Specifying CPU Requirements for HPC Applications via ML Techniques

Priyanka Bharti[1] and Rajeev Ranjan[2]
[1] School of C&IT, REVA University, Bengaluru, INDIA
[2] School of CSA, REVA University, Bengaluru, INDIA
[1] priyankabharti@reva.edu.in, [2]rajeevranjan@reva.edu.in

ABSTRACT

Resource distribution in data centers is difficult for service providers because of the structures of usage and condition setup decisions. Customers encounter issues to anticipate the amount of CPU and memory required for job execution, and henceforth are not ready to assess when work yield shall be accessible to plan for next analyses. Systems that utilize cluster scheduler structures to gauge job execution time exists in the literature. Notwithstanding, we have seen that such methods are not appropriate for anticipating CPU utilization. In this paper, we assist customers to figure out their applications CPU usage utilizing machine learning (ML) techniques. We analyze how scheduler can be utilized to predict CPU utilization through ML techniques, and its evaluation on two frameworks containing an enormous number of user jobs.

Keywords: **HPC, CPU Prediction, Machine Learning**

1 Introduction

The job analyzers in HPC environment depend on clients specifying job requirements. The job requirement has to be specified in terms of memory usage, number of processors required, and time for execution of a job [1,2]. These qualities are hard for the client to determine in view of different choices of jobs and condition setups and perplexing effects of these characteristics on by and large job execution [3, 4]. Client's task assignments and its effect on planning choices are explored by few authors [5, 6].

The processor and memory requirement influence the execution of customer's task. If customers demand a particular quantity of CPU but utilize less, the CPUs are said to be underutilized [7]. Frameworks, for example, XSEDE use estimation techniques for jobs to be executed in the pipeline. As per our research, we saw that applying those methodology to CPU usage does not cause the best estimates. In the current paper, we present a tool for estimating CPU usage utilizing cluster scheduler using machine learning (ML) that could be utilized for various parameters appraisal identified with resource conveyance in HPC circumstances.

Our contributions are as follows: 1. Prediction of CPU usage based on ML tool 2. Assessment of the ML tool to see how it delivers for handling the CPU usage prediction task. The organization of paper is as follows. Related works are given in section 2. Proposed ML technique is given in section 3, section 4 presents system evaluation/results. Finally, section 5 covers conclusion.

2 Related Works

Existing research are generally focused on optimizing time w.r.t waiting and execution. The endeavors are useful in optimizing different features. If we can predict well in advance about how much time a job will be hold up in the queue, it can help us in placing the job appropriately. For example, authors in [1] inspected procedures and calculations to enhance queue waiting time expectations.

The framework in [2] relies upon historical data. Authors in [6] developed a procedure for evaluating job runtime and queue waiting time. The procedure relies upon existing algorithms to make prediction as accurate as possible. Authors in [3] developed a methodology for arranging tasks/jobs in light of framework made work runtime expectations.

Endeavors on CPU utilization depend generally on benchmarking rather than on scheduler logs [1, 3, 4, 5, 6, 7]. For instance, authors in [6] assessed diverse ML techniques for envisioning spatio-fleeting technique for resource prediction. Another model is from authors in [7] who utilized virtualization technique for resource prediction. Their procedure depends on virtualization technology and a backslide based model to delineate local framework into a virtualized one [8,9,10].

3 Proposed Work

Machine learning is a procedure for computational learning centering most AI applications. In ML, frameworks or computations upgrade themselves through data training without relying upon explicit programming. ML calculations are broad systems fit for doing estimates while in the meantime picking up from more than trillions of observations. The framework description of the proposed system alongside the expectation methods are discussed as follows.

3.1 Framework Description

The information flow in the proposed tool involves three central phases: (i) information gathering and transformation, (ii) model development, and (iii) performing forecast. The first phase gauges resource utility through historical information. The second phase interfaces with the load sharing facility (LSF) group scheduler to amass fitting info for optimizing the

performance parameters. While, the third phase forecasts CPU resource utilization, which is discussed in this paper.

Information accumulation is very important for prediction. We collect it via online and offline environment. In online environment, real time data is collected and submitted to the group. It stacks all the data from LSF. In offline mode, tool collects data from the database. Further, in online mode the data gathered are classified into two categories: features and labels. The condition triggering the event is put under features, and occurrence of event is put under labels. Our research focuses on job attributes like customer id, the resource requirement in terms of CPU core and memory requirement.

The features that are gathered are preprocessed and stored in the database. The data now gets changed or converted to meaningful information that should be used for prediction. The quantity of CPU cores, which is 8 cores is marked as label to specify the requirement. For the training purpose the features and labels from the database are utilized. The window size is taken as 10000 entries for our experiment. A feature is the input we have fed to the system and the label is the output that we expect.

Features and labels are split for training and validation, respectively. As a rule, we don't expect that models that have just retained the labels, will have the capacity to sum up when assessed in validation stage. Models that have taken in the labels, then again, will as a rule have the capacity to sum up on the grounds that they can perceive mark structures in the validation sets that are like the label structure that they learned in the training set.

3.2 Prediction Techniques

In this section, we discuss the ML computations utilized in our tool. We have used four strategies that are given here: Support Vector Machines (svm), Random Forests (rforest), Multilayer Perceptrons (mlp), and k-Nearest Neighbor (knn).

1) Support Vector Machines (SVM): SVMs are ML counts at first proposed for two-assemble classification issues that utilize kernel techniques to depict vectors to (conceivably) non-straight high-dimension highlight spaces. In machine learning, SVMs are regulated learning models with related learning calculations that dissect information utilized for grouping and relapse investigation. Given an arrangement of training examples, each set apart as having a place with either of two classifications, a SVM preparing calculation assembles a model that appoints new precedents to one classification or the other. The proposed tool uses multi-label classification using one to many strategy, with regularization consistent C = 0.01.

2) Random Forests (rf): Random forests are a troupe learning strategy for grouping, relapse and different assignments, that work by developing a huge number of choice trees at preparing time and yielding the class that is the method of the classes or mean expectation of the individual trees. In this work, we use forests having 20 trees for splitting the jobs.

3) Multilayer Perceptron: Multi Layer perceptron (mlp) is a feedforward neural system with at least one layers among information and yield layer. Feedforward implies that information streams in a single heading from contribution to yield layer (forward). This sort of system is prepared with the backpropagation learning calculation. Two frameworks are utilized in our tool, a fixed and a dynamic framework.

4) k-Nearest Neighbors (knn): In example acknowledgment, the k-closest neighbors calculation is a non-parametric strategy utilized for arrangement and relapse. In the two cases, the info comprises of the k nearest preparing models in the element space. The yield relies upon whether knn is utilized for arrangement or relapse: In knn grouping, the yield is a class participation. We have used ballot based classification and a regression model, using k=5.

4 Results

The frameworks used for assessment of our tool is 26-hub POWER8 cluster, and a generation system made out of x86 nodes. Applications from the POWER8 clump have distinctive qualities and more regular HPC extraordinary jobs that needs to be done. The x86 structure runs production applications, which are routinely executed by clients.

In Tables 1 and 2 we take a look at the execution of the mode and each ML procedure for the validation and test sets in the midst of all of the 5,000-work segments. We highlight the fundamental five methodologies in the two sets. One would plan to see an equivalent precedent in the validation and testing set. However, the five strategies are truly one of a kind in those sets. Especially, svm performs well on validation set.

Table 1. Validation Performance for x86 system

segment	mode	SVM	rforest	mlp	knn
0	Online	0.7726	0.8220	0.7642	0.7631
1	Online	0.8072	0.8477	0.8005	0.8202
2	Offline	0.8036	0.8365	0.7823	0.8145
3	Offline	0.8028	0.8366	0.7863	0.8122
4	Offline	0.7033	0.7813	0.7008	0.7546

Table 2. Test Performance for x86 system

segment	Mode		SVM
rforestmlpknn 0		Online	0.8606
0.3948	0.6546		0.8610 1
Offline	0.7448	0.1202	0.7544
0.7448 2		Offline	0.8640
0.0484	0.8698		0.8630 3
Offline	0.8836	0.2464	0.8918
0.7726 4		Offline	0.8038
0.7568	0.7812	0.8011	

Also, the Random Forests (rforest) method performs well with everything taken into account, beside a few segments in the x86 structure. As a result, in case we depend just on this procedure, we may have a superior execution in a couple of conditions in light of the speculative nature of the system. Multilayer Perceptron with cross-validation (mlp) appears to over-fit. Even more basically, the more refined knn is not dependable. This system is used in progress in the XSEDE grid to foresee line holding up time. We have implemented this procedure ourselves for cross breed cloud conditions with sensibly extraordinary results for both running time and queue time.

Tables 3 and 4 exhibit how the execution of the polling based system vacillates as the amount of voters increase. It is easy to analyze that a polling based approach diminishes the peril of relying upon a single indicator. The fragment 0 of the x86 structure, for example, using three models beats the best indicator by around 26\%. It does not come without a cost, be that as it may, as including more voters diminishes execution barely.

Table 3. Validation Performance for POWER8 system

segment	Mode		SVM
rforestmlpknn			
0	Online	0.9509	0.9528
0.9996	0.9187 1		Online
0.9641	0.9642	0.9992	0.9460
2	Offline	0.9704	0.9740
0.9994	0.9663 3		Offline
0.9537	0.9543	0.9996	0.9439
4	Offline	0.8767	0.8786
0.9982	0.8459		

Table 4. Test Performance for POWER8 system

segment	Mode	SVM	rforestmlpknn
0	Online	0.8745	0.0013
0.8780	0.8737 1		Online
0.8510	0.0003	0.8545	0.8500
2	Offline	0.8325	0.0031
0.8317	0.8287		
3	Offline	0.8264	0.0092
0.8360	0.8162 4		Offline
0.7412	0.0232	0.7474	0.7400

5 Conclusion

End users in HPC environment continue to experience issues to decideresource prerequisites for execution of their jobs. Novel ideas are required to enhance usage of clustered resources in data centers of Cloud. For such scenarios, we proposed here a system for anticipating CPU necessities of jobs submitted to a few clusters using ML techniques. In future, we plan to extend this sort of system for HPC Cloud with regard to resource management, and security issues.

References

[1] C. B. Lee, A. Snavely, "On the user–scheduler dialogue: studies of user-provided runtime estimates and utility functions", Journal of High Performance Computing Applications, 20 (4), 495-506, 2017.

[2] C. B. Lee, Y. Schwartzman, J. Hardy, A. Snavely, "Are user runtime estimates inherently inaccurate?", Proceedings of the International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), Springer, 253-263, 2014.

[3] D. Tsafrir, Y. Etsion, D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates", IEEE Transactions on Parallel and Distributed Systems 18 (6), 789-803, 2017.

[4] S.-H. Chiang, A. Arpaci-Dusseau, M. K. Vernon, "The impact of more accurate requested runtimes on production job scheduling performance", Proceedings of the International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), Springer, 103-127, 2012.

[5] D. Tsafrir, D. G. Feitelson, "The dynamics of backfilling: solving the mystery of why increased inaccuracy may help", Proceedings of IEEE International Symposium on the Workload Characterization, IEEE, 131-141, 2016.

[6] D. Zotkin, P. J. Keleher, "Job-length estimation and performance in backfilling schedulers", Proceedings of the International Symposium on High Performance Distributed Computing (HPDC), IEEE, 236-243, 2017.

[7] M. Hovestadt, O. Kao, A. Keller, A. Streit, "Scheduling in HPC resource management systems: Queuing vs. planning", Proceedings of the International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), Springer, 1-20, 2013.

[8] R. L. Cunha, E. R. Rodrigues, L. P. Tizzei, M. A. Netto, "Job placement advisor based on turnaround predictions for HPC hybrid clouds", Future Generation Computer Systems 67, 35-46, 2017.

[9] A. Coates, A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization", Proceedings of the 28th International Conference on Machine Learning (ICML-11), 921-928, 2017.

[10] C. Cortes, V. Vapnik, "Support-vector networks", Journal of Machine Learning, 20 (3), 273-297, 2005.