Score-P – A Joint Performance Measurement
Run-Time Infrastructure

Anupam Karmakar | Lenovo Global Technology
Score-P Consortium

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA
Score-P Functionality Highlights

• Provide typical functionality for HPC performance tools
• Support all fundamental concepts of partner’s tools
• MPI, OpenMP, and hybrid parallelism (and serial)
• Enhanced functionality (OpenMP 4.0, CUDA, highly scalable I/O)
Score-P Design Goals

• Functional requirements
  – Generation of call-path profiles and event traces
  – Recording time, visits, communication data, hardware counters
  – Support for MPI, SHMEM, OpenMP, PTHREAD, CUDA, OpenCL and combinations

• Non-functional requirements
  – Portability: all major HPC platforms
  – Scalability: Peta-scale, and possibly beyond
  – Low measurement overhead
  – Easy and uniform installation through UNITE framework
  – Robustness
  – Open Source: New BSD License
Hands-on Session:
IvyMUC cluster
Score-P Performance Analysis Exercise

- Reference preparation for validation
- Program instrumentation

- Summary measurement collection
- Summary analysis report examination
- Summary experiment scoring

- Summary measurement collection with filtering
- Filtered summary analysis report examination

- Event trace collection
- Event trace examination & analysis
NPB-MZ-MPI / BT Instrumentation

• Check the preferred compilers

% module list

• Copy tutorial sources to your working directory (ideally on a parallel filesystem like as $SCRATCH)

% mkdir SCOREP
% cd SCOREP
% tar -zxvf /lrz/sys/courses/PPHPS18/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
Code Instrumentation

- Load required modules for Score-p

```
% module load qt cube scorep
INFO(Score-p): invoking mpi.intel
```

Currently Loaded Modulefiles:
1) admin/1.0  4) mkl/11.3  7) subversion/1.8  10) qt/4.8
2) tempdir/1.0  5) mpi.intel/5.1  8) git/latest  11) png/1.5.13
3) intel/16.0  6) lrz/default  9) scorep/3.0  12) cube/4.3
NPB-MZ-MPI / BT Instrumentation

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

```bash
# SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#
# Items in this file may need to be changed for each platform.
#
COMPFLAGS = -qopenmp -xHost # intel
...
#
# The Fortran compiler used for MPI programs
#
#MPIF77 = mpif90

MPIF77 = scorep mpif90

# Alternative variants to perform instrumentation
...

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

Uncomment the Score-P compiler wrapper specification
Build Instrumented Code

- Return to root directory and clean-up

% make clean

- Re-build executable using Score-P compiler wrapper

% make bt-mz CLASS=C NPROCS=4
============
= NAS PARALLEL BENCHMARKS 3.3 =
= MPI+OpenMP Multi-Zone Versions =
= F77 =
============
cd BT-MZ; make CLASS=C NPROCS=4 VERSION=
make[1]: Entering directory './PPHPS18/NPB3.3-MZ-MPI/BT-MZ'
make[2]: Entering directory './PPHPS18/NPB3.3-MZ-MPI/sys'
c -o setparams setparams.c -lm
make[2]: Leaving directory './PPHPS18/NPB3.3-MZ-MPI/sys'
../sys/setparams bt-mz 4 C
make[2]: Entering directory './PPHPS18/NPB3.3-MZ-MPI/BT-MZ'
scorep mpir90 -c -xHost -O3 -gopenmp -xHost bt_scorep_user.F
scorep mpir90 -c -xHost -O3 -gopenmp -xHost initialize.f
scorep mpir90 -c -xHost -O3 -gopenmp -xHost exact_solution.f
scorep mpir90 -c -xHost -O3 -gopenmp -xHost exact_rhs.f
Measurement Configuration: scorep-info

• Score-P measurements are configured via environmental variables:

```bash
% 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
 [... More configuration variables ...]
Summary Measurement Collection

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

% cd bin.scorep
% cp ../jobscript/CoolMUC-3/scorep.sbatch .

• Check the jobsctipt

% vim scorep.sbatch
export NPB_MZ_BLOAD=0
export OMP_NUM_THREADS=4
export SCOREP_EXPERIMENT_DIRECTORY=scorep_sum
mpiexec -n 4 ./bt-mz_C.4

• Submit job to the queue

% sbatch -reservation=course_PPoHPS scorep.sbatch
Summary Measurement Collection

• Check the output logs from your run

% less xxxx.out

NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

Number of zones:  8 x 8
Iterations: 200  dt:  0.000100
Number of active processes:  4

Use the default load factors with threads
Total number of threads:  16  ( 4.0 threads/process)

Calculated speedup =  16

Time step  1

[... More application output ...]
Score-P Performance Analysis Exercise

- Reference preparation for validation
- Program instrumentation

- Summary measurement collection
- Summary analysis report examination
- Summary experiment scoring

- Summary measurement collection with filtering
- Filtered summary analysis report examination

- Event trace collection
- Event trace examination & analysis
Summary Analysis Report

• Experiment directory .scorep_sum containing
  – scorep.cfg: record of the measurement configuration
  – profile.cubex: analysis report that was collated after measurement

```bash
$ ls
bt-mz_C.4  scorep_??????
$ ls scorep_??????
profile.cubex  scorep.cfg
```

• Interactive exploration with CUBE

```bash
$ cube scorep_sum/profile.cubex
```

[CUBE GUI showing summary analysis report]
Congratulations!

• If you made it this far, you successfully used Score-P to
  – instrument the application
  – analyze its execution with a summary measurement, and
  – examine it with one the interactive analysis report explorer GUIs

• ... revealing the call-path profile annotated with
  – the “Time” metric
  – Visit counts
  – MPI message statistics (bytes sent/received)

• ... but how good was the measurement?
  – The measured execution produced the desired valid result
  – however, the execution took rather longer than expected!
    - even when ignoring measurement start-up/completion, therefore
    - it was probably dilated by instrumentation/measurement overhead
Summary Analysis Result Scoring

• Report scoring as clear text output

```
% scorep-score scorep_sum/profile.cubex
```

Estimated aggregate size of event trace:
Estimated requirements for largest trace buffer (max_buf):
Estimated memory requirements (SCOREP_TOTAL_MEMORY):
(hint: When tracing set SCOREP_TOTAL_MEMORY=41GB to avoid intermediate flushes or reduce requirements using USR regions filters.)

<table>
<thead>
<tr>
<th>flt</th>
<th>type</th>
<th>max_buf[B]</th>
<th>visits</th>
<th>time[s]</th>
<th>time[%]</th>
<th>time/visit[us]</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td></td>
<td>43,081,533,638</td>
<td>6,586,493,505</td>
<td>2198.78</td>
<td>100.0</td>
<td>0.33</td>
<td>ALL</td>
</tr>
<tr>
<td>USR</td>
<td></td>
<td>42,988,632,934</td>
<td>6,574,788,217</td>
<td>1027.77</td>
<td>46.7</td>
<td>0.16</td>
<td>USR</td>
</tr>
<tr>
<td>OMP</td>
<td></td>
<td>88,100,608</td>
<td>10,975,232</td>
<td>1164.35</td>
<td>53.0</td>
<td>106.09</td>
<td>OMP</td>
</tr>
<tr>
<td>COM</td>
<td></td>
<td>4,697,810</td>
<td>722,740</td>
<td>2.43</td>
<td>0.1</td>
<td>3.36</td>
<td>COM</td>
</tr>
<tr>
<td>MPI</td>
<td></td>
<td>102,286</td>
<td>7,316</td>
<td>4.22</td>
<td>0.2</td>
<td>576.55</td>
<td>MPI</td>
</tr>
</tbody>
</table>

160 GB total memory
41 GB per rank!

• Region/callpath classification
  – MPI (pure MPI library functions)
  – OMP (pure OpenMP functions/regions)
  – USR (user-level source local computation)
  – COM (“combined” USR + OpenMP/MPI)
  – ANY/ALL (aggregate of all region types)
Summary Analysis Report Breakdown

• Score report breakdown by region

```bash
% scorep-score -r scorep_sum/profile.cubex

[...]
[...]
flt type max_buf[B] visits time[s] time[\%] time/visit[us] region
ALL 43,081,533,638 6,586,493,505 2198.78 100.0 0.33 ALL
USR 42,988,632,934 6,574,788,217 1027.77 46.7 0.16 USR
OMP 88,100,608 10,975,232 1164.35 53.0 106.09 OMP
COM 4,697,810 722,740 2.43 0.1 3.36 COM
MPI 102,286 7,316 4.22 0.2 576.55 MPI

USR 13,812,365,034 2,110,313,472 271.61 12.4 0.13 matmul_sub_
USR 13,812,365,034 2,110,313,472 248.43 11.3 0.12 matvec_sub_
USR 13,812,365,034 2,110,313,472 472.63 21.5 0.22 binvcrhs_
USR 596,197,758 87,475,200 15.67 0.7 0.18 lhsinit_
USR 596,197,758 87,475,200 11.96 0.5 0.14 binvrhs_
USR 447,869,968 68,892,672 7.47 0.3 0.11 exact_solution_
```

Almost 40 GB just for these 6 regions
Score-P Performance Analysis Exercises

- Reference preparation for validation
- Program instrumentation

- Summary measurement collection
- Summary analysis report examination
- Summary experiment scoring

- Summary measurement collection with filtering
- Filtered summary analysis report examination

- Event trace collection
- Event trace examination & analysis
Summary Analysis Report Filtering

• Report scoring with prospective filter listing 6 USR regions

```
% cat ../config/scorep.filt
SCOREP_REGION_NAMES_BEGIN EXCLUDE
binvcrhs*
matmul_sub*
matvec_sub*
exact_solution*
binvcrhs*
lhs*init*
timer_*

% scorep-score -f ../config/scorep.filt scorep_sum/profile.cubex
```

Estimated aggregate size of event trace: 447MB
Estimated requirements for largest trace buffer (max_buf): 112MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):
447 MB of memory in total, 112 MB per rank!

(hint: When tracing set SCOREP_TOTAL_MEMORY=97MB to avoid intermediate flushes or reduce requirements using USR regions filters.)
## Summary Analysis Report Filtering

- Score report breakdown by region

```bash
% scorep-score -r -f ../config/scorep.filt > scorep_sum/profile.cubex
```

<table>
<thead>
<tr>
<th>filter</th>
<th>type</th>
<th>max buf [B]</th>
<th>visits</th>
<th>time [s]</th>
<th>time [%]</th>
<th>time/visit [us]</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>ALL</td>
<td>43,081,533,638</td>
<td>6,586,493,505</td>
<td>2198.78</td>
<td>100.0</td>
<td>0.33</td>
<td>ALL</td>
</tr>
<tr>
<td>-</td>
<td>USR</td>
<td>42,988,632,934</td>
<td>6,574,788,217</td>
<td>1027.77</td>
<td>46.7</td>
<td>0.16</td>
<td>USR</td>
</tr>
<tr>
<td>-</td>
<td>OMP</td>
<td>88,100,608</td>
<td>10,975,232</td>
<td>1164.35</td>
<td>53.0</td>
<td>106.09</td>
<td>OMP</td>
</tr>
<tr>
<td>-</td>
<td>COM</td>
<td>4,697,810</td>
<td>722,740</td>
<td>2.43</td>
<td>0.1</td>
<td>3.36</td>
<td>COM</td>
</tr>
<tr>
<td>-</td>
<td>MPI</td>
<td>102,286</td>
<td>7,316</td>
<td>4.22</td>
<td>0.2</td>
<td>576.55</td>
<td>MPI</td>
</tr>
</tbody>
</table>

**Filtered routines marked with ‘+’**

- ALL 92,930,812 11,709,917 1171.00 53.3 100.00 ALL-FLT
- FLT 42,988,602,852 6,574,783,588 1027.77 46.7 0.16 FLT
- OMP 88,100,608 10,975,232 1164.35 53.0 106.09 OMP-FLT
- COM 4,697,810 722,740 2.43 0.1 3.36 COM-FLT
- MPI 102,286 7,316 4.22 0.2 576.55 MPI-FLT
- USR 30,108 4,629 0.00 0.0 0.33 USR-FLT

+ USR 13,812,365,034 2,110,313,472 271.61 12.4 0.13 matmul_sub_
+ USR 13,812,365,034 2,110,313,472 248.43 11.3 0.12 matvec_sub_
+ USR 13,812,365,034 2,110,313,472 472.63 21.5 0.22 binvarhs_
+ USR 596,197,758 87,475,200 15.67 0.7 0.18 linsinit_
+ USR 596,197,758 87,475,200 11.96 0.5 0.14 binvarhs_
+ USR 447,869,968 68,892,672 7.47 0.3 0.11 exact_solution_
Filtered Summary Measurement

• Set new experiment directory and re-run measurement with new filter configuration
  – Adjust configuration and re-run measurement

```bash
%vim scorep.sbatch
export OMP_NUM_THREADS=4
export SCOREP_EXPERIMENT_DIRECTORY=scorep_sum_with_filter
export SCOREP_FILTERING_FILE=../config/scorep.filt

mpiexec -n 4 ./bt_mz_C.4
```

– Submit job

```bash
%sbatch scorep.sbatch
```
Filtered Summary Analysis

• Scoring of new analysis report as clear text output

```bash
% scorep-score scorep_sum/profile.cubex
```

Estimated aggregate size of event trace: 447MB
Estimated requirements for largest trace buffer (max_buf): 112MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY): 120MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=120MB to avoid intermediate flushes or reduce requirements using USR regions filters.)

<table>
<thead>
<tr>
<th>flt</th>
<th>type</th>
<th>max_buf[B]</th>
<th>visits</th>
<th>time[s]</th>
<th>time[%]</th>
<th>time/visit[us]</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>117,089,350</td>
<td>11,709,917</td>
<td>855.14</td>
<td>100.0</td>
<td>73.03</td>
<td>ALL</td>
<td></td>
</tr>
<tr>
<td>OMP</td>
<td>112,237,312</td>
<td>10,975,232</td>
<td>850.87</td>
<td>99.5</td>
<td>77.53</td>
<td>OMP</td>
<td></td>
</tr>
<tr>
<td>COM</td>
<td>4,697,810</td>
<td>722,740</td>
<td>2.58</td>
<td>0.3</td>
<td>3.57</td>
<td>COM</td>
<td></td>
</tr>
<tr>
<td>MPI</td>
<td>124,120</td>
<td>7,316</td>
<td>1.69</td>
<td>0.2</td>
<td>231.43</td>
<td>MPI</td>
<td></td>
</tr>
<tr>
<td>USR</td>
<td>30,108</td>
<td>4,629</td>
<td>0.00</td>
<td>0.0</td>
<td>0.34</td>
<td>USR</td>
<td></td>
</tr>
</tbody>
</table>

• Notice the significant reduction in runtime (measurement overhead)
  – Not only reduced time for USR regions, but MPI/OMP reduced too!
Score-P Performance Analysis Exercise

- Reference preparation for validation
- Program instrumentation

- Summary measurement collection
- Summary analysis report examination
- Summary experiment scoring

- Summary measurement collection with filtering
- Filtered summary analysis report examination

- Event trace collection
- Event trace examination & analysis
Warnings and Tips Regarding Tracing

• Traces can become extremely large and unwieldy
  – Size is proportional to number of processes/threads (width), duration (length) and detail (depth) of measurement

• Traces containing intermediate flushes are of little value
  – Uncoordinated flushes result in cascades of distortion
  – Reduce size of trace
  – Increase available buffer space

• Traces should be written to a parallel file system
  – /work or /scratch are typically provided for this purpose

• Moving large traces between file systems is often impractical
  – However, systems with more memory can analyze larger traces
  – Alternatively, run trace analyzers with undersubscribed nodes
Trace Measurement Collection...

• Adjust configuration and re-run the application using the tracing mode of Score-P

```bash
% vim scorep.sbatch

export OMP_NUM_THREADS=6
export SCOREP_EXPERIMENT_DIRECTORY=scorep_trace
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_ENABLE_TRACING=true
export SCOREP_ENABLE_PROFILING=false
export SCOREP_TOTAL_MEMORY=140M

mpiexec -n 4 ./bt_mz_C.4
```

• Submit job

```bash
%sbatch scorep.sbatch
```
Trace Measurement Collection...

- Separate trace file per thread written straight into new experiment directory .*/scorep_trace
- Interactive trace exploration with Vampir

```
% module load vampir/9.0
% vampir scorep_trace/traces.otf2
```
Further Information

• Community instrumentation & measurement infrastructure
  – Instrumentation (various methods)
  – Basic and advanced profile generation
  – Event trace recording
  – Online access to profiling data

• Available under New BSD open-source license

• Documentation & Sources:
  – http://www.score-p.org

• User guide also part of installation:
  – <prefix>/share/doc/scorep/{pdf,html}/

• Contact: info@score-p.org
• Bugs: support@score-p.org
Acknowledgements

• Course material courtesy
  – Ilya Zukhov, JSC/FZ-Juelich
  – Carmen Navrrete, LRZ APP team
  – VI-HPS: www.vi-hps.org
  – Score-P Team
  – Scalasca and Cube Team