CS 431 / CS 531 Introduction to Performance
Portland State University Winter 2019
HW #1
Due 5pm Wednesday Jan 16

This homework is designed to be conducted on the Linux Lab machines. If you use any other machines or versions of software, please check that your materials work correctly on the Linux Lab machines, because that is where they will be graded.

Please submit your work as a .pdf OR .docx file attachment to an email. Send to karavan at pdx.edu before 5pm on Wednesday Jan 16. LINKS TO GOOGLE DOCS WILL NOT BE ACCEPTED.

0. Read the Class Syllabus (the class webpage is cs.pdx.edu/~karavan/perf).

1. We will follow these steps in the Linux Lab during our class period:
Read and Step through the examples in the Introduction to Gprof link from the class Lectures page

2. Read the gprof paper (the 2004 version):
Susan L. Graham, Peter B. Kessler, and Marshall K. McKusick. 2004. gprof: a call graph execution profiler. SIGPLAN Not. 39, 4 (April 2004), 49-57.
You can get the paper free online at acm.org.dl You must be on campus, connected by VPN, or logged into library.pdx.edu. DO NOT Pay $ for the paper.

3. Submit your answers to the following questions:

3A. gprof hands on

In this exercise you will use gprof to examine and improve a piece of code called simpleCode.c. Turn in your answer to question 4.

1. Look over the source code provided. Build and run the code. Step through the code in the debugger, examining the values of number and searchkey.
2. Evaluate the code using gprof
3. Define and try a few changes to the code to improve the performance
4. Write a short summary of what you learned about the code, what you tried and why, the results, and whether or not the results matched your expectations.

3B. Questions on the reading

1. What is the difference between their "dynamic call graph" and "static call graph"? Can you give an example of something that would not appear in a dynamic call graph but would appear in a static call graph?
2. [old paper alert] What is a "time-sharing" system? We don't use this term much anymore, but we use time-sharing systems all the time. What is an example of one you use at PSU?
3. How accurate are the %time amounts reported by gprof?
4. Gprof samples the program counter – what does this contain? What is the key benefit of this method? What is an important downside to this method?

5. [From the 2003 update] What was one challenge the authors mention to using the profiler on the operating system kernel?

3C. Showcasing gprof

Develop a code in C or C++ that yields interesting results with gprof. You may use "example code 2" as a starting point. Think about what characteristics in the source code might lead to any of the following: an illustration of a case where gprof results are not quite accurate; an illustration of a difference between a longer function called a few times versus a shorter function called a large number of times; recursion; or any other gprof feature you want to highlight.

Turn in your code plus a brief description of what it shows.