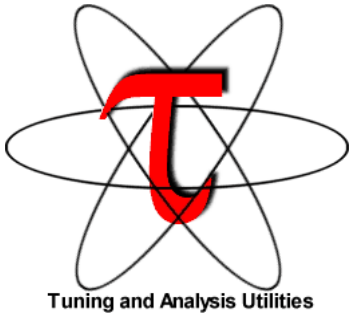
VI-HPS

TAU Performance System®



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

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
  - …

Installing TAU on your laptop for paraprof (GUI)

- Microsoft Windows
  - Install Java from Oracle.com
  - http://tau.uoregon.edu/tau.exe
  - Install, click on a ppk file to launch paraprof
- macOS
  - Install Java 11.0.3:
    - Download http://tau.uoregon.edu/java.dmg
    - If you have multiple Java installations, add to your ~/.zshrc (or ~/.bashrc as appropriate):
    - export PATH=/Library/Java/JavaVirtualMachines/jdk-11.0.3.jdk/Contents/Home/bin:$PATH
    - java -version
  - Download and install TAU (copy to /Applications from dmg):
    - http://tau.uoregon.edu/tau.dmg
    - export PATH=/Applications/TAU/tau/apple/bin:$PATH
    - paraprof app.ppk &
- Linux (http://tau.uoregon.edu/tau.tgz)
  - ./configure; make install; export PATH=<taudir>/x86_64/bin:$PATH
  - paraprof app.ppk &

Login to Hawk

- Setup preferred program environment compilers using Intel compilers with OMPT

```
% ssh -Y hawk.hww.hlrs.de -l <USER>
% module load tau
% cp -r /zhome/academic/HLRS/hlrs/hpcjgrac/TAU/workshop .
% cd workshop
% cat handson.txt
% cat NPB3.3-MPI-MZ/bin/rt.sh
#!/bin/sh
export OMP_NUM_THREADS=4
mpirun -n 16 tau_exec -T ompt,v5,papi,mpi -ompt ./bt-mz.B.16
```

Using TAU on Hawk at HLRS

- Allocate a single node (Reservation: R_tw):
  - qsub -I –q R_tw -l select=1:mpiprocs=128 -l walltime=0:30:00
- Load the TAU module for the workshop:
  - module load tau
- Copy the workshop examples to your workspace:
  - cp -r /zhome/academic/HLRS/hlrs/hpcjgrac/TAU/workshop .
- Build an example and run with TAU:
  - cd workshop/NPB3.3-MZ-MPI
  - make clean; make -j8; cd bin; ./clean.sh; cat r.sh rt.sh
  - ./rt.sh
  - pprof –a | more ; paraprof --pack bt.ppk; <scp to your laptop and launch with paraprof>; paraprof bt.ppk

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 generic MPI with GCC compiler
#---------------------------------------------------------------------
OPENMP = -qopenmp          # Intel compiler
#OPENMP = -qopenmp         # GCC compiler

...
#---------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------------
F77 = mpif77

# Alternative variant to perform instrumentation
#MPIF77 = scorep --user  mpif77

# PREP is a generic preposition macro for instrumentation preparation
#MPIF77 = $(PREP) mpif77
...
```

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 Archer:       *
 *       make bt-mz CLASS=B NPROCS=16                           *
 ***********************************************************
```

- Type "make" for instructions

Building an NPB-MZ-MPI Benchmark

```
% make bt-mz CLASS=B NPROCS=16
make[1]: Entering directory `BT-MZ'
make[2]: Entering directory `sys'
gcc  -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;  ftn  -c  -O3 -fopenmp        print_results.f
cd ../common;  ftn  -c  -O3 -fopenmp        timers.f
mpif77  -O3 -fopenmp   -o ../bin/bt-mz_C.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 benchmark class (S, W, A, B, C, D, E): CLASS=**B**
  - the number of MPI processes: NPROCS=**16**

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
  - 16 processes each with 4 threads should be reasonable for 1 compute node of HPE Hawk
  - bt-mz_B.16 should run in
    - around 3 seconds with gcc mpt
    - around 3 seconds with intel mpt

NPB-MZ-MPI / BT Reference Execution

```
% qsub -I -q R_tw -l select=1:mpiprocs=128 -l walltime=0:30:00
% module load tau
% cd bin
% cat r.sh ; cat rt.sh
% ./r.sh
 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:    16
 Use the default load factors with threads
 Total number of threads:    64   (  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 = 3.46
```

- 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

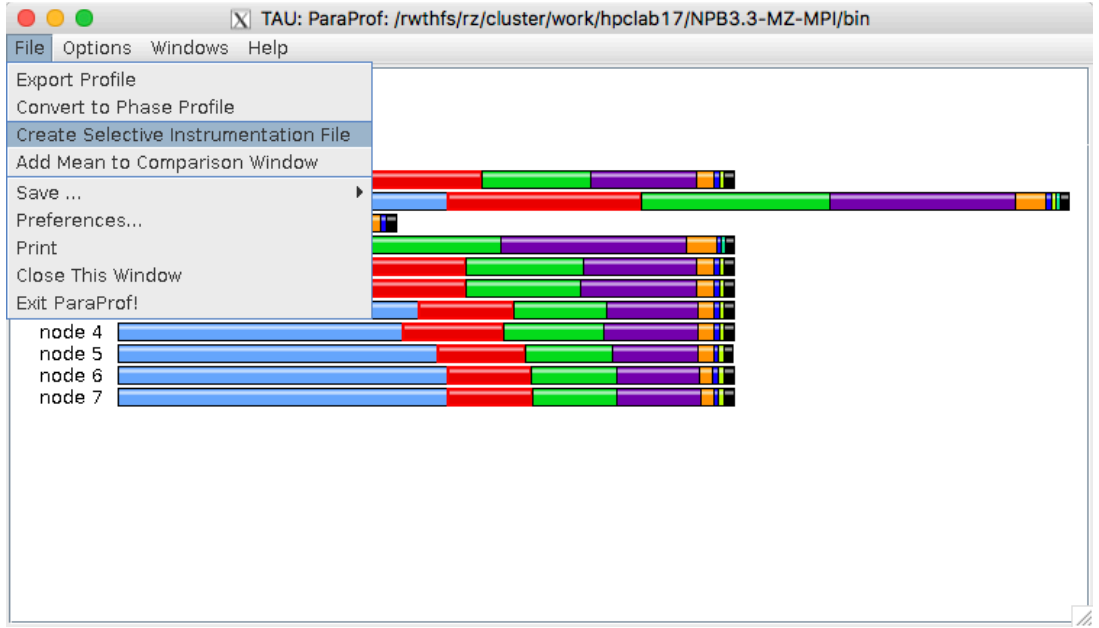
NPB-MZ-MPI / BT Execution with TAU

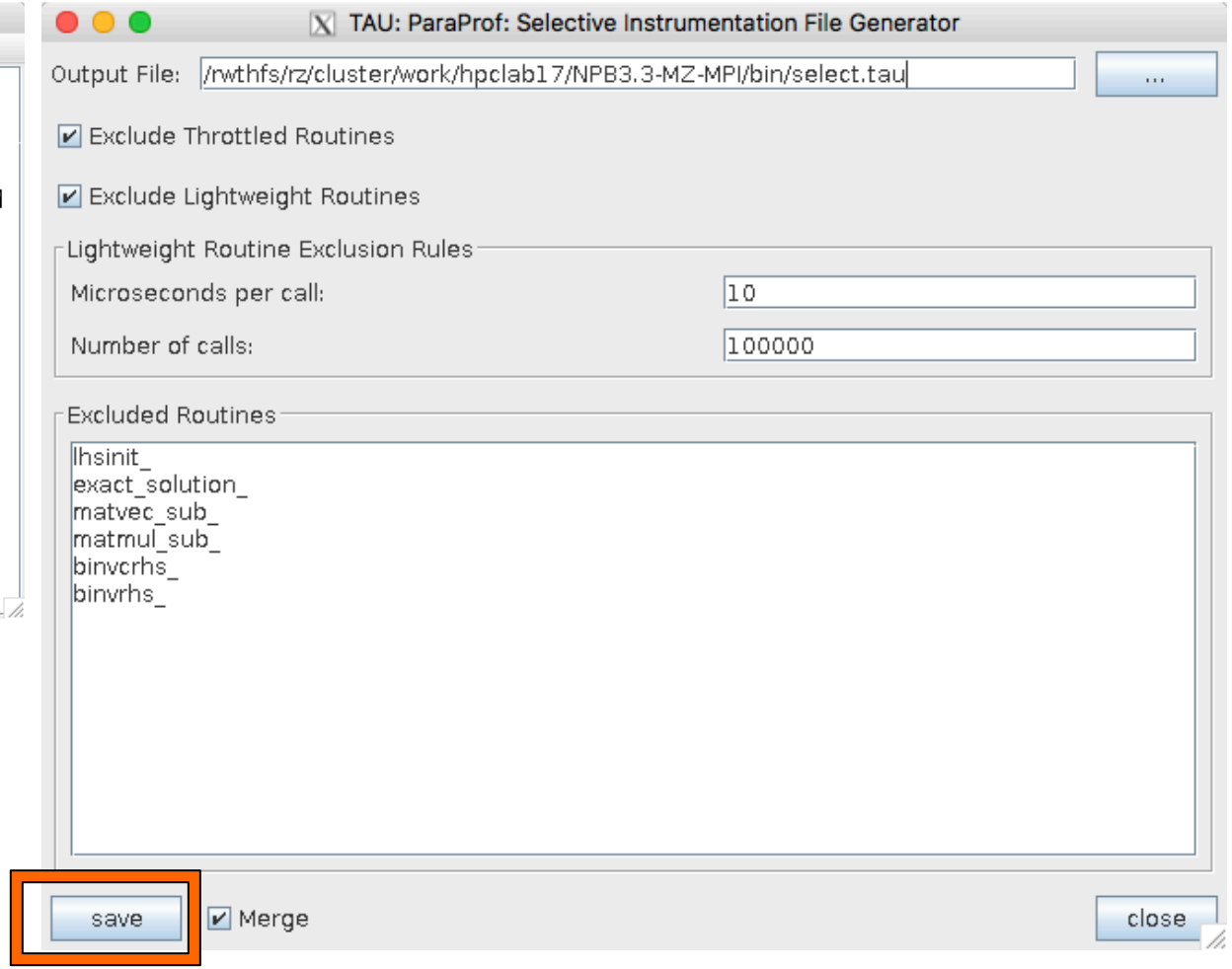
```
% cat rt.sh rt1.sh rt2.sh

% ./rt.sh
% paraprof --pack bt.ppk
<SCP to your laptop>
% paraprof bt.ppk &

% ./rt1.sh
% paraprof --pack bt_ebs.ppk
<SCP to your laptop>
% paraprof bt_ebs.ppk &

% ./rt2.sh
% vampir traces.otf2 &
```

NPB-MZ-MPI / BT Source Instrumentation with TAU

```
Choose an Opari source instrumentation option:
% cd workshop/NPB3.3-MZ-MPI;
% ls $TAU/Makefile*
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt-openmp-opari

% make clean
% make F77=tau_f77.sh
% cd bin
% ./r.sh
% paraprof --pack bt_opari.ppk
<SCP to your laptop>
% paraprof bt_opari.ppk &
```

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





% paraprof

Compile-Time Options for TAU's compiler scripts (e.g., tau_f90.sh)

Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

-optVerbose	Turn on verbose debugging messages
-optCompInst	Use compiler based instrumentation
-optNoCompInst	Do not revert to compiler instrumentation if source instrumentation fails.
-optTrackIO	Wrap POSIX I/O call and calculates vol/bw of I/O operations (configure TAU with *–iowrapper*)
-optTrackGOMP	Enable tracking GNU OpenMP runtime layer (used without –opari)
-optMemDbg	Enable runtime bounds checking (see TAU_MEMDBG_* env vars)
-optKeepFiles	Does not remove intermediate .pdb and .inst.* files
-optPreProcess	Preprocess sources (OpenMP, Fortran) before instrumentation
-optTauSelectFile="<file>"	Specify selective instrumentation file for *tau_instrumentor*
-optTauWrapFile="<file>"	Specify path to *link_options.tau* generated by *tau_gen_wrapper*
-optHeaderInst	Enable Instrumentation of headers
-optTrackUPCR	Track UPC runtime layer routines (used with tau_upc.sh)
-optLinking=""	Options passed to the linker. Typically $(TAU_MPI_FLIBS) $(TAU_LIBS) $(TAU_CXXLIBS)
-optCompile=""	Options passed to the compiler. Typically $(TAU_MPI_INCLUDE) $(TAU_INCLUDE) $(TAU_DEFS)
-optPdtF95Opts=""	Add options for Fortran parser in PDT (f95parse/gfparse) …

Compile-Time Options (contd.)

- Optional parameters for the TAU_OPTIONS environment variable:

% tau_compiler.sh

-optMICOffload	Links code for Intel MIC offloading, requires both host and MIC TAU libraries
-optShared	Use TAU's shared library (libTAU.so) instead of static library (default)
-optPdtCxxOpts=""	Options for C++ parser in PDT (cxxparse).
-optPdtF90Parser=""	Specify a different Fortran parser
-optPdtCleanscapeParser	Specify the Cleanscape Fortran parser instead of GNU gfparser
-optTau=""	Specify options to the tau_instrumentor
-optTrackDMAPP	Enable instrumentation of low-level DMAPP API calls on Cray
-optTrackPthread	Enable instrumentation of pthread calls

See tau_compiler.sh for a full list of TAU_OPTIONS.

…

Compiling Fortran Codes with TAU

- If your Fortran code uses free format in .f files (fixed is default for .f), you may use:
  % export TAU_OPTIONS= '-optPdtF95Opts="-R free" -optVerbose '

- To use the compiler based instrumentation instead of PDT (source-based):
  % export TAU_OPTIONS= '-optCompInst -optVerbose'

- To use an instrumentation specification file:
  % export TAU_OPTIONS= '-optTauSelectFile=select.tau -optVerbose -optPreProcess'
  % cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  # this statement instruments all outer loops in all routines. # is wildcard as well as comment in first column.
  END_INSTRUMENT_SECTION

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)
```

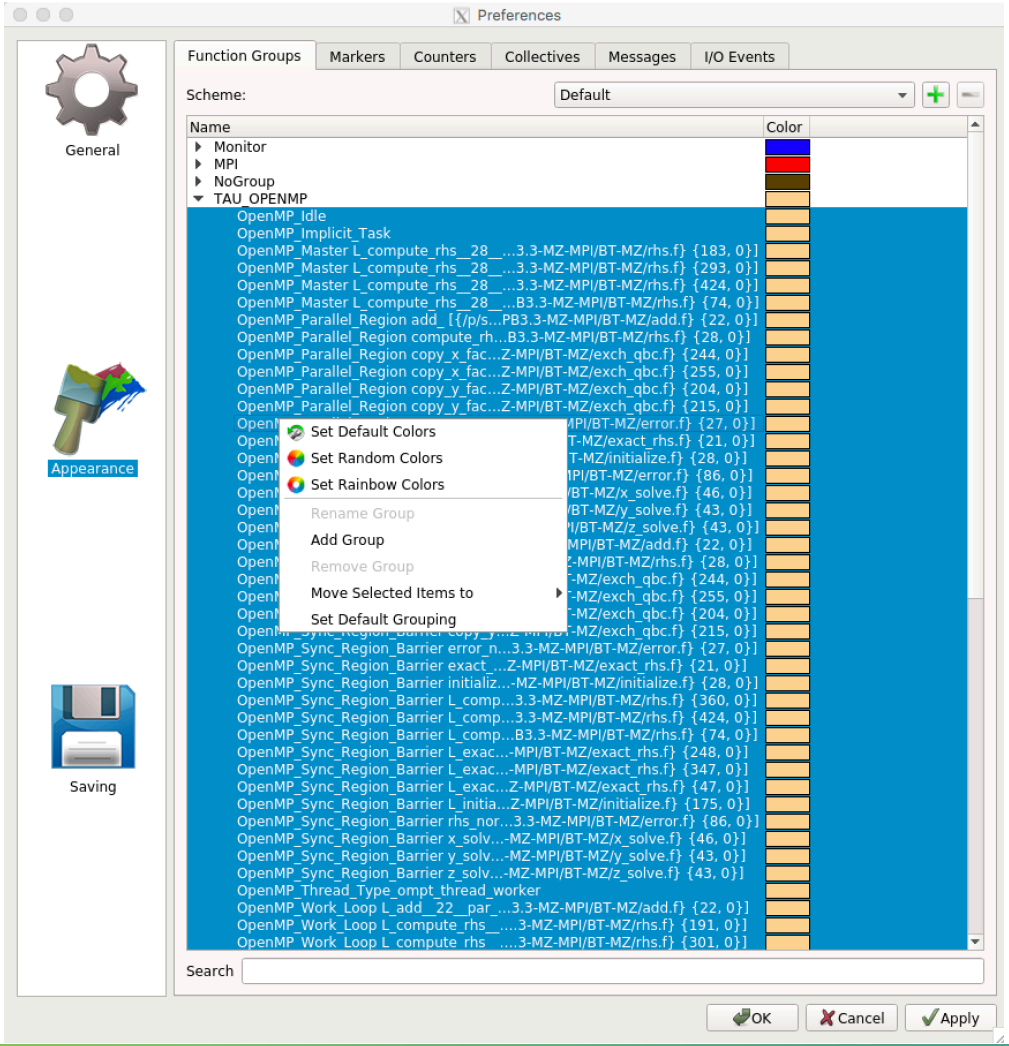
TAU's Selective Instrumentation (Filter) File Format

```
Create a select.tau file and copy it over to the workshop/NPB-3.3-MZ-MPI directory:
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% make clean
% export TAU_MAKEFILE=$TAU/Makefile.tau-icpc-papi-mpi-pdt-openmp-opari
% make F77=tau_f77.sh
% cd bin; ./clean.sh
% ./r.sh
% paraprof --pack bt_opt.ppk
<SCP bt_opt.ppk to your laptop>
% paraprof bt_opt.ppk &
```

To generate OTF2 traces for Vampir [TU Dresden]

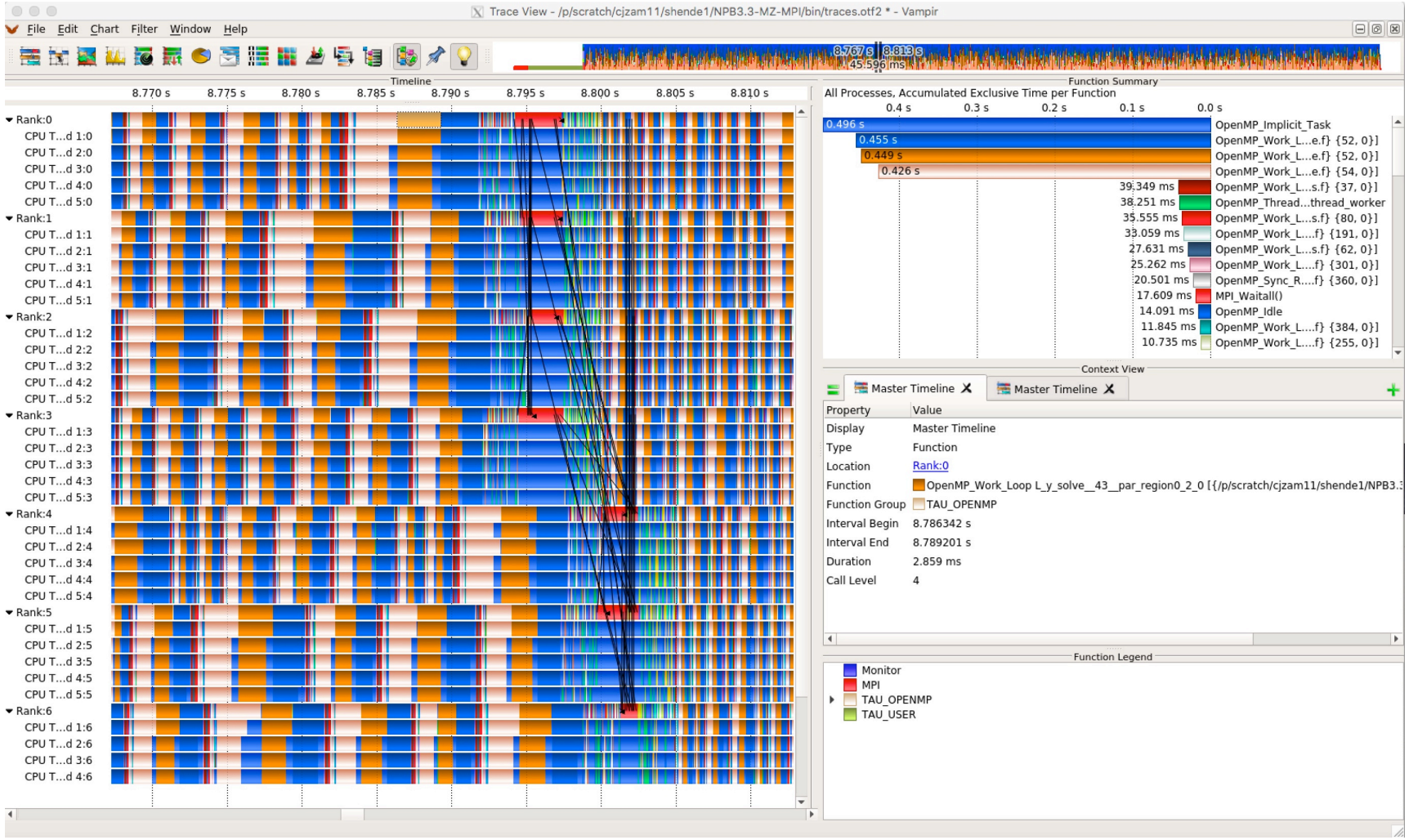
```
% cat rt2.sh
#!/bin/sh
export OMP_NUM_THREADS=4
export TAU_TRACE=1
export TAU_TRACE_FORMAT=otf2


mpirun -np 16 tau_exec -T ompt,v5,papi,mpi -ompt ./bt-mz.B.16


# Launch: vampir traces.otf2 &
```

Change Colors for Events in Vampir [TU Dresden]



- File -> Preferences -> Appearance
- Multi-select functions under TAU_OPENMP
- Select “Set Random Colors”
- Click Apply, OK

Vampir [TU Dresden] Provides a Timeline Display



TAU generates OTF2 traces

% cat run.pbs
…
export TAU_TRACE=1
export TAU_TRACE_FORMAT=otf2

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

Performance Research Lab, University of Oregon, Eugene, USA

Support Acknowledgments

- US Department of Energy (DOE)
  - Office of Science contracts, ECP
  - 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















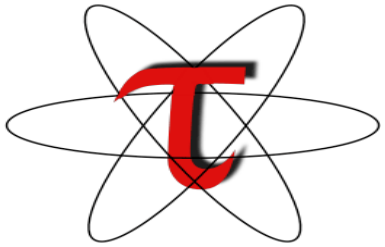




Acknowledgement

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]

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

Free download, open source, BSD license