Progress Report on Computational Grid CPU Burst Time Estimation Using Machine Learning Algorithms

Aseem Sharma
PID: 6329681

Ashfaq Ali Shafin
PID: 6265743

Kratin Tiwari
PID: 6329698

1. The Problem

Our main goal is to predict the CPU burst time of a process before execution. Many process scheduling algorithms like Shortest Job First (SJF), Shortest Remaining Job First (SRJF), and modified Round Robin (RR) scheduling need the CPU burst time in advance to operate properly. One approach to predict process time is exponential averaging, which depends on the processing time of the past processes. This approach performs poorly in multiple user systems as different users have different process times due to different usage. A machine learning-based approach was applied in previous research where only 5,000 jobs were included to predict burst time. We have included a dataset with 414,176 jobs to analyze the problem and make a robust solution.

2. The Solution

The solution that we will present in this project is an extension of another research. As stated above we will be including more jobs to apply our machine learning algorithms on than the original research. This will give a better analysis for finding the CPU burst times. For this solution, primarily, we will be proceeding with data pre-processing. The dataset 'GWA-T-4 AuverGrid' contains 414,176 jobs that were not included in the original research and this is what makes our solution unique and challenging to do. This will help us answer several questions such as the variation in performance when all the jobs are included as input data. Another question that we would be able to answer is how different machine learning algorithms perform and how they differentiate from each other. Lastly, how the inclusion of Artificial Neural Networks helps us achieve different results than traditional Machine Learning algorithms.

When it comes to data pre-processing, prior to its practical usage, data must be preprocessed. The concept of data preprocessing is the transformation of raw data into a clean data set. Before running the method, the dataset is preprocessed to check for missing values, noisy data, and other irregularities. After data preprocessing we will move on to apply several machine learning algorithms such as XGBoost, Linear Regression, Random Forest, Decision Tree, Support Vector Machine, Gradient Boosting, K-Nearest Neighbors. For the aforementioned algorithms, we will be using several machine learning libraries. To name some of the libraries we make use of are Scikit-Learn, Pandas, NumPy, and Matplotlib.

Our approach to solve the problem is shown in figure-1 using a flow chart. We have completed all the tasks of the flow chart.

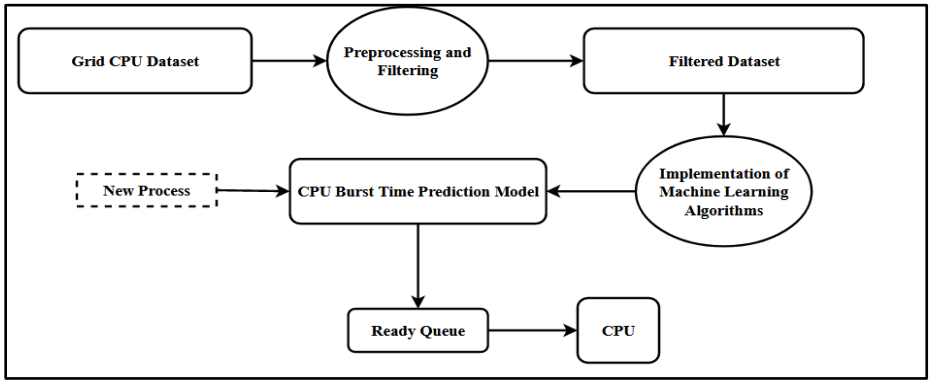

Figure 1: Flow chart of CPU Burst Time Prediction Model

3. Initial Results

Data Preprocessing: The dataset 'GWA-T-4 AuverGrid' contains 414,176 jobs with 29 columns. First, we have removed all the columns that contain only '-1' values, which indicates the value is not available. We have transformed the string-based userID and groupID into vector-based userID and groupID in order to use these columns as features. Second, we removed all the rows where the status of the job was '-1' indicating the job either failed or did not finish processing. After the preprocessing, we end up with 347,611 jobs with 33 features. We have normalized our dataset using min-max scaling to observe the effects of normalization on ML algorithms.

Data Training and Testing: We have used multi-variate machine learning algorithms to predict CPU burst time. Our approach incorporated 5-fold cross-validation for training and testing. In 5-fold cross-validation method, 80% of the data is used to train, and then the remaining 20% is used to test. We randomized the dataset before training so that the ML algorithms are prevented from biased learning order of the training set.

Preliminary Result: Table 1 shows the output results of different ML algorithms, time for training and testing for 5-fold cross-validation, and parameters of the algorithms.

Table1: Average Cross-Validation Result of Different Machine Learning Algorithms

Algorithm Name	Parameters	Time (in Seconds)	Normalization of Dataset	Correlation Coefficient
K-Nearest Neighbors	N_neighbors = 5	1934.93	False	0.84
		2070.43	True	0.896
Linear Regression	Default	1.88	False	0.74
Gradient Boosting	Default	339.93	False	0.859
Decision Tree	Max_Depth =20	15.4	False	0.891
Random Forest	Max_Depth =3	194.08	False	0.804
	Max_Depth =20	971.81	False	0.922
SVM	C=100, Max_Iteration = 10,000	4591.89	False	0.71
XGBoost	n_estimators = 100	119.34	False	0.859

4. Next Steps

We are planning to improve the preliminary result using different parameters in the above mention machine learning algorithms. For example, the normalization of data in the K-Nearest Neighbor algorithm significantly improves the R2_Score. Such improvement may be achieved with other algorithms if we examine the algorithms with different parameters. Even though we wanted to implement Artificial Neural Network (ANN) in our experiment, due to high time consumption and poor output result, we have decided to not explore further on ANN realm.

In our final draft paper, we will include all the output results, time consumed for each algorithm. Up to this point of our research, we can clearly see that the correlation coefficient (R2_Score) is around 0.90. This indicates the features in the dataset are positively co-related to CPU burst time.