Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

VI-HPS Team

Score-P
Scalable performance measurement infrastructure for parallel codes
Performance engineering workflow

- Preparation
  - Prepare application with symbols
  - Insert extra code (probes/hooks)

- Measurement
  - Collection of performance data
  - Aggregation of performance data

- Optimization
  - Modifications intended to eliminate/reduce performance problem

- Analysis
  - Calculation of metrics
  - Identification of performance problems
  - Presentation of results
Fragmentation of tools landscape

- Several performance tools co-exist
  - Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
  - Limited or expensive interoperability
- Complications for user experience, support, training

**Tools**

- **Vampir**
  - VampirTrace
  - OTF
- **Scalasca**
  - EPILOG / CUBE
- **TAU**
  - TAU native formats
- **Periscope**
  - Online measurement
Design goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation and sampling
  - Flexible measurement without re-compilation
  - Recording time, visits, communication data, hardware counters
  - Access and reconfiguration also at runtime
  - Support for MPI, SHMEM, OpenMP, Pthreads, CUDA, OpenCL, OpenACC and their valid combinations
  - Highly scalable I/O

- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Robustness
  - Open Source: 3-clause BSD license
Score-P overview

**Applications**
- Vampir
- Scalarsca
- CUBE
- TAU
- TAUdb
- Periscope

**Score-P measurement infrastructure**
- Event traces (OTF2)
- Call-path profiles (CUBE4, TAU)
- Hardware counter (PAPI, rusage, PERF, plugins)
- Online interface

**Instrumentation wrapper**
- Process-level parallelism (MPI, SHMEM)
- Thread-level parallelism (OpenMP, Pthreads)
- Accelerator-based parallelism (CUDA, OpenCL, OpenACC)
- Source code instrumentation (Compiler, PDT, User)
- Sampling interrupts (PAPI, PERF)
Hands-on: NPB-MZ-MPI / BT
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
  - 2.2 Filtered summary analysis report examination
- 3.0 Event trace collection
  - 3.1 Event trace examination & analysis
Local installation

- **VI-HPS tools**
  - CUBE release preview installed locally
  - Load environment modules, then load tool modules

```
% module use /p/scratch/share/VI-HPS/JURECA/mf
% module load Intel IntelMPI Score-P CubeGUI
```

- **Go to working directory with tutorial exercise**

```
% cd $SCRATCH/$USER/NPB-3.3-MZ-MPI
% ls -F
BT-MZ/ Makefile README.install SP-MZ/ config/ sys/
LU-MZ/ README README.tutorial bin/ common/ jobscript/
```
NPB-MZ-MPI / BT instrumentation

```
# The Fortran compiler used for MPI programs
MPIF77 = mpif77

# Alternative variants to perform instrumentation
...#MPIF77 = scorep --user mpif77
...MPIF77 = $(PREP) mpif77

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

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

Uncomment the generic compiler wrapper specification
NPB-MZ-MPI / BT instrumented build

% make clean

% make bt-mz CLASS=C NPROCS=8 PREP="scorep"

```bash
cd BT-MZ; make CLASS=C NPROCS=8 VERSION=
makesh: Entering directory 'BT-MZ'
cd ..; make -C NPROCS=8 clean
cc -o setparams setparams.c -lm
cc -o bt-mz setparams bt-mz 8 C
scorep mpif77 -c -O3 -fopenmp bt.f

[...]
```

```bash
cd ../sys; cc -o setparams setparams.c -lm
cc -o bt-mz setparams bt-mz 8 C
```

```bash
cscorep mpif77 -c -O3 -fopenmp timers.f
```

```bash
scorep mpif77 -c -O3 -fopenmp -o ../../../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_gbc.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

<table>
<thead>
<tr>
<th>Configuration Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>% scorep-info config-vars --full</td>
<td></td>
</tr>
<tr>
<td>SCOREP_ENABLE_PROFILING</td>
<td>Description: Enable profiling</td>
</tr>
<tr>
<td><strong>SCOREP_ENABLE_TRACING</strong></td>
<td>Description: Enable tracing</td>
</tr>
<tr>
<td>SCOREP_TOTAL_MEMORY</td>
<td>Description: Total memory in bytes for the measurement system</td>
</tr>
<tr>
<td>SCOREP_EXPERIMENT_DIRECTORY</td>
<td>Description: Name of the experiment directory</td>
</tr>
<tr>
<td>SCOREP_FILTERING_FILE</td>
<td>Description: A file name which contain the filter rules</td>
</tr>
<tr>
<td>SCOREP_METRIC_PAPI</td>
<td>Description: PAPI metric names to measure</td>
</tr>
<tr>
<td>SCOREP_METRIC_RUSAGE</td>
<td>Description: Resource usage metric names to measure</td>
</tr>
</tbody>
</table>

- Score-P measurements are configured via environmental variables
Summary measurement collection

- Change to the directory containing the new executable before running it with the desired configuration
- Check settings
- Submit job

```
% cd bin.scorep
% cp ../jobscript/jureca/scorep.sbatch .
% vi scorep.sbatch

[...]
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum
[...]
% sbatch ./scorep.sbatch
```

Leave other lines commented out for the moment
Summary measurement collection

% less npb_btmz_scorep.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

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 examination

- Creates experiment directory including:
  - A record of the measurement configuration (scorep.cfg)
  - The analysis report that was collated after measurement (profile.cubex)
- Interactive exploration with Cube
Further information

- Community instrumentation & measurement infrastructure
  - Instrumentation (various methods)
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data
- Available under 3-clause BSD open-source license
- Documentation & Sources:
  - [http://www.score-p.org](http://www.score-p.org)
- User guide also part of installation:
  - `<prefix>/share/doc/scorep/{pdf,html}/`
- Support and feedback: support@score-p.org
- Subscribe to news@score-p.org, to be up to date