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

VI-HPS Team

Score-P
Scalable performance measurement infrastructure for parallel codes

# 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
  - but there wasn't much GPU-related performance information
  - and 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 customisation & 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

# Mastering heterogeneous applications

- Record CUDA application events and device activities

```
% export SCOREP_CUDA_ENABLE=default
```

- Record OpenCL application events and device activities

```
% export SCOREP_OPENCL_ENABLE=api,kernel
```

- Record OpenACC application events

```
% export SCOREP_OPENACC_ENABLE=yes
```

  - Can be combined with CUDA if it is a NVIDIA device

```
% export SCOREP_CUDA_ENABLE=kernel
```

Up to now:
using default values

For all available options check:
*scorep-info config-vars --full*

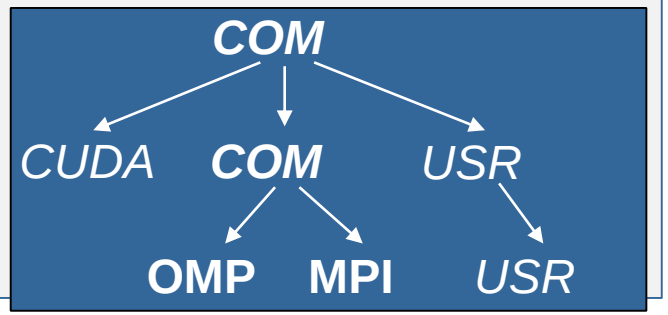
# TeaLeaf CUDA extended summary measurement

```
% edit scorep.sbatch
% cat scorep.sbatch

# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep-tea_leaf-8.extended
export SCOREP_CUDA_ENABLE=default,driver,sync
export SCOREP_CUDA_BUFFER=48MB
#export SCOREP_FILTERING_FILE=../config/scorep.filt

# Run the application
srun ./tea_leaf



% sbatch scorep.sbatch
```

- Set new experiment directory and re-run measurement with extended CUDA event configuration
- Submit job

# TeaLeaf summary analysis result scoring

```
% scorep-score scorep-tea_leaf-8.extended/profile.cubex

Estimated aggregate size of event trace:                   2132MB
Estimated requirements for largest trace buffer (max_buf): 312MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       316MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=316MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type  max_buf[B]     visits time[s] time[%] time/visit[us]  region
         ALL 326,402,801 79,047,257  563.08   100.0           7.12  ALL
         USR 139,104,758 36,515,070    8.34     1.5           0.23  USR
        CUDA 100,904,778 26,426,831  504.82    89.7          19.10  CUDA
         COM  36,088,754  9,637,220    8.70     1.5           0.90  COM
         MPI  32,385,584  3,391,032   41.23     7.3          12.16  MPI
      SCOREP  17,918,927  3,077,104    0.00     0.0           0.00  SCOREP
```




- Report scoring as textual output

2.1 GB total memory
316MB per MPI process

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

# Score-P filtering: Automatic Generation of Filter Files

- **Basic usage**: `scorep-score -g` default heuristic targets:
  - Buffer usage: relevancy
  - Time per visits: overhead
- Creates annotated filter file:
  - `initial_scorep.filter`
  - Repeated calls create backups
  - Usage with `-f <file>` results in inclusion
- **Objective**:
  - Starting point for filtering
  - Syntax introduction

```
-g [<list>]
Generation of an initial filter file with the name
'initial_scorep.filter'. A valid parameter list has the form
KEY=VALUE[,KEY=VALUE]*. By default, uses the following control
parameters:

    `bufferpercent=1,timepervisit=1`

A region is included in the filter file (i.e., excluded from
measurement) if it matches all of the given conditions, with the
following keys:
- `bufferpercent`       : estimated memory requirements exceed the
                          given threshold in percent of the total
                          estimated trace buffer requirements
- `bufferabsolute`      : estimated memory requirements exceed
                          the given absolute threshold in MB
- `visits`              : number of visits exceeds the given
                          threshold
- `timepervisit`        : time per visit value is below the given
                          threshold in microseconds
- `type`                : region type matches the given value
                          (allowed: 'usr', 'com', 'both')
```

# Score-P filtering: Automatic Generation of Filter Files

```
. . .
#
# Generated with the following parameters:
#  - A region has to use at least 1% of the estimated trace buffer.
#  - A region has to have a time/visits value of less than 1 us.
#  - A region that is of type USR.
#
# The file contains comments for each region providing additional information
# regarding the respective region.
# The common prefix for the files is:
# '/'
#
# Please refer to the Score-P user guide for more options on filtering.
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    # type=USR max_buf= 63,576,240 visits=16,628,248, time=   2.17s (  0.4%), time/visit=    0.13us
    # name='bool std::operator< <char, std::char_traits<char>, std::allocator<char> >(std::__cxx11:: [...]
    # file='p/software/juwelsbooster/stages/2022/software/GCCcore/11.2.0/include/c++/11.2.0/bits/ [...]
    MANGLED _ZStltIcSt11char_traitsIcESaIcEEbRKNSt7__cxx1112basic_stringIT_T0_T1_EESA_
. . .

SCOREP_REGION_NAMES_END
```

Introductory preamble

Used generation parameters ( either default or explicit )

Common path prefix to simplify file paths in entries

Annotated entry with comments containing `scorep-score` stats

Additional entries

# TeaLeaf summary analysis report filtering – Preview mode

```
% scorep-score -r -f ../config/scorep.filter \
      scorep-tea_leaf-8.extended/profile.cubex
Estimated aggregate size of event trace:                   1186MB
Estimated requirements for largest trace buffer (max_buf): 173MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       177MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=177MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type  max_buf[B]     visits time[s] time[%] time/visit[us]  region
 -       ALL 326,402,801 79,047,257  563.08   100.0           7.12  ALL
 -       USR 139,104,758 36,515,070    8.34     1.5           0.23  USR
 -      CUDA 100,904,778 26,426,831  504.82    89.7          19.10  CUDA
 -       COM  36,088,754  9,637,220    8.70     1.5           0.90  COM
 -       MPI  32,385,584  3,391,032   41.23     7.3          12.16  MPI
 -    SCOREP  17,918,927  3,077,104    0.00     0.0           0.00  SCOREP

 *       ALL 180,590,173 40,887,545  554.22    98.4          13.55  ALL-FLT
 +       FLT 145,812,628 38,159,712    8.86     1.6           0.23  FLT
 *      CUDA 100,904,778 26,426,831  504.82    89.7          19.10  CUDA-FLT
 -       MPI  32,385,584  3,391,032   41.23     7.3          12.16  MPI-FLT
 *       COM  27,912,950  7,540,860    7.40     1.3           0.98  COM-FLT
 -    SCOREP  17,918,927  3,077,104    0.00     0.0           0.00  SCOREP-FLT
 *       USR   1,469,130    451,718    0.77     0.1           1.71  USR-FLT

 +       USR  63,576,240 16,628,248    2.17     0.4           0.13  bool std::operator< < [...]
 -      CUDA  35,739,496  9,310,175    5.47     1.0           0.59  cuPointerGetAttributes
 -    SCOREP  17,918,927  3,077,104    0.00     0.0           0.00  tea_leaf
 +       USR  16,354,962  4,194,104    0.58     0.1           0.14  std::map<int, dim3, [...]
 +       USR  16,354,884  4,194,080    0.61     0.1           0.15  std::_Rb_tree<int, std::pair […]
```

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

Filtered routines marked with '+'

region names with full signatures

# TeaLeaf summary analysis report filtering – Preview incl. Counters

```
% scorep-score -f ../config/scorep.filter -c 2 \
      scorep-tea_leaf-8.extended/profile.cubex

Estimated aggregate size of event trace:                   3486MB
Estimated requirements for largest trace buffer (max_buf): 506MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       510MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=510MB to avoid
intermediate flushes or reduce requirements using USR regions filters.)

flt     type    max_buf[B]     visits time[s] time[%] time/visit[us]  region
 -       ALL 1,006,819,003 79,047,257  563.08   100.0           7.12  ALL
 -       USR   454,765,555 36,515,070    8.34     1.5           0.23  USR
 -      CUDA   329,881,005 26,426,831  504.82    89.7          19.10  CUDA
 -       COM   117,982,465  9,637,220    8.70     1.5           0.90  COM
 -       MPI    60,485,278  3,391,032   41.23     7.3          12.16  MPI
 -    SCOREP    43,704,700  3,077,104    0.00     0.0           0.00  SCOREP

 *       ALL   530,123,873 40,887,545  554.22    98.4          13.55  ALL-FLT
 +       FLT   476,695,130 38,159,712    8.86     1.6           0.23  FLT
 *      CUDA   329,881,005 26,426,831  504.82    89.7          19.10  CUDA-FLT
 *       COM    91,253,875  7,540,860    7.40     1.3           0.98  COM-FLT
 -       MPI    60,485,278  3,391,032   41.23     7.3          12.16  MPI-FLT
 -    SCOREP    43,704,700  3,077,104    0.00     0.0           0.00  SCOREP-FLT
 *       USR     4,802,925    451,718    0.77     0.1           1.71  USR-FLT
```

- Report scoring with prospective filter

3.5 GB of memory in total, 506 MB per rank!

(Including 2 metric values)

# TeaLeaf filtered summary measurement

```
% edit scorep.sbatch
% cat scorep.sbatch

# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_tea_leaf_sum.filtered
export SCOREP_CUDA_ENABLE=default,driver,sync
export SCOREP_CUDA_BUFFER=48MB
export SCOREP_FILTERING_FILE=../config/scorep.filt

# Run the application
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
mpirun ./tea_leaf



% sbatch scorep.sbatch
```

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

# Score-P filtering

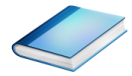
```
% cat ../config/scorep.filt
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    binvcrhs*
    matmul_sub*
    matvec_sub*
    exact_solution*
    binvrhs*
    lhs*init*
    timer_*
SCOREP_REGION_NAMES_END

% 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, Intel compiler)
    - Add cmd-line option **`--instrument-filter`**
    - No overhead for filtered regions but recompilation

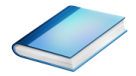
# Source file name filter block



- 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
```

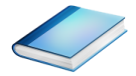
# Region name filter block



- 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