

Score-P instrumentation and measurement infrastructure

Demo/Hands-on: Filtering & optimized measurement



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 analysis & report examination

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	10,645,054,457	1,625,436,293	1541.75	100.0	0.95	ALL
	USR	10,605,417,964	1,621,707,837	355.03	23.0	0.22	USR
	OMP	38,329,952	3,539,200	1066.95	69.2	301.47	OMP
	COM	1,182,792	181,968	116.53	7.6	640.37	COM
	MPI	123,708	7,284	3.23	0.2	443.64	MPI
	SCOREP	41	4	0.00	0.0	59.93	SCOREP

40GB

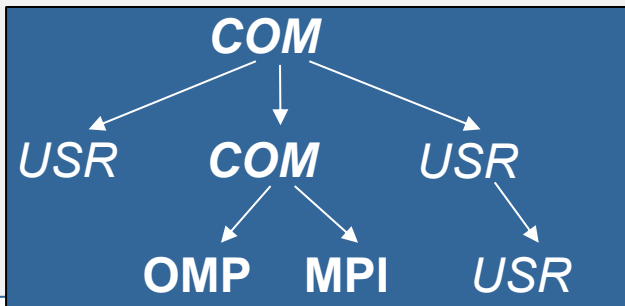
10GB

10GB

- Report scoring as textual output

~40 GB total memory
~10 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



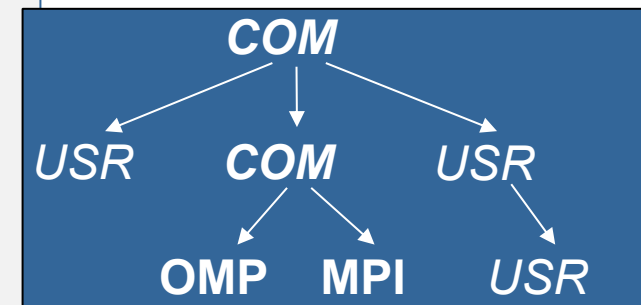
BT-MZ summary analysis report breakdown

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

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

flt	type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
	ALL	10,645,054,457	1,625,436,293	1541.75	100.0	0.95	ALL
	USR	10,605,417,964	1,621,707,837	355.03	23.0	0.22	USR
	OMP	38,329,952	3,539,200	1066.95	69.2	301.47	OMP
	COM	1,182,792	181,968	116.53	7.6	640.37	COM
	MPI	123,708	7,284	3.23	0.2	443.64	MPI
	SCOREP	41	4	0.00	0.0	59.93	SCOREP

USR	3,421,305,420	522,844,416	143.83	9.3	0.28	binvrhs_
USR	3,421,305,420	522,844,416	117.87	7.6	0.23	matvec_sub_
USR	3,421,305,420	522,844,416	80.01	5.2	0.15	matmul_sub_
USR	150,937,332	22,692,096	6.77	0.4	0.30	lhsinit_
USR	150,937,332	22,692,096	4.33	0.3	0.19	binvrhs_
USR	50,726,676	7,779,336	1.45	0.1	0.19	exact_solution_



~9.8 GB just for these
6 regions

BT-MZ summary analysis score

- Summary measurement analysis score reveals
 - Total size of event trace would be ~40 GB
 - Maximum trace buffer size would be ~9.8 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 23% 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=103MB to avoid
intermediate flushes or reduce requirements using
USR regions filters.)
```

362MB
91MB
103MB

- Report scoring with prospective filter listing 7 USR regions

362 MB of memory in total,
103 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/ visit[us]	region
-	ALL	10,645,054,457	1,625,436,293	1541.75	100.0	0.95	ALL
-	USR	10,605,417,964	1,621,707,837	355.03	23.0	0.22	USR
-	OMP	38,329,952	3,539,200	1066.95	69.2	301.47	OMP
-	COM	1,182,792	181,968	116.53	7.6	640.37	COM
-	MPI	123,708	7,284	3.23	0.2	443.64	MPI
-	SCOREP	41	4	0.00	0.0	59.93	SCOREP
* (highlighted)	ALL	39,636,543	3,728,461	1187.48	77.0	318.49	ALL-FLT
+	FLT	10,605,417,940	1,621,707,832	354.26	23.0	0.22	FLT
-	OMP	38,329,952	3,539,200	1066.95	69.2	301.47	OMP-FLT
* (highlighted)	COM	1,182,792	181,968	116.53	7.6	640.37	COM
-	MPI	123,708	7,284	3.23	0.2	443.64	MPI
* (highlighted)	USR	50	5	0.77	0.1	154549.52	USR-FLT
-	SCOREP	41	4	0.00	0.0	59.93	SCOREP
+ (highlighted)	USR	3,421,305,420	522,844,416	143.83	9.3	0.28	binvcrhs_
+ (highlighted)	USR	3,421,305,420	522,844,416	117.87	7.6	0.23	matvec_sub_
+ (highlighted)	USR	3,421,305,420	522,844,416	80.01	5.2	0.15	matmul_sub_
+ (highlighted)	USR	150,937,332	22,692,096	6.77	0.4	0.30	lhsinit_
+ (highlighted)	USR	150,937,332	22,692,096	4.33	0.3	0.19	binvrhs_
+ (highlighted)	USR	50,726,676	7,779,336	1.45	0.1	0.19	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
```