Predictive Modeling and Scalability Analysis for Large Graph Analytics

Sourav Medya[1,2], Ludmila Cherkasova[2], and Ambuj Singh[1]

[1]University of California, Santa Barbara, USA, medya@cs.ucsb.edu, ambuj@cs.ucsb.edu
[2] Hewlett Packard Labs, Palo Alto, USA, lucy.cherkasova@hpe.com

Abstract*—Many HPC and modern large graph processing applications belong to a class of scale-out applications, where the application dataset is partitioned and processed by a cluster of machines. Assessing the application scalability is one of the primary goals during such application implementation. Typically, in the design phase, programmers are limited by a small size cluster available for their experiments. Therefore, predictive modeling is required for the analysis of the application scalability and its performance in a larger cluster. While in an increased size cluster, each node will process a smaller portion of the original dataset, a higher communication volume between a larger number of nodes may cripple the application scalability and provide diminishing performance benefits. One of the main challenges is the analysis of bandwidth demands due to an increased communication volume in a larger size cluster. In this paper[1], we introduce a novel regression-based approach to assess the scalability and performance of a distributed memory program for execution in a large-scale cluster. Our solution involves 1) a limited set of traditional experiments performed in a small size cluster and 2) an additional set of similar experiments performed with an "interconnect bandwidth throttling" tool, which exposes the bandwidth impact on the application performance. These measurements are used in creating an ensemble of analytical models for performance and scalability analysis. Using a linear regression approach, step by step, we incorporate into the model the following important parameters: *i)* the number of cluster nodes and application processes, *ii)* the dataset size, and *iii)* interconnect bandwidth. We demonstrate our solution, its power, and accuracy using a popular Graph500 benchmark, which implements a Breadth First Search algorithm on large, synthetically generated graphs. By utilizing measurements collected in a 32-node cluster, we are able to project the program performance in a large size cluster with hundreds of nodes. The proposed approach and derived models help to provide an early feedback to programmers on the scalability and efficiency of their solution.

I. Introduction

During the past decade, graph algorithms have received much attention and became increasingly important for solving many problems in social networks, web connectivity, scientific computing, data mining, and other domains. The sizes of analyzed graphs have grown significantly: the graphs with billions of vertices and hundreds of billions of edges set new processing frontiers. A traditional way for improving graph application performance is to store and process its working set in memory. As the problem size increases and it cannot fit into memory of a single server, the distributed computing and memory resources are required for holding the entire dataset in memory and processing it.

Message passing interface (MPI) is a standard programming paradigm for scale-out, distributed memory applications. Performance of such applications inherently depends on performance of communication layer in the cluster. Over the past decade, traditional networking often gets replaced by high-speed interconnects with Remote Direct Memory Access (RDMA) technology for optimizing performance of distributed memory applications. During the last couple of years, many Big Data applications, such as Hadoop, Spark, Memcached, etc., were re-written to take advantage of high-performance RDMA-capable interconnects [1], [2], [3], [4] which provide fast and high-bandwidth communications.

Designing and implementing an efficient and scalable graph processing application is a challenging task. Complex MPI-based programs interleave computations and communications which makes it difficult to perform an accurate analysis of communication layer impact on application performance and predict scaling properties of the program. The scalability problem had existed for decades and some elaborate and sophisticated ensembles of tools and simulators [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] were proposed by HPC community to attack this challenging problem.

Typically, during the initial implementation and debugging phases, a programmer is limited to experiments in a small size cluster for application testing and profiling. The unavailability of large scale execution environment makes it difficult (if not impossible) to evaluate the scaling characteristics of the designed program.

In this work, we discuss a novel approach for predictive modeling and scalability analysis of large graph analytics. We consider the Graph500 benchmark [15], recently introduced by HPC community for measuring and comparing computer's performance in memory retrieval. It implements a Breadth First Search algorithm on graphs and uses as an input synthetically generated, scale-free graphs, which could be easily scaled to extremely large sizes. Using a limited set of traditional experiments in a small size cluster, it is very difficult to assess and predict the impact of interconnect bandwidth on the application performance at scale. Typically, in a small cluster, the communication volume generated by the application is yet small and the interconnect bandwidth is plentiful. Therefore, the interconnect contention does not manifests itself in an observable way. For assessing the application bandwidth demands and their

[1]The paper was originated and partially completed during Sourav Medya's internship at Hewlett Packard Labs.

scaling trends, we perform an additional set of experiments with the "interconnect bandwidth throttling" tool [16], which helps to expose communication demands of the program and reveals the impact of limited interconnect bandwidth on the application performance. Our earlier work-in-progress report [17] sketches the initial approach and provides some preliminary evidence of its effectiveness.

In this paper, we design *an ensemble of analytical models* for scalability and performance analysis of underlying program in a large scale environment and provide a detailed evaluation of these models. We start with creating *a base linear regression model*, which predicts the program completion time as a function of the cluster size and the number of application processes, for a given (fixed size) input dataset. Then by combining collected data for different sizes of input datasets, we derive *a generalized linear regression model*, which predicts the program completion time as a function of the dataset size and the cluster size. Both of these models do not reflect the impact of interconnect bandwidth on the program completion time since the available interconnect bandwidth in the small size cluster is plentiful for the program execution. Finally, by using the experimental data with the "interconnect bandwidth throttling" tool, we derive and incorporate the impact of interconnect bandwidth into *a refined regression model*. This refined model provides a more accurate performance and scalability projection of the application behavior in a large-scale environment.

II. Critical Factors

In this work, we focus on the performance and scalability analysis of large graph analytics which utilizes a distributed memory framework. Despite a diversity of graph applications, they exhibit a set of unique, common characteristics discussed in literature [18]: graph computations are data-driven, they are memory intensive with random data access pattern. We demonstrate the problem and our approach by considering the Graph500 benchmark [15], which implements a Breadth First Search (BFS) algorithm.

For analyzing a program performance and its execution efficiency on a large-scale distributed cluster, the following factors are critically important (see Figure 1):

1. **Algorithm Efficiency**: Graph problems could be implemented in many different ways in terms of graph data partitioning for parallel processing as shown in the top layer of Figure 1. Data partitioning and algorithm details play an important role in scalability analysis as it impacts algorithm's communication style and the communication volume (in a distributed scenario), and thus the application completion time. There are a few well-known data partitioning approaches, such as *1-D (vertex-based)* and *2-D (edge-based)* partitioning algorithms proposed for parallel processing of BFS [19]. These algorithms have very different communication patterns, which impact their performance and scalability. The 2-D partitioning algorithm was theoretically proven to be a scalable algorithm [19].

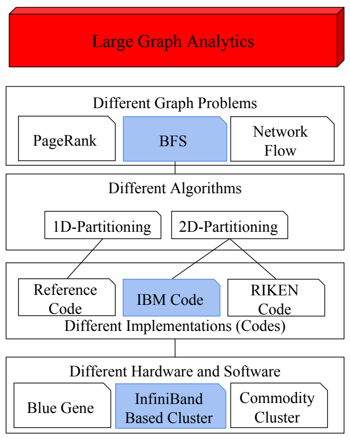


Figure 1. Performance and scalability: example of critical factors.

2. **Implementation Efficiency**: Program performance and its scalability further depends on the code that implements the selected/designed algorithm. This is the middle layer shown in Figure 1. In spite of excellent theoretical properties of the 2-D partitioning algorithm, its inefficient implementation may result in a poorly performing and badly scaling program. Therefore, the efficient implementation is a critical part of application performance and scalability.

3. **Underlying System Hardware and Software**: Finally, the underlying system hardware, that is designated for program execution, is extremely important for the program performance and its scalability as shown by the bottom layer in Figure 1. Specially designed systems, such as Blue Gene and K Computer, have proprietary, custom-built interconnects which provide enhanced support for MPI collectives, and therefore, demonstrate superior performance compared to commodity (network-based) clusters and more advanced InfiniBand-based ones.

Therefore, the scalability analysis and performance prediction depends on the underlying graph problem/algorithm, its parallel implementation, and the underlying system software and hardware. Figure 1 illustrates these critical factors with some examples from each category. In the paper, we demonstrate our approach by using the 2-D partitioning IBM implementation of BFS algorithm [15] executed in the InfiniBand-based cluster (shown by blue boxes in Figure 1).

III. Our Approach

Before presenting the overview of our approach and explaining the intuition behind it, we provide some details of Graph500 benchmark and our experimental environment.

Graph500 Benchmark [15]: It was introduced for measuring a computer's performance in memory retrieval. It implements a breadth-first search (**BFS**) algorithm in undirected large graphs generated by a scalable, synthetic data generator based on a Kronecker graph. The graph size is described by the **scale** s. Scale s defines the graph with 2^s vertices and $16 \cdot 2^s$ edges. For example, graph of scale 27 has 134 Million vertices and 2.1 Billion edges, graph of scale 28 has 268 Million vertices and 4.2 Billion edges, etc.

The benchmark performs a **BFS** of graph vertices from a randomly chosen vertex in the graph. Graph500 benchmark measures the elapsed times of 64 BFS runs (from randomly chosen 64 initial source vertices). Benchmark performance is measured in the achieved throughput of *traversed edges per second* (**TEPS**).

Currently, there are at least six different implementations of Graph500. In our work, we use the MPI-based IBM implementation, which utilizes the 2-D (edge-based) data partitioning algorithm. The released code supports a non-power-of-2 process count, where *the total process count* **must be** *of the form:* $2 \cdot (p^2)$ for some positive integer p. This formula sets strict rules on the configurations that can be used in our scalability analysis.

Experimental Testbed: In our experiments, we use a *32-node cluster* connected via FDR InfiniBand (56 Gbits/s). Each node is based on HP DL360 Gen9 servers with two sockets, each with 14 cores, 2 GHz, Xeon E5-2683 v3, 35 MB last level cache size, and 256 GB DRAM per node. The combined amount of DRAM in the cluster is 8 TB.

For scalability analysis, we are interested in a ***strong scaling*** formulation, where the problem data size is kept fixed and the number of processors (or nodes) to execute the program is increased. In order to satisfy the constraints of the process count of the IBM's Graph500 code, we use the *following configuration* in our study: each node is configured with *18 MPI processes* and *1 thread* per process. For this configuration, we could execute Graph500 benchmark in the cluster with *4, 9, 16,* and *25* nodes, which implies up to $25 * 18 = 450$ processes or cores. The maximum graph size in the experiments is scale of 30, i.e., 1072 Million vertices and 16.8 Billion edges.

Our Modeling Approach: In our 32-node experimental cluster, we could collect measurements based on four possible cluster sizes and different graph dataset scales. Since each benchmark execution provides 64 BFS runs, the collected measurements form a representative training set for designing the regression-based models of application performance in the experimental (small size) cluster.

First, we derive *a base performance model* for a given dataset size. Then, we present a more general model, which incorporates the graph dataset size as a parameter.

However, the set of collected measurements in the small 32-node cluster does not reflect the performance impact of the interconnect bandwidth for configurations with larger numbers of nodes and increased communication volumes. Therefore, the derived base performance model might be "idealistic" and "overly optimistic" in predicting the scaling characteristics of the studied application. To compensate for a lack (unavailability) of performance measurements related to the interconnect bandwidth impact in a larger cluster, we perform an additional set of experiments with the interconnect bandwidth throttling tool *InterSense* [16]. This tool helps to expose the communication demands of the program and to analyze the impact of limited interconnect bandwidth on the application performance. These experiments enable us to design *a refined analytical model* that incorporates the interconnect bandwidth as a parameter to reflect its impact on application scalability and performance.

IV. The Ensemble of Performance Models

In this section, we introduce the ensemble of analytical models for evaluating the application scalability and performance, and discuss the models' advantages and trade-offs.

A. Linear Regression Models

In a general case, the *Completion Time (***CT***)* of a distributed memory program can be modeled as follows:

$$CT = ProcessingTime + CommunicationTime$$

As the number of processors in the cluster is increased to p, one would expect that *ProcessingTime* in this equation will improve by p times. With the assumption that the data is evenly distributed over the processes, the processing time can be approximated as $O(\frac{1}{p})$.

To estimate the *CommunicationTime* of a distributed memory program as a function of number of processors p is a more challenging task, and often it relies on algorithm properties. A theoretical analysis of 2D partitioning implementation [19] provides the evaluation of its communication pattern: the number of messages per processor is $O(\sqrt{p})$. The expected evenly distributed data per process is $O(\frac{D}{p})$ or $O(\frac{1}{p})$ when data size D is fixed. We combine these asymptotic analysis (messages and data per process) and formulate communication time as $O(\frac{1}{\sqrt{p}})$. Here, if we drop one of the components from the formulation the quality of the model decreases for obvious reasons. Note that one might formulate the CT in a different way based on the different algorithmic properties of the considered graph problem and its solution.

Since we execute 18 MPI processes per node, we use $p = 18n$, where n represents the number of nodes in the cluster. For presentation simplicity, we omit the additional constants 18 and $\sqrt{18}$ from the equations below.

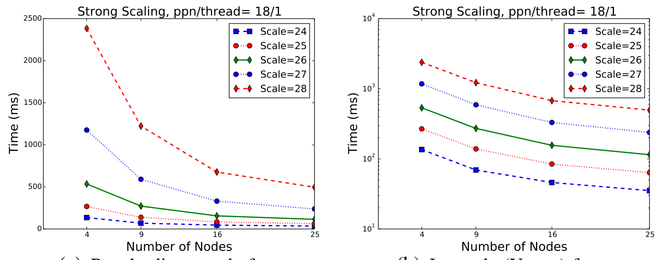
Base Linear Regression Model: we can derive the *CT* as linearly dependent on $\frac{1}{p}$ and $\frac{1}{\sqrt{p}}$ or $\frac{1}{n}$ and $\frac{1}{\sqrt{n}}$:

$$CT(p) = C_1^p * \frac{1}{p} + C_2^p * \frac{1}{\sqrt{p}} \quad (1)$$

$$CT(n) = C_1^n * \frac{1}{n} + C_2^n * \frac{1}{\sqrt{n}} \quad (2)$$

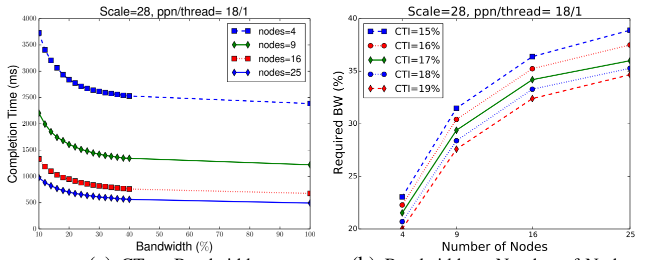
Figure 2 (a) shows measured CTs of Graph500 benchmark executed in our cluster for different graph sizes (defined by *scale* s) and a number of nodes in the cluster.

Note, that Figure 2 (b) shows the same measurements with Y-axes in logscale format. If $CT(n)$ would scale as $O(\frac{1}{n})$ (i.e., no communication overhead) then in Figure 2 (b) one can expect a straight line with a negative slope near 1 [20]. However, it is not the case, and the communication time represents an essential component of the overall CT of Graph500 execution.



(a) Regular linear-scale format. (b) Logscale (Y-axes) format.

Figure 2. The Graph500 CT in a strong scaling scenario.



(a) CT vs Bandwidth (b) Bandwidth vs Number of Nodes

Figure 3. For scale 28: (a) Bandwidth Impact on the CT and (b) Interconnect Bandwidth Demands.

Eq. 2 provides the problem formulation for *CT* as linearly dependent on $\frac{1}{n}$ and $\frac{1}{\sqrt{n}}$, where C_1^n and C_2^n are coefficients which need to be derived from the asymptotic analysis. We aim to find these coefficients via linear regression. Using a small size cluster with N nodes in total we obtain measured CTs for BFS code on all possible sub-cluster configurations with n nodes ($n \le N$). So, we have data points as a pair, $(time,\ number\ of\ nodes)$. We use these experimental data in the set of equations (as shown below) and solve this set for finding the coefficients C_1^n and C_2^n via linear regression:

$$\begin{aligned} CT_1(n_1) &= C_1^n * \frac{1}{n_1} + C_2^n * \frac{1}{\sqrt{n_1}} \\ CT_2(n_2) &= C_1^n * \frac{1}{n_2} + C_2^n * \frac{1}{\sqrt{n_2}} \\ \cdots \quad & \cdots \quad \cdots \quad \cdots \end{aligned}$$

where CT_i is the corresponding CT when n_i nodes are used.

A popular method for solving such set of equations is Non-negative Least Squares Regression, which we use here. In statistics, this is an approach for modeling the relationship between a scalar dependent variable (e.g., *CT* here) and one or more independent variables (e.g., $\frac{1}{n}$ and $\frac{1}{\sqrt{n}}$). The set of coefficients C_1^n and C_2^n is the model that describes the relationship. The designed *base linear regression model* enables the prediction of the *CT* as a function of the number of nodes in the cluster.

Generalized Base Linear Regression Model: Now, we generalize our model to describe *CT* as a function of data size as well. Note, the data size, D is interpreted by scale s. Eq. 2 describes the behavior for different (fixed) scales. Graph of scale $s+1$ has twice the number of vertices and edges than the graph of scale s. Assuming the data is evenly distributed over the nodes in the cluster, we derive the following equation as a generalization of Eq. 2.

$$CT(n, D) = C_1^{n,D} * \frac{D}{n} + C_2^{n,D} * \frac{D}{\sqrt{n}} \tag{3}$$

This *generalized model* enables predicting the application scalability and performance for different cluster sizes and graph sizes. This model is more general than the base model described by Eq. 2. However, for a particular dataset size, Eq. 2 is more precise and accurate. So, there is a trade-off between precision and generality in these two models.

B. Estimating The Communication Bandwidth Demands

With an increased number of cluster nodes the communication volume becomes a dominant component in Eq 1. Past literature [20] shows that in a strong scaling scenario, the program performance gets additionally impacted when system interconnect bandwidth starts affecting the communication time. We need to assess the increased bandwidth demands of a communication volume as a function of the increased cluster size. Unfortunately, there are no existing tools or common approaches to analyze the utilized (required) interconnect bandwidth during the execution of general MPI program. It is a very challenging task due to a variety of existing MPI collectives and MPI calls that could involve different sets of nodes and communication styles.

To overcome this challenge, we apply *InterSense* [16] - a special interconnect emulator, which can control (throttle) the interconnect bandwidth to determine how much bandwidth the program needs before its completion time becomes impacted. It enables us to accurately estimate the required (needed) bandwidth by the program.

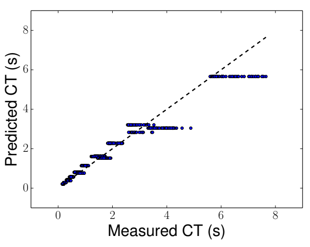
Fig. 3(a) shows the outcome of bandwidth throttling experiments for a graph scale 28. Each line shows the CTs at different (controlled by *InterSense*) interconnect bandwidth percentage. These plots show that there is a non-linear relation between Graph500 completion times and available interconnect bandwidth. To get accurate estimates on the required interconnect bandwidth, we experiment with 2% interval in the range $10-40\%$. As expected, the CTs are higher with larger scales and smaller process configurations.

For a particular configuration, we define *Completion Time Increment (**CTI**)* at an available bandwidth (bw) percentage as the increment in percentage w.r.t the CT when 100% bandwidth is available.

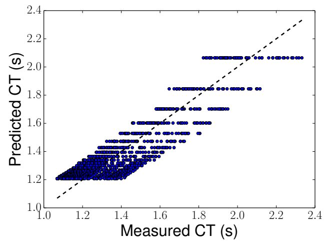
$$CTI_{bw} = \frac{CT\ at\ bw - CT\ at\ 100}{CT\ at\ 100} \tag{4}$$

CTI basically reflects how far the CT at an available bandwidth percentage is from the expected CT at the full bandwidth (100%) available. Figure 3(a) shows that different cluster configurations have different CTI when the available interconnect bandwidth is varied.

Bandwidth Demand ($BW_{CTI,n}$) is defined as the percentage of bandwidth required to achieve the predefined CTI for a particular number of nodes configuration n.



(a) At Bandwidth 20%



(b) Eq. 5

Figure 4. (a) CTs with respect to number of nodes and data size at smaller available bandwidth. (b) Regression model by Eq. 5: relation between CT as a function of interconnect bandwidth.

Figure 3(b) shows that for a particular CTI, the bandwidth demands $BW_{CTI,n}$ are increasing with increment in the number of nodes (we show the results for scale 28).

The analyzed bandwidth demands $BW_{CTI,n}$ for other graph scales show a similar increasing bandwidth demands trend. Therefore, we need to model the impact of increasing communication volumes on the Graph500 performance in a larger cluster. This can help in estimating the cluster size, where a communication cost becomes a highly dominant component, at which point the performance (scalability) benefits in the further increased cluster would lead to a diminishing return.

C. Completion Time With Different Available Bandwidth

From the measurements in the previous section, two following facts are evident. First, *CT* depends on the available interconnect bandwidth. Second, the communication volume in an increased size cluster results in a higher bandwidth demand. Let us look into dependency between *CT*, data size, and a number of nodes at a given (fixed) bandwidth. As we decrease the bandwidth with the bandwidth throttling tool, the question is whether Eq. 3 is valid for describing *CT* at any fixed bandwidth or not.

Figure 4(a) shows the regression results for Graph500 measurements obtained with a decreased interconnect bandwidth of 20% (using a bandwidth throttling tool) for the model defined by Equation 3. The measurements are collected from the experiments with four different cluster sizes (number of nodes $4, 9, 16, 25$) and graph scales $26-29$.

In the experiments we use cross validation. At random, $4/5$ of the data points are used for training and the rest for prediction. Two well studied measures to show quality of regression are R^2 and **"mean squared error" (MSE)**. We describe these measures in more details in Section V. We get R^2 as 0.976 and 0.972 (**close to** 1 **is better**) for bandwidth of 10% and 20% respectively. The corresponding MSEs (**close to** 0 **is better**) are 0.17 and 0.13. Both types of errors reflect a high quality of regression results. We experiment with other available bandwidth percentage and get similar results. From these results, we conclude that at a particular (fixed) bandwidth the CT model follows Equation 3.

The next question to answer is the following: when data size and number of nodes are fixed, what is the relationship between *CT* and available bandwidth? To answer this question we refer to Figure 3(a). CT has a non-linear (rather exponential) relationship with available bandwidth when cluster size and data size are fixed. So, we formulate the *CT* at an available interconnect bandwidth percentage (bw) in an exponential form, i.e., $O(\alpha^{\frac{100}{bw}})$. To predict the constant we use a regression approach and rewrite the equation as follows:

$$CT(bw) = C^{bw} * \alpha^{\frac{100}{bw}} \tag{5}$$

In Eq. 5, the coefficient C^{bw} defines the regression constant and *Exponential Factor (α)* defines the exponent component of the curve. We decide on a good value of α by running a binary search between 1 and 3 assuming that the exponent is not very high. The binary search optimizes α based on the quality of R^2 and MSE. To show that this formulation is of good quality, we perform regression with a cluster of 25 nodes and graph scale of 29. Figure 4(b) shows the regression results. We get R^2 and MSE as 0.8 and 0.012 respectively (where α is 1.07). These results show a good prediction of *CT* in terms of available bandwidth.

Refined Linear Regression Model: Now, we attempt to formulate *CT* in terms of available bandwidth percentage bw, a number of cluster nodes n, and a data size D. We combine Eq. 2 (or Eq. 3) with Eq. 5 in the following way:

$$CT(n, bw) = C_1^{n,bw} * \frac{1}{n} + C_2^{n,bw} * \frac{\alpha^{\frac{100}{bw}}}{\sqrt{n}} \tag{6}$$

$$CT(n, bw, D) = C_1^{n,bw,D} * \frac{D}{n} + C_2^{n,bw,D} * \frac{D * \alpha^{\frac{100}{bw}}}{\sqrt{n}} \tag{7}$$

Eq. 7 is a generalized version of Eq. 6 as it describes CT as a function of data size as well. The equations Eq. 6 and 7 describe the CTs as a function of cluster size and available interconnect bandwidth. Thus they represent a complete model of Graph500 CT (or application scalability) in terms of important system variables.

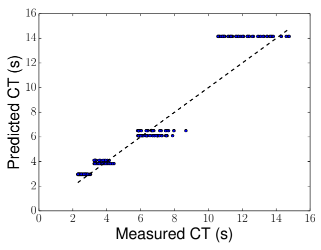
We summarize the derived regression and prediction models and their parameters in Table I. We describe the input variables, the predicted component, the factors which are fixed and the structure of the training data points for every model. Note, α should be tuned to reflect the desired performance impact for the models which take it as an input.

Models	Input Variable	Fixed	Predict	Training Data Point
Eq.2	n	bw, D	CT	(CT, n)
Eq.3	n, D	bw	CT	(CT, n, D)
Eq.5	bw, α	n, D	CT	(CT, bw)
Eq.6	n, bw, α	D	CT	(CT, n, bw)
Eq.7	n, D, bw, α	$-$	CT	(CT, n, D, bw)

Table I
SUMMARY OF DIFFERENT REGRESSION MODELS.

V. EVALUATION

In this section, we evaluate accuracy of regression for the ensemble of analytical models introduced in previous Section IV. Two of the important measures for evaluating the quality of regression are the following: "mean squared error" (MSE) and R^2. Then we analyze the application scalability and performance projections obtained using these models.


(a) Scale = 30

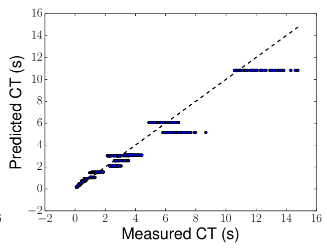

(b) All Scales $(26 - 30)$

Figure 5. Regression results for (a) Eq. 2, where the *CT* model includes the number of nodes as a parameter (for a fixed graph scale); and (b) Eq. 3, where the *CT* model includes the number of nodes and a graph scale as parameters.

A. *Accuracy of the Linear Regression Models*

The Base and Generalized Linear Regression Models: For a fixed problem data size in Graph500, we perform experiments with different number of nodes in our 32-node cluster (using the same configuration per node), collect measurement data, and then solve Eq. 2 with linear regression for finding coefficients C_1^n and C_2^n. The solution is based on collected measurements of approximately 250 data points. One run of Graph500 for one particular graph scale produces 64 BFS measurements, and it helps in creating a representative set of measurements.

Table II shows all the coefficients and the errors for these experiments and the derived *base linear regression model* defined by Equation 2.

Data Scale	C_1^n	C_2^n	R^2	MSE
26	30.4	0.98	0.94	0.002
27	68.66	1.92	0.95	0.008
28	138.59	4.004	0.97	0.03
29	326.12	9.598	0.95	0.399
30	629.38	29.03	0.96	1.735

Table II
ACCURACY OF REGRESSION FOR EQUATION 2.

The errors reflect a high quality of the regression results. The MSE measure highly depends on the numerical values of the results. As larger size graphs result in higher completion times, the deviation is larger. So, MSE becomes larger for higher graph scales. There are interesting observations about the computed coefficients C_1^n and C_2^n. We need to remind that in Eq. 2, there are two parts forming an overall Graph500 CT: a graph processing time at each cluster node (weighted by a coefficient C_1^n) and a communication time (weighted by a coefficient C_2^n).

The computed coefficient C_1^n is greater than C_2^n in all cases. This result shows that the processing time dominates the benchmark execution time (i.e., *CT*) in a small/medium cluster. At the same time, in a larger size cluster, the communication time starts to dominate the CT because n grows faster than $\sqrt{n}$ in the denominator of Eq. 2.

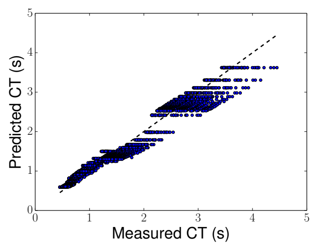
The ratio (C_1^n/C_2^n) decreases as the data size increases from scale 27 to scale 30. This explains that communication time component becomes more significant with a larger size.

Figure 5 (a) shows the regression results for problem scale 30 (scales $26 - 29$ are similar). Each point (x, y) in these figures presents a measured value on X-axes and the predicted value on Y-axes. As we can see, the designed *base linear regression model* fits well the collected measurements.

Figure 5 (b) shows the regression results for Eq. 3. This equation describes the *generalized base linear regression model* which can predict the Graph500 CT as a function of the cluster size and the graph size. This single model is derived using measurements from the experiments with different graph scales. The collected measurement set consists of approximately 1250 data points.

In these experiments, we use 2^{s-25} as a base for D, where s denotes the graph scale. Note, that graph with scale $s + 1$ is two times larger (in the number of vertices and edges) than the graph of scale s. We use a minimum scale of 26 in the experiments. So, 25 is assumed to be the base and the equations will have normalized constants. The computed coefficients $C_1^{n,D}$ and $C_2^{n,D}$ are 19.65 and 0.82. The corresponding R^2 and MSE are 0.98 and 0.406 respectively. These low errors indicate the good prediction capability of the model.

In conclusion, the model defined by Eq. 3 is a generalized version of the base linear regression model defined by Eq. 2. Note, the coefficients for these two models are quite different. Eq. 3 is more general as it can describe the Graph500 CTs for different graph scales, whereas Eq. 2 is specific to a particular graph size. On the other hand, MSE (0.406) for Eq. 3 is higher than Eq. 2 for scale 26, 27, 28 and 29. Apparently, generalization of the model gives lesser accuracy and inclusion of measurements for graph scale 30 in training and testing of the model (Eq. 3) makes a higher contribution in MSE.


(a) Scale 28

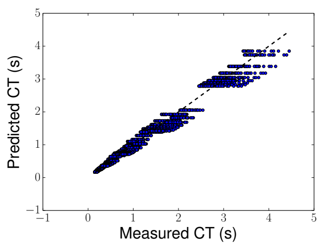

(b) Scale $26 - 28$

Figure 6. Regression results for (a) Eq. 6: the *CT* model includes the number of nodes and interconnect bandwidth as parameters, (b) Eq. 7: the *CT* model also includes the graph scale.

The Refined Regression Model: The *refined regression model* is derived with Eq. 6, where the *CT* is defined as a function of the cluster size and interconnect bandwidth, but it has a fixed graph size (scale).

This model is derived using Graph500 benchmark experiments with a fixed problem data size, different number of nodes (in our 32-node cluster), and different available bandwidth percentages enforced by the bandwidth throttling tool *InterSense*. To get a representative number of data

Data Scale	$C_1^{n,bw}$	$C_2^{n,bw}$	α	R^2	MSE
26	26.15	1.412	1.13	0.943	0.0025
27	59.49	2.845	1.12	0.952	0.01
28	118.026	5.968	1.11	0.969	0.0266
29	300.058	11.711	1.11	0.955	0.263

Table III
ACCURACY OF REGRESSION FOR EQUATION 6.

Data Scale	$C_1^{n,bw,D}$	$C_2^{n,bw,D}$	α	R^2	MSE
$26-28$	16.77	0.561	1.14	0.980	0.014
$26-29$	15.25	0.985	1.08	0.966	0.119

Table IV
ACCURACY OF REGRESSION FOR EQUATION 7.

points, we performed experiments from 10% to 50% of available bandwidth with a 2% interval.

We collected approximately 5200 measurement data points, and then solved Eq. 6 with linear regression for finding coefficients $C_1^{n,bw}$, and $C_2^{n,bw}$. The value of α is decided with binary search between 1 and 3 (assuming that the exponential component is a low constant, the low value is experimentally verified) based on optimizing the mentioned two types of errors. Figure 6(a) summarizes the regression results for the *Refined Regression Model* described by Eq. 6. Table III shows all the results for the coefficients $(C_1^{n,bw}, C_2^{n,bw})$, the variable α and the regression errors.

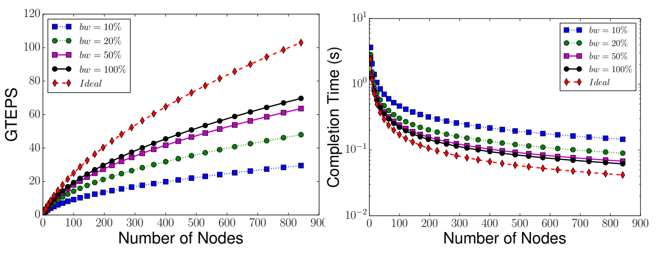
The Generalized Refined Regression Model: The *generalized refined regression model* is defined by Eq. 7, where the *CT* is defined as a function of the cluster size, the interconnect bandwidth, and the graph scale.

This model is designed using Graph500 benchmark experiments with different graph scales, different number of nodes (in our 32-node cluster), and different available bandwidth percentages (10% to 50% with 2% increment interval) enforced by the bandwidth throttling tool *InterSense*. In these experiments, we use 2^{s-25} for D, where s denotes the graph scale. We have two sets of experiments including different ranges of graph data sizes. The first set (shown in Fig. 6(b)) includes graph scales from 26 to 28 and use approximately $15,000$ data points. The second set (plots omitted) includes graph scales from 26 to 29 and use $20,000$ data points approximately.

We solve Eq. 7 with linear regression for finding coefficients $C_1^{n,bw,D}$, and $C_2^{n,bw,D}$. The value of α is optimized based on two types of errors (MSE and R^2). Table IV shows all the results for the coefficients $(C_1^{n,bw,D}, C_2^{n,bw,D})$, the variable α and the mentioned errors. The errors show that the proposed model produces high quality in both settings.

B. Performance and Scalability Projections

Projections with the Refined Base Regression Model: Ideally, the shape of *CT* for Graph500 benchmark (implemented using 2-D partitioning algorithm) is defined by the *base linear regression model* described by Eq. 2. However, our experiments with bandwidth throttling tool and the related analysis show, that the communication time increases due to higher communication volumes and increased interconnect bandwidth demands in a larger cluster. Therefore,



(a) TEPS (b) CT

Figure 7. Performance projections: (a) GTEPS (GigaTEPS), and (b) *CT* (logscale) using the *base linear regression model* (Eq. 2) versus the *refined linear regression model* (Eq. 6).

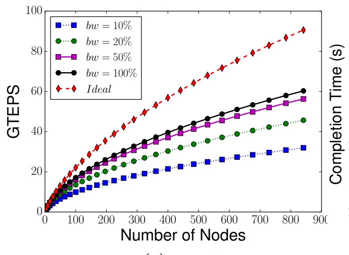
CT will be higher (worse) than anticipated by the *base model* and Eq. 2, and it will be more accurately described by the *refined regression model* and Eq. 6. Note, that the other performance metric of interest is *Traversed Edges Per Second (TEPS)*. The *TEPS* metric is inverse to the *CT*, i.e., higher values are better. This metric is to measure performance of different implementations for traversal algorithms.

Now, we use the model coefficients derived by regression and experiments (Table II and Table III respectively), and compare the Graph500 results projected by the *base linear regression model* and the *refined regression model*. Note, that the models' coefficients are different for different graph scales. Using these models, we can project CTs and TEPS of Graph500 for different graph scales over a large number of nodes in the cluster.

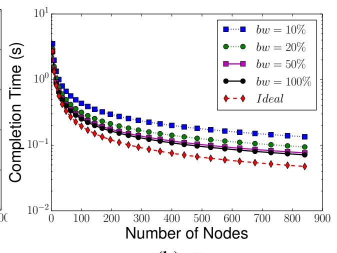
Fig. 7 shows the results for scale 28 (scale $26, 27$ and 29 are similar). In Fig. 7, the line "Ideal" stands for model defined by Eq. 2. The other lines are derived using Eq. 6 for different interconnect bandwidth. The Graph500 performance metrics GTEPS (i.e., GigaTEPS) and CT in the "Ideal" model are overly optimistic and significantly overestimate the achievable application performance. The *refined regression model*, which incorporates the interconnect bandwidth into account, provides a more accurate and realistic projection of Graph500 performance in a larger size cluster. We use logscale at Y-axes for showing CT (Fig. 7 (b)) in order to better reflect performance projection differences and to stress the impact of interconnect bandwidth on the benchmark performance in a larger size cluster.

One additional interesting observation in Figure 7 is the following. If we enlarge the initial part of this figure, then we can see that CTs defined by Eq. 2 ("Ideal") and Eq. 6 with bandwidth of 50% and 100% are practically the same in the "small" cluster range (i.e., less than 32 nodes). For this number of nodes the communication volume is small and is not impacted by the interconnect bandwidth. The deviation between the lines gets more significant with the larger number of nodes, when the increased communication volume gets impacted by the available interconnect bandwidth.

Projection with the Refined Generalized Regression Model: Following similar arguments and logic as above (in the case of the *refined base model*), we compare the performance projections defined by Eq. 3 and Eq. 7 as shown



(a) TEPS



(b) CT

Figure 8. Projections: (a) GTEPS and (b) CT (logscale) using the *generalized base model* defined by Eq. 3 vs *refined generalized model* defined by Eq. 7 for graph scale 28.

in Figure 8. We produce these projections for scale 28.

In Figure 8, the line "Ideal" stands for the *generalized base model* defined by Eq. 3. The other lines are derived using the *refined generalized model* defined by Eq. 7 for different interconnect bandwidth. As expected, the Graph500 performance metrics (TEPS and CT) produced by the "Ideal" model (Eq. 3) are overly optimistic and the performance metrics computed with the refined model (which takes the interconnect bandwidth into account) provides a more accurate and realistic projection.

It is worth noting that the performance projections defined by Eq. 6 and Eq. 7 for the same size (scale 28) are very close to each other: with differences less than 15% as one can see by comparing Figures 7 (a)-(b) and Figures 8 (a)-(b), which justifies our expectations of the performance outcomes.

The main benefit of the designed refined generalized model is that it can project the application performance for different graph sizes (even those for which we did not have training data). However, some "back of the envelope" calculations should be performed, e.g., making sure that the considered larger graph fits in memory of the cluster, etc.

With the insights from these experiments, we project (Table V) the application performance for larger cluster sizes, and in particular, the size (number of nodes), where the communication cost becomes the dominant component and leads to a diminishing return in the scalability equation.

Eq. 2 (Ideal)	**Eq. 6 (Refined)** bw (100%)	**Eq. 3 (Gen. Ideal)**	**Eq. 7 (Gen. Ref.)** bw (100%)
81	25	81	16

Table V
SUMMARIZING THE MODELS SHOWING WHEN (THE NUMBER OF NODES) THE COMMUNICATION TIME COMPONENT BECOMES DOMINANT FOR SCALE 28.

VI. Related Work

A large group of tools is based on the program "pos-mortem" analysis, i.e., these tools record application runtime performance behavior (variety of metrics, system calls, etc.) and analyze the collected measurements after the program terminates, such as Scalasca [7], TAU [21], HPC-Toolkit [8], Extrae [10], Paraver [11], just to name a few. Some tools like Scalasca and Tau can interoperate and redirect calls to each other. They perform in-depth studies of concurrent behavior using event tracing, and then provide scalability bottleneck analysis by identifying potentially inefficient pieces of the program. However, most of these frameworks and tools do not support predictive modeling. The collected traces could be fed to different simulators, such as Dimemas [12] or xSim [14], where additional "what-if" and scalability analysis can be performed. There is a large body of work on analytic models of targeted applications. However, most of them involve detailed application profiling and understanding semantics of the code. For example, in [9], the authors generate an empirical performance model automatically. The goal of this model to identify "performance bugs" and assist programmers in optimizing their programs.

The work most closely related to ours utilizes various regression-based techniques to predict application performance [22], [23], [24]. Neural networks [23] and piece-wise polynomial regression [24] are applied to predict the program execution time using a "black-box" approach. Barnes et al. [22] identify techniques to separate computation and communication. The authors (similar to our work) observe that program computation and communication times scale differently with processor count changes. The approach aims to estimate the communication delays more accurately, while in our work we analyze communication bandwidth impact on the communication time. To automate the analytical model derivation, we introduce a novel approach based on a bandwidth-throttling tool [16], [25] which enables us to analyze and incorporate into the model the necessary additional information about application performance sensitivity to bandwidth limitations under different processor counts.

VII. Conclusion

In this work, we discuss a new approach for assessing the scalability and performance of large graph analytics by using Graph500 benchmark as a motivating example. We show a set of critical factors that needs to be taken into account for scalability analysis of a distributed memory program. Since the scalability of many distributed programs is limited by their communication volume and the available interconnect bandwidth, we show how one can analyze the performance impact and estimate the required interconnect bandwidth in a larger cluster from the experiments performed in a small cluster with an "interconnect bandwidth throttling" tool. We create the ensemble of predictive models for scalability and performance analysis of a given application, and can project the program performance in a large size cluster with hundreds of nodes.

VIII. Acknowledgments

Research was partially supported by the IIS-1219254 grant from the NSF.

References

[1] M. Wasi-ur Rahman *et al.*, "High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand," in *IPDPS Workshops and PhD Forum*, 2013.

[2] M. Wasi-ur-Rahman *et al.*, "MapReduce over Lustre: Can RDMA-Based Approach Benefit?" in *EuroPar*, 2014.

[3] X. Lu *et al.*, "Accelerating Spark with RDMA for Big Data Processing: Early Experiences," in *Hot Interconnects*, 2014.

[4] J. Jose *et al.*, "Memcached Design on High Performance RDMA Capable Interconnects," in *ICPP*, 2011.

[5] M. Casas, R. M. Badia, and J. Labarta, "Automatic Analysis of Speedup of MPI Applications," in *ICS*, 2008.

[6] K. Barker, K. Davis, A. Hoisie, D. Kerbyson, M. Lang, S. Pakin, and J. Sancho, "Using Performance Modeling to Design Large-Scale Systems," *Computer*, vol. 42, Nov., 2009.

[7] M. Geimer *et al.*, "The scalasca performance toolset architecture," *J. on Concurr. Comput.: Pract. Exper.*, vol. 22, 2010.

[8] L. Adhianto *et al.*, "HPCTOOLKIT: tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, 2010.

[9] A. Calotoiu, T. Hoefler, M. Poke, and F. Wolf, "Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes," in *SC*, 2013.

[10] "Extrae Instrumentation Package," http://www.bsc.es/computer-sciences/extrae.

[11] "Paraver: Performance Analysis Tools: Details and Intelligence," http://www.bsc.es/computer-sciences/paraver.

[12] "Dimemas: Predict Parallel Performance Using a Single CPU Machine," http://www.bsc.es/computer-sciences/dimemas.

[13] C. Rosas, J. Gimenez, and J. Labarta, "Scalability Prediction for Fundamental Performance Factors," *Journal on Supercomputing Frontiers and Innovations*, vol. 1, no. 2, 2014.

[14] ——, "Scaling to a million cores and beyond: Using lightweight simulation to understand the challenges ahead on the road to exascale." *Journal on Future Generation Computer Systems*, vol. 30, 2014.

[15] "Graph500," http://www.graph500.org/.

[16] Q. Wang *et al.*, "InterSense: Interconnect Performance Emulator for Future Scale-out Distributed Memory Applications," in *MASCOTS*, 2015.

[17] S. Medya *et al.*, "Towards Performance and Scalability Analysis of Distributed Memory Programs on Large-Scale Clusters," in *ICPE*, 2016.

[18] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry, "Challenges in Parallel Graph Processing," *Parallel Processing Letters*, vol. 17, no. 1, 2007.

[19] A. Buluç and K. Madduri, "Parallel breadth-first search on distributed memory systems," in *SC*, 2011.

[20] A. Buluc and J. R. Gilbert, "Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments," *J. on Scientific Computing*, vol. 34, no. 4, 2012.

[21] S. S. Shende and A. D. Malony, "The Tau Parallel Performance System," *IJHPCA*, vol. 20, no. 2, 2006.

[22] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, and M. Schulz, "A Regression-based Approach to Scalability Prediction," in *ICS*, 2008.

[23] E. Ipek, B. R. de Supinski, M. Schulz, and S. A. McKee, "An approach to performance prediction for parallel applications," in *Euro-Par*, 2005.

[24] B. C. Lee, D. M. Brooks, B. R. de Supinski, M. Schulz, K. Singh, and S. A. McKee, "Methods of inference and learning for performance modeling of parallel applications," in *PPoPP*, 2007.

[25] Q. Wang, L. Cherkasova, J. Li, and H. Volos, "Interconnect Emulator for Aiding Performance Analysis of Distributed Memory Applications," in *ICPE*, 2016.