

Understanding the Network Performance of Distributed Graph Processing

Anil Yelam, Audrey Randall

Abstract

An abstract is like a movie trailer. It offers a preview, highlights key points, and helps the audience decide whether to view the entire work. Abstracts are the pivot of a research paper because many journal editorial boards screen manuscripts only on the basis of the abstract.

Notes from Heidari et al. [6] Look at the paper for citations of all the statements below.

Q. Which algorithm to choose? Needs to be global, traversing the whole graph. Examples include page rank and connected components, degree distribution.

1 Introduction

The importance of balanced execution of big data frameworks (Papers to link: Osterhout[8], Themis[9], TritonSort[10]). Traditionally, while network was a part of the balancing equation, the network bandwidths considered were still less than 10gbps. However, with 40g networks becoming common and much higher bandwidths becoming available, we want to focus running balanced algorithms at these breakneck speeds.

Why we picked graph processing? Balanced sort has been well studied, there has been relatively less work in balanced performance for graph processing algorithms, especially at high bandwidths.

What we did or found out, in a nutshell.

Q. Which programming model? Programming models used in existing frameworks (Section 3) - distributed vs shared memory architecture. While shared mem architectures work well for gigabtes of data (Scalability but at what cost paper), they don't scale to terabytes. There's also GPU and FPGA based acceleration proposed in some works, but we want to restrict ourselves to frameworks that run on commodity clusters.

Q. Which Programming abstraction?: 1. Vertex Centric: how it works and the shortcomings. Discuss some of the example frameworks. 2. Edge centric: how it works and the shortcomings. Discuss some of the example frameworks. 3. Component centric: how it works and the shortcomings. Discuss some of the example frameworks. 4. Others: Like MapReduce and other general purpose big-data processing frameworks.

2 Background

All about the graph processing frameworks and their computational models, basically a brief summary of [6]. And a bunch of related papers we have read, like comparison papers [5, 1] and some earlier graph processing systems these papers point to - such as Pregel[7], PowerGraph [4] and others.

Q. What sort of distributed coordination: 1. Synchronous - using barriers between iterations, makes life easier for programmer but sensitive to stragglers. 2. Asynchronous - better performance, but much complex to program and ensure error recovery. 3. Hybrid

Q. Input partitioning: 1. Static vs Dynamic: Do we repartition graph at runtime depending on algo-

rithm or straggler issues. 2. Edge cut vs Vertex cut: How do we partition a non-uniform graph? What's best for the algorithm? This is one of the key design choices that network performance could affect. Usually frameworks make one of these choices to minimize number of edges or vertices between nodes to save on network, but is there a better way assuming infinite network bandwidth? (Think more about this!). This is something we could vary in our experiments.

Q. Others: Disk vs Memory based, fault tolerance with detection or recovery where choices are either clear or not important for our purposes.

3 Discussion

Here goes the discussion on what choices we made for our implementation w.r.t all the choices we discussed in the previous section. We will go with global algorithm like pagerank, on input graphs with uniform vertex and edge distribution to start with. We will go with synchronous model inspite of some performance disadvantages as it makes programming easier and is widely used. Since the kind of partitioning (vertex centric, edge centric or graph centric) should not matter that much for very uniform graphs, we will start with the most popular one i.e., vertex centric model. Choosing uniform graphs should help alleviate the skewed straggler problems affecting the synchronous model as well. We will go with static partitioning since page rank keeps all the vertices active throughout the run.

Then we do a theoretical analysis on the kind of network performance we could get. Given nodes of CPU c , memory m and network n for an input graph $g = (v, e)$ what would a perfectly written distributed page rank program with above characteristics give us in terms of network performance?

We pick a real-world processing framework (i.e., giraph) with the same characteristics and run it. How does it do with our expectations? What optimizations we had to do? What is the unexpected part of it? What overheads does it add?

4 Apache Giraph

Description of Apache Giraph[3]. And bunch of optimizations made by Facebook[2].

5 Implementation

Experimental setup and the steps involved, in detail.

6 Evaluation

1. Section on characteristics of the input graphs we picked, such as its vertex degree distribution, and what we expect for this particular input. 2. The results we saw

References

- [1] AMMAR, K., AND ÖZSU, M. T. Experimental analysis of distributed graph systems. *Proc. VLDB Endow.* 11, 10 (June 2018), 1151–1164.
- [2] CHING, A., EDUNOV, S., KABILJO, M., LOGOTHETIS, D., AND MUTHUKRISHNAN, S. One trillion edges: Graph processing at facebook-scale. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1804–1815.
- [3] GIRAPH. <http://giraph.apache.org/>. accessed: 2019-05-03.
- [4] GONZALEZ, J. E., LOW, Y., GU, H., BICKSON, D., AND GUESTRIN, C. Powergraph: Distributed graph-parallel computation on natural graphs. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2012), OSDI'12, USENIX Association, pp. 17–30.
- [5] GUO, Y., BICZAK, M., VARBANESCU, A. L., IOSUP, A., MARTELLA, C., AND WILLKE, T. L. How well do graph-processing platforms perform? an empirical performance evaluation and analysis. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium* (Washington, DC, USA, 2014), IPDPS '14, IEEE Computer Society, pp. 395–404.
- [6] HEIDARI, S., SIMMHAN, Y., CALHEIROS, R. N., AND BUYYA, R. Scalable graph processing frameworks: A taxonomy and open challenges. *ACM Comput. Surv.* 51, 3 (June 2018), 60:1–60:53.
- [7] MALEWICZ, G., AUSTERN, M. H., BIK, A. J., DEHNERT, J. C., HORN, I., LEISER, N., AND CZAJKOWSKI, G. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 135–146.

- [8] OUSTERHOUT, K., RASTI, R., RATNASAMY, S., SHENKER, S., AND CHUN, B.-G. Making sense of performance in data analytics frameworks. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2015), NSDI'15, USENIX Association, pp. 293–307.
- [9] RASMUSSEN, A., LAM, V. T., CONLEY, M., PORTER, G., KAPOOR, R., AND VAHDAT, A. Themis: An i/o-efficient mapreduce. In *Proceedings of the Third ACM Symposium on Cloud Computing* (New York, NY, USA, 2012), SoCC '12, ACM, pp. 13:1–13:14.
- [10] RASMUSSEN, A., PORTER, G., CONLEY, M., MADHYASTHA, H. V., MYSORE, R. N., PUCHER, A., AND VAHDAT, A. Tritonsort: A balanced and energy-efficient large-scale sorting system. *ACM Trans. Comput. Syst.* 31, 1 (Feb. 2013), 3:1–3:28.