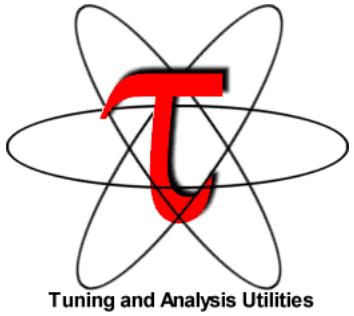
VI-HPS

TAU Performance System®
Hands-On



Sameer Shende
sameer@cs.uoregon.edu
University of Oregon
http://tau.uoregon.edu

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

TAU hands-on exercises

TAU tutorial exercise objectives

- Familiarise with usage of TAU 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
  - …

Local Installation (*Wrangler, TACC*)

- Setup preferred program environment compilers

```
% ssh -Y <login>@wrangler.tacc.utexas.edu
% source ~tg457572/tau.bashrc
% cp ~tg457572/pkgs/tar/NPB3.3-MZ-MPI.tar.gz $WORK;

# If you have previous performance data from Score-P, you may view it with TAU's paraprof
% paraprof profile.cubex &
```

For PerfExplorer:

```
% cp ~tg457572/pkgs/tar/data.tgz. ; tar zxf data.tgz; cd data
% cat README; cd tau; ./upload.sh; perfexplorer &
  and follow the steps
```

Local Installation (*Stampede2, TACC*)

- Setup preferred program environment compilers

```
% ssh -Y <login>@stampede2.tacc.utexas.edu
% source ~tg828282/Tutorial/vihps-intel.sh

# If you have previous performance data from Score-P, you may view it with TAU's paraprof
% paraprof profile.cubex &
```

For PerfExplorer:

```
% cp ~tg457572/pkgs/tar/data.tgz. ; tar zxf data.tgz; cd data
% cat README; cd tau; ./upload.sh; perfexplorer &
  and follow the steps
```

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's ready to “make” one or more of the benchmarks and install them into a (tool-specific) “bin” subdirectory

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
  - Proposed hands-on setup on Stampede2:
    - 2 compute nodes with 1 Intel Xeon Phi 7250 CPU (Knights Landing, KNL) each
    - 32 MPI processes with 4 OpenMP threads each
  - bt-mz_C.32 should run in less than 30 seconds

Compiler and MPI modules (On Wrangler, TACC)

- Select appropriate compiler / MPI combination (if not done automatically)

```
. ~tg457572/tau.bashrc
```

- Copy tutorial sources to your scratch directory

```
% cd $WORK
% tar zxvf ~tg457572/pkgs/tar//NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

Compiler and MPI modules (On Stampede2, TACC)

- Select appropriate compiler / MPI combination (if not done automatically)

```
. ~tg828282/Tutorial/vihps-intel.sh
```

- Copy tutorial sources to your scratch directory

```
% cd $SCRATCH
% tar zxvf ~tg828282/Tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

NPB-MZ-MPI / BT: config/make.def : Use default compiler!

```
#                SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS.
#
#-----------------------------------------------------------------------

#-----------------------------------------------------------------------
# Configured for generic MPI with GCC compiler
#-----------------------------------------------------------------------
#COMPILER = -homp            # Cray/CCE compiler
COMPILER = -fopenmp          # GNU/GCC compiler
#COMPILER = -qopenmp         # Intel compiler

...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
# MPIF77 = mpif77 # OpenMPI with Intel compiler
MPIF77 = mpif77
# Alternative variant to perform instrumentation
# MPIF77 = tau_f90.sh
# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP) mpif77 -f77=ifort
#MPIF77 = scorep …
...
```

Default (no instrumentation)

Uncomment MPIF77 and comment Score-P's compiler wrapper to generate uninstrumented binary!

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 HPC systems:   *
 *       make bt-mz CLASS=C NPROCS=8                            *
 ****************************************************************
```

- Type "make" for instructions

Building an NPB-MZ-MPI Benchmark (Wrangler)

```
% make bt-mz CLASS=B NPROCS=16
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 16 B
make[2]: Entering directory `../BT-MZ'
mpif77  -c  -O3 -fopenmp       bt.f
[…]
mpif77  -c  -O3 -fopenmp       mpi_setup.f
cd ../common;  mpif77  -c  -O3 -fopenmp        print_results.f
cd ../common;  mpif77  -c  -O3 -fopenmp        timers.f
mpif77  -O3 -fopenmp   -o ../bin/bt-mz_B.16 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_B.16
make[1]: Leaving directory `BT-MZ'
```

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, lu-mz, sp-mz
  - the number of MPI processes: NPROCS=**32**
  - the benchmark class (S, W, A, B, C, D, E): CLASS=**C**

Shortcut: % **make suite**

NPB-MZ-MPI / BT reference execution (Wrangler)

```
% cd bin
% cp ../jobscript/wrangler/reference.sbatch .
% less reference.sbatch
% sbatch reference.sbatch
% less mzmpibt.o<job_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:    16
 Use the default load factors with threads
 Total number of threads:     64  (  4.0 threads/process)
 Calculated speedup =     62.56
 Time step    1
 Time step   20
  [...]
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 5.45
```

- 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

NPB-MZ-MPI / BT reference execution (Stampede2)

```
% cd bin
% cp ../jobscript/stampede2/reference.sbatch .
% less reference.sbatch
% sbatch reference.sbatch
% less mzmpibt.o<job_id>
 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:    32
 Total number of threads:      128  (  4.0 threads/process)

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

 BT-MZ Benchmark Completed.
 Time in seconds = 22.34
```

- 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

NPB-MZ-MPI / BT with TAU's OMPT replacement library (tau_exec –T ompt)

```
% cd bin
% cp ../jobscript/stampede2/tau.sbatch .
% sbatch ./tau.sbatch
% cat mzmpibt.o<job_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:    16
 Use the default load factors with threads
 Total number of threads:     64  (  4.0 threads/process)
 Calculated speedup =     62.56
 Time step    1
 Time step   20
  [...]
 Time step  180
 Time step  200
 Verification Successful

 BT-MZ Benchmark Completed.
 Time in seconds = 5.45
% paraprof &
% paraprof --pack bt_ebs.ppk
<Copy file over to desktop using scp>
% paraprof bt_ebs &
```

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

tau_exec launches un-instrumented binaries!

NPB-MZ-MPI / BT with TAU's Sampling (tau_exec –ompt -ebs)

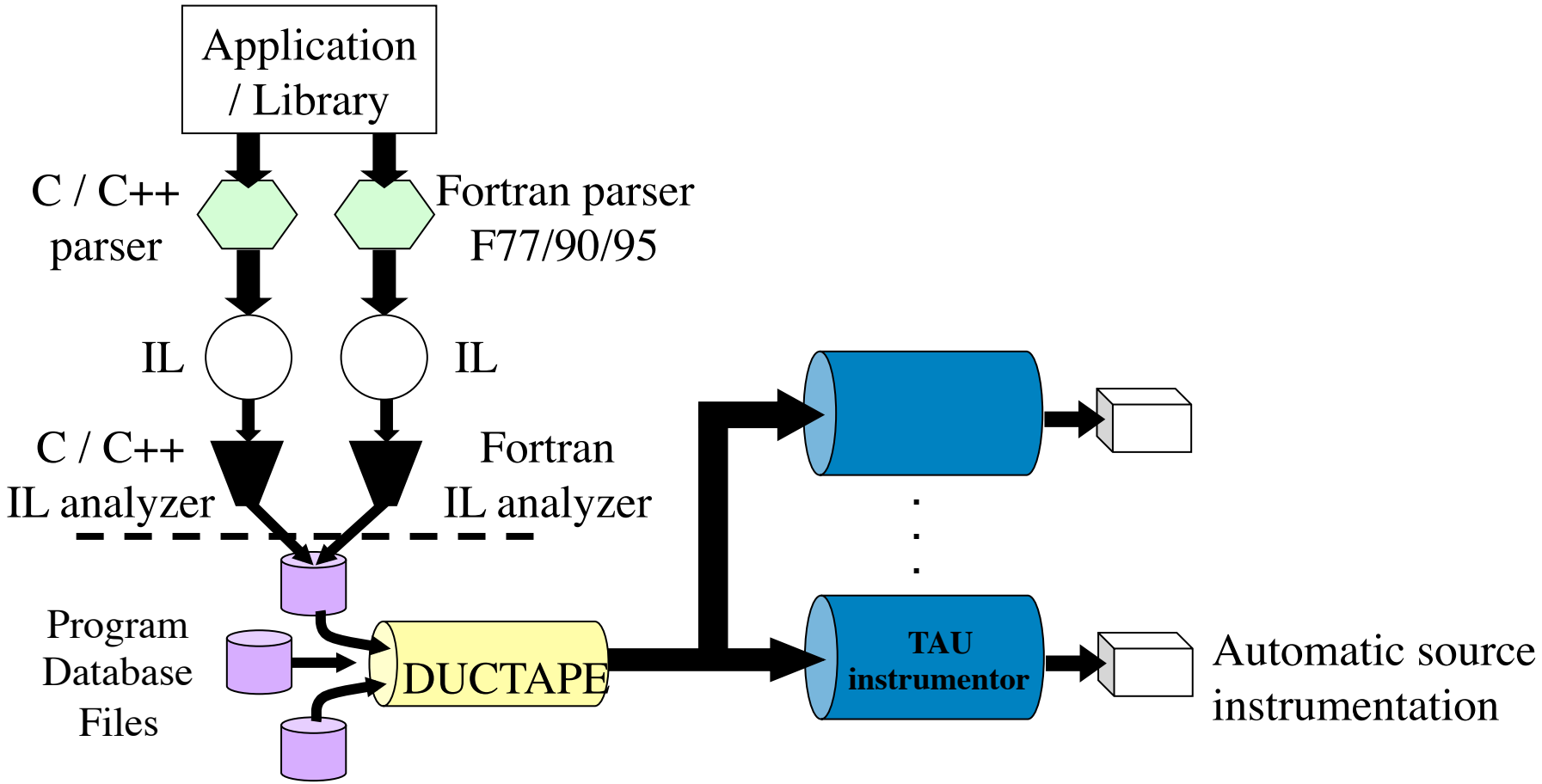
```
% cd bin
% cp ../jobscript/stampede2/tau.sbatch .
# Uncomment the -ebs line at the end and
% sbatch ./tau.sbatch
% cat mzmpibt.o<job_id>
 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
 Number of zones:  16 x  16
 Iterations:  200   dt:   0.000100
 Number of active processes:    32
 Total number of threads:      128  (  4.0 threads/process)

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

 BT-MZ Benchmark Completed.
 Time in seconds = 24
% paraprof &
% paraprof --pack bt_ebs.ppk
<Copy file over to desktop using scp>
% paraprof bt_ebs &
```

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

tau_exec launches un-instrumented binaries!

TAU's Source Instrumentation

TAU's Static Analysis System: Program Database Toolkit (PDT)



Source instrumentation using TAU

```
Uncomment the MPIF77 line for TAU in config/make.def
% vi config/make.def
MPIF77 = mpiifort

# Alternative variants to perform instrumentation
#MPIF77 = psc_instrument -t user,mpi,omp -s ${PROGRAM}.sir  mpiifort
#MPIF77 = scalasca -instrument  mpiifort
#MPIF77 = tau_f90.sh -tau_makefile=/home/00494/tg457572/pkgs/tau_latest/x86_64/lib/Makefile.tau-icpc-ompt-tr6
#MPIF77 = vtf77 -vt:hyb -vt:f77  mpiifort

% make clean; make suite
% cd bin.tau
% cp ../jobscript/tau.sbatch .
% sbatch tau.sbatch
% paraprof &
OR

% paraprof --pack bt.ppk
(scp bt.ppk to your laptop and use paraprof locally)
You may install from http://tau.uoregon.edu/tau.dmg or tau.exe
% paraprof bt.ppk
```

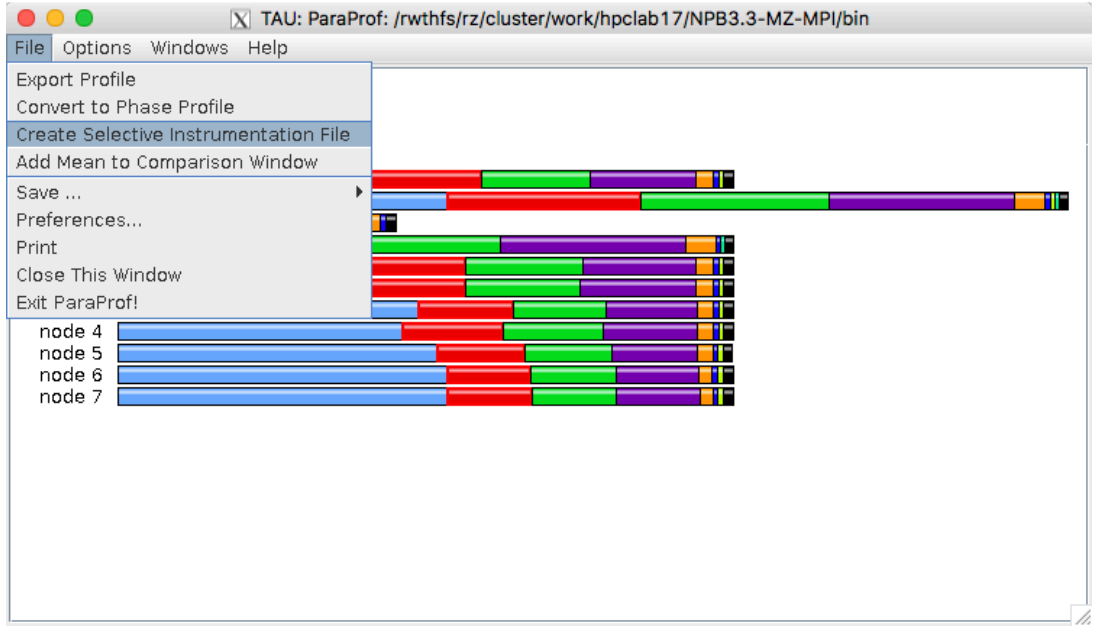
- Copy jobscript and launch as a hybrid MPI+OpenMP application

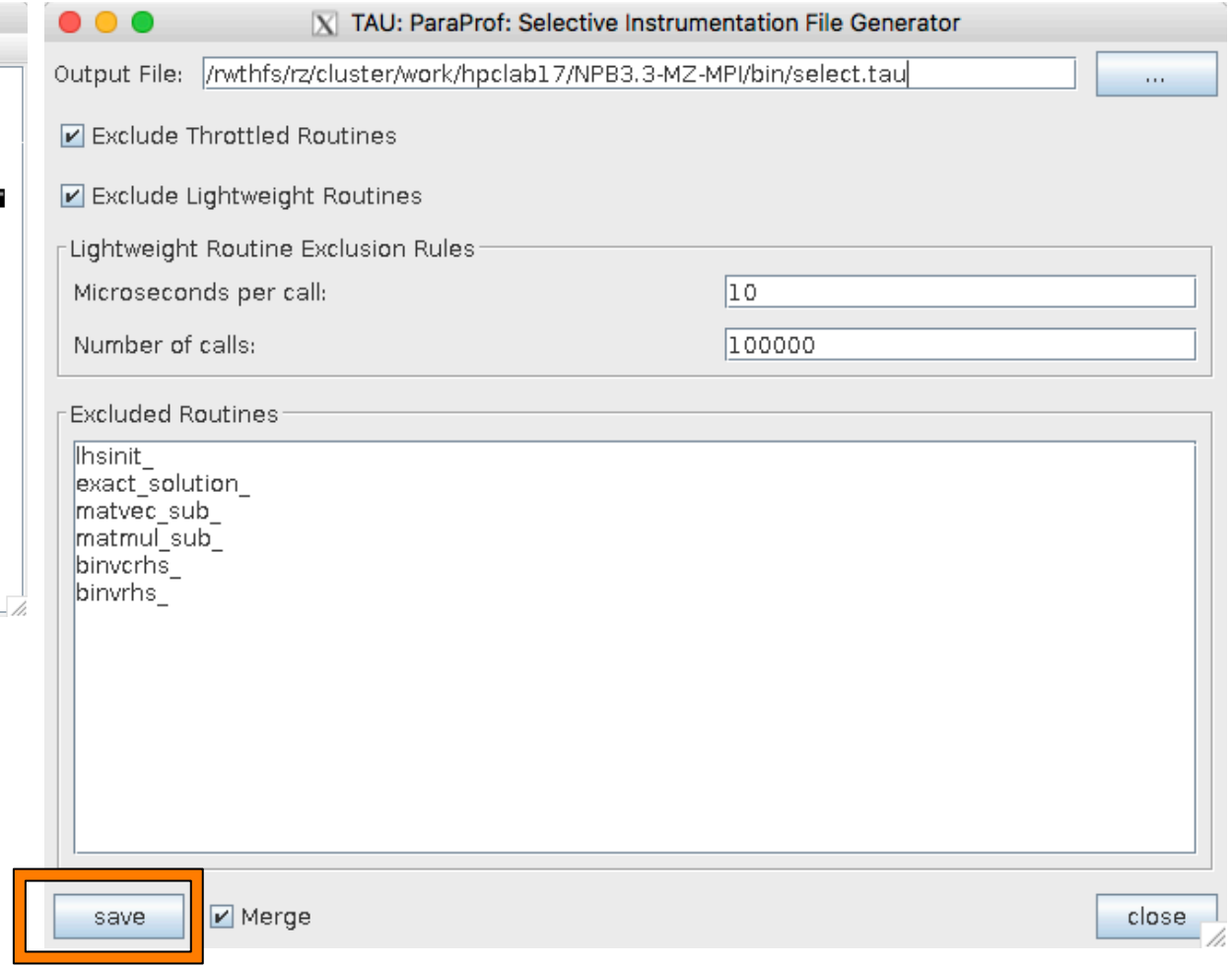
Hint: save the benchmark output (or note the run time) to be able to refer to it later

TAU's Selective Instrumentation (Filter) File Format

```
% export TAU_OPTIONS='-optTauSelectFile=/path/to/select.tau …'
% cat select.tau
BEGIN_INCLUDE_LIST
int main#
int dgemm#
END_INCLUDE_LIST
BEGIN_FILE_INCLUDE_LIST
Main.c
Blas/*.f77
END_FILE_INCLUDE_LIST
# replace INCLUDE with EXCLUDE list for excluding routines/files

BEGIN_INSTRUMENT_SECTION
loops routine="foo"
loops routine="int main#"
END_INSTRUMENT_SECTION
% export TAU_SELECT_FILE=select.tau    (to use at runtime)
```

Create a Selective Instrumentation File, Re-instrument, Re-run





% paraprof

TAU's Runtime Environment Variables

Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_CALLPATH	0	Setting to 1 turns on callpath profiling
TAU_TRACK_MEMORY_FOOTPRINT	0	Setting to 1 turns on tracking memory usage by sampling periodically the resident set size and high water mark of memory usage
TAU_TRACK_POWER	0	Tracks power usage by sampling periodically.
TAU_CALLPATH_DEPTH	2	Specifies depth of callpath. Setting to 0 generates no callpath or routine information, setting to 1 generates flat profile and context events have just parent information (e.g., Heap Entry: foo)
TAU_SAMPLING	1	Setting to 1 enables event-based sampling.
TAU_TRACK_SIGNALS	0	Setting to 1 generate debugging callstack info when a program crashes
TAU_COMM_MATRIX	0	Setting to 1 generates communication matrix display using context events
TAU_THROTTLE	1	Setting to 0 turns off throttling. Throttles instrumentation in lightweight routines that are called frequently
TAU_THROTTLE_NUMCALLS	100000	Specifies the number of calls before testing for throttling
TAU_THROTTLE_PERCALL	10	Specifies value in microseconds. Throttle a routine if it is called over 100000 times and takes less than 10 usec of inclusive time per call
TAU_CALLSITE	0	Setting to 1 enables callsite profiling that shows where an instrumented function was called. Also compatible with tracing.
TAU_PROFILE_FORMAT	Profile	Setting to "merged" generates a single file. "snapshot" generates xml format
TAU_METRICS	TIME	Setting to a comma separated list generates other metrics. (e.g., ENERGY,TIME,P_VIRTUAL_TIME,PAPI_FP_INS,PAPI_NATIVE_<event>:<subevent>)

Runtime Environment Variables

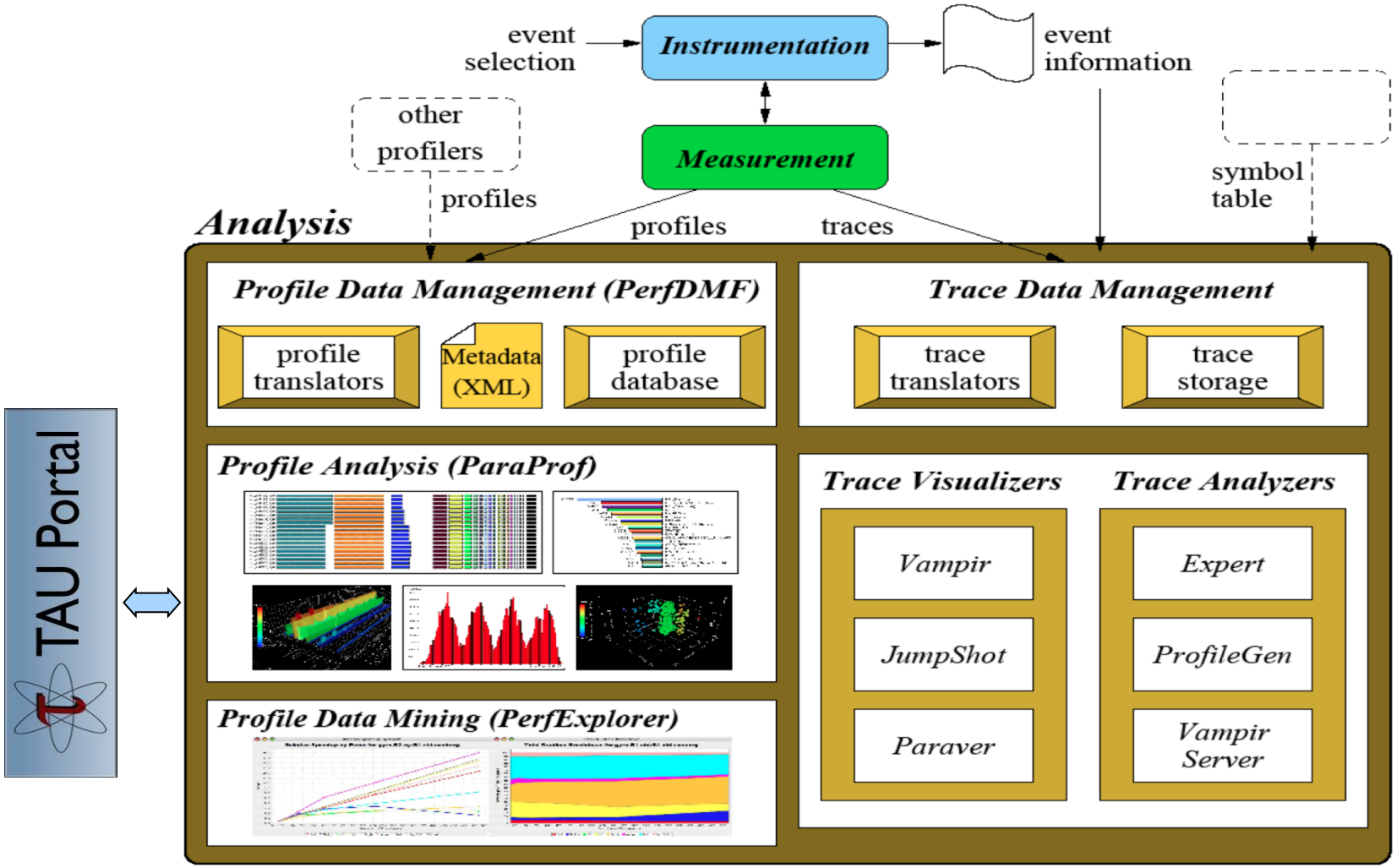
Environment Variable	Default	Description
TAU_TRACE	0	Setting to 1 turns on tracing
TAU_TRACE_FORMAT	Default	Setting to “otf2” turns on TAU’s native OTF2 trace generation (configure with –otf=download)
TAU_EBS_UNWIND	0	Setting to 1 turns on unwinding the callstack during sampling (use with tau_exec –ebs or TAU_SAMPLING=1)
TAU_EBS_RESOLUTION	line	Setting to “function” or “file” changes the sampling resolution to function or file level respectively.
TAU_TRACK_LOAD	0	Setting to 1 tracks system load on the node
TAU_SELECT_FILE	Default	Setting to a file name, enables selective instrumentation based on exclude/include lists specified in the file.
TAU_OMPT_SUPPORT_LEVEL	basic	Setting to “full” improves resolution of OMPT TR6 regions on threads 1.. N-1. Also, “lowoverhead” option is available.
TAU_OMPT_RESOLVE_ADDRESS_EAGERLY	0	Setting to 1 is necessary for event based sampling to resolve addresses with OMPT TR6 (-ompt=download-tr6)

Runtime Environment Variables

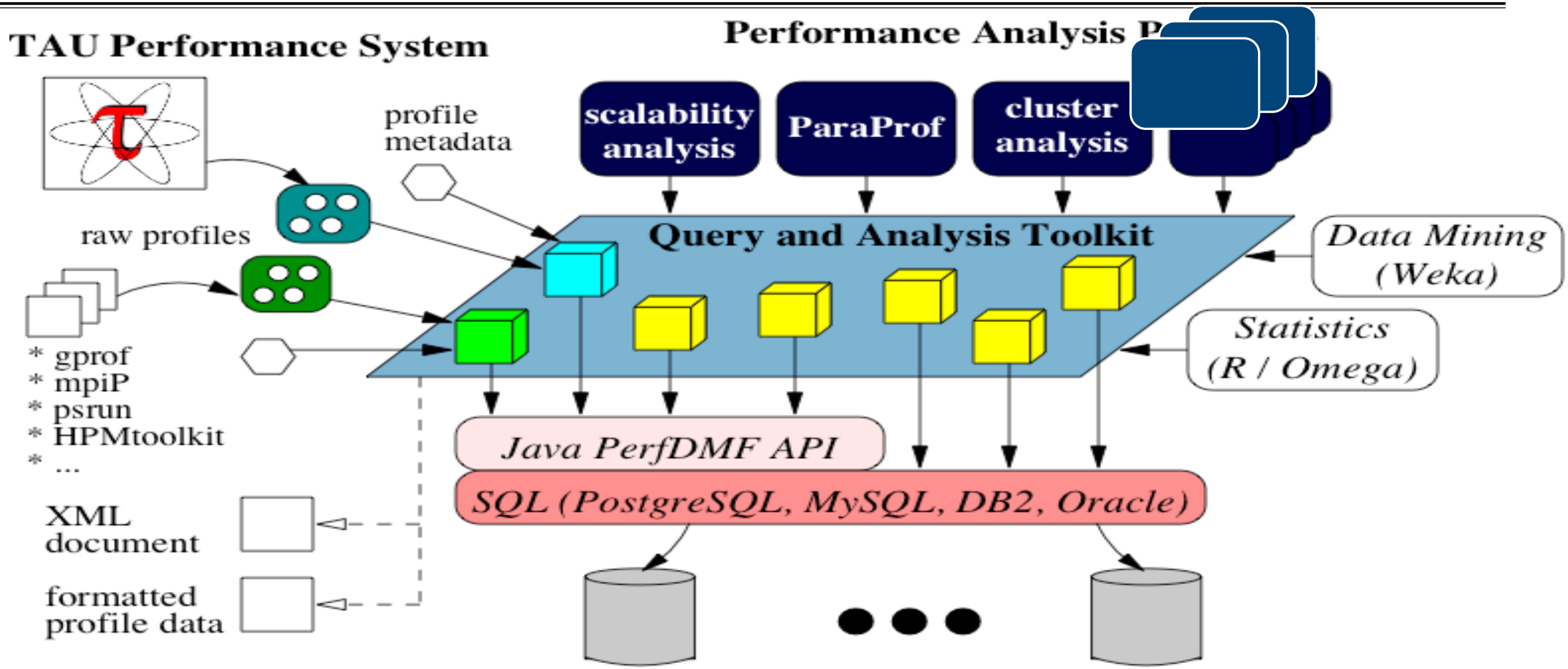
Environment Variable	Default	Description
TAU_TRACK_MEMORY_LEAKS	0	Tracks allocates that were not de-allocated (needs –optMemDbg or tau_exec –memory)
TAU_EBS_SOURCE	TIME	Allows using PAPI hardware counters for periodic interrupts for EBS (e.g., TAU_EBS_SOURCE=PAPI_TOT_INS when TAU_SAMPLING=1)
TAU_EBS_PERIOD	100000	Specifies the overflow count for interrupts
TAU_MEMDBG_ALLOC_MIN/MAX	0	Byte size minimum and maximum subject to bounds checking (used with TAU_MEMDBG_PROTECT_*)
TAU_MEMDBG_OVERHEAD	0	Specifies the number of bytes for TAU's memory overhead for memory debugging.
TAU_MEMDBG_PROTECT_BELOW/ABOVE	0	Setting to 1 enables tracking runtime bounds checking below or above the array bounds (requires –optMemDbg while building or tau_exec –memory)
TAU_MEMDBG_ZERO_MALLOC	0	Setting to 1 enables tracking zero byte allocations as invalid memory allocations.
TAU_MEMDBG_PROTECT_FREE	0	Setting to 1 detects invalid accesses to deallocated memory that should not be referenced until it is reallocated (requires –optMemDbg or tau_exec –memory)
TAU_MEMDBG_ATTEMPT_CONTINUE	0	Setting to 1 allows TAU to record and continue execution when a memory error occurs at runtime.
TAU_MEMDBG_FILL_GAP	Undefined	Initial value for gap bytes
TAU_MEMDBG_ALINGMENT	Sizeof(int)	Byte alignment for memory allocations
TAU_EVENT_THRESHOLD	0.5	Define a threshold value (e.g., .25 is 25%) to trigger marker events for min/max

TAU's Analysis Tools: PerfExplorer

TAU Analysis



TAUdb: Performance Data Management Framework



Using TAUdb

- Configure TAUdb (Done by each user)

  % taudb_configure --create-default
  - Choose derby, PostgreSQL, MySQL, Oracle or DB2
  - Hostname
  - Username
  - Password
  - Say yes to downloading required drivers (we are not allowed to distribute these)
  - Stores parameters in your ~/.ParaProf/taudb.cfg file
- Configure PerfExplorer (Done by each user)

  % perfexplorer_configure
- Execute PerfExplorer

  % perfexplorer

Local Installation (*Stampede2*)

- Setup preferred program environment compilers

```
% . ~tg828282/Tutorial/vihps-intel.sh
% cp ~tg457572/pkgs/tar/data.tgz . ; tar zxf data.tgz
% cd data
% cat README
  and follow the steps
% cd tau
% ./upload.sh
% perfexplorer
```

Local Installation (*Wrangler, Stampede*)

- Setup preferred program environment compilers

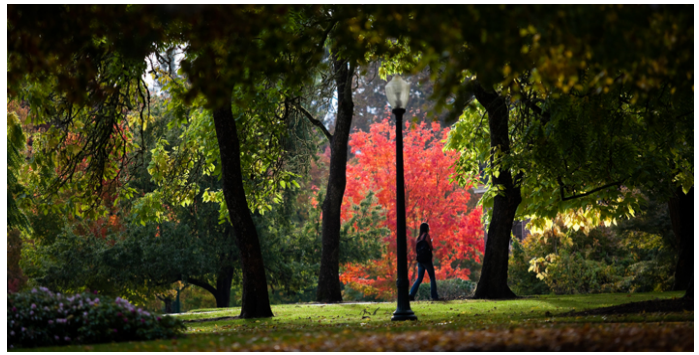
```
% source ~tg457572/tau.bashrc
% cp ~tg457572/pkgs/tar/data.tgz . ; tar zxf data.tgz
% cd data
% cat README
  and follow the steps
% cd tau
% ./upload.sh
% perfexplorer &
```

Local Installation (*Stampede2, TACC*)

- Setup preferred program environment compilers

```
% . ~tg828282/Tutorial/vihps-intel.sh
% cp ~tg457572/pkgs/tar/data.tgz . ; tar zxf data.tgz
% cd data
% cat README
  and follow the steps
% cd tau
% ./upload.sh
% perfexplorer
```

Performance Research Lab, University of Oregon, Eugene, USA



Support Acknowledgments

- US Department of Energy (DOE)
  - Office of Science contracts
  - SciDAC, LBL contracts
  - LLNL-LANL-SNL ASC/NNSA contract
  - Battelle, PNNL contract
  - ANL, ORNL contract
- Department of Defense (DoD)
  - PETTT, HPCMP
- National Science Foundation (NSF)
  - Glassbox, SI-2
- NASA
- CEA, France
- Partners:
  - University of Oregon
  - ParaTools, Inc., ParaTools, SAS
  - The Ohio State University
  - University of Tennessee, Knoxville
  - T.U. Dresden, GWT
  - Juelich Supercomputing Center

Acknowledgement

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative.

Download TAU from U. Oregon

http://tau.uoregon.edu

http://www.hpclinux.com [LiveDVD, OVA]

http://e4s.io [Containers for Extreme-Scale Scientific Software Stack]

Free download, open source, BSD license