HPCA E0-243

Assignment-01

Obtaining CPI Stack for Programs using Hardware Performance Counters and Linear Regression

Participants-

Deepanshu (MTech Coursework CSA), **Serial No.** - 04-04-00-10-42-20-1-17981
deepanshu1@iisc.ac.in

Rohit Patel (MTech Coursework CSA), **Serial No.** - 04-04-00-10-42-20-1-18139
rohitpatel@iisc.ac.in

Introduction –

In this assignment, we are required to obtain the CPI(Cycles per Instruction) for different SPEC benchmark programs using hardware performance monitoring counters. We use different miss events in our experiment and these are branch-load-misses,branch-misses,L1-dcache-load-misses,L1-icache-load-misses,dTLB-load-misses,iTLB-load-misses,cache-misses. We collected the data from the hardware counters using perf tool on a linux environment and used this data to train our regression model for different SPEC benchmark programs. For each program-input pair, we built a separate regression model which gives the CPI stack for that program and evaluate them using different statistics like RMSE, R^2 , adjusted R^2 values, Residuals, F-statistic and p-value. We build the CPI stacks for three SPEC INT benchmark programs and three SPEC FP benchmark programs.

Experiment –

Tools Used –

Perf – It is a performance monitoring tool which is used to obtain the values for hardware performance monitoring counters that are present in the architecture.

Operating System – We have used linux operating system in which we installed perf.

How did we perform the experiment-

First of all we collected the few thousands data points for each spec benchmark program using perf.

perf command used to collect the data-

perf stat –d -I 100 -e branch-load-misses:u,branch-misses:u,L1-dcache-load-misses:u,L1-icache-load-misses:u,dTLB-load-misses:u,iTLB-load-misses:u,cache-misses:u,cycles:u,instructions:u bash run.sh 2>data_test_namd.txt

Normalization - We normalized the feature values in our train data so that individual feature values are always between 0 and 1.

We made the regression model using python for different spec benchmark programs and using them we build the CPI stacks for different SPEC benchmark programs and their results are given below.

Results for our experiments on different spec benchmark programs-

SPEC INT Benchmark Programs-

1-Leela_r

Interval used to collect the performance counter values 30ms-

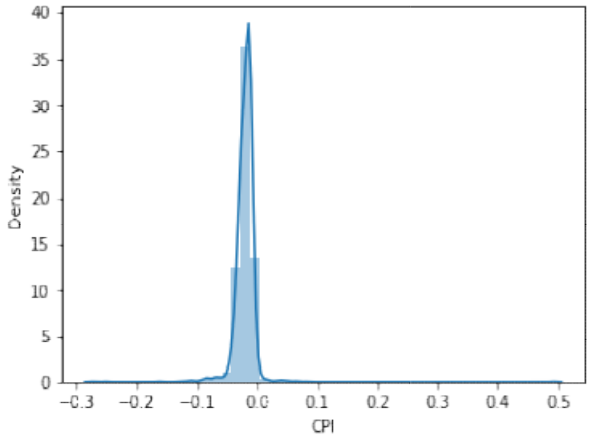
Coefficients of our linear regression model-

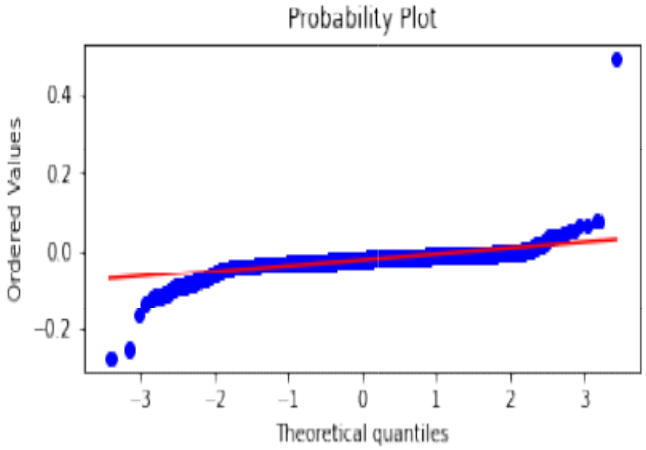
Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0.0	0.2584	0.1102	0.0909	0.0	0.0	0.1432	0.5083

Statistical values which assess the quality of our model

RMSE	R^2	Adjusted R^2	F-statictic
0.0289	0.501	0.501	524.2

Residual Graphs





2- deepsjeng_r

Interval used to collect the performance counter values 100ms-

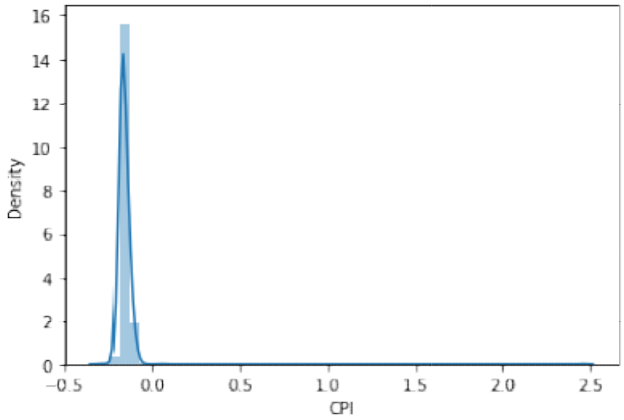
Coefficients of our linear regression model-

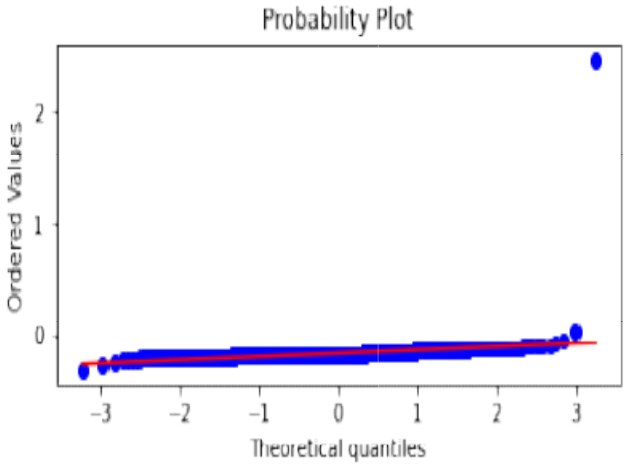
Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0.0912	0.0	0.0925	0.0062	0.0	0.3298	0.0200	0.4723

Statistical values which assess the quality of our model

RMSE	R^2	Adjusted R^2	F-statictic
0.0486	0.778	0.775	308.8

Residual Graphs





3- Omnetpp_r

Interval used to collect the performance counter values -300ms

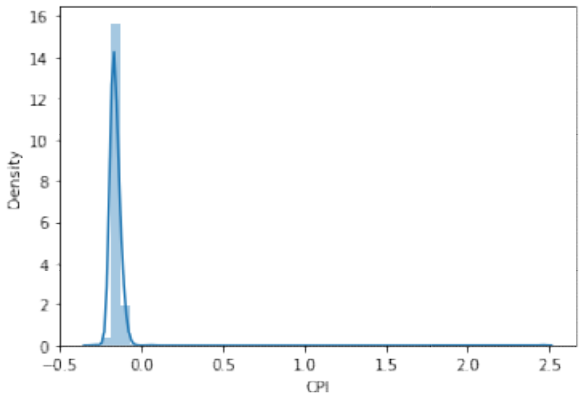
Coefficients of our linear regression model-

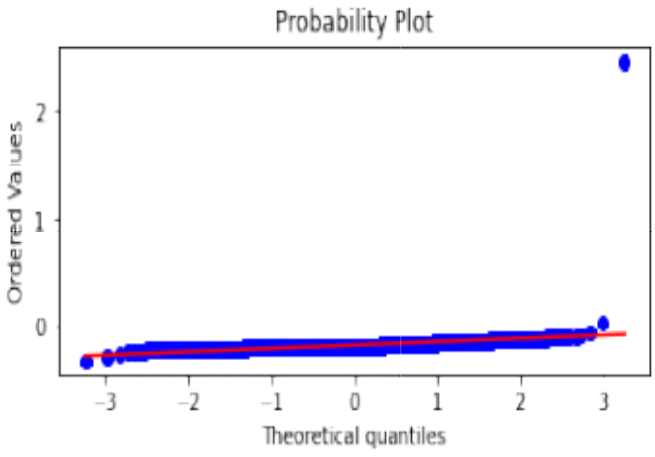
Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0	0	0	0.3513	0.6709	0.05917	0.9077	0.10028

Statistical values which assess the quality of our model

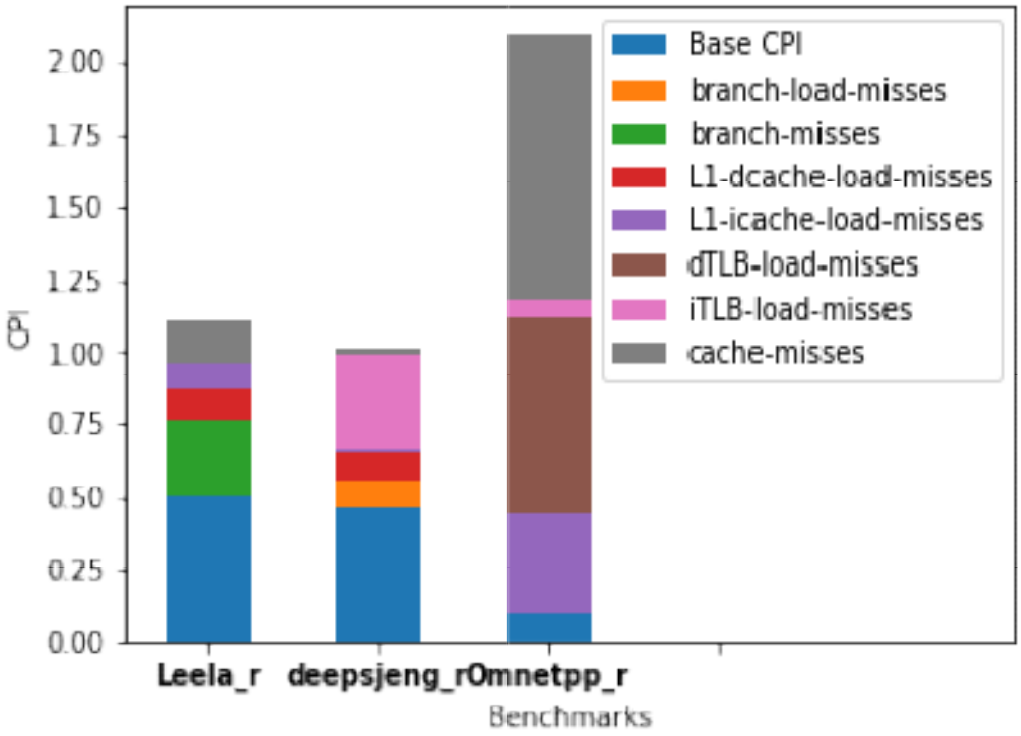
RMSE	R^2	Adjusted R^2	F-statictic
0.1820	0.762	0.761	935.8

Residual Graphs





CPI stacks for SPEC INT



SPEC FP Benchmark Programs-

1- cactuBSSN_r Benchmark

Interval used to collect the performance counter values- 500 msec on refrence input

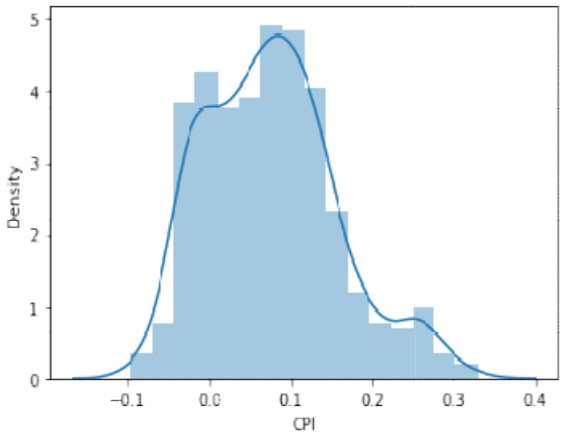
Coefficients of our linear regression model-

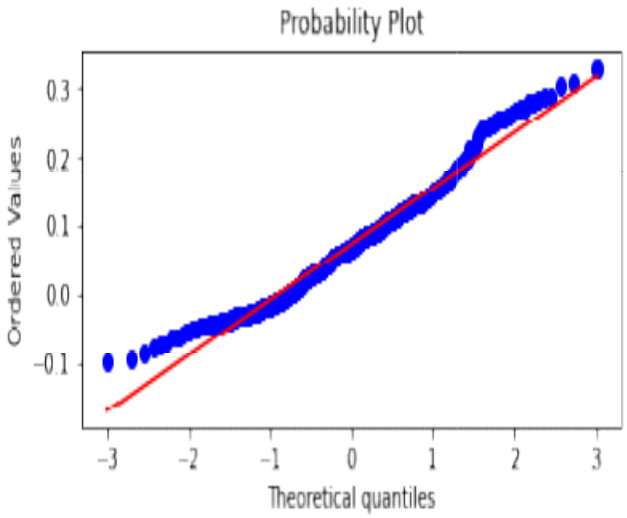
Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0.0	0.035172	0.05715	0.03082	0.0	0.00895	0.096101	0.55921

Statistical values which assess the quality of our model

RMSE	R^2	Adjusted R^2	F-statictic
0.090713	0.788	0.784	250.0

Residual Graphs





2- blender_r Benchmark

Interval used to collect the performance counter values – 300 msec on train input

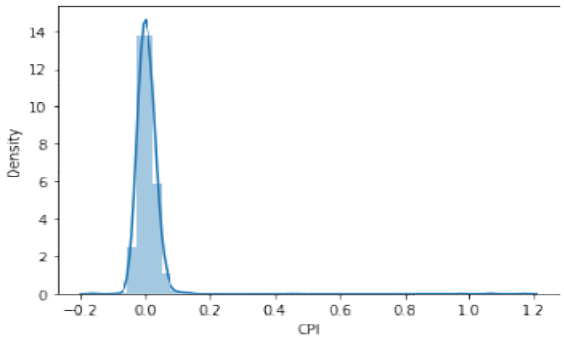
Coefficients of our linear regression model-

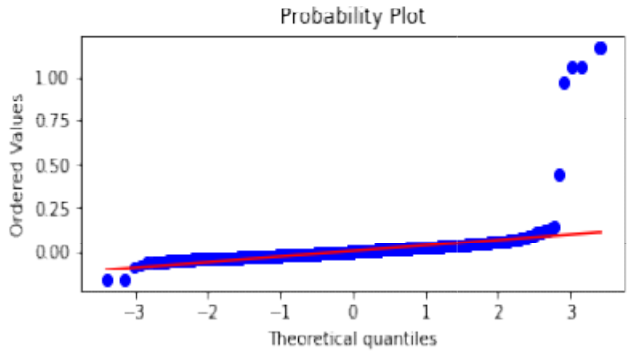
Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0.05484	0.0	0.09950	0.0	0.24393	0.13525	0.126603	0.39459

Statistical values which assess the quality of our model

RMSE	R^2	Adjusted R^2	F-statictic
0.05492	0.927	0.927	1237

Residual Graphs





3- povray_r Benchmark

Interval used to collect the performance counter values – 500 msec on reference input

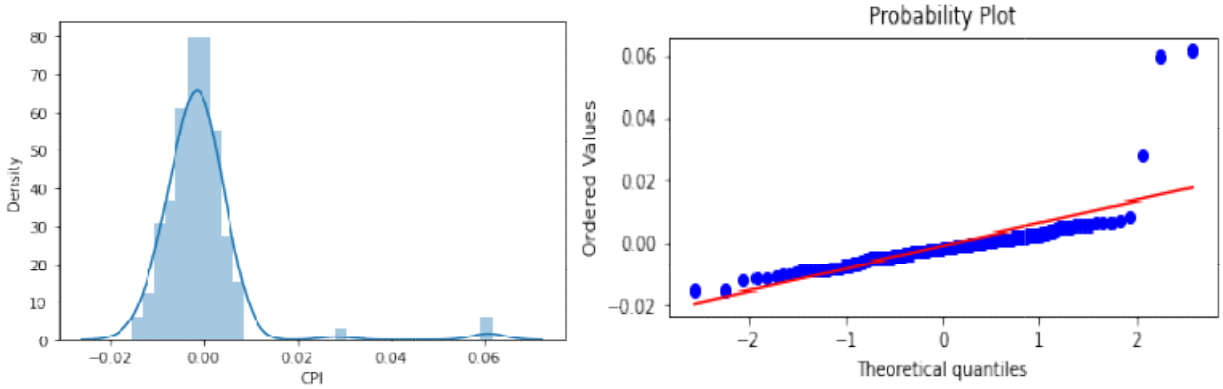
Coefficients of our linear regression model-

Branch-load-misses	Branch-misses	L1-dcache-load-misses	L1-icache-load-misses	dTLB-load-misses	iTLB-load-misses	Cache-misses	Base CPI
0.0532	0.0	0.04792	0.00915	0.0	0.01245	0.01923	0.35429

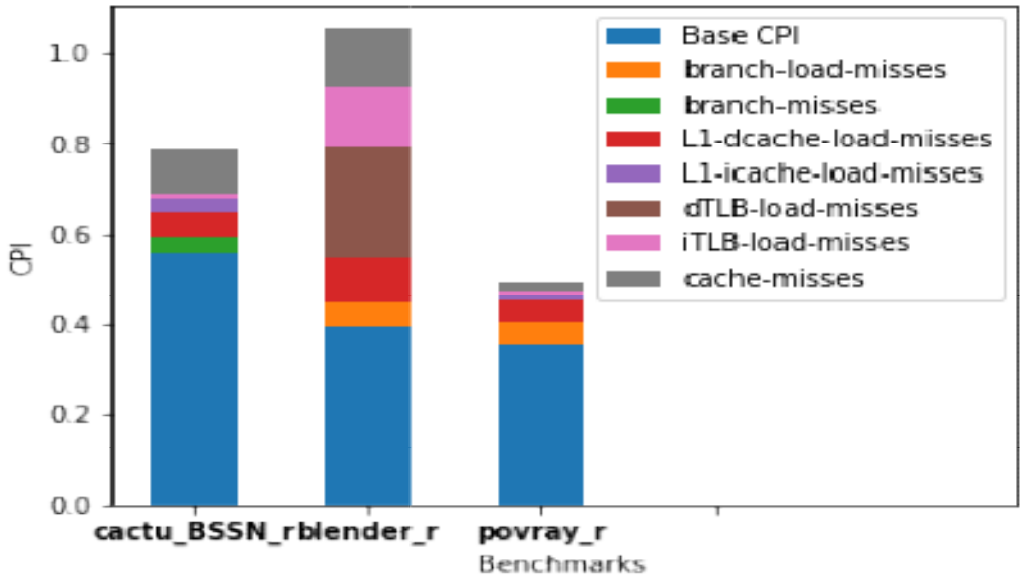
Statistical values which assess the quality of our model

RMSE	R^2	Adjusted R^2	F-statictic
0.00934	0.937	0.937	17270

Residual Graphs



CPI stacks for SPEC FP Benchmark programs -



Conclusion –

Our hardware system supports only 4 hardware counters at a time and we need to collect the data points for 7 miss events. So the data points are counted in round robin fashion and they are multiplexed as you can see in the train data text file and also some events are overlapped and because of that we were getting some negative coefficients in the regression model. So to make the coefficients non negative so that the contribution of various miss events supposed to be additive and that's why we implemented lasso linear regression model so that all the miss events which are overlapping could get forced to be 0 and other miss events can become non negative.