Big Data Research ••• (••••) •••–•••

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Big Graph Mining: Frameworks and Techniques

Sabeur Aridhi [a,*], Engelbert Mephu Nguifo [b,c]

[a] Aalto University, School of Science, P.O. Box 12200, FI-00076, Finland
[b] Clermont University, Blaise Pascal University, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France
[c] CNRS, UMR 6158, LIMOS, F-63173 Aubiere, France

ARTICLE INFO

Article history:
Received 6 January 2016
Received in revised form 12 June 2016
Accepted 24 July 2016
Available online xxxx

Keywords:
Big graphs, data mining
Pattern mining
Graph processing frameworks

ABSTRACT

Big graph mining is an important research area and it has attracted considerable attention. It allows to process, analyze, and extract meaningful information from large amounts of graph data. Big graph mining has been highly motivated not only by the tremendously increasing size of graphs but also by its huge number of applications. Such applications include bioinformatics, chemoinformatics and social networks. One of the most challenging tasks in big graph mining is pattern mining in big graphs. This task consists on using data mining algorithms to discover interesting, unexpected and useful patterns in large amounts of graph data. It aims also to provide deeper understanding of graph data. In this context, several graph processing frameworks and scaling data mining/pattern mining techniques have been proposed to deal with very big graphs. This paper gives an overview of existing data mining and graph processing frameworks that deal with very big graphs. Then it presents a survey of current researches in the field of data mining/pattern mining in big graphs and discusses the main research issues related to this field. It also gives a categorization of both distributed data mining and machine learning techniques, graph processing frameworks and large scale pattern mining approaches.

1. Introduction

Over the last decade, big graph mining has attracted considerable attention. This field has been highly motivated, not only by the increasing size of graph data, but also by its huge number of applications. Such applications include the analysis of social networks [1,2], Web graphs [3], as well as spatial networks [4]. It has emerged as a hot topic that consists on the deliver of deeper understanding of the graph data. Frequent pattern mining is a main task in this context and it has attracted much interest. Several algorithms exist for frequent pattern mining. However, they are mainly used on centralized computing systems and evaluated on relatively small databases [5]. Yet, modern graphs are growing dramatically which makes the above cited approaches face the scalability issue. Consequently, several parallel and distributed solutions have been proposed to solve this problem [6–11]. In addition to that, many distributed frameworks have been used to deal with the existing deluge of data. These distributed frameworks abstract away most of the challenges of building a distributed system and offer simple programming models for data analysis [12]. Most of them are quite simple, easy to use and able to cope with potentially unlimited datasets.

In this paper, we first study existing works on the field of big data analytics. Thus, we present a survey on distributed data mining and machine learning approaches. Then, we study existing graph processing frameworks and we highlight pattern mining solutions in big graphs. With reference to the literature we can identify many different types of distributed graph mining techniques, with respect to the format of the input data, to produce many different kinds of patterns. We also give a categorization of both techniques for big data analytics, graph processing frameworks and large scale pattern mining approaches. Techniques for big data analytics are described according to their related programming model and the supported programming language. Graph processing frameworks are described according to their related programming model, the type of resources used by each framework and whether the framework allows asynchronous execution or not. Pattern mining approaches are described according to the input, the output of each approach and the used programming model.

The remainder of the paper is organized as follows. In the following section, we present existing works on big data analytics. In Section 3, we present an overview of graph processing frameworks and graph processing related approaches. Specifically, we present

* Corresponding author.
E-mail addresses: sabeur.aridhi@gmail.com (S. Aridhi), mephu@isima.fr (E. Mephu Nguifo).

http://dx.doi.org/10.1016/j.bdr.2016.07.002
2214-5796/

works that deal with pattern mining techniques in big graphs. Finally, we discuss the presented approaches in Section 4.

2. Big data analytics

In this section, we review related works on MapReduce and distributed data mining and machine learning techniques in the context of Big Data.

2.1. MapReduce

MapReduce [13] is a framework for processing highly distributable problems across huge datasets using a large number of computers. It was developed within Google as a mechanism for processing large amounts of raw data, for example, crawled documents or web request logs. This data is so large, it must be distributed across thousands of machines in order to be processed in a reasonable amount of time. This distribution implies parallel computing since the same computations are performed on each CPU, but with a different dataset. MapReduce is an abstraction that allows to perform simple computations while hiding the details of parallelization, data distribution, load balancing and fault tolerance. The central features of the MapReduce framework are two functions, written by a user: Map and Reduce. The Map function takes as input a pair and produces a set of intermediate key-value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function. The Reduce function accepts an intermediate key and a set of values for that key. It merges these values together to form a possible smaller set of values.

Hadoop is the open-source implementation of MapReduce. It is composed of two components. The first component is the Hadoop Distributed File System (HDFS) for data storage. The second one is the wide spread MapReduce programming paradigm [13]. Hadoop provides a transparent framework for both reliability and data transfers. It is the cornerstone of numerous systems which define a whole ecosystem around it. This ecosystem consists of several packages that runs on top of Hadoop including:

- PIG [14], a high level language for Hadoop;
- HBase [15], a column-oriented data storage on top of Hadoop;
- Hive [16], a framework for querying and managing large datasets residing in distributed storage using a SQL-like language called HiveQL;
- Spark [17], a powerful general purpose processing framework that provides an ease of use tool for efficient analytics of heterogeneous data;
- Storm [18], an open source framework for processing large structured and unstructured data in real time;
- Flink [19], an open source framework for processing data in both real time mode and batch mode;
- H2O [20], a system that brings database-like interactiveness to Hadoop.

2.2. Distributed machine learning and data mining techniques

Data mining and machine learning hold a vast scope of using the various aspects of Big Data technologies for scaling existing algorithms and solving some of the related challenges [21]. In the following, we present existing works on distributed machine learning and data mining techniques.

2.2.1. NIMBLE

NIMBLE [22] is a portable infrastructure that has been specifically designed to enable the implementation of parallel machine learning (ML) and data mining (DM) algorithms. The NIMBLE approach allows to compose parallel ML-DM algorithms using reusable (serial and parallel) building blocks that can be efficiently executed using MapReduce and other parallel programming models. The programming abstractions of NIMBLE have been designed with the intention of parallelizing ML-DM computations and allow users to specify data parallel, iterative, task parallel, and even pipelined computations. The NIMBLE approach has been used to implement some popular data mining algorithms such as k-Means Clustering and Pattern Growth-based Frequent Itemset Mining, k-Nearest Neighbors, Random Decision Trees, and RBRP-based Outlier Detection algorithm. As shown in Fig. 1, NIMBLE is organized into four distinct layers:

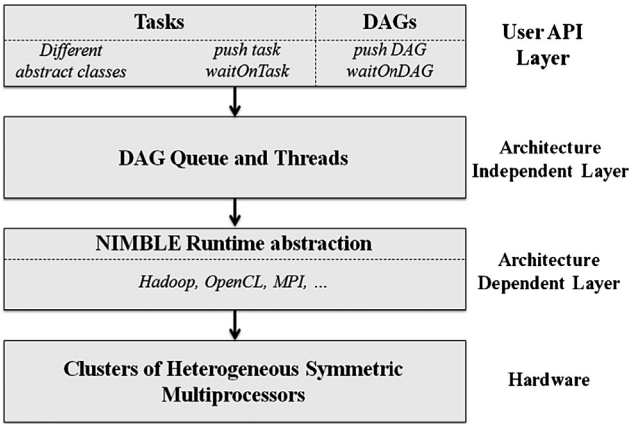

Fig. 1. An overview of the software architecture of NIMBLE.

1. The user API layer, which provides the programming interface to the users. Within this layer, users are able to design tasks and Directed Acyclic Graphs (DAGs) of tasks to express dependencies between tasks. A task processes one or more datasets in parallel and produces one or more datasets as output.
2. The architecture independent layer, which acts as the middleware between the user specified tasks/DAGs, and the underlying architecture dependent layer. This layer is responsible for the scheduling of tasks, and delivering the results to the users.
3. The architecture dependent layer, which consists of harnesses that allow NIMBLE to run portably on various platforms. Currently, NIMBLE only supports execution on the Hadoop framework.
4. The hardware layer, which consists of the used cluster.

2.2.2. SystemML

SystemML [23] is a system that enables the development of large scale machine learning algorithms. It first expresses a machine learning algorithm in a higher-level language called Declarative Machine learning Language (DML). Then, it executes the algorithm in a MapReduce environment. This DML language exposes arithmetical and linear algebra primitives on matrices that are natural to express a large class of machine learning algorithms. As shown in Fig. 2, SystemML is organized into four distinct layers:

- The Language component: It consists of user-defined algorithms written in DML.
- The High-Level Operator Component (HOP): It analyzes all the operations within a statement block and chooses from multiple high-level execution plans. A plan is represented in a DAG of basic operations (called hops) over matrices and scalars.
- The Low-Level Operator Component (LOP): It translates the high-level execution plans provided by the HOP component into low-level physical plans on MapReduce.

Please cite this article in press as: S. Aridhi, E. Mephu Nguifo, Big Graph Mining: Frameworks and Techniques, Big Data Res. (2016), http://dx.doi.org/10.1016/j.bdr.2016.07.002