

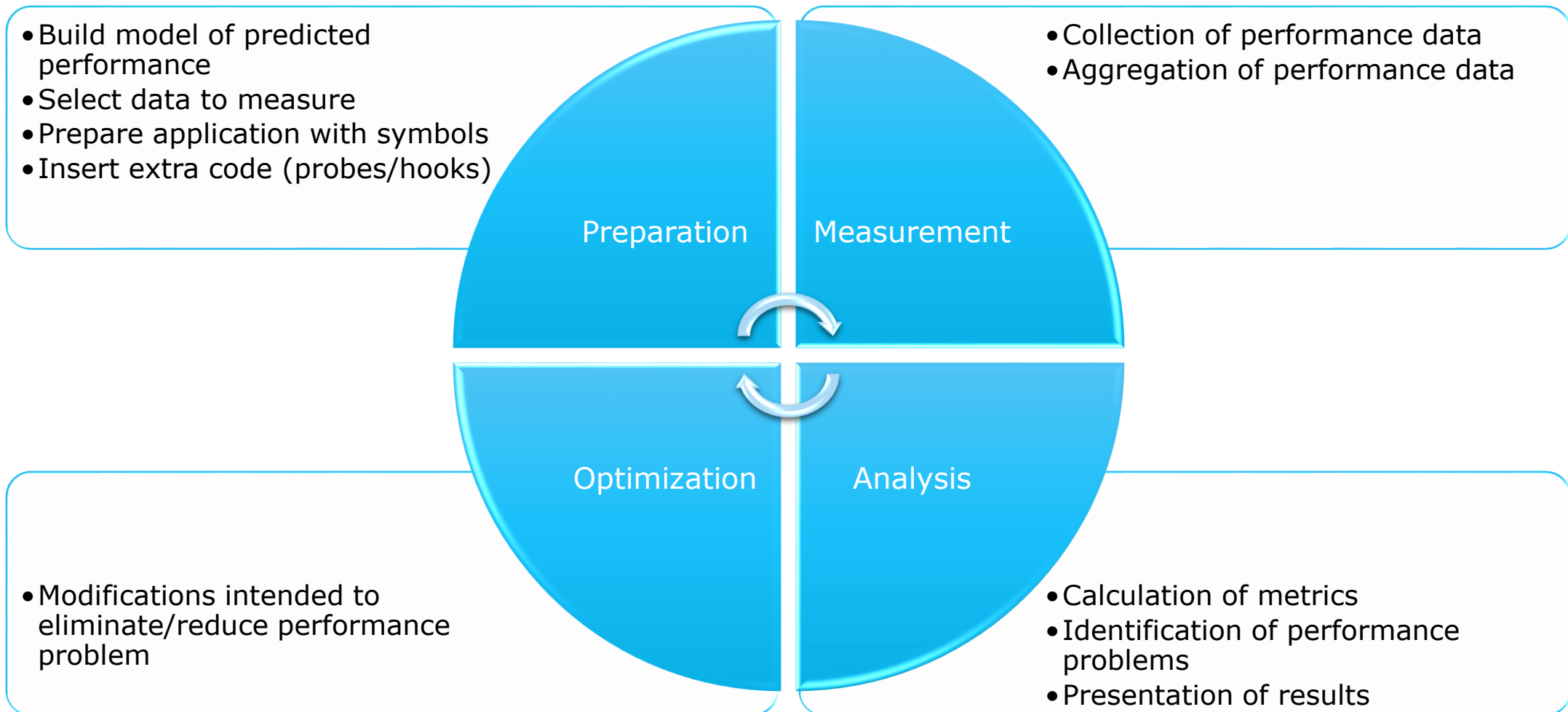
# Score-P – A Joint Performance Measurement Run-Time Infrastructure for Scalasca, TAU, and Vampir

---

VI-HPS Team



# Performance engineering workflow





## Score-P

---

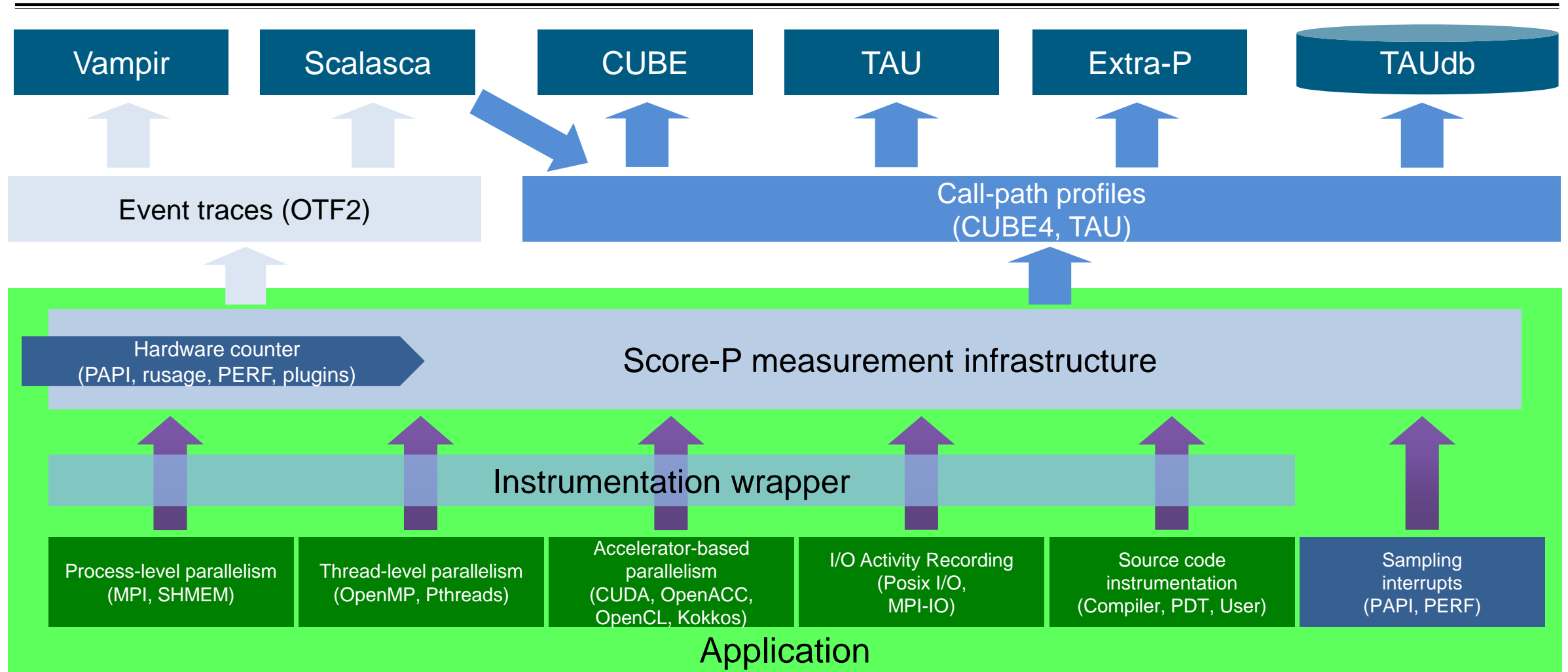
- Infrastructure for instrumentation and performance measurements
- Instrumented application can be used to produce several results:
  - Call-path profiling: CUBE4 data format used for data exchange
  - Event-based tracing: OTF2 data format used for data exchange
- Supported parallel paradigms:
  - Multi-process: MPI, SHMEM
  - Thread-parallel: OpenMP, Pthreads
  - Accelerator-based: CUDA, OpenCL, OpenACC, Kokkos
- Open Source; portable and scalable to all major HPC systems
- Initial project funded by BMBF
- Further developed in multiple 3<sup>rd</sup>-party funded projects

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung

# Score-P overview



# 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



## Hands-on: NPB-MZ-MPI / bt-mz\_C.8

---



# 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

## Tutorial exercise objectives

---

- Familiarise with usage of VI-HPS tools
  - complementary tools' capabilities & interoperability
- Prepare to apply tools productively to *your* applications(s)
- Exercise is based on a small portable benchmark code
  - unlikely to have significant optimisation opportunities
- Optional (recommended) exercise extensions
  - analyse performance of alternative configurations
  - investigate effectiveness of system-specific compiler/MPI optimisations and/or placement/binding/affinity capabilities
  - investigate scalability and analyse scalability limiters
  - compare performance on different HPC platforms
  - ...



## Compiler and MPI modules (Archer2)

---

- Select modules for the PrgEnv-gnu tool chain

```
% module swap PrgEnv-cray PrgEnv-gnu
```

Default is PrgEnv-cray

- Copy tutorial sources to your “WORK” directory

```
% cd /work/ta066/ta066/$USER  
% tar zxvf /work/y23/shared/tutorial/NPB3.3-MZ-MPI.tar.gz  
% cd NPB3.3-MZ-MPI
```

Use “WORK” for building and submitting

- Directory for data exchange during the workshop

```
% /work/ta066/ta066/shared/
```

## NPB-MZ-MPI Suite

---

- The NAS Parallel Benchmark suite (MPI+OpenMP version)

- Available from:

<http://www.nas.nasa.gov/Software/NPB>

- 3 benchmarks in Fortran77
- Configurable for various sizes & classes
- Move into the NPB3.3-MZ-MPI root directory

```
% ls
bin/      common/  jobscript/  Makefile  README.install  SP-MZ/
BT-MZ/    config/  LU-MZ/      README    README.tutorial  sys/
```

- Subdirectories contain source code for each benchmark
  - plus additional configuration and common code
- The provided distribution has already been configured for the tutorial, such that it is ready to “make” one or more of the benchmarks
  - but config/make.def may first need to be adjusted to specify appropriate compiler flags

# NPB-MZ-MPI / BT: config/make.def

```
#           SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.
#
#-----
#-----
# Configured for PrgEnv-gnu
#-----
#COMPFLAGS = -homp -G2 -N 255 # cce
COMPFLAGS = -fopenmp -fallow-argument-mismatch -ffixed-line-length-none # gnu
...

#-----
# The Fortran compiler used for MPI programs
#-----
MPIF77 = ftn

# Alternative variants to perform instrumentation
...

#MPIF77 = scorep --user ftn
...
```

Uncomment COMPILER flags  
according to current environment

Default (no instrumentation)

Hint: uncomment a compiler  
wrapper to do instrumentation

# Building an NPB-MZ-MPI Benchmark

```
% make
=====
=      NAS PARALLEL BENCHMARKS 3.3      =
=      MPI+OpenMP Multi-Zone Versions   =
=      F77                                =
=====

To make a NAS multi-zone benchmark type

    make <benchmark-name> CLASS=<class> NPROCS=<nprocs>

where <benchmark-name> is "bt-mz", "lu-mz", or "sp-mz"
      <class>           is "S", "W", "A" through "F"
      <nprocs>         is number of processes

[...]

*****
* Custom build configuration is specified in config/make.def *
* Suggested tutorial exercise configuration for Meggie:      *
*      make bt-mz CLASS=C NPROCS=8                          *
*****
```

- Type "make" for instructions

# Building an NPB-MZ-MPI Benchmark

```
% make bt-mz CLASS=B NPROCS=28
make[1]: Entering directory `BT-MZ'
make[2]: Entering directory `sys'
cc -o setparams setparams.c -lm
make[2]: Leaving directory `sys'
../sys/setparams bt-mz 8 C
make[2]: Entering directory `../BT-MZ'
ftn -g -c -O3 -fopenmp bt.f
[...]
ftn -g -c -O3 -fopenmp mpi_setup.f
cd ../common; ftn -g -c -O3 -fopenmp print_results.f
cd ../common; ftn -g -c -O3 -fopenmp timers.f
ftn -g -O3 -fopenmp -o ../bin/bt-mz_C.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
make[2]: Leaving directory `BT-MZ'
Built executable ../bin/bt-mz_C.8
make[1]: Leaving directory `BT-MZ'
```

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the benchmark class (S, W, A, B, C, D, E): **CLASS=C**
  - the number of MPI processes: **NPROCS=8**

Shortcut: `% make suite`

## NPB-MZ-MPI / BT (Block Tridiagonal Solver)

---

- What does it do?
  - Solves a discretized version of the unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Performs 200 time-steps on a regular 3-dimensional grid
- Implemented in 20 or so Fortran77 source modules
  
- Uses MPI & OpenMP in combination
  - 8 processes each with 6 threads should be reasonable for 1 compute node of ARCHER2
  - bt-mz\_C.8 should run in less than 12 seconds with the PrgEnv-gnu toolchain

## NPB-MZ-MPI / BT Reference Execution

```
% cd bin
% cp ../jobscript/archer2/run.sbatch .
% less run.sbatch
% sbatch run.sbatch

% cat slurm-<job_id>.out
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones: 16 x 16
Iterations: 200 dt: 0.000300
Number of active processes: 8
Use the default load factors with threads
Total number of threads: 48 ( 6.0 threads/process)

Time step 1
Time step 20
[...]
Time step 180
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 11.67
```

- Copy jobscript and launch as a hybrid MPI+OpenMP application

Hint: save the benchmark output (or note the run time) to be able to refer to it later

# 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



## Local installation (ARCHER2)

---

- Setup access to Scalasca and associated tools, accessible via “other-software”
  - Required for each shell session
  - Score-P and Scalasca installations are toolchain specific (GCC10 recommended)

```
% module swap PrgEnv-cray PrgEnv-gnu
% module load load-epcc-module other-software
% module load scalasca
```

- Check `module avail scalasca` for alternate Score-P/Scalasca modules available
- Copy tutorial sources to your personal workspace (if not already done)

```
% cd /work/ta066/ta066/$USER
% tar zxvf /work/y23/shared/tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

## NPB-MZ-MPI / BT instrumentation

```
#-----  
# The Fortran compiler used for MPI programs  
#-----  
#MPIF77 = ftn  
  
# Alternative variants to perform instrumentation  
...  
MPIF77 = scorep --user ftn  
  
#MPIF77 = SCOREP_WRAPPER_INSTRUMENTER_FLAGS="--user" \  
# SCOREP_WRAPPER_COMPILER_FLAGS="-fopenmp -fallow-argument-mismatch -ffixed-line-length-none" \  
# scorep-ftn  
  
# 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  
Alternatively, use compiler  
wrapper scorep-ftn

## NPB-MZ-MPI / BT instrumented build

```
% 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 8 C
scorep --user ftn -g -c -O3 -fopenmp bt.f
[...]
cd ../common; scorep --user ftn -g -c -O3 -fopenmp timers.f
[...]
scorep --user ftn -g -O3 -fopenmp -o ../bin.scorep/bt-mz_C.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_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/archer2/scorep.sbatch .
% cat scorep.sbatch
...
# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...
#export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L2_TCM
#export SCOREP_METRIC_RUSAGE=ru_stime
#export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
#export SCOREP_TIMER=gettimeofday

# Run the application
srun./bt-mz_C.8

% sbatch scorep.sbatch
```

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

Leave these lines commented out for the moment

- Submit job

# Summary measurement collection

---

```
% less slurm-<job_id>.out

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:      8
Use the default load factors with threads
Total number of threads:    48  (  6.0 threads/process)

Calculated speedup = 47.97

Time step    1

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

- Check the output of the application run

# BT-MZ summary analysis report examination

```
% ls
Slurm-<job_id>.out scorep_bt-mz_sum/
% ls -l scorep_bt-mz_sum
MANIFEST.md
profile.cubex
scorep.cfg

% cube scorep_bt-mz_sum/profile.cubex
# alternatively
% square scorep_bt-mz_sum/
[CUBE GUI showing summary analysis report]

% paraprof scorep_bt-mz_A_4x4_sum/profile.cubex
[TAU ParaProf GUI showing summary analysis report]
```

- Creates experiment directory including
  - A brief content overview (MANIFEST.md)
  - A record of the measurement configuration (scorep.cfg)
  - The analysis report that was collated after measurement (profile.cubex)
- Interactive exploration with Cube

## Hint:

Copy 'profile.cubex' to local system (laptop) using 'scp' to improve responsiveness of GUI

Reference results available:  
</work/ta066/shared/examples>

## BT-MZ summary analysis report remapping

---

```
% ls -l scorep_bt-mz_sum/
MANIFEST.md
profile.cubex
scorep.cfg

# remap the Score-P way
% cube_remap2 -d -o scorep_bt-mz_sum/summary.cubex \
scorep_bt-mz_sum/profile.cubex

# remap the Scalasca way
% square scorep_bt-mz_sum/profile.cubex
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Displaying ./scorep_bt-mz_sum/summary.cubex...

[CUBE GUI showing summary analysis report]
```

- `profile.cubex` contains raw measurement data
- Enhance by *remapping*, i.e., transform given metric tree into metric hierarchy
  - Score-P way:  
`cube_remap2 -d profile.cubex`
  - Scalasca way:  
`square profile.cubex`



## Further information

---

- Community instrumentation & measurement infrastructure
  - Instrumentation (various methods)
  - Basic and advanced profile generation
  - Event trace recording
- Available under 3-clause 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](mailto:support@score-p.org)
- Subscribe to [news@score-p.org](mailto:news@score-p.org), to be up to date