Hands-on:
NPB-MZ-MPI / BT

VI-HPS Team

Tutorial exercise objectives

- Familiarize 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
  - …

Stampede2

- Intel Xeon Phi 7250 (Knights Landing, **KNL**)
  - MIC-AVX512 architecture
  - 68 cores on single socket
  - 4 hardware threads per core
  - = 272 hardware threads per node
  - 1.4 GHz
  - 96 GB DDR4 + 16 GB MCDRAM
- 4200 KNL compute nodes
  - “normal” queue (max 256 nodes, 48 hours)
  - “development” queue (max 16 nodes, 2 hrs)
- Workshop reservations
  - 60 nodes
  - VI-HPS_KNL_DAY1 (DAY2..5)

- Intel Xeon Platinum 8160 (Skylake, **SKX**)
  - CORE-AVX512 architecture
  - 48 cores on two sockets
  - 2 hardware threads per core
  - = 96 hardware threads per node
  - 2.1 GHz nominal (1.4 – 3.7 GHz)
  - 192 GB DDR4
- 1736 SKX compute nodes
  - “skx-normal” queue (max 128 nodes, 48 hours)
  - “skx-dev” queue (max 4 nodes, 2 hours)
- Workshop reservations
  - 20 nodes
  - VI-HPS_SKX_DAY1 (DAY2..5)

Access to Stampede2

```
# Connect to a Stampede2 login node
% ssh -Y userid@stampede2.tacc.utexas.edu
```

- Logging in to Stampede2
  - use assigned training account ID, password & token code

```
$HOME
$WORK
$SCRATCH

/home1/03529/tg828282/Tutorial
(shortcut: ~tg828282/Tutorial)
```

Tutorial materials

- File systems & directories
  - Use $SCRATCH for the tutorial
    - Fast Lustre file system, ~30 PB
    - No backup
    - Files may be automatically purged 10 days after last modification

- More extensive documentation:
  - https://portal.tacc.utexas.edu/user-guides/stampede2

Compiling & job submission

- Development environment: Intel compilers with Intel MPI
  - Use Intel’s MPI compiler wrappers
    - `mpiicc`
    - `mpiicpc`
    - `mpiifort`

`module load gcc`

GCC compilers with Intel MPI

- `mpicc`
- `mpicxx`
- `mpifc`

- Stampede2 uses the SLURM batch system
  - Jobs submitted from tutorial accounts with provided job scripts will automatically be run in a 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 provided shell code snippet to add local tool installations to $PATH
  - Required for each shell session

```
% source ~tg828282/Tutorial/vihps-intel.sh
```

- Copy tutorial sources to your working directory, ideally on a parallel file system (recommended: $SCRATCH)

```
% cd $SCRATCH
% tar zxvf ~tg828282/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

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 Stampede2:    *
 *       make bt-mz CLASS=C NPROCS=32                          *
 ***************************************************************
```

- Type “make” for instructions

Building an NPB-MZ-MPI benchmark

```
% make bt-mz CLASS=C NPROCS=32
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 32 C
make[2]: Entering directory `../BT-MZ'
mpiifort -c  -g -O3 -qopenmp       bt.f
                                     […]
mpiifort -c  -g -O3 -qopenmp       mpi_setup.f
cd ../common;  mpiifort -c  -g -O3 -qopenmp       print_results.f
cd ../common;  mpiifort -c  -g -O3 -qopenmp       timers.f
mpiifort -g -O3 -qopenmp   -o ../bin/bt-mz_C.32 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.32
make[1]: Leaving directory `BT-MZ'
```

- 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): CLASS=**C**

Shortcut: `% make suite`

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 Stampede2:
    - 2 compute nodes with 1 Intel Xeon Phi 7250 CPU (Knights Landing, KNL) each
    - 32 MPI processes with 4 OpenMP threads each
  - bt-mz_C.32 should run in less than 30 seconds

NPB-MZ-MPI / BT reference execution

```
% cd bin
% cp ../jobscript/stampede2/reference.sbatch .
% less reference.sbatch
% sbatch reference.sbatch
% less mzmpibt.o<job_id>
 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:     32
 Total number of threads:      128  (  4.0 threads/process)

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

 BT-MZ Benchmark Completed.
 Time in seconds = 22.34
```

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

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
  - See next slide for details
- Make clean and build new tool-specific executable

```
% make clean
% make bt-mz CLASS=C NPROCS=32
Built executable ../bin.$(TOOL)/bt-mz_C.32
```

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

```
% cd bin.$(TOOL)
% cp ../jobscript/stampede2/$(TOOL).sbatch .
% sbatch $(TOOL).sbatch
```

NPB-MZ-MPI / BT: config/make.def

```
#                SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.
#
#---------------------------------------------------------------------------

#---------------------------------------------------------------------------
# Configured for generic MPI with INTEL compiler
#---------------------------------------------------------------------------
#OPENMP  = -fopenmp      # GCC compiler
OPENMP = -qopenmp        # Intel compiler

...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
MPIF77 = mpiifort

# Alternative variant to perform instrumentation
#MPIF77 = scorep --user mpiifort

# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP) mpiifort
...
```

Default (no instrumentation)

Hint: uncomment a compiler wrapper to do instrumentation

Acknowledgement

Thanks to the

Texas Advanced Computing Center (TACC)

for supporting this workshop by providing
training accounts and compute time!