



## TAU 2.18.2 Quick Reference

---

### What is TAU?

The TAU Performance System® is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, Java, Python. It comprises 3 main units: ***Instrumentation***, ***Measurement***, and ***Analysis***.

### Instrumentation

---

Instrumentation of user codes is performed via the ***tau\_compiler*** wrapper script system. We provide wrapper scripts for C, C++, and Fortran. Instrumentation is performed via either source code modification or compiler-based instrumentation.

For example:

Original command:

```
mpicc -c foo.c
mpicxx -c foo.cpp
mpif90 -c foo.f90
```

TAU instrumentation command:

```
tau_cc.sh -c foo.c
tau_cxx.sh -c foo.cpp
tau_f90.sh -c foo.f90
```

The ***tau\_compiler*** system is configurable via the **TAU\_OPTIONS** environment variable.

Common options:

<code>-optVerbose</code>	Enable verbose output (default: on)
<code>-optKeepFiles:</code>	Do not remove intermediate files
<code>-optShared:</code>	Use shared library of TAU
<code>-optCompInst:</code>	Use compiler-based instrumentation
<code>-optPDTInst:</code>	Use PDT-based instrumentation
<code>-optTauSelectFile="/path/to/select.tau":</code>	Specify selective instrumentation file

To choose a TAU installation configuration, set the **TAU\_MAKEFILE** environment variable to the chosen "stub makefile".



### Selective Instrumentation and Throttling

---

To reduce overhead and increase accuracy, it is recommended that some routines not be instrumented, those that are short running routines and are called many times. A good rule of thumb is to exclude those routines that have less than 10 microseconds per call inclusive time, and are called more than 100,000 times. By default, the TAU measurement system will throttle these routines at runtime. A throttled routine will still have hooks into the measurement system, but those hooks will be disabled, reducing, but not eliminating the overhead. To specify a selective instrumentation file, create a text file and use the following guide to fill it in.

- Wildcards for routine names are specified with the # mark (because \* symbols show up in routine signatures.) The # mark is unfortunately the comment character as well, so to specify a leading wildcard, place the entry in quotes.
- Wildcards for file names are specified with \* symbols.
- Exclude routines and files by specifying blocks as follows:

```
BEGIN_EXCLUDE_LIST
int foo(int)
void Sequencer::threadRun(Sequencer *)
"#H5#"
END_EXCLUDE_LIST
```

```
BEGIN_FILE_EXCLUDE_LIST
foo.f90
bar_*.c
END_FILE_EXCLUDE_LIST
```

- If a selective instrumentation file contains only *include* sections, then all routines/files are excluded unless specified in the include list. The include versions of these sections are `BEGIN_INCLUDE_LIST` and `BEGIN_FILE_INCLUDE_LIST`.
- Selective instrumentation files can be created automatically from ParaProf from the *File->Create Selective Instrumentation File* menu item. A command line utility, ***tau\_reduce*** also performs a similar task.



## TAU 2.18.2 Quick Reference

---

### Measurement

Once an application has been built with TAU, its performance can be measured using the TAU runtime library. The measurement library can be configured at runtime through the use of environment variables. The most important variables are given below:

TAU_VERBOSE	Enabled verbose output
TAU_METRICS*	Select metrics to measure (Default: WALLCLOCK)
TAU_PROFILE	Enable profile output (Default: on)
TAU_TRACE	Enable trace output
TAU_CALLPATH	Enable callpath profiling
TAU_CALLPATH_DEPTH	Set callpath depth (Default: 2)
TAU_COMPENSATE	Attempt to compensate for profiling overhead in profiles
TAU_TRACK_MESSAGE	Track MPI message statistics (profiling), messages lines (tracing)
TAU_COMM_MATRIX	Generate MPI communication matrix data
TAU_THROTTLE	Throttle lightweight routines (Default: on)
TAU_THROTTLE_PERCALL	Throttling per call threshold, in microseconds (Default: 10)
TAU_THROTTLE_NUMCALLS	Throttling number of calls threshold (Default: 100000)
TAU_SYNCHRONIZE_CLOCKS	For tracing, synchronize clocks between nodes (Default: on)

When the application completes execution, a profile or trace (if configured) will be generated for each node. They are named *profile.<nid>.0.<tid>*, or *tautrace.<nid>.0.<tid>*. Large sets of profiles can be packed into a single, compressed file using:

```
% paraprof --pack file.ppk
```

\* TAU\_METRICS specifies a colon-separate list of metrics. Every installation of TAU includes TIME (using unix gettimeofday). On Linux and CrayCNL systems, we provide the high resolution LINUXTIMERS metric, on BGL/BGP systems we provide BGLTIMERS and BGPTIMERS. If TAU has been built with PAPI (-papi=/path/to/papi), then all of the PAPI counters (given via PAPI's **papi\_avail** command) will be available, in addition we provide PAPI\_TIME and PAPI\_VIRTUAL\_TIME metrics to access the PAPI provided wallclock and virtual time metrics. An example TAU\_METRICS setting might be given with the following command.

```
% export TAU_METRICS="LINUXTIMERS:PAPI_FP_OPS"
```



### Analysis

---

To view profiles and traces, we provide the profile analyzers, **pprof** and **ParaProf**, and the trace visualize, **jumpshot**. To use pprof and paraprof, simply execute them inside the directory containing the profiles. For example, to see profiles on the command line, execute:

```
% pprof
```

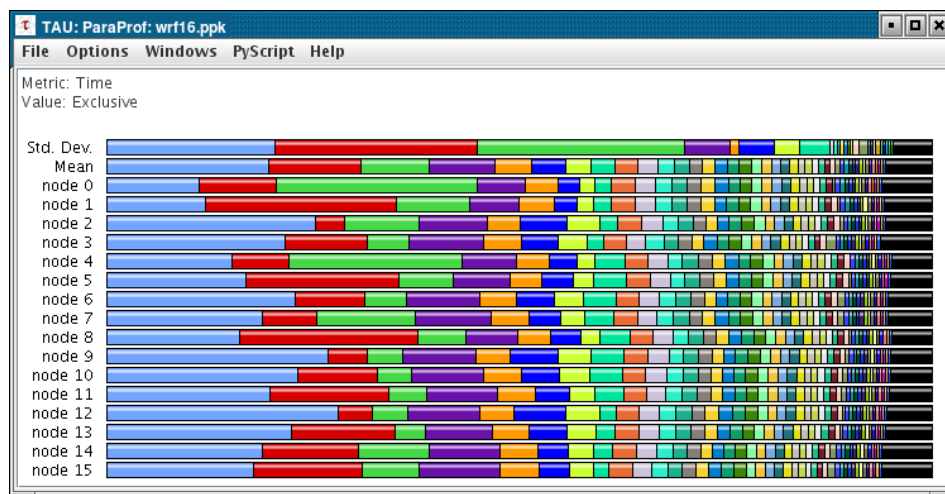
```
NODE 0;CONTEXT 0;THREAD 0:
```

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive Name usec/call
100.0	24	590	1	1	590963 main
95.9	26	566	1	2	566911 multiply
47.3	279	279	1	0	279280 multiply-opt
44.1	260	260	1	0	260860 multiply-regular

### ParaProf

---

To use launch ParaProf, execute **paraprof** from the command line where the profiles are located. Launching ParaProf will bring up the manager window and a window displaying the profile data as shown below.





## TAU 2.18.2 Quick Reference

---

### Jumpshot

---

To use Argonne's Jumpshot (bundled with TAU), first merge and convert TAU traces to slog2 format:

```
% tau_treemerge.pl
% tau2slog2 tau.trc tau.edf -o tau.slog2
% jumpshot tau.slog2
```

Launching Jumpshot will bring up the main display window showing the entire trace, zoom in to see more detail.

