Gprof Profiler

-By Pallav & Vivek

What is profiling?

- Profiling (program/software profiling) is an important aspect of software engineering.

- Profiling gives us a detailed feedback of way the programs are executed.

- It is a form of **dynamic program analysis** that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or frequency and duration of function calls.

...

- Through profiling one can determine the parts in code that are time consuming and need to be re-written.

Gprof

gprof is an abbreviation for GNU Profiler, used to profile programs/pieces of code (C, C++, fortran etc.).

It is a performance analyzing tool, originally developed by a group from UCB for Berkley Unix 4 and improved by Jay Fenlason (GNU project for GNU Binutils).

It is built-in into the gcc compiler.

Advantage

- The information collected as the result of profiling enables analysis of programs which are too large and/or complex to analyze by reading the source.

How it works?

Instrumentation code is automatically inserted into program code during the compilation phase (when we run the program with “-pg” of the gcc compiler) to gather caller-function data.

At run-time, the instrumentation code helps gather caller-function data and the sampling information is saved in ‘gmon.out’. It can be analysed with the ‘gprof’ tool.

...

The gprof output consists of two parts:

(i) The flat profile : The flat profile gives the absolute execution time spent in each function and its percentage of the total running time, and also the no. of instances of the function being called.

(ii) The call graph : This is the textual representation of the call graph. It displays the parent (the function that called it) and the child (the function that is called by it) of each function.

Shortcomings

- Results of profiling are dependent on the way the program is run. Results of profiling gives a measure of the execution time of the functions/code-segments that are actually run/invoked while executing the program. Parts of the program that are not run and features which remain unused don't appear in the profiling information. This is so because gprof collects data during the run-time of the program. Hence, each instance of program execution may show different results each time.

...

- gprof cannot measure kernel time (syscalls or I/O waiting), and only user-space code is profiled.
- The run-time figures, that gprof gives you, are based on a sampling process, so they are subject to statistical inaccuracy. If a function runs only a small amount of time, so that on the average the sampling process ought to catch that function in the act only once, there is a pretty good chance it will actually find that function zero times, or twice.

Using gprof

In terminal command line:

- *$ g++ -pg -o <output_file_name> <program_name.cpp>*
- *$./<output_file_name>*
- *$gprof [options] <output_file_name> gmon.out [>output_file]*

Options for running gprof

1. -e <function_name>

The `-e function' option tells gprof to not print information about the function <function_name> (and its children...) in the call graph. The function will still be listed as a child of any functions that call it, but its index number will be shown as `[not printed]'. More than one `-e' option may be given; only one function_name may be indicated with each `-e' option.

...

2. -E <function_name>

The -E function option works like the -e option, but time spent in the function (and children who were not called from anywhere else), will not be used to compute the percentages-of-time for the call graph. More than one `-E' option may be given; only one <function_name> may be indicated with each `-E' option.

...

3. -f <function_name>

The `-f function' option causes gprof to limit the call graph to the function <function_name> and its children (and their children...). More than one `-f' option may be given; only one <function_name> may be indicated with each `-f' option.

...

4. -F <function_name>

The `-F function' option works like the -f option, but only time spent in the function and its children (and their children...) will be used to determine total-time and percentages-of-time for the call graph. More than one `-F' option may be given; only one <function_name> may be indicated with each `-F' option. The `-F' option overrides the `-E' option.

...

5. -k from... to...

The `-k' option allows you to delete from the profile any arcs from routine *from* to routine *to*.

...

6. -v

The `-v' flag causes gprof to print the current version number, and then exit.

...

7. -z

If you give the `-z' option, gprof will mention all functions in the flat profile, even those that were never called, and that had no time spent in them. This is useful in conjunction with the `-c' option for discovering which routines were never called.

References:

Further reading:

http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html

Video :

https://www.youtube.com/watch?v=lYl2h-dJcrA&list=PLFEC7739448CD4884

THANK YOU :)