Performance Profiling Tools Tutorial

CS-255 Systems Programming Lab

Giannis Malliotakis – jmal@csd.uoc.gr
Giorgos Xanthakis – gxanth@csd.uoc.gr
Motivation

Let's say you've written a program
- It looks to be working
- But the performance is bad
- You want to find the underlying cause *efficiently*
- How do you do it?

Another example:
- Your server/machine is underperforming
- You want to check resource usage and running programs
- How do you do it? *(in Linux)*
A roadmap of available tools

- Many tools available, for different system components
• Display Linux processes
• Continuous output, until pressing 'q'
• Lots of functionality, press 'h' to display help
An interactive Linux process viewer

Similar to top, but with better UI, some extra features (mouse support!)
vmstat

- Virtual Memory Stat
- View system memory and processor statistics
- Configurable output frequency, count
<table>
<thead>
<tr>
<th>total</th>
<th>used</th>
<th>free</th>
<th>shared</th>
<th>buffers</th>
<th>cached</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem:</td>
<td>3938</td>
<td>2462</td>
<td>1476</td>
<td>87</td>
<td>437</td>
</tr>
<tr>
<td>-/+ buffers/cache:</td>
<td>388</td>
<td>3550</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swap:</td>
<td>3813</td>
<td>6</td>
<td>3807</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**free**
- Display system memory statistics
- Configurable output unit, report duration
mpstat

- Processor statistics
- Various processor time metrics
- Configurable CPU set to display stats
- I/O related statistics
- Displayed per device
- Configurable report frequency/duration
Profiling

- Program = Algorithms + Data Structures
- Both can be debugged to avoid errors
- But how to optimize efficiently?
- Profiling: dynamic program performance analysis
- Program analysis can examine many things:
  1. Time spent in functions (stalls/delays)
  2. Call paths and function call frequencies
  3. Memory Consumption
GNU Profiler (Not a debugger!)

Compile (and link) your program with `-pg`

Run your program as you would normally

Once your program exits there should be a `gmon.out` file

Run `gprof <gprof options> <your executable> gmon.out` to perform profiling

Common `gprof` options:

1. `-p`: flat profile, shows the time your program spent executing each function
2. `-q`: call graph analysis, view function calls in a tree-like manner
### gprof output

```bash
gprof output
```

```
mallias@ArchCorsair ~/Documents/HY225/EX04 % gprof -p -b sudoou gaom.out
flat profile:
```

```bash
each sample counts as 0.01 seconds.
% cumulative self self total name
% seconds seconds calls ms/call ms/call
60.73 0.18 0.18 4841 0.04 0.05 sudoku_init_choices
7.41 0.20 0.02 2087055 0.00 0.00 grid_clear_choice
7.41 0.22 0.02 456892 0.00 0.00 grid_choice_is_valid
7.41 0.24 0.02 4841 0.00 0.00 sudoku_choice_next
7.41 0.26 0.02 4788 0.00 0.00 sudoku_is_solvable
3.71 0.37 0.01 4942 0.00 0.00 sudoku_is_correct
0.00 0.27 0.00 1647725 0.00 0.00 grid_read_choice
0.00 0.27 0.00 1289593 0.00 0.00 grid_read_value
0.00 0.27 0.00 742673 0.00 0.00 grid_read_count
0.00 0.27 0.00 229593 0.00 0.00 grid_clear_count
0.00 0.27 0.00 123118 0.00 0.00 grid_remove_choice
0.00 0.27 0.00 447 0.00 0.00 grid_update_value
0.00 0.27 0.00 4398 0.00 0.00 sudoku_eliminate_choice
0.00 0.27 0.00 4398 0.00 0.00 sudoku_update_choice
0.00 0.27 0.00 729 0.00 0.00 grid_set_choice
0.00 0.27 0.00 454 0.00 0.00 grid_clear_unique
0.00 0.27 0.00 81 0.00 0.00 grid_set_count
0.00 0.27 0.00 2 0.00 0.00 sudoku_print
0.00 0.27 0.00 1 0.00 0.00 grid_print_unique
0.00 0.27 0.00 1 0.00 0.00 grid_set_unique
0.00 0.27 0.00 1 0.00 0.00 sudoku_read
0.00 0.27 0.00 1 0.00 0.00 sudoku_solution_is_unique
0.00 0.27 0.00 1 0.00 0.00 270.27 sudoku_solve
```

```
mallias@ArchCorsair ~/Documents/HY225/EX04 % gprof -b -q sudoou gaom.out
Call graph
granularity: each sample hit covers 2 byte(s) for 3.70% of 0.27 seconds
```
Going a level deeper: perf

- `perf` is a robust Linux profiler
- Can monitor lots of system events (use `perf list` to check them out)
- Configurable monitoring frequency
- Useful mainly for the Linux kernel (`perf_events`), but can also be used in userspace
- Userspace command interface, use `perf <perf_command> <perf_options> <your command>`
Some useful `perf` commands

- `perf stat` : obtain event counts
- `perf record` : record events for later reporting
- `perf report` : event breakdown
- `perf top` : live event count
```
Performance counter stats for './sudoku':

<table>
<thead>
<tr>
<th>Counter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-clock:u</td>
<td>638.81 ms</td>
<td></td>
</tr>
<tr>
<td>context-switches:u</td>
<td>0</td>
<td>0.000 CPUs utilized</td>
</tr>
<tr>
<td>cpu-migrations:u</td>
<td>245</td>
<td>0.000 K/sec</td>
</tr>
<tr>
<td>page-faults:u</td>
<td>2,240,081,999</td>
<td>0.388 K/sec</td>
</tr>
<tr>
<td>cycles:u</td>
<td>1,854,715,553</td>
<td>3.952 GHz</td>
</tr>
<tr>
<td>instructions:u</td>
<td>321,735,923</td>
<td>510.034 M/sec</td>
</tr>
<tr>
<td>branches:u</td>
<td>2,293,988</td>
<td>0.71% of all branches</td>
</tr>
<tr>
<td>branch-misses:u</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Has a unique solution: 0
```
perf record and report

mallias@ArchCorsair ~/Documents/HY255/EX04 % perf record -F 99 -g ./sudoku < puzzles/hard.txt

8 0 0 0 0 0 0
0 0 3 0 0 0 0
0 7 0 0 0 7 0
0 5 0 0 0 7 0
0 0 0 0 0 0 7
0 0 0 0 0 0 0
0 0 1 0 0 0 0 8
0 0 8 5 0 0 0 1 0
9 0 0 0 0 0 0 4 0 0

8 1 2 7 5 3 0 6 4 9
9 4 3 6 0 2 1 7 5
6 7 5 4 9 1 2 8 3
1 5 4 2 3 7 8 9 6
3 4 9 8 4 5 7 2 1
8 5 7 1 0 9 5 3 4
5 2 1 9 7 4 3 6 8
4 3 8 5 2 0 9 1 7
7 9 6 3 1 8 4 5 2

Has a unique solution: 0
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.029 MB perf.data (97 samples) ]

Tip: For memory address profiling, try: perf mem record / perf mem report
Flame Graphs

- Perf output can be very long (especially for large programs)
- This is where flame graphs come in
- Visualize function call stacks (and other data) efficiently
- Interactive svg output
Sudoku solve sample CPU
flame graph
Check the man pages for tool details/options (top, htop, vmstat, free, iostat, mpstat, perf)

Performance analysis in 60 seconds: https://netflixtechblog.com/linux-performance-analysis-in-60-000-milliseconds-accc10403c55

gprof documentation: https://sourceware.org/binutils/docs/gprof/
gprof tutorial: https://linoxide.com/gprof-performance-analysis-programs/
perf wiki: https://perf.wiki.kernel.org/index.php/Main_Page
Brendan Gregg's perf examples: http://brendangregg.com/perf.html
Flame Graphs: http://brendangregg.com/flamegraphs.html