BIG GRAPH: TOOLS, TECHNIQUES, ISSUES, CHALLENGES AND FUTURE DIRECTIONS

Dhananjay Kumar Singh[1] and Ripon Patgiri[2]

Department of Computer Science & Engineering
National Institute of Technology Silchar
Assam, India-788010
[1]eng.dksingh@gmail.com
[2]ripon@cse.nits.ac.in

ABSTRACT

Analyzing interconnection structures among the data through the use of graph algorithms and graph analytics has been shown to provide tremendous value in many application domains (like social networks, protein networks, transportation networks, bibliographical networks, knowledge bases and many more). Nowadays, graphs with billions of nodes and trillions of edges have become very common. In principle, graph analytics is an important big data discovery technique. Therefore, with the increasing abundance of large scale graphs, designing scalable systems for processing and analyzing large scale graphs has become one of the timeliest problems facing the big data research community. In general, distributed processing of big graphs is a challenging task due to their size and the inherent irregular structure of graph computations. In this paper, we present a comprehensive overview of the state-of-the-art to better understand the challenges of developing very high-scalable graph processing systems. In addition, we identify a set of the current open research challenges and discuss some promising directions for future research.

KEYWORDS

Big Data, Big Graph, Graph Processing, Graph Analytics, Graph Parallel Computing, Distributed Processing, Graph Algorithms

1. INTRODUCTION

The Big Data delineates huge data set to store, process and analyze. These data are generated by the daily aggrandizement of data from various sources, call for Big Data to knob these perplex data. Therefore, the conventional means of handling data are now obsolete, emerging Big Data with full-fledged, and qui vive for research in these data. Therefore, NoSQL is a prominent field in Big Data. The Big Graph is part of NoSQL which is gigantic in size, perplex to handle, and arduous to visualize.

Recently people, devices, processes and other entities have been more connected than at any other point in history. In general, the complex relationships, interactions and interdependencies between objects are naturally modelled as graphs. Therefore, graphs have been used to represent data sets in a wide range of application domains, such as social science, astronomy, computational biology, telecommunications, semantic web, protein networks, and many more. In practice, graph analytics is an important and effective big data discovery tool. For example, it enables identifying influential persons in a social network, inspecting fraud operations in a complex interaction network and recognizing product affinities by analyzing community buying patterns.

Nowadays, graphs with millions and billions of nodes and edges have become very common. For example, in 2012, Facebook has reported that its social network graph contains more than a billion users (nodes) and more than 140 billion friendship relationships (edges). The enormous growth in graph sizes requires huge amounts of computational power to analyze. In practice, distributed processing of large scale graphs is a challenging task due to their size in addition to their inherent irregular structure and the iterative nature of graph processing and computation algorithms. Graph algorithms are becoming increasingly important for analyzing large datasets in many fields. Real-world graph data follows a pattern of sparsity that is [1]not uniform, but highly skewed towards a few items. Implementing graph traversal, statistics and machine learning algorithms on such data in a scalable manner is quite challenging. As a result, several graph analytics frameworks such as Giraph, FlashGraph, GraphChi, X-Stream and many more, have been developed, each offering a solution with different programming models and targeted at different users.

The rest of this paper is organized as follows: Section 2 provides basic information about Big Graph. In Section 3, we discussed about Big Graph processing systems like Apache Giraph, GPS, and many more. We present Big Graph Analytics frameworks like Ringo, PowerGraph and so on in Section 4. Graph Algorithms for solving many problems in scientific computing, data mining and other domains, are discussed in Section 5. The future directions of Big Graph are discussed in Section 6. And finally, Section 7 concludes the paper.

2. BIG GRAPH

We can simply define Big Graph as,

“**Big Data + Structure = Big Graph**”

Big graphs are ubiquitous, ranging from social networks and mobile call networks to biological networks and the World Wide Web. The sources of real-world large-scale graphs include:

- Social graphs (Facebook, Twitter, Google+, LinkedIn, etc.)
- Endorsement graphs (web link graph, paper citation graph, etc.)
- Location graphs (map, power grid, telephone network, etc.)

The size of large scale graphs used in the recent literature is given in table 1.

[1]`https://www.ibm.com/developerworks/library/os-giraph`

Table 1: Large Scale Graphs in current literature[1]

Name	Nodes	Edges
Web Graph	More than 20 billion nodes (pages)	More than 160 billion edges (hyperlinks)
Facebook	More than a billion nodes (users)	More than 140 billion edges (friendship relationships)
LinkedIn	Almost 8 million nodes	Almost 60 million edges
SemanticWeb	3.7 million nodes (objects)	400 million edges (facts)

3. BIG GRAPH PROCESSING

The growth of graph-structured data in modern applications such as social networks and knowledge bases creates a crucial need for scalable platforms and parallel architectures that can process it in bulk.

3.1. Pregel

Pregel[3] is a scalable, general-purpose system for implementing graph algorithms in a distributed environment. It is known as the first Bulk Synchronous Parallel (BSP), an implementations that provides a native API specifically for graph algorithms using a "think like a vertex" computing paradigm. The basic computation model of Pregel is shown in figure 1.

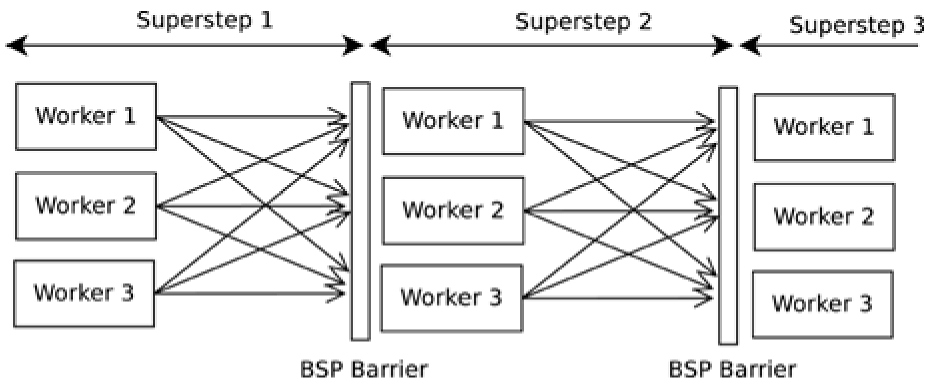

Figure 1. Pregel Computation Model with three supersteps and three workers[4].

In Pregel, programs are expressed as a sequence of iterations (supersteps), in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges. The input graph is loaded once at the start of a program and all computations are executed in-memory.

3.2. Giraph

Apache Giraph[1] is an iterative graph processing system built for high scalability. It runs workers as map-only jobs on Hadoop and uses HDFS for data input and output. Giraph adds several features including master computation, sharded aggregators, edge-oriented input, out-of-core computation, and more. It also uses Apache ZooKeeper for coordination, checkpointing, and failure recovery schemes. With a steady development cycle and a growing community of users worldwide, Giraph is a natural choice for unleashing the potential of structured datasets at a massive scale. For example, it is currently used at Facebook to analyze the social graph formed by users and their connections.

3.3. Mizan

Mizan[8] is a scalable framework for supporting graph mining algorithms in large parallel computing infrastructures. It is a layer between the users' code and a variety of computing infrastructures, such as Linux clusters, cloud environments and supercomputers. Mizan examines the graph structure and decides the placement of the data and the low level message passing mechanism transparently to the users' code. It uses message passing and implements an instance of the Bulk Synchronous Parallel model.

3.4. GPS

GPS[9] is an open-source system for scalable, fault-tolerant, and easy-to-program execution of algorithms on extremely large graphs. It is a distributed system, designed to run on a cluster of machines, such as Amazon's EC2. GPS offers Large Adjacency List Partitioning (LALP), an optional performance optimization of algorithms that send to all of its neighbors the same message. GPS also features an optional dynamic migration scheme. Dynamic migration repartitions the graph during the computation by migrating vertices between workers, to improve workload balance and network usage.

3.5. GraphLab

GraphLab[11] is a framework for asynchronous parallel graph computations in machine learning. It differs from Pregel in that it does not work in bulk synchronous steps, but rather allows the vertices to be processed asynchronously based on a scheduler. The vertex functions can run at any time as long as specified consistency rules are obeyed. It is therefore well-suited for the machine learning types of applications for which it is defined, where each vertex accumulates information from its neighbors' states and updates its state, possibly asynchronously. It extends the shared memory GraphLab abstraction to the distributed setting by refining the execution model, relaxing the scheduling requirements, and introducing a new distributed data-graph, execution engines, and fault-tolerant systems.

3.6. Graph Sample and Hold

Graph Sample and Hold (gSH)[7] is a stream sampling framework that supports analytics over big graphs. The nice property of gSH consists in building theoretically sound unbiased estimators derived from the sample graph, but still robust for the estimation of different properties of the target (big) graph, yet conveying in high accuracy and tolerable approximation errors.

3.7. Pregelix

Pregelix[6] is a dataflow-based Pregel-like system built on top of the Hyracks parallel dataflow engine. It combines the Pregel API from the systems world with data-parallel query evaluation techniques from the database world in support of large scale graph analytics. This combination leads to effective and transparent out-of-core support, scalability, and throughput, as well as increased software simplicity and physical flexibility. To the best of our knowledge, Pregelix is the only open source Pregel-like system that scales to out-of-core workloads efficiently, can sustain multi-user workloads, and allows runtime flexibility.

4. BIG GRAPH ANALYTICS

Analytics is the ability to discover meaningful patterns and interesting insights into data. Graph analytics is a special piece of analytics where the underlying data can be modeled as a set of graphs. Graph analytics is a rapidly developing area where a combination of graph-theoretic, statistical and database techniques are applied to model, store, retrieve, and perform analyses on graph-structured data.

4.1. In-Memory Big Graph Analyitcs

PowerGraph: A scalable, distributed graph computation framework written in C++. PowerGraph[12] supports both the highly-parallel bulk-synchronous Pregel model of computation as well as the computationally efficient asynchronous GraphLab model of computation. PowerGraph exploits the Gather-Apply-Scatter (GAS) model of computation to factor vertex-programs over edges, splitting high-degree vertices and exposing greater parallelism in natural graphs. It allows vertex-partitioning to effectively place its large scale graph in a distributed environment.

GraphX: GraphX[14] enables distributed dataflow frameworks such as Spark to naturally express and efficiently execute iterative graph algorithms. To achieve performance parity with specialized graph systems, GraphX recasts graph-specific optimizations as distributed join optimizations and materialized view maintenance. By leveraging advances in distributed dataflow frameworks, GraphX brings low-cost fault tolerance to graph processing. GraphX API enables the composition of graphs with unstructured and tabular data and permits the same physical data to be viewed both as a graph and as collections without data movement or duplication.

Ringo: A system for construction and analysis of large scale graphs on a single large memory multicore machine that combines high productivity analysis with fast and scalable execution times. Ringo [10] table operations, transformations between tables and graphs, and several graph algorithms are fully parallelized to take full advantage of the multi-core environment, and the set of graph algorithms available for parallel execution is under constant expansion.

4.2. SSD-Based Big Graph Analytics

FlashGraph: It stores vertex state in memory and edge lists on SSDs. FlashGraph[19] runs on top of the set-associative file system (SAFS), a user-space filesystem designed to realize both high IOPS, and lightweight caching for SSD arrays on non-uniform memory and I/O systems. It uses an asynchronous user-task I/O interface to reduce overhead associated with accessing data in the filesystem and overlap computation with I/O. FlashGraph selectively accesses edge lists required by a graph algorithm from SSDs to reduce data access; it conservatively merges I/O requests to increase I/O throughput and reduce CPU consumption; it further schedules the order of processing vertices to help merge I/O requests and maximize the page cache hit rate.

4.3. Disk-Based Big Graph Analytics

GraphChi: A disk-based system for computing efficiently on graphs with billions of edges. GraphChi[13] is able to execute several advanced data mining, graph mining, and machine learning algorithms on very large graphs, using just a single consumer-level computer. It partitions the vertices into disjoint intervals and breaks large edge list into smaller shards containing edges with destinations in corresponding intervals. GraphChi uses a vertex-centric processing model, which gathers data from neighbors by reading edge values, computes and applies new values to the vertices, and scatters new data to neighbors by writing values on the edges.

X-Stream: An edge-centric graph processing system, uses streaming partitions to utilize the sequential streaming bandwidth of the storage medium for graph processing. X-Stream[17] introduces an edge-centric scatter-gather processing model. In the scatter phase, it streams every edge, and generates updates to propagate vertex states. In the gather phase, it streams every update, and applies it to the corresponding vertex state.

TurboGraph: A disk-based graph engine that process billion-scale graphs very efficiently by using modern hardware on a single PC. TurboGraph[15] is the first truly parallel graph engine that exploits full parallelism, including multi-core parallelism and FlashSSD IO parallelism, and full overlap of CPU processing and I/O processing as much as possible.

Chaos: A graph processing system designed for analytics on big graphs using small clusters. Chaos [16] builds on the X-Stream single-machine graph processing system, but scales out to multiple machines. It treats the aggregate storage of all machines as a single at disk and uses work stealing to balance the load across nodes in the cluster. With very limited pre-processing, Chaos achieves sequential storage access, computational load balance and I/O load balance.

GridGraph: A system for processing large-scale graphs on a single machine using 2-level hierarchical partitioning. GridGraph[18] breaks graphs into 1D-partitioned vertex chunks, and 2D-partitioned edge blocks using a first fine-grained level partitioning in preprocessing. It uses a new streaming-apply the model that streams edges sequentially and applies updates onto vertices instantly.

4.4. Issues and Challenges of Big Graph Analytics

High-degree vertex: Graphs with high-degree vertices are computationally challenging and contribute heavily communication and storage overhead. In addition, these graphs are difficult to partition.

Sparseness: Splitting sparse graph requires more communication, more computation and more synchronization.

Data-driven computations: Graph computations are often completely data-driven. The computations performed by a graph algorithm are dictated by the vertex and edge structure of the graph on which it is operating rather than being directly expressed in code. As a result, parallelism based on partitioning of computation can be difficult to express because the structure of computations of the algorithm is not known apriori.

Unstructured problems: Similar to difficulties encountered in parallelizing a graph problem based on its computational structure, irregular structure of graph data makes it difficult to extract parallelism by partitioning the problem data.

In-memory challenge: The large-scale graph does not fit in a single memory location, because of its immense in size. Instead of SSD or HDD, the graph data should reside in the RAM, such that the response time would become minimal.

Poor locality: Because graphs represent the relationships between entities and because these relationships may be irregular and unstructured, the computations and data access patterns tend not to have very much locality. Performance in contemporary processors is predicated upon exploiting locality. Thus, high performance can be hard to obtain for graph algorithms, even on serial machines.

Communication overhead: The high-degree vertices incur communication overheads. Today, the high-degree vertices are in millions, but tomorrow will be in the billions, and beyond.

Load balancing: When we analyze graphs such as power-law graphs, then we have to pay extra attention to load balancing.

4.5. Applications of Big Graph Analytics

Graph analytics has wide ranging applications in many diverse domains such as Internet and overlay management, road networks, online social networks, etc.

4.5.1. Machine Learning

One of the most popular application of machine learning is recommendation systems. One approach to the design of recommendation systems that has wide use is collaborative filtering. The Netflix movie recommendation task uses collaborative filtering to predict the movie ratings for each user, based on the ratings of similar users.

4.5.2. Social Network Analysis

Graphs are employed heavily in online social networks. The reason for this popularity is that graphs offer a natural way of representing various kinds of relationships that are important for these applications.

4.5.3. Semantic Networks

A semantic network is a graph structure for representing knowledge in patterns of interconnected nodes and arcs. Declarative graphic representation is common to all semantic networks that can be used to represent knowledge and support automated systems for reasoning about the knowledge.

4.5.4. Biological Networks

Interactions arise naturally in biology and it can be assembled into networks of graphs where the nodes are biological entities and edges represent molecular interactions, associations between diseases.

4.5.5. Friend Recommendations

Existing social networking services recommend friends to users based on their social graphs, which may not be the most appropriate to reject a user's preferences on friend selection in real life.

5. GRAPH ALGORITHMS

Graph algorithms are becoming increasingly important for solving many problems in scientific computing, data mining and other domains.

5.1. PageRank

PageRank[20] is a method for computing a ranking for every web page based on the graph of the web. It is used by Google to rank webpages based on the idea that more important websites likely receive more links from other websites. PageRank has applications in search, browsing, and traffic estimations.

5.2. Connected Components

A connected component, in an undirected graph, is a connected subgraph of the graph. In a directed graph, connected component can either be weakly connected or strongly connected. Weakly and strongly connected component are respectively weakly and strongly connected subgraphs of a graph.

5.3. Single Source Shortest Path (SSSP)

The single-source shortest path problem is a classical problem in the research field of graph algorithm. In SSSP problem, we have to find the shortest paths from a source vertex ‘v’ to all other vertices in a graph.

5.4. Triangle Counting

In this algorithm, each vertex shares its neighborhood list with each of its neighbors. Each vertex, then checks if any of their neighbors overlap with the neighborhood list(s) they received. With directed edges and no cycles, the total number of such overlaps gives the number of triangles in the graph. With undirected edges, the total number of overlaps gives 3-times the number of triangles.

5.5. Collaborative Filtering

It is used, for example, to recommend products based on purchases of other users with similar interests. This is a machine learning algorithm that estimates how a given user would rate an item given an incomplete set of (user, item) ratings.

6. FUTURE DIRECTION

In the era of big data, interest in analysis and extraction of information from large data graphs is increasing rapidly. Graphs are now widely used for data modeling in application domains for which identifying relationship patterns, rules, and anomalies are useful. These domains include web graph, social networks, semantic web, protein-protein interaction networks, bibliographical networks, etc. The ever-increasing size of graph-structured data for these applications creates a critical need for scalable systems that can process large amounts of it efficiently.

7. CONCLUSION

The usage of large scale graph processing platforms is rapidly expanding in both academia and industry. In principle, large scale graph processing platforms are increasingly important as more and more problems require dealing with graphs. To this end, we presented a thorough survey of the state-of-the-art of the emerging platforms in this domain. In addition, we have provided an overview of the recent studies for benchmarking and evaluating some of the existing platforms. Finally, we identified and presented a set of the current open research challenges and also presented some of the promising directions for future research. In general, we believe that there are still many opportunities for new innovations and optimizations in the domain of large scale graph processing. Hence, we consider this article as an important step in helping researchers to understand the domain and guiding them towards the right direction to improve the state-of-the-art.

REFERENCES

[1] Apache Giraph. http://giraph.apache.org/

[2] Twitter: Most Followers. http://friendorfollow.com/twitter/most-followers/

[3] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A System for Large-Scale Graph Processing. In Proceeding of SIGMOD'10, Pages 135-146, June 6-11, 2010.

[4] Minyang Han, Khuzaima Daudjee, Khaled Ammar, M. Tamer zsu, Xingfang Wang, and Tianqi Jin. An Experimental Comparison of Pregel-like Graph Processing Systems. In VLDB'14, Volume 7 Issue 12, Pages 1047-1058, August 2014.

[5] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One Trillion Edges: Graph Processing at Facebook-Scale. In VLDB'15 Endowment, Volume 8 Issue 12, Pages 1804-1815, August 2015.

[6] Yingyi Bu, Vinayak Borkar, Jianfeng Jia, Michael J. Carey, and Tyson Condie. Pregelix: Big(ger) Graph Analytics on A Dataow Engine. In Proceeding of VLDB Endowment, Volume 8 Issue 2, Pages 161-172, October 2014.

[7] Ahmed, N.K., Du_eld, N.G., Neville, J., and Kompella, R.R. Graph Sample and Hold: A Framework for Big-Graph Analytics. In KDD'14, Pages 1446-1455, 2014

[8] Zuhair Khayyat, Karim Awara, Amani Alonazi, Hani Jamjoom, Dan Williams, and Panos Kalnis. Mizan: A System for Dynamic Load Balancing in Large-scale Graph Processing. In EuroSys'13, Pages 169-182, April, 2013.

[9] Semih Salihoglu and Jennifer Widom. GPS: A Graph Processing System. In SSDBM'13, Article 22, 2013. DOI=http://dx.doi.org/10.1145/2484838.2484843.

[10] Yonathan Perez, Rok Sosi, Arijit Banerjee, Rohan Puttagunta, Martin Raison, Pararth Shah, and Jure Leskovec. Ringo: Interactive Graph Analytics on Big-Memory Machines. In SIGMOD'15, Pages 1105-1110, May 31-June 4, 2015.

[11] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. In VLDB'12, Volume 5 Issue 8, Pages 716-727, April 2012.

[12] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In OSDI'12, Pages 17-30, 08 October 2012.

[13] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. GraphChi: Large-Scale Graph Computation on Just a PC. In OSDI'12, Pages 31-46, 08 October 2012.

[14] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. GraphX: Graph Processing in a Distributed Dataow Framework. In OSDI'14, Pages 599-613, 06 October 2014.

[15] Wook-Shin Han, Sangyeon Lee, Kyungyeol Park, Jeong-Hoon Lee, Min-Soo Kim, Jinha Kim, and Hwanjo Yu. TurboGraph: A Fast Parallel Graph Engine Handling Billion-scale Graphs in a Single PC. In KDD'13, Pages 77-85, 11 August 2013.

[16] A. Roy, L. Bindschaedler, J. Malicevic, and W. Zwaenepoel. Chaos: Scale-out Graph Processing from Secondary Storage. In SOSP'15, Pages 410-424, 2015.

[17] Amitabha Roy, Ivo Mihailovic, and Willy Zwaenepoel. X-Stream: Edge-centric Graph Processing using Streaming Partitions. In SOSP'13, Pages 472-488, 2013.

[18] Xiaowei Zhu, Wentao Han, and Wenguang Chen. 2015. GridGraph: Large-Scale Graph Processing on a Single Machine Using 2-Level Hierarchical Partitioning. In USENIX ATC'15, Pages 375-386, 2015.

[19] Da Zheng, Disa Mhembere, Randal Burns, Joshua Vogelstein, Carey E. Priebe, and Alexander S. Szalay. FlashGraph: Processing Billion-Node Graphs on an Arrayof Commodity SSDs. In FAST'15, Pages 45-58, 2015.

[20] L Page, S Brin, R Motwani, and T Winograd. The PageRank Citation Ranking: Bringing Order to the Web. In Stanford InfoLab, 1999.