Artificial Intelligence Approach for Batch Completion Time Prediction

Ahmad Alnafessah
Computing Department
Imperial College London
London, United Kingdom
a.alnafessah17@imperial.ac.uk

***Abstract*—In the production environment, the data center has obtained significant popularity as a cost-efficient platform. Conventional data centers have an enormous amount of over-provisioned computing resources for production applications that accommodate fluctuating workloads and peak demands. Most of the time, conventional data centers in production suffer from over-provisioning and have low utilization, which often less than 50% utilization is used of computing resources. We propose a model that is agnostic of the business logic internal to each job. Instead, it learns from data by applying artificial neural networks and compares the proposed model with queueing-theoretic model on historical measurements of job completion time and CPU run-queue size. The proposed solution not only predicts completion time for two jobs with high accuracy, but also saves considerable time in training the model to predict the time for more than two jobs and easily can be generalized to cover unforeseen workload combinations.**

***Keywords*-Artificial intelligence; Machine learning; Big data; Neural networks; Performance engineering**

I. Introduction

Bu et al., [1] present a task placement strategy to alleviate and estimate the task slowdown affected by the interference among virtual machines for MapReduce applications task scheduling optimization at the application level. The strategy is based on an exponential interference prediction model to schedule tasks based on that model. Their model is initially constructed through offline training data using different benchmarks, which are TeraSort, RWrite, Grep, WCount, TeraGen, Kmean, Bayes, and PiEst(1000). The authors [1] use the Gauss-Newton algorithm, which is an interactive process that regularly updates the parameters to calculate the coefficients that minimize the sum of squared errors (SSE) to reach the optimal solution. They evaluate the proposed solution by running CPU and IO-intensive applications that run on 72 nodes and virtual computing resources. Their results shows that the proposed scheduling strategy is capable of speeding up 6.5 times for individual jobs and 1.9 times improvement of system throughput compared to other scheduling strategies. The authors claim that their model could be applied to other virtual cluster schedulers.

For batch workload such as MapReduce, it is typically deployed on physical servers to avoid virtualized environments' performance overhead. Sharma et al., [2] utilize the unused computing resources by consolidating the batch jobs to improve application performance and show the strengths and weaknesses of both virtual and native environments. The authors consider the data center with a native and virtual machine environment to propose a hierarchical scheduler to manage resources of both interactive and batch workloads effectively. The scheduler has two phases, which are job classification and dynamic resource management to improve system utilization. The first phase has the profiling process of jobs to estimate job completion time (JCT) and calculate virtualization overheads to enhance the placement of jobs between native environments and virtual machines. This happens on a separate small training cluster that has both native and virtual environments to calculate the overhead for both to choose the ideal environment. Sharma et al., [2] show that the proposed model can enhance job completion time by 40% and contributes 45% improvement for computing resource utilization. The issue with the proposed model is that the correction only works after the interference or problems appear within the system. So we need a previous estimation model to detect possible risks and take action before performance degradation arises.

Classification techniques are machine learning approaches that have the ability to assign samples to target classes. In performance prediction, the classification approaches are essentially based on the assumption that tasks can be grouped into classes with similar performance behaviors.

Delimitrou et al [3] present Quasar, which uses classification techniques for an incoming job based on a short test run of the application to determine the suitable computing resource to pack workloads on available resources. The purpose is to extrapolate the performance behavior for unseen configurations from other applications with similar computer resource usage patterns, as exposed by the test phase. The authors assume that the workload can be partitioned into jobs that have similar behaviors. It is time-consuming for complex workloads to test every job behavior that has to be profiled under different resource configurations to assign it in a particular class. In addition, the other

Table I: The representation of Jobs names as a part of input features.

Exp	batik	jython	luindex	lusearch	sunflow	xalan
Experiment 1	1	0	0	0	1	0
Experiment 2	0	0	2	0	0	0
Experiment 3	0	0	1	0	1	0
Experiment 4	1	0	0	1	0	0

co-located jobs may affect each other during the run time, which makes it hard to cover all the possible combinations of workload behaviors.

The core contributions in this paper are as follows:

- The proposed model is agnostic of the business logic internal to each job. Instead, it learns from data by applying artificial neural networks and compare it with queueing-theoretic model on historical measurements of job completion time and CPU run-queue size (i.e., the number of active threads in the system).
- The model captures multi-threading, operating system scheduling, job priorities
- A validation on 2500 experiments based on the DaCapo benchmarking suite has been carried out, confirming the predictive capabilities of the model

II. Methodology

Our goal is to establish the completion time prediction error. We assume to have profiling data for individual jobs, and attempt to predict the completion times when four jobs run at the same time within the system. The prediction method works without any need for measurement from executions with four jobs, everything is predicted using only the profiling data and the neural networks model. The neural networks model with backpropagation and conjugate gradient are used to train the neural networks. In this work, the feature selection process covers hyperthreading, CPU nice, RunQ, throughput, and type of job as input features to the neural networks for training purposes.

In order to instantiate the model, the following profiling data was collected for all jobs by running them on the servers:

- Types of jobs which are represented in Table I
- noth: hyperthreading
- nice: CPU nice. The larger nice values mean lower priority. The default priority has a nice value of 0.
- RunQ: The number of active threads in the system
- Tput: Throughput

Using the input features above, the model will predict the completion times Cj for all running jobs, taking into account the CPU contention from all the other running batch jobs.

III. Evaluation

The proposed model is validated against batch workloads available in the DaCapo suite [4]. Dacapo is chosen because it is one of the most popular Java benchmark and it is ideal for scientific purposes and evaluation. This is a suite of Java benchmarks cited in over 1100 scientific papers and coauthored by Intel, IBM Research, and leading academic institutions.

The DaCapo benchmarks issuing HTTP or SOAP calls are excluded, as we expected these to be more typical of transactional workloads. The remaining benchmarks are as follows:

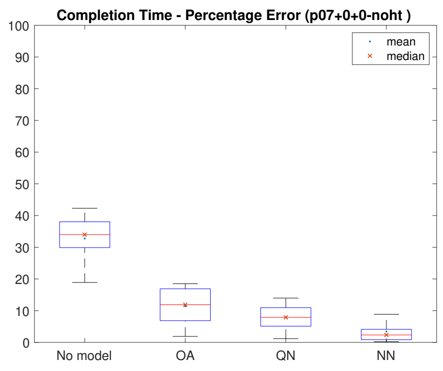
- Luindex: Document indexing with Apache Lucene. Mostly single threaded
- lusearch: Document search with Apache Lucene. Multithreaded.
- Xalan: Java XML/XSLT processing. Multithreaded.
- jython: Java-based scripting (Python-like). Mostly single threaded.
- batik: image generation based on Apache Batik. Single threaded.
- sunflow: ray tracing. Multithreaded

Over 2500 experiments are carried out using five different multi-core servers running Ubuntu Linux. When running 2 of the above benchmarks together, CPU utilization in our tests is typically around 70% to 90% on machines with 8-16 logical cores. With four benchmarks, the machines were always saturated near 100% utilization. We also varied the benchmark mix, the hyper-threading setup, job priorities and randomize the sleep time in-between successive runs of a benchmark within the same experiment. The default setup was to run experiments without hyperthreading and no sleep time. Each experiment was stopped after 5 minutes, since performance behavior stabilized fairly quickly.

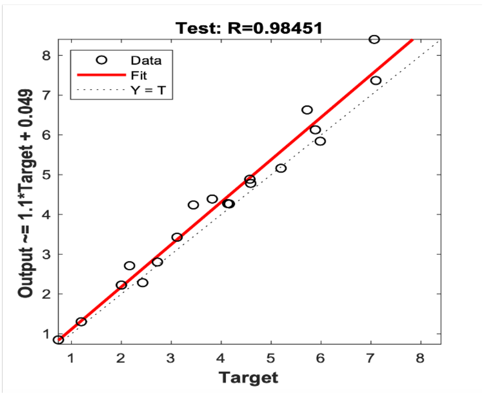
IV. Results

The proposed methodology is evaluated on an isolated Linux system. We avoid using a virtual environment to make sure that all the performance metrics are accurately measured. The testing results of the proposed model against two jobs are shown in Figure 1.

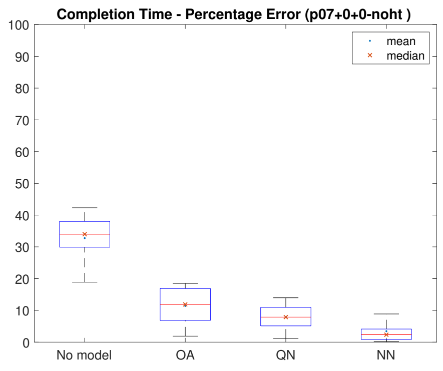
The boxplot diagram in Figure 1(c) indicates the mean (blue) and median (red) of the prediction error



(a) MAPE



(b) R value



(c) MAPE

Figure 1: Test of the proposed model against two jobs

for data collected across all possible of job combinations of the DaCapo benchmarks (e.g., xalan-xalan, xalan-batik, jython-luindex, xalan-batik, ..., etc). The blue box is the inter-quantile range (25th-75th percentile). The R-value (Figure 1(b)) and MAPE (Figure 1(c)) for predicting the completion time of two jobs are 98% and 5%, respectively. The goal is to generalize the solution to cover more than two jobs.

The proposed model is tested using four jobs against three well-known other methods: Queueing theory, operational analysis and no model. Figure 2 shows a comparison against other three well-known methods: Queueing theory, operational analysis [5], and no model. The boxplot diagram in Figure 2 indicates the mean and median of the prediction error for data collected across all possible of four job combinations of the DaCapo benchmarks (e.g., xalan-xalan-xalan-batik, jython-luindex-xalan-batik, ...).

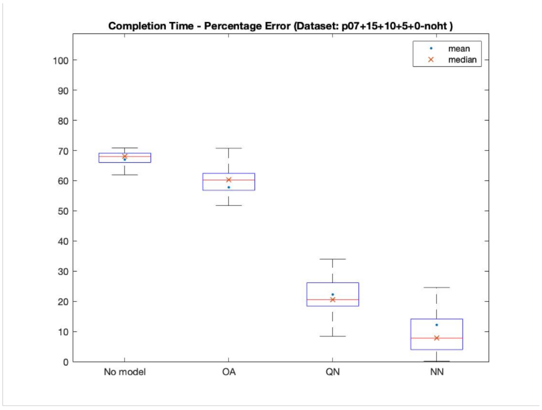


Figure 2: Predicting four jobs

The proposed solution significantly outperforms other methods by achieving 9% for mean percentage error compared with queueing theory, operational analysis, and no model, which achieve 21%, 15%, and 68%, respectively. The proposed tool not only predicts completion time for two jobs with high accuracy, but also saves considerable time in training the model to predict the time for more than two jobs and can easily be generalized to cover unforeseen workload combinations.

V. Conclusion

A completion time prediction method is developed, featuring increasing accuracy in return for more profiling data. The neural networks model seems more effective when the system uses or does not use job priorities or sleep time. The proposed solution not only predicts completion time for two jobs with high accuracy, but also saves considerable time in training the model to predict throughput for more than two jobs and easily can be generalized to cover unforeseen workload combinations.

Acknowledgment

This work was supported in part by the National Center for Artificial Intelligence and Big Data Technologies, King Abdulaziz City for Science and Technology (KACST), Saudi Arabia, and in part by the European Union's Horizon 2020 Research and Innovation Program, RADON, under Grant 825040.

REFERENCES

[1] X. Bu, J. Rao, and C.-z. Xu, "Interference and locality-aware task scheduling for mapreduce applications in virtual clusters," in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, 2013, pp. 227–238.

[2] B. Sharma, T. Wood, and C. R. Das, "Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 102–111.

[3] C. Delimitrou and C. Kozyrakis, "Quasar: resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[4] S. M. Blackburn, R. Garner, C. Hoffmann, A. M. Khang, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer *et al.*, "The dacapo benchmarks: Java benchmarking development and analysis," in *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, 2006, pp. 169–190.

[5] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.