Hands-on: Cray XC30 *Archer* 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 (Archer Cray XC30)

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

```
% module use /work/y14/shared/modules
% module avail
------- /work/y14/shared/modules/ --------
cube/4.2.2(default)                    scalasca/2.1(default)
scorep/1.3-trunk(default)              vampir/8.2.1(default)
[...]
```

  - Copy tutorial sources to your $WORK directory

```
% cd $WORK
% cp -r /work/y14/shared/tutorial/NPB3.3-MZ-MPI .
% cd NPB3.3-MZ-MPI
```

- 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'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=C NPROCS=8                           *
 ***************************************************************
```

Building an NPB-MZ-MPI Benchmark

- 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**

```
% make bt-mz CLASS=C NPROCS=8
cd BT-MZ; make CLASS=C NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c
../sys/setparams bt-mz 8 C
ftn -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  ftn -c  -O3 -fopenmp timers.f
ftn –O3 -fopenmp -o ../bin/bt-mz_C.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
Built executable ../bin/bt-mz_C.8
make: Leaving directory 'BT-MZ'
```

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
  - 8 processes with 6 threads each should be reasonable for 2 compute nodes
  - bt-mz_C.8 should run in around 20 seconds
  - bt-mz_D.8 should take around 16x longer

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

```
% cd bin
% cp ../jobscripts/archer/run.pbs .
% less run.pbs
% qsub run.pbs
% cat <job_id>.out
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
 Number of zones:   8 x   8
 Iterations:  200   dt:   0.000100
 Number of active processes:     8
 Total number of threads:       48  (  6.0 threads/process)

 Time step    1
 Time step   20
  [...]
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 18.02
```

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.$(TOOL)
% cp ../jobscripts/crayxt/$(TOOL).pbs
% qsub $(TOOL).pbs
```

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 = ftn

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

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

# 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