A Comparison of Parallel Graph Processing Platforms

Samuel Pollard
Department of Computer and Information Science
University of Oregon
Eugene, OR, USA
Email: spollard@cs.uoregon.edu

Boyana Norris
Department of Computer and Information Science
University of Oregon
Eugene, OR, USA
Email: norris@cs.uoregon.edu

***Abstract*—In this paper we analyze the performance of four parallel graph processing systems to determine their performance and scalability using synthetic and real-world datasets. We perform analysis for three algorithms: breadth first serach, single source shortest paths, and PageRank. This paper examines previously overlooked aspects of parallel programming performance such as file I/O in addition to a more detailed performance analysis.**

I. Introduction

Our research is motivated by the current state of parallel graph processing. The most comprehensive survey, released in 2014, identified and taxonomized over 80 different parallel graph processing systems [1]. These systems operate with a wide range of parallelism paradigms and target architectures such as GPU [2], [3], shared memory [pretty much everything], a combination of CPU and GPU [4], distributed database querying, [5], distributed filesystem based approaches [6], distributed memory with MPI [7], domain-specific languages [8], as well as novel communication schemes [9].

Since 2014, the problem has compounded with the addition of even more proprietary and open source projects such as [10], [11] [name some more]. At the outset, this plethora of choices makes the question, "which system is the best for my problem?" daunting. There has even been a propagation of so-called "reference implementations" which implement the most common graph algorithms [cite GAP, GraphBIG, Galois]. Thus, even selecting a standard and a benchmark over which to compare various implementations is nontrivial. To quote Andrew Tanenbaum, "The nice thing about standards is that you have so many to choose from."

Another issue among parallel graph processing systems is the lack of comprehensive comparisons. One possible reason for this is the considerable effort involved in getting each system to run: satisfying dependencies and ensuring correctly formatted data the data are nontrivial for most tasks. Beyond this, each system may have a different method of measuring performance. Thus, one of the contributions of this paper is to provide a "level playing field" for each graph processing system. Graphalytics [12] attempts to remedy this but [state the limitations of graphalytics]

II. Algorithms

The canonical performance leaderboard for parallel graph processing is the Graph500 [13]. The advantage of the Graph500 is it provides standardized measurement specifications and dataset generation. The primary drawback with the Graph500 is it measures a single algorithm: breadth first search (BFS).

This report attempts to add similar rigor to other graph algorithms. This is done by borrowing heavily from the Graph500 specification; we use a Kronecker graph [14] with initial parameters of $A = 0.57, B = 0.19, C = 0.19$, and $D = 1-(A+B+C) = 0.05$. The Graph500 Benchmark 1 ("Search") is concerned with two kernels: the creation of a graph data structure from an edge list stored in RAM and the BFS[1]. Some clarifications are in order: for the first kernel, the time to read the graph from disk is not considered. Furthermore, these data structures need only be created once and BFS is performed on 64 roots. The second kernel creates a tree with distances from the root, but need only label the existing data structure (and not output the entire graph, for example).

One straightforward extension is computing the Single-Source Shortest Paths algorithm (SSSP). We use the same graph and the same source vertices as in BFS. We use SSSP and PageRank in this paper because of their simplicity and ubiquity.

III. The Existing Systems

This report explores a small sample of the existing graph processing platforms with a focus on the so-called "reference implementations:" the Graph500[2], GAP [15], and GraphBIG [16]. GraphMat is also used because it has been cited as the most performant [cite]. Our target is shared memory CPU processing.

Other popular libraries such as the Parallel Boost Graph Library [17] are not considered here because they are a

[1]For a complete specification, see http://graph500.org/specifications

[2]We used version 2.1.4 from http://graph500.org/referencecode.

PITA to get working and the authors never responded to my emails.

IV. Machine Specifications

Table I shows the specifications of the research computer (named Arya).

CPU Model	Intel Xeon(R) E5-2699 v3 @ 2.30GHz
CPU Sockets	2
CPU Cores	72
CPU Clock	3600MHz
RAM Size	256GB
RAM Freq	1866MHz
Max RAM Freq	2133MHz
GPU Model	GM204 [GeForce GTX 980]

TABLE I

Machine specifications. The disparity between the CPU's advertised clock speed and the "CPU Clock" row is a result of the Turbo Boost technology which can increase the clock speed to a limit. We use the manufacturer's published maximum clock speeds which can be found at http://ark.intel.com.

V. Performance

In Table II, BFS is breadth-first search, SSSP is single-source shortest paths, LCC is local clustering coefficient, PR is PageRank, CDLP is community detection using label propagation, and WCC is weakly connected components. For the algorithms used, see [19].

	openg	powergraph
CDLP	181.667	1226
PR	302.333	974
LCC	321.333	1036.67
WCC	87.6667	697.667
SSSP	4061.33	29022.3

TABLE II

Performance Results for the `dota-league` dataset with 61,670 vertices and 50,870,313 edges.

System	Load Graph	Make Data Structure	Run BFS
Graph500	0.3474	0.3971	0.003380
GraphBIG	37.22	N/A	0.1528
GraphMat	0.1511	1.101	0.1031
GAP	2.351	0.1935	0.001393

TABLE III

Times are in seconds. The Graph500 generates the graph instead of loading it into a file. GraphBIG builds the graph and reads in the file simultaneously. These results were averaged across 64 roots. [TODO: Why does the first root often have degree 0?]

[Compare lines of code for implementation]

VI. Graph Processing Taxonomy

This is in the spirit of [1]. Here, "|" means "or" and "+" means "and." FOSS means Free and Open Source Software. The quotes around "yes" for HPC mean that the product claims to be amenable to high performance computing. Whether these actually achieve their goal is one of the purposes of this project.

VII. Conclusion

We have presented an updated survey of parallel graph processing frameworks supplementary to [1]. From this, we have selected a representative subset of frameworks on which performance is analyzed and have stored these results in a database. To facilitate parallel graph processing, hardware information and performance results are automatically populated (as were all the tables in this paper). These performance results are then used to provide simple recommendations of the optimally-performing framework given a particular algorithm and problem size.

References

[1] N. Doekemeijer and A. L. Varbanescu, "A survey of parallel graph processing frameworks," Delft University of Technology, Tech. Rep., 2014.

[2] J. Zhong and B. He, "Medusa: Simplified graph processing on gpus," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1543–1552, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2013.111

[3] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "Pegasus: A peta-scale graph mining system implementation and observations," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ser. ICDM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 229–238. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2009.14

[4] A. Gharaibeh, L. Beltrão Costa, E. Santos-Neto, and M. Ripeanu, "A yoke of oxen and a thousand chickens for heavy lifting graph processing," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '12. New York, NY, USA: ACM, 2012, pp. 345–354. [Online]. Available: http://doi.acm.org/10.1145/2370816.2370866

[5] M. A. Rodriguez, "The gremlin graph traversal machine and language," *CoRR*, vol. abs/1508.03843, 2015. [Online]. Available: http://arxiv.org/abs/1508.03843

[6] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "Graphx: A resilient distributed graph system on spark," in *First International Workshop on Graph Data Management Experiences and Systems*, ser. GRADES '13. New York, NY, USA: ACM, 2013, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/2484425.2484427

[7] S. Hong, S. Depner, T. Manhardt, J. Van Der Lugt, M. Verstraaten, and H. Chafi, "Pgx.d: A fast distributed graph processing engine," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: ACM, 2015, pp. 58:1–58:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807620

[8] S. Hong, H. Chafi, E. Sedlar, and K. Olukotun, "Green-marl: A dsl for easy and efficient graph analysis," in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVII. New York, NY, USA: ACM, 2012, pp. 349–362. [Online]. Available: http://doi.acm.org/10.1145/2150976.2151013

[9] N. Edmonds, J. Willcock, and A. Lumsdaine, "Expressing graph algorithms using generalized active messages," in *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '13. New York, NY, USA: ACM, 2013, pp. 289–290. [Online]. Available: http://doi.acm.org/10.1145/2442516.2442549

[10] U. Cheramangalath, R. Nasre, and Y. N. Srikant, "Falcon: A graph manipulation language for heterogeneous systems," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 4, pp. 54:1–54:27, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2842618

Name	Type	HPC	Parallelism	Target	FOSS	Source	Notes
PowerGraph	Framework	“yes”	both	CPU	yes	[18]	[a]
GraphBIG	Benchmark	“yes”	shared	CPU|GPU	yes	[16]	[b]

TABLE IV
TOOLS USED FOR GRAPH PROCESSING

[a]The current version is a closed-source product by Turi though PowerGraph v2.2 is on Github.
[b]Only works on Linux.

[11] Y. Perez, R. Sosič, A. Banerjee, R. Puttagunta, M. Raison, P. Shah, and J. Leskovec, “Ringo: Interactive graph analytics on big-memory machines,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’15. New York, NY, USA: ACM, 2015, pp. 1105–1110. [Online]. Available: http://doi.acm.org/10.1145/2723372.2735369

[12] M. Capotă, T. Hegeman, A. Iosup, A. Prat-Pérez, O. Erling, and P. Boncz, “Graphalytics: A big data benchmark for graph-processing platforms,” in *Proceedings of the GRADES’15*, ser. GRADES’15. New York, NY, USA: ACM, 2015, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/2764947.2764954

[13] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Arg, “Introducing the graph500,” Cray User’s Group, Tech. Rep., 2010.

[14] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, “Kronecker graphs: An approach to modeling networks,” *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756039

[15] S. Beamer, K. Asanovic, and D. A. Patterson, “The GAP benchmark suite,” *CoRR*, vol. abs/1508.03619, 2015. [Online]. Available: http://arxiv.org/abs/1508.03619

[16] L. Nai, Y. Xia, I. G. Tanase, H. Kim, and C.-Y. Lin, “GraphBIG: Understanding graph computing in the context of industrial solutions,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’15. New York, NY, USA: ACM, 2015, pp. 69:1–69:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807626

[17] D. Gregor, N. Edmonds, B. Barrett, and A. Lumsdaine, “The parallel boost graph library,” http://www.osl.iu.edu/research/pbgl, 2005.

[18] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. 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)*. Hollywood, CA: USENIX, 2012, pp. 17–30. [Online]. Available: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez

[19] A. Iosup, T. Hegeman, W. L. Ngai, S. Heldens, A. P. PÃľrez, T. Manhardt, H. Chafi, M. Capota, N. Sundaram, M. Anderson, I. G. Tanase, Y. Xia, L. Nai, and P. Boncz, “Ldbc graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms, a technical report,” Delft University of Technology, Tech. Rep., 2016.