Graph-Optimizer: Towards Predictable Large-Scale Graph Processing Workloads

Ana-Lucia Varbanescu
a.l.varbanescu@utwente.nl
University of Twente
Enschede, The Netherlands

Andrea Bartolini
a.bartolini@unibo.it
University of Bologna
Bologna, Italy

ABSTRACT

We present Graph-Optimizer, a module of the Graph-Massivizer platform, that uses optimised BGOs and composition rules to capture and model a graph processing workload, and further combines the workload model with hardware and infrastructure models, predicting performance and energy consumption. Combined with design space exploration, such predictions enable co-designed workload implementations to fit a requested performance objective and guarantee their performance bounds during execution.

CCS CONCEPTS

• **Hardware** → *Analysis and design of emerging devices and systems*; • **Computer systems organization** → **Parallel architectures**; *Distributed architectures*; • **Software and its engineering** → **Software performance**; **Massively parallel systems**; **Designing software**; • **Computing methodologies** → **Parallel computing methodologies**; **Modeling methodologies**.

KEYWORDS

Graph-Optimizer, Graph Massivizer, basic graph operations (BGOs), model-based system co-design, heterogeneous computing

ACM Reference Format:
Ana-Lucia Varbanescu and Andrea Bartolini. 2023. Graph-Optimizer: Towards Predictable Large-Scale Graph Processing Workloads. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3578245.3585340

1 INTRODUCTION

Graph processing applications focus on the analysis of data represented as a set of entities (vertices) and their connections (edges). Graph processing workloads are common in many application domains - from social networks analysis to logistics, and from text or image analysis to bioinformatics - because they offer the opportunity to analyse interconnected entities, as well as determine and predict their evolution. As such, there are many types of graphs [1, 11, 12], and many more algorithms to process them [8, 9].

As the data increases in size (number of entities) and complexity (number and types of edges) [15], graph processing workloads suffer in terms of performance and scalability [5, 7]. Thus, graph processing needs new algorithms, suitable for and heterogeneous, accelerated systems [3, 10, 18].

In this work we present Graph-Optimizer, a framework that provides model-based performance guarantees for graph processing workloads on heterogeneous, accelerated systems. Graph-Optimizer combines graph-aware BGO models for different processors with data partitioning and communication models, to provide performance and energy consumption estimates. In this talk, we introduce the design of Graph-Optimizer, we emphasize its advantages over state-of-the-art, and indicate how, given BGO models, they can be combined towards a predictive model. We further present the roadmap for building Graph-Optimizer.

2 RELATED WORK

Modeling graph processing workloads and predicting their perfomrance remain challenging for three reasons: workloads complexity, hardware complexity, and data-dependent performance.

Graph processing workloads. Current work on optimizing graph processing workloads focuses on two main directions: the design and implementation of dedicated, hand-tuned (parallel and/or distributed) algorithms, or the construction of graph processing platforms or systems (GPPs/GPSs).

Hand-tuned graph processing algorithms are designed and optimized for a given operation, platform and, often, type of graph. For example, for breadth-first search traversal, thereare more than 20 GPU-based algorithms, and several approaches developed exclusively to (dynamically) combine these to further improve performance [2, 18, 19]. Similar efforts exist for PageRank and Centrality metrics. Hand-tuned algorithms remain difficult to design and implement, and are often non-portable across systems and/or graphs.

Build for usability, GPPs/GPSs offer a convenient alternative to hand-tuned algorithms. Specifically, they offer users a set of basic graph processing operations (BGOs), an API to apply them, and one or several back-ends to match or different systems. Users design graph processing applications as a workflow of BGOs, analysing and/or transforming the input graph towards the required result. Several surveys and analyses of such platforms [6, 8, 9, 16] speak of their advantages in terms of programmability and portability, and highlight the limited overhead such platforms might have.

Neither hand-tuned nor GPPs/GPSs provide accurate performance and/or energy consumption guarantees for the workload being processed. For hand-tuned algorithms, benchmarking remains the norm, while GPPs/GPSs provide no performance models for their APIs and/or implementations. Graph-Optimizer will provide such models for basic primitives, and build complex workload models through composition.

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3585340

Data modeling. The performance of graph processing workloads depends significantly on graph properties [8, 9, 18]. Capturing that dependency in analytical models remains challenging, and difficult to include in workloads models. Current work focuses on either machine-learning approaches, where the workload is considered a mix of the algorithm and input data [18], or on observing and using correlation between specific properties (i.e., vertex degree distribution, diameter, etc.) and performance profiles of specific algorithms. Graph-Optimizer will use a combination of the two for the BGO models. The composition of these models will also need to take partitioning models into account to determine the communication volume, but will not depend on graph properties.

Hardware models The complexity of heterogeneous systems combining parallel CPUs and accelerators is well documented [14]. Graph processing workloads are notoriously difficult for such parallel hardware [13]. Graph-Optimizer will focus on accurate models for BGOs, combining microbenchmarking with analytical and statistical models [4, 17], and accurate models of data transfer infrastructure to scale from BGOs to real workloads.

3 GRAPH-OPTIMIZER DESIGN AND VALIDATION

Figure 1 presents the high-level architecture of Graph-Optimizer. The tool enables users to express graph processing as a workflow of basic graph operations (BGOs), and, using hardware models for both processing units and communication infrastructure, can assess the performance (e.g., in terms of execution time or energy) of the given workload on a specific hardware configuration. Through a design-space exploration procedure, Graph-Optimizer can select the most suitable node-level system for the problem at hand.

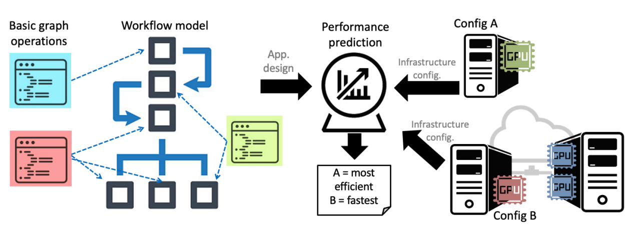

Figure 1: A high-level architecture of Graph-Optimizer

We will validate Graph-Optimizer using several configurations of heterogeneous systems, using AMD and Intel CPUs, as well as accelerators such as NVIDIA and AMD GPUs.

Our first validation case-study is a synthetic combination of different BGOs, including: graph sampling and up-scaling, graph traversal, centrality calculation, and, optionally, community detection or clustering based on the centrality scores. Concretely, we will use existing implementations of the selected BGOs to provide a first approximation of the workload. Next, we model the BGOs and calibrate these models on the different hardware devices through microbenchmarking. Finally, we add the infrastructure model (as the system "executing" the data communication and/or dependencies) and compose the overall application model. In turn, this model is used to predict the performance of graph workloads on different hardware configurations and input data.

The validation is successfil when Graph-Optimizer provides accurate predictions. Specifically. the framework must *guarntee* a performance lower bound, provide correct ranking of the different configurations (thus enabling design-space exploration), and should not be ore than 25% off from the measured performance.

4 SUMMARY AND OUTLOOK

The first step in developing Graph-Optimizer is top provide optimized BGOs and their performance models. Next, we will focus on data partitioning and model composition rules to support larger workloads, and validate Graph-Optimizer on different synthetic mixes of BGOs. Finally, we will enable the use of Graph-Optimizer for design-space exploration and to drive the Graph-Greenifier tool towards scaling to massive data and large-scale systems.

ACKNOWLEDGMENTS

This invited talk is co-funded by EU H2020 Graph-Massivizer, under Grant Agreement N° 101093202.

REFERENCES

[1] Ahmed et al. 2011. Network sampling via edge-based node selection with graph induction. (2011).
[2] Scott Beamer, Krste Asanovic, and David Patterson. 2012. Direction-optimizing Breadth-First Search. In *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–10. https://doi.org/10.1109/SC.2012.50
[3] Scott Beamer, Krste Asanovic, and David A. Patterson. 2015. The GAP Benchmark Suite. *CoRR* abs/1508.03619 (2015). arXiv:1508.03619 http://arxiv.org/abs/1508.03619
[4] Nicola Bombieri, Federico Busato, et al. 2016. MIPP: A microbenchmark suite for performance, power, and energy consumption characterization of GPU architectures. *2016 11th IEEE Symposium on Industrial Embedded Systems (SIES)* (2016), 1–6.
[5] The Graph 500 Steering Committee. 2010–2016. The Graph 500 List. http://www.graph500.org.
[6] Niels Doekemeijer. 2014. A Survey of Parallel Graph Processing Frameworks. https://www.tudelft.nl/ewi/over-de-faculteit/afdelingen/software-technology/distributed-systems/research-1/publications/technical-reports. Online; accessed 26 Feb 2023.
[7] Graphalytics.org. 2019. Graphalytics Global Competition. https://graphalytics.org/competition.
[8] Guo et al. 2014. Benchmarking Graph-processing Platforms: A Vision. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*.
[9] Iosup et al. 2016. LDBC Graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. *Proceedings of the VLDB Endowment* 13 (2016).
[10] Khorasani et al. 2015. Scalable SIMD-Efficient Graph Processing on GPUs. In *Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques (PACT '15)*.
[11] Leskovec. 2006. Stanford Network Analysis Platform (SNAP). *Stanford University* (2006).
[12] Leskovec. 2008. Kronecker Graphs (presentation). (2008).
[13] Andrew Lumsdaine, Douglas P. Gregor, Bruce Hendrickson, and Jonathan W. Berry. 2007. Challenges in Parallel Graph Processing. *Parallel Process. Lett.* 17 (2007), 5–20.
[14] Sparsh Mittal and Jeffrey S. Vetter. 2015. A Survey of CPU-GPU Heterogeneous Computing Techniques. *ACM Computing Surveys (CSUR)* 47 (2015), 1 – 35.
[15] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, et al. 2021. The Future is Big Graphs: A Community View on Graph Processing Systems. *Commun. ACM* 64, 9 (aug 2021), 62–71. https://doi.org/10.1145/3434642
[16] Xuanhua Shi, Zhigao Zheng, et al. 2018. Graph Processing on GPUs: A Survey. *ACM Comput. Surv.* 50, 6, Article 81 (jan 2018).
[17] Rico van Stigt, Stephen Nicholas Swatman, and Ana-Lucia Varbanescu. 2022. Isolating GPU Architectural Features Using Parallelism-Aware Microbenchmarks. In *ACM/SPEC ICPE'22*.
[18] Verstraaten et al. 2018. Mix-and-Match: A Model-driven Runtime Optimisation Strategy for BFS on GPUs. In *2018 IEEE/ACM 8th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*.
[19] Carl Yang, Aydın Buluç, and John D. Owens. 2018. Implementing Push-Pull Efficiently in GraphBLAS. In *ICPP'18*. Article 89, 11 pages.