HP-SEE

TAU Performance System

www.hp-see.eu

Petar Jovanovic
HPC Administrator
Institute of Physics Belgrade
petarj@ipb.ac.rs

HP-SEE

High-Performance Computing Infrastructure
for South East Europe's Research Communities

Outline

- Overview of TAU
- Instrumentation and Measurement Options
- Analysis tools
- Notes on installation and building
- Example

Overview of TAU (1)

- TAU is a performance evaluation tool
- Supports parallel profiling and tracing
- Uses PDT for automatic source level instrumentation
- Profiling and tracing can measure time as well as hardware performance counters from CPU
- Highly portable and is free (BSD style license)
- Comes bundled with analysis tools

Overview of TAU (2)

- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely ported parallel performance profiling system
  - Computer system architectures and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message passing, hybrid, gpgpu
- Integration in complex software, systems and applications

Instrumentation Approach (1)

- Based on direct performance observation
  - Direct instrumentation of program code (probes)
  - Instrumentation invokes performance measurement
  - Event measurement: performance data, meta-data, context
- Instrumentation levels
  - source code, object code, runtime system, library code, executable code, operating system
- Different levels provide different information
- Different tools needed for each level
- Static instrumentation
  - program instrumented prior to execution
- Dynamic instrumentation
  - program instrumented at runtime

Instrumentation Approach (2)

- TAU suports several measurement options (profiling, tracing, profiling with hw counters, etc.)
- Each measurement configuration corresponds to a unique stub makefile and library that is generated when configure script is run during installation.
  - Implies the need for administrator to install all possible configurations users might want.
- Configuration is performed through a number of environment variables.
- Target applications are run using TAU's compile or runtime wrappers
- Analysis of gathered performance data is done using tools such as paraprof, jumpshot, etc.

Analysis Tools

- Paraprof
  - graphical representation of profiling data
  - profiles of various metrics, on functions or threads
  - callpath profile and call graph
  - MPI communication matrix
- Jumpshot
  - used for trace visualization.
  - provides timeline and histogram views
  - scalable level of detail support
- PerfExplorer and PerfDMF
  - performance data mining
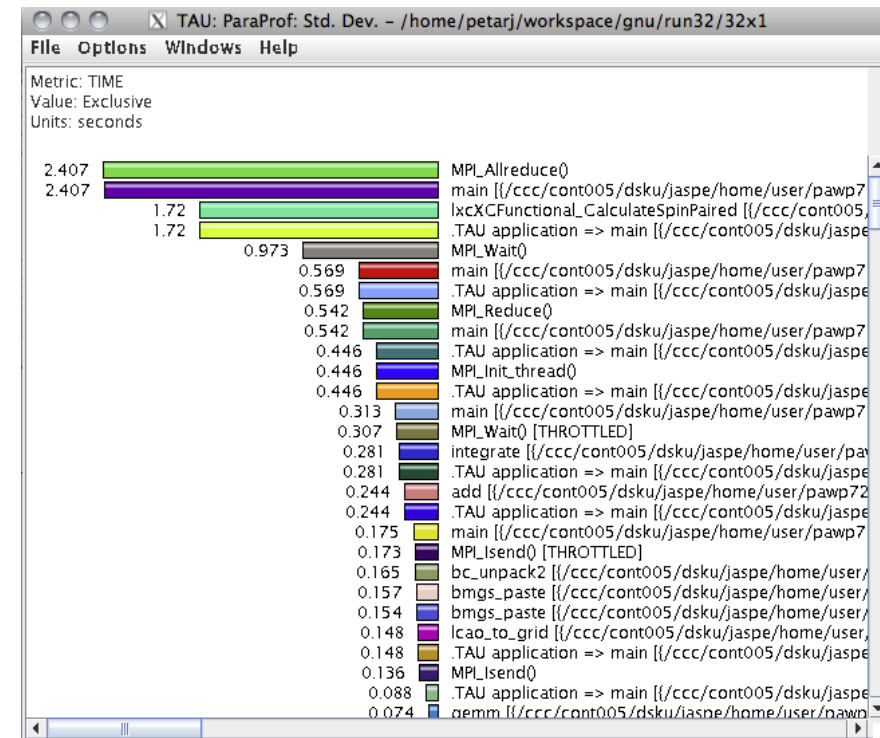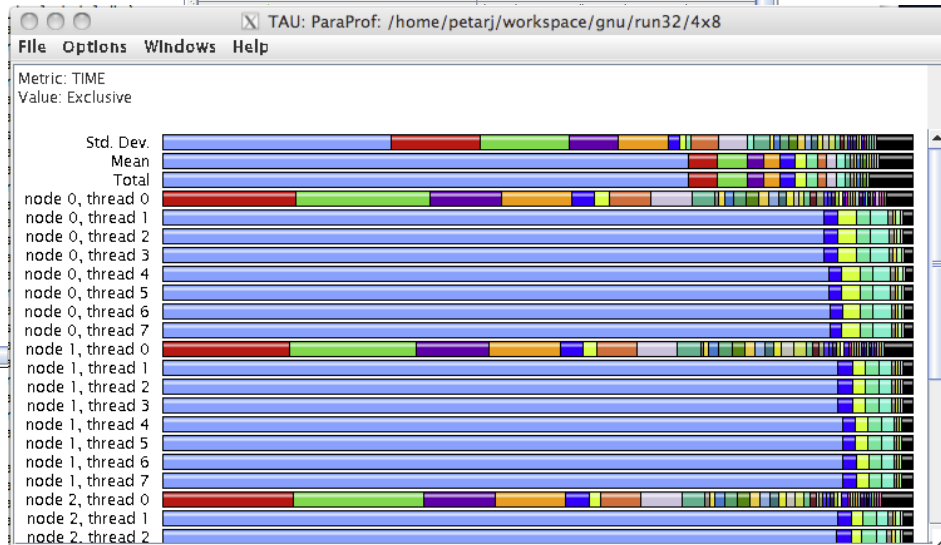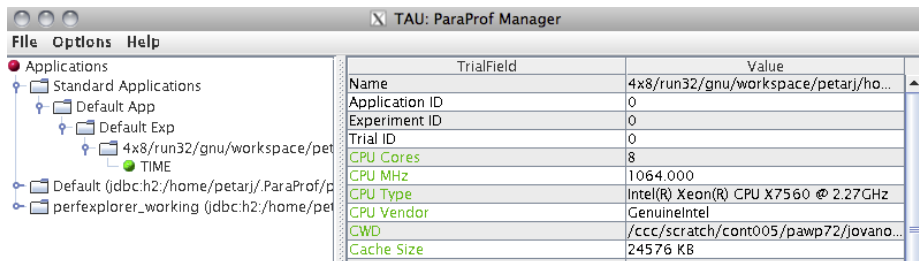 - used for management of large databases of performance data

Analysis Tools
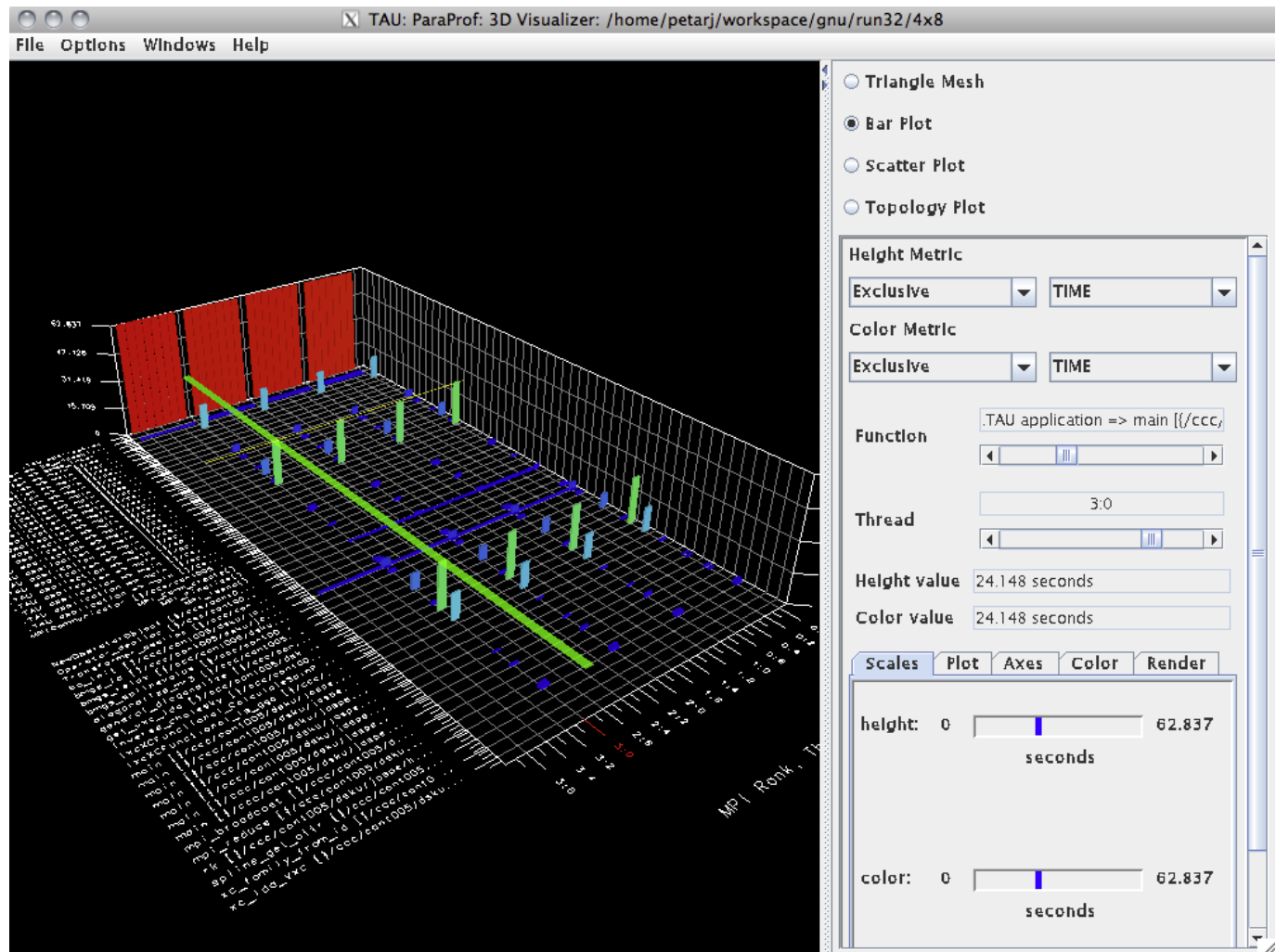
- Paraprof

Analysis Tools

- Paraprof

Analysis Tools

- Paraprof
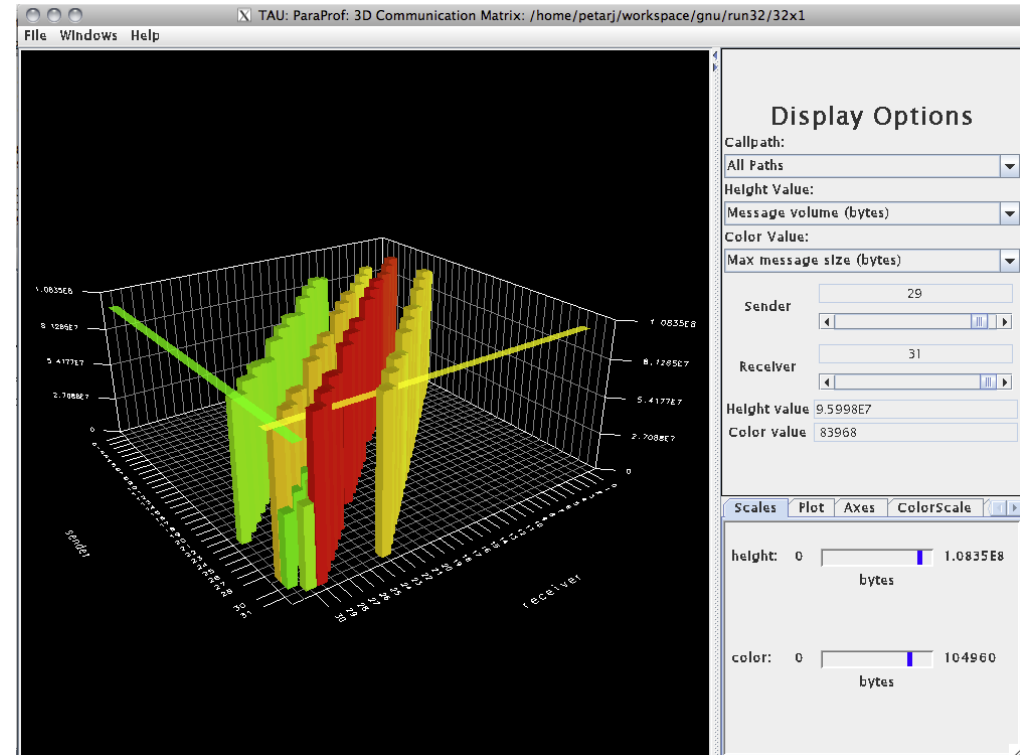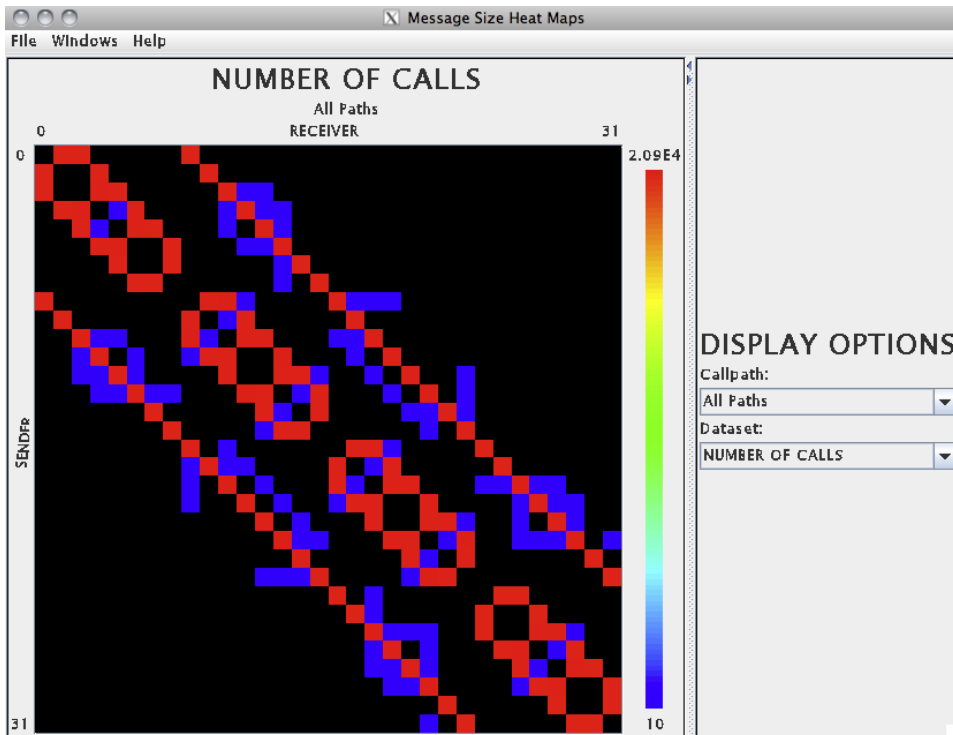
Analysis Tools

- Jumpshot

Notes on installation and building

- Usually responsibility of system administrator
- Since much of what can be done depends on how TAU was built, odds are user will have to build her own custom installation
- Java GUI is provided for configuration and building:

```
$ ./tau_setup
$ ./installtau
```
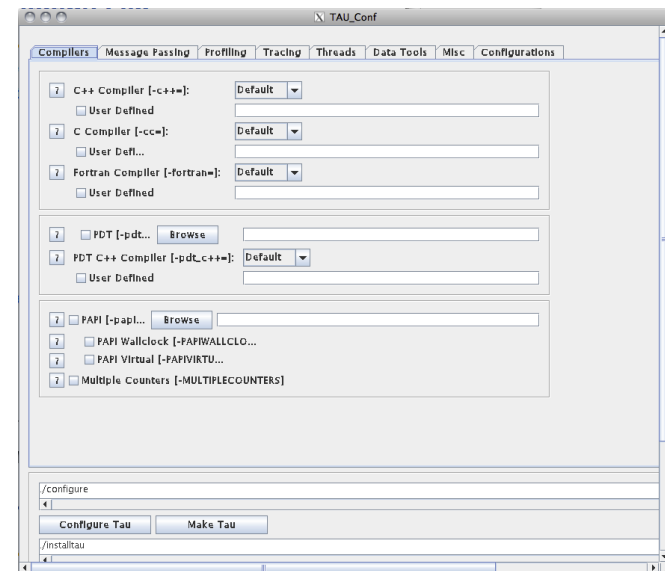
Example

- Demonstrates TAU instrumentation of a threaded C application (using pthreads)
- Covers basic use of bundled profile and trace analysis tools
- The code is part of standard examples from TAU package which can be found in $TAUROOT/examples/cthreads

References

- TAU site: http://www.cs.uoregon.edu/Research/tau/home.php
- TAU documentation: http://www.cs.uoregon.edu/Research/tau/docs.php
- VI-HPS: http://www.vi-hps.org/training/material