Hands-on / Demo:
Instrumenting NPB-MZ-MPI / BT & summary profile measurement

Markus Geimer
Jülich Supercomputing Centre
Performance analysis steps

- 0.0 Reference execution for validation
- 1.0 Program instrumentation
  - 1.1 Summary measurement collection
- 2.0 Summary experiment scoring
  - 2.1 Summary measurement collection with filtering
  - 2.2 Filtered summary analysis report examination
- 3.0 Event trace collection
  - 3.1 Event trace examination & analysis
First step: Set up the environment

- In addition to the compiler/MPI environment

```bash
module load Intel
module load IntelMPI
```

set up Score-P, Scalasca, and Cube by loading the corresponding modules

```bash
module load Score-P
module load Scalasca
module load CubeGUI
```
Second step: Adjust build configuration

- Edit `config/make.def` to adjust build configuration
  - Modify specification of compiler/linker: MPIF77

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

# Alternative variants to perform instrumentation
... 
MPIF77 = scorep --user mpiifort

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

Comment out the compiler specification...

...and uncomment the Score-P compiler wrapper specification
Third step: Build the instrumented benchmark

% make clean
% make bt-mz NPROCS=8 CLASS=C
cd BT-MZ; make CLASS=B NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ..;/sys; icc -o setparams setparams.c -lm
./sys/setparams bt-mz 30 B
scorep mpiifort -c -O3 -qopenmp bt.f

[...]
cd ../common; scorep mpiifort -c -O3 -qopenmp timers.f
scorep mpiifort -O3 -qopenmp ..;/bin.scorep/bt-mz_C.8 
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.scorep/bt-mz_C.8
make: Leaving directory 'BT-MZ'

- Return to root directory and clean-up
- Re-build executable using Score-P compiler wrapper
Measurement configuration: scorep-info

% scorep-info config-vars --full
SCOREP_ENABLE_PROFILING
   Description: Enable profiling
   [...]  
SCOREP_ENABLE_TRACING
   Description: Enable tracing
   [...]  
SCOREP_TOTAL_MEMORY
   Description: Total memory in bytes for the measurement system
   [...]  
SCOREP_EXPERIMENT_DIRECTORY
   Description: Name of the experiment directory
   [...]  
SCOREP_FILTERING_FILE
   Description: A file name which contain the filter rules
   [...]  
SCOREP_METRIC_PAPI
   Description: PAPI metric names to measure
   [...]  
SCOREP_METRIC_RUSAGE
   Description: Resource usage metric names to measure
   [...]  

- Score-P measurements are configured via environmental variables
Scalasca convenience command: scalasca –analyze (scan)

% scalasca –analyze
Scalasca 2.4: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
   where {options} may include:
   -h    Help: show this brief usage message and exit.
   -v    Verbose: increase verbosity.
   -n    Preview: show command(s) to be launched but don't execute.
   -q    Quiescent: execution with neither summarization nor tracing.
   -s    Summary: enable runtime summarization. [Default]
   -t    Tracing: enable trace collection and analysis.
   -a    Analyze: skip measurement to (re-)analyze an existing trace.
   -e    exptdir : Experiment archive to generate and/or analyze.
               (overrides default experiment archive title)
   -f    filtfile : File specifying measurement filter.
   -l    lockfile : File that blocks start of measurement.
   -m    metrics : Metric specification for measurement.

- Scalasca measurement collection & analysis nexus
scan: Automatic measurement configuration

- scan configures Score-P measurement by automatically setting some environment variables and exporting them
  - E.g., experiment title, profiling/tracing mode, filter file, ...
  - Precedence order:
    - Command-line arguments
    - Environment variables already set
    - Automatically determined values

- Also, scan includes consistency checks and prevents corrupting existing experiment directories

- For tracing experiments, after trace collection completes the Scalasca automatic parallel trace analysis is initiated
  - Uses identical launch configuration to that used for measurement (i.e., the same allocated compute resources)
Forth step: Summary measurement collection

- Change to the directory containing the new executable before running it with the desired configuration
- Check settings
- Submit job
Forth step: Summary measurement collection (cont.)

% 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: 8

Use the default load factors with threads
Total number of threads: 48 (6.0 threads/process)

Calculated speedup = 47.97

Time step 1

[... More application output ...]
BT-MZ summary analysis report

% ls -F
bt-mz_C.8* mzmpibt.e<job_id> mzmpibt.o<job_id>
scalasca.sbatch scorep_bt-mz_C_8x6_sum/

% ls scorep_bt-mz_C_8x6_sum/
profile.cubex scorep.cfg scorep.log

- Creates experiment directory including
  - The analysis report that was collated after measurement (profile.cubex)
  - A record of the measurement configuration (scorep.cfg)
  - The measurement log output (scorep.log)