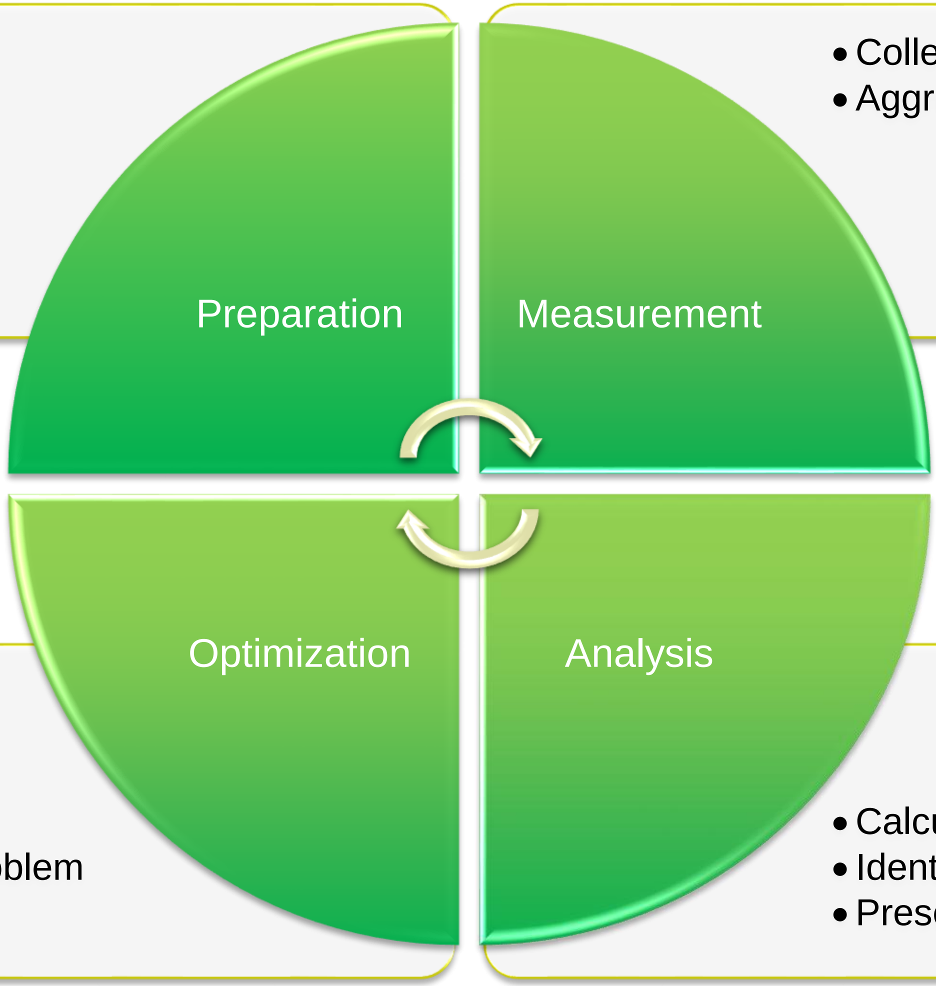
Measurement with Score-P

Marc Schlütter, JSC

Performance engineering workflow



Preparation:

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

Measurement:

- Collection of performance data
- Aggregation of performance data

Optimization:

- Modifications intended to
- eliminate/reduce performance problem

Analysis:

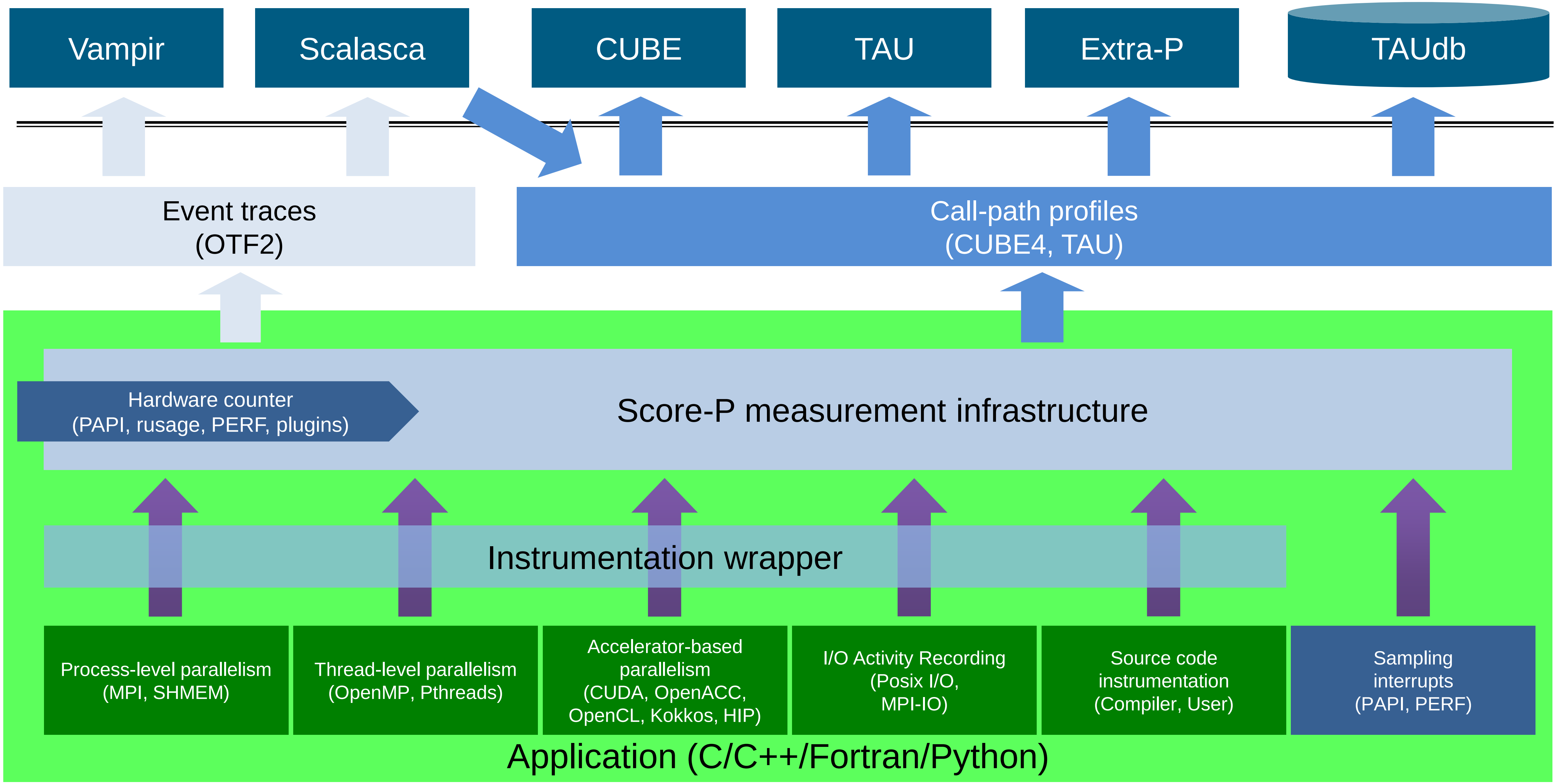
- Calculation of metrics
- Identification of performance problems
- Presentation of results

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

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA


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

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 MareNostrum 5 – GPP

- Using GCC + OpenMPI tool chain on the GPP partition:

```
% module purge
% module load gcc/12.3.0 openmpi/4.1.5-gcc12.3 scorep/9.4-gcc-ompi papi/7.1.0-gcc
% module load cubelib/4.9.1 scalasca/2.6.2-gcc-ompi
```

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

```
% cd /gpfs/scratch/nct_362
% mkdir -p users/<yourName>/hands-on && cd users/<yourName>/hands-on
% tar xf /gpfs/scratch/nct_362/exercises/scorep.tar.gz
% cd scorep/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
- Tuning flags for Sapphire Rapids (not a huge difference for BT-MZ, but good practice!)

NPB-MZ-MPI / BT build

```
% make bt-mz CLASS=D NPROCS=32
cd BT-MZ; make CLASS=D NPROCS=32 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 32 D
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_D.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/bt-mz_D.32
make: Leaving directory 'BT-MZ‘
```

- Benchmark name:
  - **bt-mz**, lu-mz, sp-mz
- Number of MPI processes:
  - NPROCS=**32**
- Benchmark class:
  - S, W, A, B, C, **D**, E
  - CLASS=**D**
- Alternatively:
  - `make suite`

NPB-MZ-MPI / BT job submission

```
% cd bin
% cp ../jobscript/marenostrum5/run.slurm .
% cat run.slurm
#!/bin/bash
#SBATCH --job-name=bt-run     # Job name
...
##### SLURM MN5 settings
# for taking hybrid case into account
export SRUN_CPUS_PER_TASK=${SLURM_CPUS_PER_TASK}

##### BT-MZ configuration
# disable load balancing with threads
export NPB_MZ_BLOAD=0
PROCS=${SLURM_NPROCS}
CLASS=D
export OMP_NUM_THREADS=${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:  32 x  32
 Iterations: 250    dt:   0.000020
 Number of active processes:    32

 Use the default load factors with threads
 Total number of threads:    224  (  7.0 threads/process)

 Calculated speedup =    223.86

 Time step    1
 [... More application output ...]
 Time step  200
 [... More application output ...]
 BT-MZ Benchmark Completed.
 Time in seconds = 61.94
```

- 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=D NPROCS=32 PREP=scorep
cd BT-MZ; make CLASS=D NPROCS=32 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 32 D
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_D.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_D.32
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/marenostrum5/scorep.slurm .
% vi scorep.slurm
#!/bin/bash
#SBATCH -J bt-scorep
...
##### Measurement configuration
#export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_EXPERIMENT_DIRECTORY=scorep-btmz-${CLASS}-\
${SLURM_NPROCS}x${SLURM_CPUS_PER_TASK}

#export SCOREP_ENABLE_TRACING=true
#export SCOREP_TOTAL_MEMORY=250M
...
<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-D-32x7/
MANIFEST.md  profile.cubex  scorep.cfg
% less scorep-bt_mz-8x8-profile/MANIFEST.md
% less scorep-bt_mz-8x8-profile/scorep.cfg
% less bt-scorep.o*
...
Time in seconds =                   132.64
...



% # optional
% cube scorep-btmz-D-32x7/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

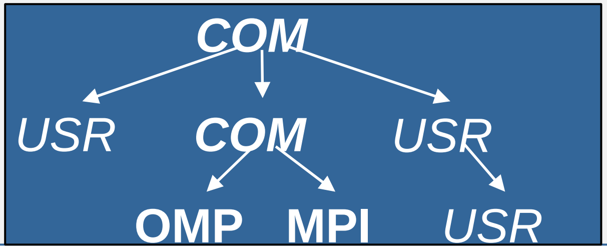
NPB-MZ-MPI / BT summary analysis result scoring

```
% scorep-score scorep-btmz-D-32x7/profile.cubex

Estimated aggregate size of event trace:                   3332GB
Estimated requirements for largest trace buffer (max_buf): 105GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       105GB
(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 112,035,087,899 137,518,952,689 29382.90   100.0           0.21  ALL
         USR 111,908,772,534 137,419,580,865 11380.33    38.7           0.08  USR
         OMP     121,893,856      95,047,680 17750.19    60.4         186.75  OMP
         COM       2,935,270       3,612,640    14.97     0.1           4.14  COM
         MPI       1,588,606         711,472   237.42     0.8         333.70  MPI
      SCOREP              41              32     0.00     0.0          56.03  SCOREP
```

3332 GB total memory
105 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-D-32x7/profile.cubex
  [...]
  [...]
flt     type      max_buf[B]          visits  time[s] time[%] time/visit[us]  region
         ALL 112,035,087,899 137,518,952,689 29382.90   100.0           0.21  ALL
         USR 111,908,772,534 137,419,580,865 11380.33    38.7           0.08  USR
         OMP     121,893,856      95,047,680 17750.19    60.4         186.75  OMP
         COM       2,935,270       3,612,640    14.97     0.1           4.14  COM
         MPI       1,588,606         711,472   237.42     0.8         333.70  MPI
      SCOREP              41              32     0.00     0.0          56.03  SCOREP

         USR  36,395,971,872  44,677,967,872  5074.31    17.3           0.11  binvcrhs
         USR  36,395,971,872  44,677,967,872  3479.03    11.8           0.08  matmul_sub
         USR  36,395,971,872  44,677,967,872  2510.98     8.5           0.06  matvec_sub
         USR     957,566,506   1,152,495,616   152.76     0.5           0.13  lhsinit
         USR     957,566,506   1,152,495,616   103.12     0.4           0.09  binvrhs
         USR     878,133,152   1,080,078,336    60.03     0.2           0.06  exact_solution
         OMP       9,782,976       3,598,336     0.53     0.0           0.15  !$omp parallel @exch_qbc.f:204
         OMP       9,782,976       3,598,336     0.52     0.0           0.14  !$omp parallel @exch_qbc.f:215
         OMP       9,782,976       3,598,336     0.49     0.0           0.14  !$omp parallel @exch_qbc.f:244
         OMP       9,782,976       3,598,336     0.55     0.0           0.15  !$omp parallel @exch_qbc.f:255
[...]
```

Majority of the 105 GB just for these 6 regions

NPB-MZ-MPI / BT summary analysis score

- Summary measurement analysis score reveals total size of event trace ~3300 GB
- Maximum trace buffer size would be ~105 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-D-32x7/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 3 USR regions

NPB-MZ-MPI / BT summary analysis report filtering

```
% cat initial_scorep.filter
...
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
   # type=USR max_buf= 36,395,971,872 [..]
   # name='binvcrhs'
   # file='BT-MZ/solve_subs.f'
   MANGLED binvcrhs_

           ...
SCOREP_REGION_NAMES_END

% scorep-score -f initial_scorep.filter \
> scorep-bt_mz-8x8-profile/profile.cubex

Estimated aggregate size of event trace:                   86GB
Estimated requirements for largest trace buffer (max_buf): 2784MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       2798MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=2798MB to avoid intermediate flushes
or reduce requirements using USR regions filters.)
```

- Report scoring with prospective filter listing 6 USR regions

86 GB of memory in total,
2784 MB per rank!
Refinement needed!

NPB-MZ-MPI / BT summary analysis report filtering

```
% scorep-score -r -f initial_scorep.filter \
> scorep-btmz-D-32x7/profile.cubex
flt     type         max_buf[B]             visits  time[s] time[%] time/visit[us]  region
 -       ALL 112,035,087,899 137,518,952,689 29382.90   100.0           0.21  ALL
 -       USR 111,908,772,534 137,419,580,865 11380.33    38.7           0.08  USR
 -       OMP     121,893,856      95,047,680 17750.19    60.4         186.75  OMP
 -       COM       2,935,270       3,612,640    14.97     0.1           4.14  COM
 -       MPI       1,588,606         711,472   237.42     0.8         333.70  MPI
 -    SCOREP              41              32     0.00     0.0          56.03  SCOREP

 *       ALL   2,919,186,417   3,485,049,073 18318.59    62.3           5.26  ALL-FLT
 +       FLT 109,187,915,616 134,033,903,616 11064.32    37.7           0.08  FLT
 *       USR   2,792,819,848   3,385,677,249   316.01     1.1           0.09  USR-FLT
 -       OMP     121,893,856      95,047,680 17750.19    60.4         186.75  OMP-FLT
 *       COM       2,935,270       3,612,640    14.97     0.1           4.14  COM-FLT
 -       MPI       1,588,606         711,472   237.42     0.8         333.70  MPI-FLT
 -    SCOREP              41              32     0.00     0.0          56.03  SCOREP-FLT

+       USR  36,395,971,872  44,677,967,872  5074.31    17.3           0.11  binvcrhs
+       USR  36,395,971,872  44,677,967,872  3479.03    11.8           0.08  matmul_sub
+       USR  36,395,971,872  44,677,967,872  2510.98     8.5           0.06  matvec_sub
-       USR     957,566,506   1,152,495,616   152.76     0.5           0.13  lhsinit
-       USR     957,566,506   1,152,495,616   103.12     0.4           0.09  binvrhs
-       USR     878,133,152   1,080,078,336    60.03     0.2           0.06  exact_solution
```

Filtered routines marked with ‘+’

- Score report breakdown by region
- Adapt initial filter file to include the 3 additional routines

NPB-MZ-MPI / BT summary analysis report filtering

```
% scorep-score -r -f initial_scorep.filter \
> scorep-btmz-D-32x7/profile.cubex
Estimated aggregate size of event trace:                   3871MB
Estimated requirements for largest trace buffer (max_buf): 122MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       136MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=136MB to avoid intermediate flushes
or reduce requirements using USR regions filters.)
[...]
+       USR  36,395,971,872  44,677,967,872  5063.46    17.2            0.11  binvcrhs
+       USR  36,395,971,872  44,677,967,872  3469.07    11.8            0.08  matmul_sub
+       USR  36,395,971,872  44,677,967,872  2493.53     8.5            0.06  matvec_sub
+       USR     957,566,506   1,152,495,616   152.54     0.5            0.13  lhsinit
+       USR     957,566,506   1,152,495,616   102.90     0.4            0.09  binvrhs
+       USR     878,133,152   1,080,078,336    59.91     0.2            0.06  exact_solution

% cat initial_scorep.filter
SCOREP_REGION_NAMES_BEGIN
 EXCLUDE
   lhsinit
   binvrhs
   exact_solution

   # type=USR max_buf= 36,395,971,872 visits= 44,677,967,872, ...
   # name='binvcrhs'
   # file='BT-MZ/solve_subs.f'
   MANGLED binvcrhs_
```

3.9 GB of memory in total, 136 MB per rank!

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

- Refined filter reduces the requirements to useful sizes.
- More information on manual and automatic filtering in the user documentation and `scorep-score --help`

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-D-32x7/prof

Estimated aggregate size of event trace:                   10GB
Estimated requirements for largest trace buffer (max_buf): 297MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       311MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=311MB to avoid intermediate flushes
or reduce requirements using USR regions filters.)
```

Increased memory requirements from the prior 136 MB per rank!

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

- Marenostrum 5 needs **`#SBATCH --constraint=perfparanoid`** to function

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

```
% <editor> scorep.slurm
...
#SBATCH -J bt-scorep-filter
#SBATCH --constraint=perfparanoid # for PAPI counters
...
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:  32 x  32
Iterations: 250    dt:   0.000020
Number of active processes:    32

Use the default load factors with threads
Total number of threads:    224  (  7.0 threads/process)

 Calculated speedup =    223.86

 Time step    1
 [... More application output ...]
 BT-MZ Benchmark Completed.
 Time in seconds = 61.90
```

- Output from filtered run

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

```
% ls scorep-btmz-D-32x7-filter/
MANIFEST.md  profile.cubex  scorep.cfg   scorep.filter


% cube_remap2 -s -d -o scorep-btmz-D-32x7-filter/summary.cubex scorep-btmz-D-32x7-filter/profile.cubex
++++++++++++ Remapping operation begins ++++++++++++++++++++++++++
Reading scorep-btmz-D-32x7-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  0.0225998 seconds.
Add scalasca specific metrics ...
Add metrics "idle threads" and "limited parallelism"...done.
...
Writing scorep-btmz-D-32x7-filter/summary.cubex ... done.

Remapping done in 1.49807 seconds.
++++++++++++ Running Checks on >scorep-btmz-D-32x7-filter/summary.cubex< ++++++++++++++++

File opened and checked in 0.0722102 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