SPEC Benchmark
25 March, 2019

The goal of this assignment is to understand the basics of SPEC intrate benchmark and how this benchmark can be used for evaluation purposes. This assignment also focuses on how to profile code and use that profiling information to develop a better understanding of the code.

SPEC Setup

- Download the SPEC benchmark by running the following command:
  scp aos@192.168.1.161:cpu2017-1_0_5.iso .
  password: aos
  This will download the ISO image in the current folder.
- To install CPUSPEC2017, follow the steps below:
  1) Mount the ISO image by right-clicking on the .iso file and choosing “Open with Disk Image Mounter”.
  2) Open a terminal in the mounted file system and type ./install.sh -d path_to_destination_directory.
  3) Open a terminal in the directory where CPUSPEC2017 has been installed.
  4) Run “source shrc” or “source cshrc”. SPEC is now set up.

Downloading llvm-clang

- Run the following commands to install the compiled packages of llvm and clang:
  sudo apt-get install llvm
  sudo apt-get install clang

Run the benchmarks using gcc

- To run the benchmarks using gcc, you need to select a config file present in the config folder of CPUSPEC2017.
- Create a copy of the example config file which can run with gcc, x86 architecture and linux. The file will most likely be named “Example-gcc-linux-x86.cfg” by default.
- You need to make the following changes in the copy of the config file created:
  1) Modify the label name to in the label section.
     %define label test1
  2) Find the number of cores present in your system by running nproc on the terminal.
  3) In the Preprocessor section, set the value of build_ncpus to the number of cores present in your system.
     % define build_ncpus 4
  4) Go to the Compilers section and set the path to the location where your gcc binaries are present. For example,
     % define gcc_dir /usr/bin
     preENV_LD_LIBRARY_PATH = %{gcc_dir}/gcc
     SPECLANG = %{gcc_dir}/

```
CC                  = $(SPECLANG)gcc   -std=c99   %{model}
CXX                 = $(SPECLANG)g++  -std=c++03 %{model}
```

- Run the individual C benchmarks present in the intrate test-suite by running the following command:
  runcpu --config=name_of_your_config_file --size=test --copies=1 --noreportable --iterations=1 500.perlbench_r/502.gcc_r/505.mcf_r/525.x264_r/557.xz_r
- Note the execution time taken by all of the C benchmarks present in the intrate test-suite using gcc compiler.

Run the benchmarks using clang

- To run the benchmarks using clang, you need to select a config file present in the config folder of CPUSPEC2017.
- Create a copy of the example config file which can run with clang-llvm, x86 architecture and linux. For example, Example-clang-llvm-linux-x86.cfg.
- You need to do the following changes to the copy of the config file created:
  1) Modify the label name to something in the label section.
     %define label test1
  2) Find the number of cores present in your system by running nproc on the terminal.
  3) In the Preprocessor section, set the value of build_ncpus to the number of cores present in your system.
     % define build_ncpus 4
  4) Go to the Compilers section, set the path to the location where your llvm-clang binaries are present. For example,

```
%  define  gcc_dir     /usr/bin
BASE_DIR              = /home/Desktop
LLVM_PATH             = $[BASE_DIR]/Softwares
LLVM_ROOT_PATH  = $[LLVM_PATH]/llvm.build
LLVM_BIN_PATH      = $[LLVM_ROOT_PATH]/bin
LLVM_LIB_PATH      = $[LLVM_ROOT_PATH]/lib
LLVM_INCLUDE_PATH  = $[LLVM_ROOT_PATH]/include
```

- Run the individual C benchmarks present in the intrate test-suite by running the following command:
  runcpu --config=name_of_your_config_file --size=test --copies=1 --noreportable --iterations=1 500.perlbench_r/502.gcc_r/505.mcf_r/525.x264_r/557.xz_r
- Note the execution time taken by all of the C benchmarks present in the intrate test-suite using clang compiler.

Plot the graph of gcc vs. clang

- Plot a double bar graph of execution times taken by all the C benchmarks of intrate test-suite (y-axis) for the gcc and clang compilers (x-axis).
- Compare the execution times of both the compilers for C benchmarks of intrate test-suite.

Run the benchmarks using gprof

- To get information about the execution times of individual routines, use gprof to profile the chosen benchmark.
- While running the benchmarks, profiling information can be generated. However, the gprof command needs to be run manually.
- To use gprof with gcc, gmon.out file is generated. It contains the profiling information. To generate this file, go to Baseline Tuning Flags section in gcc config file used previously. Change the value of OPTIMIZE from
  "-g -O3 -march=native -fno-unsafe-math-optimizations -fno-tree-loop-vectorize"
  to
  "-g -O3 -march=native -fno-unsafe-math-optimizations -fno-tree-loop-vectorize -Wall -pg -no-pie".
  Run the individual benchmarks using the command specified in the previous section. When the execution completes, you can find the gmon.out file in the run->run_base_something folder of the benchmark (example, CPUSPEC->benchspec->CPU->557.xz_r). Open the terminal in run->run_base_something folder of the benchmark. Run the following command in the terminal:
  gprof name_of_the_base_executable_file gmon.out (output on terminal)
  or
  gprof name_of_the_base_executable_file gmon.out > file1.txt (output in file1.txt)
  Executable file can also be found in the run->run_base_something folder of every benchmark.
  You can find the output of the command either on the terminal or in the file (created in the same folder) depending on the command that was run.
- To use gprof with clang, gmon.out file is generated. It contains profiling information. To generate this file, go to Baseline Tuning Flags section in the clang-llvm config file used previously. Change the value of COPTIMIZE from
  "-O3 -mavx"
  to
  "-O3 -mavx -Wall -pg".
  Run the individual benchmarks using the command specified in the previous section. When the execution completes, you can find the gmon.out file in the run->run_base_something folder of the benchmark (example, CPUSPEC->benchspec->CPU->557.xz_r). Open the terminal in run->run_base_something folder of the benchmark. Run the following command in the terminal:
  gprof name_of_the_base_executable_file gmon.out (output on terminal)
  or
  gprof name_of_the_base_executable_file gmon.out > file1.txt (output in file1.txt)
  Executable file can also be found in the run->run_base_something folder of every benchmark.

You can find the output of the command either on the terminal or in the file (created in the same folder) depending on command which was run.

Plot the graph of the routines

- Identify all the routines taking more than 5% of the total execution time for a specific benchmark in the intrate test-suite. The required information can be found in the flat profile section of the output resulted using gprof.
- Plot a bar graph between the identified routines (x-axis) and the percentage of execution time (y-axis) for every C benchmark of the intrate test-suite. This exercise needs to be done for both the compilers (gcc and clang).

Submit a report which consists of the following:

1) A double bar graph plotting the performance of the benchmarks for both the compilers as described in section “Plot the graph of gcc vs. clang” of the current document.
2) 10 bar graphs plotting the percentage of execution times consumed for the routines whose percentage is greater than 5. Graphs should be plotted for the results of both the compilers for every benchmark (There are five C benchmarks in total).
3) Briefly write your interpretation of the results.

The report should be a pdf file. The naming convention to be followed is HW11_RollNo_Name.pdf.

NOTE: The weightage of the HW is 3 marks.

References

- https://www.spec.org/cpu2017/Docs/install-guide-unix.html
- https://www.howtoforge.com/tutorial/how-to-install-and-use-profiling-tool-gprof/
- https://stackoverflow.com/questions/42620074/gprof-produces-empty-output