Towards Benchmarking Graph-Processing Platforms

Yong Guo
TU Delft
The Netherlands
Yong.Guo@tudelft.nl

Marcin Biczak
TU Delft
The Netherlands
M.Biczak@tudelft.nl

Ana Lucia Varbanescu
University of Amsterdam
The Netherlands
A.L.Varbanescu@uva.nl

Alexandru Iosup
TU Delft
The Netherlands
A.Iosup@tudelft.nl

Claudio Martella
VU University Amsterdam
the Netherlands
claudio.martella@vu.nl

Theodore L. Willke
Systems Architecture Lab
Intel Corporation, USA
theodore.l.willke@intel.com

ABSTRACT

Graph-processing platforms are increasingly used in a variety of domains. Although both industry and academia are developing and tuning graph-processing algorithms and platforms, the performance of graph-processing platforms has never been explored or compared in-depth. Thus, users face the daunting challenge of selecting an appropriate platform for their specific application. To alleviate this challenge, we propose an empirical method for benchmarking graph-processing platforms. We implement a benchmarking suite, which includes a comprehensive process and a selection of representative metrics, datasets, and algorithmic classes. In our process, we focus on evaluating basic performance, resource utilization, scalability, and overhead. Our selection includes 5 classes of algorithms and 7 graphs. We use our suite on 6 platforms and, besides valuable insights gained for each platform, we also present the first comprehensive comparison of graph-processing platforms.

1. INTRODUCTION

Large-scale graphs are increasingly used in a variety of revenue-generating applications, such as social applications, online retail, business intelligence and logistics, and bioinformatics [1, 3]. By analyzing the structure and characteristics of graphs, analysts are able to predict the behavior of the customer, and tune and develop new applications and services. However, the diversity of the available graphs, of the processing algorithms, and of the graph-processing platforms currently available to analysts makes the selection of a platform an important challenge; we address it here.

Although performance studies of individual platforms exist [4, 5], they have been so far restricted in scope and size. New performance evaluation and benchmarking suites are needed to respond to the three sources of diversity, that is, to provide comparative information about the performance and other non-functional characteristics of different platforms, through the use of empirical methods and processes.

We propose an empirical performance-evaluation method for (large-scale) graph-processing platforms. Our method relies on defining a comprehensive evaluation process, and on selecting representative datasets, algorithms, and metrics for evaluating important aspects of graph-processing platforms—execution time, resource use, vertical and horizontal scalability, and overhead. Using this method, we create the equivalent of a benchmarking suite by selecting and implementing five graph algorithms and seven large-scale datasets from different application domains.

We implement our benchmarking suite on six popular platforms currently used for graph processing—Hadoop, YARN, Stratosphere, Giraph, GraphLab, and Neo4j—and conduct a comprehensive performance study. Our work also aligns with the goals and ongoing activity of the SPEC Research Group and its Cloud Working Group, of which some of the authors are members.

2. METHOD AND BENCHMARKING SUITE FOR GRAPH-PROCESSING PLATFORMS

In this section we present an empirical method for evaluating the performance of graph-processing platforms. Our method includes four stages: identifying the performance aspects and metrics of interest; defining and selecting representative datasets and algorithms; implementing, configuring, and executing the tests; and analyzing the results. Our approach exceeds previous performance evaluation and benchmarking studies in both breadth and depth.

In this study, we focus on four performance aspects: *Raw processing power*: the ability of a platform to (quickly) process large-scale graphs. *Resource use*: the ability of a platform to efficiently utilize the resources it has. *Scalability*: the ability of a platform to maintain its performance behavior when resources are added to its infrastructure. *Overhead*: the part of wall-clock time the platform does not spend on true data processing.

The main goal of the graph selection is to select graphs with different characteristics but with comparable representation. Table 1 shows a summary of the characteristics of the selected graphs. The graphs have diverse sources, and a wide range of different size and graph metrics.

We focus on algorithm functionality and select one exemplar of each of the following five algorithmic classes: general statistics, graph traversal (used in Graph500), connected components, community detection, and graph evolution. Gen-

Table 1: Summary of datasets.

	Graphs	#V	#E	d	$\bar{D}$	Size	Directivity
G1	Amazon	262.1 K	1.2 M	1.8	4.7	18 MB	directed
G2	WikiTalk	2.4 M	5.0 M	0.1	2.1	87 MB	directed
G3	KGS	293.3 K	16.6 M	38.5	112.9	210 MB	undirected
G4	Citation	3.8 M	16.5 M	0.1	4.4	297 MB	directed
G5	DotaLeague	61.2 K	50.9 M	2,719.0	1,663.2	655 MB	undirected
G6	Synth	2.4 M	64.2 M	2.2	53.6	964 MB	undirected
G7	Friendster	65.6 M	1.8 B	0.1	55.1	31 GB	undirected

d is the link density of the graphs ($\times 10^{-5}$). $\bar{\mathbf{D}}$ is the average vertex degree of undirected graphs and the average vertex in-degree (or average vertex out-degree) of directed graphs.

Table 2: Selected platforms.

Platform	Version	Type	Release date
Hadoop	hadoop-0.20.203.0	Generic, Distributed	2011-05
YARN	hadoop-2.0.3-alpha	Generic, Distributed	2013-02
Stratosphere	Stratosphere-0.2	Generic, Distributed	2012-08
Giraph	Giraph 0.2 (revision 1336743)	Graph, Distributed	2012-05
GraphLab	GraphLab version 2.1.4434	Graph, Distributed	2012-10
Neo4j	Neo4j version 1.5	Graph, Non-distributed	2011-10

eral statistics (STATS) computes the number of vertices and edges and the average of the local clustering coefficient of all vertices. Breadth-first search (BFS) is a widely used algorithm in graph processing; it is often a building block for more complex algorithm. Connected components (CONN) is an algorithm for extracting groups of vertices that can reach each other via graph edges. Community detection (CD) is important for social network applications, to find groups whose constituent nodes form more relationships within the group than with nodes outside the group. Graph evolution (EVO) is an algorithm that can predict how a graph structure will evolve over time.

3. EXPERIMENTAL RESULTS

In this section we create a full benchmarking suite, by implementing the graph-processing algorithms of a selected set of test platforms. Table 2 summarizes our selected platforms. The complete results are available through our technical report [2]. We present here a selection of the experimental results. We use common best-practices for tuning each platform. We deploy the tested platforms on 20 up to 50 computing machines of our multi-cluster DAS4.

We present here the results of the execution time obtained for running one selected algorithm (BFS), in Figure 1. Hadoop always performs worse than the other platforms, mainly because Hadoop has a significant I/O between two continuous iterations. Stratosphere's ability to optimize the execution plan based on code annotations regarding data sizes and flows, its programming model, and the much more efficient use of the network channel make it perform much better than Hadoop and YARN. In contrast to the generic data-processing platforms, for Giraph and GraphLab the input graphs are read only once, and then stored and processed in-memory. Both Giraph and GraphLab realize a dynamic computation mechanism, by which only selected vertices will be processed in each iteration. A two-level main-memory cache allows Neo4j to achieve excellent hot-cache execution times, especially when the graph data accessed by the algorithms fits in the cache.

For testing horizontal scalability, we increase the number of machines from 20 to 50 by a step of 5, and keep using a single computing core per machine. We use the BFS algorithm and the two largest real graphs, Friendster and DotaLeague for this experiments (Figure 2). Most of the platforms presents significant horizontal scalability only for Friendster, except for GraphLab, which exhibits little scala-

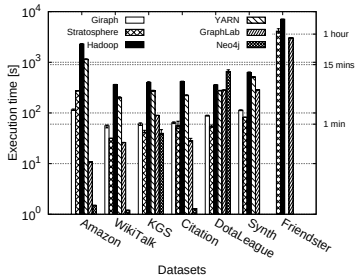

Figure 1: The execution time of algorithm BFS of all datasets of all platforms.

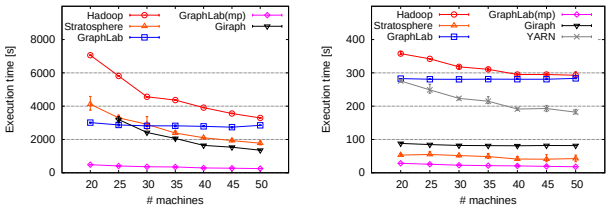

Figure 2: The horizontal scalability of processing G7 (left) and G5 (right).

bility for both datasets. The horizontal scalability of GraphLab is constrained by the graph loading phase using one single file. We thus explore tuning GraophLab: for GraphLab(mp) we split the input file into *m*ultiple separate *p*ieces, as many as the MPI processes. GraphLab(mp) has much lower execution time than GraphLab, for both datasets.

4. CONCLUSION

A quickly increasing number of data-intensive platforms can process large-scale graphs, and have thus become potentially interesting for a variety of users and application domains. To compare in-depth the performance of graph-processing platforms, and thus facilitate platform selection and tuning, we have proposed in this work an empirical method and applied it to a comprehensive performance study of six graph-processing platforms.

Our method defines an empirical performance evaluation process and selects metrics, datasets, and algorithms; thus, it acts as a *benchmarking suite* despite not covering all the methodological and practical aspects of a true benchmark. Using our method, we have conducted a first detailed, comprehensive, real-world performance evaluation of six popular platforms for graph-processing. Our results show quantitatively and comparatively the highlights and weaknesses of the tested platforms.

5. REFERENCES

[1] Y. Guo and A. Iosup. The Game Trace Archive. In *NetGames*, 2012.

[2] Y. Guo, et al. PDS-2013-004, Delft University of Technology. `http://www.pds.ewi.tudelft.nl/research-publications/technical-reports/2013/`.

[3] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *ACM SIGKDD*, 2005.

[4] Y. Low, et al. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. In *VLDB*, 2012.

[5] D. Warneke and O. Kao. Nephele: Efficient Parallel Data Processing in the Cloud. In *MTAGS*, 2009.