Documentation Details

Inside each benchmark folder we have:

1) *script.txt*:
This contains the linux terminal command used for capturing the data points separately for train set and the test set. The stderr terminal output is redirected to the corresponding file as mentioned in the command. Perf is used for collecting the data of performance counters.

2) text_to_csv_converter.py:
This python file extracts the data stored in the .txt file and stores into .csv form.

3) benchmark_r.py:
This python file contains the code to train the model based on the train dataset collected in the above steps.
This python file also contains the code used to calculate the various parameters based on the trained model and the test dataset collected in step 1.

4) benchmark_r_train.csv:
Contains the data set used for training the regression model.

5) benchmark_r_test.csv:
Contains the test data collected.

All the features used in the model are used by their names in the .csv files, e.g. "iTLB-load-misses". The column "ns" refers to the duration-time value expressed in nano secs.

PROCEDURE

- The CPI stacks for six SPEC 2017 benchmarks are required to be obtained. We have chosen 3 SPECINT benchmarks and 3 SPECFP benchmarks for experiment. The benchmarks are run in our laptop using the train/test input data provided.
- Processor used is intel i7 (7th gen) and operating system used is Linux (Ubuntu).
- The hardware performance monitoring counters are read using perf performance monitoring tool .
- Simple SGDRegression model is used to train the dataset and obtain the co-efficients(indicative of CPI stack). SGDRegression model is used to maintain non-negetive values of the co-efficients.
- The training dataset is obtained by collecting data points by running the “Train input” sets provided , while test dataset is obtained by running the “Test input” sets provided for the corresponding benchmark programs.
- The training dataset has at least 1500 data points, while the test dataset has at least 250 data points.
- Miss events taken into consideration are : L1-dcache-load-misses,iTLB-load-misses,L1-icache-load-misses,LLC-store-misses,branch-misses,LLC-load-misses,dTLB-load-misses,dTLB-store-misses,L2-store-misses and L2-load-misses. Co-efficients obtained as a result of training, estimates

the contribution of the corresponding single miss event to the total execution time.

• The data points are collected in the user space only.

• Effectiveness of each model is checked on the test dataset of the corresponding benchmarks. RMSE, R2 , adjusted R2 values, Residuals, F-statistic and p-value are captured on the test dataset to assess the quality of the model generated.

• The data points are normalised using MinMax Normalisation technique so as to make the feature values lie in 0-1 range. MinMax Normalisation formulation : (Feature – Feature.minvalue())/(Feature.maxvalue() - Feature.minvalue()).

• To avoid over-fitting of data, 20% of the training dataset chosen randomly (for each benchmark) , is used as cross-validation set.

• The following SPECINT benchmark programs are used :

1. 557.xz_r
2. 520.omnetpp_r
3. 531.deepsjeng_r

• The following SPECFP bvenchmark programs are used:

1. 511.povray_r
2. 507.cactuBSSN_r
3. 508.namd_r

• Python libraries such as sklearn,numpy and pandas are used to implement and obtain results from regression models used in the assignment.

• The data points, for each program-input pair, are collected at various intervals ranging from 300 ms to 6000 ms, for training the regression model.

• The duration of interval is used as target variable in regression model. This is indicative of the clocks consumed by the program while running in that interval.

- The coefficients generated out of the regression model is indicative of the cpi stack corresponding to each SPEC Benchmark.