Presentation

MPI profiling with TAU and Vampir

Martin Cuma
Center for High Performance Computing
University of Utah
mcuma@chpc.utah.edu

April 13, 2006

http://www.chpc.utah.edu
Overview

• What’s profiling good for
• Serial profilers
• What is TAU and Vampir.
• How to use TAU to produce traces.
• How to view traces in Vampir.
Why profiling

• Programs are often written inefficiently
• Compiler optimizations most often help, but not always
• Profilers provide timing information
  - coverage – timing collected at given intervals
  - trace – timing collected at entry/exit from a subroutine
Serial profilers

• It’s a good idea to first profile serial program, or single CPU MPI program

• GNU’s gprof
  ▪ compile with –pg and run executable
  ▪ view with gprof (–l – lines)

• PGI’s pgprof
  ▪ compile with –Mprof=func or –Mprof=lines and run executable
  ▪ view with gprof
gprof output example

Flat profile:

Each sample counts as 0.01 seconds.

% cumulative self self total
  time seconds seconds calls Ks/call  Ks/call name
43.82 1497.60 1497.60 100 0.01  0.01 ewald1_
25.96 2385.01 887.41 100 0.01  0.01 ewald0_
12.41 2809.06 424.05 100 0.00  0.00 parlink_
 3.99 2945.29 136.23 1221800 0.00  0.00 srfrce_
 3.47 3063.88 118.59 1224600 0.00  0.00 ewald2_
 2.96 3165.20 101.32 12754610 0.00  0.00 images_
...

granularity: each sample hit covers 4 byte(s) for 0.00% of 3417.78 seconds

index % time self children called name
 0.18 3402.43 1/1 __f90_main [2]
[1] 99.6  0.18 3402.43 1 MAIN__ [1]
 65.33 1775.41 100/100 forces_ [3]
 91.60  982.33 100/100 evbforce_ [5]
 424.05  0.00 100/100 parlink_ [7]
...
[7] 12.4 424.05 0.00 100 parlink_ [7]
 0.00  0.00 100/2005 dcell_ [75]
 0.00  0.00 100/12756410 invert_ [17]
 0.00  0.00 1/12754610 images_ [10]
pgprof example

- Both graphic and text output
- Serial and limited parallel profiling
TAU features

- tracing and profiling capabilities
- works with Fortran, C/C++, Java, Python
- automatic instrumentation of user code
- own visualization tool (paraprof)
- support for other visualizing programs (Vampir)
• Currently only support MPI tracing
• Use TAU compiler wrappers
• Built with ch_p4 (Ethernet) MPICH
• Can also work with other networks
• TAU traces must be converted to be read in Vampir
How to use TAU

0. Source TAU and Vampir environment
   source /uufs/arches/sys/pkg/tau/std/etc/tau.csh
   source /uufs/arches/sys/pkg/ita/std/etc/ita.csh

1. Generate instrumented binary
   ▪ create Makefile that sources TAU info
     TAUROOTDIR = uufs/arches/sys/pkg/tau/2.15
     include $(TAUROOTDIR)/include/Makefile
     F90 = $(TAU_COMPILER) pathf90
   ▪ compile
     make
   ▪ or compile directly
     tau_f90.sh source.f90 –o executable
     tau_cc.sh source.c –o executable
     tau_cxx.sh source.cpp –o executable
How to use TAU

2. Produce tracefile
   - run the instrumented binary
     `/uufs/arches/sys/pkg/mpich/std/bin/mpirun -np 4 -machinefile $PBS_NODEFILE ./executable`
   - convert trace files to Vampir format
     `tau_merge tautrace.*.trc myprogram.trc`  
     optionally add `–n` to break a stuck session
     `tau2vtf myprogram.trc tau.edf myprogram.vtf`

3. Analyze tracefile with Vampir (Intel TraceAnalyzer)
   `traceanalyzer test.stf`
   - look at global statistics
   - identify interesting application phases
   - investigate these in depth
Example 1 - Numerical integration

\[
\int_a^b f(x) \approx \sum_{i=1}^{n} \frac{1}{2} h \left[ f(x_{i-1}) + f(x_i) \right] = \frac{1}{2} h \left[ f(x_0) + f(x_n) \right] + \sum_{i=1}^{n-1} h \left[ f(x_i) \right]
\]
Example 1 – compile and run

• 1. Compile
  
  `tau_cc.sh trapp2.c -o trapp2_tau`

• 2. Run in PBS (script)
  
  ```
  #PBS -l nodes=4,walltime=0:10:00
  cd ~/talks/Vampir
  /uufs/arches/sy.../machinefile $PBS_NODEFILE ./trapp2_tau > trapp.out
  ```

• 3. Analyze tracefile in Vampir
  
  ```
  tau_merge tautrace.*.trc trapp2_tau.trc
tau2vtf trapp2_tau.trc tau.edf trapp2_tau.vtf
  traceanalyzer trapp2_tau.vtf
  ```
Poisson method overview

• Solve Laplace eqn. by discretization on 2D grid

\[
\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \varphi(x, y) = \nabla^2 \varphi_{x,y}
\]

\[
\approx \frac{1}{4h^2} \left( \varphi_{x+1,y} + \varphi_{x-1,y} + \varphi_{x,y+1} + \varphi_{x,y-1} - 4\varphi_{x,y} \right)
\]

• Algorithm – Jacobi iteration

while (diff > treshold)
    diff = 0;
    for(i,j)
        unew[i][j] = \frac{1}{4} (u[i+1][j] + u[i-1][j] + u[i][j+1] + u[i][j-1])
        \quad - h*h*f(i,j);
        diff += (xnew[i][j]-x[i][j])*(xnew[i][j]-x[i][j]);
    diff = sqrt(diff);
• Tracefile loading can be interrupted and resumed
• Tracefile can be loaded starting at specified time offset
ITA displays

- **Global displays** – all selected processes
  - Timeline – detailed application trace over time
  - Summary chart – aggregate profiling information
  - Activity chart – per-process profiling information
  - Communication statistics – point-to-point operations
  - Global comm. statistics – collective operations
  - I/O statistics – MPI I/O operations
  - Calling tree – global or local

- **Process displays** – single process per window
  - Timeline
  - Calling tree
  - Activity chart
ITA – timeline display

- Can zoom in time, filter messages, processes, customize the view,...
- Parallelism view

4/13/2006  http://www.chpc.utah.edu  Slide 16
• Aggregated profiling info (time, no. of calls)
• Can include/exclude called routines
• Can compare different tracefiles
• Per process profiling info
ITA – call tree

- Routine call tree display
- Can be folded/unfolded
• Data communicated between the processes
• Can filter, zoom and display in different format
ITA – global op. statistics

- Global communication data
- Can filter, zoom and display in different format

4/13/2006  http://www.chpc.utah.edu  Slide 21
ITA – process view

- Activity on each process at given time
- Can animate, filter processes, customize layout

Some useful resources

• **TAU webpage**
  
  http://www.cs.uoregon.edu/research/tau/home.php

• **Location of TAU/ITA on Arches**
  
  /uufs/arches/sys/pkg/tau
  /uufs/arches/sys/pkg/ita

• **Documentation**
  
  http://www.cs.uoregon.edu/research/tau/docs.php
  /uufs/delicatearch/sys/pkg/ita/std/doc

  http://www.chpc.utah.edu/short_courses/TAU/ITA
MPE - Jumpshot

• MPE
  ▪ library shipped with MPI
  ▪ includes tracing, error handling and process visualization routines

• Jumpshot
  ▪ trace file viewer shipped with MPICH
  ▪ not as user-friendly as Vampir
  ▪ viewer written Java, must have Java SDK to compile
  ▪ very slow and produces bulky tracefiles
  ▪ BUT - free