School of Informatics

Informatics Project Proposal
Benchmarking Graph Processing Systems

B138641
April 2019

Abstract

The aim of this project is to examine state-of-the-art graph processing systems and compare their performance and scalability in both local and parallel settings. More specifically, the project focuses on showing that oftentimes scalability comes at a cost, and in many cases such an approach is not worth the time and effort. In our research we will prove this by conducting several experiments that will demonstrate that a single-node set-up outperforms executions in multi-node environments due to the parallelization overhead that incurs.

Date: Sunday 14th April, 2019

Tutor: Pablo León-Villagrá
Supervisor: Dr Milos Nikolic

Contents

1 Introduction

Graphs have been extensively used in numerous applications and problems, such as the web, social networks, navigation, recommendation systems, and others, while the interest in the domain of graph processing and graph data analysis is constantly growing. This is based on the fact that more and more systems have been developed to process graphs and perform various graph analysis tasks in an efficient manner, such as graph processing systems [1, 2, 3, 4, 5, 6, 7, 8], and graph database systems [9, 10, 11, 12].

Although these systems have been extensively used in various applications, there have been concerns regarding some of their aspects. Firstly, many people argue that their performance is questionable and that there is a massive overhead that incurs due to the way these systems scale [13, 14, 15]. Afterwards, there is no clear indication of which system is appropriate for each occasion. For this reason, there have been many researchers focusing on creating benchmarking systems in order to evaluate the performance of such systems and observe their strengths and weaknesses [16, 17, 18, 19].

While research on benchmarking graph processing systems has provided information about the performance of these graph processing systems, it tends to be quite specific and concerns certain

aspects of these frameworks. This project was motivated by the need for a more thorough investigation that covers a broader range of cases and systems. Our research aims at showing that scalability is not always the best approach by performing various experiments under different conditions, that is, measuring performance using many different algorithms, systems, and other parameters. The main goal is to demonstrate how in many cases running such tasks with graph processing systems in single-node settings (execution on one physical machine, e.g. a laptop) is more efficient performance-wise than distributed and multi-node settings (execution on multiple physical machines, e.g. a distributed system, or a cluster).

In what follows we modularise our problem and briefly mention some of its aspects (Sections 1.1 through 1.6). Next, in Section 2, we discuss necessary and relevant knowledge to the problem's domain. Afterwards, we consider specific parts of our research and analyse steps in Section 3. Subsequently, we focus on the evaluation of our results which will be detailed in Section 4, and we highlight what we expect to see in our results in Section 5. Finally, in Section 6, we state a detailed report of the project's plan, its objectives and the deliverables.

1.1 Problem Statement

While there is research on individual graph processing systems that measures how these systems perform, there has not been extensive work exposing a more objective benchmark for these systems' performance; such a benchmark would feature the use of a wide variety of algorithms, multiple set-ups ranging from one to many cores, as well as more realistic use cases. Our work focuses on taking into account all these parameters and aims to provide a broader benchmark that will be capable of better indicating if such systems are worth using and, if so, under which settings.

1.2 Research Hypothesis and Objectives

Our hypothesis is that graph processing systems do not scale as well as they claim to do for common tasks. We were able to identify these common tasks and other useful information from a survey on graph processing [20]. Moreover, this survey revealed how the graph research community and the industry perform their analysis tasks, as well as what data they use, what kind of algorithms they run, and so forth.

We aim to support our hypothesis by conducting experiments that will demonstrate how the selected graph processing systems perform in single-node settings, as well as when scaled up, and hope to show through our results that for common tasks executions in single-node settings perform better.

1.3 Timeliness and Novelty

The graph processing domain is constantly growing and becoming more popular, given that more and more researchers and companies analyse graph data and perform graph processing tasks [21, 22]. Consequently, carrying out this research at this point is crucial since it will provide insights and indications as to how analyses and graph processing tasks can be performed, which will greatly impact the way researchers and developers approach their problems.

1.4 Significance

Possessing insights and information related to various aspects of a researcher's or developer's approach for graph data processing tasks is of great importance. This is so, because insights and information help facilitate better (in terms of performance) and less costly (in terms of

resources used) approaches. We aim to conduct a thorough experimentation that will cover widely used systems and processing tasks, as reported in [20], some of which are likely to be similar to the intended research or analysis task. As such, our work will extend current research by yielding deeper insights, and will help in the process of the decision making with regards to the technologies that will be used, the required resources, the expected performance, and other aspects, which in turn will result in better approaching a problem and providing more timely and efficient solutions.

1.5 Feasibility

Firstly, an important factor to take into account regards the configurations of the graph processing systems used. In detail, while the single-node approach is fairly easy to implement, the scaled versions require more attention, as they demand more resources. The difficulty we might face concerns our resources, given that as it is, they are limited to 8 cores. Obtaining access to a cluster environment or some computational nodes will significantly help our research, although, in case such a thing cannot be arranged, our work will be limited to single-node experimentation.

Afterwards, a secondary goal of our work is to indicate how some tasks can be carried out with the use of higher-level systems, such as graph database systems [9, 23, 10]. Examining such systems and comparing results against graph processing systems will also provide very useful information, and will result in our work covering a broader range of cases, which of course also increases the size of our audience. This part of our research will be carried out in case we manage to complete our main goal within the expected time frame. Therefore, the limiting factor to this secondary task is whether or not we have enough time left once we complete our primary objective.

1.6 Beneficiaries

Our work will impact the graph research community, as well as the industry, where developers and analysts use graph processing systems to perform their data analysis tasks. In detail, given that the results will provide insight as to whether (i) a scaled approach involving distributed graph processing systems is needed, or if (ii) a more simple, single-node, implementation is more preferable, researchers and small to medium sized companies can benefit from possessing such information because it can potentially result in them saving time, effort, and money.

This is due to the fact that scaled approaches demand more resources, which often come at a big cost. Additionally, setting up and managing these resources requires a lot of time and effort. Therefore, having a-priori knowledge about the resources needed for a specific task, or knowing the performance of an algorithm that might be similar to the one being developed, for instance, can benefit a project's planning and budget.

2 Background

While there has been extensive research in the field of graph processing systems, work on benchmarking and examining performance of these systems has been limited in certain of its aspects.

One important aspect concerns the way graph processing systems' performance is measured. More specifically, although when proposing systems, such as in [8, 6, 7], authors do showcase their implementations' performance, the cases considered mainly focus on multi-node configurations and do not provide a comparison against single-node settings. Additionally, the experiments conducted tend to feature mostly large datasets, which as seen in [20] is not always the case.

Another limiting factor in current research is the fact that although there have been researchers that have benchmarked graph processing systems [16, 17, 18, 14], in their work they consider and compare the performance of various systems against each other; instead, our focus is to show how executing tasks on single-node settings is at least as efficient as executing them on multi-node settings, as well as in many cases even better.

Lastly, regarding the part of our research that may consider also comparing graph database systems' performance, there has been limited research, such as the work done in [19], since the implementations considered in our work, for instance [9], are very recent.

3 Methodology

3.1 Approach

The core idea is to perform several graph data processing tasks (Section 3.4), with the use of different datasets that range from a hundred thousand to forty million vertices (Section 3.2), on selected graph processing systems (Section 3.3). Once we execute all these different experiments, we will compare their results against each other.

3.2 Datasets

Given that we aim at covering as many cases as possible, in order to provide broad and useful insights through our work, we consider several datasets with different characteristics; that is, different numbers of nodes, edges, edge distributions, etc. The datasets chosen are publicly available and are commonly used within the graph research field. In Table 1 we present the datasets we intend to use in our experiments and provide their characteristics.

Dataset	Nodes	Edges	Description
ego-Gplus [24]	107,614	13,673,453	Social circles from Google+
soc-Pokec [24]	1,632,803	30,622,564	Pokec online social network
cit-Patents [24]	3,774,768	16,518,948	Citation network among US Patents
twitter [25]	41,700,000	1,470,000,000	Twitter follower network

Table 1: Datasets considered along with their characteristics

Additionally, if needed, we might consider generating datasets with different distributions for their edges, for example, uniform, skewed, and others.

3.3 Graph Processing Systems

Initially, we aim at examining the graph processing systems stated below. Note that the systems are arranged in order of our interest, while those marked with an asterisk (*) are not our main focus and may be omitted in case we are limited by time.

- Timely Dataflow [1, 2]
- GraphLab [3]
- Apache Flink (Gelly) [4]
- Apache Spark (GraphX) [5, 6]

- *Apache Giraph [26]

The basis for our selection of graph processing systems was their popularity [20], as well as their timeliness.

After this, as discussed in Section 1.5, our secondary goal is to also carry out our tasks using graph database systems. In case we have time to consider this category of systems, we hope to test at least one of the following systems. Again, we order the systems based on how interested we are in examining them, their popularity, and how recent they are.

- EmptyHeaded [9, 23]
- Neo4j [10]
- *OrientDB [12]
- *ArangoDB [11]

3.4 Algorithms

Similarly to the choice of graph processing systems, our choice of algorithms is based on [20]. As such, we will consider the following types of computations.

- Finding connected components
- Neighborhood queries (e.g. finding n-th degree neighbors of a vertex)
- Finding short or shortest paths
- Subgraph matching (e.g. finding all diamond patterns)
- Ranking and centrality scores (e.g. PageRank, Betweeness or Closeness centrality)
- Aggregations (e.g. counting the numbers of triangles)
- Reachability queries (e.g. checking if vertex u is reachable from vertex v)

3.5 Environment Settings

We aim at providing two types of environment settings for our experimentations. Firstly, all experiments will be executed on a single-node setting. Afterwards, depending on the resources, experiments will also be executed in scaled environments, which will consist of multiple nodes. The scaling factor cannot be determined yet since it is dependent on the amount of resources that we will have available to us.

3.6 Limitations

Currently, we are limited by two factors. First, we have not yet gained access to a cluster or distributed environment in order to perform our experiments using the graph data processing systems in their scaled configuration. Then, another factor that limits the scope of our work is time, since the timeline for the completion of our project spans from June to mid August.

4 Evaluation

After having completed all steps and tasks mentioned in Section 3,we will carry on to analysing and interpreting our results. Our work will mainly focus on measuring the latency (total runtime) of each system to carry out each of the given tasks. Therefore, we define each system's performance as the total number of seconds it took for the system to complete the calculations of a given task.

5 Expected Outcomes

Having completed our work, we aim to show through our results that the achievable performance when executing our tasks with each of the examined graph processing systems on its single-node configuration will be at least as good as the performance on the system's multi-node settings. Such a result will provide a great argument towards avoiding unnecessary resources, or added complexity, when solving graph processing tasks or performing graph data analyses.

6 Milestones, Deliverables and Research Plan

Finally, we outline our proposed project's preset milestones in Table 2, deliverables in Table 3, as well as a timeline of all tasks in Figure 1.

Milestone	**Week**	**Description**
M_1	1	Completion of background reading
M_2	3	Completion of experiments for each graph processing system in single-node settings
M_3	5	Completion of experiments for each graph processing system in multi-node settings
M_4	7	Completion of experiments with graph database systems
M_5	10	Submission of dissertation

Table 2: Milestones defined in this project.

Deliverable	**Week**	**Description**
D_1	3	Detailed report of results and statistics of experiments with graph processing systems in single-node configuration
D_2	5	Detailed report of results and statistics of experiments with graph processing systems in multi-node configuration
D_3	7	Detailed report of results and statistics of experiments with graph database systems
D_4	10	Dissertation

Table 3: List of deliverables defined in this project.

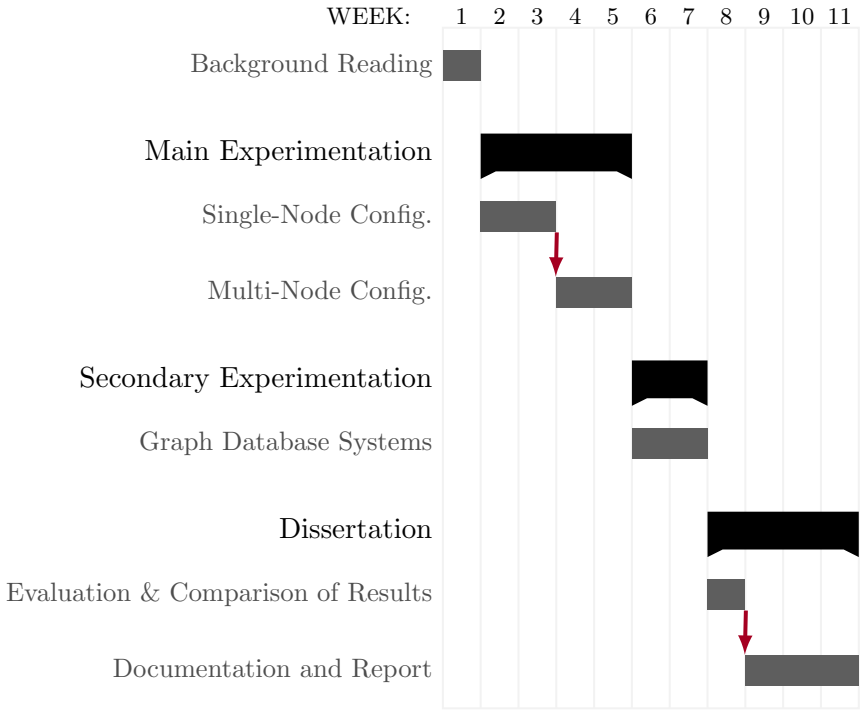

Figure 1: Gantt Chart of the activities defined for this project.

References

[1] Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: A timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 439–455, New York, NY, USA, 2013. ACM.

[2] Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: A timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 439–455, New York, NY, USA, 2013. ACM.

[3] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.*, 5(8):716–727, April 2012.

[4] Apache Flink. `https://flink.apache.org/`. Accessed: 2019-04-10.

[5] Apache Spark Graph X. `https://spark.apache.org/graphx/`. Accessed: 2019-04-10.

[6] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 599–613, Broomfield, CO, 2014. USENIX Association.

[7] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 17–30, Hollywood, CA, 2012. USENIX.

[8] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[9] Christopher R. Aberger, Andrew Lamb, Susan Tu, Andres Nötzli, Kunle Olukotun, and Christopher Ré. Emptyheaded: A relational engine for graph processing. *ACM Trans. Database Syst.*, 42(4):20:1–20:44, October 2017.

[10] Neo4j. `https://neo4j.com/`. Accessed: 2019-04-10.

[11] ArangoDB. `https://www.arangodb.com/`. Accessed: 2019-04-10.

[12] OrientDB. `https://orientdb.com/`. Accessed: 2019-04-10.

[13] Jing Fan, Adalbert Gerald Soosai Raj, and Jignesh M Patel. The case against specialized graph analytics engines. In *CIDR*, 2015.

[14] Frank McSherry, Michael Isard, and Derek G. Murray. Scalability! but at what COST? In *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, Kartause Ittingen, Switzerland, 2015. USENIX Association.

[15] Andrew Lumsdaine, Douglas Gregor, Bruce Hendrickson, and Jonathan Berry. Challenges in parallel graph processing. *Parallel Processing Letters*, 17(01):5–20, 2007.

[16] Yong Guo, Ana Lucia Varbanescu, Alexandru Iosup, Claudio Martella, and Theodore L. Willke. Benchmarking graph-processing platforms: A vision. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 289–292, New York, NY, USA, 2014. ACM.

[17] Yong Guo, Marcin Biczak, Ana Lucia Varbanescu, Alexandru Iosup, Claudio Martella, and Theodore L Willke. How well do graph-processing platforms perform? an empirical performance evaluation and analysis. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 395–404. IEEE, 2014.

[18] Mihai Capotă, Tim Hegeman, Alexandru Iosup, Arnau Prat-Pérez, Orri Erling, and Peter Boncz. Graphalytics: A big data benchmark for graph-processing platforms. In *Proceedings of the GRADES'15*, GRADES'15, pages 7:1–7:6, New York, NY, USA, 2015. ACM.

[19] Marek Ciglan, Alex Averbuch, and Ladialav Hluchy. Benchmarking traversal operations over graph databases. In *2012 IEEE 28th International Conference on Data Engineering Workshops*, pages 186–189. IEEE, 2012.

[20] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. The ubiquity of large graphs and surprising challenges of graph processing. *Proc. VLDB Endow.*, 11(4):420–431, December 2017.

[21] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One trillion edges: Graph processing at facebook-scale. *Proc. VLDB Endow.*, 8(12):1804–1815, August 2015.

[22] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1031–1040, New York, NY, USA, 2011. ACM.

[23] EmptyHeaded. `https://github.com/HazyResearch/EmptyHeaded`. Accessed: 2019-04-10.

[24] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[25] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[26] Apache Giraph. `https://giraph.apache.org/`. Accessed: 2019-04-10.