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*—This is a survey of existing graph analytics frameworks.**

I. Introduction

[Talk about WHY we care about graph processing, and parallel in particular. Talk about what parallel graph processing is.]

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 such as GPU [Medusa], shared memory [pretty much everything], a combination of CPU and GPU [2], distributed database querying, [3], distributed filesystem based approaches [4], distributed memory with MPI [cite], domain-specific languages [Green Marl], as well as novel communication schemes [activemessages, sockets].

Since 2014, the problem has compounded with the addition of even more proprietary and open source projects such as [5], [6] [name some more]. At the outset, this plethora of choices makes the question, "which system is the best for my problem?" incredibly difficult. There has even been a propagation of so-called "reference implementations" which provide implementations of the most common graph algorithms [GAP, GraphBIG]. 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 with the parallel graph processing library designers is the lack of comprehensive comparison. One possible reason for this is the considerable effort of getting each library and package to work, satisfying dependencies, and ensuring the data is in the correct format for each system. 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 [7] attempts to remedy this but has not seen widespread adoption.

II. 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. The manufacturer's published maximum clock speeds can be found at http://ark.intel.com.

	BFS	SSSP	PR
Graph500			
PBGL			
GAP			
GraphBIG			
Galois			
PowerGraph			

TABLE II
Performance. Note that implementations of every algorithm are not availabe for every platform.

III. Algorithms and Systems

The canonical performance leaderboard for paralell graph processing is the Graph500 [8]. The advantage of the Graph500 is it provides standardized measurement requirements and dataset generation. The primary drawback with using reference implementations for the Graph500 is the standard only supports a single algorithm: breadth first search.

This report explores a small sample of the existing graph processing platforms with a focus on the so-called "reference implementations:" the Graph500 (from http://www.graph500.org), GAP [9], and GraphBIG [10]. Parallel Boost Graph Library [11] and PowerGraph [12] implementations are also provided because of their popularity and the availability of high quality reference implementationsF.

IV. Performance

Graphalytics without the use of the Granula plugin produces performance measurement in two forms: runtime in seconds and traversed edges per second.

In Table III, 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 [13].

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

V. 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.

VI. 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] 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

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

[4] 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

[5] 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

[6] 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

[7] 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

[8] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Arg, "Introducing the graph500," Cray User's Group, Tech. Rep., 2010.

[9] 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

[10] 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

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

[12] 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

[13] A. Iosup, T. Hegeman, W. L. Ngai, S. Heldens, A. P. PĂŤrez, 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.

Name	Type	HPC	Parallelism	Target	FOSS	Source	Notes
PowerGraph	Framework	“yes”	both	CPU	yes	[12]	[a]
GraphBIG	Benchmark	“yes”	shared	CPU|GPU	yes	[10]	[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.