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

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

August 2012

Tutorial exercise objectives

- Familiarise with usage of VI-HPS tools
  - complementary tools' capabilities & interoperability
- 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 (CSCS 'todi')

- Connect to XK6 via gateway using X11 forwarding

```
% ssh -X ela.cscs.ch
ela% ssh -X todi
```

- Ensure module for desired PrgEnv is loaded

```
% module list
% module swap PrgEnv-cray PrgEnv-gnu
```

- For interactive execution, allocate a compute partition

```
% salloc --res=sschool -N 4
```

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

```
% cp -r /project/test/scalasca/VI-HPS/tutorial/NPB3.3-MZ-MPI  $SCRATCH
```

(Additional tutorial exercises can also be provided.)

Local Scalasca installation (CSCS)

- Load your desired PrgEnv, then the Scalasca module

```
% module swap PrgEnv-cray PrgEnv-gnu
% module avail scalasca
scalasca/1.3.3  scalasca/1.4   scalasca/1.4.1(default)  scalasca/1.4.2
% module load scalasca/1.4.2
```

- Distinct Scalasca modules for each Cray XT PrgEnv
  - Cray/CCE, GNU/GCC, Intel, Pathscale, PGI compilers
  - Scalasca module loaded based on current PrgEnv
  - Scalasca instrumentation by compiler is compiler-specific
    - measurement collection, analysis & examination shouldn't be
- Sample experiments provided for examination

```
% ls /project/test/scalasca/VI-HPS/samples/scalasca
README      epik_bt-mz_B_16x4_sum.nofilt/       epik_bt-mz_16x4_sum/
            epik_bt-mz_B_16x4_sum_PAPI_FP_OPS/  epik_bt-mz_16x4_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=16                          *
 ************************************************************
```

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=**16**
  - the benchmark class (S, W, A, B, C, D, E, F): CLASS=**B**

```
% make bt-mz CLASS=B NPROCS=16
cd BT-MZ; make CLASS=B NPROCS=16 VERSION=
gmake: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 16 B
ftn -c  -O3 -fopenmp  bt.f
...
ftn -c  -O3 -fopenmp  setup_mpi.f
cd ../common;  ftn -c  -O3 -fopenmp  print_results.f
cd ../common;  ftn -c  -O3 -fopenmp  timers.f
ftn -O3 -fopenmp  -o ../bin/bt-mz_B.16 \
    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.16
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.16 x4 is reasonable, runs in around 10 seconds
    - excess processes idle when run with more than compiled
    - typically runs more efficiently with more processes than threads
  - CLASS=C does much more work and takes much longer!
  - Disable thread load balancing with envvar NPB_MZ_BLOAD=0

NPB-MZ-MPI/BT reference execution

- Set OMP_NUM_THREADS and launch as an MPI application

```
% cd bin;  OMP_NUM_THREADS=4  aprun -n 16 -d 4 ./bt-mz_B.16
 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

 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 = 8.86
```

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

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=16
Built executable ../bin.$(TOOL)/bt-mz_B.16
```

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

```
% cd bin.$(TOOL)
% export OMP_NUM_THREADS=4 ...
% aprun -n 16 -d 4 ./bt-mz_B.16     # mpiexec -np 16 ./bt-mz_B.16
```

NPB-MZ-MPI 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 = -fopenmp  # PrgEnv-gnu
#---------------------------------------------------------------------
# The Fortran compiler used for hybrid MPI programs
#---------------------------------------------------------------------
MPIF77 = ftn  # mpif77

# Alternative variants to perform instrumentation
#MPIF77 = psc_instrument ftn         # mpif77
#MPIF77 = scalasca -instrument ftn   # mpif77
#MPIF77 = tau_f77.sh -tau_makefile=...
#MPIF77 = vtf77 -vt:hyb -vt:f77 ftn  # mpif77
#MPIF77 = scorep --user  ftn         # mpif77

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

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

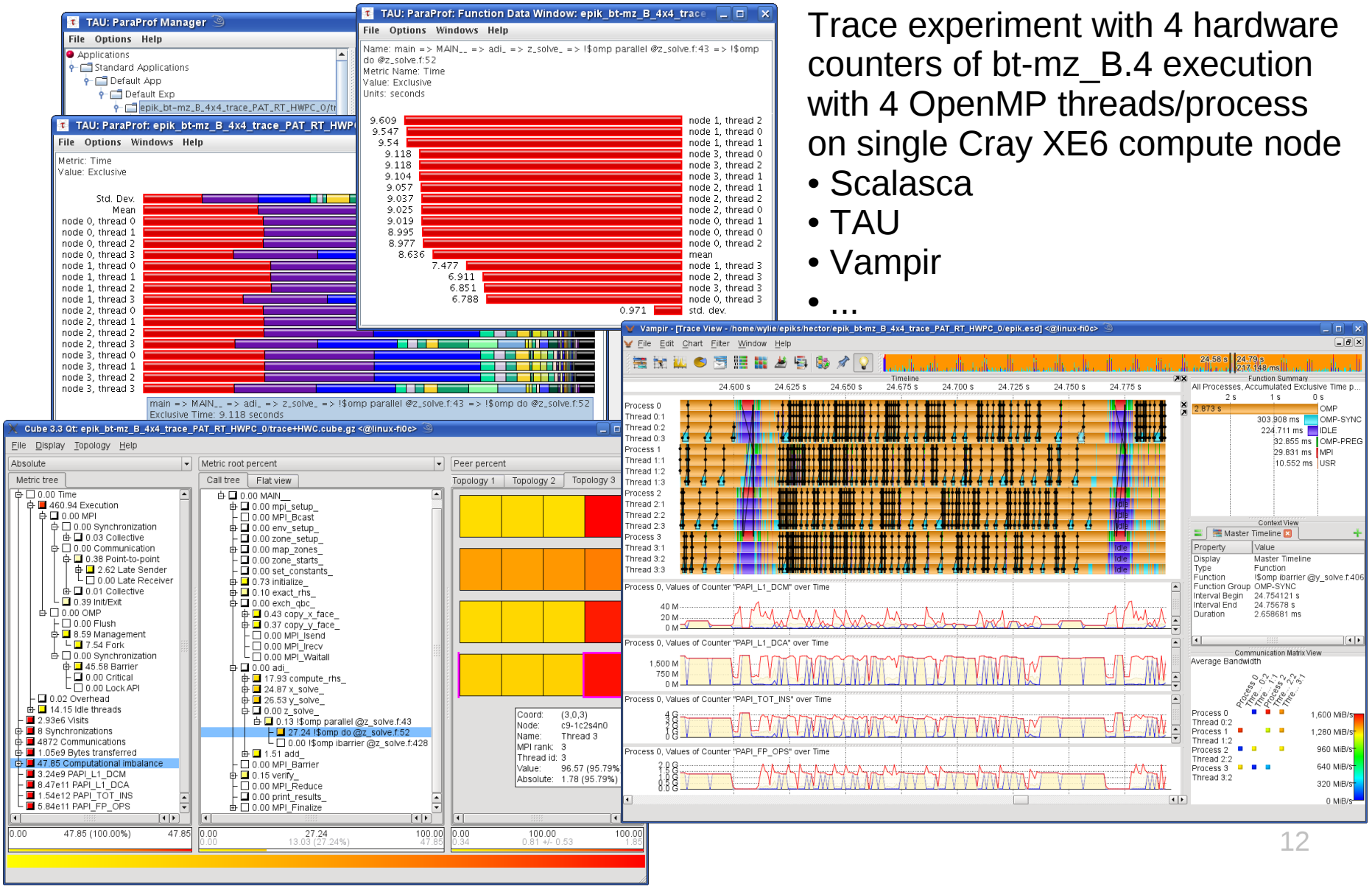
Set flag according to PrgEnv

Default (no instrumentation)

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

... or this for generic variant

NPB-MZ-MPI/BT analyses with VI-HPS tools



Trace experiment with 4 hardware counters of bt-mz_B.4 execution with 4 OpenMP threads/process on single Cray XE6 compute node

- Scalasca
- TAU
- Vampir
- …

Cray 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
- Cray systems are certainly different
  - MPI compiler: `cc, CC, ftn`
  - MPI launcher: `aprun -n …`
  - OpenMP flag: `-fopenmp / -homp / -openmp / -mp` (for GNU, CCE, Intel & PGI/Pathscale, respectively)