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

arm
JÜLICH Forschungszentrum
Leibniz Supercomputing Centre
TECHNISCHE UNIVERSITÄT DARMSTADT
Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE
Barcelona Supercomputing Center
Lawrence Livermore National Laboratory
RWTH AACHEN UNIVERSITY
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart

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

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

BT-MZ summary analysis result scoring

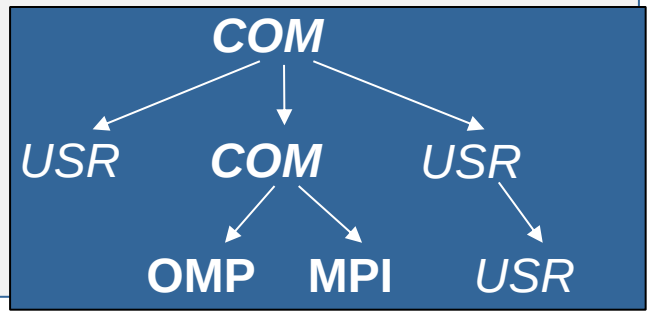
```
% scorep-score scorep_bt-mz_sum/profile.cubex

Estimated aggregate size of event trace:                   160 GB
Estimated requirements for largest trace buffer (max_buf):   6 GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):         6 GB
(warning: The memory requirements cannot be satisfied by Score-P to avoid
 intermediate flushes when tracing. Set SCOREP_TOTAL_MEMORY=4G to get the
 maximum supported memory or reduce requirements using USR regions filters.)

flt type     max_buf[B]        visits  time[s] time[%] time/visit[us]  region
     ALL  5,421,104,056 6,586,922,497  8162.56   100.0           1.24  ALL
     USR  5,407,570,350 6,574,832,225  3960.99    48.5           0.60  USR
     OMP     15,783,372    10,975,232  4085.92    50.1         372.29  OMP
     MPI        944,200       386,560    92.05     1.1         238.13  MPI
     COM        665,210       728,480    23.60     0.3          32.40  COM
```

160 GB total memory
6 GB per rank!

- Report scoring as textual output

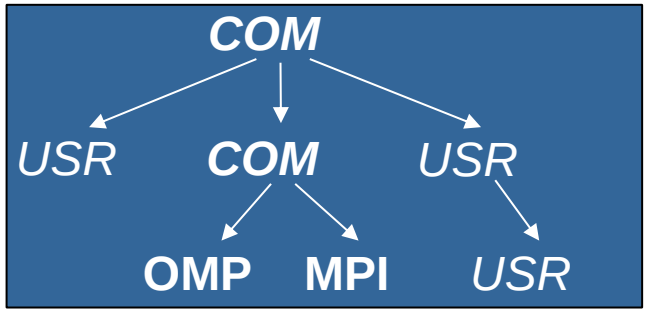


- Region/callpath classification
  - **MPI** pure MPI functions
  - **OMP** pure OpenMP regions
  - **USR** user-level computation
  - **COM** "combined" USR+OpenMP/MPI
  - **ANY/ALL** aggregate of all region types

BT-MZ summary analysis report breakdown

```
% scorep-score -r scorep_bt-mz_sum/profile.cubex
  [...]
  [...]
flt type     max_buf[B]        visits  time[s] time[%] time/visit[us]  region
     ALL  5,421,104,056 6,586,922,497  8162.56   100.0           1.24  ALL
     USR  5,407,570,350 6,574,832,225  3960.99    48.5           0.60  USR
     OMP     15,783,372    10,975,232  4085.92    50.1         372.29  OMP
     MPI        944,200       386,560    92.05     1.1         238.13  MPI
     COM        665,210       728,480    23.60     0.3          32.40  COM

     USR  1,741,005,318 2,110,313,472  1204.11    14.8           0.57  matmul_sub_
     USR  1,741,005,318 2,110,313,472   851.97    10.4           0.40  matvec_sub_
     USR  1,741,005,318 2,110,313,472  1754.58    21.5           0.83  binvcrhs_
     USR     76,367,538    87,475,200    65.93     0.8           0.75  lhsinit_
     USR     76,367,538    87,475,200    59.43     0.7           0.68  binvrhs_
     USR     56,913,688    68,892,672    24.62     0.3           0.36  exact_solution_
```



More than
5 GB just for these
6 regions

BT-MZ summary analysis score

- Summary measurement analysis score reveals
  - Total size of event trace would be ~160 GB
  - Maximum trace buffer size would be ~6 GB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.7% of the trace requirements are for USR regions
    - purely computational routines never found on COM call-paths common to communication routines or OpenMP parallel regions
  - These USR regions contribute around 49% of total time
    - however, much of that is very likely to be measurement overhead for frequently-executed small routines
- Advisable to tune measurement configuration
  - Specify an adequate trace buffer size
  - Specify a filter file listing (USR) regions not to be measured

BT-MZ summary analysis report filtering

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

% scorep-score -f ../config/scorep.filt -c 2 \
      scorep_bt-mz_sum/profile.cubex

Estimated aggregate size of event trace:                   1156 MB
Estimated requirements for largest trace buffer (max_buf):   41 MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):        49 MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=49MB to avoid \
>intermediate flushes
 or reduce requirements using USR regions filters.)
```

- Report scoring with prospective filter listing 6 USR regions

1,1 GB of memory in total, 49 MB per rank!

(Including 2 metric values)

BT-MZ summary analysis report filtering

```
% scorep-score -r -f ../config/scorep.filt \
      scorep_bt-mz_sum/profile.cubex
flt type    max_buf[B]        visits  time[s] time[%] time/      region
                                                      visit[us]
 -   ALL 5,421,104,056 6,586,922,497  8162.56   100.0      1.24  ALL
 -   USR 5,407,570,350 6,574,832,225  3960.99    48.5      0.60  USR
 -   OMP    15,783,372    10,975,232  4085.92    50.1    372.29  OMP
 -   MPI       944,200       386,560    92.05     1.1    238.13  MPI
 -   COM       665,210       728,480    23.60     0.3     32.40  COM

 *   ALL    17,390,726    12,138,209  4201.91    51.5    346.17  ALL-FLT
 +   FLT 5,407,531,376 6,574,784,288  3960.65    48.5      0.60  FLT
 -   OMP    15,783,372    10,975,232  4085.92    50.1    372.29  OMP-FLT
 -   MPI       944,200       386,560    92.05     1.1    238.13  MPI-FLT
 *   COM       665,210       728,480    23.60     0.3     32.40  COM-FLT
 *   USR        38,974        47,937     0.34     0.0      7.14  USR-FLT

 +   USR 1,741,005,318 2,110,313,472  1204.11    14.8      0.57  matmul_sub_
 +   USR 1,741,005,318 2,110,313,472   851.97    10.4      0.40  matvec_sub_
 +   USR 1,741,005,318 2,110,313,472  1754.58    21.5      0.83  binvcrhs_
 +   USR    76,367,538    87,475,200    65.93     0.8      0.75  lhsinit_
 +   USR    76,367,538    87,475,200    59.43     0.7      0.68  binvrhs_
 +   USR    56,913,688    68,892,672    24.62     0.3      0.36  exact_solution_
```

- Score report breakdown by region

Filtered routines marked with '+'

BT-MZ filtered summary measurement

```
% cd bin.scorep
% cp ../jobscript/stampede2/scorep.sbatch .
% vim scorep.sbatch

# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum_filter
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=50M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_ENABLE_TRACING=true

# Run the application
ibrun ./bt-mz_${CLASS}.${PROCS}

% sbatch ./scorep.sbatch
```

- Set new experiment directory and re-run measurement with new filter configuration
- Submit job

Score-P filtering

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

% export SCOREP_FILTERING_FILE=\
../config/scorep.filt
```

Region name filter block using wildcards

Apply filter

- Filtering by source file name
  - All regions in files that are excluded by the filter are ignored
- Filtering by region name
  - All regions that are excluded by the filter are ignored
  - Overruled by source file filter for excluded files
- Apply filter by
  - exporting **`SCOREP_FILTERING_FILE`** environment variable
- Apply filter at
  - Run-time
  - Compile-time (GCC-plugin only)
    - Add cmd-line option **`--instrument-filter`**
    - No overhead for filtered regions but recompilation

Source file name filter block

- Keywords
  - Case-sensitive
  - `SCOREP_FILE_NAMES_BEGIN`, `SCOREP_FILE_NAMES_END`
    - Define the source file name filter block
    - Block contains `EXCLUDE`, `INCLUDE` rules
  - `EXCLUDE`, `INCLUDE` rules
    - Followed by one or multiple white-space separated source file names
    - Names can contain bash-like wildcards `*`, `?`, `[]`
    - Unlike bash, `*` may match a string that contains slashes
- `EXCLUDE`, `INCLUDE` rules are applied in sequential order
- Regions in source files that are excluded after all rules are evaluated, get filtered

```
# This is a comment
SCOREP_FILE_NAMES_BEGIN
  # by default, everything is included
  EXCLUDE */foo/bar*
  INCLUDE */filter_test.c
SCOREP_FILE_NAMES_END
```

Region name filter block

- Keywords
  - Case-sensitive
  - SCOREP_REGION_NAMES_BEGIN, SCOREP_REGION_NAMES_END
    - Define the region name filter block
    - Block contains EXCLUDE, INCLUDE rules
  - EXCLUDE, INCLUDE rules
    - Followed by one or multiple white-space separated region names
    - Names can contain bash-like wildcards *, ?, []
- EXCLUDE, INCLUDE rules are applied in sequential order
- Regions that are excluded after all rules are evaluated, get filtered

```
# This is a comment
SCOREP_REGION_NAMES_BEGIN
  # by default, everything is included
  EXCLUDE *
  INCLUDE bar foo
          baz
          main
SCOREP_REGION_NAMES_END
```

Region name filter block, mangling

- Name mangling
  - Filtering based on names seen by the measurement system
    - Dependent on compiler
    - Actual name may be mangled
- `scorep-score` names as starting point (e.g. `matvec_sub_`)
  - Use `*` for Fortran trailing underscore(s) for portability
  - Use `?` and `*` as needed for full signatures or overloading

```
void bar(int* a) {
    *a++;
}
int main() {
    int i = 42;
    bar(&i);
    return 0;
}
```

```
# filter bar:
# for gcc-plugin, scorep-score
# displays 'void bar(int*)',
# other compilers may differ

SCOREP_REGION_NAMES_BEGIN
  EXCLUDE void?bar(int?)
SCOREP_REGION_NAMES_END
```

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
- 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