Tuning hands-on: NPB-MZ-MPI / BT

Barnard Tuning Settings

- Modules: load your toolchain of choice (for the workshop, iimpi is preferred: Intel compilers, Intel MPI)
- Compiler flags: see https://doc.zih.tu-dresden.de/software/compilers/ for a starting reference. We use `-mavx -msse4.2 -march=core-sapphirerapids`.

Barnard Runtime Settings

- Assign tasks to cores based on compute/memory bandwidth needs: https://doc.zih.tu-dresden.de/jobs_and_resources/binding_and_distribution_of_tasks/
- OpenMP configuration: generally, pin your threads! Can be a large performance improvement. For this example:

```
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

export OMP_PROC_BIND=close

export OMP_PLACES=cores
```

NPB-MZ-MPI / BT suite

```
% cd $VIHPS_WORKSPACE
% mkdir hands-on && cd hands-on
% tar xvzf $VIHPS_ROOT/hands-on/score-p.tar.gz
% cd score-p
% ls
bin/
bin.scorep/
BT-MZ/
common/
config/
jobscript/
LU-MZ/
Makefile
README
README.install
README.tutorial
SP-MZ/
sys/
```

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - http://www.nas.nasa.gov/Software/NPB
- Start in the $VIHPS_WORKSPACE/hands-on/score-p directory

NPB-MZ-MPI / BT configuration

```
% <editor> config/make.def
...
MPIF77 = mpif77 -f77=ifort
...
#---------------------------------------------------------------
# Global *compile time* flags for Fortran programs
#---------------------------------------------------------------
FFLAGS  = -O3 -g $(OPENMP) -mavx -msse4.2 -march=sapphirerapids
```

- Specify classic ifort
- Tuning flags for Sapphire Rapids (not a huge difference for BT-MZ, but good practice!)

NPB-MZ-MPI / BT build

```
% make bt-mz CLASS=C NPROCS=4
cd BT-MZ; make CLASS=C NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 4 W
mpif77 -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  mpif77 -c  -O3 -fopenmp timers.f
mpif77 -O3 -fopenmp -o ../bin/bt-mz_C.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_C.4
make: Leaving directory 'BT-MZ'
```

- Benchmark name:
  - **bt-mz**, lu-mz, sp-mz
- Number of MPI processes:
  - NPROCS=**4**
- Benchmark class:
  - S, W, A, B, **C**, D, E
  - CLASS=**C**

NPB-MZ-MPI / BT job submission

```
% cp jobscript/[barnard|claix-2023]/bt-mz.sbatch .
% cat bt-mz.sbatch

# SBATCH -J reference
...
# Generic OpenMP thread pinning
export OMP_PROC_BIND=close
export OMP_PLACES=cores
...

% sbatch bt-mz.sbatch
```

- Bring appropriate job script into main benchmark directory
- Note the job name (used to sort output) and the OpenMP thread pinning variables (for your own codes)
- Note the output locations (site-specific!)
- Run with workshop account and reservation (

NPB-MZ-MPI / BT reference execution

```
% cat reference/bt-mz.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:     4

 Use the default load factors with threads
 Total number of threads:     48   (  12.0 threads/process)

 Calculated speedup =     47.99

 Time step    1
 [... More application output ...]
 Time step  200
 [... More application output ...]
 BT-MZ Benchmark Completed.
 Time in seconds = 10.77
```

- Launch as a hybrid MPI+OpenMP application

Save the benchmark run time to be able to refer to it later. (Beware of potential over-subscription)