Introduction to Parallel Processing

Tutorial : Profiling and Tracing

09/28/2022

Professor Amanda Bienz

What is Profiling?

- Dynamic program analysis
- Measures various components of program, such as:
  - Memory usage
  - Cache hits / misses
  - Duration of function calls
- Useful in program optimization

Valgrind

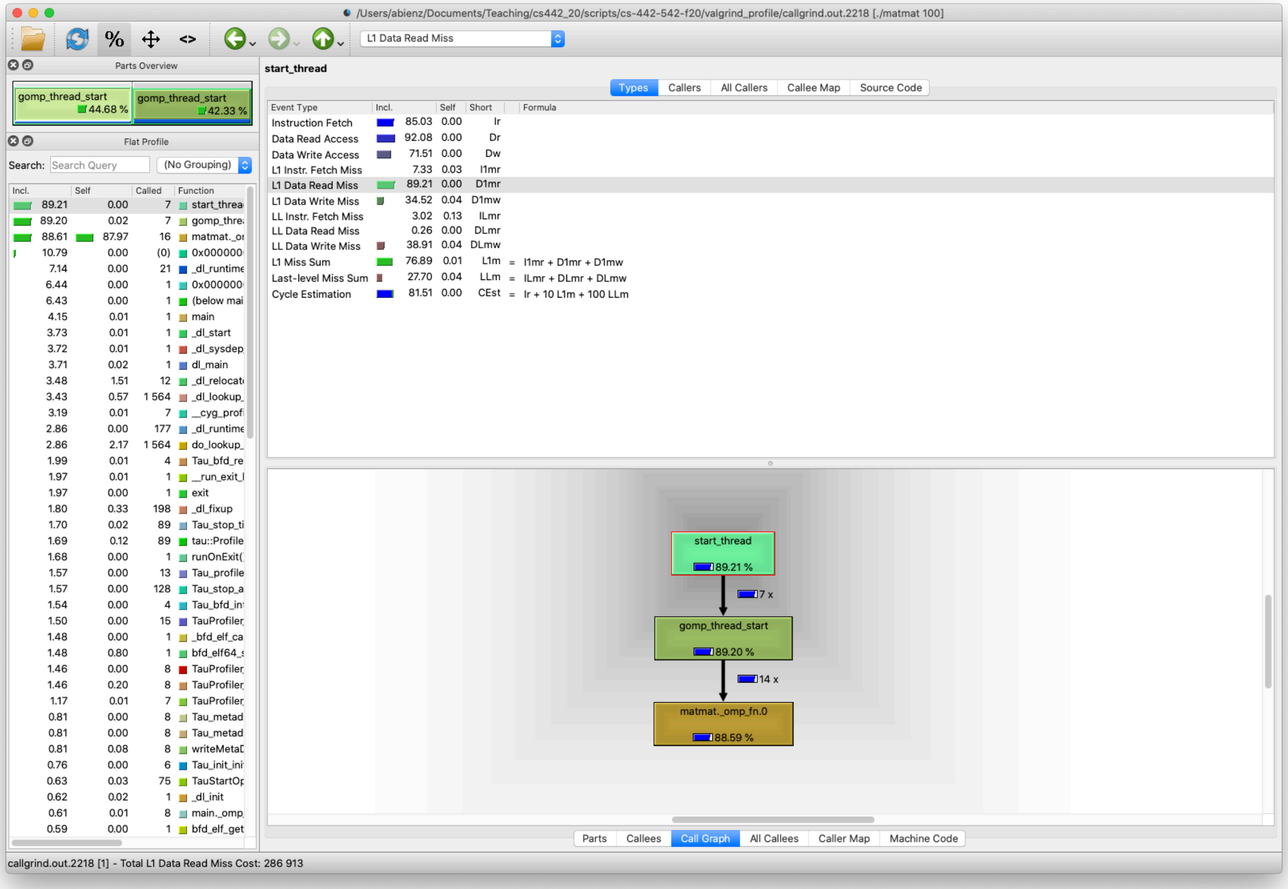
- If you use a programming language without garbage cleanup, I **highly** recommend checking out Valgrind : https://www.valgrind.org
- Debugs and profiles
  - Detects memory management bugs
    - Memory leaks (forget to free data)
    - Accessing unallocated data
  - Detailed profiling can help find program bottlenecks
- Simple command 'valgrind <valgrind_args> ./<filename> <args>’

Cachegrind

- Part of Valgrind
- Measures cache hits / cache misses and cache hit rates
- This can be very helpful in determining if your program is utilizing cache
- ‘valgrind —tool=cachegrind ./<filename> <fileargs>

QCachegrind Profiler

- ‘valgrind —tool=callgrind —dump_instr=yes —simulate-cache=yes —collect-jumps=yes ./<filename> <fileargs>

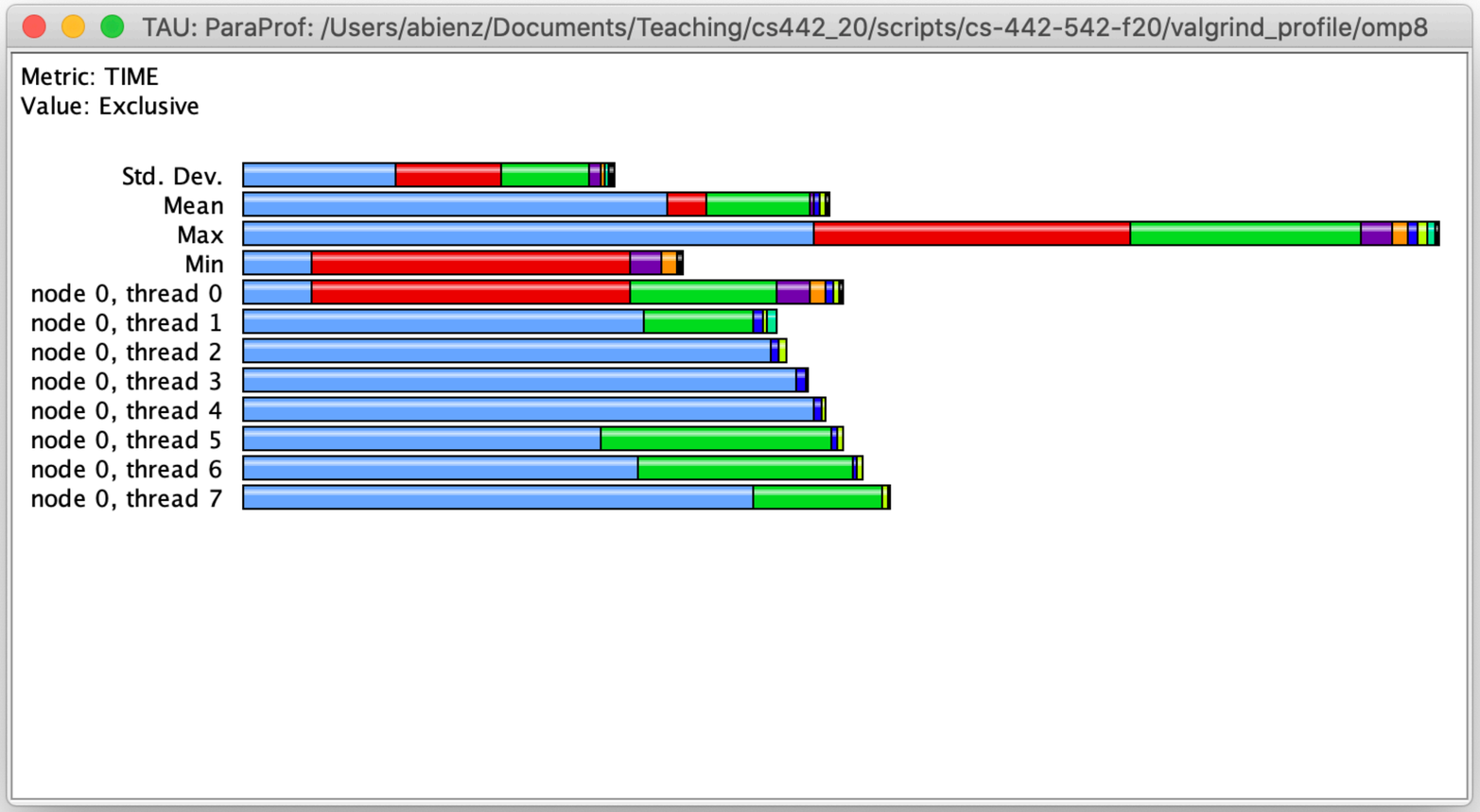


Tau Profiler

- Easy to install (I was able to download and use it on Wheeler) https://www.cs.uoregon.edu/research/tau/home.php
- Profiles how much time is spent in different functions, less about cache hits / misses
- Need to export TAU_MAKEFILE to appropriate file, for instance the version of Tau that is installed with MPI support
- Instead of gcc matmat.c …, replace gcc with tau_cc.sh
- Then run program as normal to get profiles
- Can run MPI code with various process counts to get a variety of profiles

Paraprof

- Comes with TAU
- Visualization tool for profiles



Tracing

- Shows when are where performance is achieved
- Often can be shown as timeline (what happened at each instance during your program)
- Rather than having functions display by duration, displays functions as they are executed during timeline
- In my opinion, this is often most useful way to find performance issues with code, particularly in parallel

Tracing with TAU

- Compile program with tau_cc.sh as before
- Before running program export TAU_TRACE=1
- Get profiles (as normal), an events.0.edf, and files called tautrace
- To visualize:
  - tau_treemerge.pl
  - tau2slog2 tau.trc tau.edf -o tau.slog2
  - jumpshot tau.slog2

Jumpshot

- Comes with TAU to visualize tracing