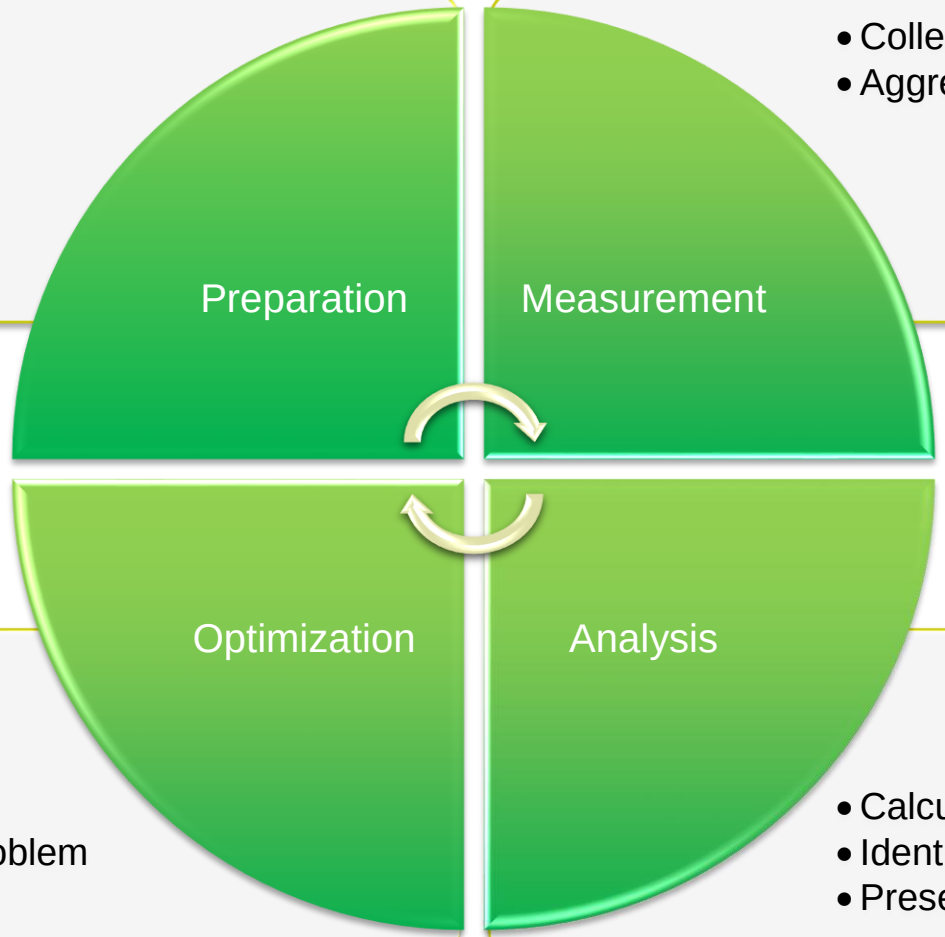
Measurement with Score-P

Marc Schlütter, JSC

Performance engineering workflow

- Prepare application with symbols
- Insert extra code (probes/hooks)



- Collection of performance data
- Aggregation of performance data

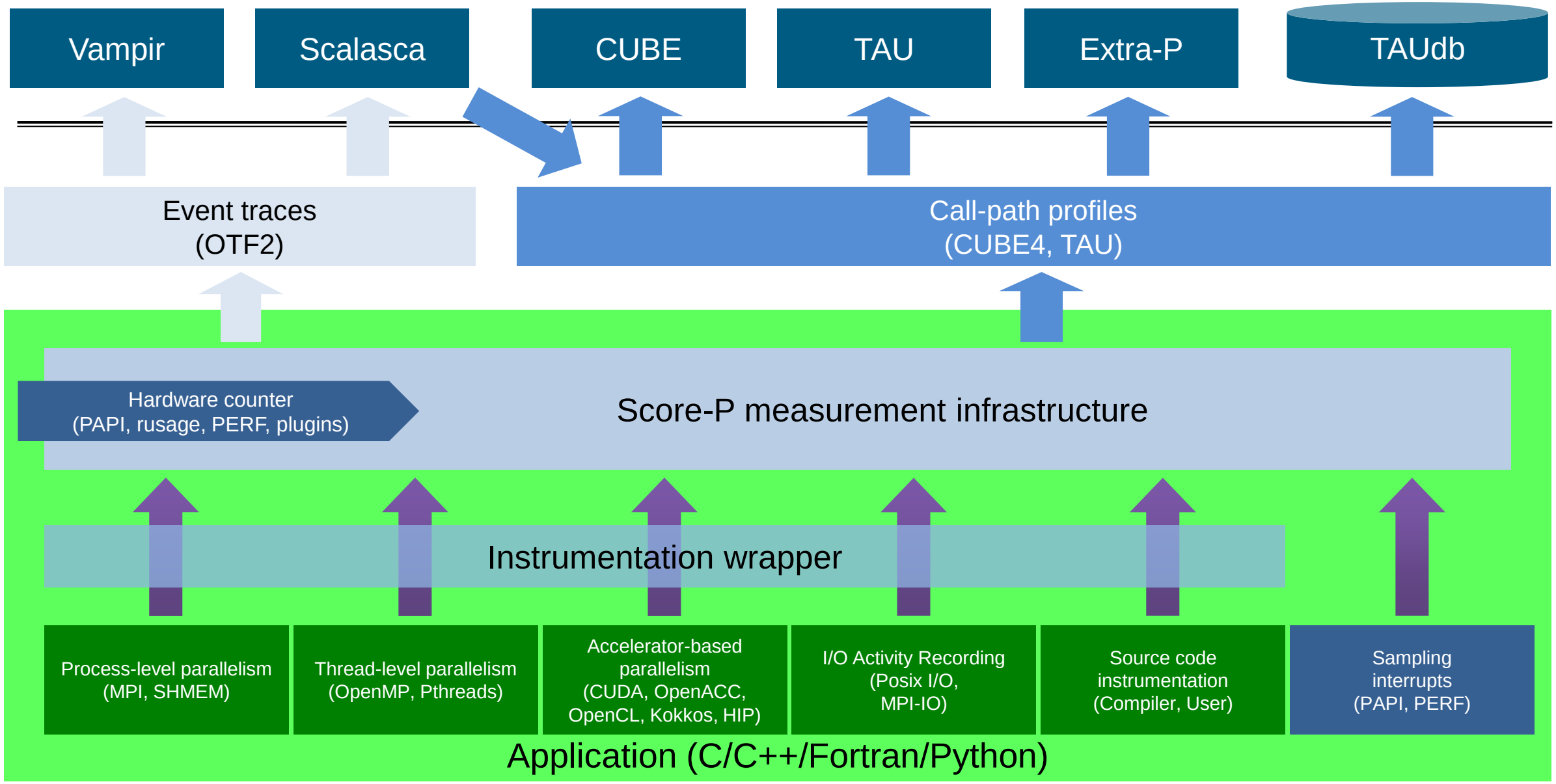
- Modifications intended to
- eliminate/reduce performance problem

- Calculation of metrics
- Identification of performance problems
- Presentation of results

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, ROCm, OpenCL, OpenACC
- Open Source; portable and scalable to all major HPC systems
- Initial project funded by BMBF
- Close collaboration with PRIMA project funded by DOE

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

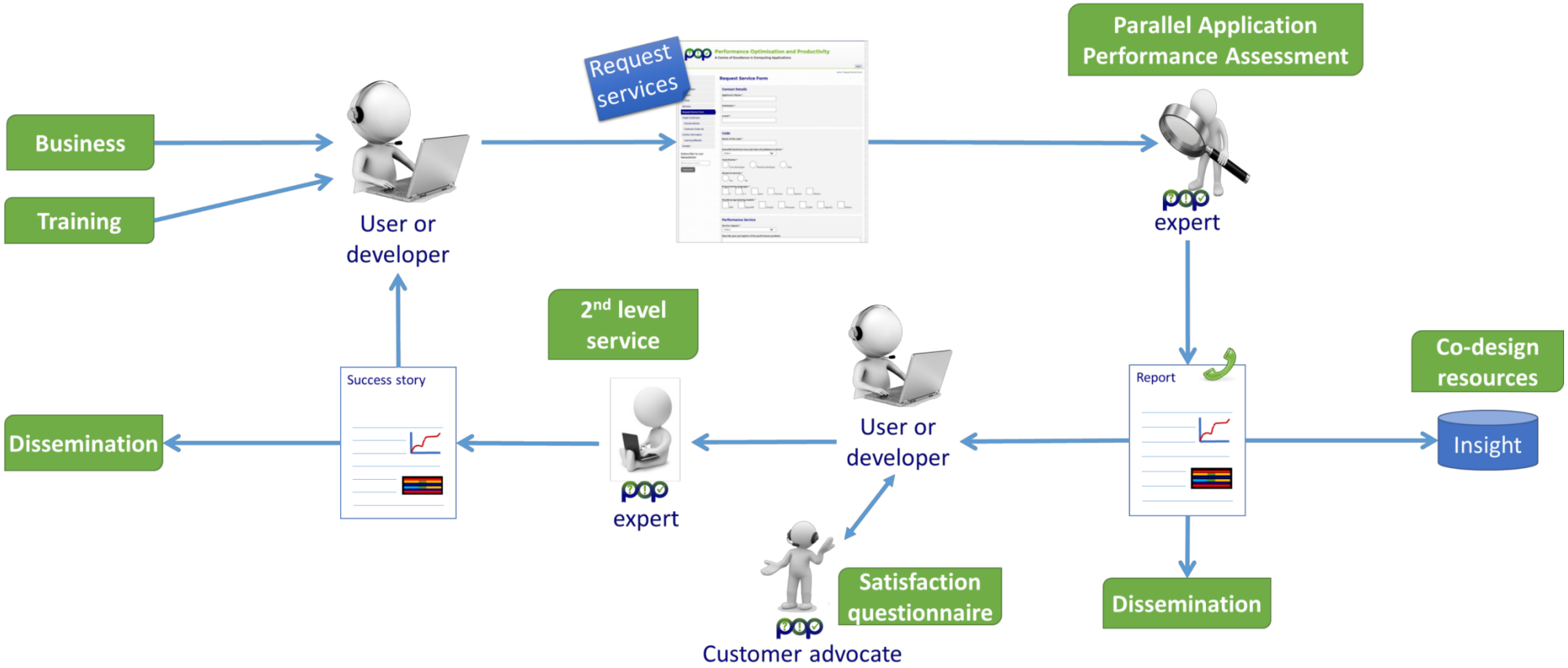

Vampir
Scalasca
CUBE
TAU
Extra-P
TAUdb
Event traces
(OTF2)
Call-path profiles
(CUBE4, TAU)
Hardware counter
(PAPI, rusage, PERF, plugins)
Score-P measurement infrastructure
Instrumentation wrapper
Process-level parallelism
(MPI, SHMEM)
Thread-level parallelism
(OpenMP, Pthreads)
Accelerator-based
parallelism
(CUDA, OpenACC,
OpenCL, Kokkos, HIP)
I/O Activity Recording
(Posix I/O,
MPI-IO)
Source code
instrumentation
(Compiler, User)
Sampling
interrupts
(PAPI, PERF)
Application (C/C++/Fortran/Python)

Partners

- Forschungszentrum Jülich, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Darmstadt, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

POP CoE (Center of Excellence)

Free Services: Performance Assessments & Second level Services



https://pop-coe.eu/request-service-form

contact: pop@bsc.es

Reference hands-on: NPB-MZ-MPI / BT

Performance analysis steps

- Reference preparation for validation

- Program instrumentation
- Summary measurement collection

- Summary experiment scoring
- Trace measurement collection with filtering

Environment for Score-P on Emmy 3 CPU Partition

- Using GCC + OpenMPI tool chain on the Emmy 3 CPU partition:

```
% ssh -X <userid>glogin-p3.hpc.gwdg.de
% module use $PROJECT/modules
% module load gcc/14.2.0 openmpi/4.1.7
% module load scorep/10.1-emmyp3-gcc-openmpi scalasca/2.6.2-emmyp3-gcc-openmpi cubelib/4.9.1
```

- Setting up the hands-on materials for Score-P in the training scratch space:
  - Today: Looking at MPI+OpenMP with the NAS Parallel Benchmark suite

```
% mkdir -p /scratch/lustre-grete/projects/tr_20260831-vi-hps-tuning/$USER/scorep
% cd /scratch/lustre-grete/projects/tr_20260831-vi-hps-tuning/$USER/scorep
% tar xfv $PROJECT/training-materials/scorep/NPB-BT-MZ.tar.gz
% cd NPB-BT-MZ
% ls
BT-MZ  LU-MZ  Makefile  README  README.install  README.tutorial  SP-MZ common
config  jobscript  sys
```

NPB-MZ-MPI / BT configuration

```
% <editor> config/make.def
...
# PREP is a generic macro for instrumentation preparation
MPIF77 = $(PREP)  mpif90  -fallow-argument-mismatch
...
#---------------------------------------------------------------
# Global *compile time* flags for Fortran programs
#---------------------------------------------------------------
FFLAGS  = -O3 $(OPENMP)
```

- Specify mpif90
- Any tuning (not a huge difference for BT-MZ, but good practice!)

NPB-MZ-MPI / BT build

```
% make bt-mz CLASS=C NPROCS=16
cd BT-MZ; make CLASS=C NPROCS=16 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 16 C
mpif90 -fallow-argument-mismatch -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  mpif90 -fallow-argument-mismatch -c  -O3 \
-fopenmp timers.f
mpif90 -fallow-argument-mismatch -O3 -fopenmp \
-o ../bin/bt-mz_C.16 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/bt-mz_C.16
make: Leaving directory 'BT-MZ'
```

- Benchmark name:
  - **bt-mz**, lu-mz, sp-mz
- Number of MPI processes:
  - NPROCS=**16**
- Benchmark class:
  - S, W, A, B, **C**, D, E
  - CLASS=**D**
- Alternatively:
  - `make suite`

NPB-MZ-MPI / BT job submission

```
% cd bin
% cp ../jobscript/emmyp3/run.slurm .
% cat run.slurm
#!/bin/bash
#SBATCH --job-name=bt-run
#SBATCH --partition=standard96s:shared
#SBATCH --reservation=vi-hps-tuning
...

##### BT-MZ configuration
# disable load balancing with threads
export NPB_MZ_BLOAD=0
PROCS=${SLURM_NPROCS}
CLASS=C
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

##### Run the application
srun ./bt-mz_$CLASS.$PROCS

% sbatch run.slurm
```

- Bring appropriate job script into main benchmark directory
- Note the job name (used to sort output) and the OpenMP thread pinning variables (for your own codes)
- Note the output locations (site-specific!)
- Run with workshop account and reservation

NPB-MZ-MPI / BT reference execution

```
% cat bt-run.o*
 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

 Use the default load factors with threads
 Total number of threads:    96  (  6.0 threads/process)

 Calculated speedup =    95.83

 Time step    1
 [... More application output ...]
 Time step  200
 [... More application output ...]
 BT-MZ Benchmark Completed.
 Time in seconds = 14.42
```

- Launch as a hybrid MPI+OpenMP application

Save the benchmark run time to be able to refer to it later. (Beware of potential over-subscription)

Performance analysis steps

- Reference preparation for validation

- Program instrumentation
- Summary measurement collection

- Summary experiment scoring
- Trace measurement collection with filtering

NPB-MZ-MPI / BT instrumented build

```
% make clean; make bt-mz CLASS=C NPROCS=16 PREP=scorep
cd BT-MZ; make CLASS=C NPROCS=16 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 16 C
mpif90 -fallow-argument-mismatch -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  scorep mpif90 -fallow-argument-mismatch -c  -O3 \
-fopenmp timers.f
scorep mpif90 -fallow-argument-mismatch -O3 \
-fopenmp -o ../bin.scorep/bt-mz_C.16 \
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.16
make: Leaving directory 'BT-MZ'
```

- Re-build executable prefixing the compiler with the `scorep` command

NPB-MZ-MPI / BT summary measurement collection

```
% cd bin.scorep
% cp ../jobscript/emmyp3/scorep.slurm .
% vi scorep.slurm
#!/bin/bash
#SBATCH -J bt-scorep
...
##### Measurement configuration
#export SCOREP_FILTERING_FILE=initial_scorep.filter
export SCOREP_EXPERIMENT_DIRECTORY=scorep-btmz-${CLASS}-\
${SLURM_NPROCS}x${SLURM_CPUS_PER_TASK}

#export SCOREP_ENABLE_TRACING=true
#export SCOREP_TOTAL_MEMORY=150M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

...
<save and exit>
% sbatch scorep.slurm
```

- Point the script to the instrumented executable
- Configure measurement variables
- Run instrumented application

Measurement configuration: scorep-info

```
% 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 ...]
```

- Score-P measurements are configured via environment variables

NPB-MZ-MPI / BT summary analysis report examination

```
% ls scorep-btmz-C-16x6/
MANIFEST.md  profile.cubex  scorep.cfg
% less scorep-btmz-C-16x6/MANIFEST.md
% less scorep-btmz-C-16x6/scorep.cfg
% less bt-scorep.o*
...
Time in seconds =                    33.92
...



% # optional
% cube scorep-btmz-C-16x6/profile.cubex

  [CUBE GUI showing summary analysis report]
```

- Creates experiment directory including
  - Experiment directory overview (`MANIFEST.md`)
  - A record of the measurement configuration (`scorep.cfg`)
  - The analysis report that was collated after measurement (`profile.cubex`)

Congratulations!?

- If you made it this far, you successfully used Score-P to
  - instrument the application
  - record its execution with a summary measurement, and
  - [optional] 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

Performance analysis steps

- Reference preparation for validation

- Program instrumentation
- Summary measurement collection

- Summary experiment scoring
- Trace measurement collection with filtering

Goals of scoring and filtering

- Evaluate how *expensive* measurement of various regions is
  - Time cost is roughly fixed per event
  - Short functions are relatively more expensive
  - Space cost for tracing is linear in number of events
- Determine which expensive regions are *unnecessary* to measure
  - Frequently called, short execution, and non-scaling behavior
- Repeat the measurement, with those regions *filtered* at runtime to reduce overhead
  - Reduce space cost to zero and time cost to a hash comparison and a couple of branches
- (Optional) Apply the filter at *compile-time* in order to further reduce overhead
  - Eliminates the hash and branches; often not needed

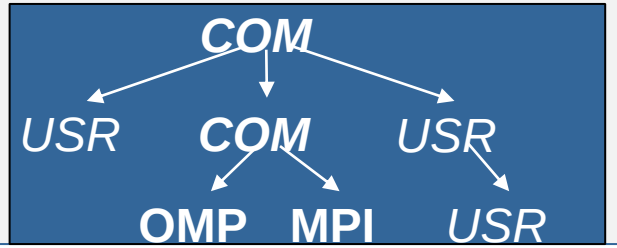
NPB-MZ-MPI / BT summary analysis result scoring

```
% scorep-score scorep-btmz-C-16x6/profile.cubex

Estimated aggregate size of event trace:                   160GB
Estimated requirements for largest trace buffer (max_buf): 11GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       11GB
(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 10,798,266,033 6,592,034,219 2795.66   100.0           0.42  ALL
         USR 10,754,591,276 6,574,805,745 1084.55    38.8           0.16  USR
         OMP     41,920,928    16,359,424 1596.60    57.1          97.60  OMP
         COM      1,178,450       725,200    2.59     0.1           3.58  COM
         MPI        616,342       143,834  111.91     4.0         778.05  MPI
      SCOREP             41            16    0.00     0.0          35.22  SCOREP
```

160 GB total memory
11 GB per rank!



Region/callpath classification

- **MPI** pure MPI functions
- **OMP** pure OpenMP regions
- **USR** user-level computation
- **COM** "combined" USR+OpenMP/MPI
- **SCOREP** measurement internals
- **ANY/ALL** aggregate of all region types

NPB-MZ-MPI / BT summary analysis report breakdown

```
% scorep-score -r scorep-btmz-C-16x6/profile.cubex
  [...]
flt     type     max_buf[B]         visits time[s] time[%] time/visit[us]  region
         ALL 10,798,266,033 6,592,034,219 2795.66   100.0           0.42  ALL
         USR 10,754,591,276 6,574,805,745 1084.55    38.8           0.16  USR
         OMP     41,920,928    16,359,424 1596.60    57.1          97.60  OMP
         COM      1,178,450       725,200    2.59     0.1           3.58  COM
         MPI        616,342       143,834  111.91     4.0         778.05  MPI
      SCOREP             41            16    0.00     0.0          35.22  SCOREP

         USR  3,454,903,374 2,110,313,472  471.13    16.9           0.22  binvcrhs
         USR  3,454,903,374 2,110,313,472  343.37    12.3           0.16  matmul_sub
         USR  3,454,903,374 2,110,313,472  228.02     8.2           0.11  matvec_sub
         USR    149,170,944    87,475,200   21.22     0.8           0.24  lhsinit
         USR    149,170,944    87,475,200   13.61     0.5           0.16  binvrhs
         USR    112,148,088    68,892,672    7.19     0.3           0.10  exact_solution
         OMP      3,357,504       617,472    0.16     0.0           0.27  !$omp parallel @exch_qbc.f:204
         OMP      3,357,504       617,472    0.16     0.0           0.26  !$omp parallel @exch_qbc.f:215
         OMP      3,357,504       617,472    0.17     0.0           0.27  !$omp parallel @exch_qbc.f:244
         OMP      3,357,504       617,472    0.17     0.0           0.27  !$omp parallel @exch_qbc.f:255
[...]
```

Majority of the 11 GB just for these 6 regions

NPB-MZ-MPI / BT summary analysis score

- Summary measurement analysis score reveals total size of event trace ~160 GB
- Maximum trace buffer size would be ~11 GB per rank
- 99.9% of the trace requirements are for USR regions
- These USR regions contribute around 30% of total time
- Conclusion: we need *filtering* to reduce overhead and remove uninteresting events!

NPB-MZ-MPI / BT summary analysis report filtering

```
% scorep-score -g scorep-btmz-C-16x6/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.
```

- Report scoring with prospective filter listing 6 USR regions

NPB-MZ-MPI / BT summary analysis report filtering

```
% cat initial_scorep.filter
...
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    # type=USR max_buf= 3,454,903,374 visits=   ...
    # name='binvcrhs'
    # file='BT-MZ/solve_subs.f'
    MANGLED binvcrhs_

    ...
SCOREP_REGION_NAMES_END

% scorep-score -f initial_scorep.filter \
> scorep-btmz-C-16x6/profile.cubex

Estimated aggregate size of event trace:                   668MB
Estimated requirements for largest trace buffer (max_buf): 42MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       54MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=54MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)
```

Manually adding ormodifying entries, Space separated and bash-like wildcards *, ?, []

- Report scoring with prospective filter listing 6 USR regions

668 MB of memory in total, 54 MB per rank!

NPB-MZ-MPI / BT summary analysis report filtering

```
% scorep-score -r -f initial_scorep.filter \
> scorep-btmz-C-16x6/profile.cubex
flt     type     max_buf[B]        visits time[s] time[%] time/visit[us]  region
 -       ALL 10,798,266,033 6,592,034,219 2795.66   100.0           0.42  ALL
 -       USR 10,754,591,276 6,574,805,745 1084.55    38.8           0.16  USR
 -       OMP     41,920,928    16,359,424 1596.60    57.1          97.60  OMP
 -       COM      1,178,450       725,200    2.59     0.1           3.58  COM
 -       MPI        616,342       143,834  111.91     4.0         778.05  MPI
 -    SCOREP             41            16    0.00     0.0          35.22  SCOREP

 *       ALL     43,751,953    17,250,731 1711.12    61.2          99.19  ALL-FLT
 +       FLT 10,754,555,110 6,574,783,488 1084.54    38.8           0.16  FLT
 -       OMP     41,920,928    16,359,424 1596.60    57.1          97.60  OMP-FLT
 *       COM      1,178,450       725,200    2.59     0.1           3.58  COM-FLT
 -       MPI        616,342       143,834  111.91     4.0         778.05  MPI-FLT
 *       USR         36,192        22,257    0.01     0.0           0.62  USR-FLT
 -    SCOREP             41            16    0.00     0.0          35.22  SCOREP-FLT

 +       USR  3,454,903,374 2,110,313,472  471.13    16.9           0.22  binvcrhs
 +       USR  3,454,903,374 2,110,313,472  343.37    12.3           0.16  matmul_sub
 +       USR  3,454,903,374 2,110,313,472  228.02     8.2           0.11  matvec_sub
 +       USR    149,170,944    87,475,200   21.22     0.8           0.24  lhsinit
 +       USR    149,170,944    87,475,200   13.61     0.5           0.16  binvrhs
 +       USR    112,148,088    68,892,672    7.19     0.3           0.10  exact_solution
 -       OMP      3,357,504       617,472    0.16     0.0           0.27  !$omp parallel ...
```

Markings:
- `+`: Filtered
- `-`: Not filtered
- `*`: Mixed

- Score report breakdown by region

Adding PAPI Performance counters

- Estimating the memory requirements due to recorded counters:

```
% <editor> scorep.slurm
...
#SBATCH -J bt-scorep-filter
% scorep-score -f initial_scorep.filter -c 2 scorep-btmz-C-16x6/profile.cubex

Estimated aggregate size of event trace:                   1639MB
Estimated requirements for largest trace buffer (max_buf): 103MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       115MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=115MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)
```

Increased memory requirements from the prior 115 MB per rank!

- With a PAPI module loaded: **`papi_avail`** for a list of available counters

NPB-MZ-MPI / BT filtered measurement collection with PAPI

```
% <editor> scorep.slurm
...
#SBATCH -J bt-scorep-filter
...
papi_avail > papi.log
...
export SCOREP_FILTERING_FILE=initial_scorep.filter
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
export SCOREP_EXPERIMENT_DIRECTORY=scorep-btmz-${CLASS}\
-${SLURM_NPROCS}x${SLURM_CPUS_PER_TASK}-filter

...
<save and exit>
% sbatch scorep.slurm
```

- Apply filter configuration and re-run measurement
- This gives you a profile with less noise
- Also, collecting a trace is now practical and easy. Remember this for later

NPB-MZ-MPI / BT filtered measurement collection with PAPI

```
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

 Use the default load factors with threads
 Total number of threads:    96  (  6.0 threads/process)

 Calculated speedup =    95.83

 Time step    1
 [... More application output ...]
 BT-MZ Benchmark Completed.
 Time in seconds = 14.80
```

- Output from filtered run

NPB-MZ-MPI / BT filtered results and post processing

```
% ls scorep-btmz-C-16x6-filter/
MANIFEST.md  profile.cubex  scorep.cfg   scorep.filter

% cube_remap2 -s -d -o scorep-btmz-C-16x6-filter/summary.cubex scorep-btmz-C-16x6-filter/profile.cubex
++++++++++++ Remapping operation begins ++++++++++++++++++++++++++
Reading scorep-btmz-C-16x6-filter/profile.cubex ...  done
Found remapping specification file inside of cube. Use it.
Create cube according the remapping specification...
 Copy source data ...

Source data copied in  0.0156318 seconds.
 Add scalasca specific metrics ...
 Add metrics "idle threads" and "limited parallelism"...done.
...
Generating global cartesian topologies.
 1 CPU topology generated.
...
Remapping done in 8.65861 seconds.
++++++++++++ Running Checks on >scorep-btmz-C-16x6-filter/summary.cubex< +++++++++++++++

File opened and checked in 0.00990545 seconds.
```

Running the remapper manually to generate the final result – creating new metrics based on existing data

Score-P: Further information

- Scalable Performance Measurement Infrastructure for Parallel Codes
  - Instrumenter, libraries, and tools to generate profile and trace measurements
  - Bundled with OTF2 (tracing), OPARI2 (OpenMP instrumentation), CubeWriter, and CubeLib (profiling)
- Available under 3-clause BSD open-source license
- Documentation & sources:
  - https://score-p.org
- User guide also part of installation:
  - <prefix>/share/doc/scorep/pdf/scorep.pdf
- Contact:
  - mailto: support@score-p.org

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

- https://www.pop-coe.eu
- pop@bsc.es
- @POP_HPC
- youtube.com/POPHPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.

EuroHPC