1.2.1-parastation-intel-papi(default)
tau/2.22.2-parastation-intel-papi
vampir/8.1.0(default)
vampirserver/8.1.0(default)
  [...]
```

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=B NPROCS=4
Built executable ../bin.$(TOOL)/bt-mz_B.4
```

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

```
% cd bin.$(TOOL)
% export …
% OMP_NUM_THREADS=4  mpiexec –np 4 ./bt-mz_B.4
```

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

```
#             SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#---------------------------------------------------------------------
# Items in this file may need to be changed for each platform.
#---------------------------------------------------------------------
...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
MPIF77 = mpif77

# Alternative variants to perform instrumentation
#MPIF77 = psc_instrument -u user,mpi,omp –s ${PROGRAM}.sir mpif77
#MPIF77 = tau_f90.sh
#MPIF77 = scalasca -instrument mpif77
#MPIF77 = vtf77 –vt:hyb -vt:f77 mpif77
#MPIF77 = scorep --user mpif77

# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP) mpif77

# This links MPI Fortran programs; usually the same as ${MPIF77}
FLINK   = $(MPIF77)
...
```

Default (no instrumentation)

Hint: uncomment one of these alternative compiler wrappers to perform instrumentation

… or use this generic variant