A Machine Learning-Based Approach to Estimate the CPU-Burst Time for Processes in the Computational Grids

Tarek Helmy, Sadam Al-Azani, Omar Bin-Obaidellah
Department of Information and Computer Science,
College of Computer Science and Engineering,
King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia,
E-mail: [helmy,g201002580,g201201820]@kfupm.edu.sa

***Abstract* — The implementation of CPU-Scheduling algorithms such as Shortest-Job-First (SJF) and Shortest Remaining Time First (SRTF) is relying on knowing the length of the CPU-bursts for processes in the ready queue. There are several methods to predict the length of the CPU-bursts, such as exponential averaging method, however these methods may not give an accurate or reliable predicted values. In this paper, we will propose a Machine Learning (ML) based approach to estimate the length of the CPU-bursts for processes. The proposed approach aims to select the most significant attributes of the process using feature selection techniques and then predicts the CPU-burst for the process in the grid. ML techniques such as Support Vector Machine (SVM) and K-Nearest Neighbors (K-NN), Artificial Neural Networks (ANN) and Decision Trees (DT) are used to test and evaluate the proposed approach using a grid workload dataset named "GWA-T-4 AuverGrid". The experimental results show that there is a strength linear relationship between the process attributes and the burst CPU time. Moreover, K-NN performs better in nearly all approaches in terms of CC and RAE. Furthermore, applying attribute selection techniques improves the performance in terms of space, time and estimation.**

Keywords - CPU-Burst; CPU Scheduling Algorithm; Machine Learning; Feature Selection.

I. INTRODUCTION

In Operating Systems (OS); processes, threads and data flow are given access to system resources by what is called scheduling algorithms. Nowadays with the modern systems and multiprogramming, the requirement to perform more than one process at a certain time is the reason behind the arising of scheduling algorithms. Different CPU scheduling algorithms have been proposed to deal with deciding which of the process that is in the ready queue to be allocated to the CPU. Different approaches are used by those scheduling algorithms for the way of selecting the process from the ready queue and allocating it to the processor [1]. First Come First Serve (FCFS), Shortest-Job-First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling (PS), Round-Robin (RR), etc. are examples of CPU scheduling algorithms which have been proposed in the literature. In SJF scheduling or SRTF, for example, along with the process there is a needs to know what is called *the length of the CPU-burst* to apply such scheduling algorithms. SJF and SRTF favor processes with small burst length compared to those with larger ones to ensure that the process with small burst time will complete and leave the system as soon as possible for the larger ones.

Amur et al. [14] stated that the CPU burst length for a process is measured as the time spent executing on the CPU from when it was scheduled, to the point where it is taken off the run queue. So the key problem with the SJF and SRTF scheduling is the need to know the length of the CPU burst. Knowing the next CPU burst length for a process is not an easy task where it needs more effort and calculations. Traditionally, there is a mathematical formula which can be used to predict the length of next CPU burst for a process. That traditional way called "Exponential Averaging (EA)" and it depends on using the process previous history to predict length of next CPU burst [1]. In Computational Grids (CGs) where multi-level platforms from multiple locations are used to provide a wide range of services to reach a common goal; scheduling task considered a vital component of a CGs infrastructure. Grid scheduling is *"the process for making the scheduling decisions involving resources over multiple administrative domains [2]"*. This process also includes searching multiple administrative domains to assign a job to a single machine or scheduling a job's tasks to multiple resources at a single site or multiple sites".

Grid Scheduling has three main phases; first phase is about discovering available resources; second phase, the most important, is about allocating and assigning jobs to feasible resources; while the third phase is about executing the jobs [3]. Allocating and assigning jobs to feasible resources in CG scheduling, the second phase, is often done by a dedicated machine in the grid. Therefore, implementing scheduling algorithms in CGs which are depending on the length of CPU-burst necessitate that dedicated machine to have the ability to predict the length of next CPU-burst.

In this paper, we proposed a ML-based approach that can be used in CGs to predict the CPU-burst length for processes to be executed. The proposed approach is benefiting from examining the most significant attributes for a process so that it can accurately predict the next CPU-burst.

The rest of the paper is organized as follows. In Section II we summarize the related work. The proposed approach and its associated model are presented in Section III. Section IV presents the detailed description of the used grid and analyzes process's related attributes. Section V presents the experiments while Section VI discusses the results. Finally, the paper is concluded in Section VII.

II. RELATED WORK

As mentioned earlier, to apply scheduling algorithms like SRTF and SJF, the CPU burst length can be estimated traditionally by what is called EA. By EA method, the length

978-1-4673-8675-3/15 $31.00 © 2015 IEEE
DOI 10.1109/AIMS.2015.11

of a process CPU-burst is approximated to the length of previous execution. On the step of knowing the CPU-burst length, Mahesh Kumar et al. [4] proposed a method to predict the length of the next CPU burst in SJF scheduling algorithm. Dual-Simplex Optimization Method (DSOM) was proposed based on what is called Linear Programming Models Dual Simplex Algorithm (LPMDSA) where that algorithm starts with a dual feasible basis constraints and works by looking for primal feasibility while keeping optimality [5]. They starts their work by using the traditional SJF scheduling algorithm and calculating the waiting time and turnaround time related to the given burst time of individual processes. Then they converted the given burst time, waiting time and turnaround time into a mathematical model called Linear Programming and applied the DSOM.

Pourali et al. [6] presented another way to estimate and predict the length of the next CPU-burst to apply SJF scheduling algorithm. The proposed technique was based on a fuzzy system as a knowledge-based rule system [7]. The fuzzy system was used to forecast the next burst time of a process to be scheduled based on past history of that process. The input of the system was the value of the previous used bursts time of a process while the output of the system is the value of the estimated next CPU-burst time. However, historical information for process execution is useful to predict future run time as it is shown in Smith et al. [9]. They presented a technique to derive a prediction for run times of parallel jobs from the run times of executing similar previous jobs. The novel aspect of that work is the use of two search techniques (greedy search and a genetic algorithm search) to determine the application attributes that yield the best definition of similarity for the purpose of making predictions. From the obtained results, they have shown that the genetic search performs better for every workload than greedy search. Their work also provides insights into the job attributes that are significant for identifying similar jobs where the names of the submitting user and the application form the most important attributes to know about jobs. ML techniques have been used to predict application resource consumption as an appealing approach pursued by several previous studies. Matsunaga et al. [10] is an example of those studies where their work comparatively assesses the suitability of several ML techniques for predicting spatiotemporal utilization of resources by applications. In their work, they were predicting execution time, memory and disk requirements for two bioinformatics applications. SVM and K-NN were used in that study as classification and regression models. From the literature, we found that there is a chance to fill the gap in academia research in predicting CPU-burst time by proposing a ML-based approach for that propose.

III. Proposed Approach

We proposed a ML-based-approach aiming at estimating the CPU-burst time of a process by looking on the most significant process attributes. ML as a technique has been used frequently in computer science and engineering fields that concern in constructing computer programs aiming to improve their performance automatically with experience. It is about learning some properties of a portion of data at hand called training set and using the other portion of that dataset as testing set to test the learned properties [11]. Four ML techniques named SMOReg, MLP, Decision tree (REPtree) and K-NN [12] are used in this work. The proposed approach consists of six steps as depicted in Fig. 1.

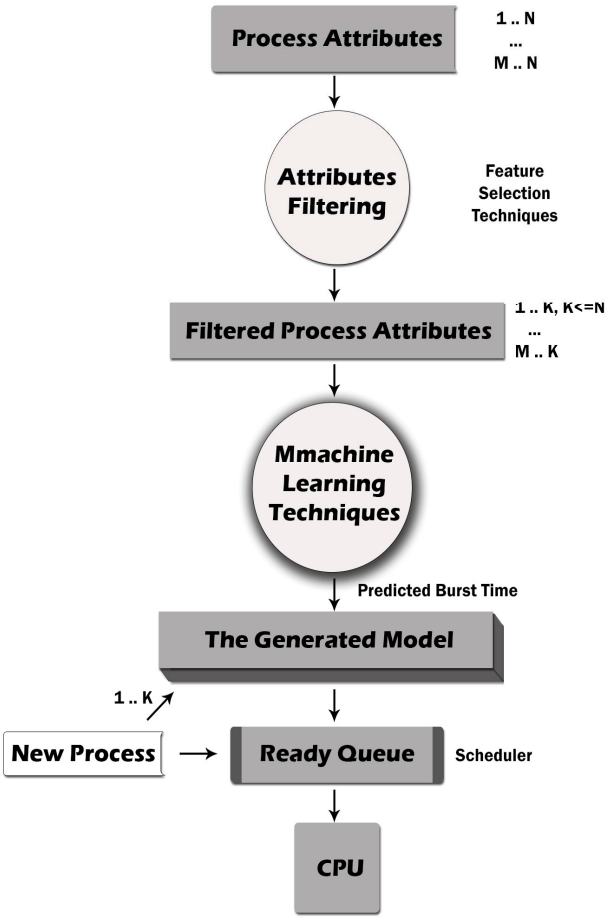

Figure 1. The proposed Approach

These steps are summarized as following:

1. ***Preparing/creating the dataset:*** in this step, a real grid workload which contains process attributes is used. Then, the process attributes were represented as feature vectors. After that, the dataset was analyzed and divided into training and testing sets.
2. ***Attributes filtering/selecting:*** feature selection techniques were applied to select the most significant attributes by applying feature selection technique.
3. ***Generating the model:*** in this step ML techniques namely SMOReg, MLP, Decision tree (REPtree) and K-NN were applied to generate the model using the training dataset.
4. ***Evaluating the model:*** in this step, the testing dataset was used to evaluate the generated model, in the previous step. The performance of the generated model

was then evaluated in terms of Correlation Coefficient (CC) and Relative Absolute Error (RAE).

5. ***Predicting the CPU burst time of the process:*** the generated model was used to predict the CPU burst time of a new process in the input queue.
6. The process with its predicted CPU burst length is sent to the ready queue to be scheduled.

IV. GRID DESCRIPTION AND ANALYSIS

For the purpose of testing the proposed model, mentioned in the previous section, to predict the CPU burst length, we used "GWA-T-4 AuverGrid" grid work load[1]. AuverGrid is a production grid platform composed of five clusters. Each cluster contains Dual 3GHz Pentium-IV Xeons running Scientific Linux. The number of the processors in AuverGrid is 475 processors distributed among five clusters 112, 84, 186, 38 and 55 CPUs, respectively. These clusters positioned in France, region of Auvergene. The Grid workload consists of 404,176 jobs; each job has 29 properties as defined in Table 1. From the 404,176 jobs in the grid workload, we selected the first 5,000 jobs. Next, the attributes were analyzed and we found that attributes numbered in Table I from 19 to 29 have no information or values. In other words, the last 11 attributes were eliminated and neglected because they have the value of -1 which means missed data. For the remaining 18 attributes from 1 to 18, we consider that the attribute **RunTime** is the actual CPU burst time.

TABLE I. DESCRIPTION OF THE GRID WORKLOADS

#	Attribute name	Description
1	**JobID**	The identified number of job.
2	**SubmitTime**	The submit time in seconds.
3	**WaitTime**	Wait time in seconds
4	**RunTime**	Run time measured in seconds
5	**NProcs**	Number of allocated processors
6	**AverageCPUTime Used**	Average of CPU time over all allocated processors.
7	**Used Memory**	Average used memory per processor in kilobytes.
8	**ReqNProcs**	Requested number of processors
9	**ReqTime**	Requested time measured in seconds.
10	**ReqMemory**	Requested memory (average per processor) per processor in kilobytes.
11	**Status**	Job completed =1, job failed=0, job cancelled =5.
12	**UserID**	String identifier for user
13	**GroupID**	String identifier for group user belongs to
14	**ExecutableID**	Name of executable (application), a natural number, between one and the number of different applications appearing in the workload
15	**QueueID**	String identifier for queue
16	**PartitionID**	String identifier for partition
17	**OrigSiteID**	String identifier for submission site
18	**LastRunSiteID**	String identifier for execution site
19	JobStructure	Single job = UNITARY, composite job = BoT
20	JobStructureParams	If JobStructure = BoT, contains batch identifier
21	UsedNetwork	Used network resources in kilobytes/second
22	UsedLocalDiskSpace	The used local disk space in megabytes
23	UsedResources	List of comma-separated generic resources (i.e. memory usage in Gb, IO data transferred, and IO wait time in seconds)
24	ReqPlatform	The requested platform, i.e. CPU Architecture, OS, OS Version
25	ReqNetwork	The requested network in kilobytes/second.
26	ReqLocalDiskSpace	The requested local disk space in megabytes
27	ReqResources	List of comma-separated generic resources.
28	VOID	Identifier for Virtual Organization
29	ProjectID	Identifier for project

[1] http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid

V. EXPERIMENTS AND RESULTS

In this section, we address two research questions:

1. Can the attributes of a born process (without historical information) be used to estimate the CPU burst length? And what are the most significant attributes among them?
2. Can the attributes with historical information of a process be used to estimate the CPU burst length? And what are the most significant attributes among them?

We mean here by processes attributes without historical information that, the related information of the process before any execution (newly born processes). That is because when the process is created, the related historical information is not available at that certain time. Additionally, we add the second research question to evaluate our method in case of availability of historical information. To distinguish between the historical and non-historical information, any attribute has value when its process is born. To answer the defined research questions, we classified the attributes of the grid into two categories; process attributes with non-historical information and process attributes with historical information. Toward this end, we analyzed the non-neglected 18 attributes one by one. Intuitively, JobID is eliminated because it is just a counter and it does not make any sense where its value was embedded in the SubmitTime attribute. That means the jobs submitted earlier have smaller value of JobID. The process attributes that are used to answer our first research question are listed and described in Table II.

TABLE II. THE PROCESS ATTRIBUTES WITHOUT HISTORICAL INFORMATION

Attribute	Reason for Selection
SubmitTime	Reflects the process arrival time.
Used	Reflects the size of memory the process will

Memory	need in advance.
ReqNProcs	Reflects how the process is heavy according to the number of processors it uses.
ReqTime	Reflects the initial expected time for the process to finish its task.
ReqMemory	Reflects the amount of memory the process will need in addition to the used memory which has.
UserID	Reflects the type of the user and his importance.
GroupID	Reflects the type of the group and its importance.
QueueID	Reflects the type of the queue and its importance.
OrigSiteID	Identifies the original site of the process.

In contrast, the attributes listed in Table III are the process attributes with historical information. The justification behind considering these attributes as historical information is that they obtained there values after performing at least one dispatching process.

TABLE III. THE PROCESS ATTRIBUTES WITH HISTORICAL INFORMATION

Attribute	**Reason for Selection**
SubmitTime	Reflects the process arrival time.
Used Memory	Reflects the size of memory the process will need in advance.
ReqNProcs	Reflects how the process is heavy according to the number of processors it uses
ReqTime	Reflects the initial expected time for the process to finish her task.
ReqMemory	Reflects the amount of memory the process will need in addition to the used memory which has.
UserID	Reflects the type of the user and his importance.
GroupID	Reflects the type of the group and its importance.
QueueID	Reflects the type of the queue and its importance.
PartitionID	Identifies which machine in a cluster was used.
OrigSiteID	Identifies the original site of the process.
WaitTime	Refers to the difference between the process's submit time and the time at which it actually began to run.
AverageCPUTimeUsed	Refers to the average of CPU time over all allocated processors.
NProcs	Refers to the number of processors the process uses.
Status	Identifies whether the process has completed its task or not; 1 if the job was completed, 0 if it failed.
ExecutableID	Reflects the type of the application and its importance.
LastRunSiteID	Identifies the site where the process finished its work on.

To predict the CPU burst length, we used different ML techniques as mentioned above. These techniques were implemented in WEKA tool [13]. The parameters can be learned using various algorithms. We investigated several parameters and then we used PUK kernel function in our experiments with other default WEKA parameters. K-NN was also used for prediction with value of k=3. MLP implements multi-layer perceptron for regression. We used MLP with investigating different architectures of MLP such that we used 20 hidden layers with learning rate with 0.1. REPTree is fast decision tree learner implements regression tree using information gain/variance and prunes it using reduced-error pruning (with back fitting) [13]. REPTree was implemented with its default parameters. In our experiments, we selected 1,000 jobs randomly to be used in estimating the CPU burst length. Eighty percent (800 jobs) were used for training and 20% (200 jobs) were for testing. In order to identify which of the process attributes play an important role in predicting the CPU burst length, we divided the selected attributes into two categories and implemented the experiments on each category attributes without historical information and attributes with historical information. The obtained results were as following:

A. Process attributes without historical information

We evaluated the efficiency of the proposed approach in terms of CC and RAE. The results are shown Table IV.

TABLE IV. THE CC AND RAE OF THE PROCESS ATTRIBUTES WITH NON-HISTORICAL INFORMATION

Exp1	**Correlation Coefficient**	**Relative Absolute Error**
K-NN	0.9559	7.9114 %
SVM	0.9535	7.5511 %
MLP	0.945	20.1198 %
REPtree	0.943	9.5093 %

We used ReliefF feature selection technique which is implemented on WEKA tool to find the most significant attributes. ReliefF tests the worth of a feature or an attribute by sampling an instance repeatedly and considering the value of the given attribute for the nearest instance of the same and different class [8]. The attributes and their ranks are shown in Table V.

TABLE V. THE RANKED ATTRIBUTES OF THE PROCESS

Attribute	**Rank**
Submit Time	0.065775
User ID	0.05697
Used Memory	0.043279
Requested Time	0.007216
Requested Memory	0.007103
Requested Number of Processors	0
OrigSiteID	-0.000985
Group ID	-0.001005
Queue Number	-0.045437

We selected those attributes that have rank greater than zero i.e., the first five attributes in Table V. Then, we used them to estimate the length of CPU burst length using the ML techniques. The results are represented in Table VI.

B. Process attributes with historical information

To answer the second question, we applied the same techniques and methodology used in the previous two experiments but using the process attributes with historical information. The results are shown Table VII. We applied the feature selection technique to select the most significant attributes among the whole attributes. The attributes and their ranks are shown in Table VIII.

TABLE VI. THE CC AND RAE OF THE REDUCED PROCESS ATTRIBUTES WITH NON-HISTORICAL INFORMATION

Exp2	Correlation Coefficient	Relative Absolute Error
K-NN	0.9687	7.3598 %
SVM	0.9605	7. 8048 %
MLP	0.9405	20.8615 %
REPtree	0.943	9.5093 %

TABLE VII. THE CC AND THE RAE OF THE PROCESS ATTRIBUTES WITH HISTORICAL INFORMATION

Exp3	Correlation Coefficient	Relative Absolute Error
K-NN	0.9874	4.4324 %
SVM	0.9672	5.7481 %
MLP	0.995	5.7876 %
REPtree	0.9933	5.2641 %

TABLE VIII. THE SELECTED ATTRIBUTES OF PROCESS ATTRIBUTES WITH RANKS

Attribute	Rank
Average CPU Time Used	0.217009
Submit Time	0.079879
User ID	0.066924
Used Memory	0.041672
Status	0.035844
Wait Time	0.014908
Requested Time	0.003665
Requested Memory	0.001369
Partition Number	0
Number of Allocated Processors	0
Executable (Application) Number	0
Requested Number of Processors	0
OrigSiteID	-0.000963
LastRunSiteID	-0.000963
Group ID	-0.000996
Queue Number	-0.04452

We also evaluated those attributes that have rank of greater than zero, i.e., the first eight attributes in Table VIII. We applied aforementioned ML techniques to predict our goal and the results are shown in Table IX.

TABLE IX. THE CC AND RAE OF THE REDUCED PROCESS ATTRIBUTES WITH HISTORICAL INFORMATION

Exp4	Correlation Coefficient	Relative Absolute Error
K-NN	0.9933	3.4641 %
SVM	0.9909	4.1309 %
MLP	0.99	5.4269 %
REPtree	0.9933	5.2641 %

VI. RESULTS AND DISCUSSION

As for process attributes with non-historical information, Fig. 2 shows the CC of the whole non-historical attributes and those reduced after applying feature selection technique. The highest CC was 0.9687obtained using K-NN in case of the reduced process attributes with non-historical information. Additionally, all prediction techniques were achieved high CC which look similar.

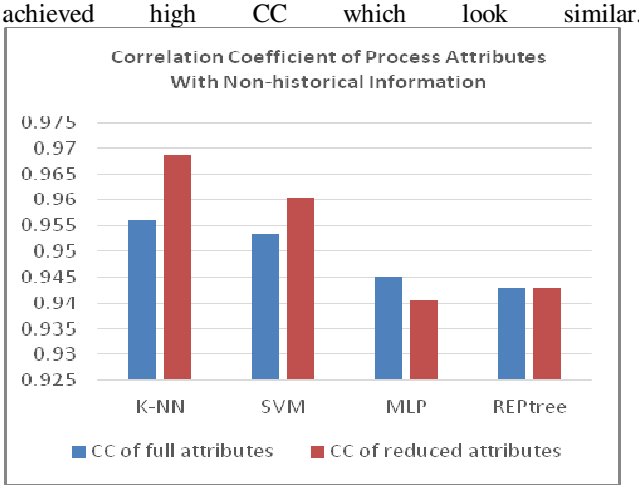

Figure 2. CC of process attributes with non-historical information

Fig. 3 shows the RAE of the whole non-historical attributes and those reduced after applying feature selection technique. The lowest RAE was 7.40% achieved using K-NN in case of the reduced process attributes with non-historical information.

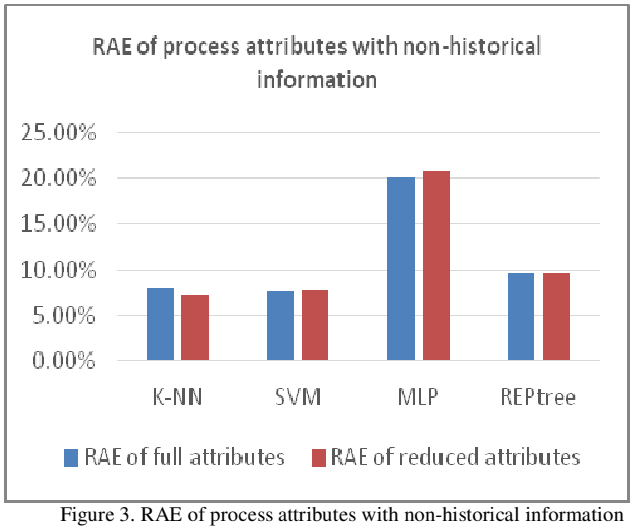

Figure 3. RAE of process attributes with non-historical information

As for the process attributes with historical information, Fig. 4 shows the CC of the process attributes with historical attributes and those reduced after applying feature selection technique. The highest CC was 0.995 obtained using MLP in case of the process attributes with historical attributes. Additionally, all prediction techniques were achieved high CC and the results look similar in the most cases.

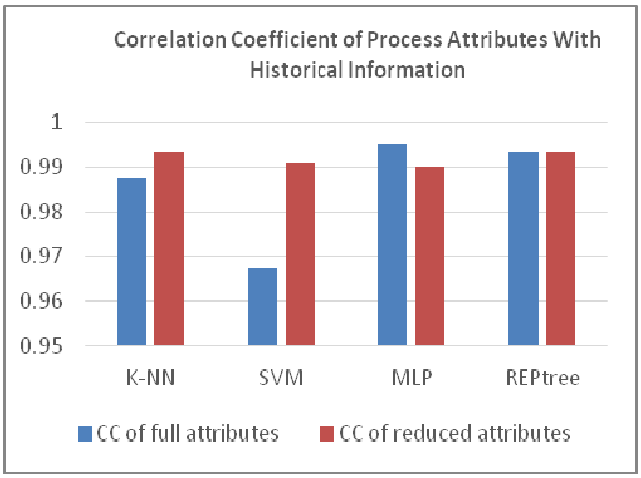

Figure 4. CC of process attributes with historical information

Fig. 5 shows the relative absolute errors of the process attributes with historical attributes and those reduced after applying feature selection technique. The lowest RAE was 3.46% achieved using K-NN in case of the reduced process attributes with historical information.

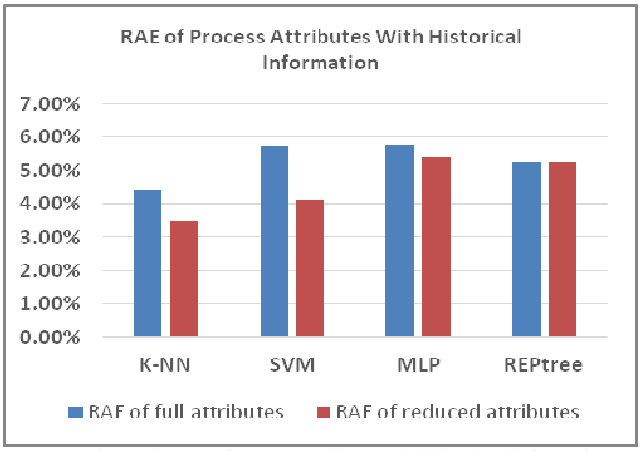

Figure 5. RAE of process attributes with historical information

VII. Conclusion and Future Work

In this paper, we proposed an approach to estimate the CPU burst length for processes in ready queue by using ML techniques including K-NN, SVM, ANN, and DT. In the proposed approach, the attributes feature selection techniques to select the most significant attributes were applied on a grid workload named "GWA-T-4 AuverGrid". The experimental results show that there is a strength linear relationship between the process attributes and the burst CPU time where K-NN performs better than other ML techniques in nearly all cases in terms of CC and RAE. Applying attributes selection techniques improve the performance in terms of space, and prediction time. Furthermore, from obtained results, it is clear that the CPU-burst length could be predicted better in processes with historical information than in processes with non-historical information. The proposed approach contributes to the implementation of SJF and SRTF in CGs. Finally, updating the model periodically using the terminated processes will improve the prediction accuracy. As a future work, we will test the proposed approach to estimate the time quantum and load balancing in the CGs.

Acknowledgements

The authors would like to acknowledge the support provided by King Fahd University of Petroleum & Minerals for conducting this research.

References

[1] Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. Operating system concepts. Vol. 8. Wiley, 2013.

[2] Mehmood Shah, Syed Nasir, Ahmad Kamil Bin Mahmood, and Alan Oxley. "Analysis and evaluation of grid scheduling algorithms using real workload traces." In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pp. 234-239. ACM, 2010.

[3] Fernández-Baca, David. "Allocating modules to processors in a distributed system." IEEE Transactions on 15, no. 11 (1989): 1427-1436.

[4] Mahesh Kumar, M. R., B. Renuka Rajendra, C. K. Niranjan, and M. Sreenatha. "Prediction of length of the next CPU burst in SJF scheduling algorithm using dual simplex method." In *Current Trends in Engineering and Technology (ICCTET), 2014 2nd International Conference on*, pp. 248-252. IEEE, 2014.

[5] Kasana, Harvir Singh, and Krishna Dev Kumar, eds. *Introductory operations research: theory and applications*. Springer Science & Business Media, 2004.

[6] Pourali, Abdolghader, and Amir Masoud Rahmani. "A Fuzzy-based Scheduling Algorithm for prediction of next CPU-burst time to implement Shortest Process Next." In *Computer Science and Information Technology-Spring Conference, 2009, IACSITSC'09*, pp. 217-220, IEEE, 2009.

[7] Kosko, Bart. *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Prentice-Hall, Inc., 1991.

[8] Kohavi, R., and R. Kohavi. 1997. "Wrappers for Feature Subset Selection." Artificial Intelligence 97(1-2):273–324. Retrieved (http://linkinghub.elsevier.com/retrieve/pii/S000437029700043X).

[9] Smith, Warren, Ian Foster, and Valerie Taylor. "Predicting application run times with historical information." *Journal of Parallel and Distributed Computing* 64, no. 9 (2004): 1007-1016.

[10] Matsunaga, Andréa, and José AB Fortes. "On the use of machine learning to predict the time and resources consumed by applications." In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 495-504, IEEE Computer Society, 2010.

[11] Tom Mitchell, *Machine Learning*, 1st Ed, pp: 52-75, 154-183, 230-244, the Mc-Graw Hill Company. Inc. International Edition, 1997.

[12] Shevade, S. K., S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy. 2000. "Improvements to the SMO Algorithm for SVM Regression." *IEEE Transactions on Neural Networks* 11(5):1188–93.

[13] Witten, Ian H., and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*.

[14] Amur, Hrishikesh R., Gautham R. Shenoy, Dipankar Sarma, and Srivatsa Vaddagiri. "Plimsoll: a DVS Algorithm Hierarchy."