Intuitive Performance Engineering with TAU Commander and ParaTools ThreadSpotter
< 1 year away!

Srinath Vadlamani
ParaTools, Inc.
srinathv@paratools.com

IXPUG Annual Meeting 2015, Oct. 1 , CRT, LBL, CA

Overview

- Peak at tools we have in the pipeline for to help applications be more HPC efficient —>KNLs
  - Review TAU
    - A complement to Intel performance engineering tools
  - Introduce interface to TAU usage: TC
  - Introduce memory analysis tool: PTTS
  - All are **open source**!
  - TAU examples:git clone git@github.com:ParaToolsInc/pt-ixpug2015.git

HPC tools in the pipeline

1)**TAU Commander (TC)**

An intuitive interface to
the TAU Performance System

2)ParaTools PTTS (PTTS)

Runtime reporting of memory usage
to guide efforts

Intuitive Performance Engineering

THE TAU PERFORMANCE SYSTEM

The TAU Performance System®

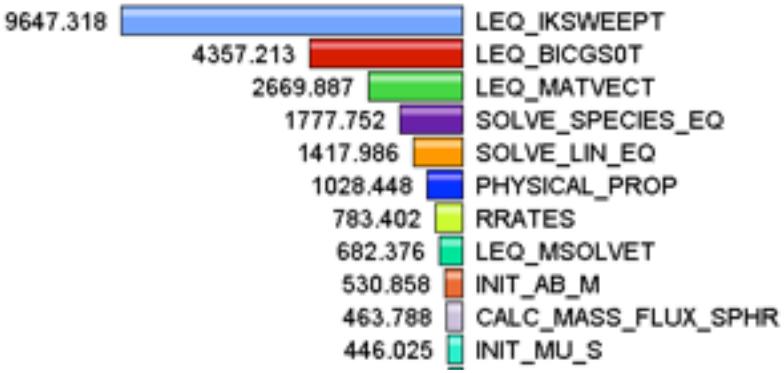
- Tuning and Analysis Utilities (**20+ year project**)
- Comprehensive performance profiling and tracing
  - Integrated, scalable, flexible, portable
  - Targets all parallel programming/execution paradigms
- Integrated performance toolkit
  - Instrumentation, measurement, analysis, visualization
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)
- Integrates with application frameworks
- Will work on KNL out of the box

TAU Supports All HPC Platforms

C/C++
CUDA
UPC
Python
GPI
Fortran
OpenACC
Java
MPI
pthreads
Intel MIC
OpenMP
Intel
GNU
LLVM
PGI
Cray
Sun
MinGW
AIX
Linux
Windows
Insert yours here
BlueGene
Fujitsu
ARM
Android
MPC
OS X

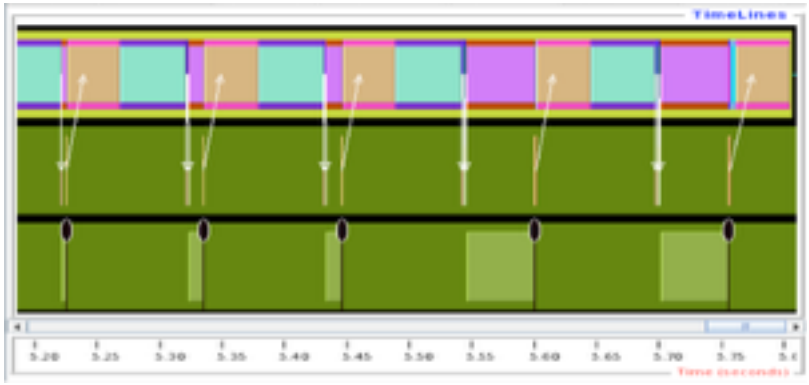
Measurement Approaches

Profiling



Shows **how much** time was spent in each routine

Tracing



Shows **when** events take place on a timeline

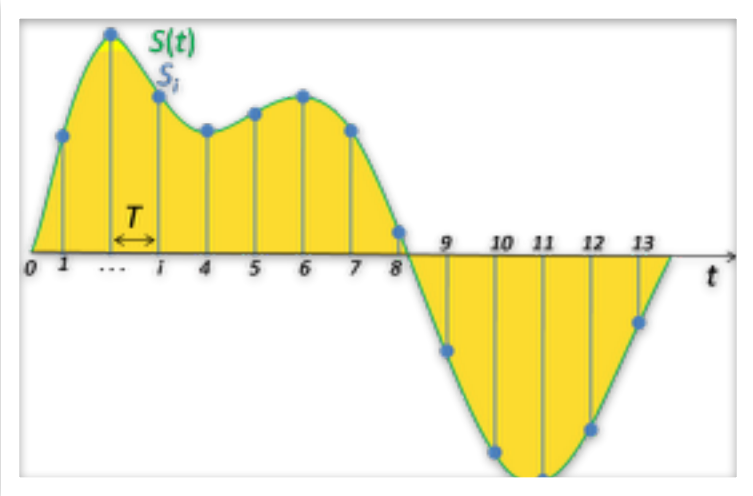
Performance Data Measurement

Direct via Probes

```
call TAU_START('potential')
// code
call TAU_STOP('potential')
```

- Exact measurement
- Fine-grain control
- Calls inserted into code

Indirect via Sampling



- No code modification
- Minimal effort
- Relies on debug symbols (**-g** option)

Insert TAU API Calls Automatically

- Use TAU's compiler wrappers
  - Replace `cxx` with `tau_cxx.sh, etc.`
  - Automatically instruments source code, links with TAU libraries.
- Use `tau_cc.sh` for C, `tau_f90.sh` for Fortran, etc.

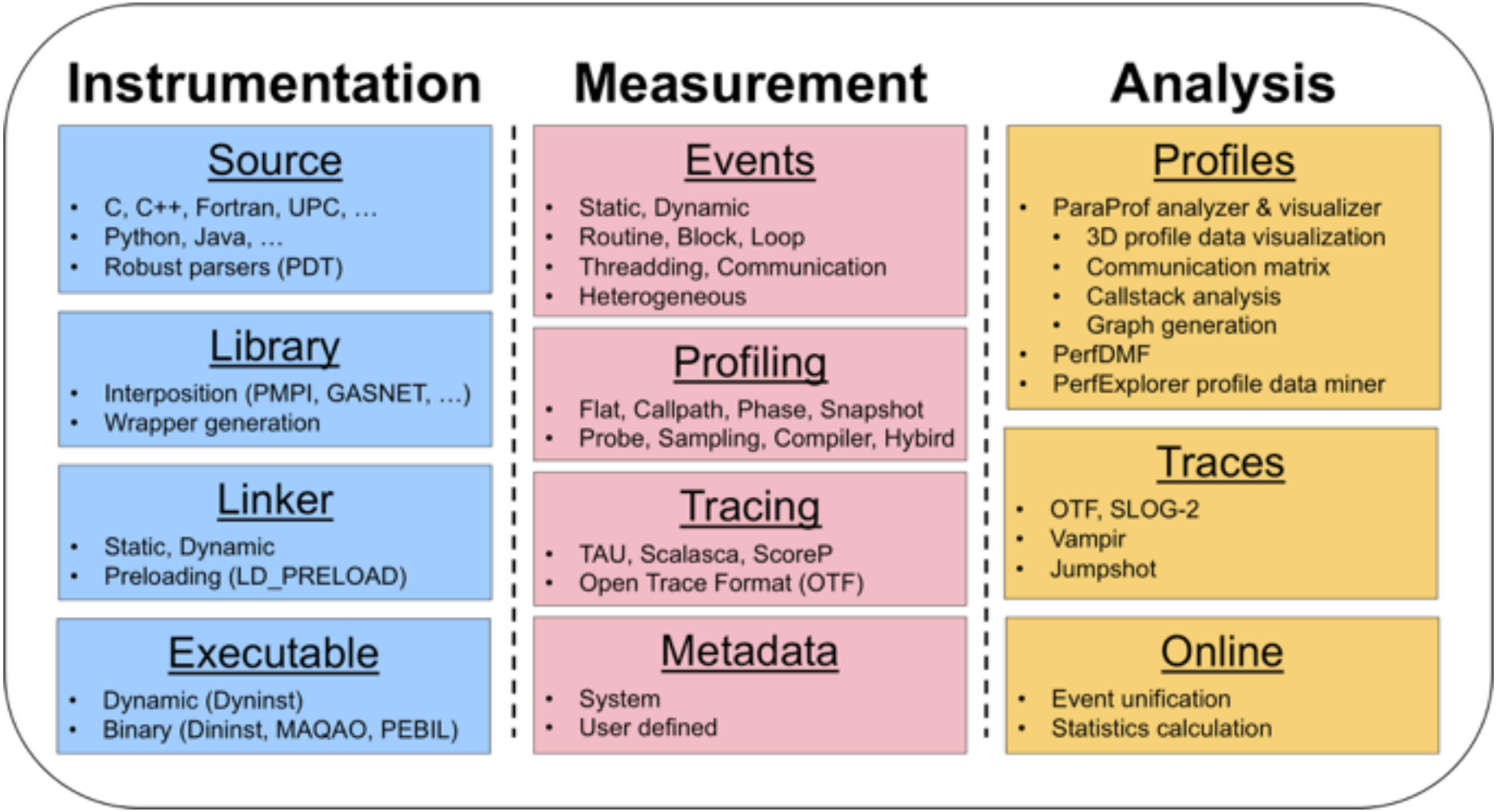
Makefile without TAU

```
CXX = mpicxx
F90 = mpif90
CXXFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
     $(CXX) $(LDFLAGS) $(OBJS) -o $@
     $(LIBS)
.cpp.o:
     $(CXX) $(CXXFLAGS) -c $<
```

Makefile with TAU

```
CXX = tau_cxx.sh
F90 = tau_f90.sh
CXXFLAGS =
LIBS = -lm
OBJS = f1.o f2.o f3.o … fn.o

app: $(OBJS)
     $(CXX) $(LDFLAGS) $(OBJS) -o $@
        $(LIBS)
.cpp.o:
     $(CXX) $(CXXFLAGS) -c $<
```

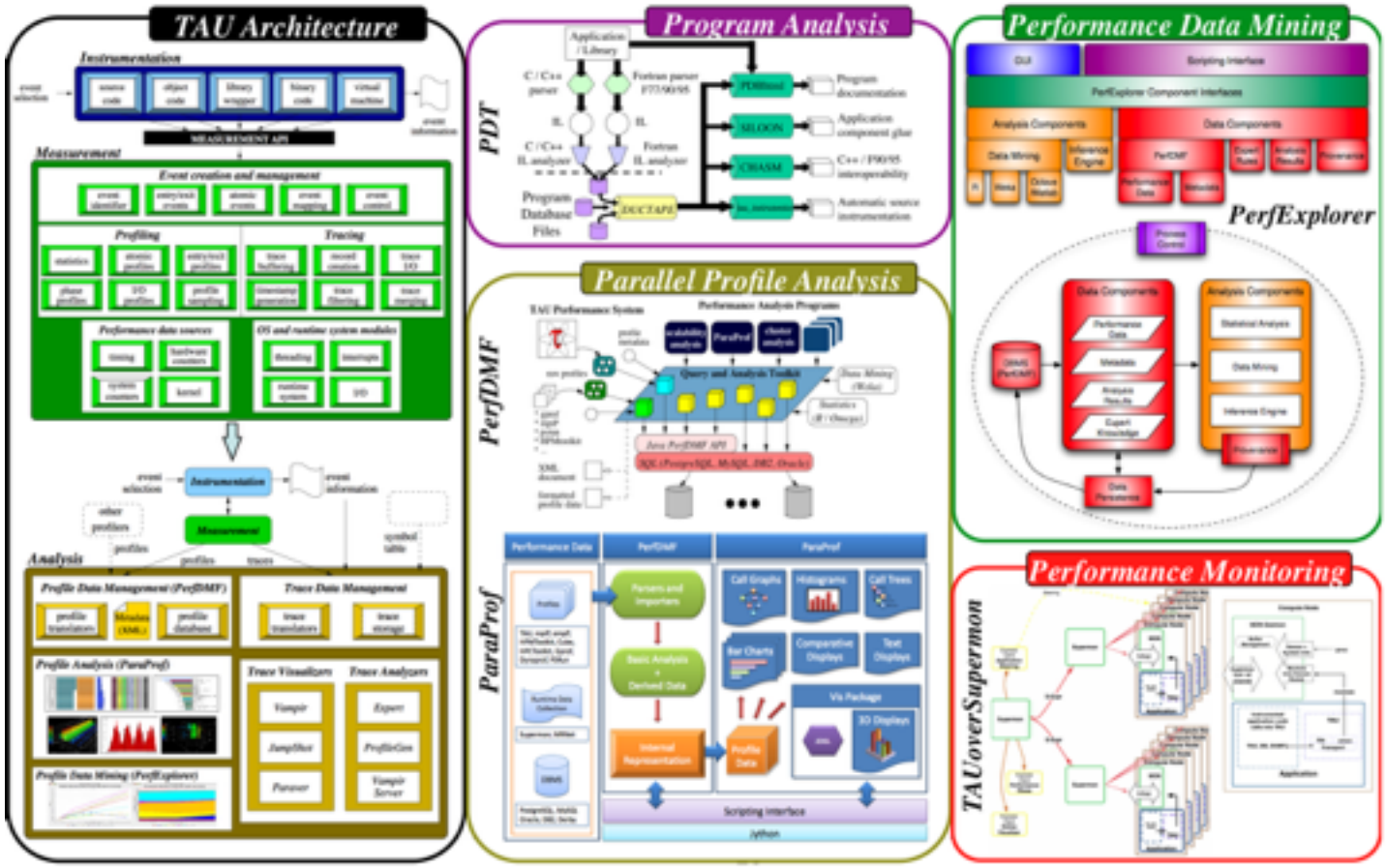
Performance Engineering Workflow



Intuitive Performance Engineering

TAU COMMANDER

TAU: Powerful and Complex



How do we navigate?

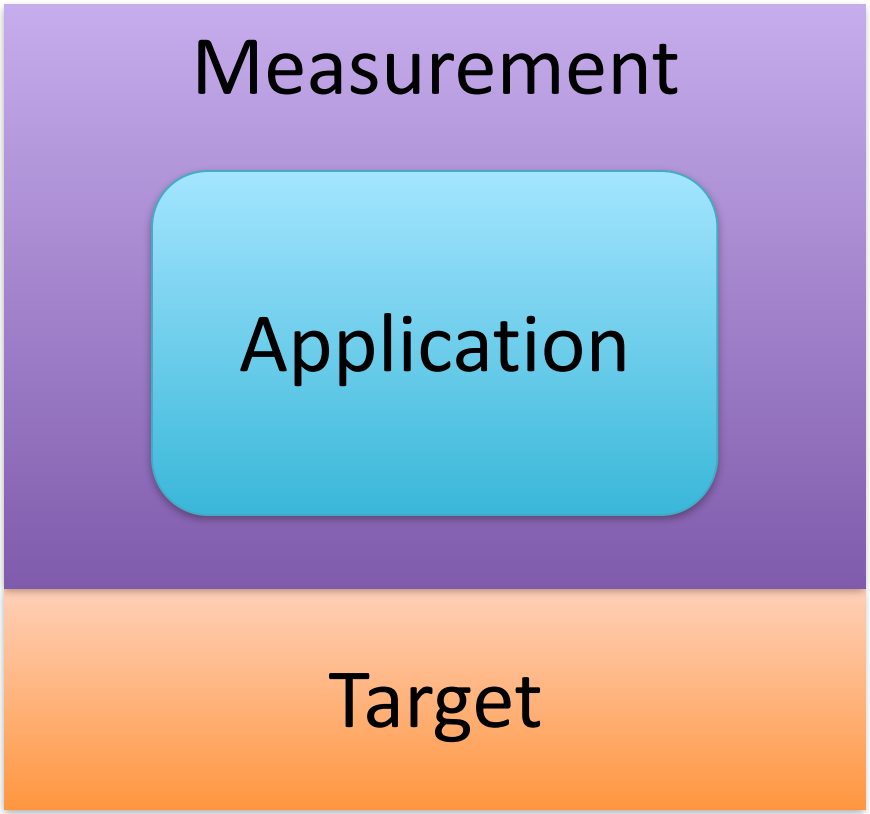
The TAU Commander Approach

- Say where you're going, not how to get there
- **TAU Projects** give context to the user's actions
  - Defines desired metrics and measurement approach
  - Defines operating environment
  - Establishes a baseline for error checking

vs.

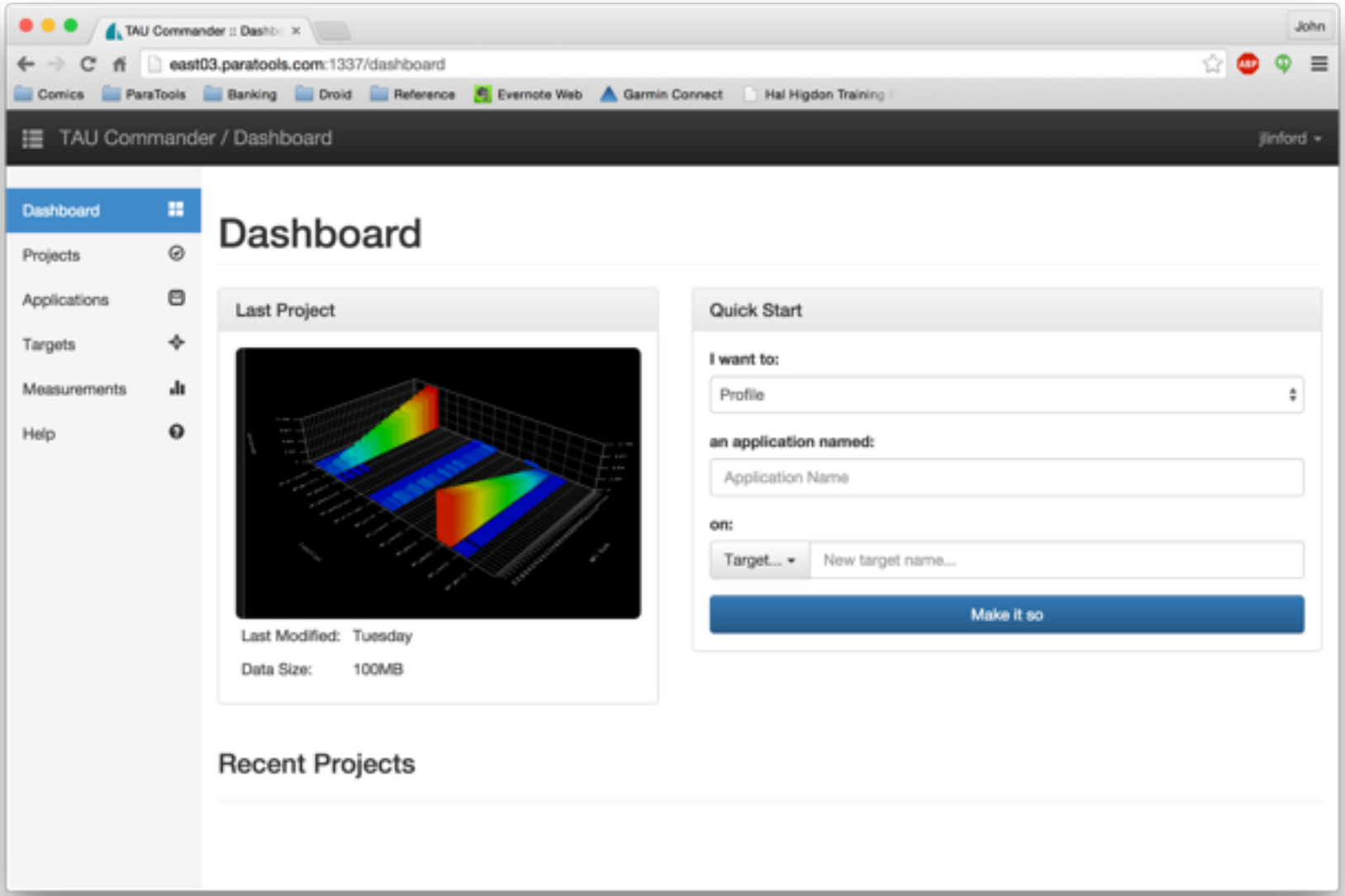
T-A-M Model for Performance Engineering

- Target
  - Installed software
  - Available compilers
  - Host architecture/OS
- Application
  - MPI
  - CUDA
  - Xeon Phi
- Measurement
  - Profile, trace, or both
  - Sample, source inst.



TAU Experiment =
(Target, Application, Measurement)

TAU Commander GUI (under devel.)



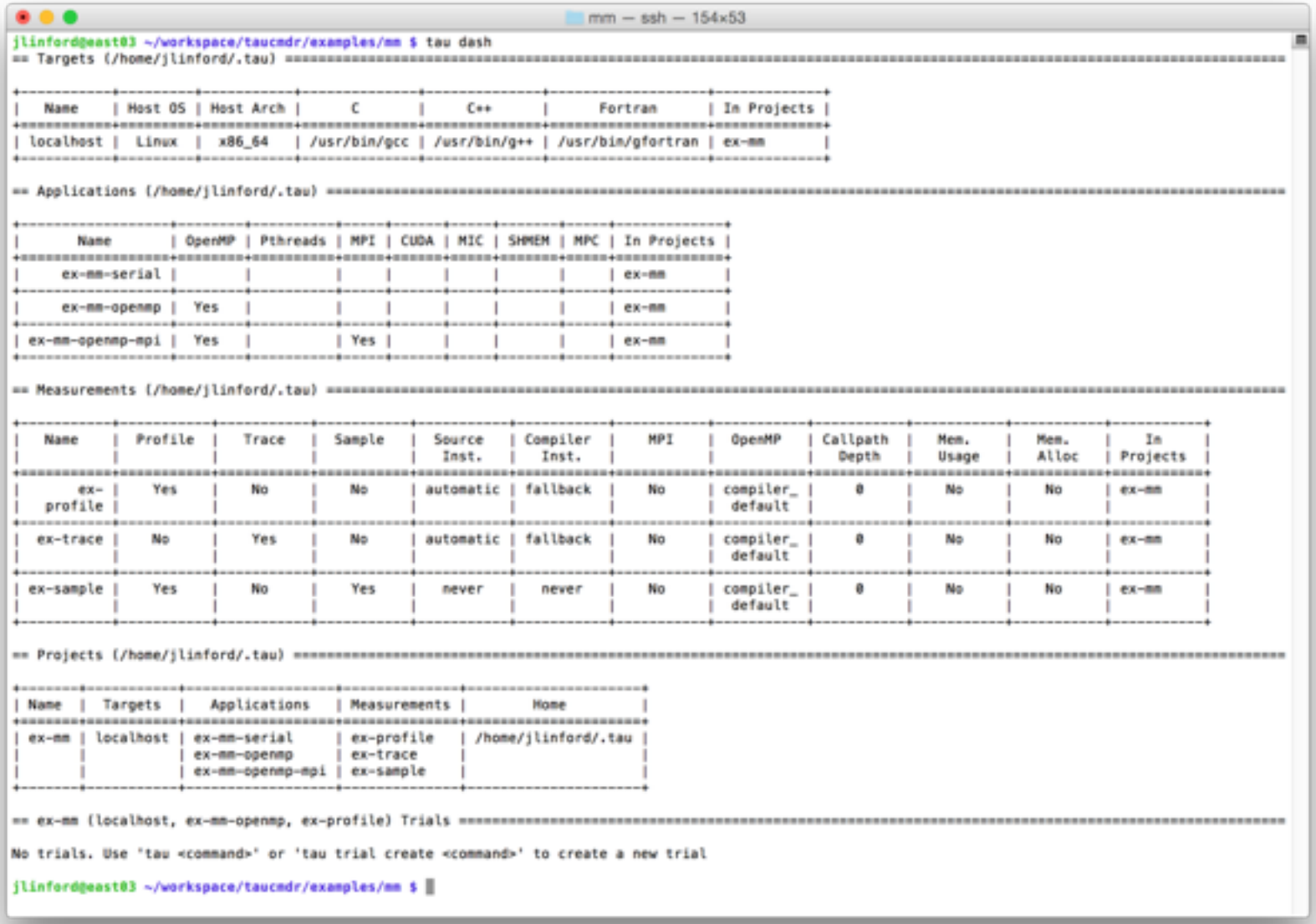
TAU Commander CLI

This command's usage

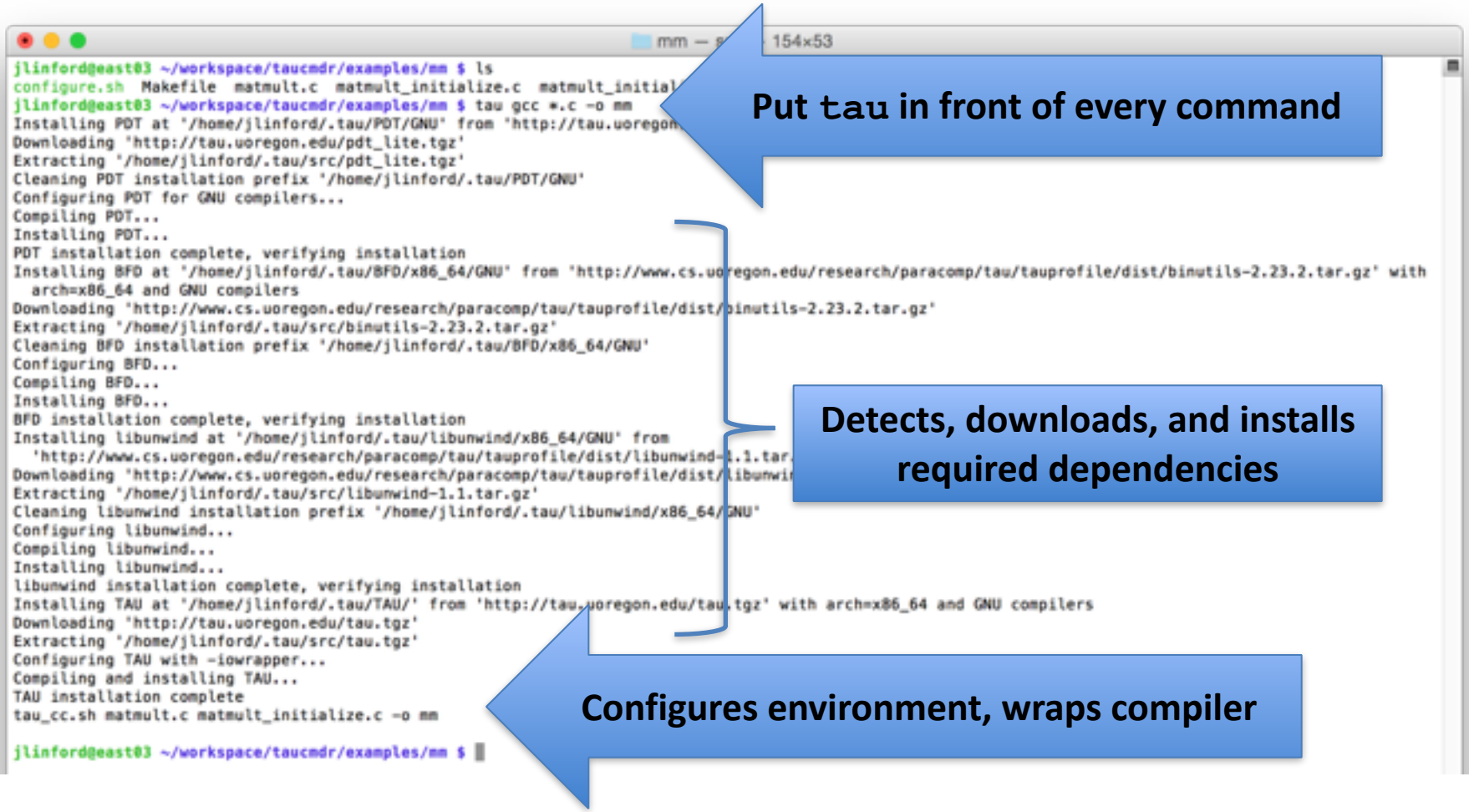
Subcommand usage

Shortcuts

TAU Commander CLI Dashboard



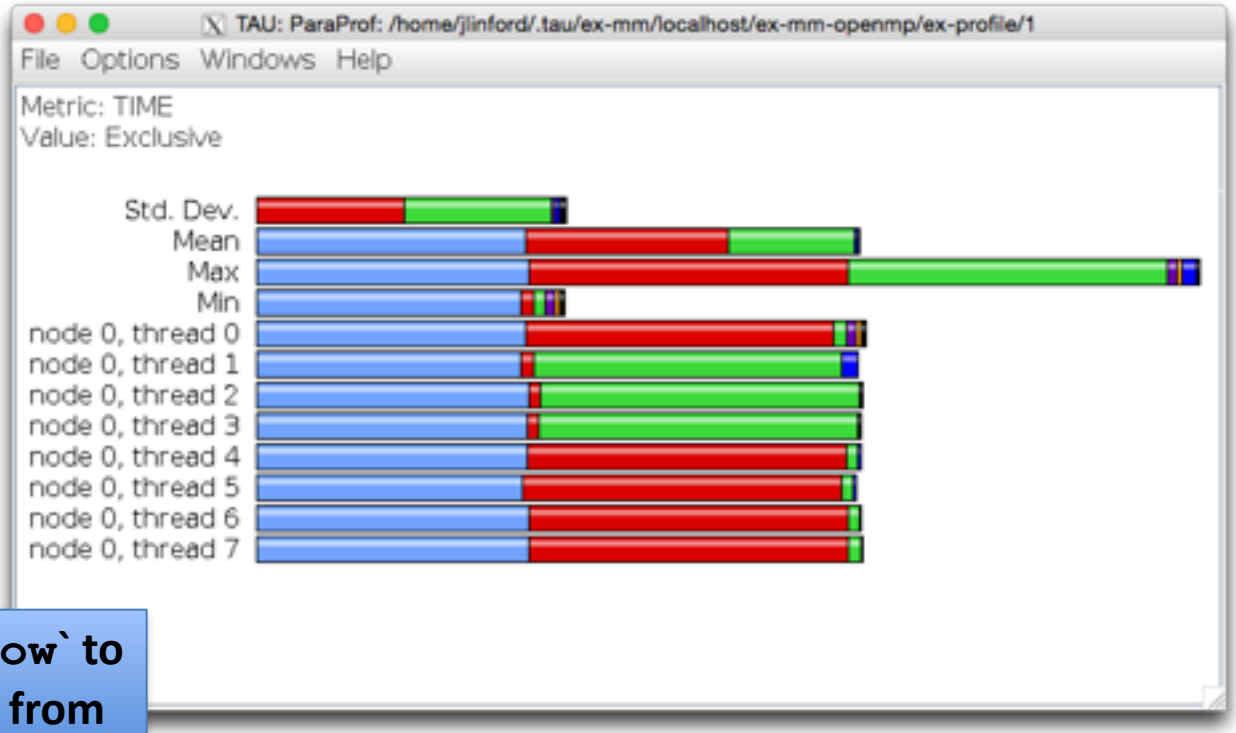
First use on a “vanilla” system



Executions create experiment trials

```
mm — ssh — 154x
jlinford@east03 ~/workspace/taucmdr/examples/mm $ export OMP_NUM_THREADS=8
jlinford@east03 ~/workspace/taucmdr/examples/mm $ tau ./mm
== BEGIN ex-mm (localhost, ex-mm-openmp, ex-profile) (2015-08-12 04:18:35.372290) ===
./mm
Done.
Found 8 profile files. Adding to trial...
== END ex-mm (localhost, ex-mm-openmp, ex-profile) (2015-08-12 04:18:36.593917) ==========
jlinford@east03 ~/workspace/taucmdr/examples/mm $
```

Put `tau` in front of every command



`tau show` to see data from last trial

Executions create experiment trials

mm — ssh — 154×53

```
jlinford@east03 ~/workspace/taucmdr/examples/mm $ tau dash
== Targets (/home/jlinford/.tau) ==============================================================================================================

+-----------+---------+-----------+-------------+-------------+-----------------+-------------+
|   Name    | Host OS | Host Arch |      C      |     C++     |     Fortran     | In Projects |
+===========+=========+===========+=============+=============+=================+=============+
| localhost |  Linux  |  x86_64   | /usr/bin/gcc | /usr/bin/g++ | /usr/bin/gfortran | ex-mm       |
+-----------+---------+-----------+-------------+-------------+-----------------+-------------+

== Applications (/home/jlinford/.tau) =========================================================================================================

+------------------+--------+----------+-----+------+-----+-------+-----+-------------+
|       Name       | OpenMP | Pthreads | MPI | CUDA | MIC | SHMEM | MPC | In Projects |
+==================+========+==========+=====+======+=====+=======+=====+=============+
|     ex-mm-serial |        |          |     |      |     |       |     | ex-mm       |
+------------------+--------+----------+-----+------+-----+-------+-----+-------------+
|     ex-mm-openmp |  Yes   |          |     |      |     |       |     | ex-mm       |
+------------------+--------+----------+-----+------+-----+-------+-----+-------------+
| ex-mm-openmp-mpi |  Yes   |          | Yes |      |     |       |     | ex-mm       |
+------------------+--------+----------+-----+------+-----+-------+-----+-------------+

== Measurements (/home/jlinford/.tau) =========================================================================================================

+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|   Name    |  Profile  |   Trace   |  Sample   |  Source   | Compiler  |    MPI    |  OpenMP   | Callpath  |   Mem.    |   Mem.    |
|           |           |           |           |   Inst.   |   Inst.   |           |           |   Depth   |   Usage   |   Alloc   |
+===========+===========+===========+===========+===========+===========+===========+===========+===========+===========+===========+
|  ex-      |    Yes    |    No     |    No     | automatic | fallback  |    No     | compiler_ |     0     |    No     |    No     |
|  profile  |           |           |           |           |           |           |  default  |           |           |           |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| ex-trace  |    No     |    Yes    |    No     | automatic | fallback  |    No     | compiler_ |     0     |    No     |    No     |
|           |           |           |           |           |           |           |  default  |           |           |           |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| ex-sample |    Yes    |    No     |    Yes    |   never   |   never   |    No     | compiler_ |     0     |    No     |    No     |
|           |           |           |           |           |           |           |  default  |           |           |           |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+

== Projects (/home/jlinford/.tau) =============================================================================================================

+-------+-----------+-----------------+--------------+---------------------+
| Name  |  Targets  |   Applications  | Measurements |        Home         |
+=======+===========+=================+==============+=====================+
| ex-mm | localhost | ex-mm-serial    | ex-profile   | /home/jlinford/.tau |
|       |           | ex-mm-openmp    | ex-trace     |                     |
|       |           | ex-mm-openmp-mpi | ex-sample    |                     |
+-------+-----------+-----------------+--------------+---------------------+

== ex-mm (localhost, ex-mm-openmp, ex-profile) Trials =========================================================================================

  2 trials of 'mm' (22.6KiB).  Use `tau trial list` to see details.

jlinford@east03 ~/workspace/taucmdr/examples/mm $
```

Each execution is a new trial

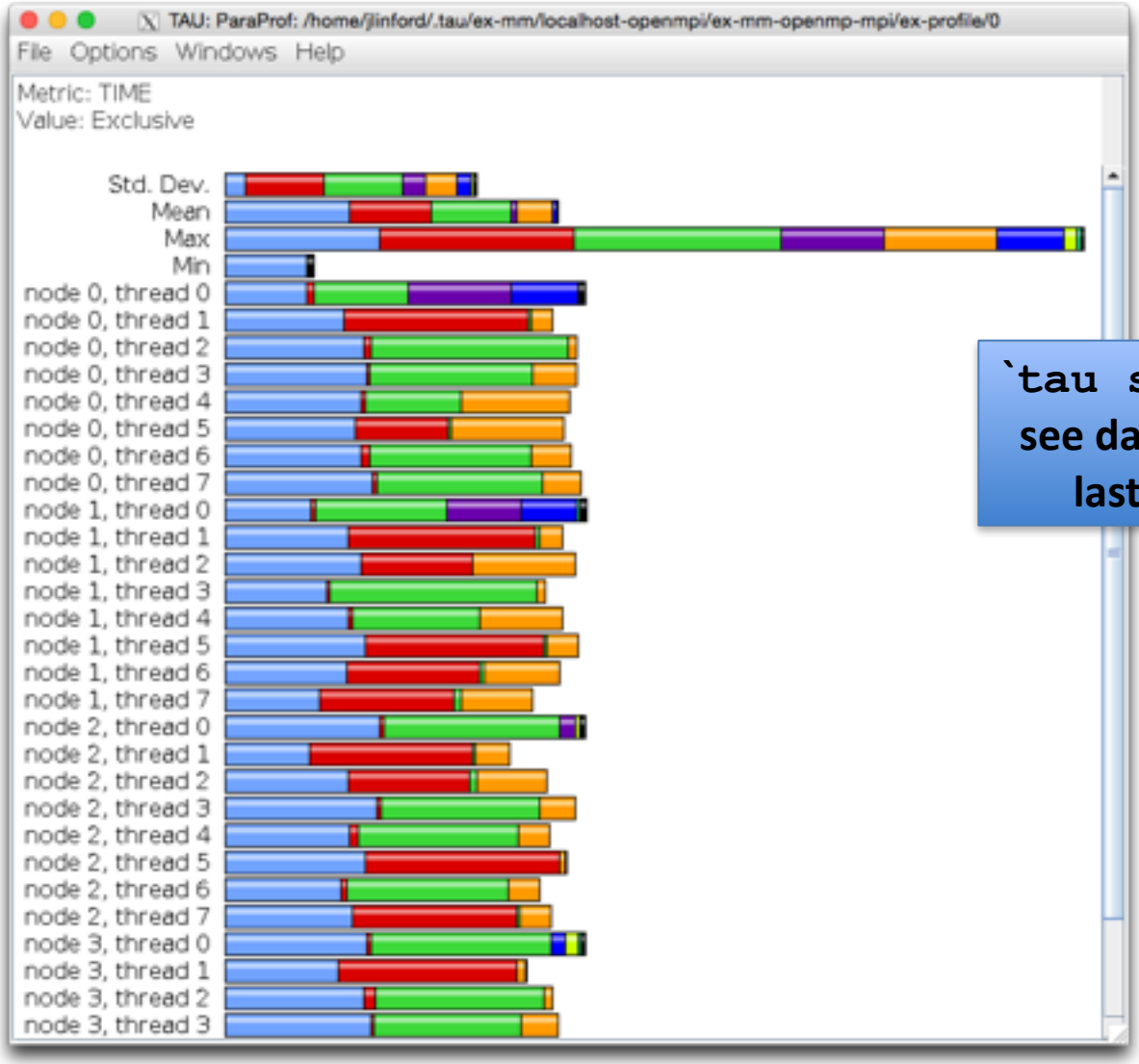
Changing from serial to MPI+OpenMP

```
== Projects (/home/jlinford/.tau) ==========

+-------+------------------+-----------------+-------------+---------------------+
| Name  |      Targets     |   Applications  | Measurements |        Home         |
+=======+==================+=================+=============+=====================+
| ex-mm | localhost        | ex-mm-serial    | ex-profile  | /home/jlinford/.tau |
|       | localhost-openmpi | ex-mm-openmp    | ex-trace    |                     |
|       |                  | ex-mm-openmp-mpi | ex-sample   |                     |
+-------+------------------+-----------------+-------------+---------------------+

== ex-mm (localhost-openmpi, ex-mm-openmp-mpi, ex-profile) Trials ==========

No trials. Use 'tau <command>' or 'tau trial create <command>' to create a new tr

jlinford@east03 ~/workspace/taucmdr/examples/mm $ tau mpicc *.c -fopenmp -o mm
Installing TAU at '/home/jlinford/.tau/TAU/' from 'http://tau.uoregon.edu/tau.tgz' with arch=x86_64 and MPI compilers
Using TAU source archive at '/home/jlinford/.tau/src/tau.tgz'
Reusing TAU source files found at '/home/jlinford/.tau/src/./tau-2.24.1'
Configuring TAU with -iowrapper...
Compiling and installing TAU...
TAU installation complete
tau_cc.sh matmult.c matmult_initialize.c -fopenmp -o mm

jlinford@east03 ~/workspace/taucmdr/examples/mm $
```

Put tau in front of every command

Automatically reconfigures TAU for MPI+OpenMP

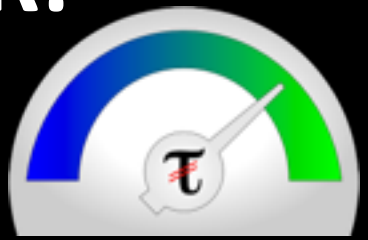
Workflow is unchanged



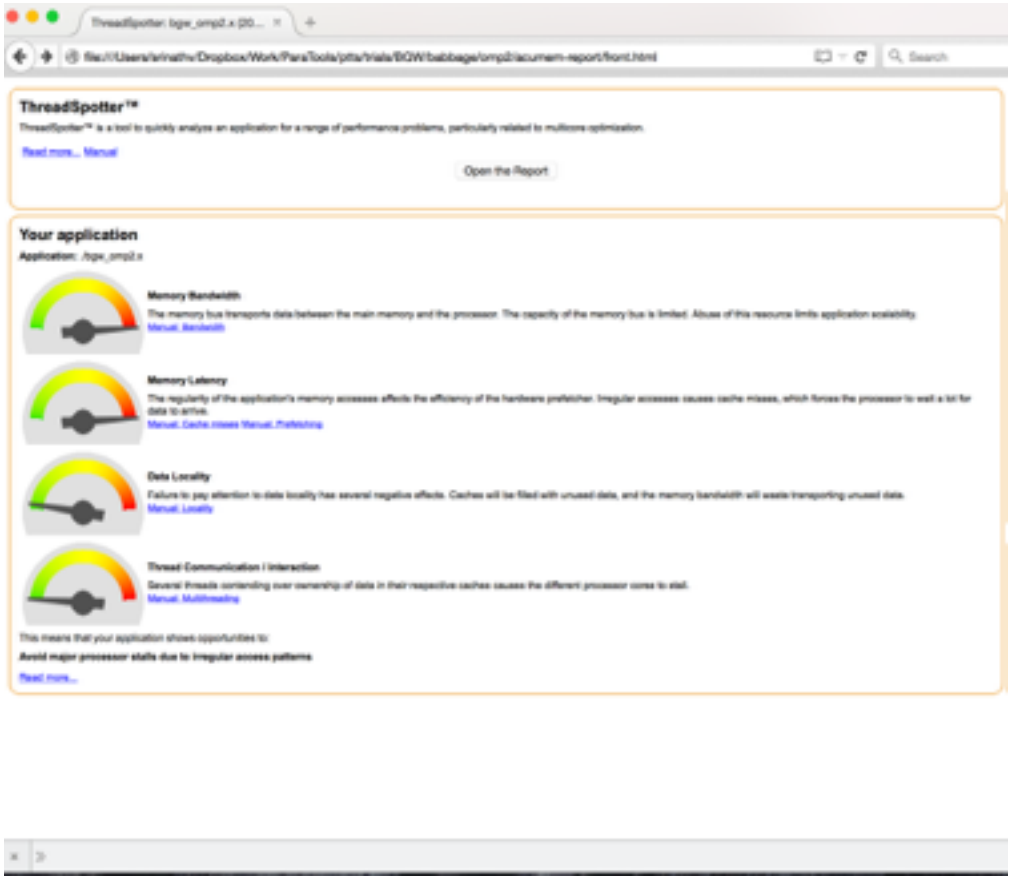
`tau show` to see data from last trial

Tool for Memory Insight

PARATOOLS THREADSPOTTER: PTTS



Compiler agnostic memory usage report via sampling.



Report topics:

- Memory Bandwidth
  - Ex: fetch ratios to main memory
- Memory Latency
  - Ex: prefetching
- Data Locality
  - *temporal* - reuse
  - *spatial* - already in cache
- threading
  - Ex: false sharing
  - Ex: cache line transfers

How to use PTTS

- You choose your compiler and run configurations.
- Sample at runtime
  - PTTS uses OS description of memory layout
  - System configurations can be supplied to PTTS
- Generate report information
  - Can be focused on different levels of memory
- Generate viewable report -> html or pdf ..

ParaTools PTTS

CASE STUDY: BGW (FORTRAN +OPENMP)

Very simple process to generate reports

- ifort -O3 -g -xAVX -openmp bgw.f90 -o bgw.x
- /project/projectdirs/acts/vadlaman/ptts/1.2.2/bin/sample -r ./bgw.x
- /project/projectdirs/acts/vadlaman/ptts/1.2.2/bin/report -i sample.smp
- /project/projectdirs/acts/vadlaman/ptts/1.2.2/bin/view-static -i report.tsr

BGW timings

BGW version	*OpenMP*	*improvement*	*timing (sec)*
bgw.x	***none***		***~60***
bgw_omp1.x	***16 threads***	***parallel/dos***	***~20***
bgw_winner.x	***16 threads***	***3D complex array -> 2d complex array***	***~6***

ParaTools PTTS

CASE STUDY: GFMCMK(FORTRAN +OPENMP)

Good memory use !=> good threading

- https://asc.llnl.gov/CORAL-benchmarks/
- GFMCmk -> stassuij algorithm
- picked to demonstrate good memory access
- Demonstration: Intel 16 on Babbage using 16 threads

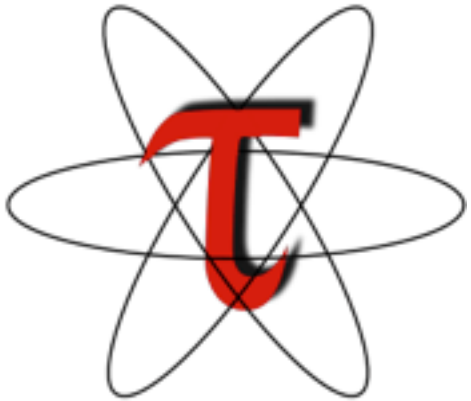
Intuitive Performance Engineering

CONCLUSION

Portable, open source and *friendly*

- TAU Commander (TC):
  - Make performance engineering easier from beginning to end.
  - Future: Jupyter for GUI and analysis
- ParaTools ThreadSpotter (PTTS):
  - Learn about memory usage of your application
  - Guided efforts
  - Future:
    - MPI support
    - Better use of architecture specifications
    - Streamline report generator

Downloads



http://github.com/ParaToolsInc/taucmdr

http://www.paratools.com/threadspotter

Free download, open source, BSD license

Acknowledgements

- Department of Energy
  - Office of Science
  - Argonne National Laboratory
  - Oak Ridge National Laboratory
  - NNSA/ASC Trilabs (SNL, LLNL, LANL)
- HPCMP DoD PETTT Program
- National Science Foundation
  - Glassbox, SI-2
- University of Tennessee
- University of New Hampshire
  - Jean Perez, Benjamin Chandran
- University of Oregon
  - Allen D. Malony, Sameer Shende
  - Kevin Huck, Wyatt Spear
- TU Dresden
  - Holger Brunst, Andreas Knupfer
  - Wolfgang Nagel
- Research Centre Jülich
  - Bernd Mohr
  - Felix Wolf





Intuitive Performance Engineering

REFERENCE

Online References

- PAPI:
  - PAPI documentation is available from the PAPI website:

    http://icl.cs.utk.edu/papi/
- TAU:
  - TAU Users Guide and papers available from the TAU website: **http://tau.uoregon.edu/**
- VAMPIR:
  - VAMPIR website:

    http://www.vampir.eu/
- Scalasca:
  - Scalasca documentation page:

    http://www.scalasca.org/
- Eclipse PTP:
  - Documentation available from the Eclipse PTP website:

    http://www.eclipse.org/ptp/

Compiling Fortran Codes with TAU

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

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

- **If your Fortran code uses C preprocessor directives (#include, #ifdef, #endif):**
  % export TAU_OPTIONS='-optPreProcess –optVerbose'

- **To use an instrumentation specification file:**
  % export TAU_OPTIONS=
  '-optTauSelectFile=select.tau -optVerbose -optPreProcess’

Example select.tau file

```
BEGIN_INSTRUMENT_SECTION
loops file="*" routine="#"
memory file="foo.f90" routine=”#"
io file="abc.f90" routine="FOO"
END_INSTRUMENT_SECTION
```

Generate a PAPI profile with 2 or more counters

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% export TAU_OPTIONS='-optTauSelectFile=select.tau -optVerbose'
% cat select.tau
  BEGIN_INSTRUMENT_SECTION
  loops routine="#"
  END_INSTRUMENT_SECTION

% export PATH=$TAU_ROOT/bin:$PATH
% make F90=tau_f90.sh
(Or edit Makefile and change F90=tau_f90.sh)
%
% qsub --env TAU_METRICS=TIME:PAPI_FP_INS:PAPI_L1_DCM -n 4 -t 15 ./a.out
% paraprof --pack app.ppk
   Move the app.ppk file to your desktop.
% paraprof app.ppk
  Choose Options -> Show Derived Metrics Panel -> "PAPI_FP_INS", click "/", "TIME", click
    "Apply"  and choose the derived metric.
```

Tracking I/O in static binaries

```
% export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-papi-mpi-pdt
% export PATH=$TAU_ROOT/bin:$PATH
% export TAU_OPTIONS='-optTrackIO -optVerbose'
% make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
% mpirun -n 4 ./a.out
% paraprof -pack ioprofile.ppk
% export TAU_TRACK_IO_PARAMS 1
% mpirun -n 4 ./a.out (to track parameters used in POSIX I/O calls as
   context events)
```

Installing and Configuring TAU

- Installing PDT:
  - wget http://tau.uoregon.edu/pdt.tgz
  - ./configure –prefix=<dir>; make ; make install
- Installing TAU:
  - wget http://tau.uoregon.edu/tau.tgz
  - ./configure -bfd=download -pdt=<dir> -papi=<dir> ...
  - make install
- Using TAU:
  - export TAU_MAKEFILE=<taudir>/<arch>/lib/Makefile.tau-<TAGS>
  - make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh

Compile-Time Options (TAU_OPTIONS)

% 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
-optMemDbg	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)
-optPdtF95Opts=""	Add options for Fortran parser in PDT (f95parse/gfparse) …

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_LEAKS	0	Setting to 1 turns on leak detection (for use with –optMemDbg or tau_exec)
TAU_MEMDBG_PROTECT_ABOVE	0	Setting to 1 turns on bounds checking for dynamically allocated arrays. (Use with –optMemDbg or tau_exec –memory_debug).
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_TRACK_IO_PARAMS	0	Setting to 1 with –optTrackIO or tau_exec –io captures arguments of I/O calls
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. Enabled by default to remove 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_COMPENSATE	0	Setting to 1 enables runtime compensation of instrumentation overhead
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., TIME:P_VIRTUAL_TIME:PAPI_FP_INS:PAPI_NATIVE_<event>\\:<subevent>)