

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

VI-HPS Team



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 of 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:

Estimated requirements for largest trace buffer (max_buf):

Estimated memory requirements (SCOREP_TOTAL_MEMORY):

(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	10,791,335,059	6,589,342,123	2360.04	100.0	0.36	ALL
	USR	10,754,591,276	6,574,805,745	926.34	39.3	0.14	USR
	OMP	34,990,128	13,667,328	1241.50	52.6	90.84	OMP
	COM	1,178,450	725,200	1.97	0.1	2.71	COM
	MPI	616,168	143,834	190.23	8.1	1322.55	MPI
	SCOREP	41	16	0.01	0.0	372.15	SCOREP

160GB

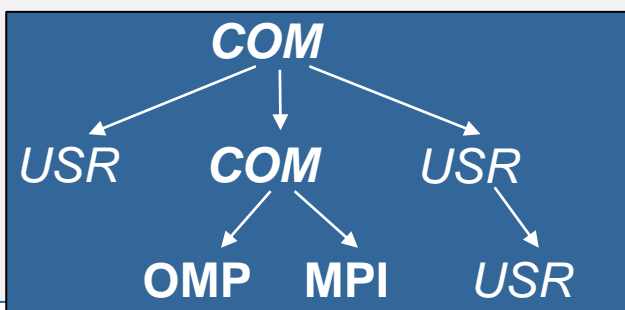
11GB

11GB

- Report scoring as textual output

160 GB total memory
11 GB per rank!

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



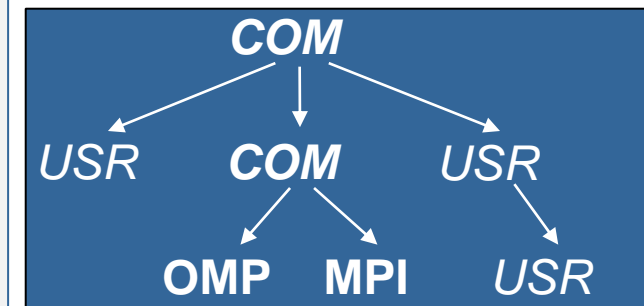
BT-MZ summary analysis report breakdown

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

```
[...]
[...]
```

flt	type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
	ALL	10,791,335,059	6,589,342,123	2360.04	100.0	0.36	ALL
	USR	10,754,591,276	6,574,805,745	926.34	39.3	0.14	USR
	OMP	34,990,128	13,667,328	1241.50	52.6	90.84	OMP
	COM	1,178,450	725,200	1.97	0.1	2.71	COM
	MPI	616,168	143,834	190.23	8.1	1322.55	MPI
	SCOREP	41	16	0.01	0.0	372.15	SCOREP

USR	3,454,903,374	2,110,313,472	373.15	15.8	0.18	binvrhs_
USR	3,454,903,374	2,110,313,472	218.75	9.3	0.10	matvec_sub_
USR	3,454,903,374	2,110,313,472	303.12	12.8	0.14	matmul_sub_
USR	149,170,944	87,475,200	14.95	0.6	0.17	lhsinit_
USR	149,170,944	87,475,200	9.69	0.4	0.11	binvrhs_
USR	112,148,088	68,892,672	6.69	0.3	0.10	exact_solution



More than
10 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 ~11 GB per rank
 - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
 - 99.5% 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 39% 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:
Estimated requirements for largest trace buffer (max_buf):
Estimated memory requirements (SCOREP_TOTAL_MEMORY):
(hint: When tracing set SCOREP_TOTAL_MEMORY=97MB to avoid
intermediate flushes or reduce requirements using
USR regions filters.)
```

1381MB
87MB
97MB

- Report scoring with prospective filter listing 7 USR regions

1.4 GB of memory in total,
87 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
                            visits time[us]
-        ALL 10,791,335,059 6,589,342,123 2360.04 100.0 0.36 ALL
-        USR 10,754,591,276 6,574,805,745 926.34 39.3 0.14 USR
-        OMP 34,990,128 13,667,328 1241.50 52.6 90.84 OMP
-        COM 1,178,450 725,200 1.97 0.1 2.71 COM
-        MPI 616,168 143,834 190.23 8.1 1322.55 MPI
-        SCOREP 41 16 0.01 0.0 372.15 SCOREP

*        ALL 36,820,329 14,558,235 1433.71 60.7 98.48 ALL-FLT
+        FLT 10,754,555,760 6,574,783,888 926.33 39.3 0.14 FLT
-        OMP 34,990,128 13,667,328 1241.50 52.6 90.84 OMP-FLT
*        COM 1,178,450 725,200 1.97 0.1 2.71 COM-FLT
-        MPI 616,168 143,834 190.23 8.1 1322.55 MPI-FLT
*        USR 35,542 21,857 0.01 0.0 0.28 USR-FLT
-        SCOREP 41 16 0.01 0.0 372.15 SCOREP-FLT

+        USR 3,454,903,374 2,110,313,472 373.15 15.8 0.18 binvcrhs_
+        USR 3,454,903,374 2,110,313,472 218.75 9.3 0.10 matvec_sub_
+        USR 3,454,903,374 2,110,313,472 303.12 12.8 0.14 matmul_sub_
+        USR 149,170,944 87,475,200 14.95 0.6 0.17 lhsinit_
+        USR 149,170,944 87,475,200 9.69 0.4 0.11 binvrhs_
+        USR 112,148,088 68,892,672 6.69 0.3 0.10 exact_solution
```

- Score report breakdown by region (w/o additional metrics)

Filtered routines marked with '+'

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, Intel in 7.0 release)
 - 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
 - Use `\` to escape special characters

```
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