TAU Commander and ParaTools ThreadSpotter

ParaTools, Inc.

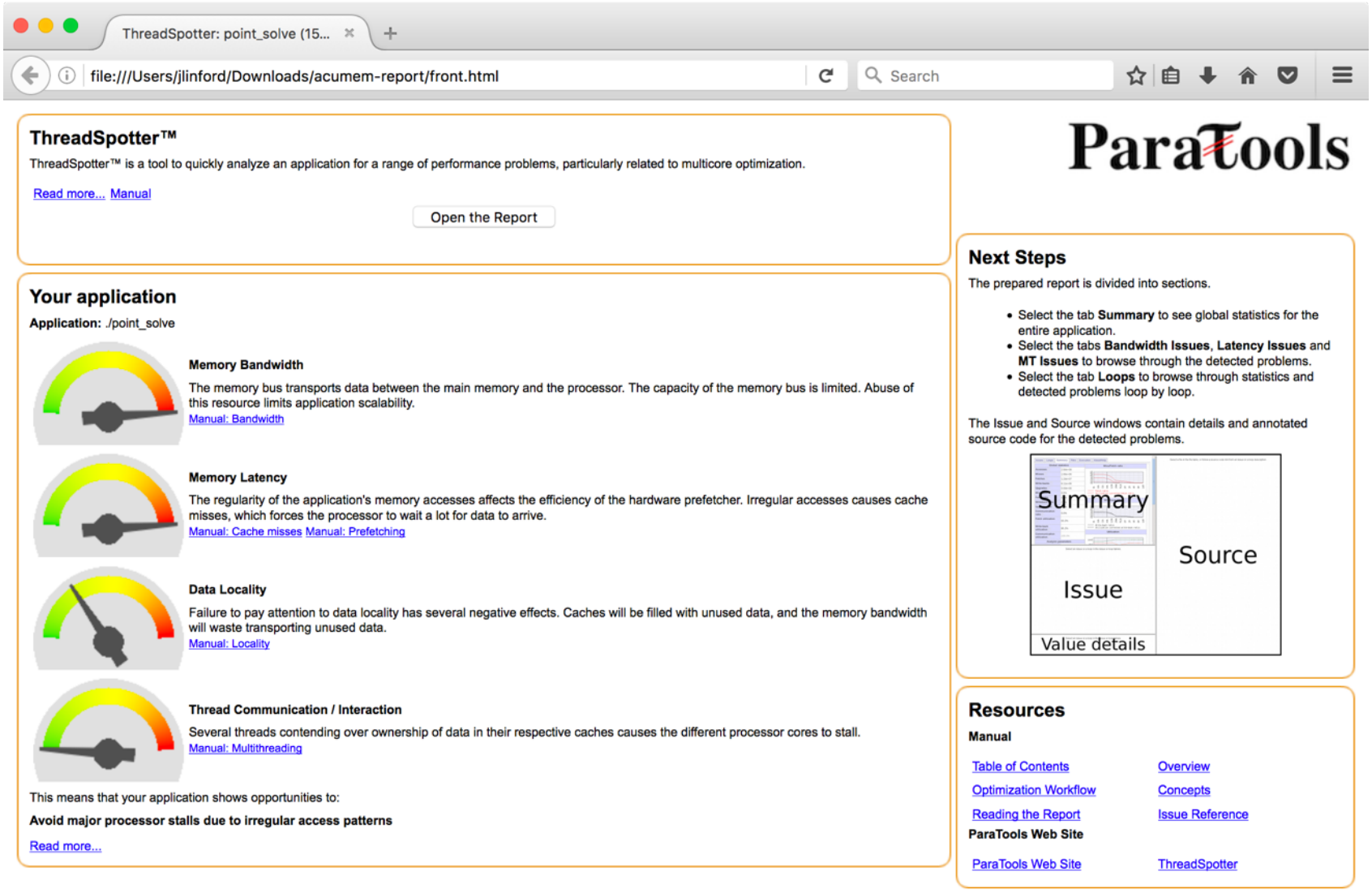
John C. Linford, Sameer Shende, et al.
{jlinford,sameer}@paratools.com

Data Movement Workshop
6 December 2016, ARL, Aberdeen, MD

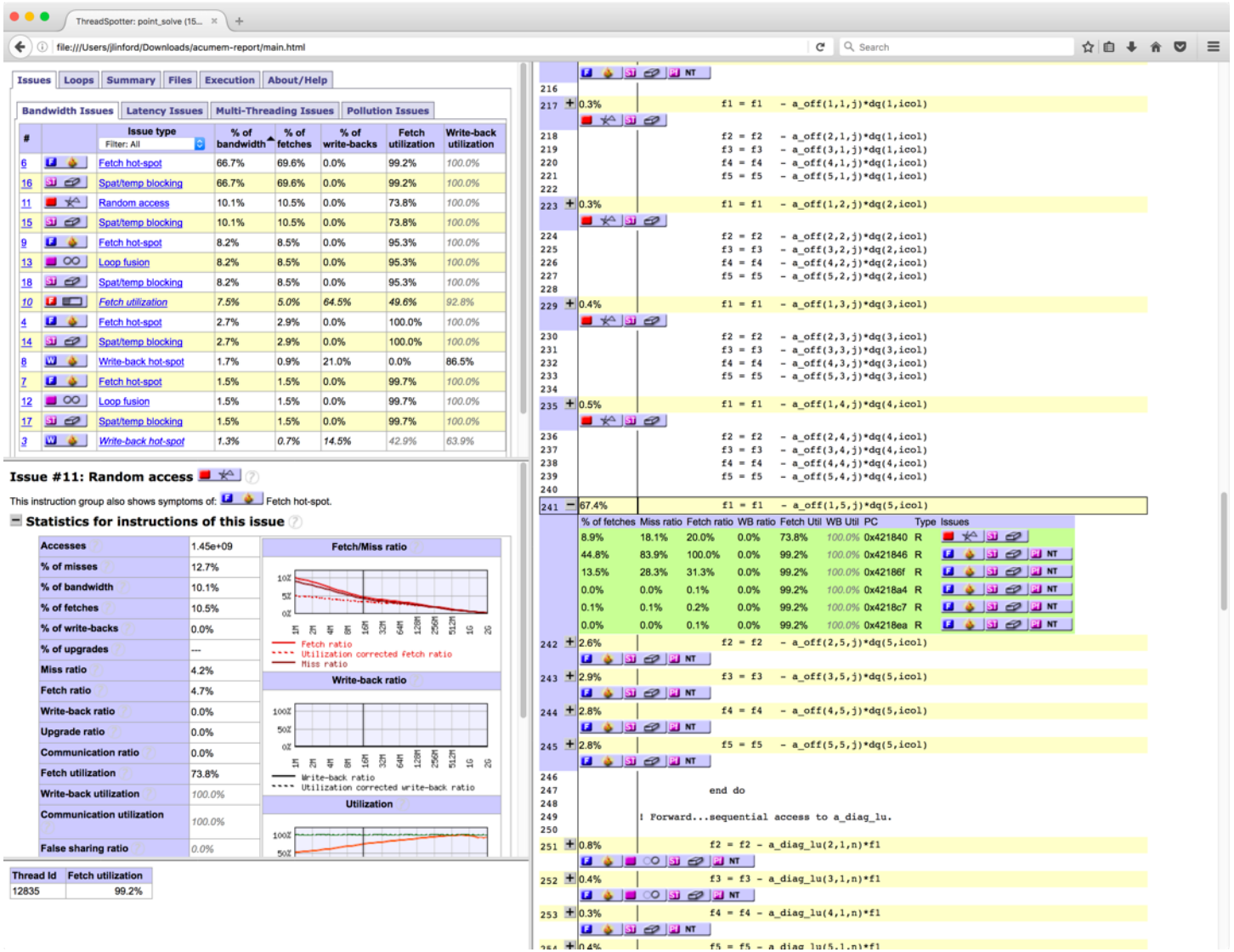
ParaTools ThreadSpotter

- A cache and memory optimization tool
  - Analyzes memory bandwidth and latency, data locality and thread communications
  - Identifies specific issues and pinpoints troublesome areas in source code
  - Provides guidance towards a resolution
  - Increase productivity for experts and non-experts

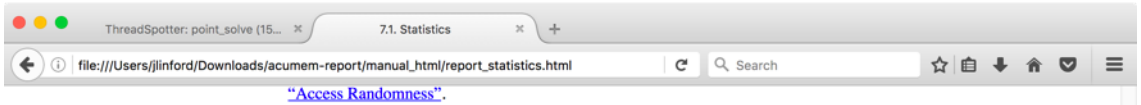
ThreadSpotter Report Front Page



ThreadSpotter Report



ThreadSpotter Online Help



"Access Randomness".

Worst instruction

Points out the instruction that causes causes the largest number of cache line fetches in this part of the program, and the source code line that generated it.

7.1.2. Reading the Diagrams

The report contains diagrams describing several cache size dependent application characteristics. The summary tab in the summary frame shows application global values, while the individual issues and loops show values related to their respective instruction groups.

The diagrams plot their values for different cache sizes, from an 8 kilobyte cache to a 16 megabyte cache in the following example. The cache size that the report focuses on is marked with a vertical black line, in this case at 64 kilobytes.

7.1.2.1. Fetch/Miss Ratio Diagram

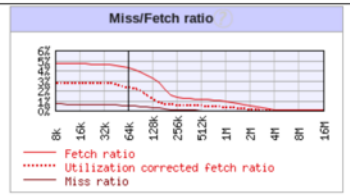


Figure 7.6. Fetch/Miss Ratio Diagram

- *Bright red line*

  Fetch ratio, the ratio of memory operations in the program, loop, issue or instruction group that, directly or indirectly through hardware prefetching, cause a data transfer between memory and cache. See Section 3.8, "Fetch Ratio".

- *Red dotted line*

  Utilization corrected fetch ratio. Fetch ratio if the fetch utilization was raised to 100%. See Section 4.8, "Utilization Corrected Fetch Ratio".

- *Dark red line*

  Miss ratio, the ratio of memory operations in the program that stall due to cache misses. The difference between the fetch ratio and the miss ratio is caused be software and hardware prefetching. See Section 3.4, "Cache Misses".

7.1.2.2. Write-Back Ratio Diagram

TAU Commander: The TAU User Interface

jlinford — jlinford@grover:~/DataMovement/miniapp1/openmp — ssh grover.nic.uorego...

```
[jlinford@grover ~/DataMovement/miniapp1/openmp $ tau --help
usage: tau [arguments] <subcommand> [options]

TAU Commander 1.0a [ www.taucommander.com ]

Positional Arguments:
  <subcommand>    See subcommand descriptions below.
  [options]       Options to be passed to <subcommand>.

Optional Arguments:
  -V, --version   Show program's version number and exit.
  -h, --help      Show this help message and exit.
  -l, --log       Record all actions to
                  '/storage/users/jlinford/.tau/debug_log'.
                    - default: False
  -q, --quiet     Suppress all output except error messages.
  -v, --verbose   Show debugging messages.

Configuration Subcommands:
  application     Create and manage application configurations.
  experiment      Create and manage experiments.
  measurement     Create and manage measurement configurations.
  project         Create and manage project configurations.
  target          Create and manage target configurations.
  trial           Create and manage experiment trials.

Subcommands:
  build           Instrument programs during compilation and/or linking.
  configure       Configure TAU Commander.
  dashboard       Show all project components.
  help            Show help for a command or suggest actions for a file.
  initialize      Initialize TAU Commander.
  select          Create a new experiment or select an existing experiment.

Shortcuts:
  tau <compiler>  Execute a compiler command
                  - Example: tau gcc *.c -o a.out
                  - Alias for 'tau build <compiler>'
  tau <program>   Gather data from a program
                  - Example: tau ./a.out
                  - Alias for 'tau trial create <program>'
  tau select      Select configuration objects to create a new experiment
                  - Alias for 'tau experiment create'
  tau show        Show data from the most recent trial
                  - Alias for 'tau trial show'

See 'tau help <subcommand>' for more information on <subcommand>.
jlinford@grover ~/DataMovement/miniapp1/openmp $
```

Questions TAU Can Answer

- **How much time** is spent in each application routine and outer *loops*? Within loops, what is the contribution of each *statement*?
- **How many instructions** are executed in these code regions? Floating point, Level 1 and 2 *data cache misses*, hits, branches taken, *vector instructions*?
- What is the **memory usage** of the code? When and where is memory allocated/de-allocated? Are there any *memory leaks*?
- What are the **I/O characteristics** of the code? What is the peak read and write *bandwidth* of individual calls, total volume?
- What is the **time spent waiting for collectives**?
- How does the application **scale**?

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

Example TAU Project

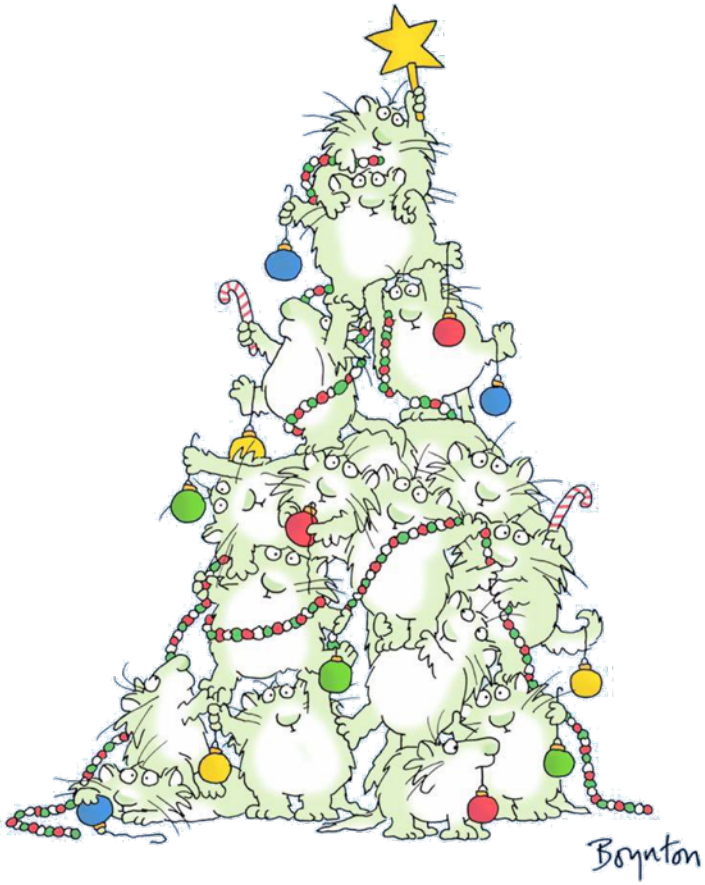
jlinford — jlinford@grover:~/DataMovement/miniapp1/baseline — ssh grover.nic.uoregon.edu — 124×49

```
[jlinford@grover ~/DataMovement/miniapp1/baseline $ tau dashboard

== Project Configuration (/storage/users/jlinford/DataMovement/miniapp1/.tau/project.json) ============================

+----------+---------+--------------+---------------------+---------------+
|   Name   | Targets | Applications |     Measurements    | # Experiments |
+==========+=========+==============+=====================+===============+
| miniapp1 | grover  |   miniapp1   | sample, profile, trace |       1       |
+----------+---------+--------------+---------------------+---------------+

== Targets in project 'miniapp1' ============================================================================================

+--------+---------+-----------+----------------+---------------+-----------------+
|  Name  | Host OS | Host Arch | Host Compilers | MPI Compilers | SHMEM Compilers |
+========+=========+===========+================+===============+=================+
| grover |  Linux  |  x86_64   |     Intel      |    System     |    OpenSHMEM    |
+--------+---------+-----------+----------------+---------------+-----------------+

== Applications in project 'miniapp1' =======================================================================================

+----------+--------+---------+-----+------+--------+--------+-----+
|   Name   | OpenMP | Pthreads | MPI | CUDA | OpenCL | SHMEM | MPC |
+==========+========+=========+=====+======+========+=======+=====+
| miniapp1 |   No   |    No   |  No |  No  |   No   |   No  |  No |
+----------+--------+---------+-----+------+--------+-------+-----+

== Measurements in project 'miniapp1' ======================================================================================

+---------+---------+-------+--------+-------------+--------------+--------+------+-----+-----+-------+
|  Name   | Profile | Trace | Sample | Source Inst. | Compiler Inst. | OpenMP | CUDA | I/O | MPI | SHMEM |
+=========+=========+=======+========+=============+================+========+======+=====+=====+=======+
| sample  |   tau   | none  |  Yes   |    never    |     never      |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+----------------+--------+------+-----+-----+-------+
| profile |   tau   | none  |   No   |  automatic  |    fallback    |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+----------------+--------+------+-----+-----+-------+
|  trace  |  none   | otf2  |   No   |  automatic  |    fallback    |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+----------------+--------+------+-----+-----+-------+

== Experiments in project 'miniapp1' ========================================================================================

+-----------------------+--------+-----------+---------+-------------+-------------+-----------------+
|         Name          | Trials | Data Size | Target  | Application | Measurement |   TAU Makefile  |
+=======================+========+===========+=========+=============+=============+=================+
| grover-miniapp1-sample |   0    |   0.0B    | grover  |  miniapp1   |   sample    | Makefile.tau-icpc |
+-----------------------+--------+-----------+---------+-------------+-------------+-----------------+

Selected Experiment: grover-miniapp1-sample

jlinford@grover ~/DataMovement/miniapp1/baseline $
```

TAU Commander

Tell the tool what you want, not how to get it

vs.



Getting Started with TAU Commander

- **tau init** [options | --help]
- **tau** <<your compiler>> foo.c
  - e.g. tau mpif90 foo.f90
- **tau** mpirun -np 8 ./a.out

Online Help

- **tau --help**
- **tau help** <something>

Tools Summary

- ParaTools ThreadSpotter
  - High-level cache and memory optimization tool.
  - Generates “plain English” reports with annotated source code.
- TAU Commander
  - Broad scope, powerful performance engineering tool.
  - Built on the TAU Performance System.
  - Easy way to get a wide variety of performance data.
    - PAPI, etc.
  - Installs and manages TAU and its dependencies.

Data Movement Workshop

FIVE POINT SOLVER MINIAPP

MINIAPP 1 Code Structure

```
do sweep = 1, n_sweeps
  do color = sweep_start, sweep_end, sweep_stride
    do ipass = 1, 2
      start = color_indices(1,color)
      end = color_boundary_end(color)

      do n = start, end
        istart = iam(n)
        iend   = iam(n+1)-1

        f(1:5) = (+/-)res(1:5)

        do j = istart, iend
          icol = jam(j)
          do i = 1, 5
            f(1:5) = f(1:5) - a_off(1:5,i,j)*dq(i,icol)
          end do
```

MINIAPP 1 Code Structure

```
do sweep = 1, n_sweeps
  do color = sweep_start, sweep_end, sweep_stride
    do ipass = 1, 2
      start = color_indices(1,color)
      end = color_boundary_end(color)

      do n = start, end
        istart = iam(n)
        iend   = iam(n+1)-1

        f(1:5) = (+/-)res(1:5)

        do j = istart, iend
          icol = jam(j)
          do i = 1, 5
            f(1:5) = f(1:5) - a_off(1:5,i,j)*dq(i,icol)
          end do
```

Unknown loop trip count

Low FLOP vector(5) kernel

Indirect index

MINIAPP 1 Kernel Unrolled

```
do j = istart,iend
  icol = jam(j)
  f1 = f1   - a_off(1,1,j)*dq(1,icol)
  f2 = f2   - a_off(2,1,j)*dq(1,icol)
  f3 = f3   - a_off(3,1,j)*dq(1,icol)
  f4 = f4   - a_off(4,1,j)*dq(1,icol)
  f5 = f5   - a_off(5,1,j)*dq(1,icol)
  f1 = f1   - a_off(1,2,j)*dq(2,icol)
  f2 = f2   - a_off(2,2,j)*dq(2,icol)
  f3 = f3   - a_off(3,2,j)*dq(2,icol)
  f4 = f4   - a_off(4,2,j)*dq(2,icol)
  f5 = f5   - a_off(5,2,j)*dq(2,icol)
  f1 = f1   - a_off(1,3,j)*dq(3,icol)
  f2 = f2   - a_off(2,3,j)*dq(3,icol)
  f3 = f3   - a_off(3,3,j)*dq(3,icol)
  f4 = f4   - a_off(4,3,j)*dq(3,icol)
  f5 = f5   - a_off(5,3,j)*dq(3,icol)
  f1 = f1   - a_off(1,4,j)*dq(4,icol)
  f2 = f2   - a_off(2,4,j)*dq(4,icol)
  f3 = f3   - a_off(3,4,j)*dq(4,icol)
  f4 = f4   - a_off(4,4,j)*dq(4,icol)
  f5 = f5   - a_off(5,4,j)*dq(4,icol)
  f1 = f1   - a_off(1,5,j)*dq(5,icol)
  f2 = f2   - a_off(2,5,j)*dq(5,icol)
  f3 = f3   - a_off(3,5,j)*dq(5,icol)
  f4 = f4   - a_off(4,5,j)*dq(5,icol)
  f5 = f5   - a_off(5,5,j)*dq(5,icol)
end do
```

```
do j = istart, iend
  icol = jam(j)
  do i = 1, 5
    f(1:5) = f(1:5) - a_off(1:5,i,j)*dq(i,icol)
  end do
```

- 56 Loads
- 26 Stores
- 50 FP-ops
  - Fused to 25
- ~0.17 FP-ops / byte
  - Fused: 0.083 FP-ops / byte

MINIAPP 1 BOTE Analysis

- 56 loads, 26 stores, 25 FP-ops ➔ Memory bound
- Vector(5) is a pain:
  - Want Vector(4) for SSE, Vector(8) for AVX2, Vector(16) for AVX512
  - Padding to improve vectorization hurts cache line utilization
- Indirect access, unknown loop trip
  - Calculated load/store in innermost loop
  - Dynamic dispatch for vectorized loop
- Already have coarse grain MPI parallelization
  - OpenMP parallelization in `n=start,end` straightforward, but will it help?

```
do j = istart, iend
  icol = jam(j)
  do i = 1, 5
    f(1:5) = f(1:5) - a_off(1:5,i,j)*dq(i,icol)
  end do
```

MINIAPP1 Questions

Question	Primary Tool	Secondary Tool
Runtime hot spots?	TAU	system_clock
Will OpenMP help?	TAU	ThreadSpotter
Cache utilization?	ThreadSpotter	TAU
Can MCDRAM help?	ThreadSpotter	TAU
Can we shuffle a_off for better performance?	ThreadSpotter	TAU
Can we vectorize? How hard should we try?	TAU	ThreadSpotter

Reminder: The compiler optimization report is not a todo list.

Data Movement Workshop

THREADSPOTTER ANALYSIS

ParaTools ThreadSpotter

```
$ sample_ts -r ./point_solve

$ report_ts -i sample.smp

$ view-static_ts -i report.tsr

$ tar cvzf acumem-report.tgz acumem-report.html acumem-report

$ scp acumem-report.tgz  <somewhere with a browser>
```

Note: When running with MPI use tau_exec:

```
$ mpirun -np 256 tau_exec -ptts ./point_solve
```

Report Front Page

ThreadSpotter: point_solve (15...

file:///Users/jlinford/Downloads/acumem-report/front.html

Search

ThreadSpotter™

ThreadSpotter™ is a tool to quickly analyze an application for a range of performance problems, particularly related to multicore optimization.

Read more... Manual

Open the Report

ParaTools

Your application

Application: ./point_solve

Memory Bandwidth

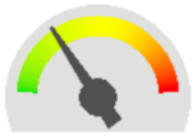
The memory bus transports data between the main memory and the processor. The capacity of the memory bus is limited. Abuse of this resource limits application scalability.

Manual: Bandwidth

Memory Latency

The regularity of the application's memory accesses affects the efficiency of the hardware prefetcher. Irregular accesses causes cache misses, which forces the processor to wait a lot for data to arrive.

Manual: Cache misses Manual: Prefetching



Data Locality

Failure to pay attention to data locality has several negative effects. Caches will be filled with unused data, and the memory bandwidth will waste transporting unused data.

Manual: Locality

Thread Communication / Interaction

Several threads contending over ownership of data in their respective caches causes the different processor cores to stall.

Manual: Multithreading

This means that your application shows opportunities to:

Avoid major processor stalls due to irregular access patterns

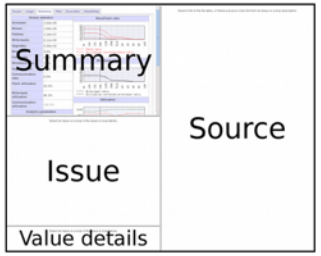
Read more...

Next Steps

The prepared report is divided into sections.

- Select the tab **Summary** to see global statistics for the entire application.
- Select the tabs **Bandwidth Issues**, **Latency Issues** and **MT Issues** to browse through the detected problems.
- Select the tab **Loops** to browse through statistics and detected problems loop by loop.

The Issue and Source windows contain details and annotated source code for the detected problems.



Resources

Manual

Table of Contents | Overview

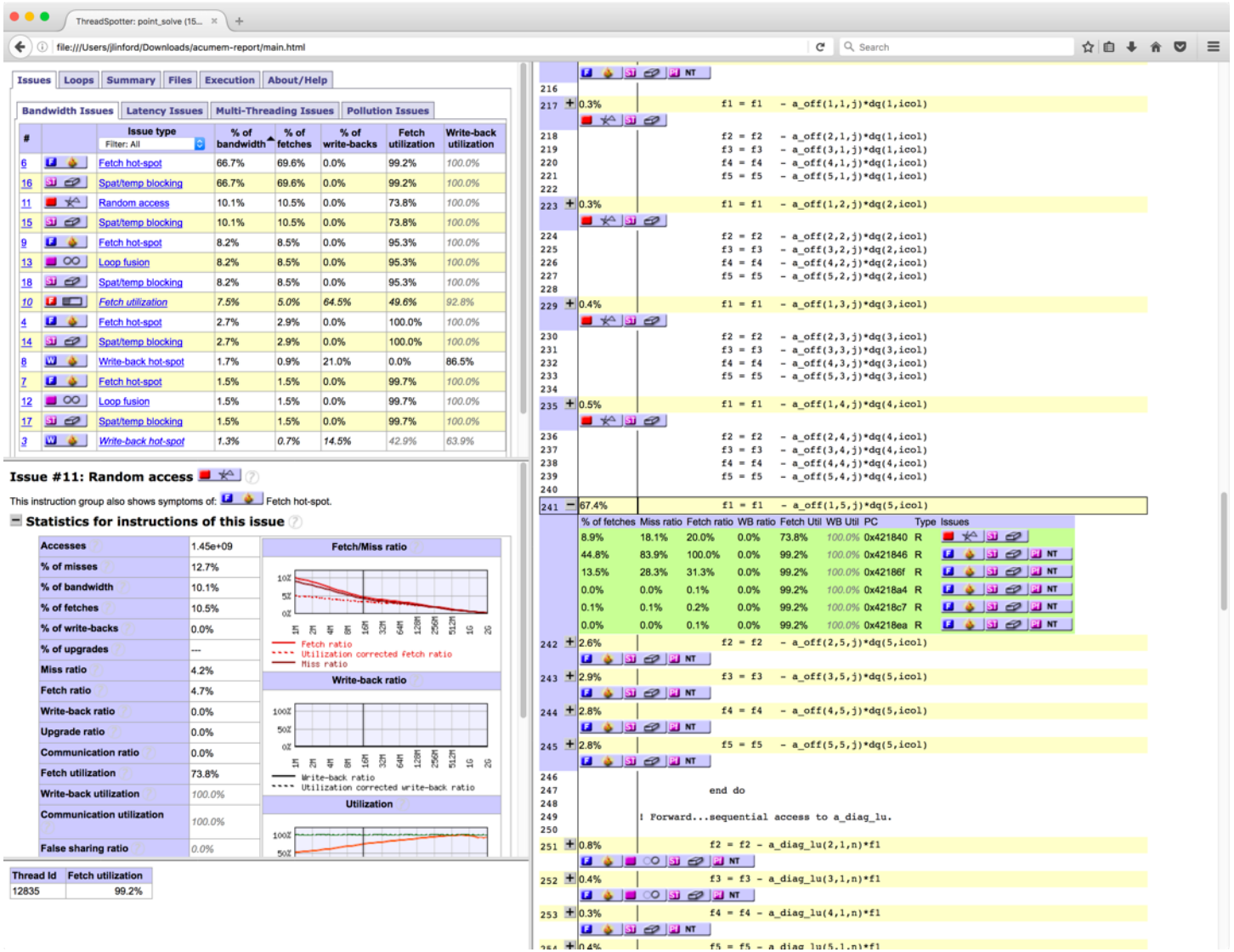
Optimization Workflow | Concepts

Reading the Report | Issue Reference

ParaTools Web Site

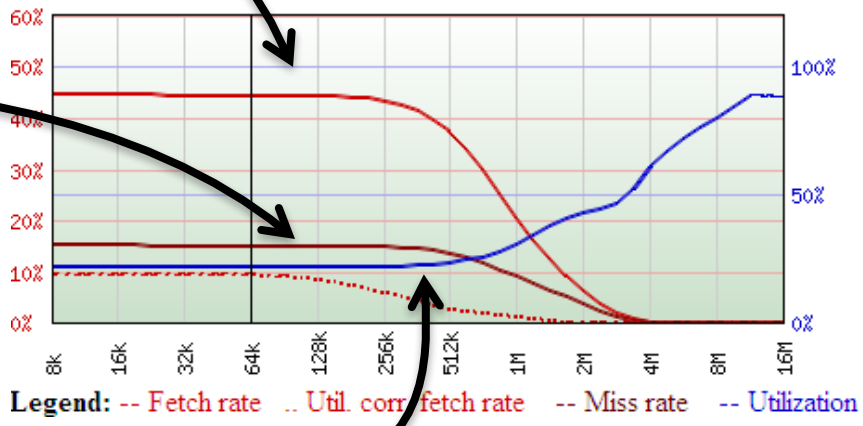
ParaTools Web Site | ThreadSpotter

Fetch Hotspots



Metrics as a function of cache size

- Fetch ratio
  - Memory operations that cause a data transfer to/from RAM
- Miss ratio
  - Memory operations that stall due to cache misses.
- Fetch utilization
  - Fraction of the data loaded into the cache that are actually used

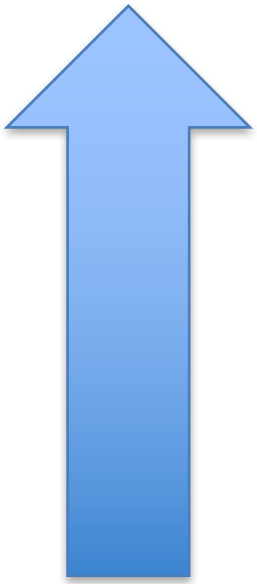


Data Movement Workshop

TAU COMMANDER ANALYSIS

Initialize TAU Project

```
$ cd ~/DataMovement/miniapp1

$ which tau
/usr/local/packages/taucmdr-unstable/bin/tau

$ tau initialize
```



- Creates a new project configuration using defaults
- Project files exist in a directory named “.tau”
- Like git, all directories below the directory containing the “.tau” directory can access the project
  - E.g. `tau dashboard` works in miniapp1/baseline

MINIAPP1 TAU Project

jlinford — jlinford@grover:~/DataMovement/miniapp1 — ssh grover.nic.uoregon.edu — 142×64

```
jlinford@grover ~ $ cd ~/DataMovement/miniapp1
jlinford@grover ~/DataMovement/miniapp1 $ which tau
/usr/local/packages/taucmdr-unstable/bin/tau
jlinford@grover ~/DataMovement/miniapp1 $ tau initialize
[TAU] System MPI C++ compiler '/usr/local/packages/intel/impi/5.0/intel64/bin/mpicxx' wraps '/usr/bin/g++'
[TAU] System MPI C compiler '/usr/local/packages/intel/impi/5.0/intel64/bin/mpicc' wraps '/usr/bin/gcc'
[TAU] System MPI Fortran compiler '/usr/local/packages/intel/impi/5.0/intel64/bin/mpif90' wraps '/usr/bin/gfortran'
[TAU] Created a new project named 'miniapp1'.
[TAU] Added application 'miniapp1' to project configuration 'miniapp1'.
[TAU] Added target 'grover' to project configuration 'miniapp1'.
[TAU] Added measurement 'sample' to project configuration 'miniapp1'.
[TAU] Added measurement 'profile' to project configuration 'miniapp1'.
[TAU] Added measurement 'trace' to project configuration 'miniapp1'.
[TAU] Created a new experiment named 'grover-miniapp1-sample'.
[TAU] Selected experiment 'grover-miniapp1-sample'.
[TAU]

== Project Configuration (/storage/users/jlinford/DataMovement/miniapp1/.tau/project.json) ==========

+----------+---------+--------------+-----------------------+---------------+
|   Name   | Targets | Applications |      Measurements     | # Experiments |
+==========+=========+==============+=======================+===============+
| miniapp1 | grover  |   miniapp1   | sample, profile, trace |       1       |
+----------+---------+--------------+-----------------------+---------------+

== Targets in project 'miniapp1' ==========

+---------+---------+-----------+----------------+---------------+-----------------+
|  Name   | Host OS | Host Arch | Host Compilers | MPI Compilers | SHMEM Compilers |
+=========+=========+===========+================+===============+=================+
| grover  |  Linux  |  x86_64   |     Intel      |    System     |    OpenSHMEM    |
+---------+---------+-----------+----------------+---------------+-----------------+

== Applications in project 'miniapp1' ==========

+----------+--------+----------+-----+------+--------+-------+-----+
|   Name   | OpenMP | Pthreads | MPI | CUDA | OpenCL | SHMEM | MPC |
+==========+========+==========+=====+======+========+=======+=====+
| miniapp1 |   No   |    No    | No  |  No  |   No   |  No   | No  |
+----------+--------+----------+-----+------+--------+-------+-----+

== Measurements in project 'miniapp1' ==========

+---------+---------+-------+--------+-------------+---------------+--------+------+-----+-----+-------+
|  Name   | Profile | Trace | Sample | Source Inst. | Compiler Inst. | OpenMP | CUDA | I/O | MPI | SHMEM |
+=========+=========+=======+========+=============+===============+========+======+=====+=====+=======+
| sample  |   tau   | none  |  Yes   |    never    |     never     |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+---------------+--------+------+-----+-----+-------+
| profile |   tau   | none  |   No   |  automatic  |   fallback    |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+---------------+--------+------+-----+-----+-------+
|  trace  |  none   | otf2  |   No   |  automatic  |   fallback    |  none  |  No  | No  | No  |  No   |
+---------+---------+-------+--------+-------------+---------------+--------+------+-----+-----+-------+

== Experiments in project 'miniapp1' ==========

+----------------------+--------+-----------+--------+-------------+-------------+----------------+
|         Name         | Trials | Data Size | Target | Application | Measurement |  TAU Makefile  |
+======================+========+===========+========+=============+=============+================+
| grover-miniapp1-sample |   0    |   0.0B    | grover |  miniapp1   |   sample    | Makefile.tau-icpc |
+----------------------+--------+-----------+--------+-------------+-------------+----------------+

Selected Experiment: grover-miniapp1-sample

jlinford@grover ~/DataMovement/miniapp1 $
```

Use `tau` to compile

```
$ cd ~/DataMovement/miniapp1/baseline
$ vi Makefile

  1 # Fortran Compiler
  2 FC = tau ifort
  3 #FC = mpif90
  4
...
 16
 17 # Program name(s)
 18 PROGRAMS = point_solve
 19
 20 .PHONY: all clean run
 21
 22 all: $(PROGRAMS)
 23
 24 run: all
 25   tau ./point_solve
 26
```

Prepend `tau` command

Prepend `tau` command

Use `tau` to compile

```
jlinford — jlinford@grover:~/DataMovement/miniapp1/baseline — ssh grover.nic.uoregon.edu — 146×49
[jlinford@grover ~/DataMovement/miniapp1/baseline $ make
./Depend.pl point_solve.f90 . \
-X ieee_arithmetic \
> point_solve.d
tau ifort -convert big_endian -O3 -ip -align array64byte -fno-alias -g -traceback -qopt-report=5 -std03 -xMIC-AVX512 -qopenmp  -c point_solve.f90
[TAU] TAU_MAKEFILE=/storage/packages/taucmdr-unstable/.system/tau/f942660b60ec2cecc0deff88257e387d/x86_64/lib/Makefile.tau-icpc
[TAU] TAU_OPTIONS=-optRevert -optNoCompInst
[TAU] /packages/intel/17/linux/bin/intel64/ifort -g -convert big_endian -O3 -ip -align array64byte -fno-alias -g -traceback -qopt-report=5 -std03
[TAU]     -xMIC-AVX512 -qopenmp -c point_solve.f90
ifort: remark #10397: optimization reports are generated in *.optrpt files in the output location
tau ifort -convert big_endian -O3 -ip -align array64byte -fno-alias -g -traceback -qopt-report=5 -std03 -xMIC-AVX512 -qopenmp  -o point_solve kind
defs.o sort.o interp_defs.o allocations.o point_solve.o lmpi_app.o lmpi.o system_extensions.o
[TAU] TAU_MAKEFILE=/storage/packages/taucmdr-unstable/.system/tau/f942660b60ec2cecc0deff88257e387d/x86_64/lib/Makefile.tau-icpc
[TAU] TAU_OPTIONS=-optRevert -optNoCompInst
[TAU] /packages/intel/17/linux/bin/intel64/ifort -g -convert big_endian -O3 -ip -align array64byte -fno-alias -g -traceback -qopt-report=5 -std03
[TAU]     -xMIC-AVX512 -qopenmp -o point_solve kinddefs.o sort.o interp_defs.o allocations.o point_solve.o lmpi_app.o lmpi.o system_extensions.o
ifort: remark #10397: optimization reports are generated in *.optrpt files in the output location
jlinford@grover ~/DataMovement/miniapp1/baseline $
```

- TAU Commander constructs a new compilation command line to match the selected experiment.
  - May replace compiler commands with TAU's compiler wrapper scripts.
  - May set environment variables, parse configuration files, etc.
  - If no changes are required then nothing is changed.

Use `tau` to run

```
jlinford — jlinford@grover:~/DataMovement/miniapp1/baseline — ssh grover.nic.uoregon.edu — 147×44
[jlinford@grover ~/DataMovement/miniapp1/baseline $ make run
tau ./point_solve
[TAU]
[TAU] == BEGIN Experiment at 2016-12-05 17:03:04.133533 ==========
[TAU]
[TAU] TAU_CALLPATH=1
[TAU] TAU_CALLPATH_DEPTH=100
[TAU] TAU_COMM_MATRIX=0
[TAU] TAU_METRICS=TIME
[TAU] TAU_PROFILE=1
[TAU] TAU_SAMPLING=1
[TAU] TAU_THROTTLE=1
[TAU] TAU_THROTTLE_NUMCALLS=100000
[TAU] TAU_THROTTLE_PERCALL=10
[TAU] TAU_TRACE=0
[TAU] TAU_TRACK_HEAP=0
[TAU] TAU_VERBOSE=0
[TAU] tau_exec -T serial,icpc -ebs ./point_solve
 Loading data...
0 Number of block 5x5 equations in data file: 1123718
 Done loading data...
 Solving Ax=b...
 Sweep seconds on master =    1.366700
 Sweep seconds on master =    1.351100
 Sweep seconds on master =    1.355900
 Sweep seconds on master =    1.351700
 Sweep seconds on master =    1.342100
 Sweep seconds on master =    1.340800
 Sweep seconds on master =    1.341700
 Sweep seconds on master =    1.342700
 Sweep seconds on master =    1.346500
 Sweep seconds on master =    1.344900
 Sweep seconds on master =    1.337900
 Sweep seconds on master =    1.337500
 Sweep seconds on master =    1.337400
 Sweep seconds on master =    1.337200
 Sweep seconds on master =    1.338000
 Total seconds taken on master =    20.17360
 Test passed.
[TAU]
[TAU] == END Experiment at 2016-12-05 17:03:27.955025 ==========
[TAU]
[TAU] Trial 1 produced 1 profile files.
jlinford@grover ~/DataMovement/miniapp1/baseline $
```

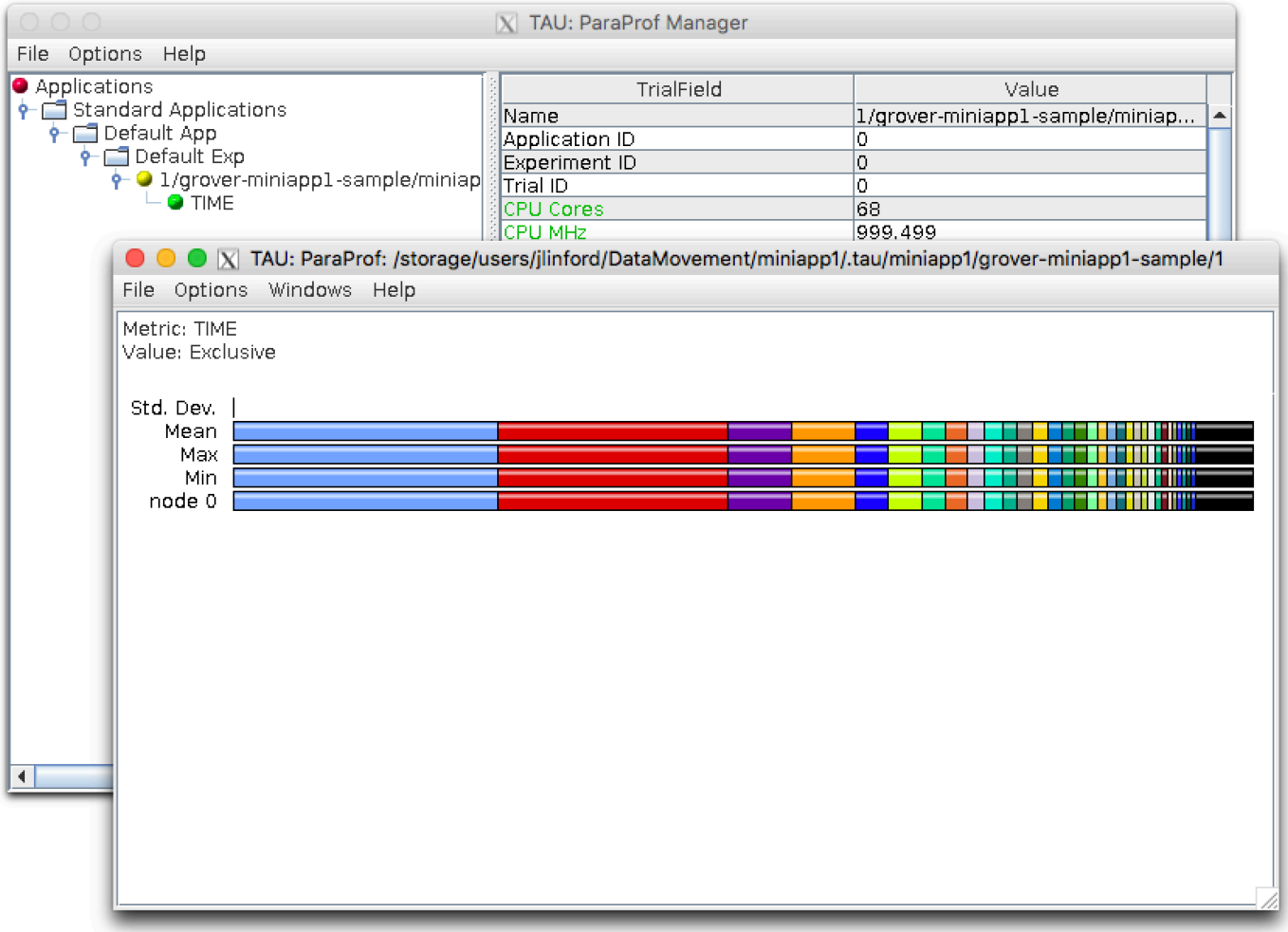
Tracks experiment metadata

Sets appropriate environment variables

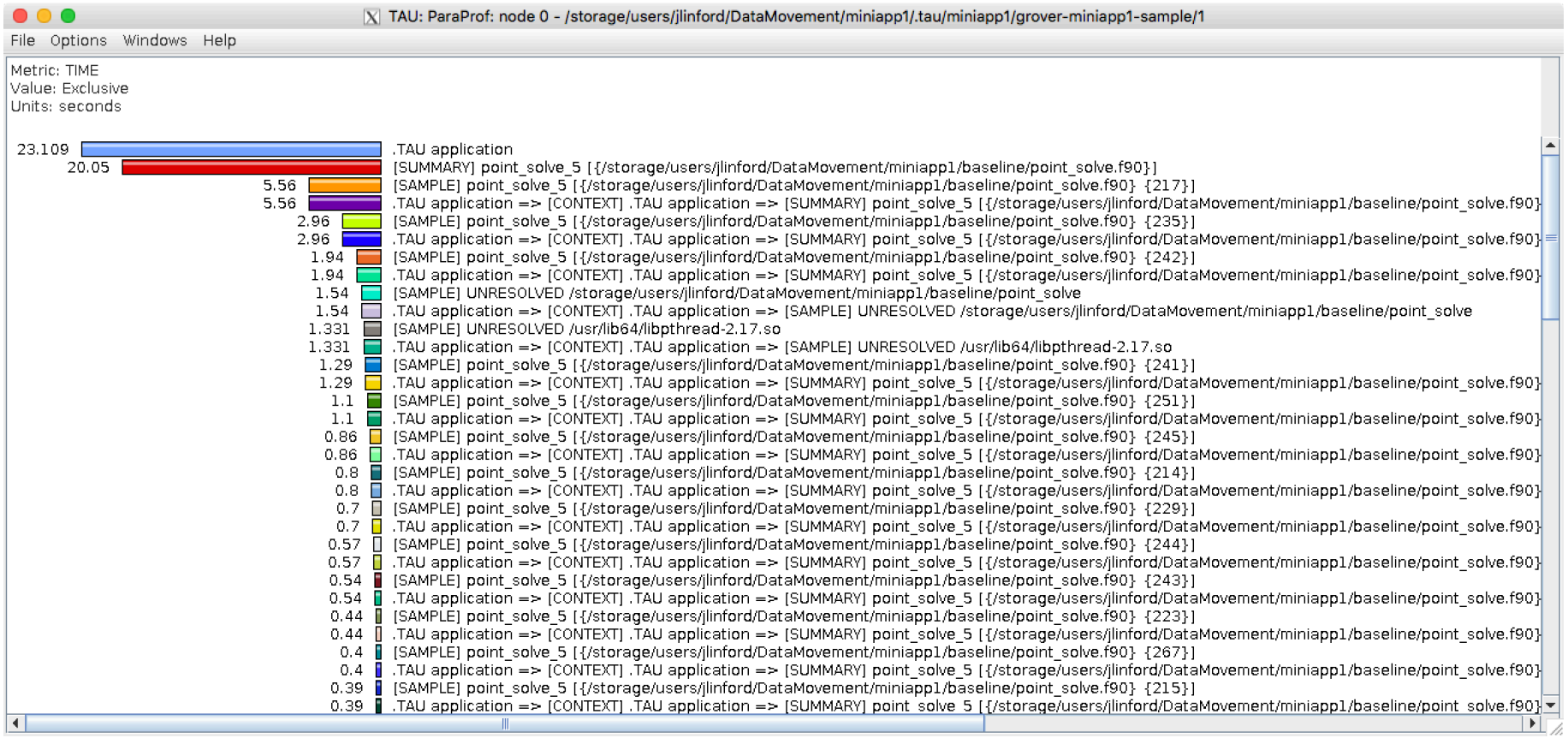
Stores generated data in a performance database.

View profile

```
$ tau show
```

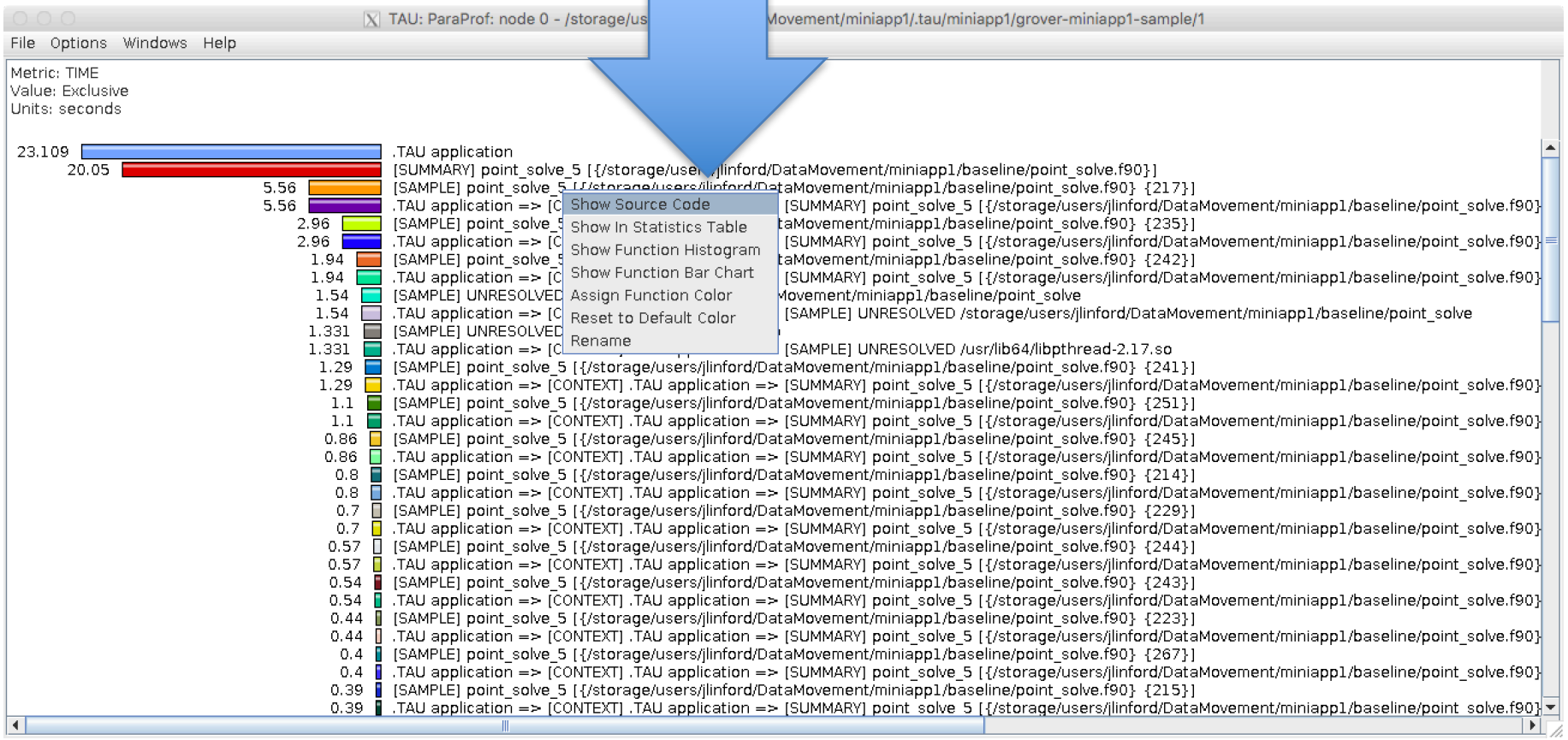


Node 0 Exclusive Time Profile

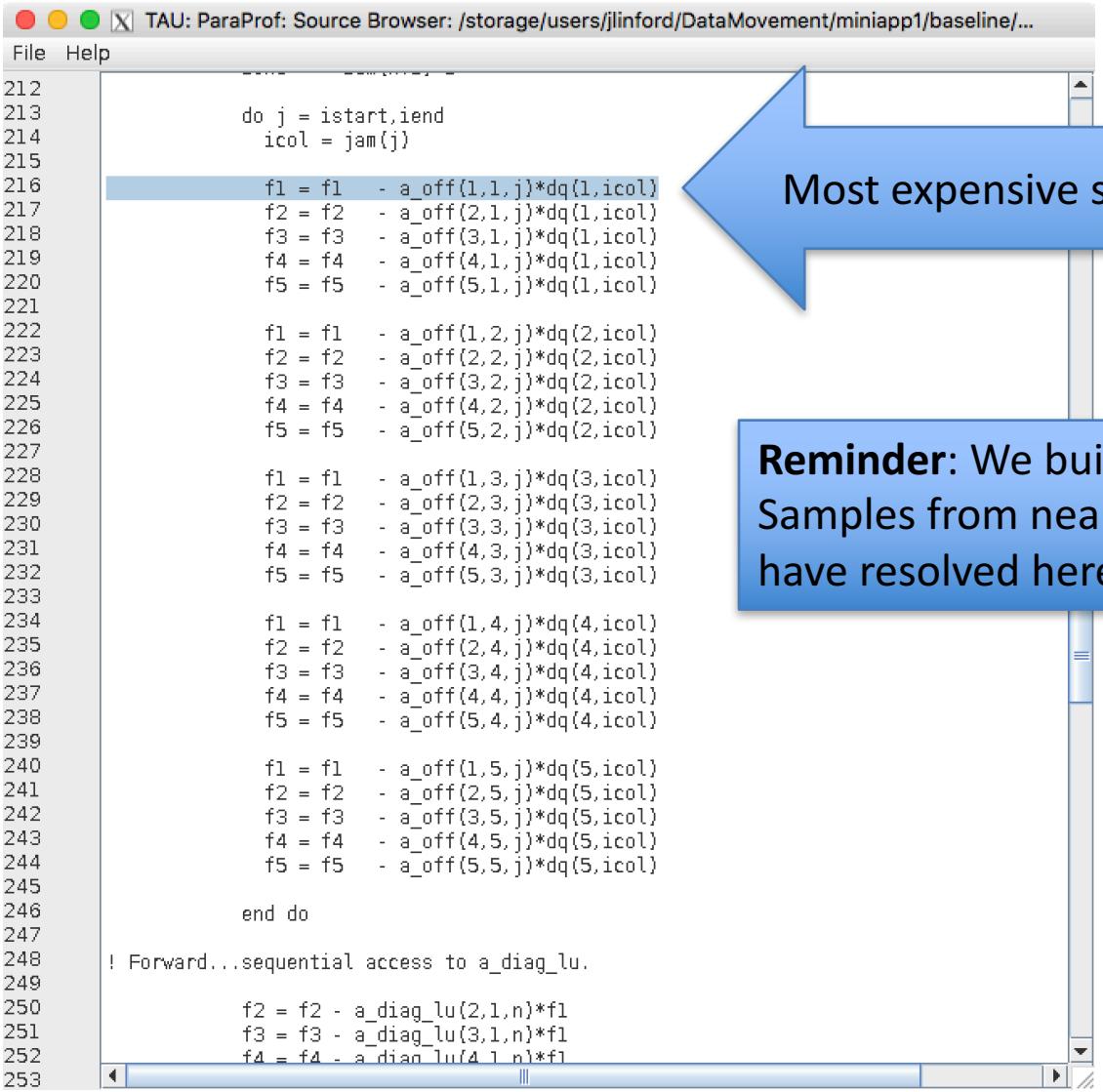


View Source Code

Right-click

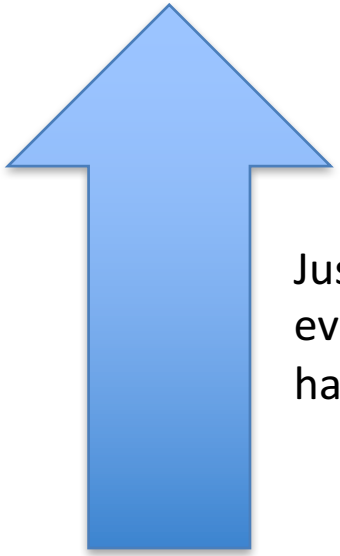


View Source Code



How to find most expensive line of code

1. **tau** initialize
2. **tau** ifort *.f90 -o foo
3. **tau** ./foo
4. **tau** show



Just put `**tau**` in front of everything and see what happens.

- This works on any supported system, even if TAU is not installed or has not been configured appropriately.
- TAU and all its dependencies will be downloaded and installed if required.

papi_avail on U. Oregon KNL (Grover)

```
$ papi_avail | grep Yes
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes   Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_TLB_DM  0x80000014  Yes   No   Data translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  Yes   No   Level 1 load misses
PAPI_L2_LDM  0x80000019  Yes   No   Level 2 load misses
PAPI_STL_ICY 0x80000025  Yes   No   Cycles with no instruction issue
PAPI_BR_UCN  0x8000002a  Yes   Yes  Unconditional branch instructions
PAPI_BR_CN   0x8000002b  Yes   No   Conditional branch instructions
PAPI_BR_TKN  0x8000002c  Yes   No   Conditional branch instructions taken
PAPI_BR_NTK  0x8000002d  Yes   Yes  Conditional branch instructions not taken
PAPI_BR_MSP  0x8000002e  Yes   No   Conditional branch instructions mispredicted
PAPI_TOT_INS 0x80000032  Yes   No   Instructions completed
PAPI_LD_INS  0x80000035  Yes   No   Load instructions
PAPI_SR_INS  0x80000036  Yes   No   Store instructions
PAPI_BR_INS  0x80000037  Yes   No   Branch instructions
PAPI_RES_STL 0x80000039  Yes   No   Cycles stalled on any resource
PAPI_TOT_CYC 0x8000003b  Yes   No   Total cycles
PAPI_LST_INS 0x8000003c  Yes   Yes  Load/store instructions completed
PAPI_L1_DCA  0x80000040  Yes   Yes  Level 1 data cache accesses
PAPI_L1_ICH  0x80000049  Yes   No   Level 1 instruction cache hits
PAPI_L1_ICA  0x8000004c  Yes   No   Level 1 instruction cache accesses
PAPI_L2_TCH  0x80000056  Yes   Yes  Level 2 total cache hits
PAPI_L2_TCA  0x80000059  Yes   No   Level 2 total cache accesses
PAPI_REF_CYC 0x8000006b  Yes   No   Reference clock cycles
```

Measuring PAPI Counters

```
$ tau measurement copy sample sample.papi \
      --metrics TIME PAPI_L1_DCM PAPI_L2_TCM
```

Space-separated list of metrics

```
$ tau select sample.papi
[TAU] Selected experiment 'grover-miniapp1-sample.papi'.
[TAU] Application rebuild required:
[TAU]   - metrics changed from [TIME] to [TIME, PAPI_L1_DCM, PAPI_L2_TCM]
```

TAU Commander advises when application should be rebuilt.

PAPI Metric Compatibility Checks

jlinford — jlinford@grover:~/DataMovement/miniapp1/baseline — ssh grover.nic.uoregon.edu — 148×30

```
jlinford@grover ~/DataMovement/miniapp1/baseline $ tau meas copy sample sample.papi --metrics TIME PAPI_L1_DCM PAPI_L1_DCA PAPI_L2_TCM PAPI_L2_TCA
[TAU] Added measurement 'sample.papi' to project configuration 'miniapp1'.
jlinford@grover ~/DataMovement/miniapp1/baseline $ tau sel sample.papi
[TAU] Created a new experiment named 'grover-miniapp1-sample.papi'.
[TAU] XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[TAU]
[TAU] CRITICAL
[TAU]
[TAU] PAPI metrics [PAPI_L1_DCM, PAPI_L1_DCA, PAPI_L2_TCM, PAPI_L2_TCA] are not compatible on this target.
[TAU]
[TAU] Hints:
[TAU]   * Use papi_avail to check metric availability.
[TAU]   * Spread the desired metrics over multiple measurements.
[TAU]   * Choose fewer metrics.
[TAU]
[TAU] TAU cannot proceed with the given inputs.
[TAU] Please check the selected configuration for errors or contact <support@paratools.com> for assistance.
[TAU]
[TAU] XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
jlinford@grover ~/DataMovement/miniapp1/baseline $
```

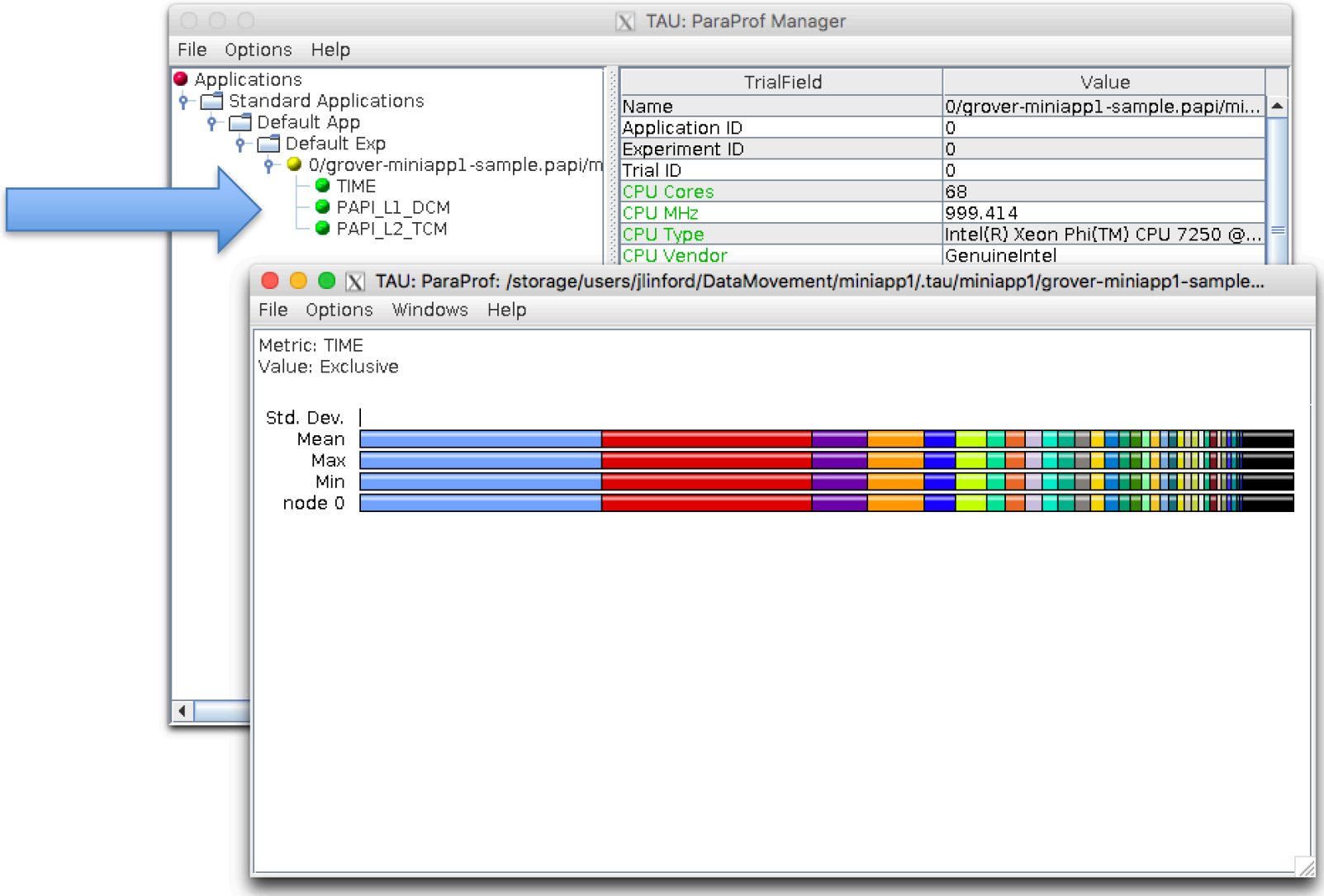
Uses papi_event_chooser to check metric compatibility.

Run exactly as before: `tau ./point_solve`

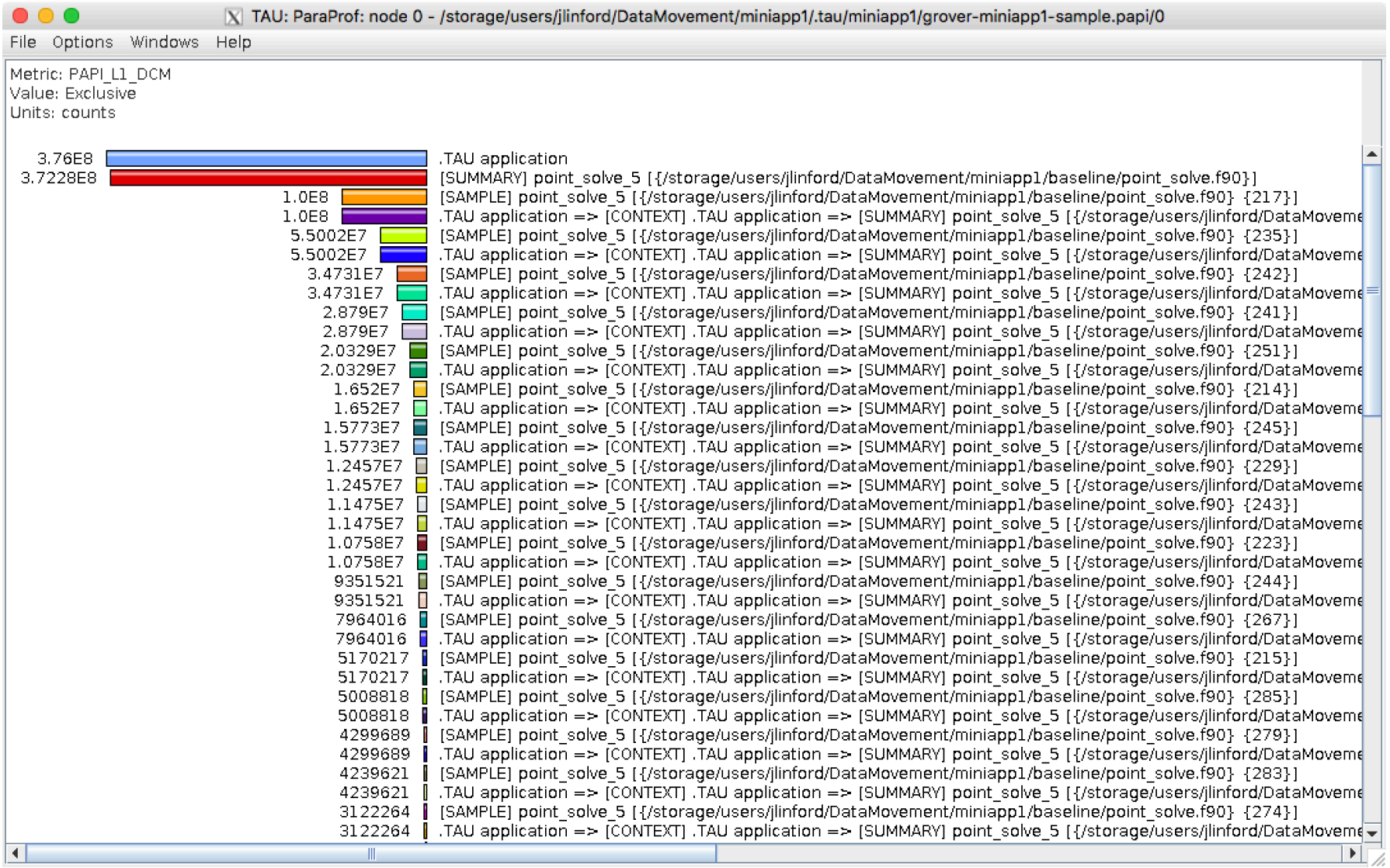
```
$ make run
tau ./point_solve
[TAU]
[TAU] == BEGIN Experiment at 2016-12-05 18:59:55.777171 ============================================================
[TAU]
[TAU] TAU_CALLPATH=1
[TAU] TAU_CALLPATH_DEPTH=100
[TAU] TAU_COMM_MATRIX=0
[TAU] TAU_METRICS=TIME,PAPI_L1_DCM,PAPI_L2_TCM
[TAU] TAU_PROFILE=1
[TAU] TAU_SAMPLING=1
[TAU] TAU_THROTTLE=1
[TAU] TAU_THROTTLE_NUMCALLS=100000
[TAU] TAU_THROTTLE_PERCALL=10
[TAU] TAU_TRACE=0
[TAU] TAU_TRACK_HEAP=0
[TAU] TAU_VERBOSE=0
[TAU] tau_exec -T serial,papi,icpc -ebs ./point_solve
 Loading data...
0 Number of block 5x5 equations in data file: 1123718
 Done loading data...
 Solving Ax=b...
 Sweep seconds on master =    1.369700
 Sweep seconds on master =    1.356500
 Sweep seconds on master =    1.358500
 Sweep seconds on master =    1.353100
 Sweep seconds on master =    1.347900
 Sweep seconds on master =    1.348200
 Sweep seconds on master =    1.348400
 Sweep seconds on master =    1.345000
 Sweep seconds on master =    1.350200
 Sweep seconds on master =    1.344700
 Sweep seconds on master =    1.344700
 Sweep seconds on master =    1.344000
 Sweep seconds on master =    1.343200
 Sweep seconds on master =    1.343400
 Sweep seconds on master =    1.343500
 Total seconds taken on master =    20.24260
 Test passed.
[TAU]
[TAU] == END Experiment at 2016-12-05 19:00:19.648510 ==============================================================
[TAU]
[TAU] Trial 0 produced 3 profile files.
```

Trial produced 3 profile files, one for each metric.

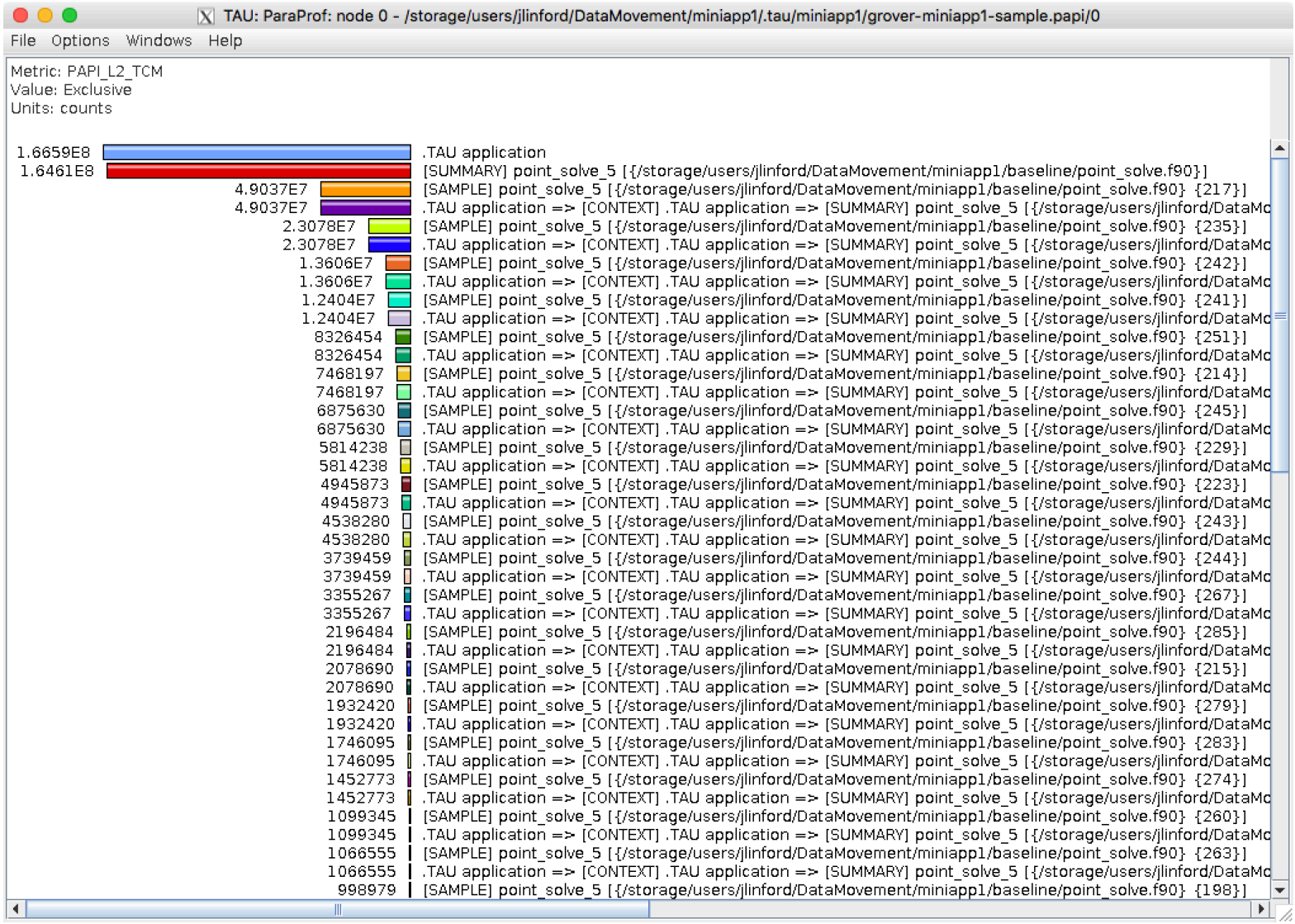
View profiles: `tau show`



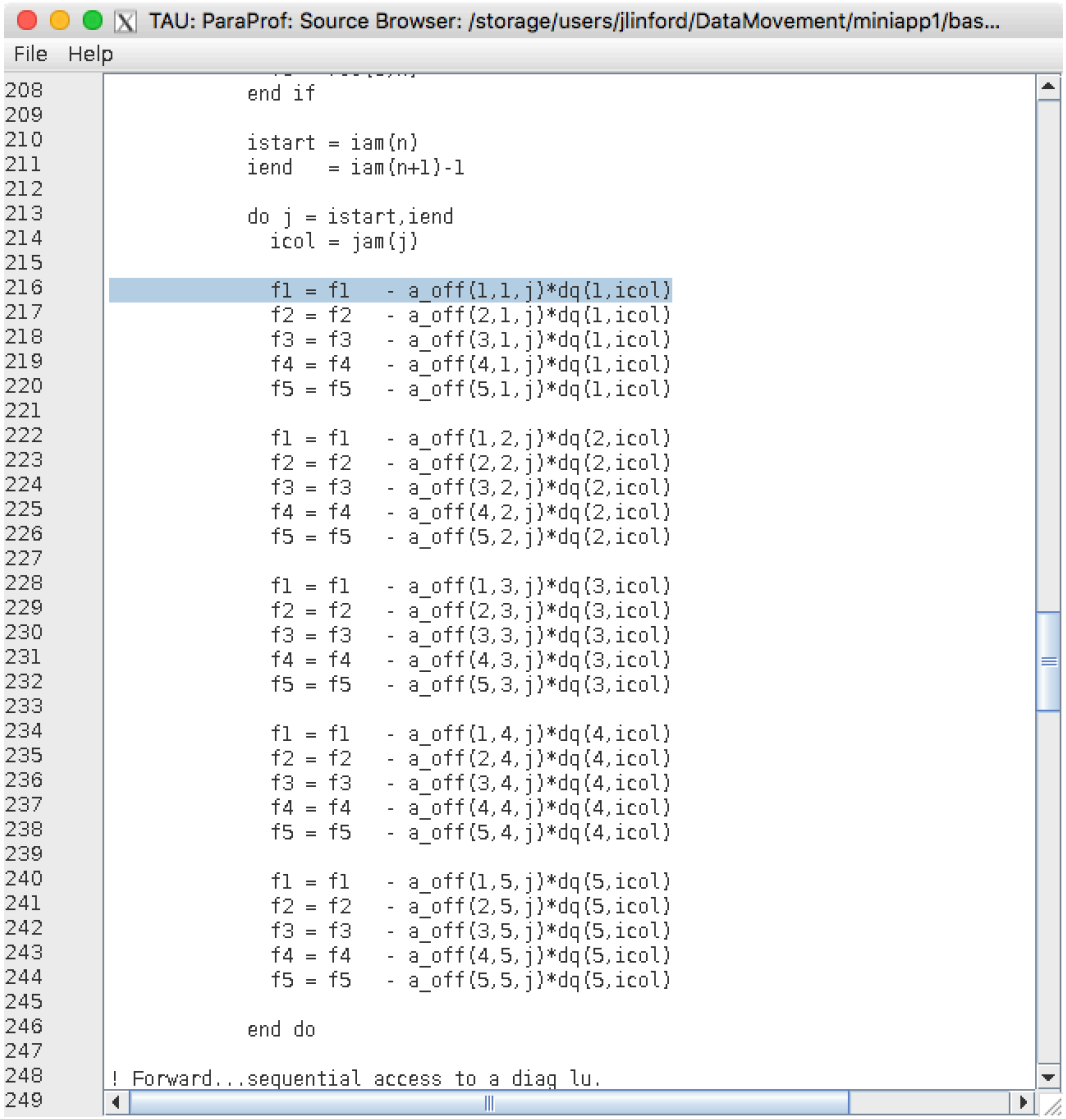
L1 Data Cache Misses



L2 Total Cache Misses



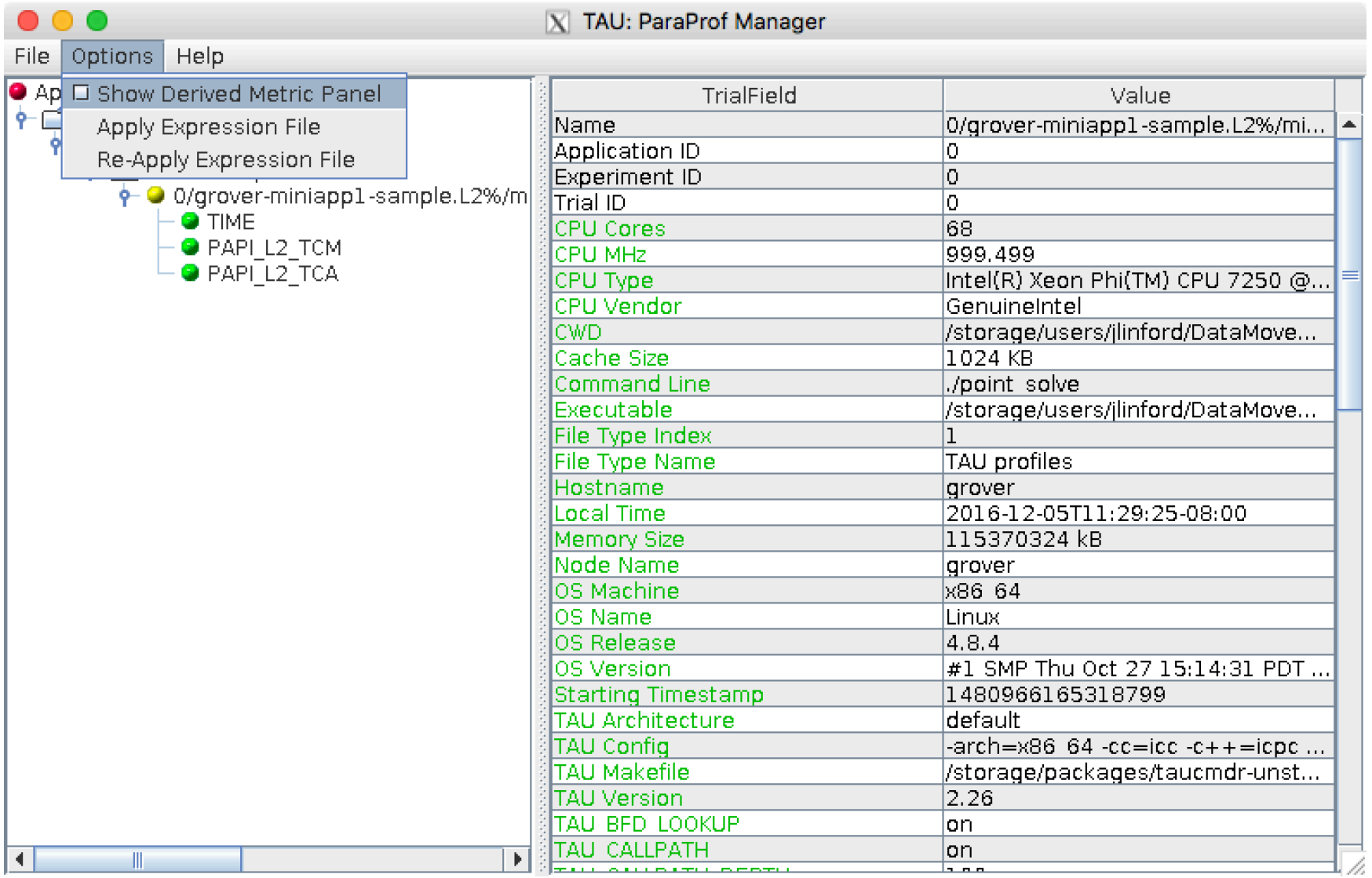
Line with the most L1/L2 misses



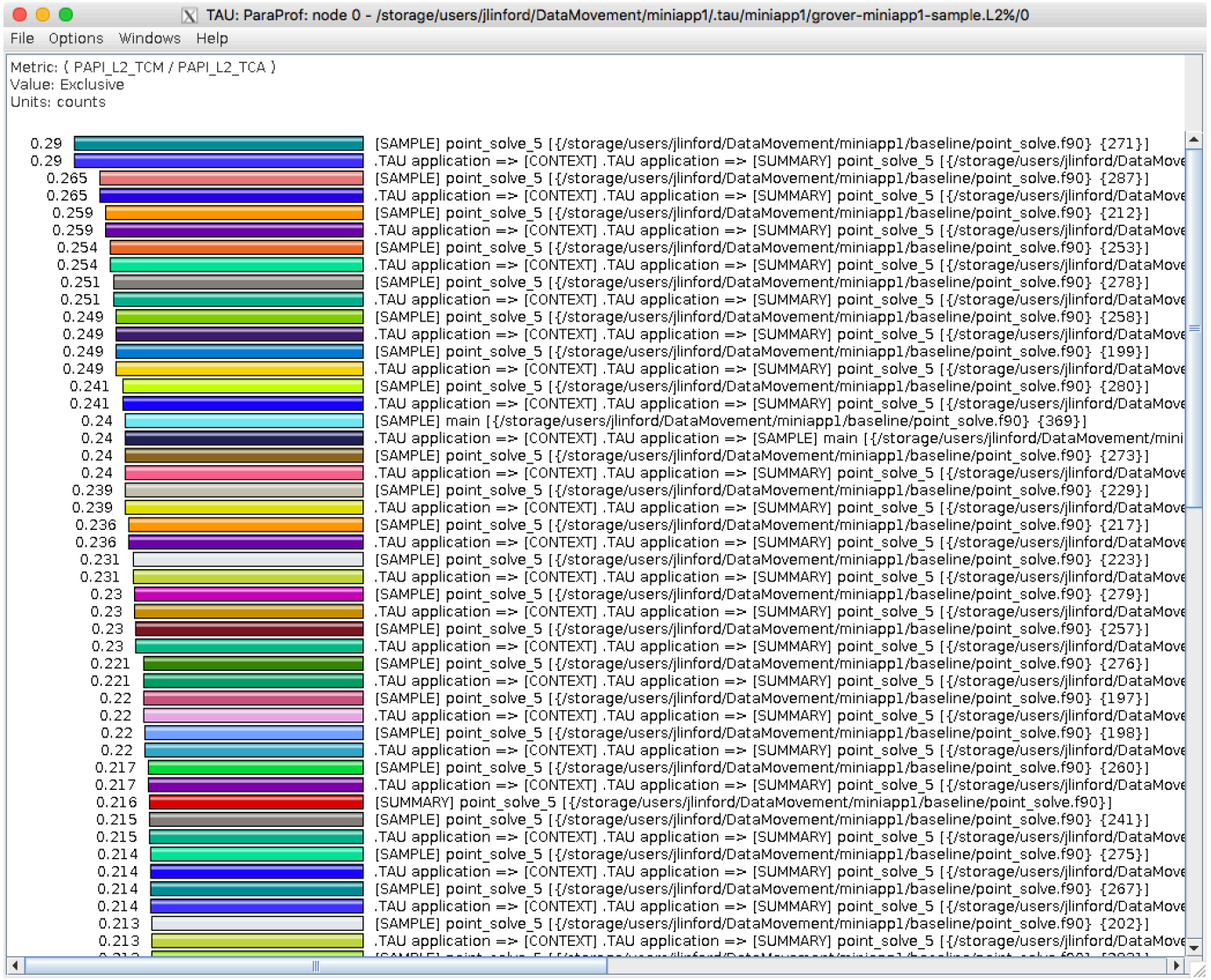
What percent of L2 accesses are misses?

```
$ tau meas copy sample.papi "sample.L2%" --metrics TIME PAPI_L2_TCM PAPI_L2_TCA
[TAU] Added measurement 'sample.L2%' to project configuration 'miniapp1'.

$ tau sel sample.L2%
[TAU] Created a new experiment named 'grover-miniapp1-sample.L2%'.
[TAU] Selected experiment 'grover-miniapp1-sample.L2%'.

$ make run
tau ./point_solve
[TAU]
[TAU] == BEGIN Experiment at 2016-12-05 19:29:24.677341 ===============
[TAU]
[TAU] TAU_CALLPATH=1
[TAU] TAU_CALLPATH_DEPTH=100
[TAU] TAU_COMM_MATRIX=0
[TAU] TAU_METRICS=TIME,PAPI_L2_TCM,PAPI_L2_TCA
```

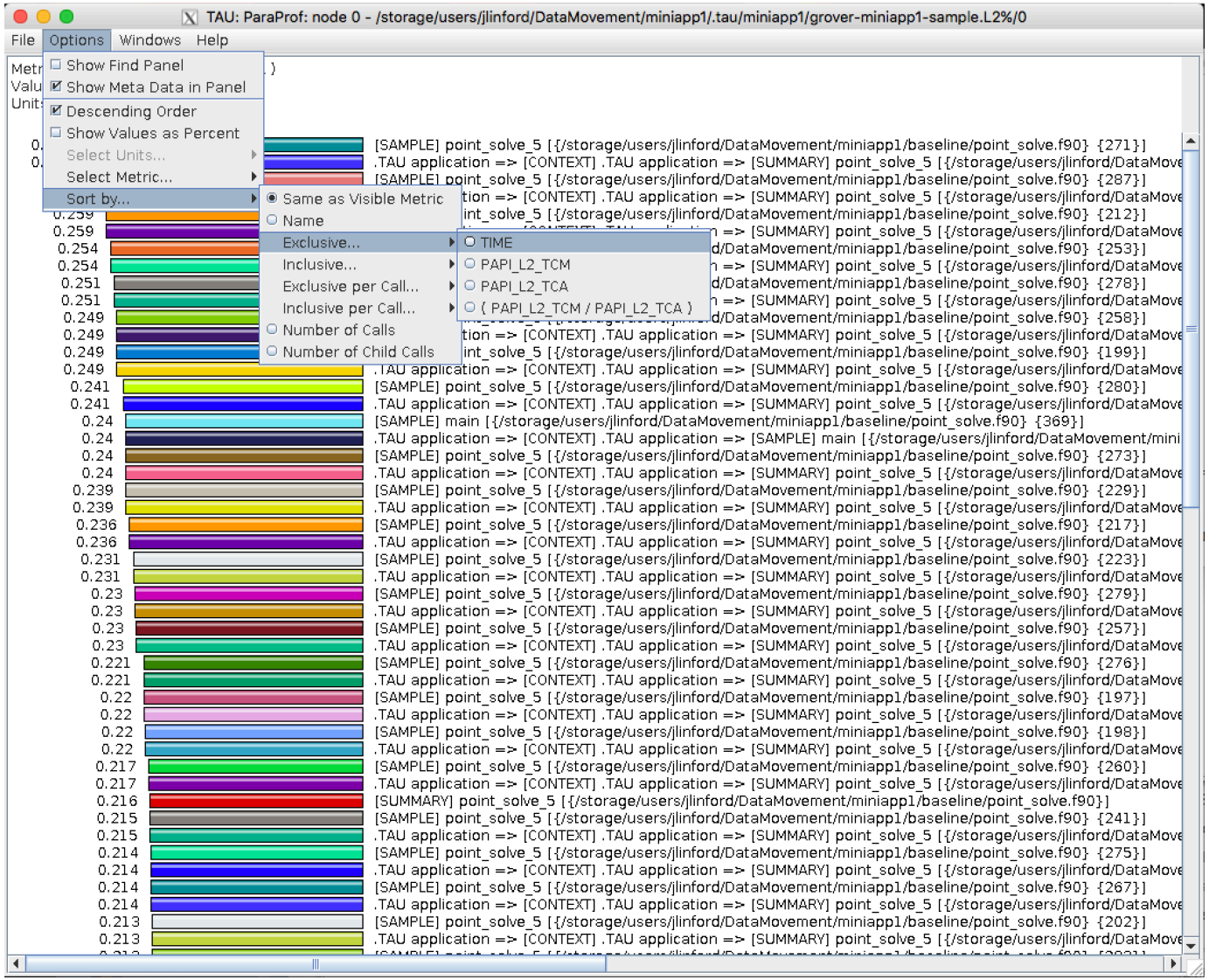
Create a new derived metric



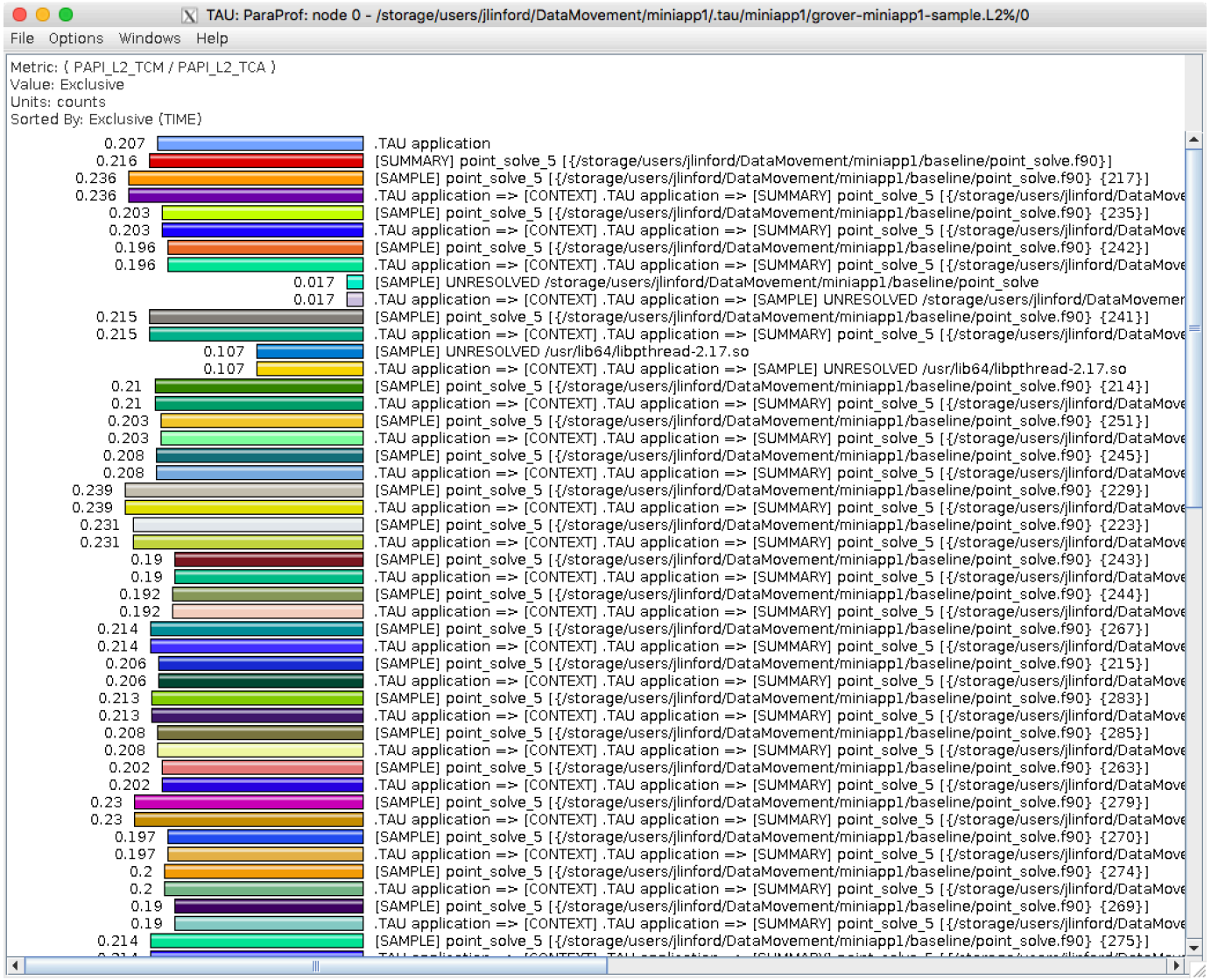
At worst, 29% of L2 fetches miss



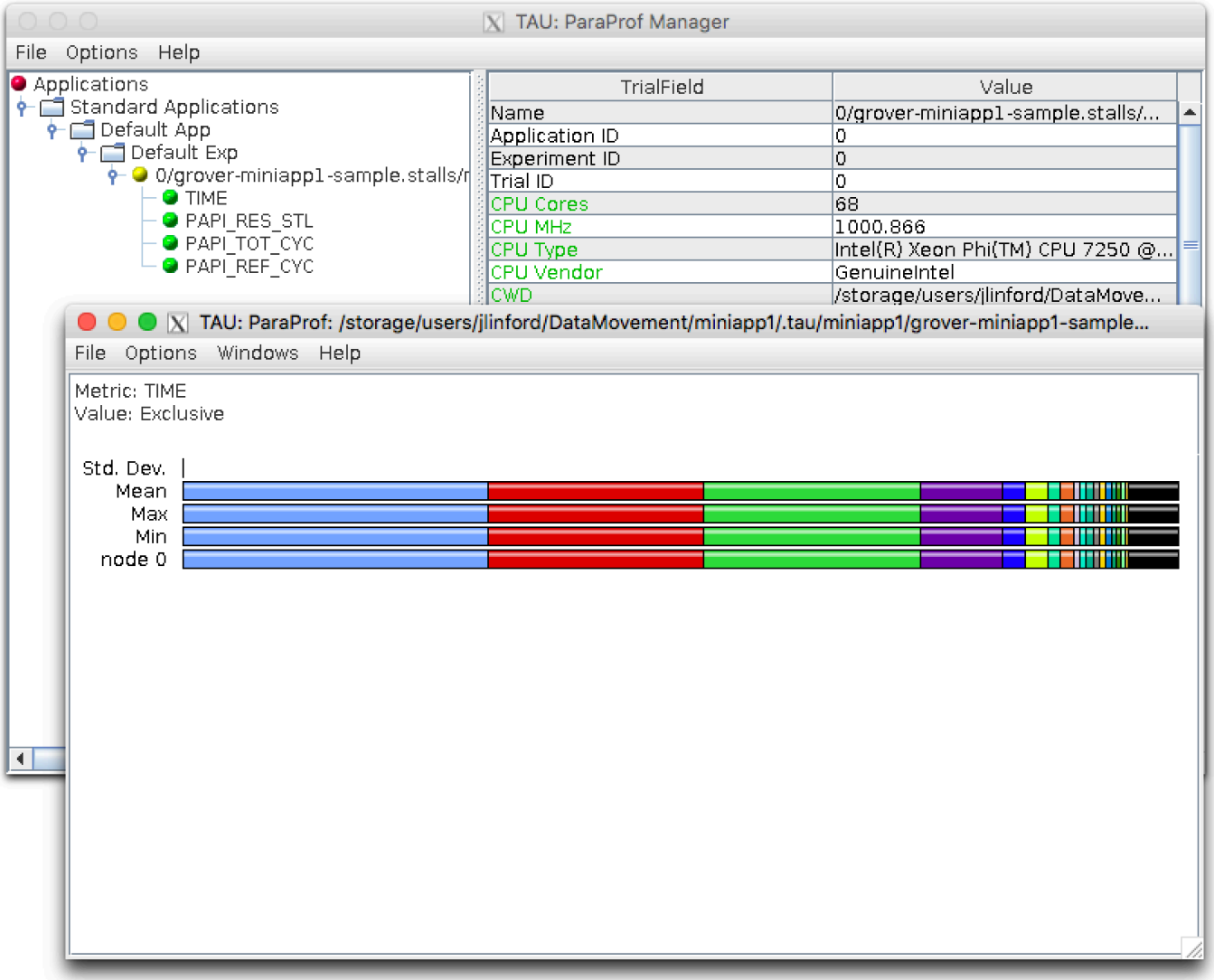
Sort by exclusive time



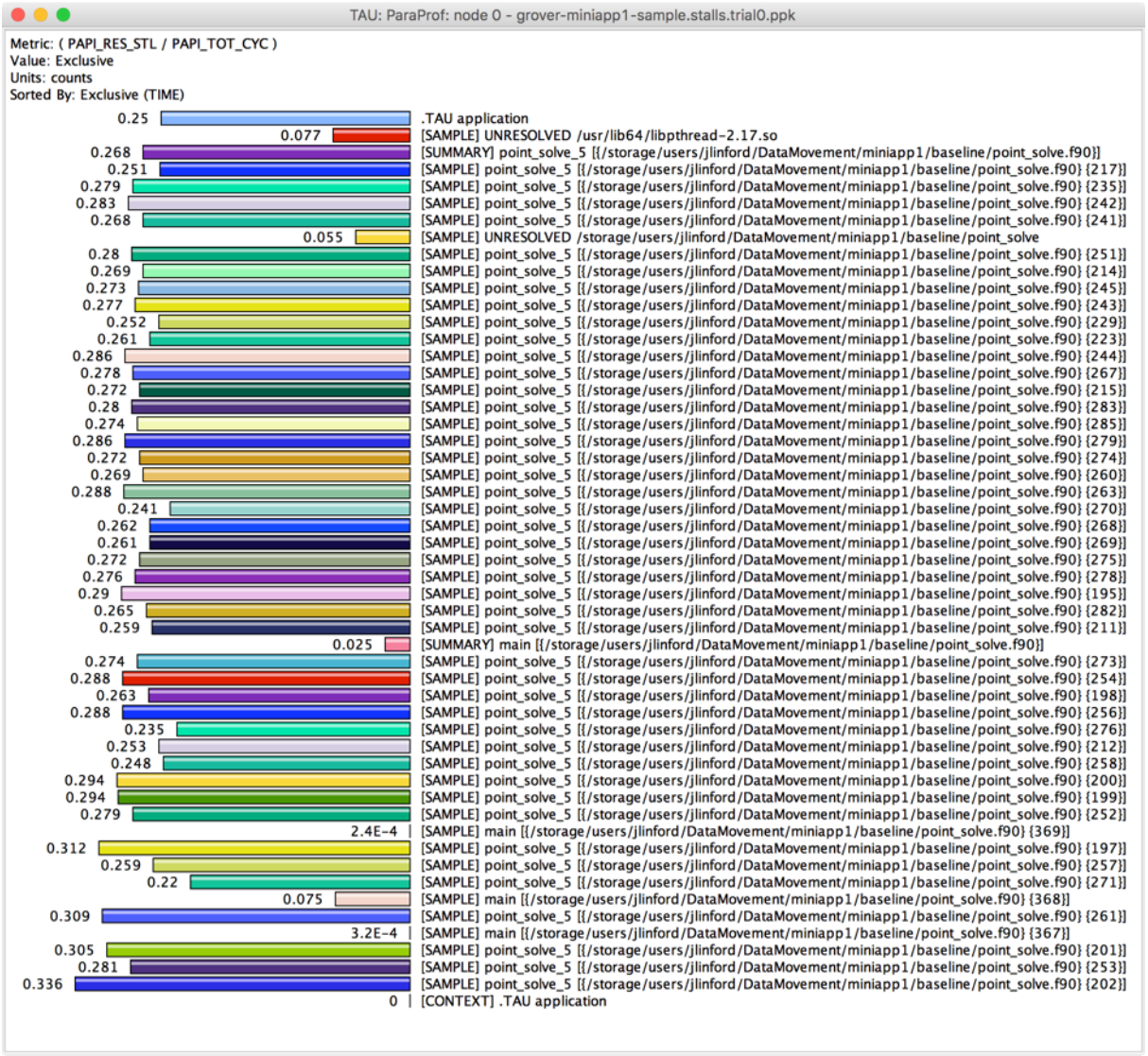
About 21% L2 fetches miss in kernel



% Cycles Stalled Waiting for Memory



% Cycles Stalled Waiting for Memory



Analysis Summary

- We know all runtime hot spots
- We know cache utilization characteristics
  - DRAM is the bottleneck
- MCDRAM will likely help
- OpenMP will likely help
- Can we shuffle a_off for better performance?
  - No: fetch utilization is already close to 100%

244	64.4%	f1 = f1 - a_off(1,5,j)*dq(5,icol)							
	% of fetches	Miss ratio	Fetch ratio	WB ratio	Fetch Util	WB Util	PC	Type	Issues
	13.0%	31.5%	31.5%	0.0%	60.4%	*100.0%*	0x421ede	R	ST
	37.9%	100.0%	100.0%	0.0%	96.2%	*100.0%*	0x421ee5	R	ST Pf NT
	12.8%	31.0%	31.0%	0.0%	96.2%	*100.0%*	0x421f0f	R	ST Pf NT

- Can we vectorize?
  - Yes, but don't try too hard

Data Movement Workshop

IMPROVING MINIAPP1

OpenMP Parallelization

```
$ cd ~/DataMovement/miniapp1/openmp

!$omp parallel default(shared)
do sweep = 1, n_sweeps
  do color = sweep_start, sweep_end, sweep_stride
    do ipass = 1, 2
      start = color_indices(1,color)
      end = color_boundary_end(color)

      !$omp do private(f1,f2,f3,f4,f5,n,j,icol,istart,iend) schedule(auto)
      do n = start, end
        istart = iam(n)
        iend   = iam(n+1)-1

        f(1:5) = (+/-)res(1:5)

        do j = istart, iend
          icol = jam(j)
          do i = 1, 5
            f(1:5) = f(1:5) - a_off(1:5,i,j)*dq(i,icol)
          end do
```

Create a new OpenMP Application Config

```
$ cd ~/DataMovement/miniapp1/openmp

# Edit Makefile as before

$ tau app copy miniapp1 miniapp1.openmp --openmp
[TAU] Added application 'miniapp1.openmp' to project configuration 'miniapp1'.

$ tau select miniapp1.openmp sample
[TAU] Selected experiment 'grover-miniapp1.openmp-sample'.
[TAU] Application rebuild required:
[TAU]   - openmp changed from False to True
```

Compile and run exactly as before

```
jlinford — jlinford@grover:~/DataMovement/miniapp1/openmp — ssh grover.nic.uoregon.edu — 115×44
[jlinford@grover ~/DataMovement/miniapp1/openmp $ make run
tau ./point_solve
[TAU]
[TAU] == BEGIN Experiment at 2016-12-05 20:09:09.858877 ============================================================
[TAU]
[TAU] TAU_CALLPATH=1
[TAU] TAU_CALLPATH_DEPTH=100
[TAU] TAU_COMM_MATRIX=0
[TAU] TAU_METRICS=TIME
[TAU] TAU_PROFILE=1
[TAU] TAU_SAMPLING=1
[TAU] TAU_THROTTLE=1
[TAU] TAU_THROTTLE_NUMCALLS=100000
[TAU] TAU_THROTTLE_PERCALL=10
[TAU] TAU_TRACE=0
[TAU] TAU_TRACK_HEAP=0
[TAU] TAU_VERBOSE=0
[TAU] tau_exec -T openmp,serial,icpc -ebs ./point_solve
 Loading data...
0 Number of block 5x5 equations in data file: 1123718
 Done loading data...
 Solving Ax=b...
 Sweep msec on master =   0.172965049743652
 Sweep msec on master =   3.521800041198730E-002
 Sweep msec on master =   3.521800041198730E-002
 Sweep msec on master =   3.514909744262695E-002
 Sweep msec on master =   3.502488136291504E-002
 Sweep msec on master =   3.540110588073730E-002
 Sweep msec on master =   3.527188301086426E-002
 Sweep msec on master =   3.524804115295410E-002
 Sweep msec on master =   3.519201278686523E-002
 Sweep msec on master =   3.761887550354004E-002
 Sweep msec on master =   3.516316413879395E-002
 Sweep msec on master =   3.516983985900879E-002
 Sweep msec on master =   3.525114059448242E-002
 Sweep msec on master =   3.523707389831543E-002
 Sweep msec on master =   3.518295288085938E-002
 Total msec taken on master =   0.684986114501953
 Test passed.
[TAU]
[TAU] == END Experiment at 2016-12-05 20:09:14.581505 ==============================================================
[TAU]
[TAU] Trial 4 produced 1 profile files.
jlinford@grover ~/DataMovement/miniapp1/openmp $
```

About 30x speedup
2.5x vs 24 Haswell 2.4GHz cores

Use MCDRAM via numactl -m

```
$ tau app copy miniapp1 miniapp1.MCDRAM
[TAU] Added application 'miniapp1.MCDRAM' to project configuration 'miniapp1'.

$ tau select miniapp1.MCDRAM sample
[TAU] Created a new experiment named 'grover-miniapp1.MCDRAM-sample'.
[TAU] Selected experiment 'grover-miniapp1.MCDRAM-sample'.

$ numactl -m 1 tau ./point_solve
```

```
$ numastat $point_sovlve_pid

Per-node process memory usage (in MBs) for PID 236528 (point_solve)
                           Node 0          Node 1           Total
                  --------------- --------------- ---------------
Huge                         0.00            0.00            0.00
Heap                         0.00            0.08            0.08
Stack                        0.00            0.04            0.04
Private                      7.50         2169.79         2177.29
----------------  --------------- --------------- ---------------
Total                        7.50         2169.91         2177.41
```

MEMKIND: Move a_off to MCDRAM

```
real(odp), dimension(:,:,:), allocatable, public :: a_off
!dec$ attributes fastmem :: a_off
```

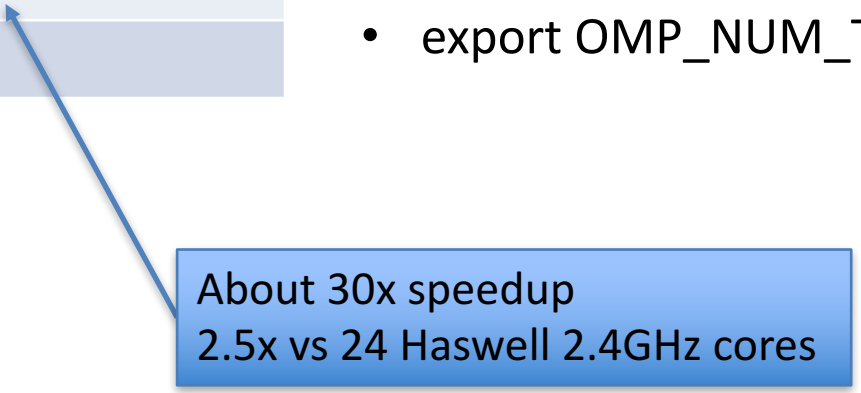
And in Makefile:

```
MEMKIND = /usr/local/packages/memkind-1.0

LDFLAGS = -L$(MEMKIND)/lib -lmemkind

run: all
    LD_LIBRARY_PATH=$(MEMKIND)/lib:$$LD_LIBRARY_PATH tau ./point_solve
```

OpenMP Tuning: Threads per Tile

One Thread per Tile	
DRAM	0.6992
MEMKIND	0.6659
numactl	0.6741

- export KMP_AFFINITY=proclist=[0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66],explicit
- export OMP_NUM_THREADS=34



OpenMP Tuning: Threads per Core

One Thread per Core	
DRAM	0.5604
MEMKIND	0.4235
numactl	0.4336

Two Threads per Core (1 per VPU)	
DRAM	0.5804
MEMKIND	**0.3829**
numactl	0.3830

Three Threads per Core	
DRAM	0.6280
MEMKIND	0.4404
numactl	0.4371

- export KMP_HW_SUBSET=1T
- export KMP_AFFINITY=compact
- export OMP_NUM_THREADS=68

- export KMP_HW_SUBSET=2T
- export KMP_AFFINITY=compact
- export OMP_NUM_THREADS=136

- export KMP_HW_SUBSET=3T
- export KMP_AFFINITY=compact
- export OMP_NUM_THREADS=204

53x speedup
4.5x vs 24 Haswell 2.4GHz cores

2M Hugepages in MCDRAM: Fortran Side

```
use, intrinsic :: iso_c_binding, only: C_F_POINTER, C_FLOAT, C_PTR, C_INT, C_SIZE_T

type(C_PTR) :: a_off_huge_ptr
integer(C_SIZE_T) :: nelm
real(odp), dimension(:,:,:), allocatable, public :: a_off_huge

interface
  function alloc_a_off(memptr, nelm) BIND(C, NAME='alloc_a_off')
    import :: C_PTR, C_INT, C_SIZE_T
    type(C_PTR) :: memptr
    integer(C_SIZE_T), value :: nelm
    integer(C_INT) :: alloc_a_off
  end function alloc_a_off
end interface

nelm = nm*nm*nja
err = alloc_a_off(a_off_huge_ptr, nelm)
if (err /= 0) then
  write(*,*) 'alloc_a_off failed:',err
  stop
end if
call C_F_POINTER(a_off_huge_ptr, a_off_huge, [nm,nm,nja])

a_off_huge = a_off
```

2M Hugepages in MCDRAM: C side

```
#if PAGESIZE == 0x200000
#define PAGESIZE_FLAG HBW_PAGESIZE_2MB
#elif PAGESIZE == 0x1000
#define PAGESIZE_FLAG HBW_PAGESIZE_4KB
#endif
```

```
int alloc_a_off(void ** memptr, size_t nelm)
{
  void * ptr;
  char * poke;
  size_t size;
  size_t span;
  int retval;

  size = nelm*sizeof(float);
  span = (size + PAGESIZE-1) & ~(PAGESIZE-1);

  retval = hbw_posix_memalign_psize(&ptr, PAGESIZE, nelm*sizeof(float), PAGESIZE_FLAG);
  if (retval != 0) {
    char const * errstr = strerror(errno);
    printf("hbw_posix_memalign_psize failed: %s\n", errstr);
    return retval;
  }

  retval = mlock(ptr, span);
  if (retval != 0) {
    char const * errstr = strerror(errno);
    printf("mlock failed: %s\n", errstr);
    return retval;
  }

  retval = posix_madvise(ptr, span, POSIX_MADV_WILLNEED | POSIX_MADV_SEQUENTIAL);
  if (retval != 0) {
    char const * errstr = strerror(errno);
    printf("posix_madvise failed: %s\n", errstr);
    return retval;
  }

  poke = ptr;
  do {
    *poke = 0xf;
    poke += PAGESIZE;
  } while (poke < (ptr+span));

  *memptr = ptr;

  return 0;
}
```

2M Hugepages in MCDRAM

```
Per-node process memory usage (in MBs) for PID 15047 (point_solve)
                           Node 0          Node 1           Total
                  --------------- --------------- ---------------
Huge                       788.00         1024.00         1812.00
Heap                         0.18            0.00            0.18
Stack                        0.03            0.00            0.03
Private                    455.11         1812.00         2267.11
---------------  --------------- --------------- ---------------
Total                     1243.32         2836.00         4079.32
```

But no performance gain…

Army HPC User Group Review

CONCLUSION

MINIAPP1 Summary

What worked

- OpenMP directives
- MEMKIND
- numactl --membind
  - But MEMKIND slightly faster.

What didn't work

- 2M hugepages
  - No change in most tests, performance loss in some
- Blocking in a_off(:,1,1)
  - Facilitates vectorization, but average loop trip count is too low to gain anything.
- Transposing a_off
  - Facilitates vectorization, but average loop trip count is low.
- Vectorizing via intrinsics
  - Not CPU bound, no performance gain.

Acknowledgements

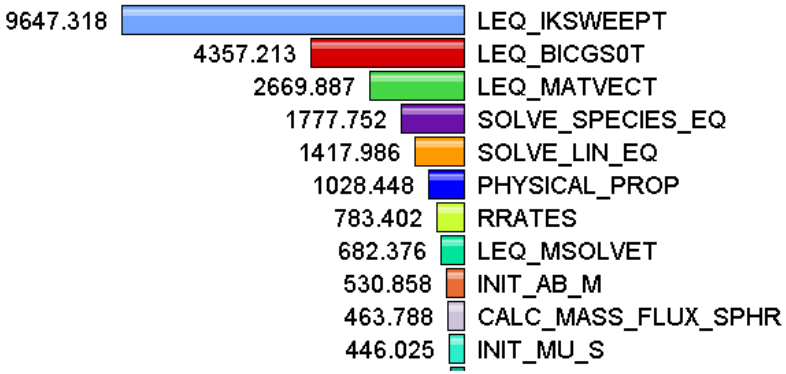
- HPCMP DoD PETTT Program
- Department of Energy
  - Office of Science
  - Argonne National Laboratory
  - Oak Ridge National Laboratory
  - NNSA/ASC Trilabs (SNL, LLNL, LANL)
- National Science Foundation
  - Glassbox, SI-2
- University of Tennessee
- University of New Hampshire
  - Jean Perez, Benjamin 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

Army HPC User Group Review

VOCABULARY

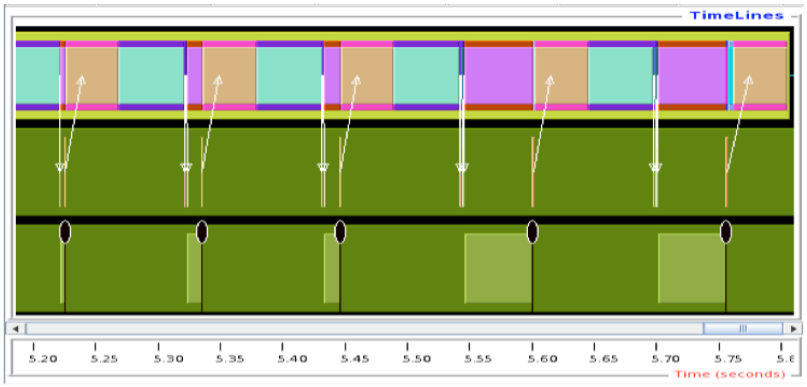
Measurement Approaches

Profiling



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

Tracing



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

Types of Performance Profiles

- *Flat* profiles
  - Metric (e.g., time) spent in an event
  - Exclusive/inclusive, # of calls, child calls, …
- *Callpath* profiles
  - Time spent along a calling path (edges in callgraph)
  - "*main=> f1 => f2 => MPI_Send*"
- *Phase* profiles
  - Flat profiles under a phase (nested phases allowed)
  - Default "main" phase
  - Supports static or dynamic (e.g. per-iteration) phases

Direct Observation Events

- Interval events (begin/end events)
  - Measures exclusive & inclusive durations between events
  - Metrics monotonically increase
  - Example: Wall-clock timer
- Atomic events (trigger with data value)
  - Used to capture performance data state
  - Shows extent of variation of triggered values (min/max/mean)
  - Example: heap memory consumed at a particular point

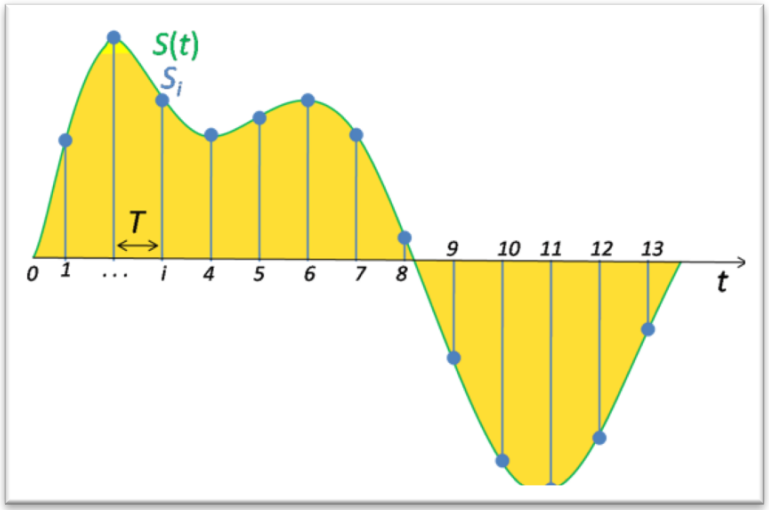
Direct vs. Indirect Measurement

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)

Inclusive vs. Exclusive Measurements

- Exclusive measurements for region only
- Inclusive measurements includes child regions