TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

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

OCLF – Tools Workshop

19-20 Jan 2022

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Score-P Hands-on: Setup Workspace

Create workspace for workshop

```
% module load scorep
% scorep-info config-summary | grep gcc/
# note gcc module version and load the module
% module load gcc/<version-from-above>
```

Change working directory to workspace:

```
% cd /scratch/ws/0/$USER-nhr-tools
```

Copy workshop material into own workspace:

```
% cp -r /projects/p_nhr_tools/Material .
% cd Material
```

Setup environment (do this every time you login):

```
% source ../prepare.source
```

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Reference Run: NPB-MZ-MPI / BT

ZIH
Center for Information Services & High Performance Computing

Performance Analysis Steps

1. Reference preparation for validation

2. Program instrumentation
3. Summary measurement collection

4. Summary experiment scoring
5. Summary measurement collection with filtering

6. Event trace collection

BT-MZ

The NAS Parallel Benchmark suite (MPI+OpenMP version)

- http://www.nas.nasa.gov/Software/NPB

Benchmark name:

- **bt-mz**, lu-mz, sp-mz

Number of MPI processes:

- NPROCS=**4**

Benchmark class:

- S, W, A, B, **C**, D, E
- CLASS=**C**

Clean environment:

```
% cd BT-MZ
% unset ${!SCOREP_*}
```

BT-MZ / Reference preparation

Build uninstrumented benchmark:

```
% make bt-mz NPROCS=4 CLASS=C
cd BT-MZ; make CLASS=C NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 4 C
[…]
Built executable ../bin/bt-mz_C.4
make: Leaving directory 'BT-MZ‘
% ls bin
bt-mz_C.4
```

BT-MZ / Reference run

Run uninstrumented benchmark:

```
% export OMP_NUM_THREADS=6
% mpirun -n 4 ./bin/bt-mz_C.4
Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:     4

 Use the default load factors with threads
 Total number of threads:     24  (  6.0 threads/process)

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    42.85
 Total processes =                       4
 Total threads   =                      24
 Mop/s total     =                 56642.84
 Mop/s/thread    =                  2360.12
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              29 Sep 2021
```

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Score-P Background

Score-P

Infrastructure for instrumentation and performance measurements

Instrumented application can be used to produce several results:

- Call-path profiling: CUBE4 data format used for data exchange
- Event-based tracing: OTF2 data format used for data exchange

Supported parallel paradigms:

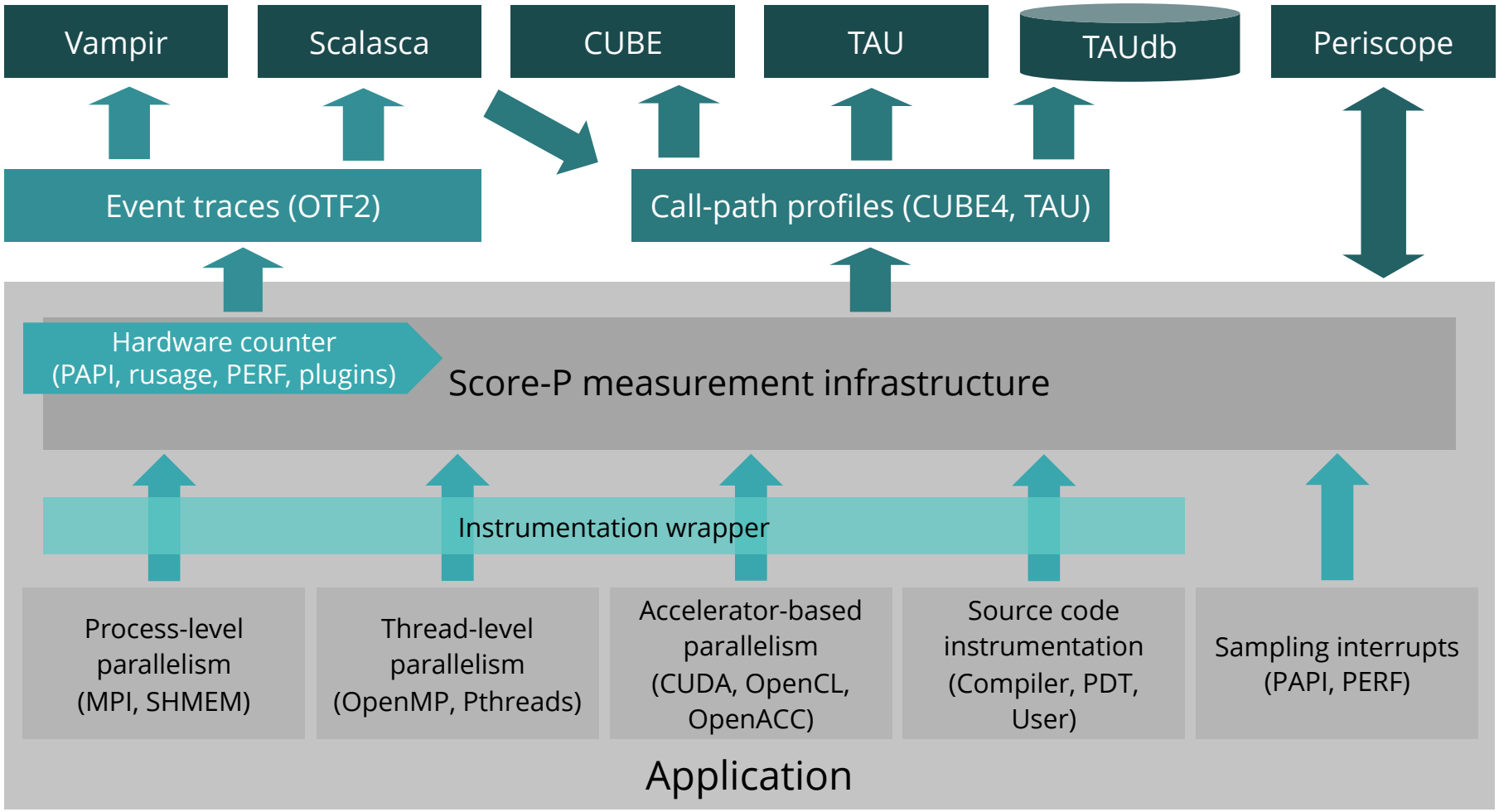
- Multi-process: MPI, SHMEM
- Thread-parallel: OpenMP, Pthreads
- Accelerator-based: CUDA, OpenCL

Open Source; portable and scalable to all major HPC systems

Initial project funded by BMBF

Close collaboration with PRIMA project funded by DOE

Score-P Overview



Broader Score-P/OTF2 Ecosystem Projects

(not presented here)

Score-P Python Bindings:

- Module that allows tracing of python scripts
- https://github.com/score-p/scorep_binding_python

Extra-P

- Automatic performance-modeling tool to identify scalability bugs
- Orchestrates Score-P measurements
- https://github.com/extra-p/extrap

Linux OTF2 Sampling

- Lightweight node-level performance monitoring tool
- https://tu-dresden.de/zih/forschung/projekte/lo2s?set_language=en

OTF(2) Profiler

- Generates profile, JSON, or graphviz files out of OTF2 traces
- https://github.com/score-p/otf2_cli_profile

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Hands-On Instrumentation: NPB-MZ-MPI / BT

Performance Analysis Steps

1. Reference preparation for validation

2. Program instrumentation
3. Summary measurement collection

4. Summary experiment scoring
5. Summary measurement collection with filtering

6. Event trace collection

BT-MZ / Instrumentation

The instrumenter command **`scorep`** is used as a prefix to the usual compile and link commands. For example, instead of

```
mpicc … -o <binary> <source-files…>
```

use the command

```
scorep mpicc … -o <binary> <source-files…>
```

Provided examples can be instrumented by adding `PREP=scorep` to the `make` invocation:

```
% make … PREP=scorep
```

BT-MZ / Instrumentation

Build instrumented benchmark:

```
% make clean
% make bt-mz NPROCS=4 CLASS=C PREP=scorep
cd BT-MZ; make CLASS=C NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 4 C
[…]
Built executable ../bin.scorep/bt-mz_C.4
make: Leaving directory 'BT-MZ‘
% ls bin.scorep
bt-mz_C.4
```

Measurement Configuration: scorep-info

Measurements with Score-P are configured via environmental 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 ...]
```

BT-MZ / Instrumented run

Run uninstrumented benchmark:

```
% export OMP_NUM_THREADS=6
% export SCOREP_EXPERIMENT_DIRECTORY=scorep-bt-mz_4x6_profile
% mpirun -n 4 ./bin.scorep/bt-mz_C.4
Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:     4

 Use the default load factors with threads
 Total number of threads:     24  (  6.0 threads/process)

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                     200
 Time in seconds =                  106.93
 Total processes =                       4
 Total threads   =                      24
 Mop/s total     =                22695.95
 Mop/s/thread    =                  945.66
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              29 Sep 2021
```

BT-MZ / Result examination

Creates experiment directory `./scorep-bt-mz_4x6_profile`

- A manifest, what is included in this experiment directory (`MANIFEST.md`)
- a record of the measurement configuration (`scorep.cfg`)
- the analysis report that was collated after measurement (`profile.cubex`)

```
% ls scorep-bt-mz_4x6_profile
MANIFEST.md  profile.cubex  scorep.cfg
% cat scorep-bt-mz_4x6_profile/MANIFEST.md
% cat scorep-bt-mz_4x6_profile/scorep.cfg
```

Congratulations!?

... but how ***good*** was the measurement?

- The measured execution produced the desired valid result
- however, the execution took rather longer than expected!

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Break

Performance Analysis Steps

1. Reference preparation for validation

2. Program instrumentation
3. Summary measurement collection

4. Summary experiment scoring
5. Summary measurement collection with filtering

6. Event trace collection

BT-MZ / Summary Analysis Result Scoring

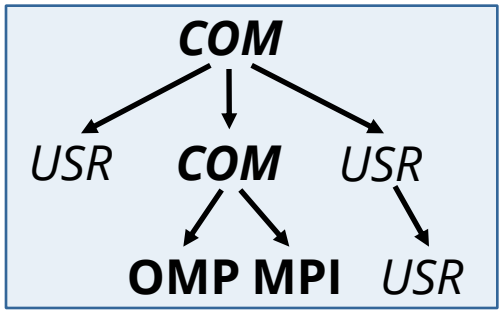
160 GB total memory
41 GB per rank!

Report scoring as textual output

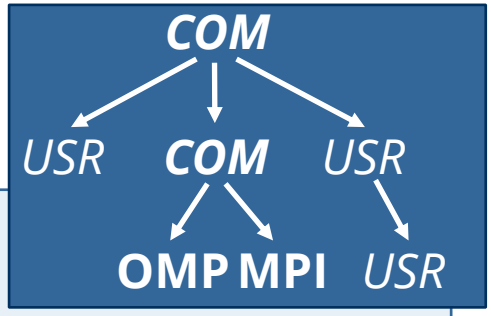
```
% scorep-score scorep-bt-mz_4x6_profile/profile.cubex

Estimated aggregate size of event trace:                   160GB
Estimated requirements for largest trace buffer (max_buf): 41GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       41GB
(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 43,161,138,617 6,591,877,701 2574.27   100.0           0.39  ALL
         USR 42,988,632,934 6,574,788,217 1061.84    41.2           0.16  USR
         OMP    167,683,712    16,359,424 1485.15    57.7          90.78  OMP
         COM      4,697,810       722,740    2.85     0.1           3.94  COM
         MPI        124,120         7,316   24.43     0.9        3338.67  MPI
      SCOREP             41             4    0.00     0.0          43.70  SCOREP
```

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_4x6_profile/profile.cubex


flt     type     max_buf[B]         visits time[s] time[%] time/visit[us]  region
         ALL 43,161,138,617 6,591,877,701 2574.27   100.0           0.39  ALL
         USR 42,988,632,934 6,574,788,217 1061.84    41.2           0.16  USR
         OMP    167,683,712    16,359,424 1485.15    57.7          90.78  OMP
         COM      4,697,810       722,740    2.85     0.1           3.94  COM
         MPI        124,120         7,316   24.43     0.9        3338.67  MPI
      SCOREP             41             4    0.00     0.0          43.70  SCOREP

         USR 13,812,365,034 2,110,313,472  446.77    17.4           0.21  binvcrhs
         USR 13,812,365,034 2,110,313,472  352.08    13.7           0.17  matmul_sub
         USR 13,812,365,034 2,110,313,472  225.19     8.7           0.11  matvec_sub
         USR    596,197,758    87,475,200   19.29     0.7           0.22  lhsinit
         USR    596,197,758    87,475,200   11.49     0.4           0.13  binvrhs
         USR    447,869,968    68,892,672    7.02     0.3           0.10  exact_...
```

BT-MZ / Summary Analysis Report Breakdown

Create filtering file for high-frequent regions

```
% scorep-score -g scorep-bt-mz_4x6_profile/profile.cubex

An initial filter file template has been generated: 'initial_scorep.filter'

To use this file for filtering at run-time, set the respective Score-P variable:

    SCOREP_FILTERING_FILE=initial_scorep.filter

For compile-time filtering 'scorep' has to be provided with the '--instrument-filter' option:

    $ scorep --instrument-filter=initial_scorep.filter

Compile-time filtering depends on support in the used Score-P installation.

The filter file is annotated with comments, please check if the selection is
suitable for your purposes and add or remove functions if needed.
```

BT-MZ / Summary Analysis Report Breakdown

Simulate filtering file

659 MB total memory
177 MB per rank!

```
% scorep-score -f initial_scorep.filter \
 scorep-bt-mz_4x6_profile/profile.cubex

Estimated aggregate size of event trace:                   659MB
Estimated requirements for largest trace buffer (max_buf): 165MB
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 43,161,138,617 6,591,877,701 2574.27   100.0           0.39  ALL
 -       USR 42,988,632,934 6,574,788,217 1061.84    41.2           0.16  USR
 -       OMP    167,683,712    16,359,424 1485.15    57.7          90.78  OMP
 -       COM      4,697,810       722,740    2.85     0.1           3.94  COM
 -       MPI        124,120         7,316   24.43     0.9        3338.67  MPI
 -    SCOREP             41             4    0.00     0.0          43.70  SCOREP

 *       ALL    172,536,441    17,094,213 1512.43    58.8          88.48  ALL-FLT
 +       FLT 42,988,602,202 6,574,783,488 1061.84    41.2           0.16  FLT
 -       OMP    167,683,712    16,359,424 1485.15    57.7          90.78  OMP-FLT
 *       COM      4,697,810       722,740    2.85     0.1           3.94  COM-FLT
 -       MPI        124,120         7,316   24.43     0.9        3338.67  MPI-FLT
 *       USR         30,758         4,729    0.00     0.0           0.33  USR-FLT
 -    SCOREP             41             4    0.00     0.0          43.70  SCOREP-FLT
```

BT-MZ / Filtered run

Run instrumented benchmark with filter:

```
% export OMP_NUM_THREADS=6
% export SCOREP_FILTERING_FILE=initial_scorep.filter
% export SCOREP_EXPERIMENT_DIRECTORY=scorep-bt-mz_4x6_profile_filtered
% mpirun -n 4 ./bin.scorep/bt-mz_C.4
 Iterations: 200    dt:   0.000100
 Number of active processes:     4

 Use the default load factors with threads
 Total number of threads:     24  (  6.0 threads/process)

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    44.71
 Total processes =                        4
 Total threads   =                       24
 Mop/s total     =                 54286.32
 Mop/s/thread    =                  2261.93
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              29 Sep 2021
```

BT-MZ / Filtered run

Verify filtering result:

```
% scorep-score scorep-bt-mz_4x6_filtered/profile.cubex

Estimated aggregate size of event trace:                   659MB
Estimated requirements for largest trace buffer (max_buf): 165MB
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 172,536,441 17,094,213 1057.91   100.0          61.89  ALL
         OMP 167,683,712 16,359,424 1049.95    99.2          64.18  OMP
         COM   4,697,810    722,740    2.65     0.3           3.67  COM
         MPI     124,120      7,316    5.30     0.5         724.87  MPI
         USR      30,758      4,729    0.00     0.0           0.34  USR
      SCOREP          41          4    0.00     0.0          45.60  SCOREP
```

Performance Analysis Steps

1. Reference preparation for validation

2. Program instrumentation
3. Summary measurement collection

4. Summary experiment scoring
5. Summary measurement collection with filtering

6. Event trace collection

BT-MZ / Tracing run

Run instrumented benchmark in trace mode:

```
% export OMP_NUM_THREADS=6
% export SCOREP_ENABLE_TRACING=yes
% export SCOREP_TOTAL_MEMORY=200M
% export SCOREP_FILTERING_FILE=initial_scorep.filter
% export SCOREP_EXPERIMENT_DIRECTORY=scorep-bt-mz_4x6_tracing
% mpirun -n 4 ./bin.scorep/bt-mz_C.4
Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:     4

 Use the default load factors with threads
 Total number of threads:     24  (  6.0 threads/process)

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    44.80
 Total processes =                       4
 Total threads   =                      24
 Mop/s total     =                 54167.91
 Mop/s/thread    =                  2257.00
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              29 Sep 2021
```

BT-MZ / trace result examination

Creates experiment directory `./scorep-bt-mz_4x6_tracing`

- A manifest, what is included in this experiment directory (`MANIFEST.md`)
- a record of the measurement configuration (`scorep.cfg`)
- the trace file collection (`traces.otf2, …`)

```
% ls scorep-bt-mz_4x6_tracing
MANIFEST.md  scorep.cfg  traces/  traces.def  traces.otf2
% cat scorep-bt_mz-4x6-tracing/MANIFEST.md
% cat scorep-bt_mz-4x6-tracing/scorep.cfg
```

Advanced Measurement Configuration: MPI

Record only for subset of the MPI functions events

```
% export SCOREP_MPI_ENABLE_GROUPS=cg,coll,p2p,xnonblock
% mpirun -np 4 <mpi_app>
 [... More application output ...]
```

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

Advanced Measurement Configuration: CUDA

Record CUDA events with the CUPTI interface

```
% export SCOREP_CUDA_ENABLE=gpu,kernel,idle
% <cuda_app>

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

All possible recording types:

runtime	CUDA runtime API
driver	CUDA driver API
gpu	GPU activities
kernel	CUDA kernels
idle	GPU compute idle time
memcpy	CUDA memory copies

Mastering build systems

Hooking up the Score-P instrumenter **`scorep`** into complex build environments like *Autotools* or *CMake* was always challenging

Score-P provides convenience wrapper scripts to simplify this

Autotools and *CMake* need the used compiler already in the *configure step*, but instrumentation should not happen in this step, only in the *build step*

```
% SCOREP_WRAPPER=off \
> cmake .. \
> -DCMAKE_C_COMPILER=scorep-gcc \
> -DCMAKE_CXX_COMPILER=scorep-g++
```

Allows to pass addition options to the Score-P instrumenter and the compiler via environment variables without modifying the *`Makefiles`*

Run **`scorep-wrapper --help`** for a detailed description and the available wrapper scripts of the Score-P installation