# Machine-Learning Based Performance Estimation for Distributed Parallel Applications in Virtualized Heterogeneous Clusters

Seontae Kim, Nguyen Pham, Woongki Baek and Young-ri Choi
School of Electrical and Computer Engineering
Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea
Email: {stkim,nguyenpham,wbaek,ychoi}@unist.ac.kr

***Abstract*—In a virtualized heterogeneous cluster, for a distributed parallel application which runs in multiple virtual machines (VMs) concurrently, there are a huge number of possible ways to place its VMs. This paper investigates a performance estimation technique for distributed parallel applications in virtualized heterogeneous clusters. We first analyze the effects of different VM configurations on the performance of various distributed parallel applications. We then present a machine-learning based performance model for a distributed parallel application. Using a heterogeneous cluster with two different types of nodes, we show that our machine-learning based models can estimate the runtimes of distributed parallel applications with modest error rates.**

## I. Introduction

In a virtualized heterogeneous cluster, a distributed parallel application is executed on a *virtual cluster* (VC), which consists of multiple VMs, concurrently. Placing VMs for a distributed parallel application is challenging, since there are a huge number of possible ways to place the VMs. The performance of a distributed parallel application varies widely depending on the configurations of VMs such as the number of the different node types used for the VMs, the number of the VMs running on each type of nodes, and the number of the VMs running on the same node. Moreover, the performance can be dramatically impacted by an application running together on the same physical node due to the interference effects between them.

There have been earlier efforts to investigate scheduling techniques which take into account heterogeneity of hardware configurations and/or interference between applications. However, scheduling single node applications is mainly studied [1]. For a distributed parallel application, the heterogeneity of a cluster is not considered [2]. For applications like distributed analytics frameworks and latency critical services, a QoS-aware management system is proposed, but the placement of a VC is not focused [3].

In this paper, we investigate a performance estimation technique for distributed parallel applications in virtualized heterogeneous clusters. We first analyze the effect of different VM configurations on the performance of various distributed parallel applications. We then present a machine-learning based performance model for a distributed parallel application.

TABLE I
VC configurations used in our experiments

| Config | # of VMs | | # of Hosts (# VMs per Host) | |
|---|---|---|---|---|
| | T1 | T2 | T1 | T2 |
| C1 | 8 | 0 | 8 (1) | 0 |
| C2 | 8 | 0 | 4 (2) | 0 |
| C3 | 4 | 4 | 4 (1) | 4 (1) |
| C4 | 4 | 4 | 2 (2) | 2 (2) |
| C5 | 0 | 8 | 0 | 4 (2) |
| C6 | 0 | 8 | 0 | 2 (4) |

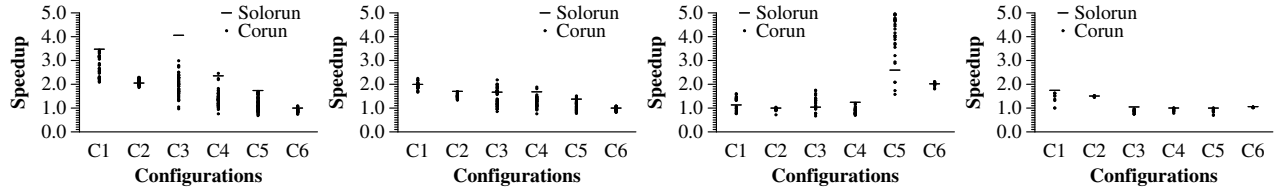
Using a heterogeneous cluster, we show that our machine-learning based models can estimate the runtimes of distributed parallel applications with modest error rates.

## II. Performance Analysis

**Experimental Setup** We use a heterogeneous cluster which consists of two types (T1 and T2) of nodes connected via 1 GE switch. Type T1 is configured with one Intel quad-core I7-3770 (IvyBridge) 3.40GHz, and 16GB memory. Type T2 is configured with two Intel hexa-core E5-2620 (SandyBridge) 2.0GHz, and 64GB memory. We use eight T1 nodes and four T2 nodes. For type T2, each node has 12 cores but we use only 8 cores, 4 cores per each socket. Xen hypervisor version 4.1.4 is installed in each of the physical nodes. For a VM, it is configured with two virtual CPUs and 5GB memory. Each parallel application is configured with 8 VMs.

**Effects of Different VC Configurations** In the above setup, the total number of different VC configurations even for a single parallel application without considering the placement of co-runners is 80. Due to the limitation on the time and cost of running experiments, we consider six VC configurations in Table I for our experiments.

Figure 1 shows the speedup of each of four distributed parallel applications, Grep and TeraGen from Spark, Namd, and 132.Zeusmp2 from SpecMPI 2007, over the six configurations. In the figure, mark "-" represents the runtime of its *solo run* without any co-runner, and mark circle represents the runtime with some co-runner(s) in each VC configuration. For each of the results, the *speedup* of an application in each run is computed as the runtime of the application in the run over the worst runtime among its solo runs in the six VC




(a) Grep (b) TeraGen (c) Namd (d) 132.Zeusmp2

Fig. 1. Performance of parallel applications on various VC configurations

configurations. From the results, we easily observe that the performance of an application varies depending on not only a VC configuration but also co-running VMs, and the degree of their effects is different for each application.

We also investigate the correlation between the resource usage patterns and performance of the parallel applications. For all the applications used in experiments (except TeraGen), we observe that their performance has a strong correlation with the usage of one resource type, either network I/O or storage I/O. In case of TeraGen, there is a strong correlation of its speedup with both of network I/O and storage I/O.

## III. Performance Estimation

Searching the optimal placement of a parallel application in a heterogeneous cluster against all the possible placements is infeasible. To make the searching process tractable, for each parallel application, we only consider *homogeneous VC configurations* which use only one type of nodes for its VMs, and *symmetric heterogeneous VC configurations* which use different types of nodes as in Table I. For both of the VC configurations, we only use ones in which the number of the VMs in each node used for the application is equal. Moreover, we restrict the placement of VC configurations for parallel applications such that each VC configuration is divided into two parts (i.e. 4 VMs, in each part, which are executed on the nodes withe the same type), and each of the VMs in the same part has the same set of co-runner(s).

On profiling a parallel application, we reduce profiling runs significantly by only using a synthetic workload that mostly consumes the dominant resource (which strongly impacts on the application performance). We implemented two types of synthetic workload generators that can have different intensity levels for network and storage resources.

For a candidate VC configuration, we profile the runtime of each application alone, and also the runtimes with synthetic workloads. We have 50 and 100 profiling runs with co-runners for the storage and network intensive applications, respectively. Note that a network synthetic workload uses the CPU resource in two levels, since the performance of a network intensive application is affected by the CPU usage of co-runners. While profiling, we measure the performance metrics including CPU idle %, send and receive for network I/O (in MB/s), and read and write for storage I/O (in MB/s), for dom0 and VMs of the target application.

In a performance model for a target application $A_{target}$, we use the inputs including a target VC configuration $VC_{target}$, a runtime and metrics of the solo run of $A_{target}$ measured on $VC_{target}$, and a placement configuration and measured metrics of the solo run for each co-runner. We used a machine-learning technique called REPTree in *Weka* [4]. Note that we use a regression tree mode which predicts the output in a real number to estimate the runtime.

TABLE II
Validation results for parallel applications using REPTree

| Application | Error rate(%) | # test cases |
|---|---|---|
| 132.Zeusmp2 | 13.19 | 50 |
| Namd | 15.14 | 100 |
| Grep | 35.69 | 200 |
| TeraGen | 26.85 | 150 |

## IV. Experimental Results

Table II shows the average error rate of the runtime estimation using REPTree for each of the four parallel applications. The error rate in each test case is computed as $\mid r_{est} - r_{act} \mid / r_{act} \times 100$, where $r_{est}$ and $r_{act}$ are the estimated and actual runtimes, respectively. For a parallel application, the performance model is built using profiled runs against synthetic workloads, and the model is validated by using the experiment results of the application running together with other parallel applications in our heterogeneous cluster. The number of test cases for each application is also given in the table. The average error rate is 22.72%.

## Acknowledgments

This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. R0190-16-2012, High Performance Big Data Analytics Platform Performance Acceleration Technologies Development) and the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2015R1C1A1A02037400).

## References

[1] Christina Delimitrou and Christos Kozyrakis. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2013.

[2] Jaeung Han, Seungheun Jeon, Young-ri Choi, and Jaehyuk Huh. Interference management for distributed parallel applications in consolidated clusters. In *Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.

[3] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and QoS-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.

[4] Eibe Frank, Mark A. Hall, and Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 4 edition, 2016.