MPI Performance Analysis
TAU: Tuning and Analysis Utilities

Berk ONAT
İTÜ Bilişim Enstitüsü

19 Haziran 2012
Outline

• TAU Parallel Performance System
• Hands on: How to use TAU
• Tools of TAU
• Analysis and Visualizations
• TAU Traces and Profiles
TAU Parallel Performance System

Features:

- Tuning and Analysis Utilities
- 15+ year project in *University of Oregon*, Research Center Jülich, LANL
- Open source and freeware analysis utilities
- Portable performance tracing and profiling capabilities
- Works with Fortran, C/C++, Java, Python
- Automatic instrumentation of user code
- Own (TAU) visualization tools (*pprof, Paraprof*)
- Support for other visualizing programs (*Jumpshut, Vampir*)
TAU Parallel Performance System

Features:

- Currently only support MPI tracing
- Use own (TAU) compiler wrappers
- Can be built with several MPI implementations
- Can also work with different networks (ethernet, infiniband, myrinet, ...)
- Targets a general complex system computation model:
  1. **Entities**: nodes / contexts / threads
  2. **Multi-level**: system / software / parallelism
  3. **Measurement and analysis abstraction**
General Complex System Model

- **Node:** physically distinct shared memory machine
  - Message passing node interconnection network
- **Context:** distinct virtual memory space within node
- **Thread:** execution threads (user/system) in context
TAU Parallel Performance System

Targets:

- Multi-level performance instrumentation
  - Multi-language automatic source instrumentation
- Flexible and configurable performance measurement
- Widely-portable parallel performance profiling system
  - Computer system architecture and operating systems
  - Different programming languages and compilers
- Support for multiple parallel programming paradigms
  - Multi-threading, message-passing, hybrid
TAU Instrumentation

- **TAU Flexible instrumentation mechanisms at multiple levels:**
  - Source code
    - Manual (TAU API, TAU Component API)
    - Automatic
      - C, C++, F77/90/95 (*Program Database Toolkit* (PDT))
      - OpenMP (directive rewriting (*Opari*), *POMP* spec)
  - Object code
    - Pre-instrumented libraries (e.g., MPI using *PMPI*)
    - Statically-linked and dynamically-loaded (e.g., Python)
  - Executable code
    - Dynamic instrumentation (pre-execution) (*DynInstAPI*)
    - Virtual machine instrumentation (e.g. Java using *JVMPIC*)
TAU Performance System Architecture
## TAU Configuration

**configure [OPTIONS]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c++=&lt;CXX&gt;, -cc=&lt;CC&gt;</code></td>
<td>Specify C++ and C compilers</td>
</tr>
<tr>
<td><code>-pthread, -sproc</code></td>
<td>Use pthread or SGI sproc threads</td>
</tr>
<tr>
<td><code>-openmp</code></td>
<td>Use OpenMP threads</td>
</tr>
<tr>
<td><code>-jdk=&lt;dir&gt;</code></td>
<td>Specify Java instrumentation (JDK)</td>
</tr>
<tr>
<td><code>-opari=&lt;dir&gt;</code></td>
<td>Specify location of Opari OpenMP tool</td>
</tr>
<tr>
<td><code>-papi=&lt;dir&gt;</code></td>
<td>Specify location of PAPI library</td>
</tr>
<tr>
<td><code>-pdt=&lt;dir&gt;</code></td>
<td>Specify location of PDT tools</td>
</tr>
<tr>
<td><code>-dyninst=&lt;dir&gt;</code></td>
<td>Specify location of DynInst Package</td>
</tr>
<tr>
<td><code>-mpi[inc/lib]=&lt;dir&gt;</code></td>
<td>Specify MPI library instrumentation</td>
</tr>
<tr>
<td><code>-shmemp[inc/lib]=&lt;dir&gt;</code></td>
<td>Specify PSHMEM library instrumentation</td>
</tr>
<tr>
<td><code>-python[inc/lib]=&lt;dir&gt;</code></td>
<td>Specify Python instrumentation</td>
</tr>
<tr>
<td><code>-otf=&lt;dir&gt;</code></td>
<td>Specify location of OTF trace library</td>
</tr>
<tr>
<td><code>-arch=&lt;dir&gt;</code></td>
<td>Specify architecture explicitly</td>
</tr>
</tbody>
</table>
TAU Configuration

configure [OPTIONS]

-TRACE
  Generate binary TAU traces

-PROFILE (default)
  Generate profiles (summary)

-PROFILECALLPATH
  Generate call path profiles

-PROFILEPHASE
  Generate phase based profiles

-PROFILEMEMORY
  Track heap memory for each routine

-PROFILEHEADROOM
  Track memory headroom to grow

-MULTIPLECOUNTERS
  Use hardware counters + time

-COMPENSATE
  Compensate timer overhead

-CPUTIME
  Use usertime + system time

-PAPIWALLCLOCK
  Use PAPI’s wall clock time

-PAPIVIRTUAL
  Use PAPI’s process virtual time

-LINUXTIMERS
  Use fast x86 Linux timers
Configuring TAU for Tracing and Profiling

bash: tau-2.17.2/ $ ./configure -c++=icpc -cc=icc
-fortran=intel -pdt=/RS/progs/pdtoolkit/3.15/
-arch=x86_64
-mpi
-mpiinc=/RS/progs/intel/impi/3.1/include64
-mpilib=/RS/progs/intel/impi/3.1/lib64
-PROFILE
-PROFILECALLPATH -PROFILEPARAM
-PROFILEMEMORY -PROFILEHEADROOM
-MULTIPLECOUNTERS -PAPIVIRTUAL
-LINUXTIMERS -CPUTIME -PAPIWALLCLOCK
-MPITRACE -TRACE
Configuring TAU for Tracing and Profiling

- Make and Install TAU

```bash
tau-2.17.2/ $ make install
```

- Will compile TAU and generate TAU Makefile under `x86_64/lib/`

```makefile
Makefile.tau-memory-headroom-callpath-param-linuxtimers-icpc-mpi-papi-perf-pdt-profile-trace-mpitrace
```
Configuring TAU for Tracing and Profiling

Configuring TAU for OpenMP Profiling and **Tracing(?)**

```
bash: tau-2.17.2/ $ ./configure
-c++=icpc -cc=icc
-ffortran=intel
-pdt=/RS/progs/pdtoolkit/3.15
-arch=x86_64
-openmp
-opari
-PROFILE -PROFILECALLPATH --PROFILEPARAM
-PROFILEMEMORY --PROFILEHEADROOM
-MULTIPLECOUNTERS -PAPIVIRTUAL
-LINUXTIMERS --CPUTIME -PAPIWALLCLOCK
-TRACE
```
Configuring TAU for Tracing and Profiling

- Make and Install TAU

  ```bash
  bash: tau-2.17.2/ $ make clean
  bash: tau-2.17.2/ $ make install
  ```

- Will compiler TAU again with new settings and generate new TAU Makefile under x86_64/lib/

Visual Configuration of TAU

bash: $./tau_setup
Compiling with TAU

Compiling your application with TAU compiler in your makefile

MPICC=\texttt{tau\_cc.sh}

\texttt{export TAU\_MAKEFILE=/RS/progs/tau/x86\_64/lib/Makefile.tau-UYBHM\_IntelTingz200510-icpc-papi-shmem-mpi-pthread-pdt-openmp-profile-trace-mpirace}

default: my\_mpi\_omp.x

.o.c:
  $(MPICC) $(CFLAGS) -c $<

my\_mpi\_omp.x: my\_mpi\_omp.o
  $(MPICC) $(CFLAGS) $(LFLAGS) -o $@
Compiling with TAU

Auto Instrumentation using TAU_COMPILERS

- \$\text{(TAU\_COMPILER)}\$ stub Makefile variable in 2.14+ release
- Invokes PDT parser, TAU instrumentor, compiler through \text{tau\_compiler.sh}\$ shell script
- Requires minimal changes to application Makefile
  - Compilation rules are not changed
  - User sets TAU\_MAKEFILE and TAU\_OPTIONS environmental variables
  - User renames the compilers (\text{CC=icc} \rightarrow \text{CC=tau\_cc.sh})
How to Use TAU

- Login to your UHEM node using -X with ssh:

  ```bash
  $ ssh -X du??@wsl-node??.uybhm.itu.edu.tr
  ```

  or use your PuTTY program in your Windows® with X11 forwarding in SSH section.

- Copy example file 20120619.tar to your directory

  ```bash
  $ cd workshop
  $ cp /RS/users/bonat/workshop/YAZOKULU/20120619.tar .
  $ tar -xvf 20120619.tar
  $ cd 20120619/mpi-analyze/tau
  ```
How to Use TAU

- Setting Up Environmental Variables:

  Use `add-ITAC-to-my-PATH.sh` script. Exit and login again with `-X`

```bash
bash: $ ./add-TAU-to-my-PATH.sh
```

Adding source TAU line to your `.bashrc` and/or `.bash_profile`

```bash
export PATH=/RS/progs/pdtoolkit/3.15/x86_64/bin:$PATH
export PATH=/RS/progs/tau/2.19.1/x86_64/bin:$PATH
export PATH=/RS/progs/papi/3.6.2/bin:$PATH
export PATH=/RS/progs/perfctr/2.6.39/bin:$PATH
export LD_LIBRARY_PATH=/RS/progs/papi/3.6.2/lib:
LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/RS/progs/perfctr/2.6.39/lib:
LD_LIBRARY_PATH
```
How to Use TAU

- Using TAU for Measurement

```bash
bash: $ export TAU_MAKEFILE=Makefile.tau-
UYBHM_IntelTingz200510-icpc-papi-shmem-mpi-pthread-pdt-
openmp-profile-trace-mpitrace
bash: $ export TAU_PROFILE=1
bash: $ export TAU_COMM_MATRIX=1
```

- Compiling your application for TAU profiling and tracing

```bash
bash: $ tau_cc.sh heat2d.c -o heat2d.x
bash: $ mpirun -np 8 ./heat2d.x
```
How to Use TAU

- Running your application for performance analysis with TAU

```bash
bash: $ export TAU_MAKEFILE=Makefile.tau-UYBHM_IntelTingz200510-icpc-papi-shmem-mpi-pthread-pdt-openmp-profile-trace-mpitrace
bash: $ export TAU_PROFILE=1
bash: $ export TAU_COMM_MATRIX=1
bash: $ tau_cc.sh heat2d.c -o heat2d.x
```

- Submitting your program to LSF for TAU performance analysis

```bash
bash: $ bsub < tau-submit.lsf
```
How to Use TAU

- Profiling with pprof and paraprof

```bash
bash: $ ls
profile.0.0.0 profile.0.0.1 profile.0.0.2 profile.0.0.3
profile.0.0.4 profile.0.0.5 profile.0.0.6 profile.0.0.7

bash: $ pprof
```

### NODE 0; CONTEXT 0; THREAD 0:

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive msec</th>
<th>Inclusive total msec</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name</th>
<th>Inclusive Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>20</td>
<td>834</td>
<td>1</td>
<td>70</td>
<td>834760</td>
<td>int main()</td>
</tr>
<tr>
<td>74.3</td>
<td>620</td>
<td>620</td>
<td>2</td>
<td>0</td>
<td>310314</td>
<td>void prtdat(int, int, float)</td>
</tr>
<tr>
<td>15.1</td>
<td>125</td>
<td>125</td>
<td>1</td>
<td>0</td>
<td>125820</td>
<td>MPI_Init()</td>
</tr>
<tr>
<td>4.4</td>
<td>36</td>
<td>36</td>
<td>1</td>
<td>0</td>
<td>36948</td>
<td>MPI_Finalize()</td>
</tr>
<tr>
<td>2.5</td>
<td>20</td>
<td>20</td>
<td>21</td>
<td>0</td>
<td>991</td>
<td>MPI_Recv()</td>
</tr>
<tr>
<td>0.9</td>
<td>7</td>
<td>7</td>
<td>42</td>
<td>0</td>
<td>171</td>
<td>MPI_Send()</td>
</tr>
<tr>
<td>0.3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2720</td>
<td>void inidat(int, int, float)</td>
</tr>
<tr>
<td>0.0</td>
<td>0.003</td>
<td>0.003</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>MPI_Comm_size()</td>
</tr>
<tr>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>MPI_Comm_rank()</td>
</tr>
</tbody>
</table>

### 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>21</td>
<td>1.44E+05</td>
<td>4</td>
<td>4.762E+04</td>
<td>6.734E+04</td>
<td>Message size received from all nodes</td>
</tr>
<tr>
<td>42</td>
<td>1.44E+05</td>
<td>4</td>
<td>2.381E+04</td>
<td>5.324E+04</td>
<td>Message size sent to all nodes</td>
</tr>
</tbody>
</table>
### How to Use TAU

**Profiling with pprof and paraprof**

#### FUNCTION SUMMARY (total):

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive msec</th>
<th>Inclusive total msec</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name</th>
<th>Inclusive Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0</td>
<td>179</td>
<td>6,740</td>
<td>8</td>
<td>1711</td>
<td>int main() C</td>
<td></td>
</tr>
<tr>
<td>35.6</td>
<td>2,397</td>
<td>2,397</td>
<td>8</td>
<td>0</td>
<td>MPI_Finalize()</td>
<td></td>
</tr>
<tr>
<td>34.8</td>
<td>2,345</td>
<td>2,345</td>
<td>663</td>
<td>0</td>
<td>MPI_Recv()</td>
<td></td>
</tr>
<tr>
<td>15.9</td>
<td>1,072</td>
<td>1,072</td>
<td>8</td>
<td>0</td>
<td>MPI_Init()</td>
<td></td>
</tr>
<tr>
<td>9.2</td>
<td>620</td>
<td>620</td>
<td>2</td>
<td>0</td>
<td>void prtdat(int, int, float *, char *) C</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>77</td>
<td>77</td>
<td>350</td>
<td>0</td>
<td>void update(int, int, int, float *,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>float *) C</td>
<td></td>
</tr>
</tbody>
</table>

#### FUNCTION SUMMARY (mean):

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive msec</th>
<th>Inclusive total msec</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name</th>
<th>Inclusive Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0</td>
<td>22</td>
<td>842</td>
<td>1</td>
<td>213.875</td>
<td>int main() C</td>
<td></td>
</tr>
<tr>
<td>35.6</td>
<td>299</td>
<td>299</td>
<td>1</td>
<td>0</td>
<td>MPI_Finalize()</td>
<td></td>
</tr>
<tr>
<td>34.8</td>
<td>293</td>
<td>293</td>
<td>82.875</td>
<td>0</td>
<td>MPI_Recv()</td>
<td></td>
</tr>
<tr>
<td>15.9</td>
<td>134</td>
<td>134</td>
<td>1</td>
<td>0</td>
<td>MPI_Init()</td>
<td></td>
</tr>
<tr>
<td>9.2</td>
<td>77</td>
<td>77</td>
<td>0.25</td>
<td>0</td>
<td>void prtdat(int, int, float *, char *) C</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>9</td>
<td>9</td>
<td>43.75</td>
<td>0</td>
<td>void update(int, int, int, float *,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>float *) C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Trace Analyzer & Coll.  19.06.2012
How to Use TAU

- Profiling with paraprof

```
bash: $ ls
profile.0.0.0 profile.0.0.1 profile.0.0.2 profile.0.0.3
profile.0.0.4 profile.0.0.5 profile.0.0.6 profile.0.0.7
bash: $ paraprof
```
Performance Analysis:
Trace Analyzer & Coll.
How to Use TAU

Profiling with paraprof
How to Use TAU

- **TAU Throttle Option**
  - Reducing overhead of TAU by using Throttle environmental variable

```
bash: $ export TAU_THROTTLE=1
bash: $ export TAU_THROTTLE_NUMCALLS=400000
bash: $ export TAU_THROTTLE_PERCALL=3000
bash: $ mpirun -np 8 ./heat2d.x
```
How to Use TAU

- **tau_reduce**: Rule-Based Overhead Analysis for pprof dump files (pprof -d)
  - Analyze the performance data to determine events with high (relative) overhead performance measurements
  - Create a select list for excluding those events
  - Rule grammar:

    
    ![Rule Grammar](image)

    
    **GroupName**: indicates rule applies to events in group
    - **Field**: is an event metric attribute (from profile statistics)
    - **Operator**: is one of >, <, or =
    - **Number** is an integer number
How to Use TAU

- **tau_reduce**: Rule-Based Overhead Analysis File

```plaintext
#exclude all routines that are members of TAU_USER and have
#less than 1000 microseconds
TAU_USER:usec < 1000

#exclude all routines that have higher than 1000 microseconds and are
#called only once.
usec < 1000 & numcalls = 1

#exclude all routines that have less than 1000 usecs per call OR
#have a percent higher than 5
usecs/call < 1000
percent < 5
```
How to Use TAU

- **tau_reduce**: Rule-Based Overhead Analysis Usage

```bash
bash: $ tau_reduce
-f filename specify filename of pprof dump file
-p print out all functions with their attributes
-o filename specify filename for select file output (default: print to screen
-r filename specify filename for rule file
-v verbose mode (for each rule, print out rule and all functions that it excludes)

bash: $ tau_reduce -f app.profile -r rulefile -o new-instfile
```
How to Use TAU

- **tau_reduce**: Rule-Based Overhead Analysis File

```bash
bash: $ pprof -d > app.profile
bash: $ cat rulefile
usecs/call > 100000

bash: $ tau_reduce -f app.profile -r rulefile -o new-instfile
bash: $ cat new-instfile

BEGIN_EXCLUDE_LIST
MPI_Finalize()
void prtdat(int, int, float *, char *) C
int main() C
END_EXCLUDE_LIST
```
How to Use TAU

- Selective Instrumentation

Instfile content:

```c
BEGIN_INSTRUMENT_SECTION
static timer name="master" file="heat2d.c" line=95 to line=132
static timer name="workers" file="heat2d.c" line=138 to line=200
END_INSTRUMENT_SECTION
```

Compiling

```bash
bash: $ tau_cc.sh -tau_makefile=Makefile- ... -tau_options=-optVerbose -opt TauSelectFile=instfile
heat2d.c -o heat2d.x
bash: $ tau-compile-with-instfile instfile
bash: $ mpirun -np 8 ./heat2d.x
bash: $ pprof
```
How to Use TAU

- Selective Instrumentation

pprof output:

<table>
<thead>
<tr>
<th>%Time</th>
<th>Exclusive</th>
<th>Inclusive Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>9,621</td>
<td>TAU application</td>
</tr>
<tr>
<td>57.4</td>
<td>5,518</td>
<td>workers</td>
</tr>
<tr>
<td>46.5</td>
<td>4,472</td>
<td>MPI_Recv()</td>
</tr>
<tr>
<td>24.1</td>
<td>2,317</td>
<td>MPI_Finalize()</td>
</tr>
<tr>
<td>14.8</td>
<td>1,425</td>
<td>void update()</td>
</tr>
<tr>
<td>5.7</td>
<td>549</td>
<td>MPI_Init()</td>
</tr>
<tr>
<td>4.6</td>
<td>437</td>
<td>master</td>
</tr>
<tr>
<td>0.3</td>
<td>31</td>
<td>MPI_Send()</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>void inidat()</td>
</tr>
<tr>
<td>0.0</td>
<td>0.019</td>
<td>MPI_Comm_size()</td>
</tr>
<tr>
<td>0.0</td>
<td>0.007</td>
<td>MPI_Comm_rank()</td>
</tr>
</tbody>
</table>
How to Use TAU

TAU Tracing Data

bash: $ ls
tautrace.0.0.0.trc tautrace.0.0.1.trc tautrace.0.0.2.trc tautrace.0.0.3.trc tautrace.0.0.4.trc tautrace.0.0.5.trc tautrace.0.0.6.trc tautrace.0.0.7.trc events.0.edf events.1.edf events.2.edf events.3.edf events.4.edf events.5.edf events.6.edf events.7.edf
bash: $ tau_treemerge.pl
tautrace.0.0.0.trc: 394 records read.
tautrace.1.0.0.trc: 670 records read.
tautrace.2.0.0.trc: 1270 records read.
tautrace.3.0.0.trc: 1270 records read.
tautrace.4.0.0.trc: 1270 records read.
tautrace.5.0.0.trc: 1270 records read.
tautrace.6.0.0.trc: 1270 records read.
tautrace.7.0.0.trc: 670 records read.
How to Use TAU

bash: $ tau_treemerge.pl
bash: $ tau2slog2 tau.trc tau.edf -o tau.slog2

7728 Records read. 95% converted
7889 Records read. 97% converted
8050 Records read. 99% converted
Reached end of trace file.

SLOG-2 Header:
version = SLOG 2.0.6
NumOfChildrenPerNode = 2
TreeLeafByteSize = 65536
MaxTreeDepth = 1
MaxBufferByteSize = 65527
Categories is FBinfo(1139 @ 115361)
MethodDefs is FBinfo(0 @ 0)
LineIDMaps is FBinfo(100 @ 116500)
TreeRoot is FBinfo(7228 @ 108133)
TreeDir is FBinfo(106 @ 116600)
Annotations is FBinfo(0 @ 0)
Postamble is FBinfo(0 @ 0)

Number of Drawables = 3363
timeElapsed between 1 & 2 = 68 msec
timeElapsed between 2 & 3 = 507 msec

bash: $ jumpshot tau.slog2
Performance Analysis:
Trace Analyzer & Coll.

How to Use TAU

TAU with Jumpshot

19.06.2012