SEARUM: a cloud-based SErvice for Association RUle Mining

Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Silvia Chiusano, Luigi Grimaudo
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy
Email: {daniele.apiletti,elena.baralis,tania.cerquitelli,silvia.chiusano,luigi.grimaudo}@polito.it

***Abstract*—Large volumes of data are being produced by various modern applications at an ever increasing rate. These applications range from wireless sensors networks to social networks. The automatic analysis of such huge data volume is a challenging task since a large amount of interesting knowledge can be extracted. Association rule mining is an exploratory data analysis method able to discover interesting and hidden correlations among data. Since this data mining process is characterized by computationally intensive tasks, efficient distributed approaches are needed to increase its scalability. This paper proposes a novel cloud-based service, named SEARUM, to efficiently mine association rules on a distributed computing model. SEARUM consists of a series of distributed MapReduce jobs run in the cloud. Each job performs a different step in the association rule mining process. As a case study, the proposed approach has been applied to the network data scenario. The experimental validation, performed on two real network datasets, shows the effectiveness and the efficiency of SEARUM in mining association rules on a distributed computing model.**

***Keywords*-association rule mining; distributed computing model; cloud-based service; network data analysis.**

I. INTRODUCTION

The capability of modern applications (e.g., social networks, computer networks, wireless sensor applications) of generating and collecting data has been rapidly increasing. Since data mining focuses on studying effective and efficient algorithms to transform huge amounts of data into useful and actionable knowledge, the interest in data mining is continuously growing both in the industrial and research domains. Industries are attracted by the business opportunities arising from the exploitation of the extracted knowledge, while researchers are interested in the challenging issues coming from the application of data mining techniques to new scenarios. Different data analytics tools rely on data mining algorithms to gain interesting insights from large volumes of semi-structured or unstructured data. Data mining techniques allow extracting previously unknown interesting patterns such as groups of data objects (cluster analysis), unusual objects (anomaly detection) and dependencies (association rule mining) [1].

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane")

When dealing with huge data collections like those of "Big Data", the computational cost of the data mining process (and in some cases the feasibility of the process itself) can potentially become a critical bottleneck in data analysis. For example, association rule mining algorithms, which find application in a wide range of different domains including medical images [2], biological data [3], and network traffic data [4], are characterized by computationally intensive tasks. Thus, parallel and distributed approaches can be adopted to increase the mining efficiency and improve algorithm scalability. The current trend explored by cloud providers, such as Windows Azure, is offering an increasingly heterogeneous portfolio of online services in the cloud, spanning traditional IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) as well as business analytics-oriented cloud-based tools.

Association rule mining is a two-step process: (i) Frequent itemset extraction and (ii) association rule generation from frequent itemsets [5]. Since the first phase represents the most computationally intensive knowledge extraction task, effective solutions have been widely investigated to parallelize the itemset mining process both on multi-core processors [6], [7], [8], [9] and with a distributed architecture [10], [11], [12], [13]. However, when a large set of frequent itemsets is extracted, the generation of association rules from this set becomes a critical task.

The challenge of our work is to design and develop an association rule mining framework as a SaaS (Software-as-a-Service) service model, to offer a data analytics service to cloud users. To our knowledge, the association rule generation from frequent itemsets on a distributed architecture has not been investigated yet, and represents a core contribution of this work. More specifically, this paper presents a cloud-based service, named SEARUM (a cloud-based SErvice for Association RUle Mining), to efficiently mine association rules on a distributed computing model. SEARUM consists of a series of distributed MapReduce jobs run in the cloud. Each job performs a different step in the association rule mining process.

As a reference case study, the proposed approach has been applied to two real network datasets. The analysis of the large amount of Internet traffic data is an important

task since a huge amount of interesting knowledge can be automatically mined to effectively support both service providers and Internet applications. To profile network communications, we analyzed traffic metrics and statistical measurements computed on traffic flows. The results showed the effectiveness and efficiency of the SEARUM architecture in mining interesting patterns on a distributed computing model.

The paper is organized as follows. Section II describes previous works. Section III describes the problem statement addressed in this paper. Section IV presents the SEARUM architecture while a preliminary evaluation of our approach is reported in Section V. Finally, Section VI draws conclusions and discusses future extensions of the proposed approach.

II. Related work

The mining task in association rule extraction entails two subtasks, the frequent itemset generation and the subsequent association rule extraction. The former subtask is more computationally intensive than the latter [5], and it has been paid more attention by the research community, leading to the definition of efficient algorithms.

Initial parallel and distributed itemset extraction stemmed from algorithms based on the Apriori approach [6]. Further improvements include FP-Tree [14], which exploits prefix-tree-like structures, and a multi-tree approach proposed in [7]. Parallel processing flows perform a sequence of three steps: firstly a horizontal subset of the data is analyzed, then a local FP-Tree is built, finally the mining process is carried out on the local FP-Tree. The candidate pattern bases from different processing flows are then merged together. The good efficiency and scalability of this approach are counterbalanced by large memory requirements and node traversal redundancy with a high number of parallel processing flows. [10] proposes an enhanced version of the merging algorithm specifically addressing the cluster computing environment. [15] discusses a tree partition approach based on a single tree. In the context of very large datasets, different constraints have been proposed in the parallel itemset extraction algorithm by [11], leading to good scalability on a 32-processor cluster. Better hardware resource exploitation and improvement of the itemset extraction on multiple-core processors have been addressed by [8], [9]. A path tiling technique has been devised to enhance the FP-Tree algorithm in terms of performance, by improving the temporal locality of data accesses at different memory levels. Cache hint optimization was firstly addressed by this technique in [9], which then found application also in the parallel itemset mining context [8].

Large-scale mining based on the MapReduce paradigm [16] is based on algorithms that distribute data and computation across a distributed architecture [12], [13]. A parallel FP-Growth algorithm has benn proposed in [12]. The algorithm, named PFP, aimed at supporting efficient query recommendation for search engines. PFP consists of two separate MapReduce jobs (i.e., no computational dependencies exists between them) and achieves an almost linear speedup. [13] proposes some improvements with respect to [12], particularly addressing efficient load balance strategies, thus achieving higher speedup and better performance.

III. Problem statement

Let $\mathcal{D}$ be a dataset whose a generic record r is a set of features. Each feature, also called *item*, is a couple *(attribute, value)*. Since we are interested in analyzing statistical features computed on traffic flows, each feature models a measurement describing the network flow (e.g., Round-Trip-Time (RTT), number of hops).

An *itemset* is a set of features. The *support count* of an itemsets I is the number of records containing I. The *support* $s(I)$ of an itemset I is the percentage of records containing I. An itemset is *frequent* when its support is greater than, or equal to, a minimum support threshold $MinSup$. *Association rules* identify collections of itemsets (i.e., set of features) that are statistically related (i.e., frequent) in the underlying dataset. Association rules are usually represented in the form $X \rightarrow Y$, where X (also called rule antecedent) and Y (also called rule consequent) are disjoint itemsets (i.e., disjoint conjunctions of features). Rule quality is usually measured by rule support and confidence. *Rule support* is the percentage of records containing both X and Y. It represents the prior probability of $X \cup Y$ (i.e., its observed frequency) in the dataset. *Rule confidence* is the conditional probability of finding Y given X. It describes the strength of the implication and is given by $\mathrm{c}(X \rightarrow Y) = \frac{s(X \cup Y)}{s(X)}$ [1].

Given a dataset $\mathcal{D}$, a support threshold $MinSup$, and a confidence threshold $MinConf$, the mining process discovers all association rules with support and confidence greater than, or equal to, $MinSup$ and $MinConf$, respectively.

Furthermore, to rank the most interesting rules, we used the lift index [1], which measures the (symmetric) correlation between antecedent and consequent of the extracted rules. The lift of an association rule $X \rightarrow Y$ is defined as [1]

$$\mathrm{lift(X, Y)} = \frac{\mathrm{c}(X \rightarrow Y)}{\mathrm{s(Y)}} = \frac{\mathrm{s}(X \rightarrow Y)}{\mathrm{s(X)s(Y)}} \qquad (1)$$

where $\mathrm{s}(X \rightarrow Y)$ and $\mathrm{c}(X \rightarrow Y)$ are respectively the rule support and confidence, and $\mathrm{s}(X)$ and $\mathrm{s}(Y)$ are the supports of the rule antecedent and consequent. If lift(X,Y)=1, itemsets *X* and *Y* are not correlated, i.e., they are statistically independent. Lift values below 1 show a negative correlation between itemsets X and Y, while values above 1 indicate a positive correlation. The interest of rules having a lift value close to 1 may be marginal. In this work the mined rules are ranked according to their lift value to focus on the subset of most (positively or negatively) correlated rules.

IV. The SEARUM architecture

SEARUM consists of a series of distributed jobs run in the cloud. Each job receives as input the result of one or more preceding jobs and performs one of the steps required for association rule mining. Currently, each job is performed by one or more MapReduce tasks run on a Hadoop cluster.

The SEARUM architecture contains the following jobs, described in details in the subsequent sections:

- Network measurement acquisition
- Data pre-processing
- Item frequency computation
- Itemset mining
- Rule extraction
- Rule aggregation and sorting

Since our case study is based on network traffic analysis, we thoroughly describe the SEARUM architecture in this application scenario. Furthermore, in the current design, SEARUM addresses a specific class of association rules, whose consequent consists of a single item.

A. Network measurement acquisition

The first step to analyse network traffic is is collecting network measurements. To this aim, a passive probe is located on the access link (vantage point) that connects an edge network to the Internet. The passive probe sniffs all incoming and outgoing packets flowing on the link, i.e., packets directed to a node inside the network and generated by a node in the Internet, and vice versa. The probe runs Tstat [17], [18], a passive monitoring tool allowing network and transport layer measurement collection. Tstat rebuilds each TCP connection by matching incoming and outgoing segments. Thus, a flow-level analysis can be performed [18]. A TCP flow is identified by snooping the signaling flags (SYN, FIN, RST). The status of the TCP sender is rebuilt by matching sequence numbers on data segments with the corresponding acknowledgement (ACK) numbers.

To evaluate the SEARUM cloud-based service in a real-world application, we focus on a subset of measurements describing the traffic flow among the many provided by Tstat. The most meaningful features, selected with the support of domain experts, are detailed in the following:

- the *Round-Trip-Time* (RTT) observed on a TCP flow, i.e., the minimum time lag between the observation of a TCP segment and the observation of the corresponding ACK. RTT is strongly related to the distance between the two nodes
- the *number of hops* (Hop) from the remote node to the vantage point observed on packets belonging to the TCP flow, as computed by reconstructing the IP Time-To-Live[1]
- the *flow reordering probability* ($P\{reord\}$), which can be useful to distinguish different paths
- the *flow duplicate probability* ($P\{dup\}$), that can highlight a destination served by multiple paths[2]
- the total *number of packets* ($NumPkt$), the total *number of data packets* ($DataPkt$), and the total *number of bytes* ($DataBytes$) sent from both the client and the server, separately (the client is the host starting the TCP flow)
- the minimum ($WinMin$), maximum ($WinMax$), and scale ($WinScale$) values of the *TCP congestion window* for both the client and the server, separately
- the *TCP port* of the server ($Port$)
- the *class of service* ($Class$), as defined by Tstat, e.g., HTTP, video, VoIP, SMTP, etc.

Based on measurements listed above, an input data record is defined by the following features: RTT, Hop, $P\{reord\}$, $P\{dup\}$, $NumPkt$, $DataPkt$, $DataBytes$, $WinMax$, $WinMin$, $WinScale$, $Port$, $Class$. To obtain reliable estimates on reordering and duplicate probabilities, only TCP flows which last more than $P = 10$ packets are considered. This choice allow focusing the analysis on long-lived flows, where the network path has a more relevant impact, thus providing more valuable information.

B. Data pre-processing

This step performs the following two activities:

- Value discretization
- Transactional format conversion

Associaton rule mining requires a transactional dataset of categorical values. The discretization step converts continuously valued measurements into categorical bins. Then, data are converted from the tabular to the transactional format. An example is reported in Table I.

Automatic discretization approaches can exploit state-of-the-art techniques (e.g., clustering, statistical-based algorithms, etc.) to select appropriate bins depending on data distribution. These approaches yielded poorly significant bins on network data considered in this study. More specifically, the most frequent values were split into too many bins with respect to the real applicative interest. Hence, discretized bins are fixed-size and determined by domain experts based on the significance in the networking context. The fixed-size bins have been determined as follows:

- RTT: a bin each 5 ms for values from 0 ms to 200 ms, an additional bin for values higher than 200 ms.
- Hop: a bin for each value from 1 to 20, an additional bin for values exceeding 20.
- $P\{reord\}$: a bin each 0.1 from 0 to 1.
- $P\{dup\}$: a bin each 0.1 from 0 to 1.

[1]The initial TTL value is set by the source, typical values being 32, 64, 128 and 255.

[2]$P\{reord\}$ and $P\{dup\}$ are computed by observing the TCP sequence and acknowledgement numbers carried by segments of a given flow. We refer the reader to [18] for more details.

- $NumPkt$, $DataPkt$, and $DataBytes$: logarithmic bins, base 10, e.g., 5432 falls in the 3-4 bin since the value is between 10^3 and 10^4.
- $WinMax$ and $WinMin$: a bin for each multiple N of 4 Kb, where N is a power of 2, e.g., the bin 8-16 means that the TCP window is between 8 and 16 times 4 Kb.
- $WinScale$, $Port$, and $Class$: a bin for each value (no discretization).

Both the value discretization and the transactional format conversion are performed by a single map only job. Each record is processed by the map function and, if the number of packets is above the threshold (10 packets), the corresponding discretized transaction is emitted as a result of the mapping. This task entails an inherently parallel elaboration, considering that can be applied independently to each record.

	RTT	$NumPkt$	$P\{reord\}$
original	7	5432	0.88
discretized	5-10	3-4	0.9
transactional	RTT=5-10	$NumPkt$=3-4	$P\{reord\}$=0.9

Table I
PRE-PROCESSING EXAMPLE

C. Item frequency computation

A second job is exploited to compute the item frequency from the transactions emitted by the pre-processing phase. An example is reported in Tables II and III. Table II has three sample transactions that represent a possible output of the pre-processing phase. A map function is exploited to process each transaction: the map emits a (key, $value$) pair for each item in the transaction, where the key is the item itself (e.g., RTT=5-10), and the $value$ is its count, i.e., always 1. A reduce function is then executed to sum all the values for each key, hence computing the support count of each item. This is a typical group-by query performed as a distributed MapReduce job. As a running example, we will consider the sample result of this job reported in Table III, as obtained by the sample transactions in Table II.

transaction 1	RTT=5-10	$NumPkt$=3-4	Hop=10
transaction 2	RTT=5-10		Hop=11
transaction 3	RTT=5-10	$NumPkt$=3-4	
transaction 3	RTT=5-10	$NumPkt$=3-4	Hop=11

Table II
SAMPLE TRANSACTIONS

D. Itemset mining

A third job performs the itemset mining by exploiting the parallel FP-growth algorithm, as described in [12]. This step consists of multiple MapReduce tasks. From the sample items of Table III, a result of this job is reported in Table IV, where only itemsets with support higher than 50% have been extracted.

item	sup count	sup
RTT=5-10	4	100%
$NumPkt$=3-4	3	75%
Hop=10	1	25%
Hop=11	2	50%

Table III
SAMPLE ITEMS

ID	itemset			sup count	sup
1	RTT=5-10			4	100%
2	RTT=5-10	$NumPkt$=3-4		3	75%
3	RTT=5-10		Hop=11	2	50%
4		$NumPkt$=3-4		3	75%
5			Hop=11	2	50%

Table IV
SAMPLE ITEMSETS

E. Rule extraction

The rule extraction step, to the best of the authors knowledge, is a novel contribution as a distributed cloud-based service. It consists of a MapReduce job, as detailed in the following. For each itemset of length k (k-itemset), the map function emits:

- a (key,$value$) pair with
  - key: the k-itemset itself
  - $value$: the k-itemset support count
- for each $(k-1)$-itemset, a (key,$value$) pair with
  - key the $(k-1)$-itemset
  - $value$ the pair (k-itemset, support count of the k-itemset).

Then, the reduce function performs the actual rule extraction. Since each $(k-1)$-itemset emitted as key contains its k-itemset and the k-itemset support count as value, the missing item in the $(k-1)$-itemset with respect to the k-itemset is the rule consequent (head), whereas the $(k-1)$-itemset is the antecedent (rule body). The support count values of the k-itemset, the $(k-1)$-itemset and the consequent item are used to compute the support, confidence, and lift of the rule, as defined in Section III. Table V reports the rules extracted from the itemsets of the running example (see Table IV).

F. Rule aggregation and sorting

A final step is executed by means of a MapReduce job to sort and aggregate the rules according to the consequent and the quality measure. As discussed in Section III, we selected the lift as rule quality measure. Sorting and aggregating on the consequent helps in analyzing the extracted rules for finding significant correlations. A sample output based on our running example is reported in Table VI.

rule	sup count	sup	conf	lift
RTT=5-10 $\rightarrow$ $NumPkt$=3-4	3	75%	75%	0.75
$NumPkt$=3-4 $\rightarrow$ RTT=5-10	3	75%	100%	1.33
RTT=5-10 $\rightarrow$ Hop=11	2	50%	50%	0.50
Hop=11 $\rightarrow$ RTT=5-10	2	50%	100%	2.00

Table V
SAMPLE RULES

antecedent		consequent
Hop=11, $NumPkt$=3-4	$\rightarrow$	RTT=5-10
RTT=5-10	$\rightarrow$	$NumPkt$=3-4
RTT=5-10	$\rightarrow$	Hop=11

Table VI
SAMPLE RULES, SORTED AND AGGREGATED

V. EXPERIMENTAL VALIDATION

A set of preliminary experiments have been performed analyzing SEARUM behavior on real datasets. We assessed (i) the percentage of the overall mining time devoted to performing each job (Section V-A), (ii) the performance of the association rule mining (Section V-B), (iii) the network knowledge characterization (Section V-C) and (iv) the number of extracted association rules by varying the support and confidence thresholds (Section V-D),

SEARUM has been applied to two real datasets obtained by performing different capture stages on a backbone link of a nation-wide ISP in Italy that offers us three different vantage points. ISP vantage points expose traffic of three different Points-of-Presence (POP) in different cities in Italy; each PoP aggregates traffic from more than 10,000 ISP customers, which range from home users to Small Office Home Office (SOHO) accessing the Internet via ADSL or Fiber-To-The-Home technology. It represents therefore a very heterogeneous scenario. We will refer to each dataset as D1 or D2 as shown in Table VII, where the number of TCP flows and the size of each dataset are also reported.

Dataset	Number of TCP flows	Size [Gbyte]
D1	11,325,006	5.28
D2	413,012,989	192.56

Table VII
NETWORK TRAFFIC DATASETS

MapReduce jobs of the SEARUM workflow (see Section IV) have been developed in Java using the Hadoop Java API. Part of the code has been developed as an extension of the Apache Mahout project [19], which provides a limited implementation of the parallel itemset mining algorithm FP-Growth to mine the top-k closed itemsets [12].

Rule extraction, instead, is a novel contribution of the authors. Experiments have been performed on a cluster of 5 nodes running Cloudera's Distribution of Apache Hadoop (CDH4). Each cluster node is a 2.67 GHz six-core Intel(R) Xeon(R) X5650 machine with 32 Gbyte of main memory running Ubuntu 12.04 server with the 3.5.0-23-generic kernel. All reported execution times are real times, including both system and user time, obtained from the Cloudera Hadoop web administration control panel.

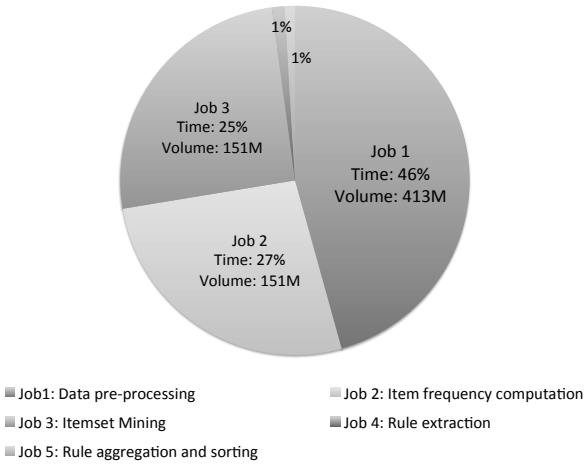


Figure 1. Dataset D2: Execution time distribution among jobs for MinSup=30% and MinConf=50%

A. Execution time distribution among jobs

Since SEARUM consists of a sequential workflow, we analyzed how much time is spent at each step. Figure 1 shows the percentage of the total execution time for each job of the SEARUM architecture.

The pre-processing job represents the most expensive step with almost 50% of the time. This behavior is due to the higher data volume to be processed at this stage with respect to the subsequent steps: pre-processing filters flows with less than 10 packets, thus reducing the data volume from 413 millions of records to 151 millions of transactions. Note that the pre-processing job is executed only once for

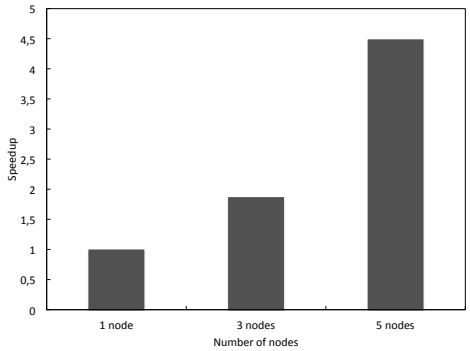
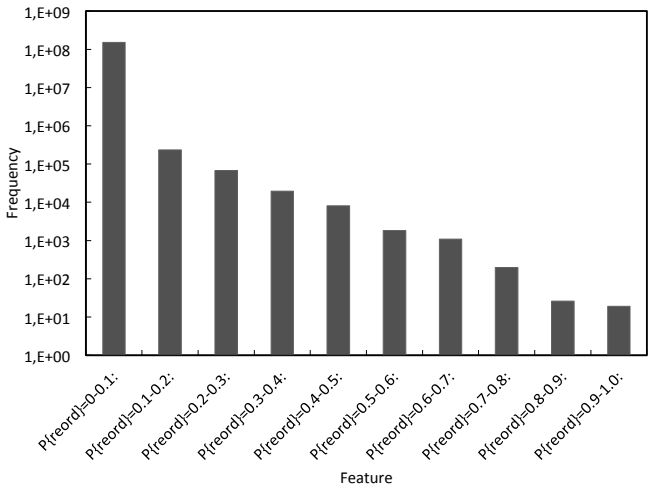
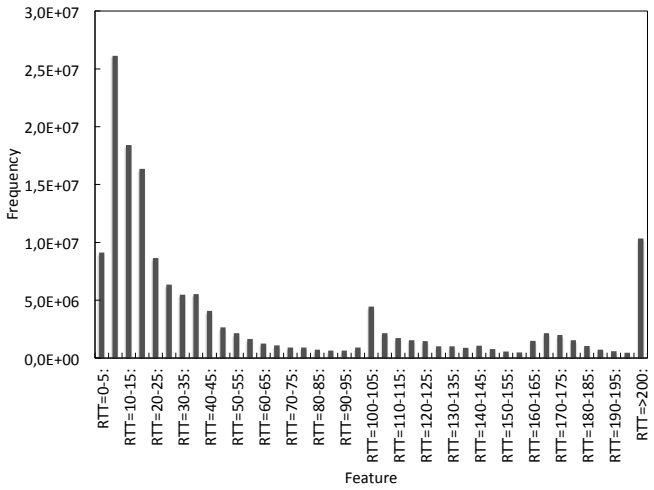


Figure 2. SEARUM speedup on D2 dataset



(a) Item distribution for the $P\{reord\}$ feature



(b) Item distribution for the RTT feature

Figure 3. Dataset D2

each discretization bin set, while it provides results that can be used many times by subsequent jobs, e.g., to mine itemsets and rules with different $MinSup$ and $MinConf$ constraints.

When the $MinSup$ value decreases, the computational complexity of the itemset mining job may significantly increase, since a very large number of itemsets may be generated. As a reference, Figure 1 reports times for MinSup=30% and MinConf=50% on dataset D2.

B. Evaluation of association rule mining

To evaluate the effectiveness of SEARUM in association rule mining, we measured the achieved speedup for different numbers of nodes. We considered 3 configurations: 1 node, 3 nodes, and 5 nodes. Dataset D2 has been used since it is the largest. Figure 2 reports the speedup when $MinSup = 30\%$ and $MinConf = 50\%$ are applied.

The first histogram in the figure (i.e., 1 node) corresponds to an execution of SEARUM on a single node. The speedup evaluation for increasing numbers of nodes is compared to the single-node performance.

The achieved speedup is the result of job distribution. The contribution of the former is especially relevant when considering large datasets and/or low minimum support thresholds ($MinSup$), which make the mining activity more computationally intensive. As reported in Figure 2 the SEARUM performance progressively improves when distributing the mining task on an increasing number of nodes. For instance, a 5-node cluster achieves a speedup of 4.5, showing that SEARUM has promising attitude to scale in larger clouds.

C. Network knowledge characterization

We evaluated the effectiveness of the proposed approach on real network traffic traces. In particular, we analyzed: (i) the usefulness of the extracted association rules in supporting the knowledge discovery process, and (ii) the item frequency distribution.

As example, the following two rules R1 and R2 are generated from dataset D1 and D2, respectively. Both rule have high confidence values and lift greater than 1 (rule support, confidence, and lift are reported in brackets after each rule).

$R_1 : \{Port = 80,\ P\{reord\} = 0 - 0.1,\ DataPkt = 1 - 2,\ DataBytes = 4 - 5\} \rightarrow Class = HTTP$ (0.313, 0.999, 1.765)

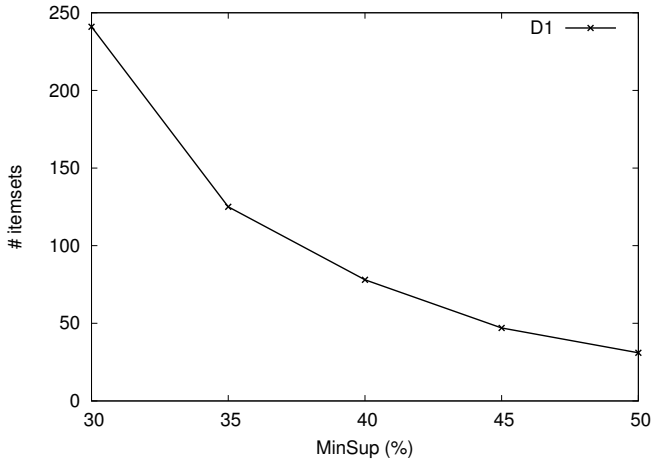
$R_2 : \{P\{dup\} = 0 - 0.1,\ NumPkt \leq 1,\ DataPkt \leq 1,\ Class = SSL\} \rightarrow Port = 443$ (0.013, 0.993, 4.944)

Based on rule R_1, the HTTP protocol is mainly used to transmit a set of TCP flows sent by the server through the TPC port 80. For these flows, the number of packets is in the range 10÷100 and a large number of bytes is transmitted (from 10,000 to 100,000). These flows can be generated when very large files are downloaded (e.g., YouTube videos).

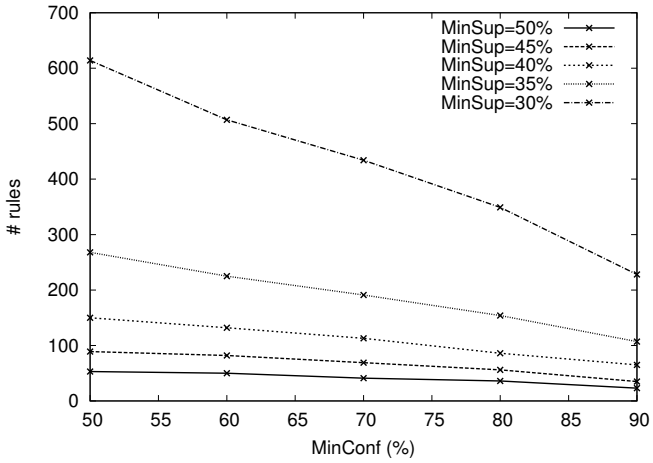
Rule R_2 reports that the TCP $Port$ 443 (HTTPS) is mainly used to transmit flows with SSL/TLS coded protocol and less than 10 packets. These flows can be generated when logging into websites through a secure connection (e.g., Facebook, Twitter).

We also analyzed the item frequency distribution to characterize the network activity. Figure 3 considers the Round-Trip-Time (RTT) and the flow reordering probability ($P\{reord\}$), which are discussed as representative features.

The item distribution for the $P\{reord\}$ feature is characterized by a very frequent item which models most TCP

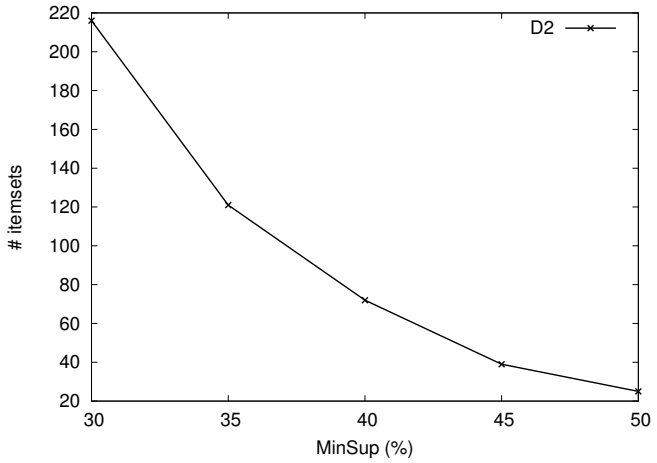


(a) Number of extracted itemsets for different $MinSup$ values

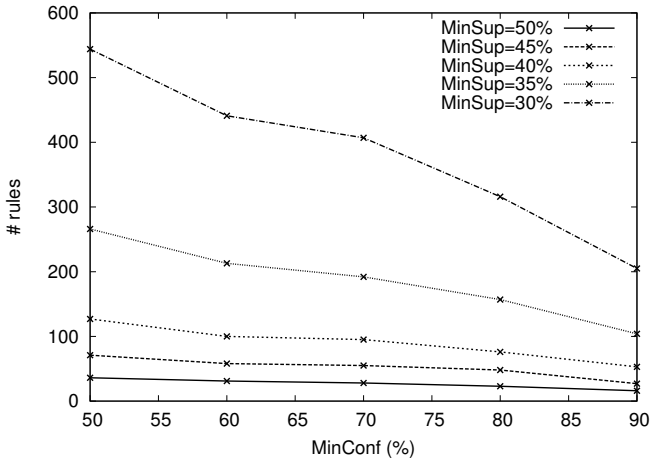


(b) Number of extracted rules for different values of $MinConf$ and $MinSup$

Figure 4. Dataset D1: Effect of $MinSup$ and $MinConf$ thresholds



(a) Number of extracted itemsets for different values of $MinSup$



(b) Number of extracted rules for different values of $MinConf$ and $MinSup$

Figure 5. Dataset D2: Effect of $MinSup$ and $MinConf$ thresholds

flows: they have a very low $P\{reord\}$, i.e., from 0 to 0.1. This data distribution analyzed over time and for different (sub)networks may be exploited to identify periods of time or (sub)networks that become less reliable or whose packets change path more frequently than usual.

The item distribution for the RTT, instead, shows four peaks:

- the first peak around 5-20 ms may represent local network traffic
- the second peak around 100 ms may represent external traffic inside the same ISP or in the same geographical zone (e.g., country, continent)
- the third peak around 170 ms may represent traffic towards long-distance destinations (e.g., other continents)
- finally, the last peak over 200 ms may represent network problems or unresponsive services

D. Effect of the support and confidence thresholds

Minimum support ($MinSup$) and confidence ($MinConf$) thresholds significantly affect the number of extracted itemsets and association rules.

When decreasing the $MinSup$ value, the number of frequent itemsets grows non linearly [5] and the complexity of the frequent itemset extraction task significantly increases. High $MinConf$ values represent a tighter constraint on rule selection. Consequently, when increasing $MinConf$ less rules are mined, but these rules tend to represent stronger correlations among data. High $MinConf$ values should be often combined with low $MinSup$ values to lead the extraction of peculiar (i.e., not very frequent) but highly correlated rules.

Figures 4(a) and 5(b) plot, for the two reference datasets, the number of extracted itemsets when varying $MinSup$. Figure 4(b) and 5(b) report the number of association rules for different $MinConf$ values.

VI. Conclusion

In this paper we presented our first implementation of a cloud-based service for association rule mining. Both the horizontal scalability of the approach and the meaningfulness of the extracted knowledge have been addressed.

Since preliminary experiments performed on two real datasets showed promising results, a more complete and optimized implementation of the proposed approach as a SaaS (Software-as-a-Service) service model may be envisioned in the long term to offer an efficient association rule mining algorithm to cloud users. In particular, future works will aim at optimizing the MapReduce workflow and expanding the workbench of the cluster architecture.

Acknowledgment

The authors would like to thank Dr. Marco Mellia for his advice and fruitful discussions.

References

[1] Pang-Ning T. and Steinbach M. and Kumar V., *Introduction to Data Mining*. Addison-Wesley, 2006.

[2] M. L. Antonie, O. R. Zaiane, and A. Coman, "Application of data mining techniques for medical image classification," *In MDM/KDD*, 2001.

[3] G. Cong, A. K. H. Tung, X. Xu, F. Pan, and J. Yang, "Farmer: finding interesting rule groups in microarray datasets," in *ACM SIGMOD '04*, 2004.

[4] M. Baldi, E. Baralis, and F. Risso, "Data mining techniques for effective and scalable traffic analysis," in *Integrated Network Management*, 2005, pp. 105–118.

[5] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB '94*, 1994, pp. 487–499.

[6] M. J. Zaki, "Parallel and distributed association mining: A survey," *IEEE Concurrency*, vol. 7, no. 4, pp. 14–25, Oct. 1999.

[7] O. R. Zaïane, M. El-Hajj, and P. Lu, "Fast parallel association rule mining without candidacy generation," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, ser. ICDM '01, 2001, pp. 665–668.

[8] L. Liu, E. Li, Y. Zhang, and Z. Tang, "Optimization of frequent itemset mining on multiple-core processor," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07, 2007, pp. 1275–1285.

[9] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim, A. Nguyen, Y.-K. Chen, and P. Dubey, "Cache-conscious frequent pattern mining on modern and emerging processors," *The VLDB Journal*, vol. 16, no. 1, pp. 77–96, 2007.

[10] I. Pramudiono and M. Kitsuregawa, "Tree structure based parallel frequent pattern mining on pc cluster," in *DEXA*, 2003, pp. 537–547.

[11] M. El-Hajj and O. R. Zaïane, "Parallel bifold: Large-scale parallel pattern mining with constraints," *Distributed and Parallel Databases*, vol. 20, no. 3, pp. 225–243, 2006.

[12] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: parallel fp-growth for query recommendation," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 107–114. [Online]. Available: http://doi.acm.org/10.1145/1454008.1454027

[13] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng, "Balanced parallel fp-growth with mapreduce," in *2010 IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT)*, 2010, pp. 243 – 246.

[14] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *SIGMOD '00*, 2000, pp. 1–12.

[15] D. Chen, C. Lai, W. Hu, W. Chen, Y. Zhang, and W. Zheng, "Tree partition based parallel frequent pattern mining on shared memory systems," in *Proceedings of the 20th international conference on Parallel and distributed processing*, ser. IPDPS'06, 2006, pp. 313–313.

[16] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[17] A. Finamore, M. Mellia, M. Meo, M. Munafò, and D. Rossi, "Experiences of internet traffic monitoring with tstat," *IEEE Network*, vol. 25, no. 3, pp. 8–14, 2011.

[18] M. Mellia, M. Meo, L. Muscariello, and D. Rossi, "Passive analysis of tcp anomalies," *Computer Networks*, vol. 52, no. 14, pp. 2663–2676, 2008.

[19] T. A. M. Project, "The Apache Mahout machine learning library. Available: http://mahout.apache.org/ Last access on March 2013," 2013.