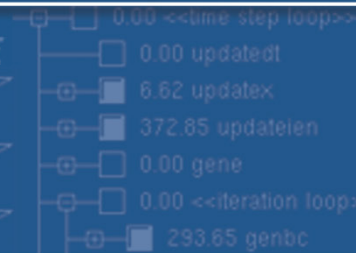


# VI-HPS

SOFTWARE



FAST SOLUTIONS

- ☒ PAPI\_L1\_DCM
- ☒ PAPI\_L1\_ICM
- ☐ PAPI\_L2\_DCM
- ☒ PAPI\_L2\_ICM
- ☒ PAPI\_L2\_TCM
- ☐ PAPI\_L2\_TCM

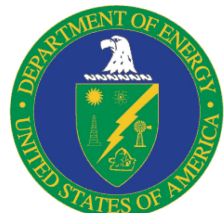
PRODUCTIVITY

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

- 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



- 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
- SILC project funded by BMBF
- Close collaboration PRIMA project funded by DOE



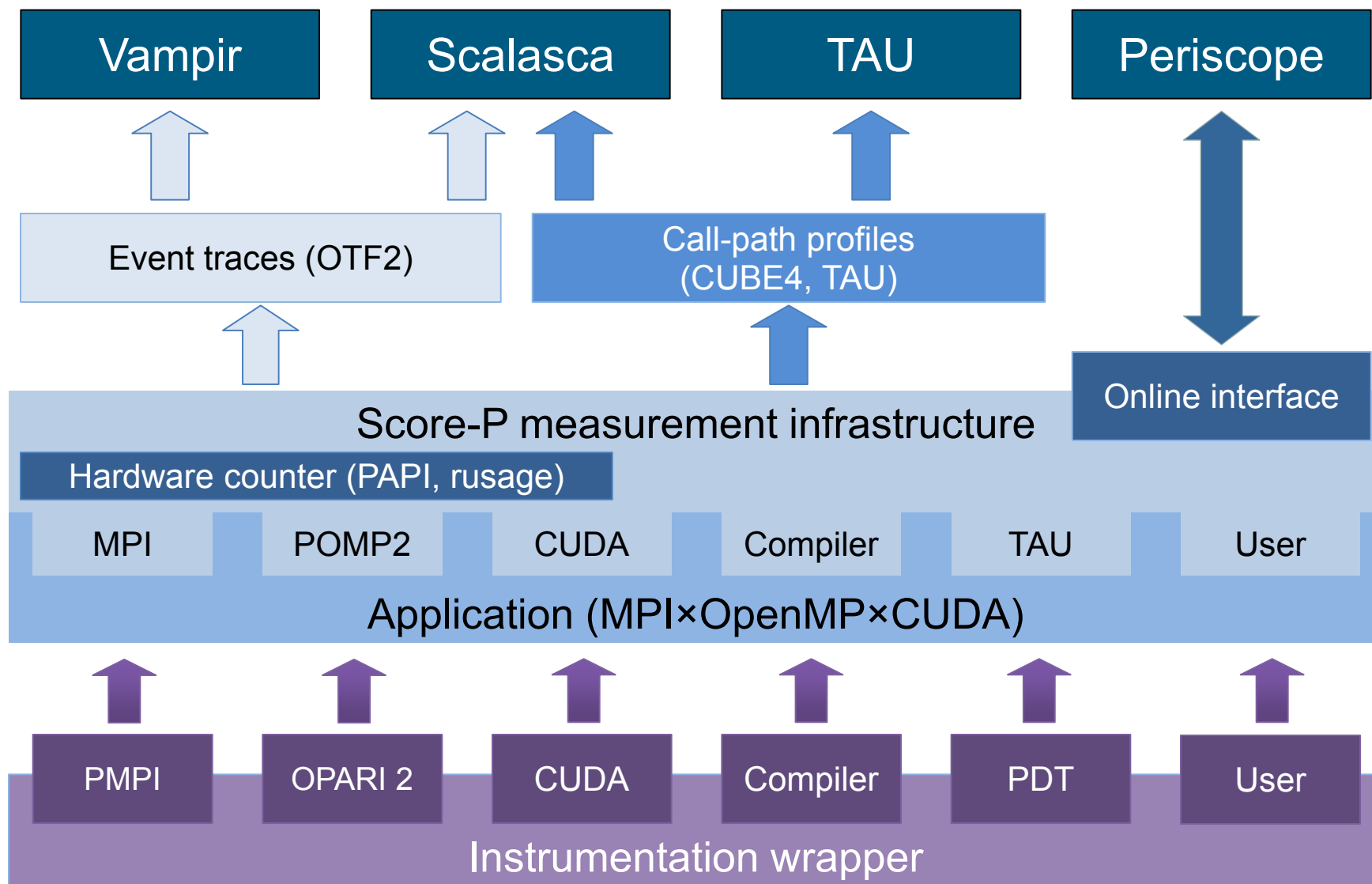
- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA



UNIVERSITY OF OREGON

- 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, OpenMP, and hybrid parallelism (and serial)
- Enhanced functionality (OpenMP 3.0, CUDA, highly scalable I/O)

- 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, OpenMP, basic CUDA, and all combinations
    - Later also OpenCL/HMPP/PTHREAD/...
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Easy and uniform installation through UNITE framework
  - Robustness
  - Open Source: New BSD License



- Scalability to maximum available CPU core count
- Support for OpenCL, HMPP, PTHREAD
- 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



# VI-HPS

SOFTWARE



0.00 <<time step loop>>  
0.00 updatedt  
6.62 updatex  
372.85 updateien  
0.00 gene  
0.00 <<iteration loop>>  
293.65 genbc



PRODUCTIVITY

FAST SOLUTIONS

☒ PAPI\_L1\_DCM  
☒ PAPI\_L1\_ICM  
☐ PAPI\_L2\_DCM  
☒ PAPI\_L2\_ICM  
☒ PAPI\_L2\_TCM  
☐ PAPI\_L2\_TCM

## Score-P hands-on: NPB-MZ-MPI / BT

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
8. Event trace collection
9. Event trace examination & analysis

- Load modules:

```
% module use /opt/vi-hps/UNITE/local

% module load UNITE
UNITE loaded

% module load VI-HPS-TW
VI-HPS-TW loaded
cube/4.2.3 loaded
scorep/1.3 loaded
scalasca/2.1 loaded
vampir/8.2.0 loaded

% module list
Currently Loaded Modulefiles:
  1) UNITE/1.1          2) cube/4.2.3        3) scorep/1.3
  4) scalasca/2.1      5) vampir/8.2.0     6) VI-HPS-TW
```

- Edit `config/make.def` to adjust 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 = mpifrtpx

# Alternative variants to perform instrumentation
...
MPIF77 = scorep --user mpifrtpx

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

Uncomment the  
Score-P compiler  
wrapper specification

- Return to root directory and clean-up

```
% make clean
```

- Re-build executable using Score-P compiler wrapper

```
% make bt-mz CLASS=B NPROCS=8
cd BT-MZ; make CLASS=B NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c -lm
../sys/setparams bt-mz 8 B
scorep --user mpifrtpx -c -Fwide -Kfast -Kopenmp bt.f
[...]
cd ../common; scorep --user mpifrtpx -c -Kfast -Kopenmp timers.f
scorep --user mpifrtpx -Kfast -Kopenmp -o ../bin.scorep/bt-mz_B.8 \
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_B.8
make: Leaving directory 'BT-MZ'
```

Tip: *make suite*

- Score-P measurements 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 ...
```

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

```
% cd bin.scorep
% cp ../jobscript/fx10/scorep.sh .
% vi scorep.sh
```

```
...
#
# Score-P configuration
#
export SCOREP_EXPERIMENT_DIRECTORY=scorep_sum
...
```

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

```
% pjsub scorep.sh
% cat scorep.sh.o<id>

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 =      31.52

Time step      1

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



- Creates experiment directory `./scorep_sum` containing
  - a record of the measurement configuration (`scorep.cfg`)
  - the analysis report that was collated after measurement (`profile.cubex`)

```
% ls
bt-mz_B.8  scorep_sum
% ls scorep_sum
profile.cubex  scorep.cfg
```

- Interactive exploration with CUBE / ParaProf

```
% cube scorep_sum/profile.cubex

[CUBE GUI showing summary analysis report]

% paraprof scorep_sum/profile.cubex

[TAU ParaProf GUI showing summary analysis report]
```

- 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

- Report scoring as textual output

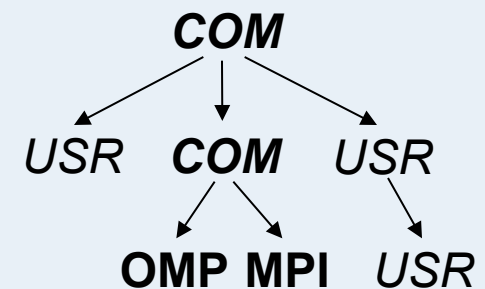
```
% scorep-score scorep_sum/profile.cubex
Estimated aggregate size of event trace: 42504345994 bytes
Estimated requirements for largest trace buffer (max_buf): 5372038254 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY=5126MB to avoid intermediate flushes
or reduce requirements using file listing names of USR regions to be filtered.)
```

flt	type	max_buf	visits	time	%time	region
	ALL	5372038254	1634100763	12406.32	100.0	ALL
	USR	5358738138	1631138913	5873.05	47.3	USR
	OMP	12389148	2743808	6465.96	52.1	
	COM	673610	185320	8.31	0.1	
	MPI	237358	32722	58.99	0.5	

40 GB total memory  
5.0 GB per rank!

- 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)



- Score report breakdown by region

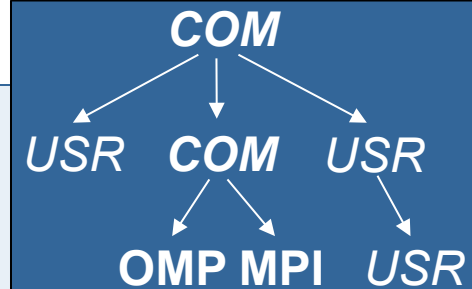
```
% scorep-score -r scorep_sum/profile.cubex
```

```
[...]
```

flt	type	max_buf	visits	time	%time	region
	ALL	5372038254	1634100763	12406.32	100.0	ALL
	USR	5358738138	1631138913	5873.05	47.3	USR
	OMP	12389148	2743808	6465.96	52.1	OMP
	COM	673610	185320	8.31	0.1	COM
	MPI	237358	32722	58.99	0.5	MPI

USR	716505830	522844416	2068.66	16.7	binvrhs_
USR	716505830	522844416	1788.68	14.4	matmul_sub_
USR	716505830	522844416	1787.72	14.4	matvec_sub_
USR	76195080	22692096	75.07	0.6	binvrhs_
USR	76195080	22692096	93.68	0.8	lhsinit_
USR	56825184	17219840	59.24	0.5	exact_solution_
OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:244
OMP	738072	102912	0.73	0.0	!\$omp parallel @exch_qbc.F:255
OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:215
OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:204
OMP	376272	102912	1.26	0.0	!\$omp do @exch_qbc.F:255
OMP	376272	102912	0.56	0.0	!\$omp implicit barrier @exch..

```
[...]
```



99.75% of  
*max\_buf* just for  
these 6 regions

- Summary measurement analysis score reveals
  - Total size of event trace would be ~40 GB
  - Maximum trace buffer size would be ~5.0 GB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.75% 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 47% 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

- Report scoring with prospective filter listing patterns for 6 USR regions (and uninteresting timer routines)

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

% scorep-score -f ../config/scorep.filt scorep_sum/profile.cubex
Estimated aggregate size of event trace: 94771514 bytes
Estimated requirements for largest trace buffer (max_buf): 13304770 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY=16MB to avoid intermediate flushes
or reduce requirements using file listing names of USR regions to be filtered.)
```

90 MB of trace data in total,  
12.7 MB per rank!

- Score report breakdown by region

```
% scorep-score -r -f ../config/scorep.filt scorep_sum/profile.cubex
[...]
```

flt	type	max_buf	visits	time	%time	region
*	ALL	13304770	2963283	6533.28	52.7	ALL-FLT
+	FLT	5358733484	1631137480	5873.04	47.3	FLT
-	OMP	12389148	2743808	6465.96	52.1	OMP-FLT
*	COM	673610	185320	8.31	0.1	COM-FLT
-	MPI	237358	32722	58.99	0.5	MPI-FLT
*	USR	4680	1433	0.01	0.0	USR-FLT
+	USR	716505830	522844416	2068.66	16.7	binvrhs_
+	USR	716505830	522844416	1788.68	14.4	matmul_sub_
+	USR	716505830	522844416	1787.72	14.4	matvec_sub_
+	USR	76195080	22692096	75.07	0.6	binvrhs_
+	USR	76195080	22692096	93.68	0.8	lhsinit_
+	USR	56825184	17219840	59.24	0.5	exact_solution_
-	OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:244
-	OMP	738072	102912	0.73	0.0	!\$omp parallel @exch_qbc.F:255
-	OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:215
-	OMP	738072	102912	0.74	0.0	!\$omp parallel @exch_qbc.F:204
-	OMP	376272	102912	1.26	0.0	!\$omp do @exch_qbc.F:255
-	OMP	376272	102912	0.56	0.0	!\$omp implicit barrier @exch..

```
[...]
```

Filtered  
routines  
marked  
with '+'

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

- Edit job script

```
% vi scorep.sh
```

- Adjust configuration

```
...  
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_sum_with_filter  
% export SCOREP_FILTERING_FILE=../config/scorep.filt  
...
```

- Submit job

```
% pjsub scorep.sh
```



- Scoring of new analysis report as textual output

```
% scorep-score scorep_sum_with_filter/profile.cubex
Estimated aggregate size of event trace:                94771514 bytes
Estimated requirements for largest trace buffer (max_tbc): 13304770 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY=16MB to avoid intermediate flushes
or reduce requirements using file listing names of USR regions to be filtered.)
```

flt	type	max_buf	visits	time	%time	region
	ALL	13304770	2963283	851.53	100.0	ALL
	OMP	12389148	2743808	822.79	96.6	OMP
	COM	673610	185320	7.96	0.9	COM
	MPI	237358	32722	20.77	2.4	MPI
	USR	4680	1433	0.01	0.0	USR

- Significant reduction in runtime (measurement overhead)
  - Not only reduced time for USR regions, but MPI/OMP reduced too!
- Further measurement tuning (filtering) may be appropriate
  - e.g., use “timer\_\*” to filter timer\_start\_, timer\_read\_, etc.

- Recording hardware counters via PAPI

```
% export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
```

- Also possible to record them only per rank

```
% export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L2_TCM
```

- Recording operating system resource usage

```
% export SCOREP_METRIC_RUSAGE=ru_stime  
% export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
```

- 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

- Available resource usage metrics

```
% man getrusage
```

```
[... Output ...]
```

```
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_msgrcv;      /* IPC messages received */  
    long    ru_nsignals;    /* signals received */  
    long    ru_nvcsw;       /* voluntary context switches */  
    long    ru_nivcsw;      /* involuntary context switches */  
};
```

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

Note:

- (1) Not all fields are maintained on each platform.
- (2) Check scope of metrics (per process vs. per thread)

1. Reference preparation for validation
2. Program instrumentation
3. Summary measurement collection
4. Summary analysis report examination
5. Summary experiment scoring
6. Summary measurement collection with filtering
7. Filtered summary analysis report examination
- 8. Event trace collection**
- 9. Event trace examination & analysis**

- 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

- Re-run the application using the tracing mode of Score-P
  - Edit [scorep.sh](#) to adjust configuration

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_trace_with_metrics
% export SCOREP_FILTERING_FILE=./config/scorep.filt
% export SCOREP_ENABLE_TRACING=true
% export SCOREP_ENABLE_PROFILING=false
% export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
% export SCOREP_TOTAL_MEMORY=36M
```

max\_buf bytes from  
*scorep-score -c 2*  
for 2 metric counters

- Submit job

```
% pjsub scorep.sh
```

- Separate trace file per thread written directly into new experiment directory `./scorep_trace`
- Interactive trace exploration with Vampir

```
% vampir scorep_trace_with_metrics/traces.otf2
```

- Record only for subset of the MPI functions events

```
% export SCOREP_MPI_ENABLE_GROUPS=cg,coll,p2p,xnonblock
```

- All possible sub-groups
  - cg Communicator and group management
  - coll Collective functions
  - env Environmental management
  - err MPI Error handling
  - ext External interface functions
  - io MPI file I/O
  - misc Miscellaneous
  - perf PControl
  - p2p Peer-to-peer communication
  - rma One sided communication
  - spawn Process management
  - topo Topology
  - type MPI datatype functions
  - xnonblock Extended non-blocking events
  - xreqtest Test events for uncompleted requests



- 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++

```
#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

```
#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... */
}
```

```
#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...
}
```

- 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 C preprocessor)

```
#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++

### scorep-1.3 (target release 2014Q1)

- *support for Fujitsu FX10 & K computer*
  - *probably with exception of C++ with OpenMP*
- support for CUDA 5.5 & 6.0, Pthreads & OpenSHMEM
- extended profile metrics (statistics)
- improved OTF2 support for SIONlib
  - only supports MPI for now

### scorep-1.4 (target release 2014Q2)

- support for Intel Xeon Phi (MIC) native & symmetric modes
- OTF2 use of SIONlib for all measurement paradigms
- recording topologies (hardware, virtual & user-defined)
- ...

### Score-P

- 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}/`
- Contact: [info@score-p.org](mailto:info@score-p.org)
- Bugs: [scorep-bugs@groups.tu-dresden.de](mailto:scorep-bugs@groups.tu-dresden.de)