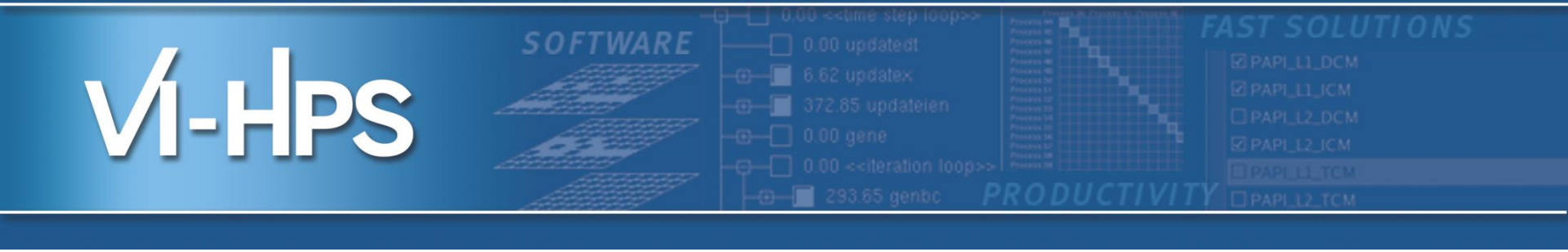


Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

Ronny Tschüter[1)], Bert Wesarg[1)],
Brian Wylie[2)], Markus Geimer[2)]

With contributions from
Andreas Knüpfer[1)] and Christian Rössel[2)]

[1)]ZIH TU Dresden , [2)]FZ Jülich

Fragmentation of Tools Landscape

- Several performance tools co-exist
- Separate measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
- Limited or expensive interoperability
- Complications for user experience, support, training

SILC Project Idea

- Start a community effort for a common infrastructure
  - Score-P instrumentation and measurement system
  - Common data formats OTF2 and CUBE4
- Developer perspective:
  - Save manpower by sharing development resources
  - Invest in new analysis functionality and scalability
  - Save efforts for maintenance, testing, porting, support, training
- User perspective:
  - Single learning curve
  - Single installation, fewer version updates
  - Interoperability and data exchange
- SILC project funded by BMBF
- Close collaboration PRIMA project funded by DOE

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

Score-P Functionality

- Provide typical functionality for HPC performance tools
- Support all fundamental concepts of partner's tools

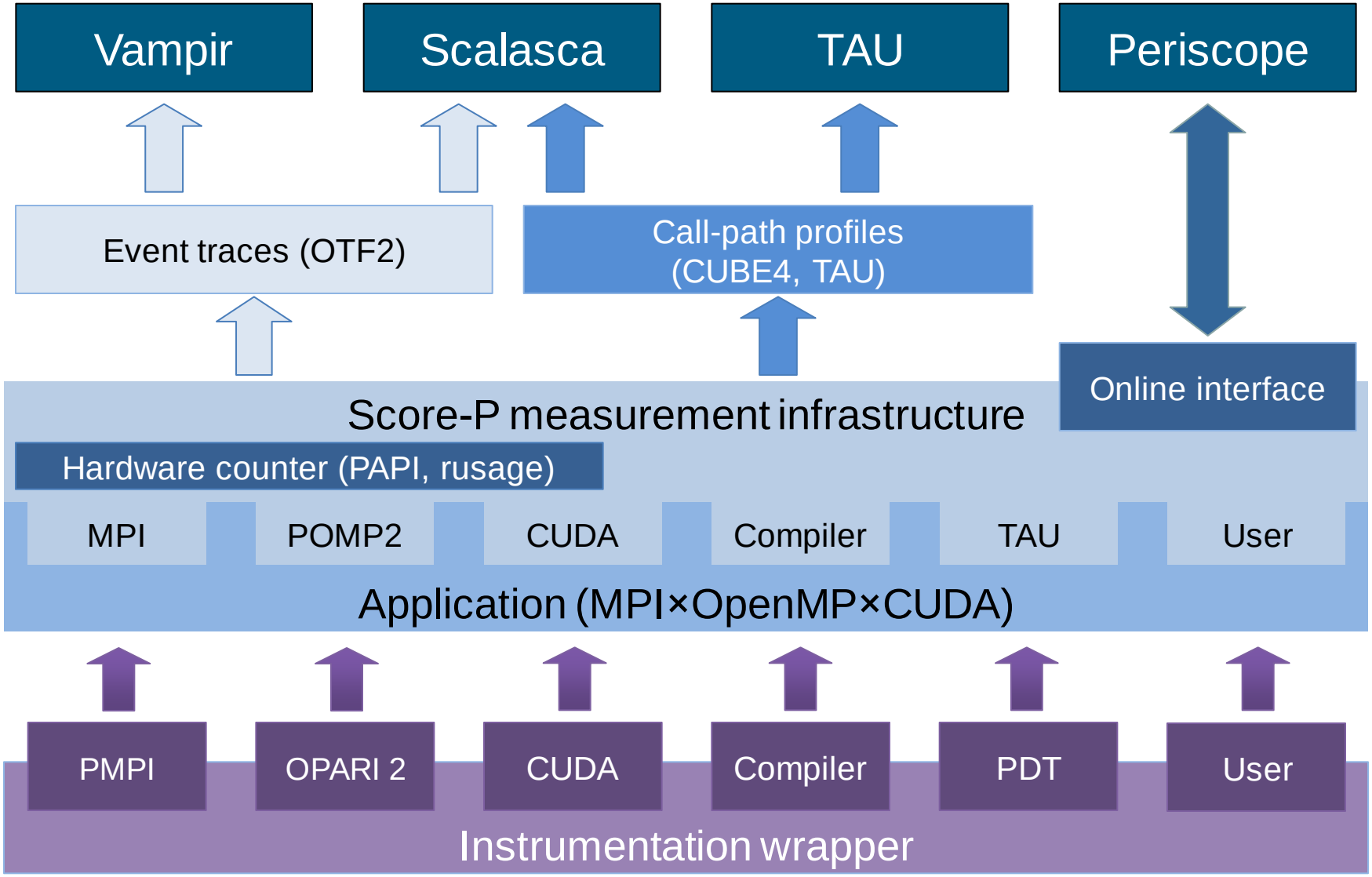
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI, OpenMP, and hybrid parallelism (and serial)
- Enhanced functionality (OpenMP 3.0, CUDA, highly scalable I/O)

Design Goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation, later also sampling
  - Recording time, visits, communication data, hardware counters
  - Access and reconfiguration also at runtime
  - Support for MPI, OpenMP, basic CUDA, and all combinations
    - Later also OpenCL/HMPP/PTHREAD/…
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Easy and uniform installation through UNITE framework
  - Robustness
  - Open Source: New BSD License

Score-P Architecture

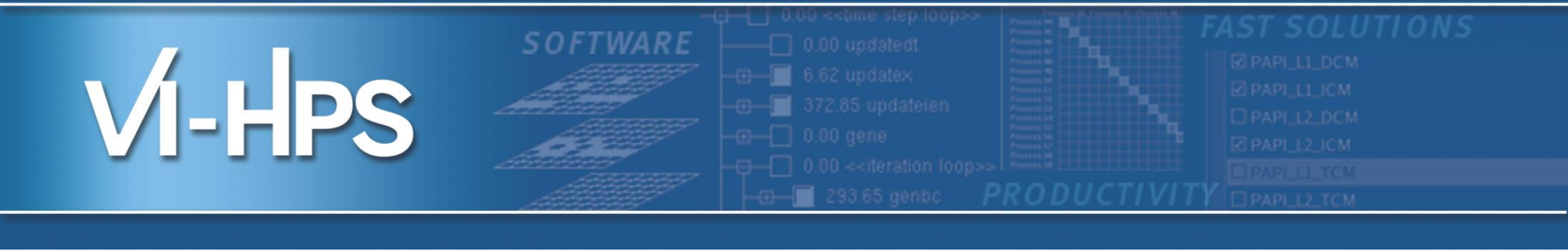


Future Features and Management

- Scalability to maximum available CPU core count
- Support for OpenCL, HMPP, PTHREAD
- Support for sampling, binary instrumentation
- Support for new programming models, e.g., PGAS
- Support for new architectures

- Ensure a single official release version at all times which will always work with the tools
- Allow experimental versions for new features or research

- Commitment to joint long-term cooperation



Score-P hands-on: NPB-MZ-MPI / BT

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

- Load modules

```
% module load UNITE
UNITE loaded
% module load scorep/1.2.1
scorep/1.2.1 loaded

% module load cube/4.2
cube/4.2 loaded
% module load tau/2.22.3b4-bgq-ibm-papi
tau/2.22.3b4-bgq-ibm-papi loaded

% module load papi/5.1.1
papi/5.1.1 loaded
```

- Change to directory containing NPB BT-MZ sources

- Edit config/make.def to adjust build configuration
  - Modify specification of compiler/linker: MPIF77

```
#           SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#---------------------------------------------------------------------
# Items in this file may need to be changed for each platform.
#---------------------------------------------------------------------
...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
#MPIF77 = mpixlf77_r

# Alternative variants to perform instrumentation
...
MPIF77 = scorep --user mpixlf77_r

# This links MPI Fortran programs; usually the same as ${MPIF77}
FLINK   = $(MPIF77)
...
```

Uncomment the Score-P compiler wrapper specification

NPB-MZ-MPI / BT Instrumented Build

- Return to root directory and clean-up

```
% make clean
```

- Re-build executable using Score-P instrumenter

```
% make bt-mz CLASS=C NPROCS=32
cd BT-MZ; make CLASS=C NPROCS=32 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c
../sys/setparams bt-mz 32 C
scorep --user mpixlf77_r -c -O3 -qsmp=omp bt.f
[...]
cd ../common; scorep --user mpixlf77_r -c -O3 -qsmp=omp timers.f
scorep --user mpixlf77_r -O3 -qsmp=omp -o ../bin.scorep/bt-mz_C.32 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin.scorep/bt-mz_C.32
make: Leaving directory 'BT-MZ'
```

Measurement Configuration: scorep-info

- Score-P measurements are configured via environment variables:

```
% scorep-info config-vars --full
SCOREP_ENABLE_PROFILING
  Description: Enable profiling
 [...]
SCOREP_ENABLE_TRACING
  Description: Enable tracing
 [...]
SCOREP_TOTAL_MEMORY
  Description: Total memory in bytes for the measurement system
 [...]
SCOREP_EXPERIMENT_DIRECTORY
  Description: Name of the experiment directory
 [...]
SCOREP_FILTERING_FILE
  Description: A file name which contain the filter rules
 [...]
SCOREP_METRIC_PAPI
  Description: PAPI metric names to measure
 [...]
SCOREP_METRIC_RUSAGE
  Description: Resource usage metric names to measure
 [... More configuration variables ...]
```

Summary Measurement Collection

- Change to the directory containing the new executable and run it

```
% cd bin.scorep
% cp ../jobscript/juqueen/scorep.ll .
% vim scorep.ll
% llsubmit scorep.ll
% cat mzmpibt.<id>
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:  16 x 16
Iterations: 200   dt:  0.000100
Number of active processes:  32
Total number of threads:     256 ( 8.0 threads/process)

Time step 1
Time step 20
[...]
Time step 180
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 135.68
```

BT-MZ Summary Analysis Report Examination

- Creates experiment directory ./scorep_bt-mz_C_32x8_sum containing
  - a record of the measurement configuration (scorep.cfg)
  - the analysis report that was collated after measurement (profile.cubex)

```
% ls
...  scorep_bt-mz_C_32x8_sum
% ls scorep_bt-mz_C_32x8_sum
profile.cubex  scorep.cfg
```

- Interactive exploration with CUBE / ParaProf

```
% cube scorep_bt-mz_C_32x8_sum/profile.cubex

              [CUBE GUI showing summary analysis report]

% paraprof scorep_bt-mz_C_32x8_sum/profile.cubex

          [TAU ParaProf GUI showing summary analysis report]
```

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

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 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

BT-MZ Summary Analysis Result Scoring

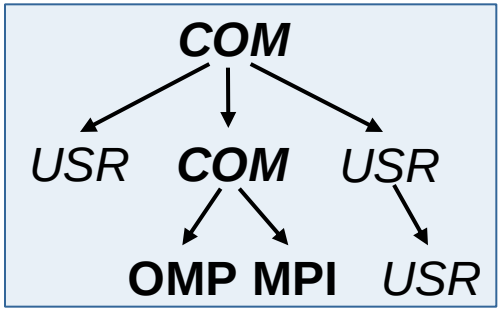
- Report scoring as textual output

```
% scorep-score scorep_bt-mz_C_32x8_sum/profile.cubex
Estimated aggregate size of event trace (total_tbc):      145288373910 bytes
Estimated requirements for largest trace buffer (max_tbc):  4593208808 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid intermediate flushes
 or reduce requirements using file listing names of USR regions to be filtered.)

flt type         max_tbc         time      % region
     ALL      4593208808     33715.36  100.0 ALL
     USR      4575636450     15163.91   45.0 USR
     OMP        21342960     18341.93   54.4 OMP
     MPI          675818       183.24    0.5 MPI
     COM          567270        26.28    0.1 COM
```

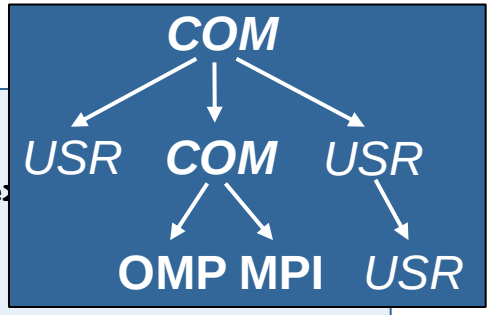
135 GB total memory
4.2 GB per rank!

- Region/callpath classification
  - MPI (pure MPI library functions)
  - OMP (pure OpenMP functions/regions)
  - USR (user-level source local computation)
  - COM ("combined" USR + OpenMP/MPI)
  - ANY/ALL (aggregate of all region types)



BT-MZ Summary Analysis Report Breakdown

- Score report breakdown by region




```
% scorep-score -r scorep_bt-mz_C_32x8_sum/profile.cube
  [...]
flt type          max_tbc          time      % region
     ALL       4593208808      33715.36  100.0 ALL
     USR       4575636450      15163.91   45.0 USR
     OMP         21342960      18341.93   54.4 OMP
                   675818        183.24    0.5 MPI
                   567270         26.28    0.1 COM

     USR       1473158346       4300.73   12.8 matvec_sub
     USR       1473158346       4251.52   12.6 matmul_sub
     USR       1473158346       6004.00   17.8 binvcrhs
     USR         64618686        322.86    1.0 lhsinit
     USR         64618686        175.13    0.5 binvrhs
     USR         48157736        109.50    0.3 exact_solution
     OMP          1331424          5.29    0.0 !$omp parallel @exch_...
     OMP          1331424          5.53    0.0 !$omp parallel @exch_...
     OMP          1331424          5.51    0.0 !$omp parallel @exch_...
  [...]
```

More than 3.9 GB just for these 6 regions

BT-MZ Summary Analysis Score

- Summary measurement analysis score reveals
  - Total size of event trace would be ~135 GB
  - Maximum trace buffer size would be ~4.2 GB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.8% 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 32% 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

- Report scoring with prospective filter listing
  7 USR regions

```
% 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 scorep_bt-mz_C_32x8_sum/profile.cubex
Estimated aggregate size of event trace (total_tbc):       643119574 bytes
Estimated requirements for largest trace buffer (max_tbc):  22589658 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid intermediate flushes
 or reduce requirements using file listing names of USR regions to be filtered.)
```

613 MB of memory in total,
22 MB per rank!

BT-MZ Summary Analysis Report Filtering

- Score report breakdown by region

```
% scorep-score -r -f ../config/scorep.filt \
> scorep_bt-mz_C_32x8_sum/profile.cubex
flt type         max_tbc         time      % region
 *   ALL        22589658     18551.62   55.0 ALL-FLT
 +   FLT      4575603472     15163.74   45.0 FLT
 -   OMP        21342960     18341.93   54.4 OMP-FLT
 -   MPI          675818       183.24    0.5 MPI-FLT
 *   COM          567270        26.28    0.1 COM-FLT
 *   USR           32978         0.17    0.0 USR-FLT

 +   USR      1473158346      4300.73   12.8 matvec_sub
 +   USR      1473158346      4251.52   12.6 matmul_sub
 +   USR      1473158346      6004.00   17.8 binvcrhs
 +   USR        64618686       322.86    1.0 lhsinit
 +   USR        64618686       175.13    0.5 binvrhs
 +   USR        48157736       109.50    0.3 exact_solution
 -   OMP         1331424         5.29    0.0 !$omp parallel @exch_...
 -   OMP         1331424         5.53    0.0 !$omp parallel @exch_...
 -   OMP         1331424         5.51    0.0 !$omp parallel @exch_...
[...]
```

Filtered routines marked with '+'

BT-MZ Filtered Summary Measurement

- Set new experiment directory and re-run measurement with new filter configuration
  - Edit job script

```
% vim scorep.ll
```

  - Adjust configuration

```
...
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_C_32x8_sum_with_filter
export SCOREP_FILTERING_FILE=../config/scorep.filt
...
```

  - Submit job

```
% llsubmit scorep.ll
```

BT-MZ Tuned Summary Analysis Report Score

- Scoring of new analysis report as textual output

```
% scorep-score scorep_bt-mz_C_32x8_sum_with_filter/profile.cubex
Estimated aggregate size of event trace:                   643119574 bytes
Estimated requirements for largest trace buffer (max_tbc):  22589658 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid intermediate flushes
 or reduce requirements using file listing names of USR regions to be filtered.)

flt type         max_tbc         time      % region
     ALL        22589658      6770.21  100.0 ALL
     OMP        21342960      6690.62   98.8 OMP
     MPI          675818        53.03    0.8 MPI
     COM          567270        26.38    0.4 COM
     USR           32978         0.17    0.0 USR
```

- Significant reduction in runtime (measurement overhead)
  - Not only reduced time for USR regions, but MPI/OMP reduced too!
- Further measurement tuning (filtering) may be appropriate
  - e.g., use “timer_*” to filter timer_start_, timer_read_, etc.

Advanced Measurement Configuration: Metrics

- Recording hardware counters via PAPI

```
export SCOREP_METRIC_PAPI=PAPI_L2_TCM,PAPI_FP_OPS
```

- Also possible to record them only per rank

```
export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L3_TCM
```

- Recording operating system resource usage

```
export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss,ru_stime
```

Note: Additional memory is needed to store metric values. Therefore, you may have to adjust SCOREP_TOTAL_MEMORY, for example as reported using “scorep-score –c”

- Available PAPI metrics
  - Preset events: common set of events deemed relevant and useful for application performance tuning
    - Abstraction from specific hardware performance counters, mapping onto available events done by PAPI internally

```
% papi_avail
```

  - Native events: set of all events that are available on the CPU (**platform dependent**)

```
% papi_native_avail
```

Note:
Due to hardware restrictions
- number of concurrently measured events is limited
- there may be unsupported combinations of concurrent events

Advanced Measurement Configuration: Metrics

- Available resource usage metrics

```
% man getrusage
 [... Output ...]

struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size */
    long   ru_ixrss;         /* integral shared memory size */
    long   ru_idrss;         /* integral unshared data size */
    long   ru_isrss;         /* integral unshared stack size */
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
    long   ru_nswap;         /* swaps */
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
    long   ru_msgsnd;        /* IPC messages sent */
    long   ru_msgrcv;        /* IPC messages received */
    long   ru_nsignals;      /* signals received */
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};

 [... More output ...]
```

Note:
(1) Not all fields are maintained on each platform.
(2) Check scope of metrics (per process vs. per thread)

Advanced Measurement Configuration: MPI

- Record only for subset of the MPI functions events

```
export SCOREP_MPI_ENABLE_GROUPS=cg,coll,p2p,xnonblock
```

- All possible sub-groups
  - cg Communicator and group management
  - coll Collective functions
  - env Environmental management
  - err MPI Error handling
  - ext External interface functions
  - io MPI file I/O
  - misc Miscellaneous
  - perf PControl
  - p2p Peer-to-peer communication
  - rma One sided communication
  - spawn Process management
  - topo Topology
  - type MPI datatype functions
  - xnonblock Extended non-blocking events
  - xreqtest Test events for uncompleted requests