Simplifying Parallel Graph Processing:

Directed Research Project Contract

Sam Pollard (spollard@cs.uoregon.edu), University of Oregon

November 23, 2016

1 Introduction

Graph processing has important differences from other High Performance Computing (HPC) applications. For example, floating point operations per second mean little when performing a breadth first search. Furthermore, properties of the input graph such as its sparsity can play a profound role in an algorithm's performance. Because of these differences, there has been movement toward standardizing performance measurements specifically directed at graph processing. The largest example is the Graph500 [6], a ranking of supercomputers in the spirit of the Top500. Other projects such as Graphalytics [4] attempt to compare performance across various platforms while projects such as the Graph Algorithm Platform Benchmark Suite [1] and GraphBIG [7] attempt to provide high quality, reference implementations next to which performance can be compared. Here, we use the definition of *platform* defined in [4]: "the combined hardware, software, and programming system that is being used to complete a graph processing task."

There are a plethora of programming paradigms, domain specific languages, and libraries for parallel graph processing. Together these cover a diverse range of applications, programming languages, and target architectures. Whereas the Graph500 measures scalability, there is impressive research on achieving the highest performance per core on shared memory systems. For example, the GraphChi platform in particular showed performance on a personal computer can have performance comparable to a medium-sized cluster [5].

Publications introducing new platforms generally compare their performance to popular existing platforms on a few standard datasets. However, these performance measurements are naturally biased towards the authors' works. Thus, when attempting to solve a graph-based problem, the question of *which* platform to use is daunting. Beyond this, platforms have unique input file formats, programming paradigms, configuration demands, and dependencies, all of which combine to yield a steep learning curve and a lack of portability, which are concerns applicable to HPC in general.

2 Project Description

This project aims to analyze the performance of graph algorithms and recommends an optimal platform to the user based on this analysis. The project aims to look beyond the number of traversed edges per second (TEPS) on just breadth-first search as in the Graph500. Potential measurements include strong and weak scalability, load balancing, memory utilization, and disk usage. The Tuning and Analysis Utilities (TAU) will be investigated as a potential source for these measurements [8]. Beyond performance analysis, hardware and middleware specifications will be reported such as data transport mechanisms and scheduling paradigms as inspired by [2].

This project will begin by building on top of the graph benchmarking tool Graphalytics (described in [4]) with increased focus on describing the hardware, communication paradigms, scheduling, and other lower-level details as recommended by [2]. Preliminary research has shown Graphalytics to be unportable and challenging to extend, so this project will provide a much-needed simplification over existing frameworks. Furthermore, this project will standardize performance analysis for each platform. This will be accomplished through comprehensive analysis of all prominent platforms. Additionally, modeling of the target architectures will be used to explain the observations performance and to predict performance on new architectures.

Lastly, this project aims to use these data to provide recommendations so users can make informed decisions about which platform is optimal for a given architecture, problem size, and algorithm. At their simplest, these recommendations will be based on existing measurements and of the form: "given machine X running algorithm Y on a problem of size Z, the optimal platform is P." This format is restrictive (the user may not know X, Y, or Z) and does not address the cost of programming on a new platform for each problem. The first step to address this would be to use a model of graph processing performance to predict the optimal platform with less information. Time permitting, the use of a domain specific language to translate high level algorithm specifications to the optimal platform will be investigated.

3 Timeline

Fall '16. Continue literature search about benchmarking for graph processing algorithms. Decide on two or three platforms as starting points to analyze performance and tabulate basic performance measurements. Look into methods to increase portability such as Spack [3].

Winter. Get preliminary measurements and begin to analyze which platforms are optimal for a given input, algorithm, and hardware. Create a model to explain observed performance differences. Use these data to make simple platform recommendations. Coalesce all noteworthy graph processing platforms, prepare an exhaustive list of these platforms. This may be the beginnings of an area exam survey paper. Time and resource-permitting: install a power monitor on the research hardware to increase quality and depth of performance measurement.

Spring. Keep up with the state of the art by adding more platforms and architectures to the measurements. Enhance portability by testing on various architectures. Compile performance results and begin writing the paper.

Summer. Enhance model of platform, algorithm, and dataset performance. Improve recommender to work in cases where the user does not have complete knowledge of the problem to be solved. Write paper and investigate which conference or journal is appropriate.

Fall '17. Submit the paper to a conference or journal. Present to DRP committee. Time permitting: investigate viability of a domain specific language (DSL). Develop the DSL so it can express the most popular algorithms on the most popular platforms.

4 Deliverables

1. A comprehensive list of graph processing platforms and descriptions of them. Categorize these by prominence so analysis and porting effort is well-spent.
2. Machine and human readable, complete, precise performance analysis for several graph processing platforms on the existing research hardware[1] and the UO cluster. These measurements will be performed on several standardized datasets and the most common algorithms such as breadth first search, local clustering coefficients, single-source shortest paths, PageRank, and community detection.
3. A recommender tool which can be easily run on various architectures.
4. A paper to submit to a conference or journal.

[1]This machine is named Arya and is an Intel Xeon-based machine with 72 cores, an NVIDIA GTX 980 GPU and 256GB RAM

5. A workshop-style tutorial of this recommender tool.
6. Time permitting: a proof of concept DSL.

5 Committee Members

1. Boyana Norris
2. Allen Malony
3. Zena Ariola

References

[1] Beamer, S. *Understanding and Improving Graph Algorithm Performance*. PhD thesis, EECS Department, University of California, Berkeley, Sep 2016.

[2] Firoz, J. S., Kanewala, T. A., Zalewski, M., Barnas, M., and Lumsdaine, A. Context matters: Distributed graph algorithms and runtime systems: A case study of distributed graph traversals. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (New York, NY, USA, 2016), PASC '16, ACM, pp. 12:1–12:10.

[3] Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., de Supinski, B. R., and Futral, S. The spack package manager: Bringing order to hpc software chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2015), SC '15, ACM, pp. 40:1–40:12.

[4] 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 *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International* (May 2014), pp. 395–404.

[5] Kyrola, A., Blelloch, G., and Guestrin, C. Graphchi: Large-scale graph computation on just a pc. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)* (Hollywood, CA, 2012), USENIX, pp. 31–46.

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

[7] Nai, L., Xia, Y., Tanase, I. G., Kim, H., and Lin, C.-Y. GraphBIG: Understanding graph computing in the context of industrial solutions. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2015), SC '15, ACM, pp. 69:1–69:12.

[8] Shende, S. S., and Malony, A. D. The tau parallel performance system. *Int. J. High Perform. Comput. Appl. 20*, 2 (May 2006), 287–311.