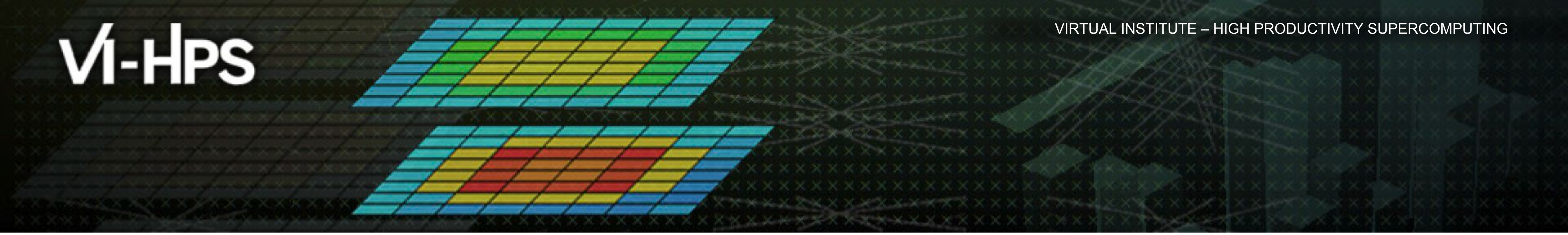


Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

VI-HPS Team

Score-P

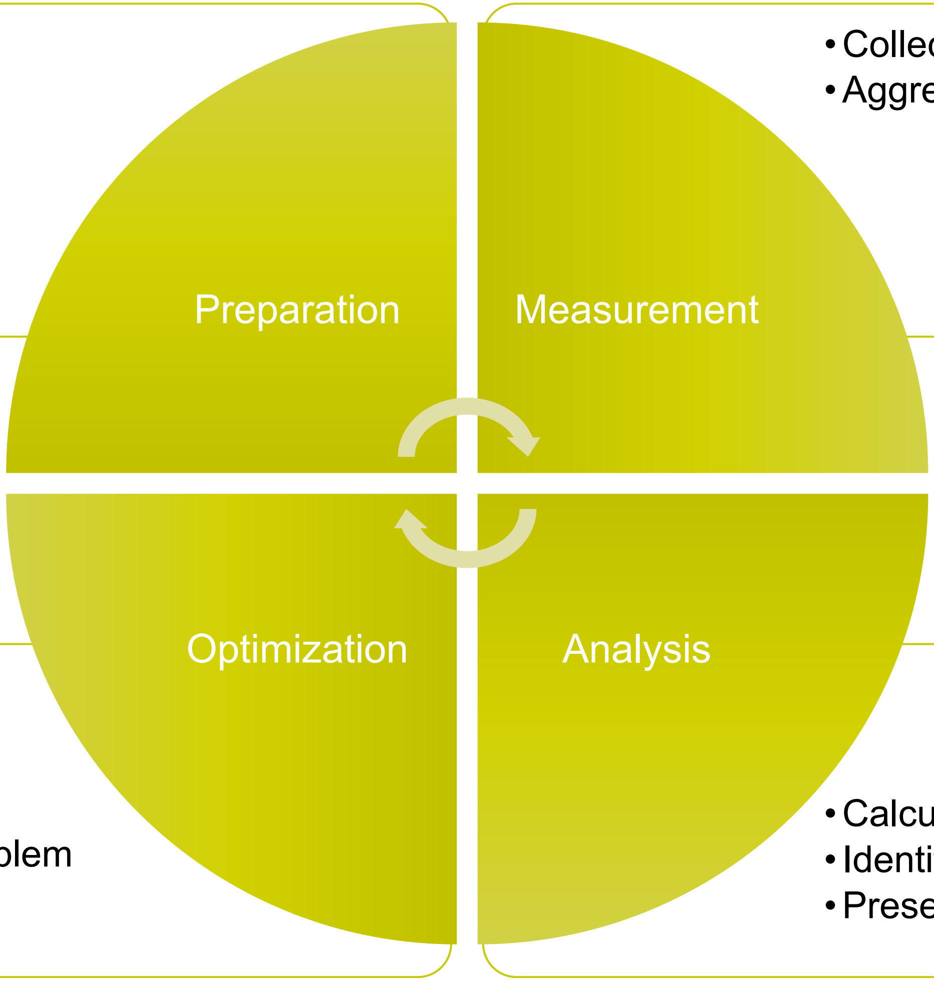
Scalable performance measurement infrastructure for parallel codes

allinea
JÜLICH FORSCHUNGSZENTRUM
Lawrence Livermore National Laboratory
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
German Research School for Simulation Sciences
RWTH AACHEN UNIVERSITY
TUM Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE

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

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

Vampir	Scalasca	TAU	Periscope
VampirTrace OTF	EPILOG / CUBE	TAU native formats	Online measurement

Score-P 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
- 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

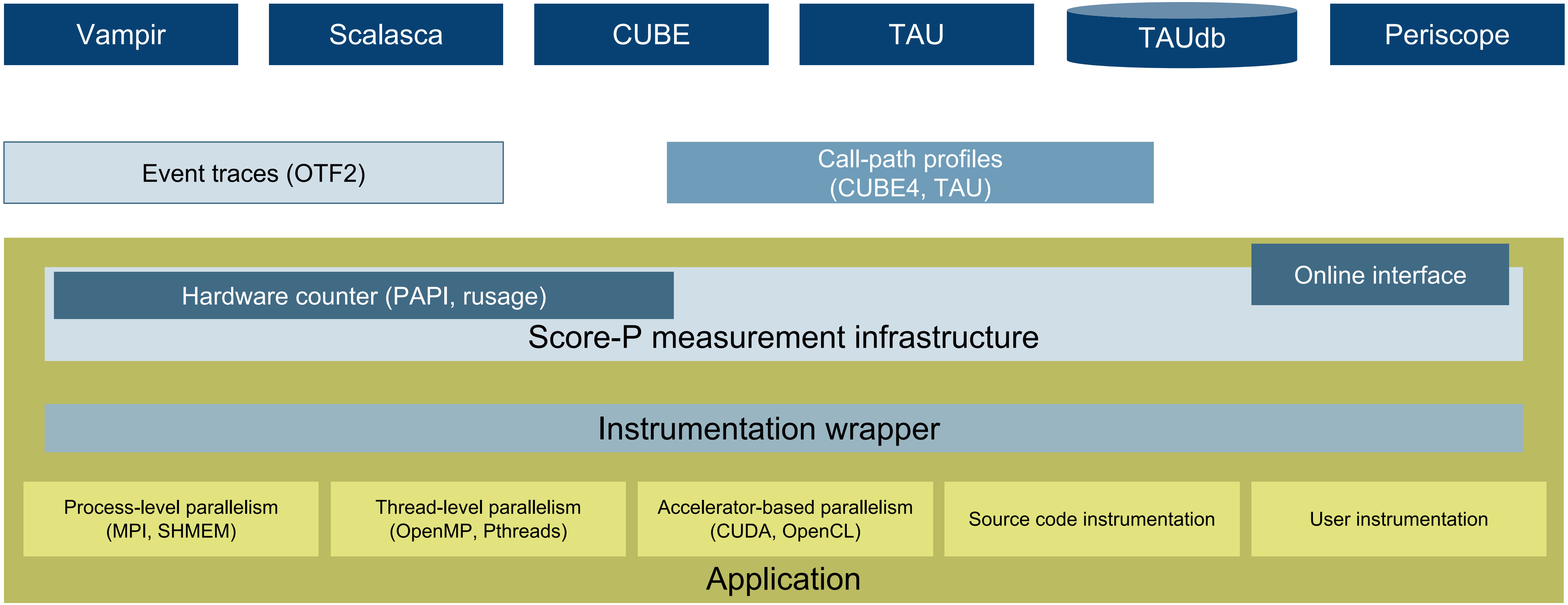
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI/SHMEM, OpenMP/Pthreads, and hybrid parallelism (and serial)
- Enhanced functionality (CUDA, OpenCL, 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, SHMEM, OpenMP, Pthreads, CUDA, OpenCL and their valid combinations
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Robustness
  - Open Source: New BSD License

Score-P Overview



Vampir

Scalasca

CUBE

TAU

TAUdb

Periscope

Event traces (OTF2)

Call-path profiles (CUBE4, TAU)

Hardware counter (PAPI, rusage)

Online interface

Score-P measurement infrastructure

Instrumentation wrapper

Process-level parallelism (MPI, SHMEM)

Thread-level parallelism (OpenMP, Pthreads)

Accelerator-based parallelism (CUDA, OpenCL)

Source code instrumentation

User instrumentation

Application

Future Features and Management

- Scalability to maximum available CPU core count
- 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

Hands-on:
NPB-MZ-MPI / BT

Score-P

allinea
JÜLICH FORSCHUNGSZENTRUM
Lawrence Livermore National Laboratory
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
German Research School for Simulation Sciences
RWTHAACHEN UNIVERSITY
TUM Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE

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

NPB-MZ-MPI / BT Instrumentation

```
$ module load intel impi
$ module load scorep

$ cp -r /home/courses/instructor06/NPB3.3-MZ-MPI_prepared ~
$ cd ~/NPB3.3-MZ-MPI_prepared
```

- Setup environment and load required modules
- Copy the prepared benchmark files to your home

NPB-MZ-MPI / BT Instrumentation

```
[...]
OPENMP = -openmp
[...]
#---------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------
#MPIF77 = mpiifort

# Alternative variants to perform instrumentation
[...]
MPIF77 = scorep --user mpiifort

# This links MPI Fortran programs; usually the same
as ${MPIF77}
[...]
```

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

Uncomment the Score-P compiler wrapper specification

NPB-MZ-MPI / BT Instrumented Build

```
$ make clean
$ make bt-mz CLASS=B NPROCS=2
[...]
scorep mpiifort -c  -O3 -fopenmp bt.f
scorep mpiifort -c  -O3 -fopenmp exact_solution.f
[...]
scorep mpiifort -O3 -fopenmp -o bt.o  initialize.o
exact_solution.o exact_rhs.o set_constants.o [...]

Built executable ../bin.scorep/bt-mz_B.2
make: Leaving directory 'BT-MZ`
```

- Return to root directory and clean-up
- Re-build executable using Score-P compiler wrapper
- Verify that the scorep command is used in the build

Measurement Configuration: scorep-info

```
$ scorep-info config-vars --full

SCOREP_ENABLE_PROFILING
  Description: Enable profiling
         Type: Boolean
      Default: true

SCOREP_ENABLE_TRACING
  Description: Enable tracing
         Type: Boolean
      Default: false

SCOREP_VERBOSE
  Description: Be verbose
         Type: Boolean
      Default: false

SCOREP_TOTAL_MEMORY
  Description: Total memory in bytes per process for the measurement system
         Type: Number with size suffixes
      Default: 16000k

[...]
```

- Score-P measurements are configured via environment variables
- Execute **scorep-info** to see a complete list

Summary Measurement Collection

```
$ cd bin.scorep
$ cp ../jobscript/froggy/scorep.slurm .
$ cat scorep.slurm
#!/bin/bash
[...]
#SBATCH --ntasks-per-socket=1
#SBATCH --cpus-per-task=10
#SBATCH -n 2

export NPB_MZ_BLOAD=0
export OMP_NUM_THREADS=10
[...]
srun ./bt-mz_B.2
```

- Change to the directory containing the new executable before running it with the desired configuration

Summary Measurement Collection

```
$ sbatch scorep.slurm
$ tail -F out.txt
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ
MPI+OpenMP Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     2

[...]
 Calculated speedup =     20.00

 Time step    1
 Time step   20
[...]
```

- Submit job
- Follow the output of the application run
- When finished, hit **ctrl+c**

BT-MZ Summary Analysis Report Examination

```
$ ls
bt-mz_B.2  scorep-XXXXXXXX out.txt scorep.slurm
$ ls scorep-XXXXXXXX
profile.cubex  scorep.cfg
```

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

- Interactive exploration with CUBE now shown by Alexandre

Congratulations!?

- If you made it this far, you successfully used Score-P to
  - instrument the application
  - analyze its execution with a summary measurement

- ... 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

- 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

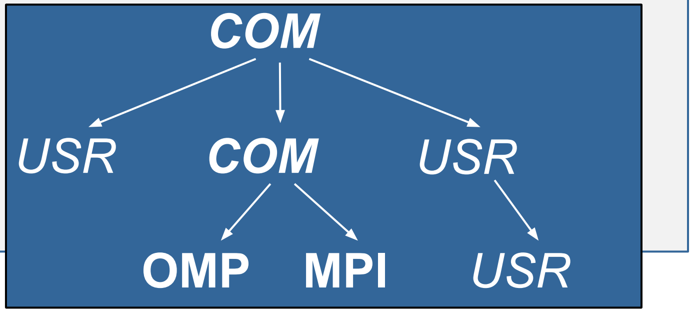
- 3.0 Event trace collection
- 3.1 Event trace examination & analysis

BT-MZ Summary Analysis Result Scoring

```
$ scorep-score scorep-XXXXXXXX/profile.cubex

Estimated aggregate size of event trace:                   40GB
Estimated requirements for largest trace buffer (max_buf): 20GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       20GB
(hint: When tracing set SCOREP_TOTAL_MEMORY=20GB 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 21,394,307,810 1,638,101,763 1250.51   100.0           0.76  ALL
          USR 21,282,804,114 1,631,137,675  496.31    39.7           0.30  USR
          OMP    109,117,376     6,781,952  752.33    60.2         110.93  OMP
          COM      2,351,570       180,890    1.01     0.1           5.59  COM
          MPI         34,750         1,246    0.86     0.1         688.57  MPI
```



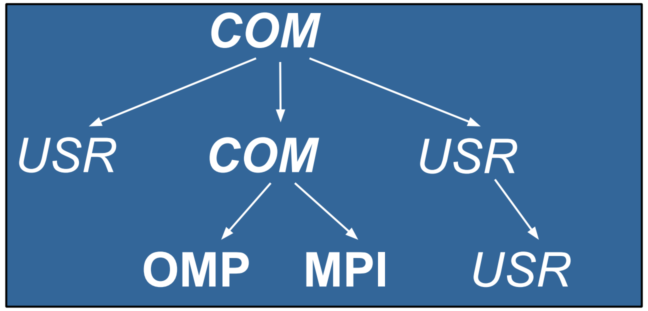
- Report scoring as textual output

40 GB total memory
20 GB per rank!

- Region/callpath classification
  - **MPI** pure MPI functions
  - **OMP** pure OpenMP regions
  - **USR** user-level computation
  - **COM** “combined” USR+OpenMP/MPI
  - **ANY/ALL** aggregate of all types

BT-MZ Summary Analysis Report Breakdown

```
$ scorep-score -r scorep-XXXXXXXX/profile.cubex
[...]
         USR  6,826,023,516   522,844,416  158.88     12.7           0.30
matmul_sub_
         USR  6,826,023,516   522,844,416  133.52     10.7           0.26
matvec_sub_
         USR  6,826,023,516   522,844,416  185.66     14.8           0.36  binvcrhs_
         USR    299,580,450    22,692,096    6.39      0.5           0.28  binvrhs_
         USR    299,580,450    22,692,096    7.83      0.6           0.35  lhsinit_
         USR    223,900,352    17,219,840    4.03      0.3           0.23
exact_solution_
         OMP      6,560,640       257,280    0.11      0.0           0.45  !$omp
parallel @exch_qbc.f:204
         OMP      6,560,640       257,280    0.12      0.0           0.46  !$omp
parallel @exch_qbc.f:255
         OMP      6,560,640       257,280    0.11      0.0           0.45  !$omp
parallel @exch_qbc.f:244
```



More than
19 GB just for these 3
regions

BT-MZ Summary Analysis Score

- Summary measurement analysis score reveals
  - Total size of event trace would be ~40 GB (for only 2 ranks!)
  - Maximum trace buffer size would be ~20 GB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.5% 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 40% 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

```
$ cat ../config/scorep.filt
SCOREP_REGION_NAMES_BEGIN EXCLUDE
binvcrhs*
matmul_sub*
matvec_sub*
exact_solution*
binvrhs*
lhs*init*
timer_*
```

- Filter files define code regions to exclude

- Can use wildcards to define regions

BT-MZ Summary Analysis Report Filtering

```
$ scorep-score -f ../config/scorep.filt -c 2 scorep-
XXXXXXXX/profile.cubex

Estimated aggregate size of event trace:                   213MB
Estimated requirements for largest trace buffer (max_buf): 107MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       127MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=127MB 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 21,394,307,810 1,638,101,763 1250.51   100.0           0.76  ALL
 -       USR 21,282,804,114 1,631,137,675  496.31    39.7           0.30  USR
 -       OMP    109,117,376     6,781,952  752.33    60.2         110.93  OMP
 -       COM      2,351,570       180,890    1.01     0.1           5.59  COM
 -       MPI         34,750         1,246    0.86     0.1         688.57  MPI
[...]
```

- Report scoring with filter file applied

213 MB of memory in total,
107 MB per rank!

BT-MZ Summary Analysis Report Filtering

```
$ scorep-score -r -f ../config/scorep.filt
scorep_8x4_sum/profile.cubex
flt type     max_buf[B]        visits time[s] time[%] time/visit[us] region
 -    ALL 21,377,442,117 6,554,106,201 4946.18   100.0           0.75 ALL
 -    USR 21,309,225,314 6,537,020,537 2326.51    47.0           0.36 USR
 -    OMP     65,624,896    16,327,168 2607.63    52.7         159.71 OMP
 -    COM      2,355,080       724,640    2.49     0.1           3.43 COM
 -    MPI        236,827        33,856    9.56     0.2         282.29 MPI

 *    ALL     68,216,855    17,085,673 2622.30    53.0         153.48 ALL-FLT
 +    FLT 21,309,225,262 6,537,020,528 2323.88    47.0           0.36 FLT
 -    OMP     65,624,896    16,327,168 2607.63    52.7         159.71 OMP-FLT
 *    COM      2,355,080       724,640    2.49     0.1           3.43 COM-FLT
 -    MPI        236,827        33,856    9.56     0.2         282.29 MPI-FLT
 *    USR             52             9    2.63     0.1      292158.12 USR-FLT

 +    USR  6,883,222,086 2,110,313,472  651.44    13.2           0.31
matvec_sub_
 +    USR  6,883,222,086 2,110,313,472  720.38    14.6           0.34
matmul_sub_
[...]
```

- Score report breakdown by region

Filtered routines marked with '+'

BT-MZ Filtered Summary Measurement

```
$ vim scorep.slurm
[...]
export NPB_MZ_BLOAD=0
export OMP_NUM_THREADS=10
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_TOTAL_MEMORY=300M

srun ./bt-mz_B.2

$ sbatch scorep.slurm
$ tail -F out.txt
```

- Now uncomment the filter file reference in the job script
- Submit job again
- New profile in new directory
- Note that the application now finishes in 15 sec, instead of 63

The End

- Community instrumentation & measurement infrastructure
  - Instrumentation (various methods)
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data
- Available under New BSD open-source license
- Documentation & Sources:
  - http://www.score-p.org
- User guide also part of installation:
  - `<prefix>/share/doc/scorep/{pdf,html}/`
- Support and feedback: support@score-p.org
- Subscribe to news@score-p.org, to be up to date