Achieving High Performance for Big Data Analytics

Oded Green
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
ogreen@gatech.edu
PhD. Advisor: Prof. David A. Bader

ABSTRACT

Irregular algorithms such as graph algorithms, sorting, and sparse matrix multiplication, present numerous programming challenges that include scalability, load balancing, and efficient memory utilization. In this age of Big Data we face additional challenges since the data is often streaming at a high velocity and we wish to make near real-time decisions for real-world events. For instance, we may wish to track Twitter for the pandemic spread of a virus. Analyzing such data sets requires combing algorithmic optimizations and utilization of massively multithreaded architectures, accelerator such as GPUs, and distributed systems. My research focuses upon designing new analytics and algorithms for continuous monitoring of dynamic social networks. Specifically, we deal with load balancing, scheduling, avoiding redundant computations, and utilizing network properties for designing dynamic graph algorithms.

Keywords

Graph algorithms; Social Network Analysis; Parallel algorithms; High Performance Computing; Dynamic Data;

Research Statement and Motivation

Achieving high performance computing for irregular algorithms such as Social Network Analysis (SNA) is challenging as the instruction flow is data dependent and requires domain expertise. The rapid changes in the underlying network necessitates understanding real-world graph properties such as the small world property [23, 25], shrinking network diameter [20], power law distribution of edges[4, 7], and the rate at which updates occur. These properties, with respect to a given analytic, can help design load-balancing techniques, avoid wasteful (redundant) computations, and create dynamic algorithms.

In the course of my research I have considered several parallel programming paradigms for a wide range systems of multithreaded platforms: x86 [10, 11, 9, 16, 13], NVIDIA's CUDA [15], Cray XMT2 [9], SSE-SIMD, Plurality's HyperCore [5, 13, 24, 12]. These unique programming models have required examination of parallel programming at multiple levels: algorithmic design, cache efficiency, fine-grain parallelism, memory bandwidths, data management, load balancing, scheduling, control flow models and more. In my poster I will present the approaches that we have taken to analyze massive social networks. I will focus on the three of my novel algorithmic contributions for dynamic data: betweenness centrality, clustering coefficients, and Merge Path.

Betweenness Centrality (BC) [8] is a widely used analytic used for finding key players in a social network. In [16] we designed and implemented a new algorithm for computing BC for dynamic graphs that extends Brandes's [3] static graph algorithm. Our dynamic algorithm is several hundred times faster that the static algorithm for edge insertion and deletion. The new dynamic graph algorithm had an increased storage complexity that required us to revisit the data structures required by Brandes's algorithm. This in turn led to the development of a new approach for computing betweenness centrality, that can be applied to multiple flavors of betweenness centrality, that increase scalability, reduce memory footprint, improve cache usage, increase problem size, and offer better loading balancing [9]. Our algorithm also proved to be twice as fast (per core) then previously published algorithm for x86 systems. We implemented several BC algorithms on two shared-memory systems including a 40-core Intel x86 system and the Cray XMT2 [19]. Our new approach can be applied to additional betweeness centrality algorithms: parallel[2, 21] and distributed [6], and approximate [1].

Clustering coefficients is another widely analytic used to state how tightly bound vertices are based on the number of closed triangles that they belong. In [10] we present a new approach to compute clustering coefficient using vertex covers. This approach avoids counting the same triangle multiple times and can be added to the well known lexicographical approach that reduces the times a triangle is counted by a factor of two. We showed that our algorithm can be extended to count larger circuits as well. From this optimization, we were able to better understand the load imbalance caused by straightforward parallelization of clustering coefficient that is due to power-law distribution of the edges. In [17] we presented two unique load balancing techniques for clustering coefficients - these approaches trade off accuracy of the load-balance with storage requirements. Our new algorithms are several times faster than previous implementations. Further, our two optimizations can be combined, thus achieving even higher speedups.

Additional social network analytic we developed is the combination of a data structure and algorithm for tracking connected components in a dynamic graph [22]. This algo-

.

rithm takes into consideration the small-world property and shrinking diameter properties and shows that it is possible (and most likely beneficial) to track the connected components that the vertices belong to using $O(1)$ memory for each vertex. Our algorithm can easily keep up with the fastest update rates that current social networks produce on a shared-memory system.

In [24, 18] we show a visual and intuitive approach for parallel merging two sorted arrays called Merge Path. In [15] we extend the Merge Path concecpt to the GPU and create the first of a kind GPU merging algorithm. A brief summary of our merging results: $32X - 35X$ speedup on a 40 core system, first of its kind cache-aware merge, and a $50X$ speedup on NVIDIA's Fermi GPU over an Intel core. In fact, merging and adjacency list intersection (which is a key building of clustering coefficients) are similar operations. Thus, Merge Path can be extended to adjacency list intersection if needed.

In [5, 13] we create new algorithm computing the 2D estimated covariance matrix which avoids redundant multiplications without increasing memory requirements or communication costs. We then extended this algorithm such that it avoids redundant additions as well [11] allowing for a sequential algorithm that is over $40X$ faster than the straightforward implementation. Our new algorithm is highly scalable due to low storage requirements and no communication costs.

1. REFERENCES

[1] D. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin / Heidelberg, 2007.

[2] D. Bader and K. Madduri. Parallel algorithms for evaluating centrality indices in real-world networks. In *International Conference on Parallel Processing (ICPP)*, pages 539 –550, Aug. 2006.

[3] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[4] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33:309 – 320, 2000.

[5] L. David, A. Galperin, O. Green, and Y. Birk. Efficient Parallel Computation of the Estimated Covariance Matrix. In *IEEE 26th Convention of Electrical and Electronics Engineers in Israel*, 2010.

[6] N. Edmonds, T. Hoefler, and A. Lumsdaine. A space-efficient parallel algorithm for computing betweenness centrality in distributed memory. In *International Conference on High Performance Computing (HiPC), 2010*, pages 1 –10, Dec. 2010.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of The Internet Topology. In *ACM SIGCOMM Computer Communication Review*, pages 251–262. ACM, 1999.

[8] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):pp. 35–41, 1977.

[9] O. Green and D. A. Bader. Faster Betweenness Centrality Based on Data Structure Experimentation. In *International Conference on Computational Science (ICCS)*. Elsevier, 2013.

[10] O. Green and D. A. Bader. Faster Clustering Coefficients Using Vetex Covers. In *Proceedings of the 5th ASE/IEEE International Conference on Social Computing*, SocialCom '13, 2013.

[11] O. Green and Y. Birk. A Computationally Efficient Algorithm for the 2D Covariance Method. In *ACM/IEEE Conference on Supercomputing*, 2013.

[12] O. Green and Y. Birk. Scheduling Directives for Shared-Memory Many-Core Processor Systems. In *Proceedings of the 2013 International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 115–124. ACM, 2013.

[13] O. Green, L. David, A. Galperin, and Y. Birk. Efficient Parallel Computation of the Estimated Covariance Matrix. In *arXiv*, 2013.

[14] O. Green, R. McColl, and D. Bader. GPU Merge Path: a GPU Merging Algorithm. In *Proceedings of the 26th ACM International Conference on Supercomputing*, pages 331–340. ACM, 2012.

[15] O. Green, R. McColl, and D. A. Bader. A Fast Algorithm For Streaming Betweenness Centrality. In *Proceedings of the 4th ASE/IEEE International Conference on Social Computing*, SocialCom '12, 2012.

[16] O. Green, L. M. Munguia, and D. A. Bader. Load Balanced Clustering Coefficients. GT Tech Report, Oct. 2013.

[17] O. Green, S. Odeh, and Y. Birk. Merge Path - a Visually Intuitive Approach to Parallel Merging.

[18] P. Konecny. Introducing the Cray XMT. Seattle, WA, USA, May 2007.

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs Over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Data Mining*, pages 177–187. ACM, 2005.

[20] K. Madduri, D. Ediger, K. Jiang, D. Bader, and D. Chavarria-Miranda. A Faster Parallel Algorithm and Efficient Multithreaded Implementations for Evaluating Betweenness Centrality on Massive Datasets. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2009.

[21] R. McColl, O. Green, and D. Bader. A New Parallel Algorithm for Connected Components in Dynamic Graphs. In *IEEE International Conference on High Performance Computing*, Dec. 2013.

[22] S. Milgram. The Small World Problem. *Psychology Today*, 2(1):60–67, 1967.

[23] S. Odeh, O. Green, Z. Mwassi, O. Shmueli, and Y. Birk. Merge Path - Parallel Merging Made Simple. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1611–1618. IEEE, 2012.

[24] S. H. Watts, Duncan J.Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393, 1998.