NPB3.3-MZ-MPI/BT tutorial example OpenMP+MPI application (BlueGene version)

Brian Wylie
Jülich Supercomputing Centre
b.wylie@fz-juelich.de

May 2012

Tutorial exercise objectives

- Familiarise with usage of VI-HPS tools
  - complementary tools' capabilities & interoperatibilty
- Prepare to apply tools productively to *your* application(s)
- Exercise is based on a small portable benchmark code
  - unlikely to have significant optimization opportunities

- Optional (recommended) exercise extensions
  - analyze performance of alternative configurations
  - investigate effectiveness of system-specific compiler/MPI optimizations and/or placement/binding/affinity capabilities
  - investigate scalability and analyze scalability limiters
  - compare performance on different HPC platforms
  - ...

Local installation (ICM 'notos')

- Connect to BG/P via gateway using trusted X11 forwarding

```
% ssh -Y atol.icm.edu.pl
delta% ssh -Y notos
```

- Ensure module for MPI compiler wrappers loaded

```
% module list
% module load mpi_default
```

- Tutorial sources should be copied to your own directory where you can then work on them

```
% cp -r /opt/prace/VI-HPS/tutorial/NPB3.3-MZ-MPI  $WRKDIR
```

(Additional tutorial exercises can also be provided.)

Local Scalasca installation (ICM 'notos')

- Load the latest Scalasca module

```
% module avail scalasca
scalasca/1.3.3                  scalasca/1.4.1
% module load scalasca/1.3.3
```

- or manually add Scalasca and its CUBE GUI to your PATH

```
% export PATH=/opt/prace/cube-3.4.1/bin:$PATH
% export PATH=/opt/prace/scalasca-1.4.2/bin:$PATH
```

- Scalasca installation including interactive explorer GUI
  - configured for default MPI and IBM XL compilers
    - currently the only supported configuration for BlueGene systems
  - additional configurations would need to be installed for other MPI libraries and compiler suites
- Sample experiments provided for examination

```
% ls /opt/prace/VI-HPS/samples/scalasca
README              epik_bt_B_dual64_sum/      epik_bt_B_dual64_trace/
                    epik_bt-mz_B_smp32x4_sum/  epik_bt-mz_smp32x4_trace/
```

NPB-MZ-MPI suite

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

```
% cd NPB3.3-MZ-MPI; ls
BT-MZ/  LU-MZ/  SP-MZ/
bin/    common/ config/ jobscript/      Makefile        README  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” 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  =
  ========================================

  To make a NAS multi-zone benchmark type

        make <benchmark-name> CLASS=<class> NPROCS=<number>

  To make a set of benchmarks, create the file config/suite.def
  according to the instructions in config/suite.def.template and type

        make suite

 ************************************************************
 * Custom build configuration is specified in config/make.def  *
 * Suggested tutorial benchmark specification:                 *
 *       make bt-mz CLASS=B NPROCS=32                          *
 ************************************************************
```

Building the NPB-MZ-MPI/BT benchmark

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the number of MPI processes: NPROCS=**32**
  - the benchmark class (S, W, A, B, C, D, E, F): CLASS=**B**

```
% make bt-mz CLASS=B NPROCS=32
cd BT-MZ; make CLASS=B NPROCS=32 VERSION=
gmake: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 32 B
mpixlf77_r -c  -O -qsmp=omp  bt.f
...
mpiflx77_r -c  -O -qsmp=omp  setup_mpi.f
cd ../common;  mpixlf77_r -c  -O -qsmp=omp  print_results.f
cd ../common;  mpixlf77_r -c  -O -qsmp=omp  timers.f
mpixlf77_r -O -qsmp=omp  -o ../bin/bt-mz_B.32 \
    bt.o make_set.o initialize.o exact_solution.o exact_rhs.o \
    set_constants.o adi.o define.o copy_faces.o rhs.o solve_subs.o \
    x_solve.o y_solve.o z_solve.o add.o error.o verify.o setup_mpi.o \
    ../common/print_results.o ../common/timers.o
Built executable ../bin/bt-mz_B.32
gmake: 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 using ADI and verifies solution error within acceptable limit
  - Intra-zone computation with OpenMP, inter-zone with MPI
- Implemented in 20 or so Fortran77 source modules

- Runs with any number of MPI processes & OpenMP threads
  - bt-mz_B.32 x4 is reasonable for a BlueGene compute node
    - excess processes idle when run with more than compiled number
  - bt-mz_B.32 x4 should run in around 30 seconds
    - typically runs more efficiently with more processes than threads
  - CLASS=C does much more work and takes much longer!

NPB-MZ-MPI/BT reference execution

- Launch as an MPI application with OMP_NUM_THREADS set

```
% cd bin;  mpirun -np 32 -env OMP_NUM_THREADS=4  ./bt-mz_B.32
 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:     32

 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 28.86
```

Hint: copy/edit example batch scripts from jobscript directory:
% llsubmit ../jobscript/run.ll

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

Tutorial exercise steps

- The tutorial steps are similar and repeated for each tool
- Use the provided NPB3.3-MZ-MPI tutorial directory

```
% cd NPB3.3-MZ-MPI; ls
BT-MZ/  LU-MZ/  SP-MZ/
bin/    common/ config/ jobscript/      Makefile        README  sys/
```

- 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=32
Built executable ../bin.$(TOOL)/bt-mz_B.32
```

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

```
% cd bin.$(TOOL)
% llsubmit ../jobscript/run.ll
# mpirun -mode smp -np 32 -env OMP_NUM_THREADS=4 ./bt-mz_B.32
```

NPB-MZ-MPI/BT build configuration definition

- config/make.def

```
#            SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#---------------------------------------------------------------------
# Items in this file may need to be changed for each platform.
…
OPENMP = -qsmp=omp  # IBM XL
#---------------------------------------------------------------------
# The Fortran compiler used for hybrid MPI programs
#---------------------------------------------------------------------
MPIF77 = mpixlf77_r

# Alternative variants to perform instrumentation
#MPIF77 = psc_instrument -t user,mpi  mpixlf77_
#MPIF77 = scalasca -instrument  mpixlf77_r
#MPIF77 = tau_f90.sh
#MPIF77 = bgvtf77 -vt:hyb -vt:f77  mpixlf77_r

# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP)  mpixlf77_r

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

Adjust OMP flag if necessary

Default (no instrumentation)

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

... or this for generic variant

BlueGene peculiarities

- Our tutorials are most often done with the Linux Live ISO so they use generic names
  - MPI compiler: `mpicc, mpicxx, mpif77, mpif90`
  - MPI launcher: `mpiexec -np …`
  - OpenMP flag: `-fopenmp`
- If your system is different, then you need to adapt compiler/launch commands accordingly
- BlueGene systems are certainly different
  - MPI compiler: `mpixlc_r, mpixlcxx_r, mpixlf77_r`
  - MPI launcher: `mpirun -np …`
  - OpenMP flag: `-qsmp=omp`