TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

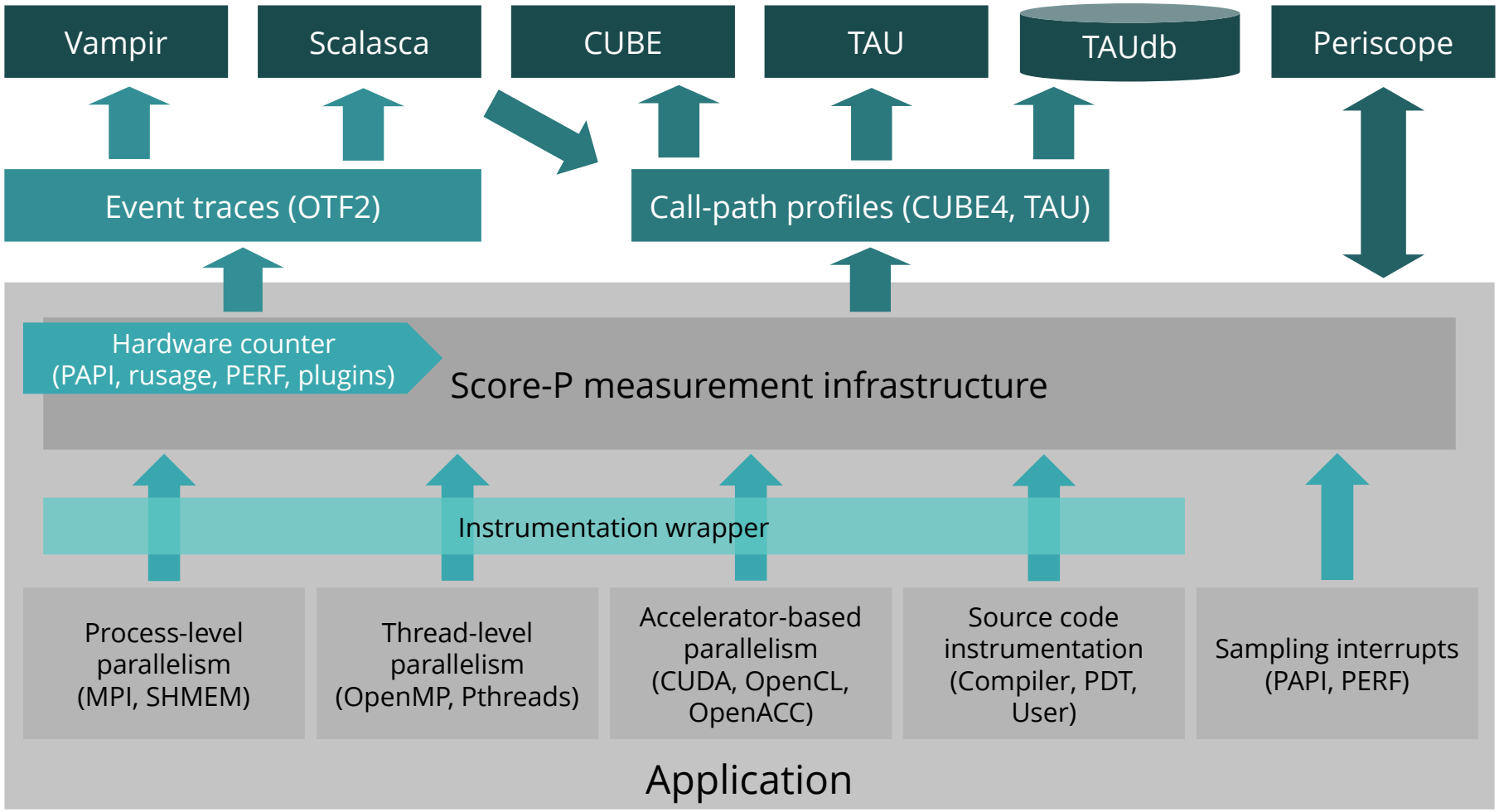
Score-P – A Joint Performance Measurement Run-Time Infrastructure

NLPE@HLRS – Tools Day

12 June 2026

ZIH
Center for Information Services & High Performance Computing

Score-P Overview



Score-P Instrumenter

- Command to modify compile/link steps to instrument application
- Must be prepended to each command
- Original compile/link command:

```
$ icpc -Xhost … -c file.cpp
```

- Modified command:

```
$ scorep icpc -Xhost … -c file.cpp
```

- Must be manually integrated into build system

Score-P Instrumenter

- Auto detects most used programming paradigms
  - Based on compiler name, compiler flags, or undefined functions
- Overwritten by command flags:

```
$ scorep --help

This is the Score-P instrumentation tool. The usage is:
scorep <options> <original command>

Common options are:
  --[no]compiler
  --[no]user
  --thread=(none|omp|pthread)
  --mpp=(none|mpi|shmem)
  --[no]cuda
  --[no]openacc
  --[no]opencl
  --io[=posix]
  --[no]memory
```

- Supported features depends on build configuration

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Center for Information Services and High Performance Computing (ZIH)

Score-P: Event trace collection

Event Trace Collection Steps

Traces can become extremely large and unwieldy

- Size is proportional to number of processes/threads (width), duration (length) and detail (depth) of measurement

Traces containing intermediate flushes are of little value

Uncoordinated flushes result in cascades of distortion

- Reduce size of trace
- Increase available buffer space

Traces should be written to a parallel file system

- /work or /scratch are typically provided for this purpose

Moving large traces between file systems is often impractical

- However, systems with more memory can analyze larger traces
- Alternatively, run trace analyzers with undersubscribed nodes

Event Trace Collection 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

Reference preparation for validation

Goals:

1. Get familiar with build instructions for the application
2. Run application to determine a reasonable input size and runtime

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:

```
# Upload the TOOLS.tar.gz from the Moodle to your home directory
% tar xvzf TOOLS.tar.gz
% cd TOOLS
% source tools-env.sh
```

BT-MZ / Reference preparation

Build uninstrumented benchmark:

```
% cd BT-MZ
% make bt-mz NPROCS=4 CLASS=C
cd BT-MZ; make CLASS=W 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:

```
% srun -n4 -c32 bin/bt
 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:     32  (  8.0 threads/process)

 Calculated speedup =     31.99
 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    20.41
 Total processes =                        4
 Total threads   =                       32
 Mop/s total     =                118925.36
 Mop/s/thread    =                  3716.42
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              27 May 2025
```

Event Trace Collection 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

Program instrumentation
Summary measurement collection

Goals:

1. Adjust build system to use Score-P instrumenter
2. Run application to determine measurement overhead

BT-MZ / Instrumentation

Edit **config/make.def** to examine 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 = $(PREP) mpif77

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

Allows prepending **scorep** to **mpif77**

BT-MZ / Instrumentation

Build instrumented benchmark:

```
% make clean
% make PREP=scorep bt-mz NPROCS=4 CLASS=C
cd BT-MZ; make CLASS=W 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 instrumented benchmark:

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep-bt_mz-4x8-profile
% OMP_NUM_THREADS=8 mpirun -np 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:     32  (  8.0 threads/process)

 Calculated speedup =     31.99
 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    65.51
 Total processes =                       4
 Total threads   =                      32
 Mop/s total     =                 37049.94
 Mop/s/thread    =                  1157.81
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              08 Jun 2026
```

BT-MZ / Result examination

Creates experiment directory `./scorep-bt_mz-4x8-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-4x8-profile
MANIFEST.md  profile.cubex  scorep.cfg
% cat scorep-bt_mz-4x8-profile/MANIFEST.md
% cat scorep-bt_mz-4x8-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!

Event Trace Collection 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

Summary experiment scoring
Summary measurement collection with filtering

Goals:

1. Determine functions causing measurement overhead or trace buffer requirements
2. Create filter to exclude these functions from measurement
3. Verify filtering reduced the measurement overhead or trace buffer requirements

BT-MZ / Summary Analysis Result Scoring

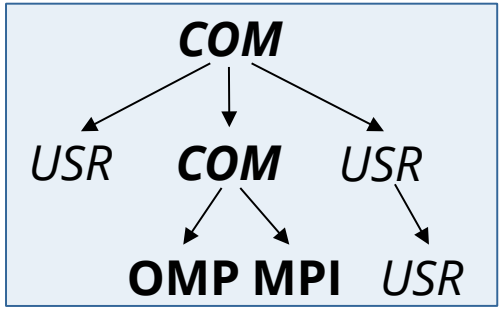
161GB total memory
41GB per rank!

Report scoring as textual output

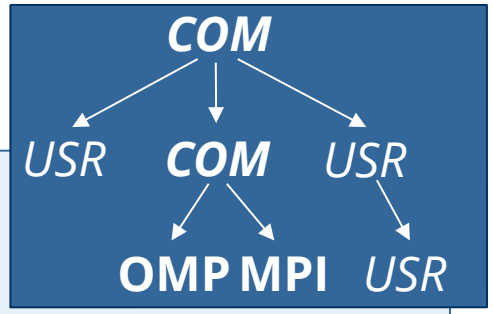
```
% scorep-score scorep-bt_mz-4x8-profile/profile.cubex

Estimated aggregate size of event trace:                   161GB
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 43,216,467,671 6,597,243,813 2059.79   100.0           0.31  ALL
         USR 42,988,648,534 6,574,790,617  887.80    43.1           0.14  USR
         OMP    222,996,992    21,723,136 1158.81    56.3          53.34  OMP
         COM      4,697,810       722,740    8.45     0.4          11.69  COM
         MPI        124,294         7,316    4.72     0.2         644.54  MPI
      SCOREP             41             4    0.01     0.0        2243.75  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-4x8-profile/profile.cubex


flt     type     max_buf[B]         visits time[s] time[%] time/visit[us]  region
         ALL 43,244,308,217 6,599,953,989 1466.39   100.0           0.22  ALL
         USR 42,988,632,934 6,574,788,217  677.23    46.2           0.10  USR
         OMP    250,853,312    24,435,712  778.31    53.1          31.85  OMP
         COM      4,697,810       722,740    1.86     0.1           2.57  COM
         MPI        124,120         7,316    8.99     0.6        1228.34  MPI
      SCOREP             41             4    0.02     0.0        3825.92  SCOREP

         USR 13,812,365,034 2,110,313,472  285.49    19.5           0.14  binvcrhs_
         USR 13,812,365,034 2,110,313,472  152.25    10.4           0.07  matvec_...
         USR 13,812,365,034 2,110,313,472  217.87    14.9           0.10  matmul_...
         USR    596,197,758    87,475,200    8.37     0.6           0.10  lhsinit_
         USR    596,197,758    87,475,200    7.11     0.5           0.08  binvrhs_
         USR    447,869,968    68,892,672    6.13     0.4           0.09  exact_...
```

BT-MZ / Summary Analysis Report Breakdown

Create filtering file for high-frequent regions

```
% scorep-score -g scorep-bt_mz-4x8-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

976 MB total memory
262 MB per rank!

Simulate filtering file

```
% scorep-score -f initial_scorep.filter \
  scorep-bt_mz-4x8-profile/profile.cubex

Estimated aggregate size of event trace:                   976MB
Estimated requirements for largest trace buffer (max_buf): 244MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       262MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=262MB 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 43,244,308,217 6,599,953,989 1466.39   100.0           0.22  ALL
 -       USR 42,988,632,934 6,574,788,217  677.23    46.2           0.10  USR
 -       OMP    250,853,312    24,435,712  778.31    53.1          31.85  OMP
 -       COM      4,697,810       722,740    1.86     0.1           2.57  COM
 -       MPI        124,120         7,316    8.99     0.6        1228.34  MPI
 -    SCOREP             41             4    0.02     0.0        3825.92  SCOREP
```

BT-MZ / Filtered run

Run instrumented benchmark with filter:

```
% export SCOREP_FILTERING_FILE=initial_scorep.filter
% ...experiment directory as you like
% OMP_NUM_THREADS=8 mpirun -np 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:     32  (  8.0 threads/process))

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                     200
 Time in seconds =                   28.91
 Total processes =                      4
 Total threads   =                     32
 Mop/s total     =                83938.02
 Mop/s/thread    =                 2623.06
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              08 Jun 2026
```

BT-MZ / Filtered run

Verify filtering result:

```
% scorep-score scorep-bt_mz-4x8-filtered/profile.cubex

Estimated aggregate size of event trace:                   870MB
Estimated requirements for largest trace buffer (max_buf): 218MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       234MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=234MB 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 227,865,495 22,460,325  888.86   100.0          39.57  ALL
         OMP 222,996,992 21,723,136  876.76    98.6          40.36  OMP
         COM   4,697,810    722,740    7.86     0.9          10.87  COM
         MPI     124,294      7,316    4.23     0.5         578.01  MPI
         USR      46,358      7,129    0.01     0.0           0.91  USR
      SCOREP          41          4    0.00     0.0        1188.77  SCOREP
```

Event Trace Collection 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

Event trace collection

Goals:

1. Create event trace with reduced overhead for further analysis

BT-MZ / Tracing run

Run instrumented benchmark in trace mode:

```
% export SCOREP_TOTAL_MEMORY=270MB
% export SCOREP_ENABLE_TRACING=true
...don't forget the experiment directory!
 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:     32  (  8.0 threads/process)

 BT-MZ Benchmark Completed.
 Class           =                        C
 Size            =            480x  320x 28
 Iterations      =                      200
 Time in seconds =                    28.61
 Total processes =                       4
 Total threads   =                      32
 Mop/s total     =                 84822.23
 Mop/s/thread    =                  2650.69
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
 Compile date    =              08 Jun 2026
```

BT-MZ / trace result examination

Creates experiment directory `./scorep-bt_mz-4x8-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-4x8-tracing
MANIFEST.md  scorep.cfg  traces/  traces.def  traces.otf2
% cat scorep-bt_mz-4x8-tracing/MANIFEST.md
% cat scorep-bt_mz-4x8-tracing/scorep.cfg
```

Download the results to your local machine and open with the Vampir client!