PROCESSING TRILLION EDGE GRAPHS IN DISTRIBUTED MEMORY

George M. Slota, The Pennsylvania State University
2014-2015 Graduate Fellow

RESEARCH SUMMARY

Graphs are a mathematical construct representing discrete entities or objects (vertices) and some form of link or interaction between them (edges). Human social interactions, the Internet, and the neural structures of the brain are just a subset of modern, real-world and extreme-scale datasets that are representable as graphs. The irregularity, scale, and complexity of such graphs present a high level of challenge for domain experts to use computational resources to extract useful insight into these graphs. As a result, many graph processing frameworks have recently been introduced with the goal to simplify the analysis of real-world graphs on commodity hardware. However, these popular frameworks lack scalability to modern massive-scale datasets, require specialized hardware to run, or simply cannot outperform optimal serial code [1]. Our previous work focused on overcoming these barriers from the ground up, developing general approaches for graph analytic optimization that are highly performant from a single node to a small cluster to a large and powerful system such as Blue Waters. There has been other prior research aimed at abstracting graph algorithms themselves, including as linear algebraic operations [2] or into a nested loop structure [3]. As such, our ongoing work has identified several other key data storage and communication abstractions that allow the straightforward implementation of broad classes of graph analytics. These findings will enable domain scientists to study graphs at a larger scale and with more complex algorithms than has previously been possible.

Our current work considers graph algorithms running on distributed systems such as Blue Waters and processing on an in-memory distributed graph representation. Here, each MPI task owns only some subset of a large graph, iteratively performing computation on its subset and communicating the results to neighboring tasks. By analyzing the computation and communication patterns typical of a number of graph algorithms, we noticed several generalizable similarities. There are two primary ways in which algorithm-specific data is updated. Either these updates are pulled by a given vertex to update some data associated with this vertex or these updates are pushed by a given vertex for its neighbors to use while computing updates to their data. In addition to this push/pull pattern, there is also a difference in the sizes of the update and work sets, where these sets can be composed of vertices for each iteration. The sets are either variable or fixed, depending on the algorithm. Overall, we identified four distinct processing patterns into which many graph algorithms fall (push-variable, push-fixed, pull-variable, pull-fixed) and created optimized outlines for these patterns. Using these patterns, we then implemented several common graph analytic algorithms. We observe high performance of our implementations, despite the generalized approach. Our implementations are noted to scale to graphs of over a trillion edges and over ten billion vertices while running on up to 8,192 nodes and 131,000 cores of Blue Waters. We can end-to-end process graphs of this scale in mere minutes on Blue Waters, including I/O, preprocessing, and output. Ongoing work aims to simplify our approach and methods further to enable its use among domain experts and graph analysts.

George Slota accepted a temporary staff position at Sandia National Labs upon graduation in May 2016. In Fall 2016 he started as an assistant professor in the Computer Science Department at Rensselaer Polytechnic Institute.

"The Blue Waters program has helped me achieve my career goals," he says, "by allowing me to focus on my research during my Ph.D. as well as enabling my access to a large-scale system that my research required."

WHY BLUE WATERS

Access to the Blue Waters systems greatly benefited this research for several reasons. The high performance I/O filesystem greatly accelerated end-to-end processing times for experiments on our large test datasets. We often observed over 50GB/s read time across a moderate number of nodes, even during periods of high concurrent usage, which enabled test data of terabytes in size to be read in seconds. This minimized node-hour waste when running high numbers of parametric tests with quick turnaround times. Blue Waters also represents the state-of-the-art in overall intra-node and inter-node communication and computational performance, which allows it to serve as a good representable testbed for generalizing our methods to be run on other current and future systems.

PUBLICATIONS AND DATA SETS

Slota, G. M., S. Rajamanickam, and K. Madduri, A Case Study of Complex Graph Analysis in Distributed Memeory: Implementation and Optimization. *Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium* (IPDPS16), (IEEE, Chicago, Illinois, May 23-27, 2016).

Slota, G. M., and K. Madduri, Parallel Color-coding. *Parallel Computing*, 47 (2015), pp. 51-69.

Slota, G. M., S. Rajamanickam, and K. Madduri, High Performance Graph Analytics on Manycore Processors. *Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium* (IPDPS15), (IEEE, Hyderabad, India, May 25-29, 2015).