GRAPH/Z: A Key-Value Store Based Scalable Graph Processing System

Tonglin Li[1], Chaoqi Ma[1], Jiabao Li[1], Xiaobing Zhou[3],
Ke Wang[1], Dongfang Zhao[1], Iman Sadooghi[1], Ioan Raicu[1,2]
[1]*Illinois Institute of Technology,* [2]*Argonne National Laboratory,* [3]*Hortonworks*

***Abstract*—The emerging applications in big data and social networks issue rapidly increasing demands on graph processing. Graph query operations that involve a large number of vertices and edges can be tremendously slow on traditional databases. The state-of-the-art graph processing systems and databases usually adopt master/slave architecture that potentially impairs their scalability. This work describes the design and implementation of a new graph processing system based on Bulk Synchronous Parallel model. Our system is built on top of ZHT, a scalable distributed key-value store, which benefits the graph processing in terms of scalability, performance and persistency. The experiment results imply excellent scalability.**

I. Introduction

With the advancement of social networks, online gaming [1–3] and scientific applications such as geospatial systems [4–6] and bioinformatics [7–12], graph data has been used ubiquitously. There have been works on work flow systems[13–17] and data streaming management systems[18–20] attempted to handle structured big data sets from scientific and commercial applications, which are typically stored in file systems(such as Hadoop HDFS[21] and FusionFS[22, 23]), SQL databases(such as Oracle and DB2) or Column Family databases (such as Hadoop Hive and Cassandra). Data mining, machine learning [24–26] and security management [27–29] techniques are also widely used to extract the value from these big data sets. However it is not easy to fully reveal and utilize the scientific and commercial value from the continuously increasing graph data sets. It's even more challenging when moving these works to clouds[30]. The traditional relational database has been used and dominated for many years, and it also works well for a long time. Graph related query is tremendously slow on the traditional relational database. An ideal solution for this problem is to replace the traditional data infrastructure with a graph-centric model, including storage and computing, thus to better serve graph-based applications in terms of performance and programmability.

Pregel [31] is a Bulk Synchronous Parallel model based distributed graph processing system developed by Google. It inspires couple of similar variation projects, such as Giraph[32] and GraphLab[33], now known as Pregel-like systems. However Pregel-like systems have some limitations. First, they only work on in-memory data and don't accept new data as soon as data loading is finished. This limits their use especially when the dataset can't fit in memory. Second, the master node coordinates both synchronization barriers and checkpointing for fault tolerance, which makes it a significant bottleneck.

We design and implement a new graph processing system Graph/Z with ZHT [34–39] as a building block. Graph/Z can be considered as another Pregel-liked graph processing system, but it inherits some important features from ZHT, a distributed key-value store, which differentiate Graph/Z from other systems. ZHT is a zero-hop distributed key-value store featured with high scalability, persistency and fault tolerance. By leveraging ZHT's persistency, Graph/Z can run with a much larger working dataset.

The contributions of this paper are as follows:

- Design and implementation of GRAPH/Z, a BSP model graph processing system on top of ZHT.
- The system utilizes data-locality and minimize data movement between nodes.
- Benchmarks up to 16-nodes scales.

II. GRAPH/Z Design and Implementation

GRAPH/Z follows Pregel's computing paradigm, which has been known as ”think like a vertex”: graph computation jobs are divided in terms of what each vertex needs to compute; edges are communication paths for transmitting results from one vertex to another. The computation is split into supersteps. At each superstep, a vertex executes a computation task, send or receive messages to its neighbors. Supersteps end with synchronization barriers.

We use ZHT as a building block in our Graph processing system so as to utilize it's feature and performance advantages. In Graph/Z, ZHT is used to support random access, remove, update and create vertices and edges. We also use the key-value entry in ZHT to handle the communication messages between nodes.

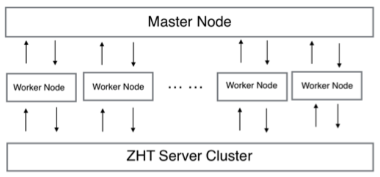
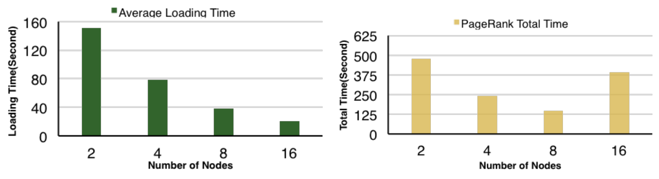


Fig. 1: Graph/Z architecture

GRAPH/Z consists of three major components, namely master node, worker node and storage server. A worker node is the a computation unit. It firstly loads assigned data partitions (to present vertices) from ZHT server, then it executes computation jobs depend on specific algorithms. In the BSP model, each vertex needs to communicate with others. If two vertexes are not located on the same node, we need to handle the communication between these two nodes. We use some special key-value entries in ZHT as message transfer station, so each worker node don't need to communicate with each directly, they only need to communicate with ZHT. As we discussed, the processes of BSP model consists of multiple supersteps, since we cannot achieve a perfect load-balance, some workers may be still working on their own vertexes while other workers have already finished their job. The main job of master node is to coordinate the supersteps, only when all



(a) Loading time (b) PageRank performance

Fig. 2: Graph/Z performance

worker nodes complete their current superstep master node will notify them to do the next superstep.

Fault Tolerance: We store all the message list, active vertex list and any information that we need in graph computation in the ZHT server. This means that all the intermediate and final results are stored in ZHT. These data can also be used as checkpoints in each superstep. If one node is failed, we can simply read these data from ZHT and restart this superstep without having to restart over. This is especially useful when processing big data sets.

III. Evaluation

We run PageRank on setups of 2, 4, 8, and 16 virtual machines. All VMs are m3.2large Amazon EC2 spot instances, located in us-west-2c. Each instance has four virtual CPUs, equivalent to 2.5 GHz Xeon Family processors, and 30GB of memory. The data set is web-Google from SNAP (Stanford Network Analysis Project), which contains 1 million vertexes and 5 million edges. We evaluated the loading time by starting all the worker nodes and calculate the average loading time in different scales(form 2 to 16). The increment of loading time is almost linear because each worker node only need to load its local vertexes and don't need to communicate with a remote node. Due to the load balance feature of ZHT, the amount of work that each worker needs to do are basically equal.

PageRank algorithm uses all the vertexes and edges in every superstep. Thus this is a good algorithm to test data locality and load balance. When running on 8 nodes, the system achieves the highest performance, and then it decreases greatly on 16 nodes. This is mainly because the average work load on each node is too small and relatively more cross-node communication is involved due to large scale.

IV. Related Work

Pregel[31] is the first influential BSP-based implementation that provides a native API specifically for programming graph algorithms while abstracting away the underlying communication details. Giraph [32] is an Apache project and a variation of Pregel, which leverages Apache Hadoop's MapReduce to handle graphs. It is used to perform graph processing on big data. Facebook used Apache Giraph to analyze one trillion edges within only 4 minutes by 200 machines. GraphLab [33] is a graph-based, high performance, distributed computation framework which was written in C++. GraphLab extends Pregel protocols and supports both synchronous and asynchronous modes.

V. Conclusion

In this project, we present the preliminary results of Graph/Z, a scalable distributed graph processing system. By adopting a distributed key-value store, we simplify the design and implementation of Graph/Z and reduce the workload on master node.

References

[1] Z. Lv, A. Halawani, S. Feng, H. Li, and S. Ur Réhman, "Multimodal hand and foot gesture interaction for handheld devices," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2014.

[2] Z. Lv, A. Tek, F. Da Silva, C. Empereur-Mot, M. Chavent, and M. Baaden, "Game on, science-how video game technology may help biologists tackle visualization challenges," *PloS one*, 2013.

[3] Z. Lv, "Wearable smartphone: Wearable hybrid framework for hand and foot gesture interaction on smartphone," in *2013 IEEE International Conference on Computer Vision Workshops*.

[4] T. Su, Z. Lv, S. Gao, X. Li, and H. Lv, "3d seabed: 3d modeling and visualization platform for the seabed," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference*.

[5] X. Li, Z. Lv, J. Hu, B. Zhang, L. Yin, C. Zhong, W. Wang, and S. Feng, "Traffic management and forecasting system based on 3d gis," in *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, IEEE, 2015.

[6] Z. Lv and T. Su, "3d seabed modeling and visualization on ubiquitous context," in *SIGGRAPH Asia 2014*, ACM.

[7] J. J.-Y. Wang, H. Bensmail, and X. Gao, "Multiple graph regularized protein domain ranking," *BMC bioinformatics*, 2012.

[8] L. Dai, X. Gao, Y. Guo, J. Xiao, Z. Zhang, *et al.*, "Bioinformatics clouds for big data manipulation," *Biology direct*, 2012.

[9] Y. Yang, N. Peyerimhoff, and I. Ivrissimtzis, "Linear correlations between spatial and normal noise in triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[10] R. Pintus, Y. Yang, and H. Rushmeier, "Athena: Automatic text height extraction for the analysis of text lines in old handwritten manuscripts," *ACM Journal on Computing and Cultural Heritage*, vol. 8, no. 1, pp. 1–1, 2015.

[11] X. Zhou, X. Sun, G. Sun, and Y. Yang, "A combined static and dynamic software birthmark based on component dependence graph," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2008.*, pp. 1416–1421, IEEE, 2008.

[12] Y. Wang, G. Agrawal, G. Ozer, and K. Huang, "Removing sequential bottlenecks in analysis of next-generation sequencing data," in *IPDPSW, 2014 IEEE International*, IEEE, 2014.

[13] T. Li, I. Raicu, and L. Ramakrishnan, "Scalable state management for scientific applications in the cloud," BigData Congress '14.

[14] Y. Wang, W. Jiang, and G. Agrawal, "Scimate: A novel mapreduce-like framework for multiple scientific data formats," CCGRID '12, 2012.

[15] Y. Wang, W. Jiang, and G. Agrawal, "SciMATE: A Novel MapReduce-Like Framework for Multiple Scientific Data Formats," in *CCGrid'12*, IEEE, 2012.

[16] Y. Wang, G. Agrawal, T. Bicer, and W. Jiang, "Smart: A MapReduce-Like Framework for In-Situ Scientific Analytics," tech. rep., OSU-CISRC-4/15-TR05, Ohio State University, 2015.

[17] Y. Wang, Y. Su, and G. Agrawal, "A novel approach for approximate aggregations over arrays," in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, p. 4, ACM, 2015.

[18] T. Li, K. Keahey, R. Sankaran, P. Beckman, and I. Raicu, "A cloud-based interactive data infrastructure for sensor networks," IEEE/ACM Supercomputing/SC'14.

[19] T. Li, K. Keahey, K. Wang, D. Zhao, and I. Raicu, "A dynamically scalable cloud data infrastructure for sensor networks," ACM ScienceCloud 15.

[20] Y. Wang, Y. Su, and G. Agrawal, "Supporting a Light-Weight Data Management Layer Over HDF5," in *CCGrid'13*, IEEE.

[21] K. Wang, N. Liu, I. Sadooghi, X. Yang, X. Zhou, T. Li, M. Lang, X.-H. Sun, and I. Raicu, "Overcoming hadoop scaling limitations through distributed task execution," IEEE Cluster'15, 2015.

[22] D. Zhao, Z. Zhang, X. Zhou, T. Li, K. Wang, D. Kimpe, P. Carns, R. Ross, and I. Raicu, "Fusionfs: Towards supporting data-intensive scientific applications on extreme-scale high-performance computing systems," in *Big Data, 2014 IEEE International Conference on*.

[23] D. Zhao, C. Shou, Z. Zhang, I. Sadooghi, X. Zhou, T. Li, and I. Raicu, "Fusionfs: a distributed file system for large scale data-intensive computing," GCASR' 13.

[24] H. Wang and J. Wang, "An effective image representation method using kernel classification," in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI 2014)*, 2014.

[25] J. Wang, H. Wang, Y. Zhou, and N. McDonald, "Multiple kernel multivariate performance learning using cutting plane algorithm," in *2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC2015)*, p. Inpress, IEEE, 2015.

[26] J. Wang, Y. Zhou, K. Duan, J. J.-Y. Wang, and H. Bensmail, "Supervised cross-modal factor analysis for multiple modal data classification," in *2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC2015)*, p. Inpress, IEEE, 2015.

[27] S. Zhang, X. Zhang, and X. Ou, "After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across iaas cloud," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pp. 317–328, ACM, 2014.

[28] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal, "Aggregating vulnerability metrics in enterprise networks using attack graphs," *J. Comput. Secur.*, vol. 21, 2013.

[29] S. Zhang, *QUANTITATIVE RISK ASSESSMENT UNDER MULTI-CONTEXT ENVIRONMENTS*. PhD thesis, Kansas State University, 2014.

[30] I. Sadooghi, J. Hernandez Martin, T. Li, K. Brandstatter, Y. Zhao, K. Maheshwari, T. Pais Pitta de Lacerda Ruivo, S. Timm, G. Garzoglio, and I. Raicu, "Understanding the performance and potential of cloud computing for scientific applications," *IEEE Transactions on Cloud Computing*, 2015.

[31] 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*, SIGMOD '10, 2010.

[32] "Apache giraph." http://giraph.apache.org. Accessed: 2015-1-30.

[33] "Graphlab." http://graphlab.org. Accessed: 2015-1-30.

[34] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," IPDPS '13.

[35] T. Li, R. Verma, X. Duan, H. Jin, and I. Raicu, "Exploring distributed hash tables in highend computing," *SIGMETRICS Performance Evaluation Review*.

[36] T. Li, X. Zhou, K. Wang, D. Zhao, I. Sadooghi, Z. Zhang, and I. Raicu, "A convergence of key-value storage systems from clouds to supercomputers," *Concurr. Comput. : Pract. Exper.(CCPE)*, 2015.

[37] T. Li, X. Zhou, K. Brandstatter, and I. Raicu, "Distributed key-value store on hpc and cloud systems," GCASR' 13, 2013.

[38] K. Brandstatter, T. Li, X. Zhou, and I. Raicu, "Novoht: a lightweight dynamic persistent NoSQL key/value store," GCASR' 13, 2013.

[39] T. Li, A. P. D. Tejada, K. Brandstatter, Z. Zhang, and I. Raicu, "ZHT: a zero-hop DHT for high-end computing environment," GCASR, 2012.