Lecture: Advanced Parallel Computing
Topic: Profiling Serial Code Using Gprof

Mary Thomas
Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 02/01/16
Updated: 02/01/16
Table of Contents
X11 forwarding to run Gnuplot on tuckoo

- Using X11 forwarding will allow you to visualize data or run graphical applications, e.g.:
  - ncview to view NetCDF data files or
  - ParaProf performance analyzer

- see http://www-rohan.sdsu.edu/faculty/mthomas/courses/f15/comp696/topics/tools/comp696-ssh-xterm.pdf
Setting up X11 (xterm) using SSH Forwarding

- Launch **SSH** X11 terminal on your computer
- **SSH** onto rohan (OS X):
  ```
  %ssh -Y rohanUserName@rohan.sdsu.edu
  ```
- **SSH** from rohan to tuckoo:
  ```
  %ssh -Y tuckooUserName@tuckoo.sdsu.edu
  ```
- Software required: some form of X11 window application
  - OSX: XQuartz X Window System
  - Windows: XMing or PuTTY (select Connection/SSH/X11)
- To Test, run the command `% xclock &`
The gprof utility is included in most Unix systems. It is used to profile program execution at the procedure level and profiles procedures according to their call graphs. gprof displays the following information:

- The parent of each procedure.
- An index number for each procedure.
- The percentage of CPU time taken by that procedure and all procedures it calls (the calling tree).
- A breakdown of time used by the procedure and its descendents.
- The number of times the procedure was called.
- The direct descendents of each procedure.
We can use profiling applications to analyze the program call tree and obtain some timings. How closely do our results agree?

PROFILING: using \texttt{−p} option in make

```
[mthomas@tuckoo ch2]$ mpicc \texttt{−p} \texttt{−o} histodat histodat.c
[mthomas@tuckoo ch2]$ ./histodat 10 1 1000 1000000
T\textunderscore wall in seconds: 0.107674 seconds

T\textunderscore whichbin in seconds: 0.085712 seconds
```

```
[mthomas@tuckoo ch2]$ gprof histodat gmon.out
\begin{tabular}{cccclll}
% & cumulative & self & calls & self & total & name \\
\hline
75.19 & 0.06 & 0.06 & 1000000 & 0.00 & 0.00 & Which\_bin \\
12.53 & 0.07 & 0.01 & 1 & 10.03 & 10.03 & Gen\_data \\
12.53 & 0.08 & 0.01 & 1 & 0.00 & 0.00 & Gen\_bins \\
0.00 & 0.08 & 0.00 & 1 & 0.00 & 0.00 & Get\_args \\
0.00 & 0.08 & 0.00 & 1 & 0.00 & 0.00 & \\
\end{tabular}
```

\texttt{GPROF} says that 75\% of the time is spent in Which\_bin, for 0.06 seconds.
Using our \texttt{T\_wall}, we measured .086 seconds

Which approach is correct? GNU Profile: \url{https://www.cs.utah.edu/dept/old/texinfo/as/gprof.html}
Code Example: run job from command line

[mthomas@tuckoo]$ cat makefile

MAKE FILE

MPICC = mpicc
CC = gcc

wave-dyn: wave-dyn.c
$(MPICC) -pg -o wave-dyn wave-dyn.c

RUN FROM COMMAND LINE

[mthomas@tuckoo]$ mpirun -np 9 ./wave-dyn-mpt 900 s c 0.5 0.25 4 ...

PROFILING: using -p option in make

[mthomas@tuckoo]$ gprof wave-dyn gmon.out

Flat profile:
Each sample counts as 0.01 seconds.

cumulative self self total
time seconds seconds calls s/call s/call name
...

### Call graph generated by Gprof

Granularity: Each sample hit covers 2 byte(s) for 0.83% of 1.21 seconds

<table>
<thead>
<tr>
<th>Index</th>
<th>% Time</th>
<th>Self</th>
<th>Children</th>
<th>Called</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.0</td>
<td>1.21</td>
<td>0.00</td>
<td>3/3</td>
<td>Fox [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.21</td>
<td>0.00</td>
<td>3</td>
<td>Local_matrix_multiply [1]</td>
</tr>
<tr>
<td>2</td>
<td>100.0</td>
<td>0.00</td>
<td>1.21</td>
<td>1/1</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>1.21</td>
<td>1</td>
<td>Fox [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.21</td>
<td>0.00</td>
<td>3/3</td>
<td>Local_matrix_multiply [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>Set_to_zero [8]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1/5</td>
<td>Local_matrix_allocate [4]</td>
</tr>
<tr>
<td>3</td>
<td>100.0</td>
<td>0.00</td>
<td>1.21</td>
<td>1/1</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>1.21</td>
<td>1</td>
<td>Fox [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>4/5</td>
<td>Local_matrix_allocate [4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>2/2</td>
<td>Calc_matrix [5]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>Setup_grid [9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>Build_matrix_type [6]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>Print_matrix [7]</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>5</td>
<td>Fox [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>4/5</td>
<td>main [3]</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>2/2</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>2</td>
<td>Calc_matrix [5]</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>Build_matrix_type [6]</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>Print_matrix [7]</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>Fox [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>Set_to_zero [8]</td>
</tr>
<tr>
<td>9</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>main [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>Setup_grid [9]</td>
</tr>
</tbody>
</table>
Limitations of Gprof

- Slows down code performance so turn off when not needed
- Primarily designed for serial code, so does not handle parallel very well.