Score-P – A Joint Performance Measurement Run-Time Infrastructure for 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
  - 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

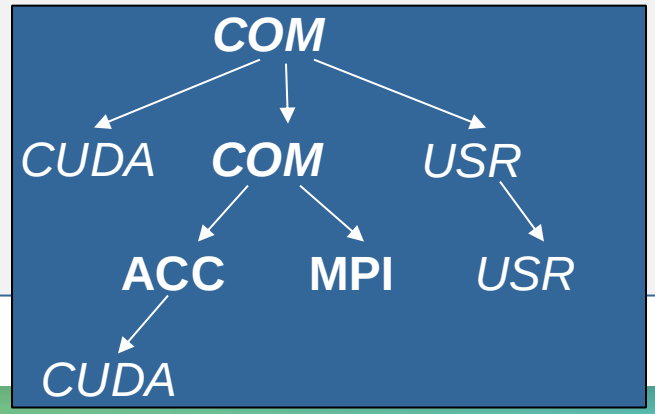
- 1.0 Program instrumentation
- 1.1 Summary measurement collection
- 1.2 Summary analysis report examination

- 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

CloverLeaf_OpenACC summary analysis result scoring

```
% scorep-score scorep_clover_leaf_8_sum/profile.cubex

Estimated aggregate size of event trace:                   89MB
Estimated requirements for largest trace buffer (max_buf): 13MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       17MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=17MB 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 13,279,285 2,553,962  391.59   100.0         153.33  ALL
        CUDA  9,680,284 1,626,386  318.67    81.4         195.94  CUDA
     OPENACC  2,684,682   737,572    0.45     0.1           0.61  OPENACC
         COM    566,514   151,964   65.39    16.7         430.28  COM
         MPI    328,420    31,904    7.05     1.8         221.05  MPI
         USR     23,920     6,128    0.03     0.0           4.24  USR
      SCOREP         41         8    0.00     0.0         466.68  SCOREP
```



- Report scoring as textual output

89 MB total memory
17 MB max per MPI process

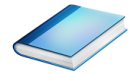
- Region/callpath classification
  - **MPI** pure MPI functions
  - **OMP** pure OpenMP regions
  - **OPENACC** pure OpenACC regions
  - **CUDA** API regions & kernels
  - **USR** user source routines + API
  - **COM** "combined" USR+MPI/ACC/etc
  - **SCOREP** measurement library
  - **ALL** aggregate of all region types

CloverLeaf_OpenACC summary analysis detailed report

```
% scorep-score -r scorep_clover_leaf_8_sum/profile.cubex

flt     type max_buf[B]    visits time[s] time[%] time/visit[us]  region
        ALL 13,279,285 2,553,962  391.59   100.0         153.33  ALL
       CUDA  9,680,284 1,626,386  318.67    81.4         195.94  CUDA
    OPENACC  2,684,682   737,572    0.45     0.1           0.61  OPENACC
        COM    566,514   151,964   65.39    16.7         430.28  COM
        MPI    328,420    31,904    7.05     1.8         221.05  MPI
        USR     23,920     6,128    0.03     0.0           4.24  USR
     SCOREP         41         8    0.00     0.0         466.68  SCOREP

       CUDA  3,807,908   927,341    0.36     0.1           0.39  cuEventQuery
       CUDA    819,984   128,620    0.44     0.1           3.39  cuLaunchKernel
       CUDA    774,332   214,124  117.83    30.1         550.27  cuStreamSynchronize
       CUDA    444,184   128,628   78.49    20.0         610.23  COMPUTE IDLE
       CUDA    406,222     8,044    1.92     0.5         239.15  pack_kernel_module_clover_pack_message_right_170_gpu
       CUDA    406,222     8,044    0.02     0.0           2.55  pack_kernel_module_clover_unpack_message_right_222_gpu
       CUDA    406,222    12,066    0.04     0.0           3.06  pack_kernel_module_clover_pack_message_top_273_gpu
       CUDA    406,222    12,066    0.03     0.0           2.43  pack_kernel_module_clover_unpack_message_top_325_gpu
       CUDA    406,222     8,044    1.92     0.5         239.12  pack_kernel_module_clover_pack_message_left_67_gpu
       CUDA    406,222     8,044    0.02     0.0           2.57  pack_kernel_module_clover_unpack_message_left_120_gpu
       CUDA    406,222    12,066    0.04     0.0           3.01  pack_kernel_module_clover_pack_message_bottom_377_gpu
       CUDA    406,222    12,066    0.03     0.0           2.42  pack_kernel_module_clover_unpack_message_bottom_430_gpu
       CUDA    149,006    37,456    0.02     0.0           0.41  cuEventRecord
       CUDA    145,444    36,360    0.12     0.0           3.35  cuMemcpyAsync
        MPI    139,641    10,460    0.02     0.0           1.83  MPI_Irecv
        MPI    139,641    10,460    0.03     0.0           3.16  MPI_Isend
       CUDA     58,994    16,148    0.01     0.0           0.50  cuPointerGetAttributes
       CUDA     52,924       524    0.00     0.0           2.90  update_halo_kernel_module_update_halo_kernel_151_gpu
```

region names (with full signatures)

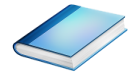
Score-P filtering: Automatic generation of filter files



- **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

CloverLeaf_OpenACC summary analysis filter preview

```
% scorep-score -r -f scorep.filter  scorep_clover_leaf_8_sum/profile.cubex

*        ALL 13,220,291 2,537,814  391.59   100.0         154.30  ALL-FLT
*       CUDA  9,621,290 1,610,238  318.67    81.4         197.90  CUDA-FLT
-    OPENACC  2,684,682   737,572    0.45     0.1           0.61  OPENACC-FLT
*        COM    566,514   151,964   65.39    16.7         430.28  COM-FLT
-        MPI    328,420    31,904    7.05     1.8         221.05  MPI-FLT
+        FLT     58,994    16,148    0.01     0.0           0.50  FLT
*        USR     23,920     6,128    0.03     0.0           4.24  USR-FLT
-     SCOREP         41         8    0.00     0.0         466.68  SCOREP-FLT

-       CUDA  3,807,908   927,341    0.36     0.1           0.39  cuEventQuery
-       CUDA    819,984   128,620    0.44     0.1           3.39  cuLaunchKernel
-       CUDA    774,332   214,124  117.83    30.1         550.27  cuStreamSynchronize
-       CUDA    444,184   128,628   78.49    20.0         610.23  COMPUTE IDLE
-       CUDA    406,222     8,044    1.92     0.5         239.15  pack_kernel_module_clover_pack_message_right_170_gpu
-       CUDA    406,222     8,044    0.02     0.0           2.55  pack_kernel_module_clover_unpack_message_right_222_gpu
-       CUDA    406,222    12,066    0.04     0.0           3.06  pack_kernel_module_clover_pack_message_top_273_gpu
-       CUDA    406,222    12,066    0.03     0.0           2.43  pack_kernel_module_clover_unpack_message_top_325_gpu
...
-       CUDA    149,006    37,456    0.02     0.0           0.41  cuEventRecord
-       CUDA    145,444    36,360    0.12     0.0           3.35  cuMemcpyAsync
-        MPI    139,641    10,460    0.02     0.0           1.83  MPI_Irecv
-        MPI    139,641    10,460    0.03     0.0           3.16  MPI_Isend
+       CUDA     58,994    16,148    0.01     0.0           0.50  cuPointerGetAttributes
-       CUDA     52,924       524    0.00     0.0           2.90  update_halo_kernel_module_update_halo_kernel_151_gpu
-       CUDA     52,924     1,048    0.00     0.0           3.42  update_halo_kernel_module_update_halo_kernel_173_gpu
-       CUDA     52,924       524    0.00     0.0           2.88  update_halo_kernel_module_update_halo_kernel_245_gpu
-       CUDA     52,924     1,048    0.00     0.0           3.45  update_halo_kernel_module_update_halo_kernel_267_gpu
```

max_buf reductions estimated with filter

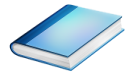
region names matching filter rules marked with '+'

CloverLeaf filtered summary measurement

```
% edit scorep.sbatch
%
# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_clover_leaf_8_sum_filter
export SCOREP_FILTERING_FILE=../config/scorep.filter

# Run the application
srun ./clover_leaf



% sbatch scorep.sbatch
```

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

Score-P filtering



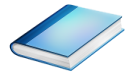
```
% cat ../config/scorep.filter
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    binvcrhs*
    matmul_sub*
    matvec_sub*
    exact_solution*
    binvrhs*
    lhs*init*
    timer_*
SCOREP_REGION_NAMES_END

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

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

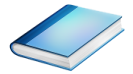
Source file name filter block



- 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
```

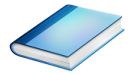
Region name filter block



- 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

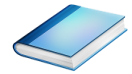


- Name mangling
  - Filtering based on names seen by the measurement system
    - Dependent on compiler
    - Actual name may be mangled
- `scorep-score` names as starting point (e.g. `matvec_sub_`)
  - Use `*` for Fortran trailing underscore(s) for portability
  - Use `?` and `*` as needed for full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

Score-P user instrumentation API



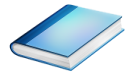
- No replacement for automatic compiler instrumentation

- Can be used to further subdivide functions
  - E.g., multiple loops inside a function
- Can be used to partition application into coarse grain phases
  - E.g., initialization, solver, & finalization

- Enabled with `--user` flag to Score-P instrumenter

- Available for Fortran / C / C++

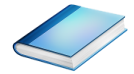
Score-P user instrumentation API (Fortran)



```
#include "scorep/SCOREP_User.inc"

subroutine foo(…)
  ! Declarations
  SCOREP_USER_REGION_DEFINE( solve )

  ! Some code…
  SCOREP_USER_REGION_BEGIN( solve, "<solver>", \
                            SCOREP_USER_REGION_TYPE_LOOP )
  do i=1,100
    [...]
  end do
  SCOREP_USER_REGION_END( solve )
  ! Some more code…
end subroutine
```

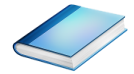
- Requires processing by the C preprocessor
  - For most compilers, this can be automatically achieved by having an uppercase file extension, e.g., `main.F` or `main.F90`

Score-P user instrumentation API (C/C++)



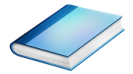
```
#include "scorep/SCOREP_User.h"

void foo()
{
  /* Declarations */
  SCOREP_USER_REGION_DEFINE( solve )

  /* Some code… */
  SCOREP_USER_REGION_BEGIN( solve, "<solver>",
                            SCOREP_USER_REGION_TYPE_LOOP )
  for (i = 0; i < 100; i++)
  {
    [...]
  }
  SCOREP_USER_REGION_END( solve )
  /* Some more code… */
}
```

Score-P user instrumentation API (C++)



```
#include "scorep/SCOREP_User.h"

void foo()
{
  // Declarations

  // Some code…
  {
    SCOREP_USER_REGION( "<solver>",
                        SCOREP_USER_REGION_TYPE_LOOP )
    for (i = 0; i < 100; i++)
    {
      [...]
    }
  }
  // Some more code…
}
```

Score-P measurement control API



- Can be used to temporarily disable measurement for certain intervals
  - Annotation macros ignored by default
  - Enabled with `--user` flag

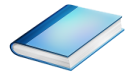
```
#include "scorep/SCOREP_User.inc"

subroutine foo(…)
  ! Some code…
  SCOREP_RECORDING_OFF()
  ! Loop will not be measured
  do i=1,100
    [...]
  end do
  SCOREP_RECORDING_ON()
  ! Some more code…
end subroutine
```

Fortran (requires C preprocessor)

```
#include "scorep/SCOREP_User.h"

void foo(…) {
  /* Some code… */
  SCOREP_RECORDING_OFF()
  /* Loop will not be measured */
  for (i = 0; i < 100; i++) {
    [...]
  }
  SCOREP_RECORDING_ON()
  /* Some more code… */
}
```

C / C++

Enriching measurements with performance counters



- Record metrics from PAPI:

```
% export SCOREP_METRIC_PAPI=PAPI_TOT_CYC
% export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L3_TCM
```

  - Use PAPI tools to get available metrics and valid combinations:

```
% papi_avail
% papi_native_avail
```

- Record metrics from Linux perf:

```
% export SCOREP_METRIC_PERF=cpu-cycles
% export SCOREP_METRIC_PERF_PER_PROCESS=LLC-load-misses
```

  - Use the `perf` tool to get available metrics and valid combinations:

```
% perf list
```

- Write your own metric plugin
  - Repository of available plugins: https://github.com/score-p

Only the master thread records the metric (assuming all threads of the process access the same L3 cache)