Estimating the Impact of Communication Schemes for Distributed Graph Processing

Tian Ye
Department of Computer Science
University of Southern California
Los Angeles, USA
tye69227@usc.edu

Sanmukh R. Kuppannagari
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
kuppanna@usc.edu

Cesar A. F. De Rose
School of Technology
PUCRS
Porto Alegre, Brazil
cesar.derose@pucrs.br

Sasindu Wijeratne
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
kangaram@usc.edu

Rajgopal Kannan
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
rajgopak@usc.edu

Viktor K. Prasanna
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
prasanna@usc.edu

***Abstract*—Extreme scale graph analytics is imperative for several real-world Big Data applications with the underlying graph structure containing millions or billions of vertices and edges. Since such huge graphs cannot fit into the memory of a single computer, distributed processing of the graph is required. Several frameworks have been developed for performing graph processing on distributed systems. The frameworks focus primarily on choosing the right computation model and the partitioning scheme under the assumption that such design choices will automatically reduce the communication overheads. For any computational model and partitioning scheme, communication schemes — the data to be communicated and the virtual interconnection network among the nodes — have significant impact on the performance. To analyze this impact, in this work, we identify widely used communication schemes and estimate their performance. Analyzing the trade-offs between the number of compute nodes and communication costs of various schemes on a distributed platform by brute force experimentation can be prohibitively expensive. Thus, our performance estimation models provide an economic way to perform the analyses given the partitions and the communication scheme as input. We validate our model on a local HPC cluster as well as the cloud hosted NSF Chameleon cluster. Using our estimates as well as the actual measurements, we compare the communication schemes and provide conditions under which one scheme should be preferred over the others.**

***Index Terms*—Distributed Graph Processing, Performance Estimation, Communication Schemes, Cluster Computing.**

I. Introduction

Graphs provide a powerful abstraction for representing real world networked data [1]. Examples such as social networks [2], biological networks [3] and transportation networks [4] illustrate their ubiquity. Extreme scale graph analytics is imperative for several real-world Big Data applications with the underlying graph structure containing millions or billions of vertices and edges [5].

This work is sponsored by the U.S. NSF under grant numbers OAC-1911229, PPoSS-2119816 , SaTC-2104264.

The massive sizes of graphs have made distributed processing essential. Local dedicated clusters are available to the researchers and practitioners at several institutions to perform distributed graph processing. In addition, public clouds, with their flexibility, elasticity and pay-as-you-go-models are also available for utilization. Moreover, the availability of several distributed graph processing frameworks, which allow easier distributed application development, has further increased the adoption of distributed graph processing [6].

In a typical distributed graph processing framework [7]–[9], the graph is partitioned and assigned to various compute nodes. The connectivity across the partitions induces communication among the compute nodes. Communication latency is significant in distributed clusters/clouds. To reduce communication, most distributed frameworks focus on choosing the right computation models or partitioning scheme.

While such design choices do reduce communication, they exert a second order impact on the same. The communication scheme, which we define as the choice of data (vertex attributes, intermediate updates, etc.) to be communicated and the virtual interconnection network among the communication nodes, has a first-order impact on the communication performance. Formal analysis on the impact of communication schemes is lacking. A brute force approach wherein an application developer evaluates all possible configurations of number of nodes and communication schemes can be prohibitively expensive both computationally and economically. With increasing adoption of cloud platforms for graph processing, such an analysis is imperative for addressing the challenges due to high cloud latencies.

In this work, we explicitly decouple the communication and computation in order to estimate the impact of communication in distributed environments. We develop performance estimation models for communication schemes used by existing most popular distributed graph processing frameworks. The performance models take as input the partitions of the graph

and the communication scheme and output the estimated communication time. Thus, our estimation models enable a preliminary analyses of the trade-offs between the partitioning schemes and the communication in an early development stage. We validate our models using a local university cluster and the cloud hosted NSF Chameleon cluster [10].

II. Contributions

The significant contributions of this work are as follows:

- We identify the communication schemes used by the most popular distributed graph processing frameworks;
- Given partitions of a graph, we develop models that estimate the performance of these communication schemes;
- We validate our performance estimation models with experiments on a local and a cloud hosted cluster;
- We use our models to provide several crucial insights:
  - We provide a concrete explanation behind the superior performance of frameworks that perform edge proportional communication (Section V-C-Insight 3).
  - We show that even though edge proportional communication frameworks have lower communication in absolute terms, the scalability of frameworks performing vertex proportional communication is much superior (Section V-C-Insight 2).
  - We show a hypothetical scenario where such vertex proportional communication frameworks can outperform the edge proportional communication frameworks in terms of communication time (Section V-C-Insight 4).
  - We conclude that partitioning scheme has no impact on vertex proportional communication.
  - We also formulate the requirements of the optimal partitioning scheme for edge proportional communication frameworks and show that graph partitioning heuristics should reduce the number of vertices which have at least one incoming edge from other partitions as opposed to minimizing the number of edges in the cut-set that is typically done (Section V-C-Insight 5).

III. Graph Processing Communication Schemes

In this work, we focus on distributed in-memory graph processing frameworks which use message passing for communication [6]. Examples of such frameworks include Pregel [11], Giraph [12], iGiraph [8], etc. In line with the aforementioned frameworks, factors such as the time for fetching data from disks into main memories are treated as preprocessing steps and not considered in the analysis.

A. Preliminaries

Platform Definition: We assume a distributed graph analytics platform consisting of a cluster of N compute nodes for processing a graph $G = (V, E)$. We assume that the data are represented at the granularity of words, which is platform dependent. To model the communication time of a message, we let t_s denote the average communication latency between two nodes, i.e., the time taken for the destination node to receive the first word from the source node over the network. Let t_w denote the average time it takes to transfer a word. Finally, let d denote the average degree of G.

Distributed Graph Processing Abstraction: We consider the Graph Processing Over Partitioning (GPOP) [13] abstraction for distributed graph processing. In this abstraction the graph G is partitioned into several partitions. The partitions are assigned to the N-node cluster. In each iteration, each node performs computations over the partitions assigned to it. The results of the computation are communicated using a partition-centric approach, i.e., the communication occurs at the granularity of partitions as opposed to vertices or edges. This process continues for a fixed number of iterations or until the algorithm converges. Furthermore, as the communication between the partitions assigned to the same node occurs via local memory and does not induce a communication on the network, the discussion in this paper, henceforth, centers around nodes of the cluster. Widely used frameworks such as Pregel, Giraph, iGiraph, etc. can be easily modeled using this abstraction.

B. Motivation for Categorization Based on Communication Schemes

Analysis of Existing Frameworks: In [14], authors develop a centralized framework with a master node dispatching the tasks and collecting the results (updated vertex attributes). In [15], the authors developed a reduce-scatter algorithm based on a ring logic. Each node computes a portion of the entire results (vertex attributes), and repeatedly delivers a portion to its successive neighbor on the logical ring. Frameworks such as Pregel [11], Giraph [12], iGiraph [8], Gluon [16] and those developed in [17], [18], etc. compute a unique message for each destination vertex using the source vertex attribute and the edge attribute. Thus, the number of messages generated is proportional to the number of (incoming) edges in the graph.

Communication Scheme Definition: By analyzing the most popular graph processing frameworks, we identify the dimensions along which they differ with respect to their communication characteristics.

I. Type of Data Being Communicated: A framework performs either of the following: (i) Broadcast the vertex attributes in the communication phase. In the computation phase, each node selects the appropriate broadcast vertex attributes using the list of incoming edges of the vertices in its partitions. (ii) In computation phase, each node generates a unique message for every outgoing edges of the vertices in its partitions. In communication phase, all-to-all personalized communication of the message is performed. We denote the former scheme as **Vertex Proportional Communication (VPC)** while the latter as **Edge Proportional Communication (EPC)**.

II. Underlying Virtual Communication Network: A framework either uses a centralized scheme where a master node collects all the data and sends it to the worker nodes

or it uses a distributed network, where the nodes communicate directly with each other. We denote the centralized network as **Master-Worker (MW)**. For distributed network based frameworks, frameworks that use EPC perform all-to-all personalized communication with direct communication between nodes. We denote this network as **Peer-to-Peer (P2P)**. However, for frameworks that use VPC and require a broadcast operation, [15] shows that an embedded ring logic is an efficient network for the same. We denote this network as **Ring**.

C. Communication Schemes

Based on the analysis in the previous section, we now formally define the communication schemes.

1) Vertex Proportional Communication (VPC): We say that a distributed application follows *Vertex Proportional Communication* scheme if the messages being communicated are the vertex attributes. Note that we do not make any assumptions regarding the computational model. Algorithm 1 shows an example of VPC-based PageRank algorithm. Each vertex computes its PageRank value and then broadcasts the value to other vertices so that they can use it in the next iteration. Thus, the amount of data to communicate depends on the number of vertices in the graph.

Algorithm 1: VPC BASED PAGERANK

Input: A graph $G^* = (V^*, E^*)$
Output: PageRank results for all vertices $PR[:]$

```
1 PR[:] ← 1/|V|
2 while Convergence > Expected Convergence do
3     for each vertex u ∈ V* do
4         sum ← 0
5         for each v ∈ Adj(u) do
6             sum ← sum + PR[v]/OutDeg(v)
7         PR[u] ← (1 − df)/|V| + df × sum
          // df = damping factor
8     All_to_All_Broadcast(PR)
```

Now, based on the virtual interconnection network, we further categorize VPC into two categories:

a) Vertex Proportional Communication on Master-Worker Network (VPC-MW): In this category, the distributed application follows a master-worker model. As shown in Figure 1(a), every worker node is connected to a master node. There is no direct data communication between worker processes. **All-to-all Broadcast** is implemented in two steps. In the first step, each worker node in the cluster sends its data to the master, and the master node collects data from all worker processes. In the second step, the master node broadcasts the data received in the first step to every worker.

b) Vertex Proportional Communication on Ring Network (VPC-Ring): In this category, the distributed application uses a virtual ring network to implement broadcast operation. **All-to-all Broadcast:** Let us denote the N nodes as M_0, M_1, ..., M_{N-1}. As shown in Figure 1(b), the ring model only contains a bidirectional link between each M_k and $M_{(k+1)\%N}$, where $k = 1, 2, \ldots, N$. In Iteration 1, each node M_k sends its initial data to its right neighbor $M_{(k+1)\%N}$, and receives data from its left neighbor $M_{(k+N-1)\%N}$. Both sending and receiving operations are non-blocking primitives, so they can be performed at the same time. After receiving data from its left neighbor, each node stores data in its local memory. For the remaining iteration j ($j = 1, 2, ..., N-1$), each node sends data that was initially contained in $M_{(k-j+1)\%N}$ to its right neighbor $M_{(k+1)\%N}$, and receives data that was initially contained in $M_{(k-j)\%N}$ from its left neighbor $M_{(k+N-1)\%N}$. After $(N-1)$ iterations, every node can get a copy of data originated from all other nodes. [15] also proves that such implementation on ring network is bandwidth optimal.

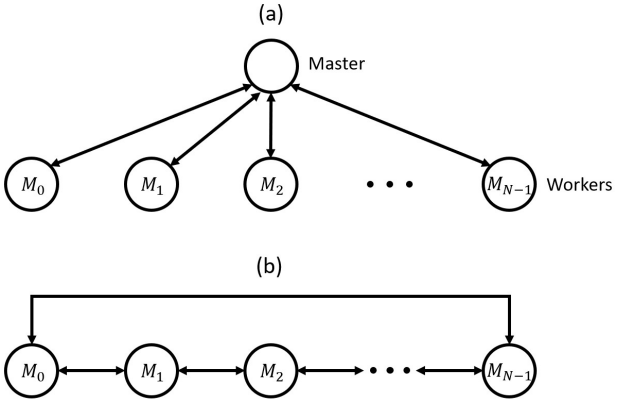

Fig. 1. (a) Master-Worker Network (b) Ring Network

2) Edge Proportional Communication (EPC): We say that a distributed application follows *Edge Proportional Communication* scheme if the messages being communicated are generated due to an operation on the edge weight and the attribute of the source vertex. Here again, we do not make any assumptions regarding the computational model. The virtual interconnection network that we assume in this case is a fully connected Peer-to-Peer network, i.e., each node can communicate with every other node directly. We do not append P2P to EPC as this is the only virtual network that is used.

Algorithm 2 shows an example of EPC-based PageRank. Each vertex computes its contribution to each of its neighbors. The computed values are communicated among nodes using all-to-all personalized communication. We also consider a widely used optimization [7], [16] that combines messages destined to the same vertex from a partition.

Remark: [6] provides a classification of distributed processing frameworks based on graph topology. Under that terminology, our categorization is applicable to the frameworks falling under the Vertex-Centric (Edge-Cut) and Component-Centric frameworks. Also note that there is no one-to-one correspondence between their categorization and ours, i.e., a Vertex-Centric (Edge-Cut) (or a Component-Centric framework) can follow either VPC or EPC scheme.

Algorithm 2: EPC BASED PAGERANK

Input: A graph $G^* = (V^*, E^*)$
Output: PageRank results for all vertices $PR[:]$

```
PR[:] ← 1/|V|
while Convergence > Expected Convergence do
    sum_iter[:] ← 0
    for each vertex u ∈ V* do
        contribute ← PR[u]/OutDeg(u)
        for each destination v ∈ Adj(u) do
            sum_iter[v] ← sum_iter[v] + contribute
    All_to_All_Personalized_Communication(sum_iter)
    for each vertex u ∈ V* do
        PR[u] ← (1 − df)/|V| + df × sum
```

IV. COMMUNICATION SCHEME PERFORMANCE MODELING

In this section, we build models to estimate the performance of the aforementioned communication schemes. The models take the partitions, their logical mapping to nodes and the communication scheme as input.

A. Theoretical Analysis of Communication Time

Let V denote the number of vertices and N denote the number of nodes. If multiple partitions are mapped to a node, we consider the union of them as a single partition. Thus, the number of partitions is equal to the number of nodes in our model. For VPC-MW scheme, there exists an additional node acting as master besides worker nodes. The average communication latency between two nodes is denoted by t_s while t_w denotes the per word transfer time in the communication network.

(1) For VPC-MW scheme, the communication time is

$$\begin{aligned} T_{VPC-mw} &= (t_s + V \times t_w) \times N + (t_s + \frac{V}{N} \times t_w) \times N \\ &= 2Nt_s + (N+1)(V \times t_w) \end{aligned} \tag{1}$$

Assuming that each worker receives data from the master sequentially, the first term is the time for sending a vector with a length of V, containing attributes for all vertices, from the master to workers. The second term is the time for sending a vector with a length of V/N from every worker to the master.

(2) For VPC-Ring scheme, the execution time for performing a broadcast on the ring can be estimated as

$$T_{VPC-ring} = (N-1)(t_s + \frac{V}{N} \times t_w) \tag{2}$$

In each step, every node performs two actions: receiving a vector with a length of V/N from its left neighbor and sending a vector with the same length to its right neighbor. All the nodes take the two actions at the same time. Such steps are repeated $(N-1)$ times in a broadcast process.

(3) For EPC scheme, the communication time is given as

$$T_{EPC} = \sum_{i=1}^{N} (t_s + t_w \sum_{j \neq i} \eta_{ij} \alpha_{ij}) \tag{3}$$

Here, α_{ij} denotes the number of vertices in partition j that have at least one incoming edge from partition i. Formally, $\alpha_{ij} = \|\mathbf{A}_{ji}\mathbf{1}\|_0$, where $\mathbf{A}_{ji}$ denotes the sub-matrix in the adjacency matrix of the graph with rows corresponding to partition j and columns corresponding to partition i and $\mathbf{1}$ denotes all ones vector of suitable size. η_{ij} denotes the average size of the combined messages. For commutative operator, $\eta_{ij} = 1$. Note that in the worst case, $\alpha_{ij} = \frac{V}{N}$.

Given a graph partitioning, we can calculate the parameters V, N and α_{ij}. However, to obtain the values of t_s and t_w, measurements on the target platform are required which is discussed in the next subsection.

B. Estimation of Latency and Throughput

We perform the following experiment on two platforms (more details in Section V-A) - *HPC platform* and *Chameleon Cloud* to estimate the value of latency t_s and throughput t_w.

On each platform, we use MPI primitives to perform round-trip communication between two different nodes and measure the communication time. Let L denote the length of communicated data. The round-trip time can be estimated as $T_{RTT} = 2(t_s + L \times t_w)$. We can estimate t_s and t_w by varying the value of L and fitting the measured T_{RTT} using a linear function with the least mean square error. For the Chameleon cloud, we estimate the latency and throughput as 0.35 ms and 1.05 ms/MiB, respectively. For HPC, we estimate them to be 0.2 ms and 0.20 ms/MiB, respectively.

C. Impact of the Performance Estimation Model

For a thorough design space exploration, an application developer will need to vary the number of nodes, partitioning techniques, communication patterns, etc. leading to a combinatorial explosion of the design space. Moreover, given the increasing large scale of graphs – billions of vertices and edges, a sub-optimal choices will have tremendous time and monetary costs as typical graph processing algorithms run for hundreds or thousands of iterations. Our performance estimation model can enable quick trade-off analysis. Additionally, our model can be used to understand the impact of various parameters, e.g., communication schemes, number of nodes, on the performance of graph processing applications. Conclusions drawn from the estimations can help accelerate design space exploration. For example, as per Section V-C-Insight 5, partitioning strategy has no impact on VPC communication time with VPC models and need not be analyzed.

V. EXPERIMENTAL EVALUATION

A. Experiment Setup

We conduct our experiments on two computing platforms. First, we experiment on the High-Performance Cluster (HPC) provided by the university. We use machines with Dual Intel

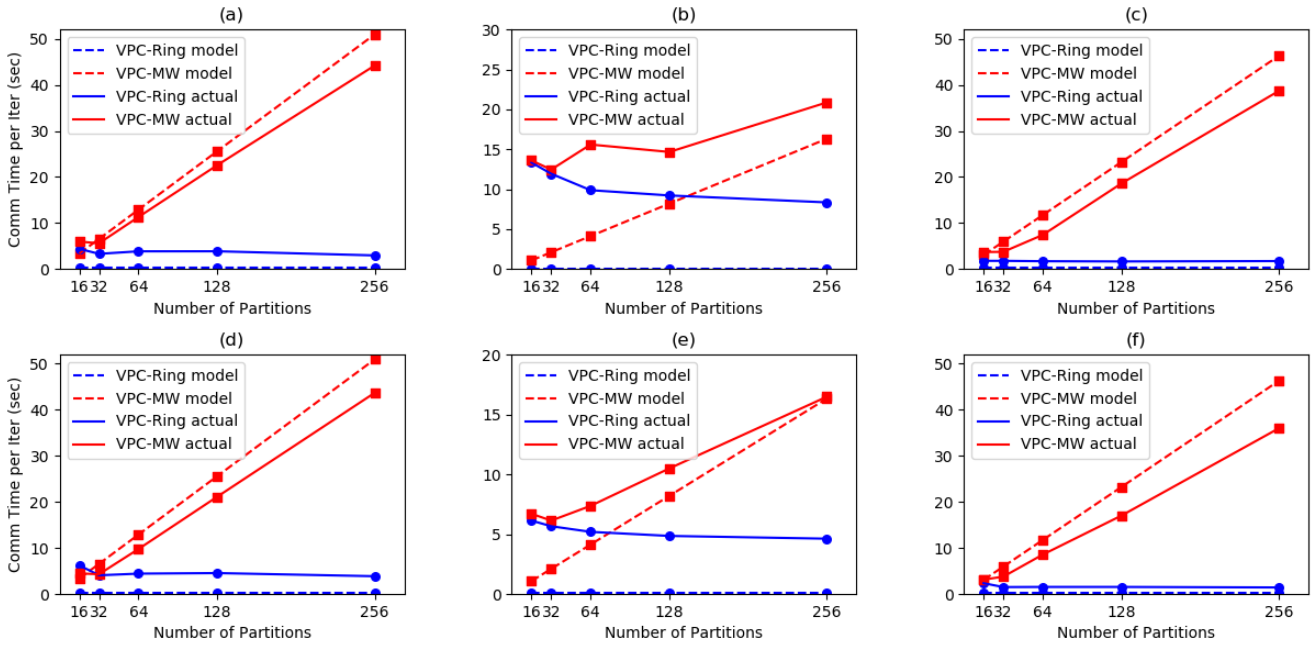

Fig. 2. Theoretical Estimation and Experimental Results for VPC on HPC. From top to bottom, the three rows are for PageRank and WCC, respectively. From left to right, the three columns are using as datasets *uk-union-2006-06-2007-05*, *twitter-2010* and *webbase-2001*, respectively.

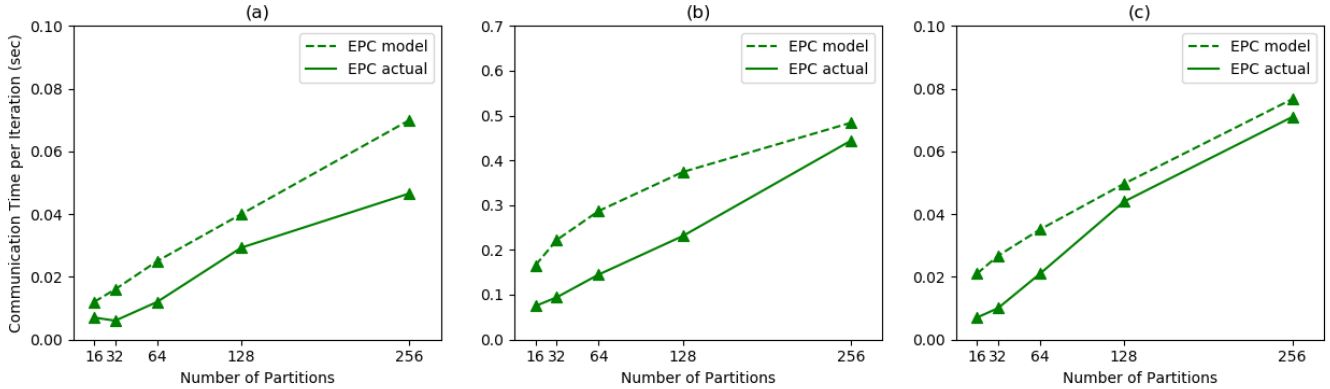

Fig. 3. Theoretical Estimation and Experimental Results for PageRank using EPC on HPC. From left to right, the three columns are using as datasets *uk-union-2006-06-2007-05*, *twitter-2010* and *webbase-2001*, respectively.

TABLE I
PROPERTIES OF DATASETS

Graph	Edges	Vertices	Avg. degree
uk-union-2006-06-2007-05 [19]	5 507 679 822	133 633 040	41.215
twitter-2010 [20]	1 468 365 182	41 652 230	35.253
webbase-2001	1 019 903 190	118 142 155	8.633

Xeon 10-core 2.4 GHz processors and up to 64GB memory on HPC cluster. Then we repeat our experiments on the MPICH3 bare-metal cluster provided by Chameleon cloud, an NSF-funded platform for large-scale cloud research. Each node has 24 Intel Xeon E5-2670 v3 2.3GHz CPUs and 128GB memory. Different machines are connected with InfiniBand.

We choose PageRank and Weakly Connected Components (WCC) algorithms as our benchmarks. We perform experiments using *uk-union-2006-06-2007-05*, *twitter-2010* and *webbase-2001* each containing over 1 billion edges with average degree varying from 8.63 to 41.21 (Table I). The graphs, processed by Webgraph [21] and LLP [22] tools, have very high locality due to optimized vertex ordering.

B. Validation of the Performance Models

We implement each algorithm using the three communication schemes. For each algorithm and communication scheme, we conduct experiments with $\{16, 32, 64, 128, 256\}$ partitions mapped to $\{1, 2, 4, 8, 16\}$ nodes respectively. That is each node, with 16 processor cores, is assigned 16 partitions so that each partition can be processed in parallel. For Chameleon, we failed to allocate enough nodes for the case of 256 partitions. As a pre-processing step, we first partition the initial graphs into subgraphs and save them into the file system. For each experiment, we measure communication time per iteration by averaging over all the iterations of the algorithm.

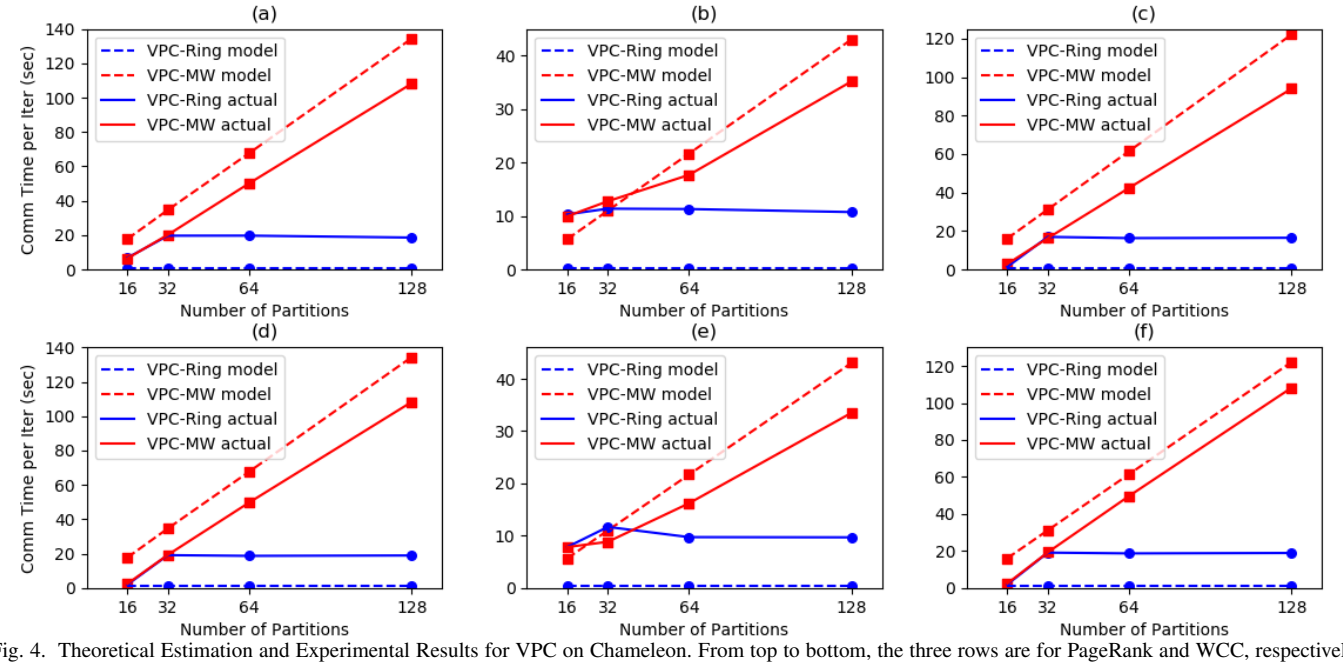

Fig. 4. Theoretical Estimation and Experimental Results for VPC on Chameleon. From top to bottom, the three rows are for PageRank and WCC, respectively. From left to right, the three columns are using as datasets *uk-union-2006-06-2007-05*, *twitter-2010* and *webbase-2001*, respectively.

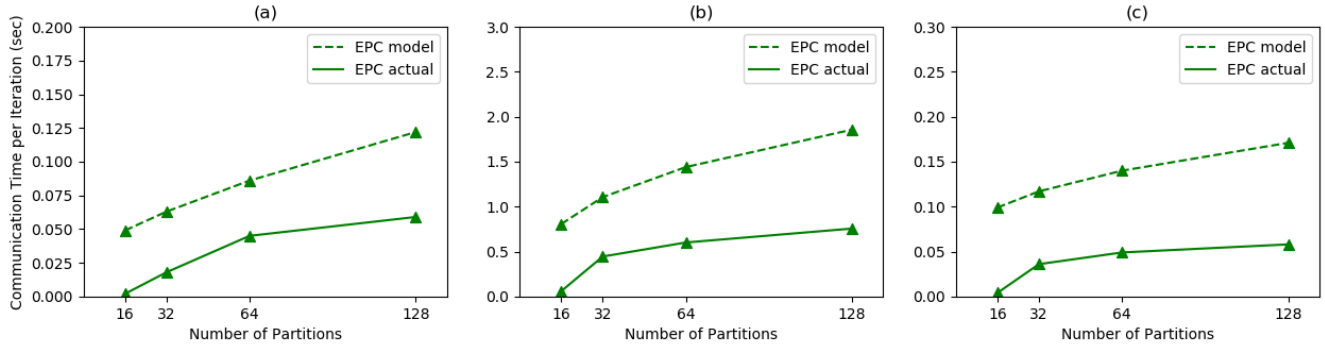

Fig. 5. Theoretical Estimation and Experimental Results for PageRank using EPC on Chameleon. From left to right, the three columns are using as datasets *uk-union-2006-06-2007-05*, *twitter-2010* and *webbase-2001*, respectively.

Figures 2-5 show the results. Dashed lines represent the estimations derived from our performance model while solid lines represent experimental results. As the platforms are running other applications at the same time, communication congestion occurs depending upon the network utilization in the data center. The impact of such congestion is hard to predict since it does not depend upon the applications we analyze. However, it is still evident that the solid lines are close to corresponding dashed lines or have similar trends in the graphs, thereby validating our models.

A few observations from the graphs are as follows: VPC-MW has a worse performance than VPC-Ring and EPC. VPC-MW's execution time increases almost linearly and hence it has a poor scalability compared with VPC-Ring. EPC performs better than VPC-Ring experimentally.

C. Discussion

Insight 1: VPC-Ring and EPC consistently outperform VPC-MW.

We derive this conclusion from both our model and experimental results. Since the latency t_s is relatively much lower than communication time, we can simplify our model by ignoring terms with t_s. The estimated communication time for the three schemes are as follows

$$T_{VPC-ring} = (N-1)(\frac{V}{N} \times t_w) \approx Vt_w \quad (4)$$

$$T_{VPC-mw} = (N+1)Vt_w \quad (5)$$

$$T_{EPC} = t_w \sum_{i=1}^{N} \sum_{j \neq i} \alpha_{ij} \leq NVt_w \quad (6)$$

From the simplified model we can see that VPC-MW requires $O(N)$ times higher communication that VPC-Ring. Moreover, even in the worst case the communication of EPC is slightly better than VPC-MW.

Insight 2: VPC-Ring has the best scalability.

Both the simplified model and experimental results show that the communication time for VPC-Ring remains constant

with the increasing number of partitions. Whereas, the communication time for VPC-MW and EPC increases almost linearly. For the latter two schemes, there is a trade-off between the communication time and the storage at each machine, which is determined by the number of partitions. Using VPC-MW or EPC as communication scheme, if we divide the original graph into more partitions, more time will be spent on communication, but each machine will require less storage.

Insight 3: In practice, EPC outperforms VPC-Ring.

For a partition i, $\sum_{j\neq i}\alpha_{ij} = \sum_{j\neq i}\|\mathbf{A}_{ji}\mathbf{1}\|_0 \leq \sum_{j\neq i}\|\mathbf{A}_{ji}\mathbf{1}\|_1 \leq d^i_{po}$, where d^i_{po} is the out degree of partition i, i.e., total number of edges with source vertex in partition i but destination vertex not in it. Now, $\sum_{i=1}^{N} d^i_{po} = Nd_{po} = V\frac{Nd_{po}}{V}$, where d_{po} is the average out degree of the partitions.

In real-world graphs which follow power law, a partitioning that leads to high in-partition connectivity and low connectivity across partitions will lead to a very small value of d_{po}. Thus, $\frac{Nd_{po}}{V} < 1$ implies better performance for EPC.

Insight 4: Hypothetical scenario where VPC-Ring will outperform EPC.

The above analysis indicates that VPC-Ring will outperform EPC when $\frac{Nd_{po}}{V} > 1$. This happens if the partitioned graph has low locality and few vertices from the same node share common destinations, which increases the average out degree among the partitions.

Insight 5: Impact of partitioning on the performance of the communication schemes.

As per our models, both $T_{VPC-ring}$ and T_{VPC-mw} are only dependent on the number of vertices in the graph and the number of partitions/nodes. Thus, partition scheme has no impact on the communication performance of VPC frameworks. An application choosing this scheme can focus on partitioning which is optimal for computation only.

For EPC, assuming $\eta_{ij} = 1$, an optimal partitioning is given by $\min\sum_{i=1}^{N}\sum_{j\neq i}\alpha_{ij}$. As α_{ij} denotes the number of vertices in partition j that have at least one incoming edge from partition i, an optimal partitioning will minimize the number of such vertices across all pairs of partitions. Clearly, this is an NP-hard problem due to the l_0 norm. Heuristics can be developed to explicitly optimize for this objective. Note that a partitioning that reduces the number of edges in the cut-set is given by $\sum_{j\neq i}\|\mathbf{A}_{ji}\mathbf{1}\|_1$, i.e., l_1 norm instead of l_0. Thus, instead of focusing on minimizing the l_1 norm, heuristics should explicitly focus on minimizing the l_0 norm.

VI. Related Work

The main focus of this paper is to model communication performance of graph processing frameworks rather than developing a new framework. Thus, we limit the discussion in this section on works that either focus on communication optimizations or develop performance models for communication time estimation for graph processing frameworks.

Communication Focused Frameworks: McCune et al. [18] provided a detailed survey of message passing and shared memory based graph processing. Heidari et al. [6] characterized communication models into message passing, shared memory and push/pull style. Both works characterize the communication models but do not provide a quantitative analysis of the communication performance. Patarasuk et al. [15] presented a framework that uses virtual ring interconnect (Ring in our work) to optimize bandwidth utilization for clusters with tree topology. Alfatafta et al. [23] proposed algorithms and architectures for several MPI collective operations, including using ring scheme to implement AllReduce and AllGather operations. However, it relies on the knowledge of the underlying system architecture and network structure which might not always be available.

Communication Performance Modeling of Graph Processing Frameworks: Xu et. al. [24] developed a performance model for vertex centric graph algorithms that captures communication bandwidth and communication latency. However, their target platform is a shared memory single node architecture and the communication is defined as the data transfer between the LLC (Last Level Cache) and external memory. In our work, we focus on modeling inter-node communication within a cluster/cloud platform. Al-Tawil et. at. [25] used LogGP, which is a simple performance model that reflects the most important parameters required to estimate the communication performance of parallel computers, to evaluate the performance of message passing interface (MPI) on different cluster platforms. Their main focus was on modeling at MPI primitive level whereas we model at a higher abstraction level of communication schemes as defined in this paper. Abdolrashidi et. al. [26] developed a model for computation and communication for vertex-centric graph algorithms and used the model to explore the trade-off of different partitioning schemes. Thus, their model can estimate communication performance for only vertex-centric algorithms. In contrast, by focusing on how the data is communicated instead of how it is generated, our model is agnostic to the underlying computation model. Thus, our model is applicable to a wide range of graph processing frameworks. Bhimani et. al. [27], [28] developed queuing theory based performance prediction model for distributed MPI systems. Their communication model is similar to the Master Worker Model (Section III-B) which requires one-to-many and many-to-one node communication. In contrast, we also model P2P communication model as widely used graph processing frameworks employ this model. Moreover, their focus is to perform accurate performance prediction of a given application, thereby requiring complex queuing theory and Machine Learning based modeling of packet transfer costs. In contrast, the focus of this paper is to develop a model that can enable trade-off analyses of various partitioning and communication scheme choices.

VII. Future Work

We will focus on the following: **Improving Performance Estimations**: Our focus in this work was to enable early stage trade-off analysis of various design choices. While our models already capture these trade-offs and trends in general, there is still some room to improve the accuracy for each individual configuration. Similar to [28], we plan to develop

stochastic and Machine Learning (ML) based models for network latency and round trip time predictions; and **Modeling Non-Stationary Algorithms:** In this work, we focused on stationary algorithms, i.e., algorithms in which all vertices participate in computations in all the iterations [13]. In contrast, in non-stationary algorithms, the vertices which participate in computations varies between iterations, e.g., Shortest path problems [13]. We will focus on developing heuristics to estimate average per-iteration performance.

VIII. Conclusion

In this work, we developed and validated performance estimation models for communication schemes used in popular distributed graph processing frameworks. The models enable analyses of the trade-offs between the partitioning schemes and the communication schemes in early development stages.

Current distributed graph processing frameworks have done extensive research on improving the performance by optimizations such as partitioning and load balancing. However, with increasing adoption of cloud platforms, optimizations which drastically reduce the communication time will be required. We believe a formal analysis into the communication schemes as presented in this paper will accelerate such innovations.

References

[1] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[2] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison, "Recent developments in exponential random graph (p*) models for social networks," *Social networks*, vol. 29, no. 2, pp. 192–215, 2007.

[3] R. Sharan and T. Ideker, "Modeling cellular machinery through biological network comparison," *Nature biotechnology*, vol. 24, no. 4, p. 427, 2006.

[4] K. Goczyłła and J. Cielatkowski, "Optimal routing in a transportation network," *European Journal of Operational Research*, vol. 87, no. 2, pp. 214–222, 1995.

[5] "Friendster social network and ground-truth communities," https://snap.stanford.edu/data/com-Friendster.html.

[6] S. Heidari, Y. Simmhan, R. N. Calheiros, and R. Buyya, "Scalable graph processing frameworks: A taxonomy and open challenges," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–53, 2018.

[7] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 17–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=2387880.2387883

[8] S. Heidari, R. N. Calheiros, and R. Buyya, "igiraph: A cost-efficient framework for processing large-scale graphs on public clouds," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2016, pp. 301–310.

[9] R. Chen, M. Yang, X. Weng, B. Choi, B. He, and X. Li, "Improving large graph processing on partitioned graphs in the cloud," in *Proceedings of the Third ACM Symposium on Cloud Computing*, ser. SoCC '12. New York, NY, USA: ACM, 2012, pp. 3:1–3:13. [Online]. Available: http://doi.acm.org/10.1145/2391229.2391232

[10] J. Mambretti, J. Chen, and F. Yeh, "Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*. IEEE, 2015, pp. 73–79.

[11] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 135–146.

[12] M. Han and K. Daudjee, "Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 9, pp. 950–961, 2015.

[13] K. Lakhotia, R. Kannan, S. Pati, and V. Prasanna, "Gpop: a cache and memory-efficient framework for graph processing over partitions," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, 2019, pp. 393–394.

[14] K. G. Narra, Z. Lin, M. Kiamari, S. Avestimehr, and M. Annavaram, "Slack squeeze coded computing for adaptive straggler mitigation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: ACM, 2019, pp. 14:1–14:16. [Online]. Available: http://doi.acm.org/10.1145/3295500.3356170

[15] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *J. Parallel Distrib. Comput.*, vol. 69, no. 2, pp. 117–124, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2008.09.002

[16] R. Dathathri, G. Gill, L. Hoang, H.-V. Dang, A. Brooks, N. Dryden, M. Snir, and K. Pingali, "Gluon: A communication-optimizing substrate for distributed heterogeneous graph analytics," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 752–768. [Online]. Available: https://doi.org/10.1145/3192366.3192404

[17] Z. Li, T. N. Hung, S. Lu, and R. S. M. Goh, "Performance and monetary cost of large-scale distributed graph processing on amazon cloud," in *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*, May 2016, pp. 9–16.

[18] R. R. McCune, T. Weninger, and G. Madey, "Thinking like a vertex: A survey of vertex-centric frameworks for large-scale distributed graph processing," *ACM Comput. Surv.*, vol. 48, no. 2, Oct. 2015. [Online]. Available: https://doi.org/10.1145/2818185

[19] P. Boldi, M. Santini, and S. Vigna, "A large time-aware graph," *SIGIR Forum*, vol. 42, no. 2, pp. 33–38, 2008.

[20] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 591–600. [Online]. Available: https://doi.org/10.1145/1772690.1772751

[21] P. Boldi and S. Vigna, "The webgraph framework i: Compression techniques," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 595–602. [Online]. Available: https://doi.org/10.1145/988672.988752

[22] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 587–596. [Online]. Available: https://doi.org/10.1145/1963405.1963488

[23] M. Alfatafta, Z. AlSader, and S. Al-Kiswany, "Cool: A cloud-optimized structure for mpi collective operations," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, July 2018, pp. 746–753.

[24] X. Wang, Y. Zhu, and Y. Chen, "Quantitative analysis of graph algorithms: Models and optimization methods," in *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, 2016, pp. 191–196.

[25] K. Al-Tawil and C. A. Moritz, "Performance modeling and evaluation of mpi," *Journal of Parallel and Distributed Computing*, vol. 61, no. 2, pp. 202–223, 2001.

[26] A. Abdolrashidi, L. Ramaswamy, and D. S. Narron, "Performance modeling of computation and communication tradeoffs in vertex-centric graph processing clusters," in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2014, pp. 55–63.

[27] J. Bhimani, N. Mi, and M. Leeser, "Performance prediction techniques for scalable large data processing in distributed mpi systems," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2016, pp. 1–2.

[28] J. Bhimani, N. Mi, M. Leeser, and Z. Yang, "New performance modeling methods for parallel data processing applications," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 29, no. 3, pp. 1–24, 2019.