CS618 Lab4 - Profiling exercises using Gprof and Perf

Deadline: September 4th, 2024 11.59PM, Individual assignment

Required source files for this lab are provided at LAB4_Profiling_gprof

Task 0: Warmup Exercise

Part A) Take a look at main/main.c to understand what this program is doing (not much, a "nop" is "no-operation", that is, sit idle for a cycle). Makefile is also provided. The time utility reports how long a program takes to run.

a) Run `make baseline`. How long did it take for the baseline to run? Given how long baseline took using time utility, how long do you expect each of the functions in the main to take from the start of each function until it finishes?
b) Now `run make overhead`. What is the difference between main_no_profiling and main_with_profiling? Which ran faster, main_no_profiling or main_with_profiling? Why?
c) Between child1 or child2, which should you remove such that main_no_profiling and main_with_profiling will have nearly the same runtime? Why remove that child?
d) Finally, `run make profile.txt`. Take a look at the profile.txt file. The gprof output has many different times it reports, such as "cumulative seconds" and "self seconds". Which timing report matches what part a) asked for?
e) How well did your estimates from part a) line up with the time reported by gprof ? What explains the difference between your estimate and the time reported by gprof ? What generated the file gmon.out?

Part B) Take a look at the main2/main2.c. When run, this program will execute 16*LONG_TIME nop instructions. By analyzing the code, what percentage of nop's should be attributed to each function?First, finish filling in the first column of this table.

Function	nop Analysis	Profile 1	Profile 2	Profile 3
main:	0%			
parent:	6.25%			
child1:				
child2:				
no_children:				

Now compile and profile the code 3 times, recording the percentage of time attributed by perf to each function. Fill in the remaining three columns of the table with the results from these profiling runs. Does the trend of time attributed to each function align with your expectations from the nop analysis? (Yes or No) Give one reasonable explanation for why you think the measured percentages from profiling don't perfectly match the nop analysis.

Task 1: Matrix chain multiplication

You are provided with a c code matchain.c that contains functions for four different implementations of Matrix chain multiplication (multiply a given sequence of matrices). The problem is not actually to perform the multiplications, but to decide the sequence of the matrix multiplications involved and then multiplying matrices obtained in the correct order.

Firstly, you need to understand the code, modify, comment and uncomment the appropriate lines so as to compile and build the executable for each version of the implementation. You are also provided with five different input files.

Usage: matchain_exe < input.txt

Task 2 : Edge Detection

You are provided with implementation of Sobel and Canny's Edge Detection algorithms in C++. You are provided with 5 different input files. Compile and build the code.

Usage :
./canny <name_of_input_image> <high_threshold> <sigma_value>" . A typical value to use as your "high_threshold" would be ~100, and a typical value for "sigma" would be 1.
./sobel <name_of_input_image> <high_threshold> <low_threshold>". A typical value to use as your "high_threshold" would be ~100, and a typical value for "low_threshold" would be 35.

For both of the above tasks,

Your need to use gprof and perf and analyze the various implementations for various inputs by comparing and analyzing the outputs of gprof and perf.

For perf, you first need to figure out what events are supported by the system that you are using and what events out of the supported events are useful and applicable for comparing the implementations.
Also what other metrics you will use to do the comparison

Provide in your report, the outputs, your analysis along with tables, comparisons, reasons and any other details that you deem fit.

Submission instructions:

1. Create a new branch called Lab4
2. For each task make a separate directory in lab4 named as taskx (x is the number of the task eg. task1 task2)
3. Inside the directory corresponding to each task, populate the required files as mentioned in the handout
4. Make a report and provide the required details corresponding to each task and upload it in the lab 4 folder. The report should be named as lab3_rollno (rollno is of the format cs24mtxx)