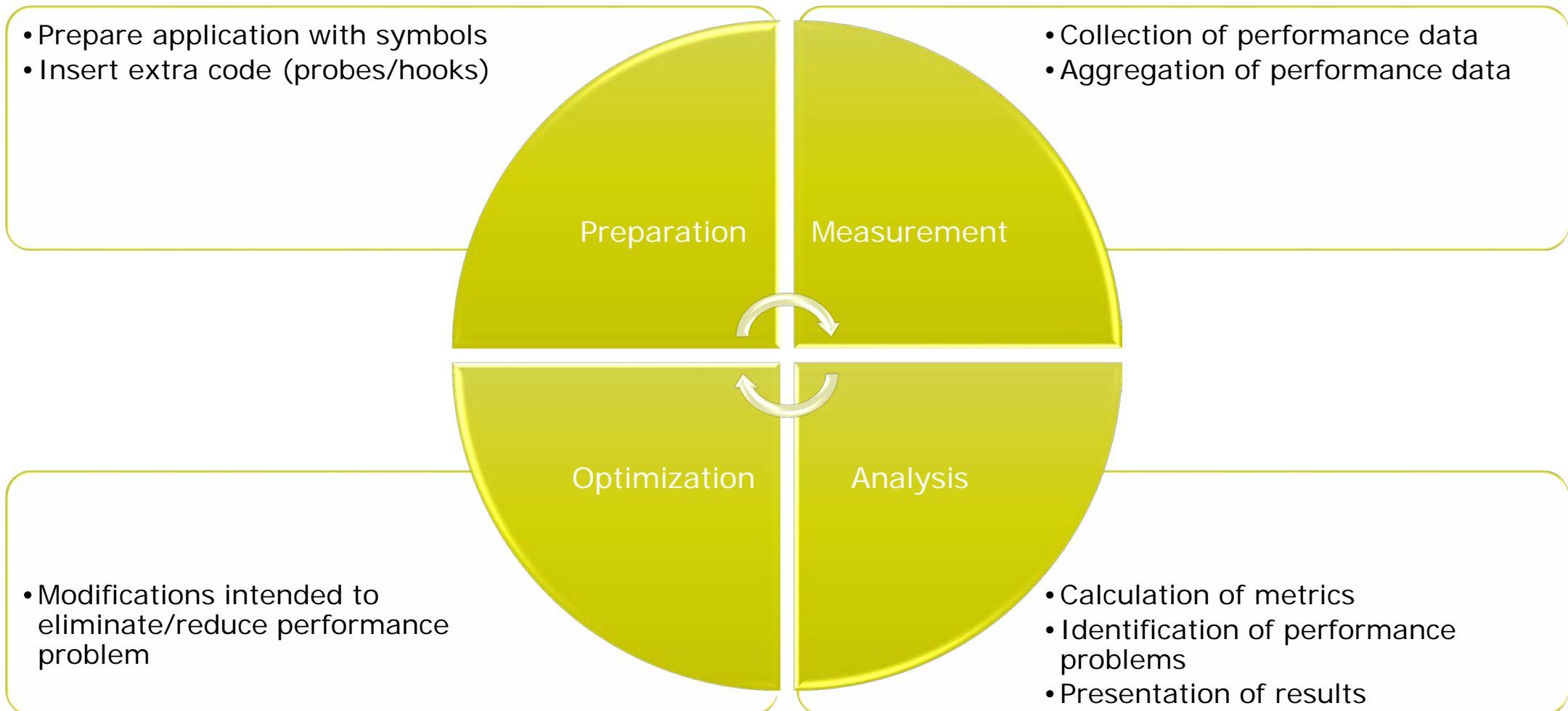


Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

VI-HPS Team



Performance engineering workflow



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

VampirTrace
OTF

Scalasca

EPILOG /
CUBE

TAU

TAU native
formats

Periscope

Online
measurement

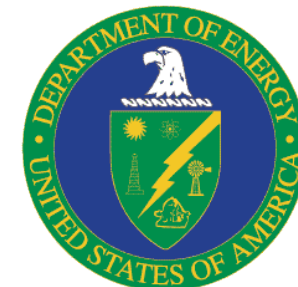
Score-P Project Idea

- 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



UNIVERSITY OF OREGON

Score-P Functionality

- Provide typical functionality for HPC performance tools
- Support all fundamental concepts of partner's tools

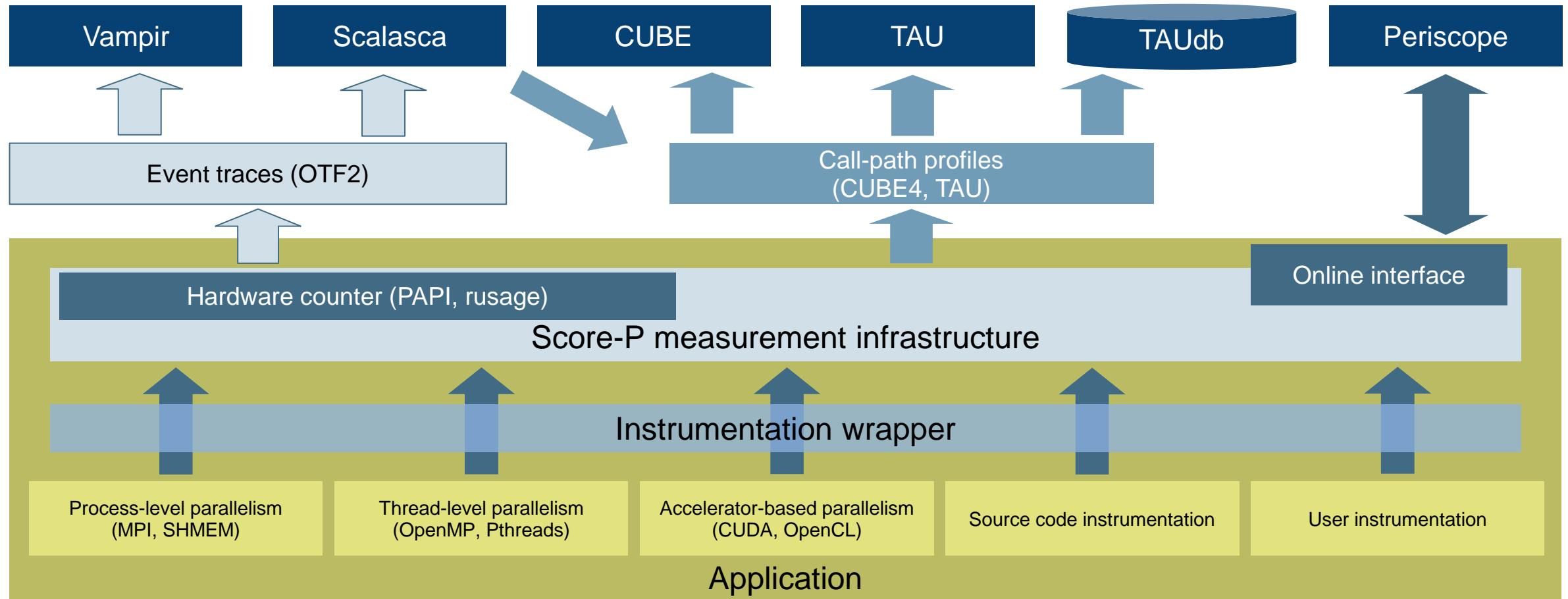
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
 - Basic and advanced profile generation
 - Event trace recording
 - Online access to profiling data

- 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



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



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

```
% source /applis/site/env.bash
% module load intel-devel
% module use /home/PROJECTS/pr-vi-hps-tw18/opt/mf
% module load scorep/1.4.1-intel-impi
% pwd
/scratch/$USER/NPB3.3-MZ-MPI/bin
```

- Setup environment and load required modules

NPB-MZ-MPI / BT Instrumentation

```
#           SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#-----
# Items in this file may need to be changed for each platform.
#-----
COMPFLAGS = -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}
FLINK    = $(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

```
% cd ..
% make clean

% make bt-mz CLASS=C NPROCS=8
cd BT-MZ; make CLASS=C NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c -lm
../sys/setparams bt-mz 4 B
scorep mpiifort -c -O3 -fopenmp bt.f
[...]
cd ../common; scorep mpiifort -c -O3 -fopenmp timers.f
scorep mpiifort -O3 -fopenmp -o ../bin.scorep/bt-mz_B.4 \
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.8
make: Leaving directory 'BT-MZ'
```

- Return to root directory and clean-up
- Re-build executable using Score-P compiler wrapper

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 environmental variables:

Summary Measurement Collection

```
% cd bin.scorep
% cp ../jobscript/froggy/scorep.oar .
% vim scorep.oar
export NPB_MZ_BLOAD=0
export OMP_NUM_THREADS=4
CLASS=C
NPROCS=8
EXE=./bt-mz_${CLASS}.${NPROCS}
export SCOREP_EXPERIMENT_DIRECTORY=\
scorep_${NPROCS}x${OMP_NUM_THREADS}_sum
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_TOTAL_MEMORY=300M

# launch
mpiexec.hydra -genvall -n $NPROCS $EXE

% oarsub -S ./scorep.oar
```

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

Comment these lines for the moment

- Submit job

Summary Measurement Collection

```
% less scorep_167691

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:    8

Use the default load factors with threads
Total number of threads:    32 ( 4.0 threads/process)

Calculated speedup =    31.99

Time step    1

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

- Check the output of the application run

BT-MZ Summary Analysis Report Examination

```
% ls
bt-mz_C.8  scorep_167691  scorep_8x4_sum
% ls scorep_8x4_sum
profile.cubex  scorep.cfg

% cube scorep_8x4_sum/profile.cubex

[CUBE GUI showing summary analysis report]
```

- 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

Congratulations!?

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

- 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_8x4_sum/profile.cubex
```

Estimated aggregate size of event trace:

Estimated requirements for largest trace buffer (max_buf):

Estimated memory requirements (SCOREP_TOTAL_MEMORY):

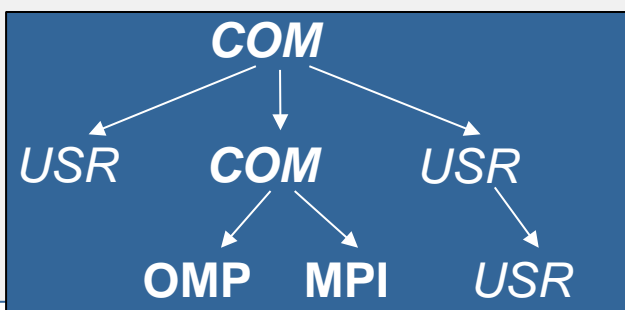
(hint: When tracing set SCOREP_TOTAL_MEMORY=20GB to avoid intermediate flushes or reduce requirements using USR regions filters.)

flt	type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
	ALL	21,377,442,117	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

159 GB
20 GB
20 GB

- Report scoring as textual output

159 GB total memory
20 GB per rank!



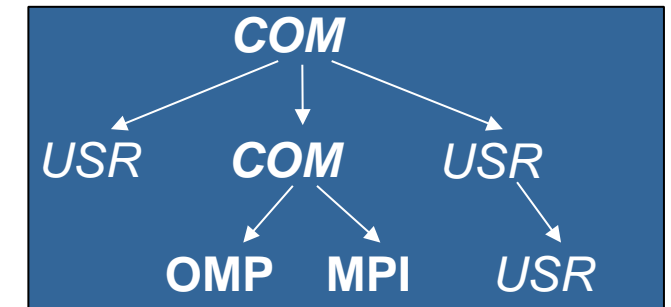
- Region/callpath classification
 - MPI pure MPI functions
 - OMP pure OpenMP regions
 - USR user-level computation
 - COM "combined" USR+OpenMP/MPI
 - ANY/ALL aggregate of all region types

BT-MZ Summary Analysis Report Breakdown

```
% scorep-score -r 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

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_
USR	6,883,222,086	2,110,313,472	881.32	17.8	0.42	binvrhs_
USR	293,617,584	87,475,200	29.93	0.6	0.34	binvrhs_
USR	293,617,584	87,475,200	33.03	0.7	0.38	lhsinit_
USR	101,320,128	31,129,600	7.78	0.2	0.25	exact_solution_



More than
18 GB just for these 6
regions

BT-MZ Summary Analysis Score

- Summary measurement analysis score reveals
 - Total size of event trace would be ~159 GB
 - Maximum trace buffer size would be ~20 GB per rank
 - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
 - 99.8% 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 32% 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
binvrhs*
matmul_sub*
matvec_sub*
exact_solution*
binvrhs*
lhs*init*
timer_*

% scorep-score -f ../config/scorep.filt -c 2 \
>scorep_8x4_sum/profile.cubex

Estimated aggregate size of event trace:
Estimated requirements for largest trace buffer (max_buf):
Estimated memory requirements (SCOREP_TOTAL_MEMORY):
(hint: When tracing set SCOREP_TOTAL_MEMORY=78MB to avoid \
>intermediate flushes
or reduce requirements using USR regions filters.)
```

521 MB
66 MB
78 MB

- Report scoring with prospective filter listing 6 USR regions

521 MB of memory in total,
66 MB per rank!

(Including 2 metric values)

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
+   USR  6,883,222,086 2,110,313,472 881.32  17.8       0.42 binvrhs_
+   USR   293,617,584  87,475,200  29.93   0.6       0.34 binvrhs_
+   USR   293,617,584  87,475,200  33.03   0.7       0.38 lhsinit_
+   USR   101,320,128  31,129,600   7.78   0.2       0.25 exact_solution_
```

- Score report breakdown by region

Filtered routines marked with '+'

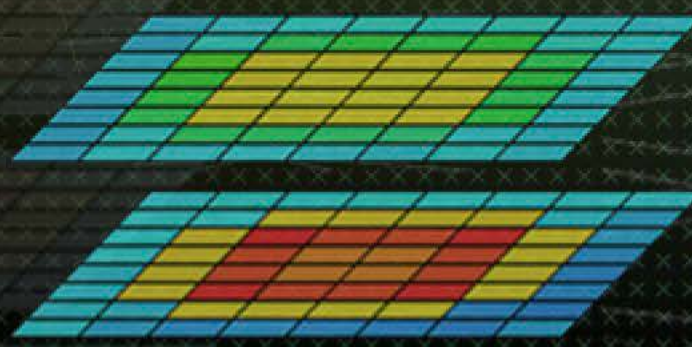
BT-MZ Filtered Summary Measurement

```
% cd bin.scorep
% cp ../jobscript/froggy/scorep.oar .
% vim scorep.oar
export NPB_MZ_BLOAD=0
export OMP_NUM_THREADS=4
CLASS=C
NPROCS=8
EXE= ./bt-mz_ $CLASS.$NPROCS
export SCOREP_EXPERIMENT_DIRECTORY=\
scorep_${NPROCS}x${OMP_NUM_THREADS}_sum_filter
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_TOTAL_MEMORY=300M

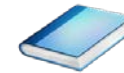
# launch
mpiexec.hydra -genvall -n $NPROCS $EXE

% oarsub -S ./scorep.oar
```

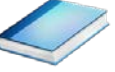
- Set new experiment directory and re-run measurement with new filter configuration
- Submit job



Score-P: Advanced Measurement Configuration



Advanced Measurement Configuration: Metrics



- Available PAPI metrics
 - Preset events: common set of events deemed relevant and useful for application performance tuning
 - Abstraction from specific hardware performance counters, mapping onto available events done by PAPI internally

```
% papi_avail
```

- Native events: set of all events that are available on the CPU (platform dependent)

```
% papi_native_avail
```

Note:

Due to hardware restrictions

- number of concurrently recorded events is limited
- there may be invalid combinations of concurrently recorded events

Advanced Measurement Configuration: Metrics



```
% man getrusage
struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long ru_maxrss; /* maximum resident set size */
    long ru_ixrss; /* integral shared memory size */
    long ru_idrss; /* integral unshared data size */
    long ru_isrss; /* integral unshared stack size */
    long ru_minflt; /* page reclaims (soft page faults) */
    long ru_majflt; /* page faults (hard page faults) */
    long ru_nswap; /* swaps */
    long ru_inblock; /* block input operations */
    long ru_oublock; /* block output operations */
    long ru_msgsnd; /* IPC messages sent */
    long ru_msrvcv; /* IPC messages received */
    long ru_nsignals; /* signals received */
    long ru_nvcsw; /* voluntary context switches */
    long ru_nivcsw; /* involuntary context switches */
};
```

- Available resource usage metrics
- **Note:**
 - (1) Not all fields are maintained on each platform.
 - (2) Check scope of metrics (per process vs. per thread)

Advanced Measurement Configuration: CUDA

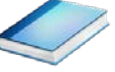


- Record CUDA events with the CUPTI interface

```
% export SCOREP_CUDA_ENABLE=gpu,kernel,idle
```

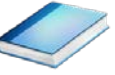
- All possible recording types
 - runtime CUDA runtime API
 - driver CUDA driver API
 - gpu GPU activities
 - kernel CUDA kernels
 - idle GPU compute idle time
 - memcpy CUDA memory copies

Score-P User Instrumentation API



- Can be used to mark initialization, solver & other phases
 - Annotation macros ignored by default
 - Enabled with [--user] flag
- Appear as additional regions in analyses
 - Distinguishes performance of important phase from rest
- Can be of various type
 - E.g., function, loop, phase
 - See user manual for details
- Available for Fortran / C / C++

Score-P User Instrumentation API (Fortran)



```
#include "scorep/SCOREP_User.inc"

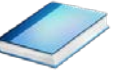
subroutine foo(...)
  ! Declarations
  SCOREP_USER_REGION_DEFINE( solve )

  ! Some code...
  SCOREP_USER_REGION_BEGIN( solve, "<solver>", \
                           SCOREP_USER_REGION_TYPE_LOOP )

  do i=1,100
    [...]
  end do
  SCOREP_USER_REGION_END( solve )
  ! Some more code...
end subroutine
```

- Requires processing by the C preprocessor

Score-P User Instrumentation API (C/C++)

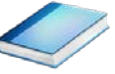


```
#include "scorep/SCOREP_User.h"

void foo()
{
    /* Declarations */
    SCOREP_USER_REGION_DEFINE( solve )

    /* Some code... */
    SCOREP_USER_REGION_BEGIN( solve, "<solver>",
                             SCOREP_USER_REGION_TYPE_LOOP )
    for (i = 0; i < 100; i++)
    {
        [...]
    }
    SCOREP_USER_REGION_END( solve )
    /* Some more code... */
}
```

Score-P User Instrumentation API (C++)



```
#include "scorep/SCOREP_User.h"

void foo()
{
    // Declarations

    // Some code...
    {
        SCOREP_USER_REGION( "<solver>",
                           SCOREP_USER_REGION_TYPE_LOOP )
        for (i = 0; i < 100; i++)
        {
            [...]
        }
    }
    // Some more code...
}
```

Score-P Measurement Control API



- Can be used to temporarily disable measurement for certain intervals
 - Annotation macros ignored by default
 - Enabled with [--user] flag

```
#include "scorep/SCOREP_User.inc"

subroutine foo(...)
  ! Some code...
  SCOREP_RECORDING_OFF()
  ! Loop will not be measured
  do i=1,100
    [...]
  end do
  SCOREP_RECORDING_ON()
  ! Some more code...
end subroutine
```

Fortran (requires Cpreprocessor)

```
#include "scorep/SCOREP_User.h"

void foo(...) {
  /* Some code... */
  SCOREP_RECORDING_OFF()
  /* Loop will not be measured */
  for (i = 0; i < 100; i++) {
    [...]
  }
  SCOREP_RECORDING_ON()
  /* Some more code... */
}
```

C / C++

Further Information

- 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