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

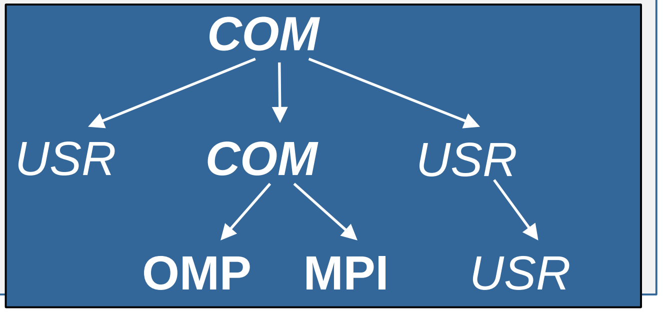
BT-MZ summary analysis result scoring

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

Estimated aggregate size of event trace:                   159GB
Estimated requirements for largest trace buffer (max_buf): 20GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       20GB
(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 21,395,581,557 6,554,106,209 2340.70   100.0           0.36  ALL
         USR 21,309,225,312 6,537,020,537 1098.88    46.9           0.17  USR
         OMP     83,713,600    16,327,168 1218.52    52.1          74.63  OMP
         COM      2,355,080       724,640    2.91     0.1           4.01  COM
         MPI        287,524        33,856   20.39     0.9         602.30  MPI
      SCOREP             41             8    0.00     0.0         593.30  SCOREP
```

~160 GB total memory
~20 GB per rank!

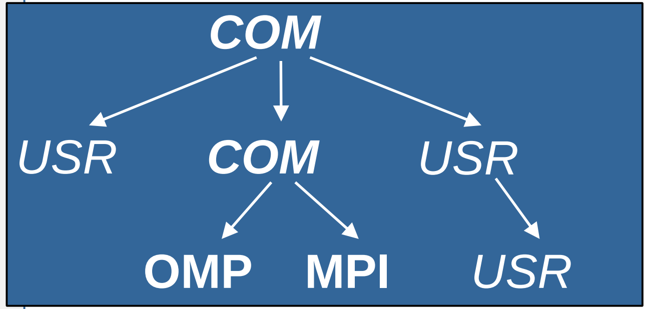


- Report scoring as textual output
- Region/callpath classification
  - **MPI** pure MPI functions
  - **OMP** pure OpenMP regions
  - **USR** user-level computation
  - **COM** “combined” USR+OpenMP/MPI
  - **ALL** aggregate of all region types

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 21,395,581,557 6,554,106,209 2340.70   100.0           0.36  ALL
         USR 21,309,225,312 6,537,020,537 1098.88    46.9           0.17  USR
         OMP     83,713,600    16,327,168 1218.52    52.1          74.63  OMP
         COM      2,355,080       724,640    2.91     0.1           4.01  COM
         MPI        287,524        33,856   20.39     0.9         602.30  MPI
      SCOREP             41             8    0.00     0.0         593.30  SCOREP

         USR  6,883,222,086 2,110,313,472  422.69    18.1           0.20  binvcrhs_
         USR  6,883,222,086 2,110,313,472  284.45    12.2           0.13  matvec_sub_
         USR  6,883,222,086 2,110,313,472  360.89    15.4           0.17  matmul_sub_
         USR    293,617,584    87,475,200   14.16     0.6           0.16  lhsinit_
         USR    293,617,584    87,475,200   11.98     0.5           0.14  binvrhs_
         USR    101,320,128    31,129,600    3.68     0.2           0.12  exact_solution_
```



~20 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 ~20 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 47% 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:                   1621MB
Estimated requirements for largest trace buffer (max_buf): 203MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       215MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=215MB to avoid
       intermediate flushes or reduce requirements using
   USR regions filters.)
```

- Report scoring with prospective filter listing 7 USR regions

1.6 GB of memory in total,
203 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 21,395,581,557 6,554,106,209 2340.70   100.0        0.36  ALL
 -       USR 21,309,225,312 6,537,020,537 1098.88    46.9        0.17  USR
 -       OMP     83,713,600    16,327,168 1218.52    52.1       74.63  OMP
 -       COM      2,355,080       724,640    2.91     0.1        4.01  COM
 -       MPI        287,524        33,856   20.39     0.9      602.30  MPI
 -    SCOREP             41             8    0.00     0.0      593.30  SCOREP

 *       ALL     86,356,295    17,085,681 1242.85    53.1       72.74  ALL-FLT
 +       FLT 21,309,225,262 6,537,020,528 1097.85    46.9        0.17  FLT
 -       OMP     83,713,600    16,327,168 1218.52    52.1       74.63  OMP-FLT
 *       COM      2,355,080       724,640    2.91     0.1        4.01  COM-FLT
 -       MPI        287,524        33,856   20.39     0.9      602.30  MPI-FLT
 *       USR             50             9    1.03     0.0 114496.02  USR-FLT
 -    SCOREP             41             8    0.00     0.0      593.30  SCOREP-FLT

 +       USR  6,883,222,086 2,110,313,472  422.69    18.1        0.20  binvcrhs_
 +       USR  6,883,222,086 2,110,313,472  284.45    12.2        0.13  matvec_sub_
 +       USR  6,883,222,086 2,110,313,472  360.89    15.4        0.17  matmul_sub_
 +       USR    293,617,584    87,475,200   14.16     0.6        0.16  lhsinit_
 +       USR    293,617,584    87,475,200   11.98     0.5        0.14  binvrhs_
 +       USR    101,320,128    31,129,600    3.68     0.2        0.12  exact_solution_
```

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

Filtered routines marked with '+'

BT-MZ filtered summary measurement

```
% cd bin.scorep
% cp ../jobscript/dine/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=100M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...
#export SCOREP_ENABLE_TRACING=true

set -x
export OMP_NUM_THREADS=6
time -p mpiexec -np 8 ./bt-mz_C.8

% 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, 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
```