Introduction to the TAU Performance System®

Leap to Petascale Workshop 2012 at Argonne National Laboratory,
ALCF, Bldg. 240,# 1416, May 22-25, 2012, Argonne, IL
Sameer Shende, U. Oregon
sameer@cs.uoregon.edu
http://tau.uoregon.edu

These slides are available on Cetus/Intrepid in
/soft/perftools/tau/ppt
Acknowledgements: U. Oregon, ParaTools, Inc.

- Dr. Allen D. Malony, Professor, CIS Dept, and Director, NeuroInformatics Center, and CEO, ParaTools, Inc.
- Dr. Kevin Huck, Computer Scientist, ParaTools, Inc.
- Dr. John Linford, Computer Scientist, ParaTools, Inc.
- Dr. Chee Wai Lee, Senior software engineer, UO
- Wyatt Spear, Software engineer, UO, ParaTools, Inc.
- Suzanne Millstein, Software engineer, UO
- Scott Biersdorff, Software engineer, UO
- Nick Chaimov, Ph.D. student, UO
- William Voorhees, Ph.D. student, UO
- Dr. Robert Yelle, Research faculty, UO
What is TAU?

• TAU is a performance evaluation tool
• It supports parallel profiling and tracing toolkit
• Profiling shows you how much (total) time was spent in each routine
• Tracing shows you when the events take place in each process along a timeline
• Profiling and tracing can measure time as well as hardware performance counters from your CPU
• TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
• It supports C++, C, Chapel, UPC, Fortran, Python and Java
• TAU runs on all HPC platforms and it is free (BSD style license)
• TAU has instrumentation, measurement and analysis tools
• To use TAU, you need to set a couple of environment variables and substitute the name of the compiler with a TAU shell script
TAU Performance System®

- Integrated toolkit for performance problem solving
  - Instrumentation, measurement, analysis, visualization
  - Portable performance profiling and tracing facility
  - Performance data management and data mining
- Based on direct performance measurement approach
- Open source
- Available on all HPC platforms
- http://tau.uoregon.edu

ParaTools
Performance Evaluation

- **Profiling**
  - Presents summary statistics of performance metrics
    - number of times a routine was invoked
    - exclusive, inclusive time/hpm counts spent executing it
    - number of instrumented child routines invoked, etc.
    - structure of invocations (calltrees/callgraphs)
    - memory, message communication sizes also tracked

- **Tracing**
  - Presents when and where events took place along a 
global timeline
    - timestamped log of events
    - message communication events (sends/receives) are tracked
      - shows when and where messages were sent
    - large volume of performance data generated leads to more 
      perturbation in the program
TAU Performance Profiling

- Performance with respect to nested event regions
  - Program execution event stack (begin/end events)
- Profiling measures inclusive and exclusive data
- Exclusive measurements for region only performance
- Inclusive measurements includes nested “child” regions
- Support multiple profiling types
  - Flat, callpath, and phase profiling
TAU Performance System Architecture

Instrumentation
- source code
- object code
- library wrapper
- binary code
- virtual machine

Measurement
- Event creation and management
  - event identifier
  - entry/exit events
  - atomic events
  - event mapping
  - event control

Profiling
- statistics
- atomic profiles
- entry/exit profiles

Phase profiles
- I/O profiles
- profile sampling

Tracing
- trace buffering
- record creation
- trace I/O
- timestamp generation
- trace filtering
- trace merging

Performance data sources
- timing
- hardware counters
- system counters
- kernel

OS and runtime system modules
- threading
- interrupts
- runtime system
- I/O
TAU Performance System Architecture

Analysis

Profile Data Management (PerfDMF)
- profile translators
- Metadata (XML)
- profile database

Profile Analysis (ParaProf)

Profile Data Mining (PerfExplorer)

Trace Data Management
- trace translators
- trace storage

Trace Visualizers
- Vampir
- JumpShot
- Paraver
- Vampir Server

Trace Analyzers
- Expert
- ProfileGen
Program Database Toolkit (PDT)

- C / C++ parser
- Fortran parser F77/90/95
- IL analyzer
- C / C++ IL analyzer
- Program documentation
- Application component glue
- C++ / F90/95 interoperability
- Automatic source instrumentation

ParaTools

Program Database Files

DUCTAPE

PDBhtml

SILOON

CHASM

tau_instrumentor
Automatic Source-Level Instrumentation in TAU

Diagram:
- TAU source analyzer
- Application source
- Parsed program
- tau_instrumentor
- Instrumentation specification file
- Instrumented source
Using TAU: A brief Introduction

- TAU supports several measurement options (profiling, tracing, profiling with hardware counters, etc.)
- Each measurement configuration of TAU corresponds to a unique stub makefile that is generated when you configure it
- To instrument source code using PDT
  - Choose an appropriate TAU stub makefile in <arch>/lib:
    - % soft add +tau-latest (on BG/Q)
    - % export TAU_MAKEFILE=/soft/perftools/tau/tau_latest/bgq/lib/Makefile.tau-bgqtimers-mpi-pdt
    - % export TAU_OPTIONS='--optVerbose ...' (see tau_compiler.sh -help)
  And use tau_f90.sh, tau_cxx.sh or tau_cc.sh as Fortran, C++ or C compilers:
    - % mpixlf90_r foo.f90
  changes to
    - % tau_f90.sh foo.f90
    - % qsub -A <...> ./a.out (to submit the job)

- Execute application and analyze performance data:
  - % pprof (for text based profile display)
  - % paraprof (for GUI)
TAU Measurement Configuration on BG/Q

% cd /soft/perftools/tau/tau_latest/bgq/lib; ls Makefile.*
Makefile.tau-pdt
Makefile.tau-mpi-pdt
Makefile.tau-bgqtimers-mpi-pdt
Makefile.tau-bgqtimers-gnu-mpi-pdt
Makefile.tau-mpi-papi-pdt
Makefile.tau-papi-mpi-openmp-opari-pdt
Makefile.tau-pthread-pdt...

- For an MPI+F90 application, you may want to start with:
Makefile.tau-mpi-pdt
  - Supports MPI instrumentation & PDT for automatic source instrumentation
    - % soft add +tau-latest
    - % export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-mpi-pdt
    - % make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
    - % qsub -n 2 -mode c16 -t 10 -A <account> ./a.out
    - % paraprof
TAU Measurement Configuration on BG/P

% cd /soft/perftools/tau/tau_latest/bgp/lib; ls Makefile.*
Makefile.tau-pdt
Makefile.tau-mpi-pdt
Makefile.tau-bgptimers-mpi-pdt
Makefile.tau-opari-openmp-mpi-pdt
Makefile.tau-mpi-papi-pdt
Makefile.tau-papi-mpi-openmp-opari-pdt
Makefile.tau-pthread-pdt...

• For an MPI+F90 application, you may want to start with:
  Makefile.tau-mpi-pdt
  – Supports MPI instrumentation & PDT for automatic source instrumentation
  – % soft add +tau-latest
  – % export TAU_MAKEFILE=$TAU/Makefile.tau-bgptimers-mpi-pdt
  – % make CC=tau_cc.sh CXX=tau_cxx.sh F90=tau_f90.sh
  – % qsub -n 2 -mode smp -t 10 -A <account> ./a.out
  – % paraprof
Parallel Profile Visualization: ParaProf

% soft add +tau-latest
% paraprof (Windows → 3D Visualization)
ParaProf: 3D Communication Matrix

% qsub –env TAU_COMM_MATRIX=1 ...
% paraprof (Windows -> 3D Communication Matrix)
## Interval, Atomic and Context Events in TAU

### Interval Event

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive msec</th>
<th>Inclusive total msec</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name usec/call</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>0.007</td>
<td>0.256</td>
<td>1</td>
<td>5</td>
<td>256 MAIN</td>
</tr>
<tr>
<td>97.3</td>
<td>0.132</td>
<td>0.249</td>
<td>5</td>
<td>5</td>
<td>50 FOO</td>
</tr>
<tr>
<td>40.6</td>
<td>0.104</td>
<td>0.104</td>
<td>5</td>
<td>0</td>
<td>21 BAR</td>
</tr>
<tr>
<td>36.3</td>
<td>0.013</td>
<td>0.093</td>
<td>3</td>
<td>3</td>
<td>31 G</td>
</tr>
</tbody>
</table>

### Context Event

USER EVENTS Profile : NODE 0, CONTEXT 0, THREAD 0

<table>
<thead>
<tr>
<th>NumSamples</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Std. Dev.</th>
<th>Event Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>MEMORY LEAK! malloc size &lt;file=foo.f90, variable=X, line=7&gt; : MAIN =&gt; FOO =&gt; BAR</td>
</tr>
<tr>
<td>2</td>
<td>52</td>
<td>48</td>
<td>50</td>
<td>2</td>
<td>MEMORY LEAK! malloc size &lt;file=foo.f90, variable=X, line=7&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>0</td>
<td>free size &lt;file=foo.f90, variable=X, line=10&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>0</td>
<td>free size &lt;file=foo.f90, variable=X, line=10&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>1</td>
<td>free size &lt;file=foo.f90, variable=X, line=15&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>1</td>
<td>free size &lt;file=foo.f90, variable=X, line=15&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>1</td>
<td>malloc size &lt;file=foo.f90, variable=X, line=13&gt; : MAIN =&gt; FOO =&gt; BAR</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>16</td>
<td>49</td>
<td>22.69</td>
<td>malloc size &lt;file=foo.f90, variable=X, line=13&gt; : MAIN =&gt; FOO =&gt; BAR</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>malloc size &lt;file=foo.f90, variable=X, line=7&gt; : MAIN =&gt; FOO =&gt; BAR</td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>48</td>
<td>60</td>
<td>14.24</td>
<td>malloc size &lt;file=foo.f90, variable=X, line=7&gt; : MAIN =&gt; FOO =&gt; G =&gt; BAR</td>
</tr>
</tbody>
</table>

### Atomic Event

% pprof

ParaTools
Jumpshot [ANL]: Trace Visualization

% qsub –env TAU_TRACE=1 ...
% tau_treemerge.pl
% tau2slog2 tau.trc tau.edf –o app.slog2
% Paratools jumpshot app.slog2
To use TAU on Cetus:

soft add +tau-latest

cp /soft/perftools/tau/tau_latest/examples/matmult/matmult.f90 .

export TAU_MAKEFILE=$TAU/Makefile.tau-bgqtimers-mpi-pdt

tau_f90.sh matmult.f90 -o matmult

qsub -A <YOUR_ACCOUNT> -n 2 --mode c8 -t 10 ./matmult

paraprof

Or use tracing:

qsub -A <YOUR_ACCOUNT> -n 2 --mode c8 -t 10 --env TAU_TRACE=1 ./matmult

tau_treemerge.pl

tau2slog2 tau.trc tau.edf -o app.slog2

jumpshot app.slog2

To profile an application on a large number of nodes:

qsub -A <YOUR_ACCOUNT> -n 64 --mode c16 -t 10 --env TAU_PROFILE_FORMAT="merged" ./matmult

See slides on Cetus: /soft/perftools/tau/ppt/*.ppt

Download TAU from http://tau.uoregon.edu for your desktop.

tau.uoregon.edu/tau.dmg or tau.uoregon.edu/tau.exe or tau.uoregon.edu/tau.tgz

Want more examples? Download workshop examples from:

/soft/perftools/tau/workshop.tar.gz
Acknowledgements

- Department of Energy
  - Office of Science
  - Argonne National Laboratory
  - ORNL
  - NNSA/ASC Trilabs (SNL, LLNL, LANL)

- HPCMP DoD PET Program

- National Science Foundation

- University of Tennessee
  - Shirley Moore
  - Daniel Terpstra

- University of Oregon
  - Allen D. Malony, Chee Wai Lee
  - W. Spear, S. Biersdorff

- TU Dresden
  - Holger Brunst, Andreas Knupfer
  - Wolfgang Nagel

- Research Centre Juelich, Germany
  - Bernd Mohr
  - Felix Wolf