Big Data Research ••• (••••) •••–•••

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Frequent Itemsets Mining for Big Data: A Comparative Analysis

Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Paolo Garza, Fabio Pulvirenti *, Luca Venturini

Politecnico di Torino, Dipartimento Automatica e Informatica, Torino, Italy

ARTICLE INFO

Article history:
Received 22 March 2016
Received in revised form 29 May 2017
Accepted 4 June 2017
Available online xxxx

Keywords:
Big Data
Frequent itemset mining
Hadoop and Spark platforms

ABSTRACT

Itemset mining is a well-known exploratory data mining technique used to discover interesting correlations hidden in a data collection. Since it supports different targeted analyses, it is profitably exploited in a wide range of different domains, ranging from NETWORK traffic data to medical records. With the increasing amount of generated data, different scalable algorithms have been developed, exploiting the advantages of distributed computing frameworks, such as Apache Hadoop and Spark.
This paper reviews Hadoop- and Spark-based scalable algorithms addressing the frequent itemset mining problem in the Big Data domain through both theoretical and experimental comparative analyses. Since the itemset mining task is computationally expensive, its distribution and parallelization strategies heavily affect memory usage, load balancing, and communication costs. A detailed discussion of the algorithmic choices of the distributed methods for frequent itemset mining is followed by an experimental analysis comparing the performance of state-of-the-art distributed implementations on both synthetic and real datasets. The strengths and weaknesses of the algorithms are thoroughly discussed with respect to the dataset features (e.g., data distribution, average transaction length, number of records), and specific parameter settings. Finally, based on theoretical and experimental analyses, open research directions for the parallelization of the itemset mining problem are presented.

1. Introduction

In recent years, the increasing availability of huge amounts of data has changed the importance of data analytic systems for Big Data and the interest towards data mining, an important set of techniques useful to extract effective and usable knowledge from data. On the one hand, the Big Data analytics scenario is very challenging for researchers. Indeed, the application of traditional data mining techniques to big volumes of data is not straightforward and some of the most popular techniques had to be redesigned from scratch to fit the new environment. On the other hand, companies are interested in the strategic benefits that Big Data could deliver. Data mining, together with machine learning [1], is the main research area on which Big Data analytics rely. It includes (i) clustering algorithms to discover hidden structures in unlabeled data [2], (ii) frequent itemsets mining and association rule mining techniques to discover interesting correlations and dependencies [3], and (iii) supervised algorithms to infer models from labeled datasets and use them to predict the label of new data [4].

Several traditional centralized mining algorithms have been proposed. They are very efficient when the datasets can be completely loaded in main memory. However, they cannot cope with Big Data, because they are not designed for a parallel and distributed environment. The recent shift towards horizontal scalability has highlighted the need of distributed/parallelized data mining algorithms able to exploit the available hardware resources and distributed Big Data frameworks (e.g., Apache Hadoop [5], Apache Spark [6]). In this survey, we focus on distributed/parallel itemset mining algorithms in the Big Data context because they represent exploratory approaches widely used to discover frequent co-occurrences from the data. These algorithms have been widely exploited in different application domains (e.g., network traffic data [7], healthcare [8], biological data [9], energy data [10], images [11], open linked data [12], document and data summarization [13–15]).

The parallelization of the frequent itemset mining problem in a distributed environment by means of the MapReduce programming paradigm and a Big Data framework is not an easy task. The main challenge is devising a smart partitioning of the problem in independent subproblems, each one based on a subset of the data, to exploit the computation power of a cluster of servers

* Corresponding author.
E-mail addresses: daniele.apiletti@polito.it (D. Apiletti), elena.baralis@polito.it (E. Baralis), tania.cerquitelli@polito.it (T. Cerquitelli), paolo.garza@polito.it (P. Garza), fabio.pulvirenti@polito.it (F. Pulvirenti), luca.venturini@polito.it (L. Venturini).

http://dx.doi.org/10.1016/j.bdr.2017.06.006
2214-5796/

in parallel. In the following, we will describe how this problem has been addressed so far and which are pros and cons of the current MapReduce- and RDD-based parallel algorithms by taking into consideration load balancing and communication costs, which are two very important issues in the distributed domain. They are strictly related to the adopted parallelization strategy and usually represent the main bottlenecks of parallel algorithms.

The contributions of this survey are the followings.

- A theoretical analysis of the algorithmic choices that have been proposed to address the itemset mining problem in the Big Data context by means of MapReduce, with the analysis of their expected impact on main memory usage, load balancing, and communication costs.
- An extensive evaluation campaign to assess the reliability of our expectations. Precisely, we ran more than 300 experiments on 14 synthetic datasets and 2 real datasets to evaluate the execution time, load balancing, and communication costs of five state-of-the-art parallel itemset mining implementations.
- The identification of strengths and weaknesses of the algorithms with respect to the input dataset features (e.g., data distribution, average transaction length, number of records), and specific parameter settings.
- The discussion of promising open research directions for the parallelization of the itemset mining problem.

This paper is organized as follow. Section 2 briefly introduces the Hadoop and Spark frameworks, while Section 3 introduces the background about the itemset mining problem, providing the main definitions and a brief description of the state-of-the-art centralized itemset mining algorithms. Section 4 describes the algorithmic strategies adopted so far to partition and parallelize the frequent itemset mining problem by means of the MapReduce paradigm, while Section 5 describes the state-of-the-art distributed algorithms and their implementations. In Section 6 we benchmark the selected algorithms with a large set of experiments on both real and synthetic datasets. Section 7 summarizes the concrete and practical lessons learned from our evaluation analysis, while Section 8 discusses the open issues raised by the experimental validation of the theoretical analysis, highlighting some possible research directions to support a more effective and efficient data mining process on Big Data collections.

2. Apache Hadoop and Spark

The availability of increasing amounts of data has highlighted the need of distributed algorithms able to scale horizontally. To support the design and implementation of these algorithms, the MapReduce [16] programming paradigm and the Apache Hadoop [5] distributed platform have been commonly used in the last decade. In the last couple of years, instead, Apache Spark [6] has become the favorite distributed platform for large data analytics, outperforming Hadoop thanks to its distributed dataset abstraction.

The success of Hadoop and Spark is mainly due to their data locality paradigm. The basic idea consists in processing data in the same node storing it instead of sending large amounts of data on the network.

Hadoop and Spark support the MapReduce paradigm, a distributed programming model introduced by Google [16]. A MapReduce application consists of two main phases, named map and reduce. The map phase applies a map function on the input data and, after processing them, it emits a set of key-value pairs. To parallelize the execution of the map phase, each node of the cluster applies the map function in isolation on a disjoint subset of the input data. Then, the map results are exchanged among the cluster nodes and the reduce phase is run. Specifically, the reduce phase considers one unique key at a time and iterates through the values that are associated with that key to emit the final results. Also the reduce phase can be parallelized by assigning to each node a subset of keys.

MapReduce-based programs implemented on Hadoop do not fit well iterative processes because each iteration requires a new reading phase from disk. This feature is critical when dealing with huge datasets. This issue motivated the improvements introduced by Spark, which enables the nodes of the cluster to cache data and intermediate results in memory, instead of reloading them from the disk at each iteration. This goal is achieved through the introduction of the Resilient Distributed Dataset (RDD) data structure, which is a read-only partitioned collection of records distributed across the nodes of the cluster. An RDD, when it is reused multiple times, is cached in the main memory of the nodes to avoid the overhead given by multiple reads from disk.

2.1. Hadoop and Spark data mining and machine learning libraries

In recent years the success of Hadoop and Spark was supported by the introduction of open source data mining and machine learning libraries. Mahout [17] for Hadoop has been one of the most popular collection of Machine Learning algorithms, providing distributed implementations of well-known clustering, classification, and itemset mining algorithms. All the current implementations are based on MapReduce. MADlib [18], instead, provides a SQL toolkit of algorithms that run over Hadoop. Finally, MLLib [19] is the Machine Learning and data mining library developed on Spark. MLlib allows researchers to exploit Spark special features to implement all those applications that can benefit from them, e.g. faster iterative procedures.

2.2. Distributed data mining approaches based on MPI and GPUs

Hadoop and Spark are not the only frameworks supporting the parallelization of data mining algorithms and their distributed execution. Specifically, the distributed execution of the data mining algorithms has been addressed also by using solutions based on Message Passing Interface (MPI) [20], one of the most adopted framework in academic environment, or more recent hardware components, such as GPUs.

For instance, the solutions proposed in [21–26] are MPI-based solutions for the itemset mining problem, whereas solutions like [27–29] take advantage of GPU-based commodity cluster. A comparative analysis of the GPU-based solutions is reported in [30].

The focus of this work is the comparison of the MapReduce-based approaches. Hadoop and Spark have been widely adopted in the research environment [31–33]. The reasons are partly related to the easier data management and better fault tolerance [34,26] but, above all, these frameworks allow the development of parallel algorithms by unexperienced users [31].

3. Frequent itemset mining

A frequent itemset represents frequently co-occurring items in a transactional dataset. More formally, let $\mathcal{I}$ be a set of items. A transactional dataset $\mathcal{D}$ consists of a set of transactions $\{t_1, \ldots, t_n\}$. Each transaction $t_i \in \mathcal{D}$ is a collection of items (i.e., $t_i \subseteq \mathcal{I}$) and is identified by a transaction identifier (tid_i). Fig. 1(a) reports an example of a transactional dataset with 4 transactions.

An itemset I is defined as a set of items (i.e., $I \subseteq \mathcal{I}$) and is characterized by a support value, which is denoted by $sup(I)$ and defined as the ratio between the number of transactions in $\mathcal{D}$ containing I and the total number of transactions in $\mathcal{D}$. In the example dataset in Fig. 1(a), for example, the support of the itemset $\{a, c, d\}$

Please cite this article in press as: D. Apiletti et al., Frequent Itemsets Mining for Big Data: A Comparative Analysis, Big Data Res. (2017), http://dx.doi.org/10.1016/j.bdr.2017.06.006