ScaleG: A Distributed Disk-based System for Vertex-centric Graph Processing (Extended Abstract)

Xubo Wang[†], Dong Wen[§], Lu Qin[♮], Lijun Chang[¶], Wenjie Zhang[§]

[†]*Zhejiang Lab, China;* [§]*University of New South Wales, Australia*
[♮]*AAII, University of Technology Sydney, Australia;* [¶]*The University of Sydney, Australia;*
[†]wangxb@zhejianglab.com; [§]{dong.wen, wenjie.zhang}@unsw.edu.au;
[♮]lu.qin@uts.edu.au; [¶]lijun.chang@sydney.edu.au

***Abstract*—Designing disk-based distributed graph systems has drawn a lot of research due to the strong expressiveness of the graph model and rapidly increasing graph volume. However, several challenges still exist in achieving both high computational efficiency and low network communication under the limitation of memory. In this paper, we design a novel distributed disk-based graph processing system, ScaleG, with a series of user-friendly programming interfaces. We propose several techniques to reduce both disk I/Os in each machine and message I/Os via the network. We manage all messages in memory and bound the volume of all messages by the number of vertices. We also carefully design the data structure to support partial computation and automatic vertex activation. We conduct extensive experiments on real-world big graphs to show the high efficiency of our system.**

I. Introduction

Efficiently processing graph data is essential in both research and practice. Numerous research interests have been shown on designing distributed graph systems to process big graphs. However, most distributed vertex-centric graph systems store all data in main memory of machines, which brings high efficiency but sacrifices scalability given the dramatic increasing data volume. Due to some intermediate results, messages, replicated vertices and edges, the memory usage can be much larger than the input graph size. Several disk-based (or called out-of-core) distributed graph systems have been studied in the literature. Among them, GraphD [2] is the state-of-the-art and follows the same programming model as Pregel [1]. GraphD adopts the semi-streaming model and only allows the vertex states resided in main memory of each machine. The adjacency lists and messages are managed as edge streams and message streams on disks, respectively.

There are still several challenges in GraphD. First, under the memory usage limitation, GraphD saves all sending messages and receiving messages on disks. Scanning and managing the message streams on disks incurs a great deal of disk I/Os. Second, several studies have shown that the volume of communication messages can be very large given the power-law degree distributions of real-world graphs. To handle such issue, the pull-based computing model is studied in the literature [3]. However, disk I/O is not considered in these in-memory systems. More importantly, the pull-based method requires an extra pull request to notify the corresponding neighbor. In addition, based on the push model of Pregel, GraphD cannot efficiently handle the partial computation in many fundamental algorithms. For example, in the algorithm of distributed graph coloring, only a small number of vertices are active and wait to be colored in many iterations. To color a vertex u, we need colors of all neighbors of u. Since each vertex can only passively receive messages in Pregel-like systems, all vertices have to send messages to their neighbors. Therefore, the main challenges in this paper are how to effectively reduce the communication messages and how to efficiently manage the messages under the limitation of memory usage.

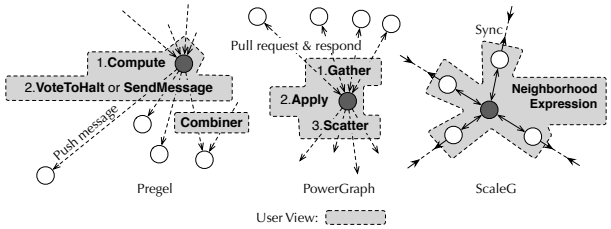

Fig. 1. Representative system models

In this paper, we make the following two contributions in designing a novel distributed disk-based system ScaleG for large graph processing in response to the above challenges. (1) Optimizations to reduce communication cost. We adopt a compute-and-sync model (Fig. 1) in disk-based graph processing system design. ScaleG provides $O(n)$ message bounds for each machine where n represents the number of vertices. This allows ScaleG to keep messages in memory and offer high communication efficiency. (2) I/O efficient computation with limited memory usage. Based on our computing model, we design a series of data structures to activate and compute all necessary vertices in one round of disk scan with no inter-machine communication cost. Memory usage is bounded by $O(n)$. In addition, ScaleG supports partial computation where only a partial set of vertices are activated. Our implementation avoids unnecessary message transmission and achieves high I/O efficiency. We conduct extensive experiments compared with several representative competitors to show the performance of ScaleG.

II. Summary of Our System

In this section, we present the main components of our distributed disk-based graph system ScaleG. Please refer to our full paper [4] for more details.

Execution model. We observe that vertices in many vertex-centric algorithms only communicate with neighbors. We

enforce such property in the system and adopt a compute-and-sync programming model. The compute phase performs the logic provided by the user, while the sync phase synchronizes the vertex states in different machines and is hidden from users. An illustration of our model is given in Fig. 1, with Pregel [1] and PowerGraph [3] as comparisons. Thanks to our computing model, all neighbors' states are locally provided in the compute phase. The programmer only needs to care about how to update the vertex based on neighbors' states and does not need to concern any logic regarding message sending, receiving or combining. The simple API of ScaleG and different algorithm implementation examples can be found in [4].

Implementation. We carefully design special data structures and mechanisms to realize our model in a disk-based graph processing system.

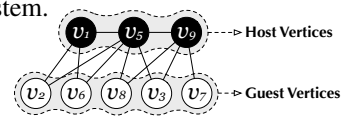

Fig. 2. An example of the graph structure in one machine

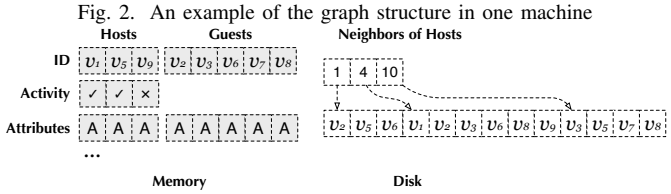

Fig. 3. The data structure for compute phase in machine 1

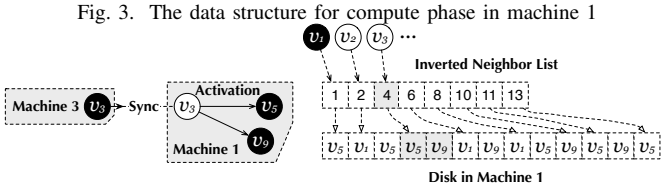

Fig. 4. The data structure for the sync phase in machine 1

In ScaleG, a given graph is partitioned to different machines in a cluster. We name the vertices assigned to a machine as host vertices and the neighbors of boundary $host$ vertices in a worker as $guest$ vertices. Similar to GraphD, ScaleG adopts the semi-external setting and a compressed sparse row (CSR) structure to maintain the states of host vertices in memory and store the neighbors of each vertex on the disk. Unlike GraphD, ScaleG additionally maintains the guest vertices in the memory of each machine to support the locally neighborhood expression for host vertices. The rationale of in-memory message management in ScaleG is supported by our computing model, which bounds the number of messages by the number of vertices in each machine. In each iteration of ScaleG, the memory usage of an arbitrary machine $\mathcal{W}$ is bounded by $O(k{\cdot}V(\mathcal{W}){+}N(\mathcal{W}))$ where k, $V(\mathcal{W})$ and $N(\mathcal{W})$ represent the number of machines, hosts and guests in $\mathcal{W}$ respectively. Fig. 2 gives an example of a subgraph partitioned to one machine in ScaleG. The corresponding data structure is given in Fig. 3.

In each iteration, we only sequentially read or skip items in the neighbor list file from disk. The total disk I/O of all machines in each iteration is bounded by $O(m/B)$, where m and B represent the number of edges in the data graph and the block size for a single disk read/write operation respectively.

In addition, we propose detailed external-memory data structure for vertex activation. The data structure for the sync phase corresponding to Fig. 2 is given in Fig. 4. Unlike GraphD, the messages with the same value would never be sent repeatedly in all studied algorithms in ScaleG. For example, in the implementation of graph coloring in ScaleG, colors of all neighbors can be locally accessed by each vertex, and in following iterations, only the changed colors will lead to an update message. In each iteration of ScaleG, the communication cost of all machines is bounded by $O(\min(n \cdot k, m))$ where n is the number of vertices. We also design an adaptive activation mechanism to further improve the system efficiency.

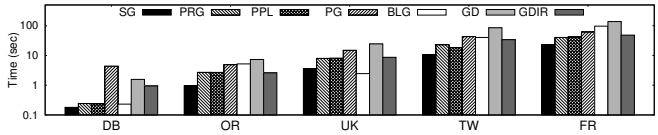

Fig. 5. Comparison with Existing Systems (Running Time)

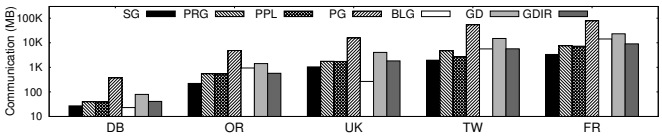

Fig. 6. Comparison with Existing Systems (Communication Cost)

III. Evaluation

We compare ScaleG with the disk-based system GraphD and also representative in-memory graph processing systems running. In our experiments, ScaleG exhibits the best overall performance over nine fundamental and various graph algorithms on six large real-world graphs. Fig. 5 and Fig. 6 show the running time and communication cost of compared systems running breadth first search. ScaleG outperforms GraphD because of no message disk I/O and less communication cost. ScaleG also outperforms in-memory systems because ScaleG saves communication cost from the message sending incurred by high-degree vertices in push-based methods and extra pull requests in pull-based methods. The extensive experimental results validate the superb efficiency of ScaleG on large graphs. Please refer to our full paper [4] for detailed experimental settings and more experimental results.

Acknowledgment

Xubo Wang is supported by China Postdoctoral Science Foundation (2021M692958). Lu Qin is supported by ARC FT200100787. Lijun Chang is supported by ARC FT180100256. Ying Zhang is supported by ARC DP180103096 and FT170100128. Wenjie Zhang is supported by ARC DP180103096 and ARC DP200101116.

References

[1] 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 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 135–146.

[2] D. Yan, Y. Huang, M. Liu, H. Chen, J. Cheng, H.Wu, and C. Zhang, "Graphd: distributed vertex-centric graph processing beyond the memory limit," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 1, pp. 99–114, 2018.

[3] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012, pp. 17–30.

[4] X. Wang, D. Wen, L. Qin, L. Chang, Y. Zhang and W. Zhang, "ScaleG: A Distributed Disk-based System for Vertex-centric Graph Processing," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2021.3101057.