TAU Commander Manual

Table of Contents

Tau Commander: Manual

Introduction:

TAU Commander activities are associated around three basic components: Target, Application and Measurement (TAM). This is illustrated in Figure 1. The first, **target**, describes the environment where data is collected. This includes the platform the code runs on, its operating system, CPU architecture, interconnectivity fabric, compilers, and installed software. The second is the **application**. The application consists of the underlying items associated with the application - whether the application uses MPI, OpenMP, threads, CUDA, OpenCL and such. The **measurements** define what

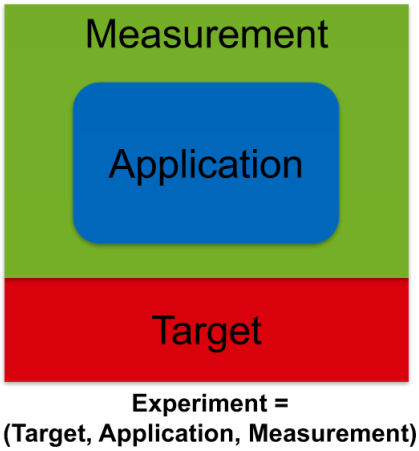

Figure 1 Basic Structure of Tau Commander

data will be collected and in what format. Even though an application uses OpenMP or MPI, the measurements may or may not measure those items In addition to that basic structure there are a couple of more components to complete the TAU Commander interface. The first is a project. A ***project*** is the container for the developers grouping of defined activities, settings and system environments. Last is the ***experiment***. An experiment consists of one target, one application and one measurement. One experiment is active and that is what will be executed when developers collect data. When an experiment is run and data is collected that completed data set is a ***trial***.

TAU Commander is supported on Linux*, and Apple MAC OS*.

Installation

TAU Commander is available from the github* repository. TAU Commander requires Python 2.7 or newer. The following command will retrieve TAU Commander for a system.

```
git clone https://github.com/ParaToolsInc/taucmdr.git
```

After doing this select where you want to install TAU Commander and use the selected path in the make install command. Then add the resulting /bin directory to your path. On Linux this might be done with:

```
cd taucmdr/bin
make install INSTALLDIR=<path>
export PATH=$PATH:<path>/bin
```

When this is completed you are ready to begin with TAU Commander. It is recommended to add the taucmdr/bin to the environment setup each time you log in so that TAU Commander is always in your path.

Initialize TAU Commander

To begin Initialize TAU: Enter `tau initialize` or simply `tau init`. This first initialization will take quite a bit of time. Not only is this command creating a project it is also downloading and building the TAU Performance System® and associated libraries that it depends on. Let this run and check for successful completion. When it completes it displays the basics of the project – shown in five tables in Figure 2.

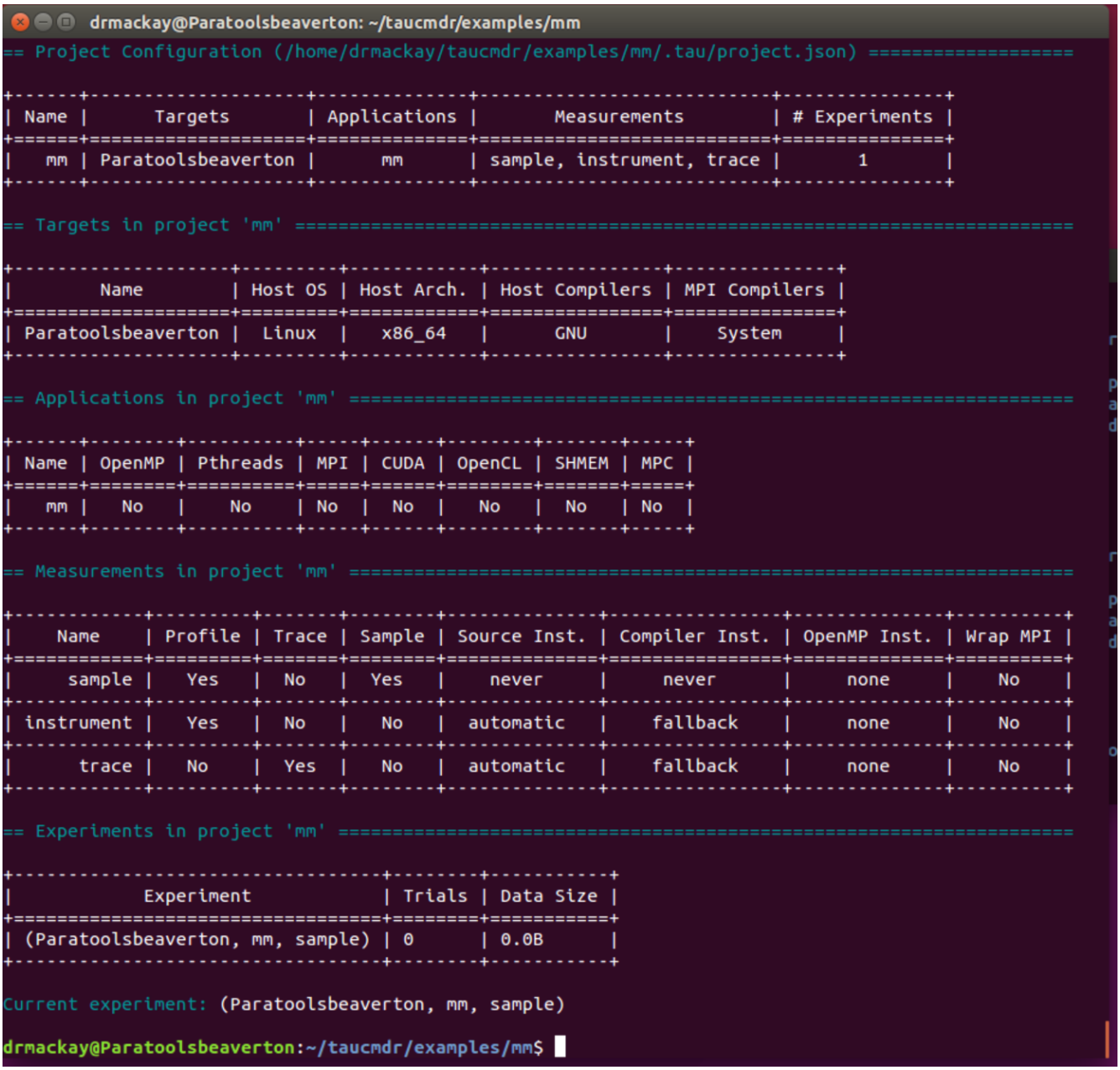

Figure 2: Tau dashboard displayed after first project initialized.

Many parameters may be defined at initialization. Most TAU Commander edit commands can be part of initialization (e.g. `tau init –MPI T`). Full list is in table below. See the various TAU Commander options for details. The default setting for many of the options is target dependent. To see default value type in –help to see settings on your target system.

Init option	Settings	Description	Defaults:
--bare	T/F	Initialize project but don't configure anything	Default: F
--tau-options	<option>	Add options to TAU_OPTIONS environment variables	Default: None
--help		Show help	
--application-name	<name>	Name of new application to be configured with init	Default: src
--cuda	T/F	Application uses cuda	
--linkage	Static/dynamic	Manner application is linked	
--mpc	T/F	Application uses mpc	
--mpi	T/F	Application uses MPI	
--opencl	T/F	Application uses opencl	
--openmp	T/F	Application uses OpenMP	
--pthreads	T/F	Application uses pthreads	
--select-file	<path>	Specify selective instrumentation file for TAU to use	
--shmem	T/F	Application uses SHMEM	
--tbb	T/F	Application uses Threading Building Blocks	
--callpath	<depth (#)>	Maximum depth for callpath recording measurement	Default: 100
--callsite	T/F	Record call site in measurement	Default: F
--compiler-inst	[mode (always, fallback, never)	Use compiler-generated callback to gather performance data	Default: never
--heap-usage	T/F	Measure heap memory usage	Default: F
--io	T/F	Measure time in POSIX I/O calls	Default: F
--keep-inst-files	T/F	Retain instrumented files after compilation	Default: F
--link-only	T/F	Don't instrument, only link the TAU library to application	Default: F
--memory-alloc	T/F	Record memory allocations/deallocations in measurements	Default: F
--metadata-merge	T/F	Merge metadata of TAU Profiles	Default: T
--metrics	<metric>	Performance Metrics to measure (e.g. PAPI events: TIME, PAPI_FP_INS, . .).	Default: TIME
--profile	[<format> (tau, merged, cubex, none)]	Measure application profiles	Default: tau

--reuse-inst-files	T//F	Reuse and preserve instrumented files after compilation	Default: F
--sample	T/F	Measure performance with event based sampling	Default: T
--source-inst	[mode (automatic, manual, never)]	Use hooks inserted into the application source code to gatgehr performance data	Default: automatic
--throttle	T/F	Throttle lightweight events to reduce overhead	
--throttle-num-calls	<count (#)>	Lightweight event call threshold	Default:10000
--throttle-per-call	[us (microseconds)]	Lightweight event duration threshold	Default: 10
--trace	[<format> (slog2, otf2, none)]	Generate application traces.	Default: otf2
--project (or --project-name)	<name>	Name of new project to initialize	Default: src
--storage-level	<levels (project, user, system)	Location of installation directory	
--arch	<arch> (options: x86_64, KNL, ppc64le, arm32, ppc64, BGL, KNC, BGP, BGQ, arm64, ibm64)	Host architecture	
--binutils	<path>	Path or url to a GNU binutlis installation or archive file	Default: download
--cc	<command>	Host C compiler command (e.g. gcc)	
--compilers	<family (Apple, BlueGene, Cray, GNU, IBM, Intel, PGI, System)>	Select host compilers from specified family	
--cuda-compilers	<family (IBM, NVIDIA)>	Select CUDA compilers	
--cuda-cxx	<command>	Cuda Compiler Command	
--cuda-fc	<command>	Cuda Fortan compiler command	
--cuda-toolkit	<path>	Path to NVIDIA CUDA Toolkit (enables opencl support)	
--cxx	<command>	Host C++ compiler command	
--fc	<command>	Host Fortran compiler command	
--from-tau-makefile	<path>	Populate targe configuration from a TAU Makefile (Warning overrides safety check)	

--libunwind	<path>| <url>|download>	Path or url for libunwind installation	Default: download
--mpi-cc	<command>	MPI C compiler	
--mpi-compilers	<family> (Cray, IBM, Intel, System)	Select all MPI compilers from given family	Default: System
--mpi-cxx	<command>	MPI C++ compiler	
--mpi-fc	<command>	MPI Fortran compiler	
--mpi-include-path	<paths>	Paths to search for MPI header files when building MPI applications	
--mpi-libraries	<paths>	Libraries to link when building MPI applications	
--mpi-library-path	<paths>	Paths to search for MPI library files when building MPI applications	
--ompt	(<path>|<url>|download| None)	Path or URL to OMPT installation or arechive file	Default: download
--os	<os> (options: CNL, Darwin, CNK, Android, Linux)	Host operating system	
--otf	<path>|<url>|download|None	Path or URL to libotf2 installation or archive file	Default: download
--papi	<path>|url>|download|None	Path or URL to PAPI installation or archive file	Default: download
--pdt	<path>|url>|download|None	Path or URL to PDT installation or archive file	Default: download
--scorep	<path>|url>|download|None	Path or URL to Score-P installation or archive file	Default: download
--shmem-cc	<command>	SHMEM C compiler command	
--shmem-compilers	<family> (options: Cray, OpenSHMEM)	Select all SHMEM compilers automatic from the given family, ignored if at least one SHMEM compiler is specified	
--shmem-cxx	<command>	SHMEM C++compiler command	
--shmem-fc	<command>	SHMEM Fortran compiler command	
--schmem-include-path	<paths>	Paths to search for SHMEM header files when building SHMEM applications	
--shmem-libraries	<flags>	Libraries to link when building SHMEM applications	
--shmem-library-path	<paths>	Paths to search for SHMEM library files when building SHMEM libraries	

--target-name	<name>	Name of new target configuration	
--tau	<path>|<url>|download|nightly	Path or URL to a TAU installation or archive file	Default: download
--upc	<command>	Universal Parallel C compiler command	

Building binaries with TAU Commander

After configuring TAU Commander and placing the bin directory in your path build with “tau” prefixes.
This is “`tau cc`” or “`tau f77`”, “`tau mpicc`”, . . .
Edit the Makefile to use compilers with “`tau`” prefixes. Changing the selected experiment typically will require that you rebuild your application binaries – depending on what changes. TAU Commander will remind you to rebuild your binary executable.

To selectively control instrumentation for files or routines that are part of the binary to be analyzed, see the select-file options in the TAU Commander Applications section.

TAU Commander Applications:

Copying a TAU Commander Application:

`tau application copy <existing_application_name> <new_application_name> [arguments]`

tau application copy arguments		
--cuda	[T/F]	Application uses NVIDIA CUDA
--linkage	<linkage> (linkage settings: static, dynamic)	Application linkage.
--mpc	[T/F]	Application uses MPC.
--mpi	[T/F	Application uses MPI
--opencl	[T/F]	Application uses OpenCL
--openmp	[T/F]	Application uses OpenMP
--pthreads	[T/F]	Application uses pthreads.
--select-file	<path>	Specify selective instrumentation file.
--shmem	[T/F]	Application uses SHMEM.
--tbb	[T/F]	Application uses Threading Building Blocks (TBB).

Optional Arguments:
-@ <level> Copy the application at the specified storage level.
-h, --help Show help message and exit.

Creating new applications:

Enter: `tau application create <new_application_name> [arguments]`

tau application create arguments		
--cuda	[T/F]	Application uses NVIDIA CUDA
--linkage	<linkage> (linkage settings: static, dynamic)	Application linkage.
--mpc	[T/F]	Application uses MPC.
--mpi	[T/F	Application uses MPI
--opencl	[T/F]	Application uses OpenCL
--openmp	[T/F]	Application uses OpenMP
--pthreads	[T/F]	Application uses pthreads.
--select-file	<path>	Specify selective instrumentation file.
--shmem	[T/F]	Application uses SHMEM.
--tbb	[T/F]	Application uses Threading Building Blocks (TBB).

Optional Arguments:
-@ <level> Copy the application at the specified storage level.
-h, --help Show help message and exit.

Delete a TAU Commander Application:

Enter: `tau application delete <application_name>`

Edit a TAU Commander Application:

Enter: `tau application edit <application_name> [arguments]`

Enter: `tau application create <new_application_name> [arguments]`

tau application edit arguments		
--cuda	[T/F]	Application uses NVIDIA CUDA
--linkage	<linkage> (linkage settings: static, dynamic)	Application linkage.
--mpc	[T/F]	Application uses MPC.
--mpi	[T/F	Application uses MPI
--new-name	<new_name>	Change the application's name
--opencl	[T/F]	Application uses OpenCL
--openmp	[T/F]	Application uses OpenMP
--pthreads	[T/F]	Application uses pthreads.
--select-file	<path>	Specify selective instrumentation file.
--shmem	[T/F]	Application uses SHMEM.
--tbb	[T/F]	Application uses Threading Building Blocks (TBB).

Optional Arguments:
-@ <level> Copy the application at the specified storage level.
-h, --help Show help message and exit.

see https://www.cs.uoregon.edu/research/tau/docs/newguide/bk01ch01s03.html for information about selective instrumentation file format.

List TAU Commander Applications in a project:

Enter: `tau application list`
`tau application list –l (long description)`
`tau application list –s (short description)`

TAU Commander Experiments

An experiment is defined by: a target, an application, a measurement.

Create experiments by entering:

```
tau experiment create <experiment_name> [arguments]
```

Tau experiment create arguments are :		
`--application`	`<name>`	Name of application configuration
`--measurement`	`<name>`	Name of measurement configuration
`--target`	`<name>`	Name of target hardware/software configutation

The entities experiment, application and measurement must already be defined.

Delete an experiment:

```
tau experiment delete <experiment_name>
```

Edit an experiment:

```
tau experiment edit <experiment_name> [arguments]
```

Tau experiment edit arguments are :		
`--application`	`<name>`	Name of application configuration
`--measurement`	`<name>`	Name of measurement configuration
`--new-name`	`<name>`	Change the experiment configuration's name
`--target`	`<name>`	Name of target hardware/software configutation

List experiments:

`tau experiment list` (optional `--l` or `--s` (long or short descriptions))

Select an experiment:

```
tau experiment select <experiment_name>
```

If the target, application, or measurement name can be implied then it may be omitted. For example, if you have only one target and only one application in your project then only the measurement name must be specified. So the command:

```
tau select <measurement_name>
```

will select as acive the experiment with the default target, application, measurement named "<target_name>-<application_name>-<measurement_name>." If that experiment does not already exist then TAU Commander will create a new experiment based on the names of the selected objects.

TAU Commander Measurements:

Copy a measurement:

```
tau measurement copy <measurement_name> <copy_name> [arguments]
```

tau measurement copy arguments		
--callpath	[depth]	Maximum depth for callpath recording
--metrics	<metric> [<metric> ...]	Performance metrics to gather, e.g. TIME, PAPI_FP_INS.
--compiler-inst	[mode] (modes: always, fallback, never)	Use compiler-generated callbacks to gather performance data.
--keep-inst-files	[T/F]	Don't remove instrumented files after compilation.
--link-only	[T/F]	Don't instrument, only link the TAU Library to the application.
--reuse-inst-files	[T/F]	Reuse and preserve instrumented files after compilation.
--sample	[T/F]	Use event-based sampling to gather performance data.
--source-inst	[mode] (modes: automatic, manual, never)	Use hooks inserted into the application source code to gather performance data.
--callsite	[T/F]	Record event callsites.
--comm-matrix	[T/F]	Record the point-to-point communication matrix.
--cuda	[T/F]	Measure cuda events via the CUPTI interface.
--io	[T/F]	Measure time spent in POSIX I/O calls.
--metadata-merge	[T/F]	Merge metadata of TAU profiles.
--mpi	[T/F]	Use MPI library wrapper to measure tIme spent in MPI methods.
--opencl	[T/F]	Measure OpenCL events
--openmp	[library] (library: ignore, opari, ompt)	Use specified library to measure time spent in OpenMP directives.
--shmem	[T/F]	Use SHMEM library wrapper to measure Time spent in SHMEM methods.
--throttle	[T/F]	Throttle lightweight events to reduceoverhead.
--throttle-num-calls	[count]	Lightweight event call count threshold.
--throttle-per-call	[us] (us is microseconds)	Lightweight event duration threshold in microseconds
--heap-usage	[T/F]	Measure heap memory usage.
--memory-alloc	[T/F]	Record memory allocation/deallocation events
--profile	[<format>] (format: tau, merged, cubex, none)	Generate application profiles.
--trace	[<format>] (format: slog2, otf2, none)	Generate application traces.

Optional Arguments:

-@ <level> Copy the application at the specified storage level.
-h, --help Show help message and exit.

Create a new measurement:

Enter `tau measurement create <measurement_name> [arguments]`

tau measurement create arguments		
`--callpath`	`[depth]`	Maximum depth for callpath recording
`--metrics`	`<metric> [<metric> ...]`	Performance metrics to gather, e.g. TIME, PAPI_FP_INS.
`--compiler-inst`	`[mode] (modes: always, fallback, never`	Use compiler-generated callbacks to gather performance data.
`--keep-inst-files`	`[T/F]`	Don't remove instrumented files after compilation.
`--link-only`	`[T/F]`	Don't instrument, only link the TAU Library to the application.
`--reuse-inst-files`	`[T/F]`	Reuse and preserve instrumented files after compilation.
`--sample`	`[T/F]`	Use event-based sampling to gather performance data.
`--source-inst`	`[mode] (modes: automatic, manual, never)`	Use hooks inserted into the application source code to gather performance data.
`--callsite`	`[T/F]`	Record event callsites.
`--comm-matrix`	`[T/F]`	Record the point-to-point communication matrix.
`--cuda`	`[T/F]`	Measure cuda events via the CUPTI interface.
`--io`	`[T/F]`	Measure time spent in POSIX I/O calls.
`--metadata-merge`	`[T/F]`	Merge metadata of TAU profiles.
`--mpi`	`[T/F]`	Use MPI library wrapper to measure tIme spent in MPI methods.
`--opencl`	`[T/F]`	Measure OpenCL events
`--openmp`	`[library] (library: ignore, opari, ompt)`	Use specified library to measure time spent in OpenMP directives.
`--shmem`	`[T/F]`	Use SHMEM library wrapper to measure Time spent in SHMEM methods.
`--throttle`	`[T/F]`	Throttle lightweight events to reduceoverhead.
`--throttle-num-calls`	`[count]`	Lightweight event call count threshold.
`--throttle-per-call`	`[us] (us is microseconds)`	Lightweight event duration threshold in microseconds
`--heap-usage`	`[T/F]`	Measure heap memory usage.
`--memory-alloc`	`[T/F]`	Record memory allocation/deallocation events
`--profile`	`[<format>] (format: tau, merged, cubex, none)`	Generate application profiles.
`--trace`	`[<format>] (format: slog2, otf2, none)`	Generate application traces.

Optional Arguments:
-@ <level> Copy the application at the specified storage level.
-h, --help Show help message and exit.

Delete TAU Commander Measurement:

```
tau delete <measurement_name>
```

Edit TAU Commander Measurements.

Enter `tau measurement edit <measurement_name> [arguments]`

tau measurement edit arguments		
`--callpath`	`[depth]`	Maximum depth for callpath recording
`--metrics`	`<metric> [<metric> ...]`	Performance metrics to gather, e.g. TIME, PAPI_FP_INS.
`--compiler-inst`	`[mode] (mode: always, fallback, never`	Use compiler-generated callbacks to gather performance data.
`--keep-inst-files`	`[T/F]`	Don't remove instrumented files after compilation.
`--link-only`	`[T/F]`	Don't instrument, only link the TAU Library to the application.
`--reuse-inst-files`	`[T/F]`	Reuse and preserve instrumented files after compilation.
`--sample`	`[T/F]`	Use event-based sampling to gather performance data.
`--source-inst`	`[mode] (mode: automatic, manual, never)`	Use hooks inserted into the application source code to gather performance data.
`--callsite`	`[T/F]`	Record event callsites.
`--comm-matrix`	`[T/F]`	Record the point-to-point communication matrix.
`--cuda`	`[T/F]`	Measure cuda events via the CUPTI interface.
`--io`	`[T/F]`	Measure time spent in POSIX I/O calls.
`--metadata-merge`	`[T/F]`	Merge metadata of TAU profiles.
`--mpi`	`[T/F]`	Use MPI library wrapper to measure tIme spent in MPI methods.
`--new-name`	`<new_name>`	Change the measurement configuration's name
`--opencl`	`[T/F]`	Measure OpenCL events
`--openmp`	`[library] (library: ignore, opari, ompt)`	Use specified library to measure time spent in OpenMP directives.
`--shmem`	`[T/F]`	Use SHMEM library wrapper to measure Time spent in SHMEM methods.
`--throttle`	`[T/F]`	Throttle lightweight events to reduceoverhead.
`--throttle-num-calls`	`[count]`	Lightweight event call count threshold.
`--throttle-per-call`	`[us] (us is microseconds)`	Lightweight event duration threshold in microseconds
`--heap-usage`	`[T/F]`	Measure heap memory usage.
`--memory-alloc`	`[T/F]`	Record memory allocation/deallocation events
`--profile`	`[<format>] (format: tau, merged, cubex, none)`	Generate application profiles.
`--trace`	`[<format>] (format: slog2, otf2, none)`	Generate application traces.

Optional Arguments:

-@ <level> Copy the application at the specified storage level.

-h, --help Show help message and exit.

The following measurement data arguments may be edited
`--callpath [depth] and –metrics [PAPI metrics]`
e.g. `tau measurement edit <measurement_name> --callpath 3`

The following measurement instrumentation settings may be defined with commands:
`--compiler-inst [mode]` (valid modes are: always, fallback, never – default = never)
`--source-inst [mode]` (valid modes are: automatic, manual, never – default = never)

List Measurements:
`tau measurement list` (optional `--l` or `--s` (long or short descriptions))

There are several TAU commands that can explicitly be added to the measurement property. Those interested can find more information in the TAU User Guide (https://www.cs.uoregon.edu/research/tau/docs/newguide/index.html).

Tau Commander Project commands

Copy a project:

```
tau project copy <project_name> <new_project_name> [optional args]
```

Tau project copy optional args		
--application	<name>	Names of application configurations in the project copy
--measurement	<name>	Names of measurement configurations in the project copy
--target	<name>	Names of target configurations in the project copy
-@	<level> (level options: project, user, system)	Copy the project at the specified storage level:
--help (or -h)		Show help message

Create a new project:

```
tau project create <project_name> [targets] [applications] [measurements]
```

Optional tau project create <project_name> arguments		
--applications	<name>	Names of application configurations in this project
--measurements	<name>	Names of measurement configurations in this project
--targets	<name>	Names of target configurations in this project
--help (or -h)		Show help message

Delete a project:

```
tau project delete <project_name>
```

Edit a project:

```
tau project edit <project_name> [arguments]
```

Tau project edit arguments		
--add	<conf> [<conf> …]	Add target, application, or measurement configurations to the project
--add-applications	< application> [<application> . . .]	Add application configurations to the project.
--add-measurements	<measurement> [<measurement> . . .]	Add measurement configurations to the project.
--add-targets	<target> [<target> ...]	Add target configurations to the project.
--new-name	<new_name>	Change the project's name.
--remove	<conf> [<conf> . . .]	Remove target, application, or measurement configurations from the project.
--remove-applications	<application> [<application> . ..]	Remove application configurations from the project.
--remove-measurements	<measurement> [<measurement . . .]	Remove measurement configurations from the project.
--remove-targets	<target> [<target> . . .]	Remove target configurations from the project.
-h, --help		Show help message and exit

List projects in directory:

```
tau project list
```

options: -d show project data in full dashboard
-h show help message

`-l` show project data in a list
`-s` show minimal project data

Select a specific project

```
tau project select <project_name>
```

Tau Commander – status

The status of the current project is displayed with dashboard

`tau dashboard` or
`tau dash`

This displays a table like that shown in Figure 2

TAU Commander trials

Creating a trial:

A trial is created by running the binary:

```
tau trial create ./a.out <args>
```

more commonly though it is simply created by using the tau prefix and execution command - that is:

```
tau ./a.out
```

or

```
tau mpirun -np # ./a.out <args>
```

The binary a.out will be executed and the TAU Commander specified data will be collected. This completes data collection to form a trial.

Delete a trial:

```
tau trial delete <trial_number>
```

Edit a trial:

```
tau trial edit <trial_number> --description <free form text>
```

Export a trial:

```
tau trial export <trial_number> [optional arg]
```

Optional argument is:

```
--destination <path>
```

Viewing data for a trial:

Enter `tau trial show` or `tau show` and TAU Commander will open up the appropriate display window to graphically show the data of the trial. This will be the last trial collected with the active or selected experiment. To see data for a different trial for this experiment just enter the trial number after show (e.g. `tau trial show 0`). Please note that trial numbers begin with 0. The first trial is trial number 0. If tau dashboard shows 3 trials those 3 trials are numbered 0, 1, 2

Alternatively, you may enter:

```
tau show <trial #>
```

This will have the same affect as:

```
tau trial show 0
```

You may also specify files instead of trial numbers, in this case enter:

```
tau show <data_file>
```

or

```
Tau trial show <data_file>
```

Optional arguments for Tau trial command are:

`--profile-tools <tool> (tool options: paraprof, pprof)` this specifies profile analysis tool

`--trace-tool <tool> (tool options: jumpshot, vampire)` Specify trace analysis tool

More Information

For more details of how to view data with paraprof see the paraprof user's manual:
https://www.cs.uoregon.edu/research/tau/docs/paraprof/index.html

For more details of how to view otf2 trace data see vampir user's manual:
https://tu-dresden.de/zih/forschung/ressourcen/dateien/projekte/vampir/dateien/Vampir-User-Manual.pdf?lang=en

For more details of how to view slog2 trace data see the jumpshot users manual:
http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/jumpshot-4/usersguide.html