Profiling with TAU: A quick tutorial

Le Yan
Scientific Computing Consultant
User Services @ HPC

Thanks to Dr. Sameer Shende
Code development

- **Debugging**
  - Make sure the code runs and yields correct results

- **Profiling**
  - Analyze performance to identify performance bottlenecks
    - “Hot spots”
    - Load balancing issues

- **Optimization**
  - Make the code run faster and/or consume less resources
Code development

- **Debugging**
  - Make sure the code runs and yields correct results

- **Profiling**
  - Analyze performance to identify performance bottlenecks
    - “Hot spots”
    - Load balancing issues

- **Optimization**
  - Make the code run faster and/or consume less resources
Profiling

• Gather performance statistics during the execution
  • Inclusive and exclusive time
  • Number of calls

• Reflects performance behavior of program entities
  • Routine
  • Loop

• Implemented through
  • Sampling: OS interrupts or hardware counters
  • Instrumentation: calls to measurement functions
What is TAU

- **Tuning and Analysis Utilities**
  - Developed at University of Oregon
- **Scalable and flexible performance analysis toolkit**
  - Performance profiling and tracing utilities
  - Performance data management and data mining
  - Automatic instrumentation through Program Database Toolkit (PDT)
  - Also provides an instrumentation API
Availability on Linux clusters

- Linux clusters
  - Softenv key: “tau-2.18-intel-11.1-mvapich-1.1”
- AIX clusters
  - Softenv key: “+tau-2.1.6”
- Note: PAPI is not available at the moment, so TAU is unable to provide hardware counters
Usage

- Add the softenv key for TAU to your .soft file and resoft
- Compile your code with the TAU scripts
  - `tau_f90.sh` for Fortran, `tau_cc.sh` for C and `tau_cxx.sh` for C++
  - Your code will be instrumented automatically
- Execute the generated executable as usual
  - Profile data files: `profile.x.x.x`
- Analyze/visualize the results with Paraprof
Exercise 1

- Add softenv key(s) and resoft
- Copy `pi.f90` from `/home/lyan1/traininglab/tau/single_file`
- Compile it with the TAU Fortran compiler
- Run it and check the results with `paraprof`
Paraprof Manager Window

LSU HPC Training Series
May 6, 2009
Main Data Window – Stacked Bar
Main Data Window – Unstacked Bar
Function Data Window – Bar Chart

[Image of a bar chart showing performance data for different nodes.]
Function Data Window - Histogram
3D View
Individual thread view

TAU: ParaProf: n,c,t 3,0,0 - /work/lyan1/ClusterTest

File Options Windows Help

<table>
<thead>
<tr>
<th>Metric: Time</th>
<th>Value: Exclusive</th>
<th>Units: seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.622</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

void LAMMPS_NS::PairList::compute(int, int) {pair...
NPI_Allreduce()
2.051
void LAMMPS_NS::Neighbor::half_bin_newton(LAMMP...
1.244
void LAMMPS_NS::Comm::communicate() {comm.cpp}
1.103
void LAMMPS_NS::Comm::reverse_communicate() {c...
0.735
void LAMMPS_NS::FixNVE::initial_integrate(int) {fix_n...
0.624
NPI_Wait()
0.403
void LAMMPS_NS::FixNVE::final_integrate() {fix_nve.c...
0.367
NPI_Init()  
0.214
NPI_Recv()
0.207
int LAMMPS_NS::Neighbor::check_distance() {neighbor...
0.189
NPI_Cart_create()
0.189
void LAMMPS_NS::Verlet::iterate(int) {verlet.cpp}  12
0.17
void LAMMPS_NS::Verlet::force_clear() {verlet.cpp}
0.134
NPI_Send()
0.101
NPI_Sendrcv()
0.094
void LAMMPS_NS::FixSetForce::post_force(int) {fix_s...
0.084
void LAMMPS_NS::Neighbor::bin_atoms() {neighbor...
0.064
void LAMMPS_NS::Comm::borders() {comm.cpp} 63
0.058
void LAMMPS_NS::Comm::exchange() {comm.cpp} 5
0.051
void LAMMPS_NS::AtomVecAtomic::unpack_reverse()
0.046
int LAMMPS_NS::AtomVecAtomic::pack_comm(int, int
0.043
void LAMMPS_NS::Timer::stamp(int) {timer.cpp} 52
0.043
void LAMMPS_NS::Timer::stamp() {timer.cpp} 43, 1
0.037
NPI_Inetv0
0.035
void LAMMPS_NS::Integrate::ev_set(int) {integrate.c...
0.025
void LAMMPS_NS::Modify::initial_integrate(int) {mod...
Comparing multiple threads
Callpath Profile
Paraprof

- **Java-based analysis and visualization tool for performance data**
- "pprof" is for text based profile display
- Can work with profile data generated by other profiling tools, e.g. MpiP
Paraprof options

- `-f <filetype>`
  - Specify type of performance data
- `-m`
  - Perform runtime monitoring of profile data
- `--pack <file>`
  - Pack profile data into one file
- `--dump`
  - Dump profile into TAU profile data format
Options for TAU Compiler Scripts

• Display available options with “tau_xxx.sh -help”

• Options
  • -optVerbose: display verbose debugging information
  • -optKeepFiles: keep intermediate files
  • -optDetectMemoryLeaks: track malloc/ free calls
  • -optGnuFortranParser: Specify the GNU gfortran PDT parser
gfparse instead of f95parse
  • -optPreProcess: Preprocess the source code before parsing
  • ...

LSU HPC Training Series
May 6, 2009
TAU Intermediate Files

- Enable the '-optKeepFiles' option when compiling the code to preserve the instrumented source code

```
[lyan1@poseidon2 single_file]$ ll
total 16
-rwxr-xr-x 1 lyan1 loniadmin 2163 Apr 17 09:23 mat_trans_alt.f90
-rw-r--r-- 1 lyan1 loniadmin 10300 Apr 17 09:50 mat_trans_alt.o

[lyan1@poseidon2 single_file]$ tau_f90.sh -optKeepFiles mat_trans_alt.f90
...

[lyan1@poseidon2 single_file]$ ll
total 1032
-rwxr-xr-x 1 lyan1 loniadmin 1578296 Apr 17 10:18 a.out
-rwxr-xr-x 1 lyan1 loniadmin 2163 Apr 17 09:23 mat_trans_alt.f90
-rw-r--r-- 1 lyan1 loniadmin 2493 Apr 17 10:18 mat_trans_alt.inst.f90
-rw-r--r-- 1 lyan1 loniadmin 10300 Apr 17 10:18 mat_trans_alt.o
-rw-r--r-- 1 lyan1 loniadmin 2019 Apr 17 10:18 mat_trans_alt.pdb
```
TAU Intermediate Files contd.

```fortran
[lyan1@poseidon2 single_file]$ cat mat_trans_alt.inst.f90
...
! Matrix dimension
data ndim /16,12/
character(len=*), parameter :: FMT1="(12(1x,i4))"
character(len=*), parameter :: FMT2="(16(1x,i4))"

integer profiler(2) / 0, 0 /
save profiler

call TAU_PROFILE_INIT()
call TAU_PROFILE_TIMER(profiler, ' &
& MATRIXTRANS_ALT1 [{mat_trans_alt.f90} {1,1}-{90,28}]')
call TAU_PROFILE_START(profiler)
call mpi_init(ierr)

call mpi_comm_size(mpi_comm_world,nprocs,ierr)
call mpi_comm_rank(mpi_comm_world,myrank,ierr)
...
```
Notes For Fortran Programmers

- Use “include ’mpif.h’” instead of “use mpi”
- If free format is used on .f files (fixed format is the default for .f files), use the “-optPdtF95OptsWithout='-R free'” option
- If more than one module files are used, use the “-optPdtGnuFortranParser” option
- If C preprocessor directives are used, use the “-optPreProcess” option
TAU Environment Variables

- TAU provides a number of environment variables to control its behavior
  - TAU_MAKEFILE
  - TAU_THROTTLE
  - TAU_OPTIONS
  - PROFILEDIR
  - TRACEDIR
  - COUNTER<N>
    - Specify what metric(s) to profile (again, PAPI is not available at the moment)
  - ...

LSU HPC Training Series
May 6, 2009
TAU_MAKEFILE

- Different makefiles corresponding to different configuration
- The default is Makefile.tau-icpc-mpi-pdt
- There are a few others

Makefile.tau-intel10_mvapich1.01-callpath-icpc-mpi-compensate-pdt
Makefile.tau-intel10_mvapich1.01-callpath-icpc-mpi-pdt
Makefile.tau-intel10_mvapich1.01-depthlimit-icpc-mpi-pdt
Makefile.tau-intel10_mvapich1.01-icpc-mpi-compensate-pdt
Makefile.tau-intel10_mvapich1.01-icpc-mpi-pdt
Makefile.tau-intel10_mvapich1.01-icpc-mpi-pdt-trace
Makefile.tau-intel10_mvapich1.01-icpc-pdt
Makefile.tau-intel10_mvapich1.01-icpc-pthread-pdt
TAU_OPTIONS

• Override the default instrumentation options for TAU

• Usage example
  
  • `export TAU_OPTIONS="-optVerbose -optKeepFiles -optPreProcess"`
TAU_CALLPATH

- Enables callpath profiling
  - Record callpath for each event
  - Need to set TAU_MAKEFILE to one of those with callpath in its name
- TAU_CALLPATH_DEPTH
  - Level to which callpath is recorded
  - Default is 2
  - Overhead increases with the depth of callpath
Other Environment Variables

- **PROFILEDIR**
  - Controls where the profile files go (default is current directory)

- **TAU_THROTTLE**
  - Enable event throttling
  - Purpose: reduce profiling overhead

- **TAU_THROTTLE_NUMCALLS**

- **TAU_THROTTLE_PERCALLS**
  - If a function executes more than $TAU_THROTTLE_NUMCALLS$ times and has an inclusive time per call of less than $TAU_THROTTLE_PERCALLS$ microseconds, then profiling of that function will be disabled after that threshold is reached
Exercise 2: Profile MrBayes with TAU

- Copy the source code of MrBayes from /home/lyan1/traininglab/tau and extract the files
- Compile the code
  - Change “MPI” to “yes” and “mpicc” to “tau_cc.sh” in Makefile
  - Change TAU_MAKEFILE to Makefile.tau-intel10_mvapich1.01-callpath-icpc-mpi-pdt
  - Build MrBayes (just type “make”)
- Run the code (with 4 cpus) and check the results
  - Copy mb.in and primate.nex from /home/lyan1/traininglab/tau/mb-run
  - Run “mpirun -np 4 ./mb < mb.in”
Selective Profiling

- **Instruct TAU**
  - Which part(s) of the code to profile
  - How they are profiled
- `-optTauSelectFile=<file>`
  - `BEGIN_(IN/EX)CLUDE_LIST/END_(IN/EX)CLUDE_LIST`
  - `BEGIN_FILE_(IN/EX)CLUDE_LIST/END_FILE_(IN/EX)CLUDE_LIST`
  - `BEGIN_INSTRUMENT_SECTION/END_INSTRUMENT_SECTION`
  - Wildcards (*,?,#) can be used
Selective profiling contd.

[lyan1@poseidon2 src]$ echo $TAU_OPTIONS
-optVerbose -optTauSelectFile=/work/lyan1/ClusterTest/tautest/lammps-mpi-only/src/select.tau
[lyan1@poseidon2 src]$ cat select.tau
BEGIN_INCLUDE_LIST

MPI#
mpi#
Mpi#

END_INCLUDE_LIST
Tracing

- Recording of information about events during execution
  - Entering/ exiting code region (function, loop, block...)
  - Thread/ process interactions (send/ receive message...)
- Save information in event record
  - Timestamp
  - CPU identifier
  - Event type and event-specific information
- Event trace is a time-sequenced stream of event records
Tracing with TAU

- Pick the correct TAU_MAKEFILE (those with “trace” in the file name)
- Compile with TAU compiler wrappers and execute
- Use external utilities to analyze the trace files
  - JUMPSHOT
  - VAMPIR
- Be careful: trace files can grow very big!
Not covered

- Database management
- Track memory and IO
- Instrumentation API
- ...
References

- **TAU documentation:**
  http://www.cs.uoregon.edu/research/tau/docs.php

- **ACTS website for TAU:**
  http://acts.nersc.gov/tau/index.html