A scalable sparse matrix-vector multiplication architecture with Accumulo and D4M

Yin Huang
Computer Science and
Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD, 21220
Email: yhuang9@umbc.edu

Yelena Yesha
Computer Science and
Electrical Engineering
UMBC
Baltimore, MD, 21220

Shujia Zhou
Computer Science and
Electrical Engineering
UMBC
Baltimore, MD, 21220

Abstract—**The increasing volume and velocity of large unstructured datasets have been calling for new technologies to store, query, and analyze data of interest. On one hand, NoSQL distributed databases have been devised to meet the need of big data. On the other hand, scalable linear algebra operations on sparse large matrix has become more and more important. Because most datasets can be represented as a sparse large graph. This paper presents a novel scalable architecture for analyzing massive graphs, with a focus on computing sparse matrix-vector multiplication (SpMV). Our architecture provides scalable linear algebra operations building on recently-developed technologies such as Accumulo, D4M and pMatlab. We store the data in Dynamic Distributed Dimensional Data Model(D4M) format for easy extraction from Accumulo database while pMatlab serves as a parallel computation engine. The principal analysis algorithm is Lanczos-SO for calculating top k eigenvalues and eigenvectors of a matrix. Experiments on Graph500 benchmark datasets demonstrate the scalability and efficiency of our architecture.**

I. Introduction

Typical data analytics normally include the following pipeline: collecting data, querying data, analyzing data and report. Nowadays Hadoop plays a fundamental role in tackling the challenges caused by increasing volume and velocity of unstructured data. The success of Hadoop relies on its two components: Hadoop Distributed File System (HDFS) for storing massive amount of data with redundancy for failure tolerance and MapReduce for batch processing.

In many applications it is intuitive to represent data as a graph to discover patterns hidden underneath. Graph representation has a wide range of applications from social sciences to physics and bio-informatics . Take social media for example, a sparse adjacent matrix for all Twitter users can be built to reflect their relationships. Construction of such a matrix, however, requires complicated operations. Moreover, to find users of similar interests, we need apply linear algebra operations to this matrix. Recent work has focused on constructing graphs from the data stored in the D4M format and applying eigendecomposition to the modularity matrix. "[**?**]". In their paper, the authors store their data on a single database node which will become the bottleneck as the data size increases. Our architecture differs from theirs in that we deploy Accumulo, a distributed NoSQL database, as our data storage.

Sparse matrix-vector multiplication (SpMV) is of great importance in sparse linear algebra given the fact that they represent the dominant cost in many iterative methods for solving large-scale linear systems and eigenvalue problems that arise in a wide variety of scientific and engineering applications. "[**?**]" For example, SpMV is the most expensive operation in Lanczos-SO algorithm employed in HEIGEN which is an eigensolver for billion-scale graphs based on Hadoop. MapReduce, however, is not the best approach for iterative algorithms due to the intermediate shuffling of data among work nodes. "[**?**]"

In this paper, we introduce a scalable massive graph analysis architecture integrating D4M and Accumulo with the focus on SpMV. This architecture encompasses the entire data analytics pipeline, from data collection to data extraction of relational structure to data analysis of the resulting graph. More important, we exploit the statistics information from Accumulo table to balance the load and distribute the load evenly among work nodes by issuing queries to fetch rows of a matrix from database table. In addition, we use pMatlab to do parallel processing. In the end, we present and compare our experimental results on Lanczos-SO algorithm against HEIGEN. Our platform shows almost twice speed-up and great scalability.

The rest of the paper is organized as follows. Section 2 describes the architecture, discusses the data storage format and the graph construction procedure. Section 3 explains Lanczos-SO algorithm. Section 4 focuses on D4M and Accumulo. Section 5 demonstrates our implementation of eigensolver using D4M and Accumulo. Section 6 is for experimental results and discussion and section 7 comes to a conclusion.

II. System architecture

Our system consists of the following 3 components: First, the bottom layer is Accumulo database where the data are stored in the D4M format, which provides an easy to use interface for accessing subsets of data. We can thus build graphs representing various types of relationships. Second is the service layer containing Matlab and MatlabMPI, both of which provide the computation resource to the upper layer. Third is the user layer where associative arrays query and store the data to be processed while pMatlab handles the parallel computation.

III. Relevant work

To be completed

IV. Conclusion

To be completed

Acknowledgment

The authors would like to thank...

References

[1] L.N. Trefethen and D. Bau III, Numberical Linear Algebra, SIAM, 1997.

[2] U Kang, Breandan Meeder, Evangelos E. Papalexakis, and Christos Faloutsos, HEigen: Spectral Analysis for Billion-Scale graphs, IEEE Transactions on knowledge and data engineering, VOL. 26, No.2, Feb 2014.

[3] Ankur Dave, Wei Lu, Jared Jackson, Roger Barga, Cloudclustering: Toward an iterative data processing pattern on the cloud.

[4] Jermey Kepner, William Arcand, etc. DYNAMIC DISTRIBUTED DIMENSIONAL DATA MODEL (D4M) DATABASE AND COMPUTATION SYSTEM

[5] J.Kepner, Parallel Matlab for Multicore and Multinode computers, SIAM Press, Philadelphia, 2009

[6] N. Bliss and J. Kepner, pMatlab Parallel Matlab Library, International Journal of High Performance Computing Applications: Special Issue on High Level Programming Languages and Modesl, J.Kepner and H. Zima (editors), Winter 2006 (November)

[7] J. Kepner and S. Ahalt, MatlabMPI, Journal of Parallel and Distributed Computing, vol. 64, issue 8, August, 2004

[8] N. Bliss, R. Bond, H. Kim, A. Reuther, and J. Kepner, Interactive Grid Computing at Lincoln Laboratory, Lincoln Laboratory Journal, vol. 16, no. 1, 2006.