Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 6, pp. 955-962, December 2020
https://doi.org/10.34163/jkits.2020.15.6.004

Journal of Knowledge Information Technology and Systems

ISSN 1975-7700 (Print), ISSN 2734-0570 (Online)

http://www.kkits.or.kr

A Study on Big Data-based GraphX Model for Social Network Service

Leesang Cho[1], Jinhong Kim[*2]

[1]*Division of Mechanical and Electronics Engineering, Hansung University*
[2]*Department of Computer Engineering, Paichai University*

ABSTRACT

Nowaday, towards adopting big data processing system has increased and it is commonly seen in every aspect of life. For this, the problem of finding connected components in undirected graphs has been well studied, and it is an essential pre-processing step to many graph computations, and a fundamental task in graph analytics applications. Recently, it has been a main area of interest in the large graph processing. However, much of the research has focused on solving the problem using High Performance Computers. In large distributed systems, the MapReduce framework dominates the processing of big data, and has been used for finding connected components in big graphs although iterative processing is not directly supported in MapReduce. Current big data processing systems have developed into supporting iterative processing and providing additional features other than MapReduce. This research investigates how to enhance the performance of finding connected components algorithm for large graph in distributed processing system. It uses the approach to considering the graph degree property in choosing the component identifier, reviewing how this can affect the efficiency of the algorithm. In the design of our proposed algorithm features provided by current new processing systems such as moving the computation more toward the data partition in Spark framework model are integrated.

KEYWORDS: Graph analytics applications, Large graph processing, Distributed systems, Big data processing systems, Spark framework model

ARTICLE INFO: Received 27 November 2020, Revised 3 December 2020, Accepted 11 December 2020.

[*]Corresponding author is with the Department of Computer Engineering, Paichai University, 155-40 Baejae-ro, Seo-Gu Daejeon, 35345, KOREA.
E-mail address: jinhkm@pcu.ac.kr

1. Introduction

Ever since, the trend towards adopting big data processing system has increased and it is commonly seen in every aspect of life. Therefore, it has become cost of storage is decreasing and the ability to capture different kind of data is growing especially [1]. In view of the diversity of data acquired nowadays and the massive amount of data stored, it needs to find new methods to deal with data beyond the traditional database, such as handling data that do not usually fit in a single machine memory or disk. One approach is to look at data as a network or a graph with edges connecting things together, those edges can take different forms of relationships [2]. This metaphor of graph is currently used in many areas: computer science, economics, sociology, biology, and many more. Almost anything can be represented as a graph. Graphs are considered to be a very flexible data model and to recognize local and global characteristics of the system, and to analyse different features of the complex networks. Extensive research has been carried out on graphs and graph processing, and have been extensively used to efficiently process data and extract knowledge [3]. However, recent graphs are beyond the ability of traditional systems to handle, either because the sizes of current graphs are very big, and they usually do not fit in a machine's memory, or because current algorithms cannot process such graphs efficiently, particularly when using the current distributed systems. Our focus in this research is on the problem for finding Connected Components efficiently in an undirected graph [4]. A component represents a graph (or subgraph) where any two vertices inside that graph are connected via paths, and there is no edge that connects any vertex outside the component [5]. This problem has been well studied, as it is an essential pre-processing step to many graph computations, and is a building block in complex graph analysis such as clustering. They have large graph process ing and much of the research so far has focused on solving the problem using High Performance Computers [6], with high computation power and equipped with very large memory capacity. Large-scale graphs (or big graphs) are usually stored using a distributed file system, like Hadoop, either in the cloud or locally [7]. Hadoop provide open source software framework of commodity computers, and the MapReduce framework dominates the processing of large-scale data on Hadoop, and it is commonly used for mining big graphs. However, iterative processing is not directly supported in MapReduce. Our research builds on the knowledge that current big data processing systems have become more advanced with features beyond MapReduce. For example, a processing system, like Spark, supports iterative processing and provides additional features other than MapReduce such as data partitioning and caching [8]. Spark also supports graph processing using GraphX, which is an open source Spark API for graph-parallel computation. Moreover, current connected component algorithms in large distributed processing system only use the traditional approach in choosing the component

identifier for each connected component based on the lexical ordering of the node ID value [9].

This paper is organized as follows. In Section 2 details Big Data. Then graph for system modeling is presented in Section 3 and Big Data-based GraphX in Section 4. Finally, Section 5 is conclusion in this paper.

2. Big Data

Big data is broadly defined as data that is too big, fast, and hard to deal with using conventional database tools. A more technical view is provided that define Big data as data that requires new technologies and architectures. This is because the database management tools or traditional data processing applications are unable to process the data in a timely, cost effective way, because it is too large to be stored and processed and too complex and varied to be analysed and visualized. Big data have a three concepts that are used to explain as bellows; 1) Volume: is the word associated with “BIG” in big data [10]. It includes the increasing massive amount of data collected and produced and goes beyond the ability to hold and process easily. 2) Variety: data come from many sources. These include, for example, web logs, sensor data, social media data, emails, images, documents and audio. Data in general comes in three types: structured, semi-structured and unstructured. Data Variety is probably the hardest to manage when processing a large amount of data. 3) Velocity: is concerned with the speed of the data coming from various sources. For example, streaming data and sensor data or data that is required to be handled in real-time.

Hadoop is an open source software framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models [11]. It is based on MapReduce. HDFS was developed for reliable, scalable, distributed computing. It allows working with thousands of computers and dealing with petabytes of data. It operates on commodity computers to store data across hundreds of computers [12]. Data nodes will host files, files are divided into chunks (usually 64 megabytes size), which are replicated on different disks (usually three times, one disk should be on a different rack). A Master node has a directory that records where each file is stored and replicated. MapReduce gives the programmer the advantages of not needing to consider the details of data distribution [13], parallel executing, replication and load balancing. Its programming concept is familiar and allows parallelised and distributed execution for jobs across clusters of computers [14]. It requires two functions as bellows; 1) Map function, which is defined by the programmer to process key-value data. Each chunk or more of a given data will be processed by the map function and gives output as key-value pairs. They are then divided among reducers in such a way that each group with the same key goes to the same reducer. 2) Reducer function, takes the key-value pairs and combines all the values associated with the same key and carries out any computation defined by the programmer. It then outputs the new value. The

reducer output could be in key-value pairs to feed another mapper in an iterative way [15].

After all, Hadoop cluster is at least one machine running the Hadoop software. In each cluster, there is a single master node with a varying number of slave nodes. Slave nodes can act as both the computing nodes for the MapReduce and as data nodes for the HDFS.

3. Graph for System Modeling

With the huge amount of data collected, there is an urgent need to efficiently deal with it and extract knowledge that no one has discovered before. One approach is to look at this data as a network with links connecting things together, those links can take different kind of forms of relationships. This metaphor of networks is currently used in many areas: computer science, economics, sociology, biology, and many more. It can effectively address many of the challenges in each area by understanding the "connectedness" of these complex systems. Modelling the relationship in a network by graphs helps to generate a natural human interpretation and simple mechanical analysis. Since then mathematicians extensively studied graph and its properties. Graphs have been used to understand complex human and natural phenomena. In general, graph is used in any domain when there is a need to find a network representation of logical or physical links between entities. Its applications spread on wide variety of domains such as: linguistics, economics, sociology, biology, chemistry, and pharmacology (e.g. graphs model the complicated structure of chemical compounds and protein structures), and computer science (e.g. Worldwide Web, workflows, XML documents, computer networks, physical connections, computer vision, video indexing, text retrieval and social networks) and many more, where graph algorithms have been developed to solve different kinds of problems. <Figure 1> shows the detailed Graph for Spark Architecture.

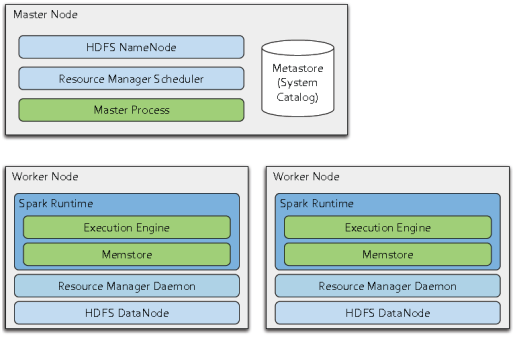


Figure 1. Graph for Spark Architecture

4. Big Data-based GraphX

There is an urgent need to deal with structured, unstructured, and heterogeneous data instead of the traditional one. Thus, graphs are now widely used because of its expressive power and the ability to connected object in different way. However, mining is hard to implement because of the structure of the data, and size of the data as the real-world graphs are very big, and usually does not fit in a machine memory. Adding to that, there is no single model that efficiently fits all the types of graph algorithms and application, nonetheless many have been developed to solve specific problems or to meet

some special classes of applications. A rising trend to capture and store any data available, especially as the cost of storage decreasing and ability to capture different kind of data increase. Pushed by the world getting more connected; more connected devices and embedded sensors and expanding networks and others all contribute to found the area of the Internet of Things (IoT); in addition, people life is more digital nowadays than ever before, and there is increasing presence of social media in our life (Facebook, Twitter, Snapchat, Instagram, and many more). The existing real-world dataset is getting large enormously, these datasets reflect different kind of relationships and can be generally efficiently represented using graph structures. However, as the graph grow larger their size and complexity go beyond single processing machine ability and make processing it with HPC systems a challenging task which is not always suitable for it. Accordingly, we have several extensions that it made to Spark to efficiently relational data and run SQL for GraphX as shown by <Figure 2>.

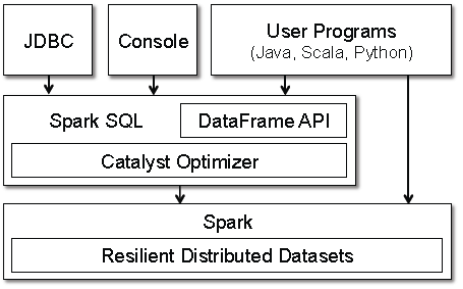


Figure 2. Spark Interface SQL for GraphX

GraphX processing library on top of Spark, and it is implemented on top of its dataset API. It supports the implementation of different graph processing models, such as vertex-centric, gather-sum-apply, and scatter-gather. GraphX is represented by a dataset of vertices and a dataset of edges, and also <Figure 3> shows the detailed description about each data processing stages Graph for Spark Framework Model.

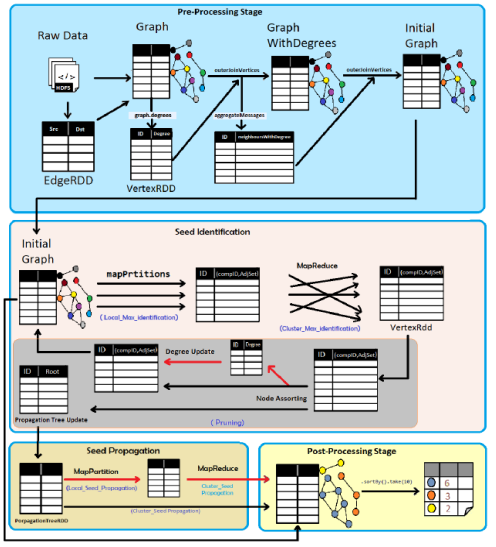


Figure 3. Spark Framework Model

As mentioned above, we have a simulation message_Identification class used for message used in the max identification & pruning steps. message_Tree class used for generate update messages for seed propagation tree T. message_Propagation used in the Seed Propagation Phase to hold the updates are generated from the root to the child nodes, where each node notifies its child nodes with its component identifier. <Figure 4> presents the class diagram of the classes used for exchanging messages between nodes.

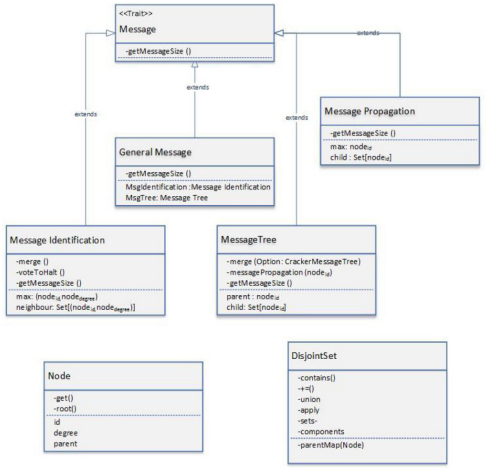


Figure 4. Class Diagram

5. Conclusions

In this paper, this research has been to examine the processing of large-scale graphs and more specifically, enhance the performance of finding connected components algorithms in large graphs. Finding connected components is an essential pre-processing step to extract knowledge about the graph. It is also a fundamental operation for some graph computations. The MapReduce framework dominates the processing of large-scale data on Hadoop, and it is commonly used for mining big data graphX. However, iterative processing is not directly supported in MapReduce. Nonetheless, some recent works show that it is possible to outperform other models for finding connected components using MapReduce. Accordingly, we need to more experiment for GraphX within Big data.

References

[1] S. T. Hwang, *A study on big data platform architecture-based conceptual measurement model using comparative analysis for social commerce*, Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 5, pp. 623-630, Oct. 2020.

[2] J. S. Kim, and J. H. Kim, *A study on adaptive smart platform for intelligent software in big data environment,* Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 3, pp. 347-355, Jun. 2020.

[3] H. K. Chang, *Context recognition method for personalization Service in bigdata environment,* Journal of Knowledge Information Technology and Systems(JKITS), Vol. 15, No. 5, pp. 631-638, Oct. 2020.

[4] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, and S. Whittle, *The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing.* Proceeding VLDB Endow, Vol. 8 No. 12, pp. 1792-1803, Aug. 2015.

[5] P. Alvaro, N. Conway, J. M. Hellerstein, and R. Marczak, *Consistency analysis in bloom: A calm and collected approach.* In Proceedings 5th Biennial Conference on Innovative Data Systems Research, pp. 249-260, 2011.

[6] B. Schilit, N. Adams, and R. Want, *Context -aware computing applications,* First Work shop on Mobile Computing Systems and Applications, pp. 85-90, 1994.

[7] D. Salber, A. K. Dey, and G. D. Abowd,

The context toolkit: aiding the development of context Aware applications, In the Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick Ireland, Jun. 2000.

[8] N. Pandeeswari, and G. Kumar. *Anomaly detection system in cloud environment using fuzzy clustering based ANN*. In: Mob. Netw. Appl. 21.3, pp. 494-505, 2016.

[9] R. J. Hyndman, E. Wang, and N. Laptev, *Large-scale unusual time series detection*. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW). IEEE, pp. 1616-1619, 2015.

[10] B. Agrawal, A. Chakravorty, C. Rong, and T. W. Wlodarczyk. *R2Time: A framework to analyse open TSDB time-series data in HBase.* In: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference, pp. 970-975, 2014.

[11] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, *Online detection of utility cloud anomalies using metric distributions.* In: Network Operations and Management Symposium (NOMS), 2010 IEEE, pp. 96-103, 2010.

[12] M. Jahrer, A. Toscher, J. Y. Lee, J Deng, H. Zhang, and J. Spoelstra. *Ensemble of collaborative filtering and feature engineered models for click through rate prediction.* In: KDDCup Workshop. 2012.

[13] P. Gaikwad, A. Mandal, P. Ruth, G. Juve, D. Krol, and E. Deelman. *Anomaly detection for scientific workflow applications on networked clouds*. In: High Performance Computing & Simulation (HPCS), 2016 International Conference on. IEEE, pp. 645-652, 2016.

[14] U. Kang, D. H. Chau, and C. Faloutsos, *Mining large graphs: Algorithms, inference, and discoveries,* in 2011 IEEE 27th International Conference on Data Engineering, pp. 243-254, 2011.

[15] T. Rabl, N. Raghunath, M. Poess, M. Bhandarkar, H.-A. Jacobsen, and C. Baru, Eds., *Advancing Big Data Benchmarks*, Vol. 8585. Cham: Springer International Publishing, 2014.

소셜 네트워크 서비스를 위한 빅데이터 기반 GraphX 모델에 관한 연구

조이상[1], 김진홍[2]

[1] *한성대학교 기계전자공학부 조교수*

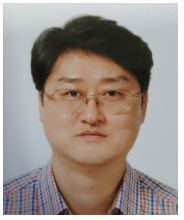
[2] *배재대학교 컴퓨터공학과 조교수*

요 약

오늘날 빅데이터 처리 시스템의 증가로 인해 삶의 모든 측면에서 일상적으로 볼 수 있다. 이를 위해, 무방향 그래프에서 연결된 구성 요소를 찾는 문제에 대해 잘 연구되고 있으며, 많은 그래프 계산에 필수적인 전처리 단계로서 그래프 분석 애플리케이션의 기본 작업이다. 최근 큰 그래프 처리에서의 주요 관심 분야이기도 하다. 하지만, 대부분의 연구는 고성능 컴퓨터를 사용하여 문제를 해결하는 데 중점을 두고 있다. 대규모 분산 시스템에서 MapReduce 프레임워크는 빅데이터 처리를 이용하며, 반복 처리가 MapReduce에서 직접 지원되지는 않지만 큰 그래프에서 연결된 구성 요소를 찾는 데 사용되고 있다. 현재 빅데이터 처리 시스템은 반복 처리를 지원하고, MapReduce 이외의 추가 기능을 제공한다. 따라서, 본 연구는 분산 처리 시스템에서 큰 그래프에 대한 연결 성분 찾기 알고리즘의 성능을 향상시키는 방법을 제안하였다. 구성 요소 식별자를 선택할 때 그래프 정도 속성을 고려하는 접근 방식을 사용하여 알고리즘의 효율성이 어떤 영향을 미칠 수 있는지 검토하였다. Spark의 프레임워크 모델로 더 많은 계산을 이동하는 것과 같이 현재 새로운 처리 시스템에서 제공되도록 제안하였다.

Acknowledgments

This research was financially supported by Hansung University.



Leesang Cho is Assistant Professor of Division of Mechanical and Electronics Engineering at the Hansung University, Seoul, Rep. of Korea. He received his Ph.D. degrees in Mechanical Engineering from Hanyang University, Rep. of Korea, in 2008. Moreover, He also served or currently serving as a reviewer and Technical Program Committee for many important Journals, Conferences and Research Project in Aircraft and Drone. His research interests include Artificial Intelligent and Big Data for Aircraft and Drone.

E-mail address: ppome815@hansung.ac.kr

Jinhong Kim is Assistant Professor of Department of Information Computer Engineering at the Pai Chai University, Daejeon, Korea. He respectively received his Ph.D. degrees in Electronic, Electrical and Computer from Sungkyunkwan University, Korea, in 2006. Moreover, He also served or currently serving as a reviewer and Technical Program Committee for many important Journals, Conferences, Symposiums, Workshop in Big Data area. His research interests include Smart Vehicular Network, Smart Platform, Machine Learning, Artificial Intelligent, Intelligent Software, Intelligent Agent System, and Big Data. He is a life member of the KKITS.

E-mail address: jinhkm@pcu.ac.kr