On the Performance Modeling of Graph Processing Primitives on GPUs

Merijn Verstraaten, Ana Lucia Varbanescu, Souley Madougou, Cees de Laat Unviersity of Amsterdam
The Netherlands
Email: {m.e.verstraaten, a.l.varbanescu, s.madougou, delaat}@uva.nl

I. Introduction

Graph processing is an important part of data science, due to the flexibility of graphs as a model for highly interrelated data. A lot of research is invested in parallel and distributed solutions for graph processing [1], [2], [3], [4], [5], [6], [7], [8]. Most such frameworks simplify graph processing by maintaining a separation between a front-end that uses high-level primitives or DSLs, and a back-end that provides high-performance implementations of these primitives.

There are many ways to implement the same primitive, and different implementations perform best on different hardware: a model that performs well on a CPU might perform horribly on a GPU or on another accelerator. Understading which implementation to choose is essential for the performance of the application at hand. To make matters worse, performance does not depend only on the underlying hardware, but also on the structural properties of the graph being processed. In this context, little progress has been made in quntifying this impact and its correlation to the graph properties [9], [10].

In this work, we model the performance of graph processing primitives running on GPUs. Our goal is to understand the methodologies and tools to be used to eventually derive a predictive model for such primitives. To do so, we compare two different modeling approaches - analytical modeling and statistical methods - and verify their usability on a specific primitive - i.e., neighbour iteration (seen in algorithms such as Page-Rank, label propagation, or graph coloring). Our analytical model focuses on the *work* and *time-complexity* of the algorithm [11]. Our statistical modeling combines machine learning with performance counters data [12].

Our results demonstrate that a predictive model can be built for such primitives only when combining analytical and statistical modeling. However, the modeling requires quite some effort, and the prediction requires a complex set of parameters combining application, platform, *and* dataset features.

II. Background and Approach

A. PageRank

PageRank is an algorithm that calculates rankings of vertices by estimating how important they are. Importance is correlated to the number of edges incoming from other vertices.

PageRank usually implemented for each node, iteratively, using two steps: (1) compute the incoming page rank from the previous iteration, and (2) normalize the new pagerank using a damping factor. These operations are repeated until convergence.

B. Approach and Results

We analyze four parallel versions of PageRank on the GPU [11]: edge-centric, vertex-centric push, vertex-centric pull, and vertex-centric pull no-div. Our initial results show significant performance differences between these implementations, related to both the platform and the input graph.

Analytical Modeling: We have devised performance models for our 4 PageRank versions by (1) evaluating the work of the algorithm, and (2) emulating the GPU work scheduler to devise a runtime estimate. The work is expressed as a function of the graph properties. The GPU execution model takes into account the high-level platform properties. Unfortunately, this simplistic model is not accurate (error up to 50% in some cases) due to the ovrsimplified GPU execution model.

Statistical Modeling: We demonstrate the use of the Black-Forest tool [12] for our irregular graph processing primitive. This attempt exposes two new challenges: (1) much more profiling data is needed, and (2) there is not enough input data for profiling. To generate suitable input instances, we have built a graph generator that allows the user to built synthetic graphs with a required set of properties[1]. These graphs are being used to collect performance counter data from the execution of our four different algorithms on the GPU. Further, with the help of BlackForest, we use the data to build the performance model *and* identify the performance bottlenecks.

Refining the Analytical Model: Using the bottleneck analysis, we attempt to refine the GPU execution model we used in the analytical modeling. Specifically, we aim to refine our initial over-simplified GPU execution model to capture these bottlenecks. Our expectation[2] is that the accuracy of the analytical model will increase significantly once the GPU execution model is refined.

III. Conclusion

Graph processing is a new challenge for HPC platforms. Therefore, new algorithms and new processing systems are emerging quickly to address its performance needs. However, most of these algorithms are not thoroughly analyzed in terms of performance. Instead, a few empirical results that prove them faster than (some) alternatives are considered sufficient proof of their superiority. This is not sufficient for choosing these algorithms as backbone implementations for processing systems running on the truly large graphs.

In this work, we attempt a more systematic approach to the performance analysis of such algorithms via performance modeling. Specifically, we compared the analytical and statistical modeling of graph processing, demonstrating the advantages and disadvantages of both. We have further combined them to build a better analytical model. We have validated our model on the neighbour iteration primitive running on GPUs, and found that this approach is feasible. However, for the near future, it is unclear whether this approach can be automated. We are now in the process of validating it for new primitives and algorithms.

[1]While this is a much challenging problem than it seems, it is outside the scope of this paper

[2]This is work in progress, to be completed in May.

References

[1] C. Avery, "Giraph: Large-scale graph processing infrastructure on hadoop," *Proceedings of the Hadoop Summit. Santa Clara*, 2011.

[2] 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*. ACM, 2015, p. 58.

[3] Y. Guo, A. L. Varbanescu, A. Iosup, and D. Epema, "An Empirical Performance Evaluation of GPU-Enabled Graph-Processing Systems," in *CCGrid'15*, 2015.

[4] Y. Lu, J. Cheng, D. Yan, and H. Wu, "Large-Scale Distributed Graph Computing Systems: An Experimental Evaluation," *VLDB*, 2014.

[5] M. Han, K. Daudjee, K. Ammar, M. T. Ozsu, X. Wang, and T. Jin, "An Experimental Comparison of Pregel-Like Graph Processing Systems," *VLDB*, 2014.

[6] B. Elser and A. Montresor, "An Evaluation Study of Bigdata Frameworks for Graph Processing," in *Big Data*, 2013.

[7] N. Satish, N. Sundaram, M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey, "Navigating the Maze of Graph Analytics Frameworks using Massive Graph Datasets," in *SIGMOD*, 2014.

[8] Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke, "How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis," in *IPDPS*, 2014.

[9] A. L. Varbanescu, M. Verstraaten, A. Penders, H. Sips, and C. de Laat, "Can Portability Improve Performance? An Empirical Study of Parallel Graph Analytics," in *ICPE'15*, 2015.

[10] M. Verstraaten, A. L. Varbanescu, and C. de Laat, "Quantifying the performance impact of graph structure on neighbour iteration strategies for pagerank," in *Euro-Par 2015: Parallel Processing Workshops*. Springer, 2015, pp. 528–540.

[11] ——, "Quantifying the Performance Impact of Graph Structure on Neighbour Iteration Strategies for PageRank," in *EuroPar 2015 Workshops*, 2015.

[12] S. Madougou, A. L. Varbanescu, C. de Laat, and R. van Nieupoort, "A Tool for Bottleneck Analysis and Performance Prediction for GPU-accelerated applications," in *IPDPSW 2016*. IEEE, 2016.