LFGraph: Simpler is Better for Distributed Graph Analytics

Imranul Hoque and Indranil Gupta*
University of Illinois, Urbana-Champaign
{ihoque2, indy}@illinois.edu

Cloud computing frameworks today are being used to process extremely large graphs with billions of vertices and hundreds of billions of edges in a distributed manner. Some examples are: Web graph, citation graphs, product co-purchase graphs, online social networks, financial networks, etc. Graph processing frameworks (e.g., [1, 2]) need to produce fast results while also maintaining a small memory footprint.

Concretely, depending on how servers share the graph data for analytics, a distributed graph processing framework can be classified as either message-passing or shared memory-based. Due to the nature of graph processing, most of these frameworks are iterative, i.e., they operate in steps. However, today's frameworks suffer from three overheads: 1) *intelligent graph partitioning* consumes significant up-front time, 2) *push-based message passing* approaches lead to large queues and incur significant communication overhead, and 3) *memory usage* is high because of local data stored at each server.

We elaborate briefly on these. Intelligent graph partitioning can reduce the time per iteration, but often constitute a major portion of the total run-time. For instance, when running PageRank using PowerGraph [1] on 8 servers, with 30 iterations (a number that Google uses [2]), the intelligent partitioning constitutes 66% of the total run-time. Without intelligent partitioning, each of the iterations takes longer. Message-passing graph engines like Pregel [2] i) queue messages from the previous iterations, which leads to queues growing rapidly and ultimately limits the number of vertices per server; and ii) push data along graph edges, which causes duplicate network traffic. Finally, shared memory-based graph processing frameworks such as GraphLab [1] use a Gather-Apply-Scatter model – they store both in- and out-links, which increases memory requirement by 2x for directed graphs.

Our system LFGraph (Laissez-Faire Graph processing engine) addresses the above shortcomings. LFGraph is an efficient, distributed in-memory graph analytics framework. Its key features include:

- LFGraph uses a *pull-based strategy* to receive data from immediate neighbors stored at a different server, thus eliminating the need to store the queued messages. This increases the number of vertices processed at a single server.
- LFGraph *reduces communication overhead* by exploiting the fact that many vertices assigned to a single server share their one-hop neighbors. Thus, instead of pulling a shared neighbor's data multiple times, LFGraph pulls it only once. This results in up to 3x reduction in network traffic compared PowerGraph. LFGraph batches data transfers.
- LFGraph *reduces memory footprint* by storing only in-edges of vertices. This causes over 2x reduction in memory footprint compared to GraphLab and PowerGraph.
- LFGraph *eliminates the need to intelligently partition* the graph. This means that a random partitioning suffices in LFGraph to retain a factor of 3 performance gain over existing mentioned systems.

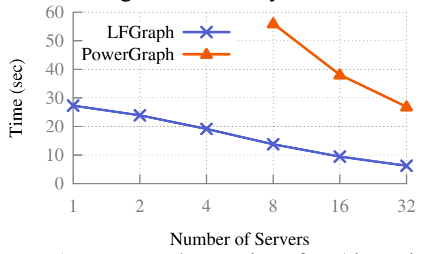

Figure 1: PageRank run-time for 5 iterations

We evaluated our high-performance C++ implementation of LFGraph by running PageRank for 5 iterations. We used the Twitter graph containing 41M vertices and 1.4B edges as the data-set. The experiments were conducted on a 32-server Emulab cluster. Each server consists of one quad-core Intel Xeon E5530 processor with 12 GB of RAM and is connected via 1 GigE. We compare our performance against PowerGraph, known to outperform existing graph processing frameworks. We use random partitioning for PowerGraph. We present the results in Figure 1. PowerGraph could not process the Twitter graph using fewer than 4 servers due to the memory requirement. We observed that LFGraph reduces PageRank runtime by a factor of 3 compared to PowerGraph.

References

[1] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In *OSDI 2012*, pages 17–30.

[2] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A System for Large-Scale Graph Processing. In *SIGMOD 2010*, pages 135–146.

*This research was supported in part by NSF grant CCF-0964471.