Performance Prediction Techniques for Scalable Large Data Processing in Distributed MPI Systems

Janki Bhimani
Northeastern University
Email: bhimani@ece.neu.edu

Ningfang Mi
Northeastern University
Email: ningfang@ece.neu.edu

Miriam Leeser
Northeastern University
Email: mel@coe.neu.edu

***Abstract*—**

Predicting performance of an application running on parallel computing platforms is increasingly becoming important due to the long development time of an application and the high resource management cost of parallel computing platforms. However, predicting overall performance is complex and must take into account both parallel calculation time and communication time. Difficulty in accurate performance modeling is compounded by myriad design choices along multiple dimensions, namely (i) process level parallelism, (ii) distribution of cores on multi-processor platforms, (iii) application related parameters, and (iv) characteristics of datasets. This research proposes a fast and accurate performance prediction approach to predict the calculation and communication time of an application running on a distributed computing platform. The major contribution of our prediction approach is that it can provide an accurate prediction of execution times for new datasets which have much larger sizes than the training datasets. Our approach consists of two models, i.e., a probabilistic self-learning model to predict calculation time and a simulation queuing model to predict network communication time. The combination of these two models provides data analysts a useful insight of optimal configuration of parallel resources (e.g., number of processes and number of cores) and application parameters setting.

Keywords*—*Performance Modeling, Queuing Networks, Markov Model, Distributed Systems

I. Introduction

High Performance Computing (HPC) systems are ubiquitous in processing data for myriad applications involving huge datasets. How to achieve the best performance with an optimal configuration of parallel resources (e.g., number of processes and number of cores) is a challenging research problem. Currently, researchers run their application codes on a typical dataset, fix application parameters, and try different configurations of parallel resources to determine the optimal one. However, if we further want to find optimal application parameters, then the investigation needs to consider all possible combinations of application parameters and parallel resources. Such an investigation becomes very expensive, requiring a large amount of time and hardware resources. In addition, on parallel computing platforms, using more parallel resources does not always guarantee performance improvement. Hence, it could be beneficial if we can predict the optimal performance point in terms of parallel resources, application parameters, and datasets.

As a motivation, we plot an example with a K-means clustering application in Figure 1. We observe that the calculation time decreases when we have more Message Passing Interface (MPI) processes; however, the time to communicate data increases. This implies that speeding up parallel calculation time may not guarantee the overall application speed-up. Thus, predicting both calculation and communication times individually is important to make good design choices.

In this paper, we develop a new performance modeling approach, named FiM, to predict both computation and communication times of a data processing application. One of the key innovations in our work is that FiM relies only on small datasets for training but can accurately predict the execution times for larger datasets.

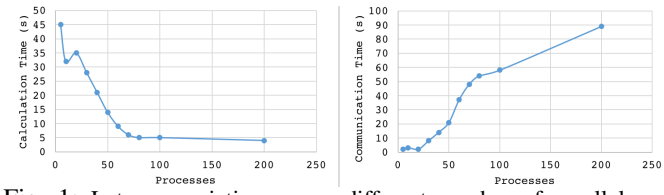

Fig. 1: Latency variation across different number of parallel processes for (a) calculation time (b) communication time

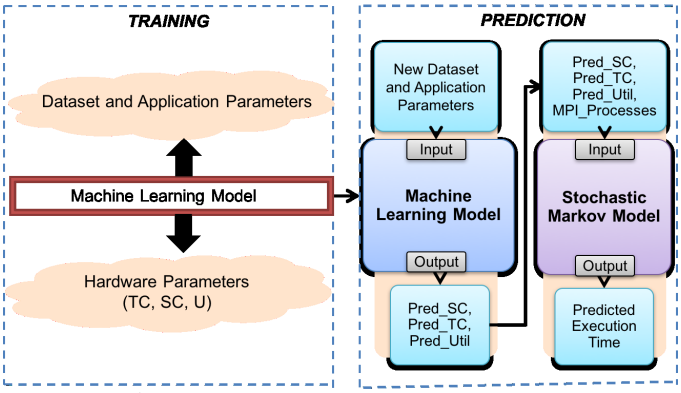

Fig. 2: Development procedure of FiM_Cal

We present two main components in FiM: (1) FiM_Cal, and (2) FiM_Com. The goal of FiM_Cal is to predict the calculation time by using a stochastic Markov model and a machine learning model. The goal of FiM_Com is to predict the communication time using a set of simulation queuing models.

II. FiM_Cal: Calculation Prediction

In this section, we present FiM_Cal, an analytical approach to predict the calculation time of an application running on a distributed platform. FiM_Cal consists of two key components, i.e., a stochastic Markov model and a machine learning model. First we use the stochastic Markov model to represent the computational processing of an application. Then, we design a machine learning algorithm to emulate hardware characteristics for estimating the hardware related inputs of our stochastic Markov model. This allows our model to predict the calculation time of a new dataset or a new application parameter setting without any new instrumentations, which thus makes an improvement over other existing methods.

In summary, our machine learning model only relies on a small datasets as the training to predict dependent variables (such as hardware parameters stalled cycles (SC), total cycles (TC), utilization (U)) for a new large dataset on a new set of application parameters. These predicted hardware details can

978-1-5090-5252-3/16/$31.00 ©2016 IEEE

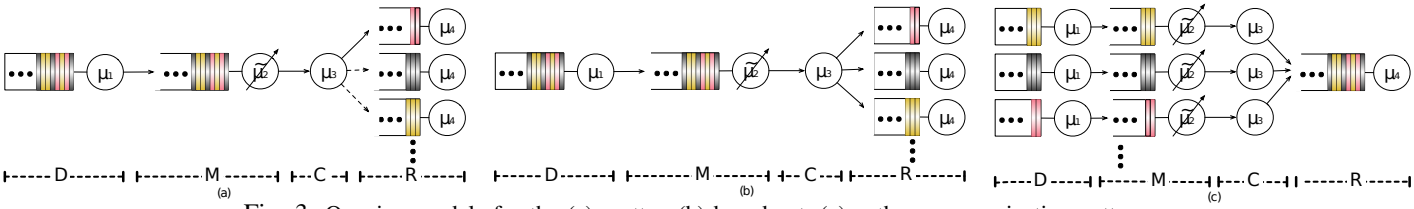

Fig. 3: Queuing models for the (a) scatter, (b) broadcast, (c) gather communication patterns

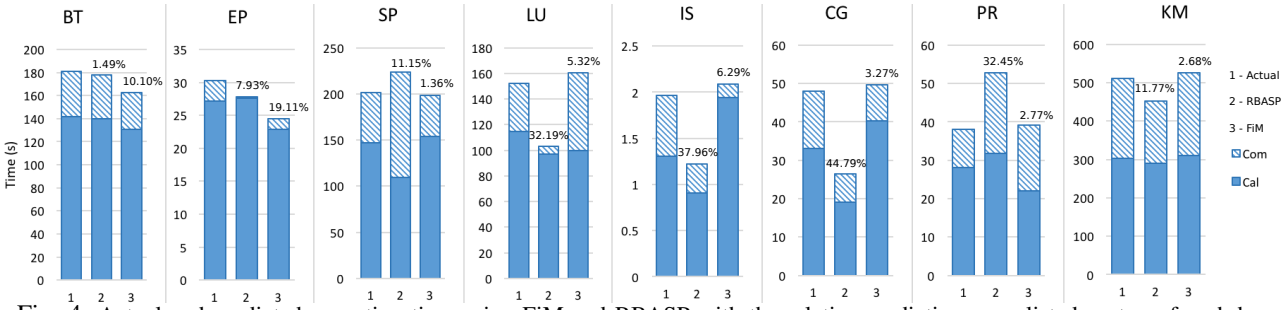

Fig. 4: Actual and predicted execution time using FiM and RBASP with the relative prediction error listed on top of each bar

then be used as an input to our stochastic Markov model to predict the calculation time for an application under this new large dataset and new application parameters without performing additional instrumentations. Figure 2 shows the overall procedure of our prediction model FiM_Cal, including the training and prediction components.

III. FiM_Com: Communication Prediction

Many data processing applications are known to be communication bound [1]. Therefore, another important aspect of our performance modeling is to predict communication time, i.e., the runtime of *data transfer*, for those applications running on a distributed system.

A. Communication Patterns

It is non-trivial to predict communication time, due to the non-deterministic latency of the communication network. In this work, we investigate various communication patterns such as uplink and downlink and build different queuing models to capture those patterns when running a data processing application such as K-means clustering on a multi-core system with MPI. Specifically, we consider communications from the master to other compute nodes as the *downlink*. Such a downlink communication can have either the scatter pattern or the broadcast pattern. Under the scatter pattern, the data is distributed among the compute nodes by the master such that each compute node gets a unique part of the data. Under the broadcast pattern, the data is broadcast to all compute nodes and each compute node receives the same copy of the data. The round robin policy is usually used in MPI to schedule packet transfer among compute nodes under the scatter communication. We also consider communications from a compute node to the master as the uplink gather pattern. When gathering, each compute node sends their own data to a shared buffer, and the master reads those data one-by-one from the buffer.

B. Communication Models

We develop a set of queuing models to capture various communication patterns. Such queuing models can give an abstraction of real communication systems in terms of packet arrival rates, delay/waiting time and packet transfer rate, which helps to estimate the overall data communication time.

Figure 3 presents our queuing models to represent the scatter, broadcast and gather patterns, respectively. Typically, communication time depends on various network properties such as initialization cost, maximum network bandwidth, network load and the amount of metadata (e.g., headers, acknowledgments and flags). Therefore, in each of these models, we use four components, i.e., `D`, `M`, `C` and `R` in Figure 3, in serial to capture different properties of network communication. Each job in the queue of `D`, `M`, `C` and `R` represents a data packets e.g., a pixel for K-means clustering.

IV. Evaluation

In this section, we evaluate our FiM prediction approach (a combination of FiM_Cal and FiM_Com), by comparing the predicted results (e.g., end-to-end execution time) with actual experimental measurements on a real distributed platform. We also compare our prediction approach with an existing work named RBASP [2]. Figure 4 shows the predicted results using RBASP and FiM for six NPB benchmarks and two iterative data processing applications. As shown in Figure 4, our FiM approach achieves a good agreement between the predicted and actual results for both calculation and communication times across all the six benchmarks and two applications.

V. Acknowledgements

This work was partially supported by National Science Foundation Career Award CNS-1452751 and AFOSR grant FA9550-14-1-0160.

References

[1] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[2] B. J. Barnes, B. Rountree, D. Lowenthal, J. Reeves, B. De Supinski, and M. Schulz, "A regression-based approach to scalability prediction," in *Proceedings of the 22nd annual international conference on Supercomputing*. ACM, 2008, pp. 368–377.