

International HPC Summer School 2018: Performance analysis and optimization

Hands-on:
NPB-MZ-MPI/BT Reference run

VI-HPS Team
Ilya Zhukov – Jülich Supercomputing Centre

Performance analysis steps

- 0.0 Reference preparation for validation

- 1.0 Program instrumentation
 - 1.1 Summary measurement collection
 - 1.2 Summary analysis report examination

- 2.0 Summary experiment scoring
 - 2.1 Summary measurement collection with filtering

- 3.0 Trace measurement with filtering
 - 3.1 Event trace examination & analysis

Performance analysis steps

- 0.0 Reference preparation for validation
- 1.0 Program instrumentation
 - 1.1 Summary measurement collection
 - 1.2 Summary analysis report examination
- 2.0 Summary experiment scoring
 - 2.1 Summary measurement collection with filtering
- 3.0 Trace measurement with filtering
 - 3.1 Event trace examination & analysis

Tutorial exercise objectives

- Familiarize with a typical **workflow**
- 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
 - Investigate reproducibility of measurements: CPU frequency/Turboboost
 - 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
 - ...

Access to Bridges

```
# Connect to the Bridges login node
% ssh -p 2222 userid@bridges.psc.edu
```

```
$HOME
/pylon5
...

/home/zhukov/ihpcss18/tutorial
```



Tutorial materials

- Logging in to Bridges

- File systems & directories

- Use \$HOME for the tutorial
 - Even if /pylon5 is faster
 - But no excessive I/O demand
 - Backed up daily

- More extensive documentation:

- <https://www.psc.edu/bridges/user-guide>

Compiling & job submission

- Development environment: GNU compiler with OpenMPI
 - Use OpenMPI compiler wrappers
 - `mpicc`
 - `mpicxx`
 - `mpif77`
- Bridges uses the SLURM batch system
 - Jobs submitted from tutorial accounts will need to specify our reservation

```
% sbatch jobscript.sbatch  
% squeue -u $USER  
% scancel <jobid>
```

← Submit job
← View job queue
← Cancel job

Local installation

- VI-HPS tools not yet installed system-wide
 - Source the provided shell code snippet to add local tool installations to \$PATH
 - Required for each shell session

```
% source /home/zhukov/ihpcss18/tools/source.me.gcc-openmpi
```

- Copy tutorial sources to a working directory, e.g. \$HOME

```
% cd $HOME  
% tar zxvf /home/zhukov/ihpcss18/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

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
 - Proposed hands-on setup on Bridges:
 - 2 compute nodes with 2 Intel Xeon E5-2695 v3 CPUs (14 cores per CPU) and 128GB RAM each
Get more details using likwid-topology
 - 8 MPI processes with 6 OpenMP threads each, not utilizing all cores
 - bt-mz_C.8 should run in around 30 seconds on Bridges

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 Bridges:      *
*      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'
icc -o setparams setparams.c -lm
make[2]: Leaving directory `sys'
../sys/setparams bt-mz 8 C
make[2]: Entering directory `../BT-MZ'
mpiifort -mmic -c -O3 -openmp          bt.f
          [...]
mpiifort -mmic -c -O3 -openmp          mpi_setup.f
cd ../common; mpiifort -mmic -c -O3 -openmp          print_results.f
cd ../common; mpiifort -mmic -c -O3 -openmp          timers.f
mpiifort -mmic -O3 -openmp -o ../bin/bt-mz_B.30 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**
 - Or use "**make suite**"

System topology

```
% likwid-topology
```

```
-----  
CPU name:      Intel(R) Xeon(R) CPU E5-2695 v3 @ 2.30GHz  
CPU type:      Intel Xeon Haswell EN/EP/EX processor  
CPU stepping:  2  
*****
```

```
Hardware Thread Topology  
*****
```

```
Sockets:      2  
Cores per socket: 14  
Threads per core: 1  
-----
```

HWThread	Thread	Core	Socket	Available
0	0	0	0	*
1	0	2	0	*
...				

- likwid-topology: thread and cache topology on x86 CPUs
- 28 cores available per node
- Hyperthreading deactivated

NPB-MZ-MPI / BT reference execution

```
% cd bin
% cp ../jobscript/bridges/reference.sbatch.C.8 .
% less reference.sbatch.C.8
#!/bin/bash
#SBATCH --res=ihsssth # use "ihsssth" on Thursday and "ihssfi" on Friday
#SBATCH -J mzmpibt # job name
#SBATCH -o ref-C.8-%j.out # stdout output file
#SBATCH -e ref-C.8-%j.err # stderr output file
#SBATCH --nodes=2 # requested nodes (28 cores/node)
#SBATCH --ntasks=8 # requested MPI tasks
#SBATCH --cpus-per-task=6 # requested logical CPUs/threads per task
#SBATCH --partition RM # partition to use
#SBATCH --export=ALL # export env varibales
#SBATCH --time=00:10:00 # max wallclock time (hh:mm:ss)
#SBATCH --exclusive

# setup modules, add tools to PATH
source /home/zhukov/ihpcss18/tools/source.me.gcc-openmpi
set -x

# OpenMP setup
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export OMP_PLACES=cores
export OMP_PROC_BIND=close

# benchmark configuration
export NPB_MZ_BLOAD=0
CLASS=C
PROCS=8
EXE=./bt-mz_${CLASS}.${PROCS}

# place two MPI ranks per socket
mpirun -x OMP_NUM_THREADS -x OMP_PLACES -x OMP_PROC_BIND --report-bindings \
--map-by ppr:2:socket:PE=$SLURM_CPUS_PER_TASK -bind-to core -n $SLURM_NTASKS $EXE
```

- Examine jobscript

NPB-MZ-MPI / BT reference execution (cont)

```
% sbatch reference.sbatch.C.8
% less ref-C.8-<job_id>.out
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:   16 x   16
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  40

[...]

Time step 180
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 19.25
```

- Copy jobscript and launch as a hybrid MPI+OpenMP application
- Reproducible? CPU frequency constant? Turboboost?

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

NPB-MZ-MPI / BT reference execution (cont)

```
% less ref-C.8-<job_id>.err
[ . . . ]
```

```
node r164 [...] rank 0 bound to socket 0 [...]: [B/B/B/B/B/B/. /. /. /. /. /. /. /.] [. /. /. /. /. /. /. /. /. /. /. /. /.]
node r164 [...] rank 1 bound to socket 0 [...]: [. /. /. /. /. /. /B/B/B/B/B/B/. /.] [. /. /. /. /. /. /. /. /. /. /. /. /.]
node r164 [...] rank 2 bound to socket 1 [...]: [. /. /. /. /. /. /. /. /. /. /. /. /. /.] [B/B/B/B/B/B/. /. /. /. /. /. /. /. /.]
node r164 [...] rank 3 bound to socket 1 [...]: [. /. /. /. /. /. /. /. /. /. /. /. /. /.] [. /. /. /. /. /. /B/B/B/B/B/B/. /.]
node r172 [...] rank 4 bound to socket 0 [...]: [B/B/B/B/B/B/. /. /. /. /. /. /. /. /.] [. /. /. /. /. /. /. /. /. /. /. /. /. /.]
node r172 [...] rank 5 bound to socket 0 [...]: [. /. /. /. /. /. /B/B/B/B/B/B/. /.] [. /. /. /. /. /. /. /. /. /. /. /. /. /.]
node r172 [...] rank 6 bound to socket 1 [...]: [. /. /. /. /. /. /. /. /. /. /. /. /. /.] [B/B/B/B/B/B/. /. /. /. /. /. /. /. /.]
node r172 [...] rank 7 bound to socket 1 [...]: [. /. /. /. /. /. /. /. /. /. /. /. /. /.] [. /. /. /. /. /. /B/B/B/B/B/B/. /.]

[...]
```

- **--report-bindings** makes each rank print to its standard error output the affinity mask that apply to itapplication
- Each socket is represented as a set of square brackets with each core represented by a dot. The core(s) that each rank is bound to is/are denoted by the letter B.