Unsupervised Network Anomaly Detection in Real-Time on Big Data

Juliette Dromard, Gilles Roudiere, Philippe Owezarski

Unsupervised Network Anomaly Detection in Real-time on Big Data

Juliette Dromard, Gilles Roudière, and Philippe Owezarski

CNRS, LAAS, 7, avenue du Colonel Roche 31031 Toulouse cedex 4, France
{juliette.dromard, gilles.roudiere, philippe.owezarski}@laas.fr

Abstract. Network anomaly detection relies on intrusion detection systems based on knowledge databases. However, building this knowledge may take time as it requires manual inspection of experts. Actual detection systems are unable to deal with 0-day attack or new user's behavior and in consequence they may fail in correctly detecting intrusions. Unsupervised network anomaly detectors overcome this issue as no previous knowledge is required. In counterpart, these systems may be very slow as they need to learn traffic's pattern in order to acquire the necessary knowledge to detect anomalous flows. To improve speed, these systems are often only exposed to sampled traffic, harmful traffic may then avoid the detector examination. In this paper, we propose to take advantage of new distributed computing framework in order to speed up an Unsupervised Network Anomaly Detector Algorithm, UNADA. The evaluation shows that the execution time can be improved by a factor of 13 allowing UNADA to process large traces of traffic in real time.

1 Introduction

Nowadays, networks suffer from many different failures and attacks like, for example DOS, DDOS, network scanning, port scanning, spreading of worms and viruses which can damage them. In order to prevent these damages, network administrators rely on intrusion detection systems (IDS). Two main types of IDS are largely dominant, signature-based IDSs and behavior-based IDSs.

Behavior-based IDSs detect all the operations which deviate from a known normal behavior whereas signature-based IDSs detect only known attacks for which they possess a signature. Therefore these two techniques rely on previous knowledges. Building new signatures and new models of normal behaviors take time as it requires manual inspection of human experts. It implies that signatures-based IDSs may not be aware of new attacks and behavior-based IDSs of new users' behaviors, as a result they may launch many false negatives or false positives.

In order to overcome these knowledge-based IDSs' issues, researchers focuse their attention on detectors which rely on no previous knowledge: the unsupervised network anomaly detectors. These systems aim at detecting anomalous flows, i.e rare flows that possess different patterns from normal flows. They rely on the two following hypothesis:

Hypothesis 1 *Intrusive activities represent a minority of the whole traffic [18]*

Hypothesis 2 *Intrusive activities's pattern is different from normal activities' patterns [17]*

Therefore the central premise of anomaly detection is that intrusive activity is a large subset of anomalous activity [17]. To detect anomalous flows,these detectors often rely on machine learning techniques (MLTs) which can be either "supervised", label data are then required to identify patterns, or "unsupervised", no previous knowledge is required to discover data's patterns.

Unsupervised network anomaly detector exploits unsupervised MLTs to identify flows which have rare patterns and are thus anomalous. However, detecting anomalies may then be time-consuming as these systems need to dive deeply in the flows' features to identify their patterns, therefore they can hardly meet IDS's real time requirements. To solve this issue, existing detectors may process only sampled data which implies that harmful traffic may be not processed and so not detected [4]. To overcome this limitation,we propose in this paper to take advantage of a new distributed computing system to deploy an unsupervised network anomaly detector on a large cluster of servers.

The remainder of this paper is organized as follows. Section 2 presents related works. Section 3 presents UNADA, an unsupervised network anomaly detector which has been previously proposed by our team. Section 4 presents the implementation and validation of UNADA over a large cluster of servers in terms of computational time and scalability. The possible future works are then discussed and we conclude in Section 6.

2 Related Works

The problem of unsupervised network anomaly detection has been extensively studied in the last decade. Existing systems generally detect anomalies others. For that purpose, many techniques can be used such as multi resolution Gamma modeling [7], Hough transform [12], the Kullback-Leibler (KL) distance [13], however, two main techniques are largely dominant in this area: principal component analysis [16] [14] (PCA) and clustering methods [6] [19].

PCA identifies the main components of the normal traffic, the flows distant from these components are considered anomalous. The pioneering contribution in this area was published by Lakhina et al. [16]. PCA based detectors' main drawback is that they they don't allow to retrieve the original traffic features of the anomalous traffic. This difficulty is overcame in [14] by using random projection technique (sketches): the source IP addresses of the anomalies can then be identified.

Most of unsupervised network anomaly detector rely on clustering techniques, [6] [19] are some relevant examples. Clustering algorithm group similar flows in clusters, to determine similarities between flows distance function like lie outside the clusters are rare flows and are then flagged as anomalous. UNADA falls

within the clustering-based detectors and presents several advantages w.r.t. existing network anomaly detectors. First, it works in a complete unsupervised way, so it can be plugged to any monitoring system and works from scratch. Secondly, it combines the notions of subspace clustering and evidence accumulation (EA) and can so overcome the curse of dimensionality [10]. Finally, UNADA is highly parallelizable and can so be easily implemented over a large cluster of nodes to process large amount of traffic in a small amount of time. These unsupervised detectors have to dive deeply in the collected traffic in order to identify patterns. As a result, they often rely on techniques which time complexity is not linear which prevents them from being scalable and real-time.

Closely related to our work is Hashdoop [11]. Hashdoop is a generic framework, based on the MapReduce paradigm, to distribute the computing of any unsupervised network anomaly detector in order to speedup the detection. In Hashdoop, the traffic is collected in time-bins which are then cut in horizontal slices: all the traffic from or to a same IP address must lie in the same slice. Many detectors are then launched in parallel, each processes the traffic of one slice, the obtained results are then aggregated. The authors claim that their framework is generic, but they only validate it on very simple detectors based on change detection. Furthermore, in Hashdoop, as all the traffic, from and to a same IP address must lie in a same slice, this latter may possess a majority of intruder's traffic and hypothesis 1 may no longer apply. Therefore no detector which relies on the hypothesis 1 and so on unsupervised MLTs like UNADA can be distributed with Hashdoop.

3 The Network Anomaly Detector

UNADA is a network anomaly detector which has already been proposed by our team in [5]. UNADA works on single-link packet-level traffic captured in consecutive time-slots of fixed length ΔT and runs in three main steps: the change detection, the clustering and the EA step.

The first step aims at detecting anomalous slots by detecting change in the traffic. To this end, multi-resolution flow aggregation is applied on the traffic at each time slot. It consists in aggregating packets at different level from finer to coarser-grained resolution in "flows". There is 8 aggregation levels, each level l_i ($i \in [1,8]$) is defined by a mask (/32, /24, /16, /8) and the IP address (IPx) on which the mask is applied which can be be either the IP source (IPsrc) or the IP destination (IPdst). For each level, the traffic is represented by a set $Y = \{y_1, ..., y_f, ...y_F\}$ where each element y_f is considered as a "flow" and F is the total number of flows. In this context, a flow y_f is a subset of the original traffic having the same IPx/mask. For each level l_i, multiple time series Z_t^{li} are then computed, each time series considers a simple metric t such as number of bytes, number of packets, number of IP flows. One point for each time series is built by slot. Then, a generic change detection algorithm is applied on each time series at each new time slot. If the detection algorithm launches an alarm, it

implies that there is a change in the traffic pattern probably due to anomalous activities: this slot requires a deeper inspection and is then flagged.

The second step aims at extracting anomalies from the flagged slots thanks to a subspace clustering algorithm. It takes as input the traffic $Y = \{y_1, ..., y_F\}$ extracted from the flagged slots and aggregated according to the level which has raised the alarm. Each flow y_f can be described by a set of A features in a vector $x_f \in \mathbb{R}^{\mathbb{A}}$. The set of vectors of every flow is denoted by a normalized matrix $X = \{x_1, , ...x_F\}$ representing the features space. Numerous features can be computed over a flow y_f such as: nDsts (# of different IPdst), nSrcs (# of different IPsrc), nPkts (# of pkts), nSYN/nPkts, nICMP/nPkts, etc. Any other attribute sensitive to anomalies can be chosen making UNADA a very flexible tool. To detect anomalous flows in Y, a subspace clustering algorithm is applied to the features space. Each vector of features is considered as a point in the clustering algorithm and each point isolated from the others and which does not belong to any cluster is identified an an outlier. In order to identify any form of cluster, UNADA is based on a density based grid algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [8] which takes two parameters, n_{min} which represents the minimum number of neighbors a point must have to be a core point in a cluster and $\varepsilon_{neighbor}$ which defines the distance of a point's neighborhood.In DBSCAN, a cluster is defined as a area in the data space of higher density. However, DBSCAN suffers from the curse of dimensionality, i.e. when the dimensionality increases the points become increasingly sparse and distance between pairs of points becomes meaningless: DBSCAN can then hardly identify clusters. To overcome this issue, UNADA is based on subspace clustering [15] and EA. The subspace clustering consists in dividing the features space into N many subspaces and performing DBSCAN on each independently. At the end of this step a clustering ensemble $P = \{p_1, ...p_N\}$ is formed where each each element p_i represents the partition obtained on the i^{th} subspace. As DBSCAN provides better results in low-dimensional spaces, the dimension of each subspace is set to 2, which gives $N = m(m-1)/2$ partitions.

The third step of DBSCAN is the EA for outliers identification (EA4O) where the N different partitions are combined to identify the anomalies. This step accumulates for each point the level of abnormality it gets in each subspace. In a subspace, if a point belongs to a cluster its level of abnormality is null, otherwise its level of abnormality is proportional to its distance with the centroid of the biggest cluster. A dissimilarity vector $D = \{d_1, .., d_F\}$ is built where each element d_f reflects the accumulated level of abnormality of the flow y_f. In order not to overwhelm the network administrator with anomalies, only the most pertinent anomalies are selected. To identify them, the dissimilarity vector is sorted and an anomaly detection threshold t_h is defined. t_h is set at the value for which the slope of the sorted dissimilarity presented a major change. Finally, every flow y_f with a dissimilarity score d_f above this threshold is considered as anomalous. UNADA's pseudo-code is presented in algorithm 1.

Algorithm 1 Steps 2 and 3 of UNADA

Initialize:
  Set n_{min} and ε
  Set D the dissimilarity vector to a null F*1 vector
for i=1:N **do**
    $P_i = DBSCAN(X, n_{min}, \varepsilon_{neighbor})$
    $UpdateD, \forall f \in F$
    if $o_i^f \in P_i$ **then**
      $w_i \leftarrow n/(n - n_{max_i})$
      $D(f) \leftarrow D(f) + d(o_i^f, C_i^{max}) * w_i$
    end if
end for
$t_h = computeTresh(D)$
Retrieve anomalous flows $F' = \{f,$ D(f)>$t_h\}$

4 Performances Evaluation of UNADA

In this paper, we aim at detecting the anomalies of a Spanish Internet service provider's core network which is crossed by around 1.2Gbit/s of data. As UNADA requires only the packets' header to detect anomalies, only 64 bits of each packet is stored. To deal with such a traffic, UNADA has to process 19,2 Mbit/s, so 1.6 TeraBytes per day, which makes a huge amount of data.

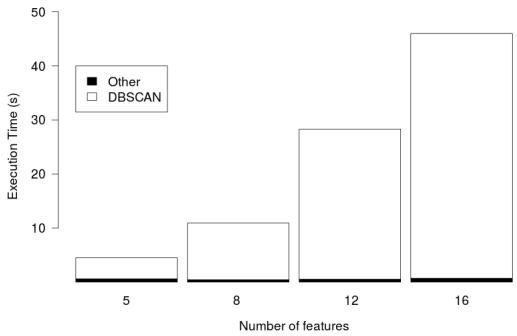

Fig. 1. UNADA's mean execution time for a 15s time slot

For UNADA's evaluation, the aggregation's level is set to IPsrc/16 and the time slot $\triangle T$ to 15 seconds. A slot is made up of around 2.000.000 packets. Furthermore, the evaluation does not consider the flows' features computation time, we assume that a dedicated hardware processes this task upstream. Figure 1 displays UNADA's mean execution time of one slot on a single machine with 16 Gbit of RAM and an Intel Core i5-4310U CPU 2.00GHz. For the local mode, Spark is not used, as it would slow down UNADA's execution and it takes advantage of two cores; one is dedicated to the Java's garbage collection and the

second to UNADA . It shows that UNADA's execution time is mainly due to the clustering step and increases with the number of features.

We analyze 60 slots of traffic and found many anomalies, most of them were induced by large point to multipoint traffic or flashcrowds which could represent from 25 to 10% of the total core network traffic. Furthermore UNADA has detected alpha flows, accounting for 20% of the whole traffic, unknown anomalous flows detected due to a hight number of ICMP packets and finally a SYN attack. Figure 2 depicts one subspace where the SYN attack can be observed, for easier viewing and diagnosis, the subspace is not normalized. Few outliers and a big cluster made up of more than 5.000 points can be observed. To improve UN-

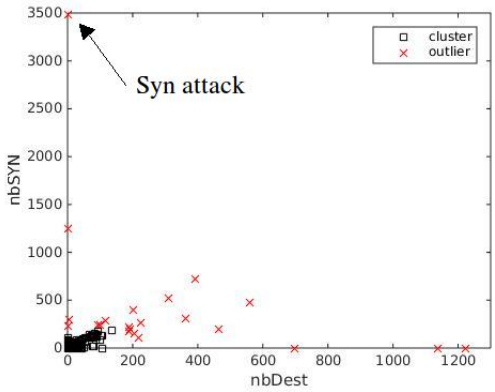

Fig. 2. Detection of a Syn attack

ADA's performances in terms of execution time and scalability, it is deployed on a large cluster of servers. To ease the cluster management and computing distribution of UNADA, we take advantage of an existing big data tool Spark 1.2.0 [1] which is an open source cluster computing framework developed by the Apache Software Foundation. Spark displays better performance than the famous Hadoop tool based on the map and reduce paradigm proposed by Google. Indeed, for certain computing tasks, Spark can fasten $\times 100$ the execution time compared to Hadoop [20]. Furthermore, Spark offers over 80 high-level operators that make it easy to build parallel applications. The validation has been performed on the Grid5000 platform [2], a large-scale and versatile testbed which provides access to a large amount of resources: 1000 nodes, 8000 cores, grouped in homogeneous clusters. In Spark, a master node runs the main() function of the application, creates the SparkContext, acquires executors on nodes in the cluster and then sends tasks to run to the executors. The memory the executors use, as well as the total number of cores across all executors can be finely tuned. UNADA's evaluation is performed on nodes with 8 GB of RAM, two CPUs at

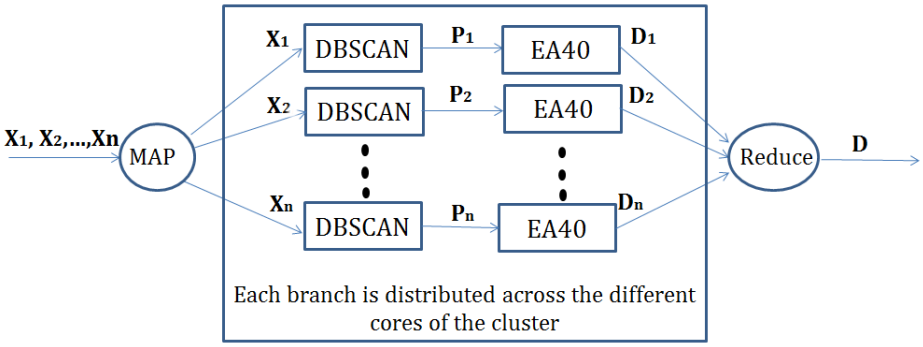

Fig. 3. UNADA's deployment over a cluster of servers with Spark

a frequency of 2.26GHz, each with 4 cores. To deploy UNADA over a cluster of servers, two Spark's high-level operators are used (see Figure 3):

- the map operator which sends across the different cores of the cluster the processing of the N subspaces. The clustering and the EA of each subspace are thus parallelized. The map function returns a dissimilarity vector for each subspace.
- the reduce operator which aggregates the dissimilarity vector obtained in each subspace. It simply sums the dissimilarity vector of each subspace to obtain the final dissimilarity vector.

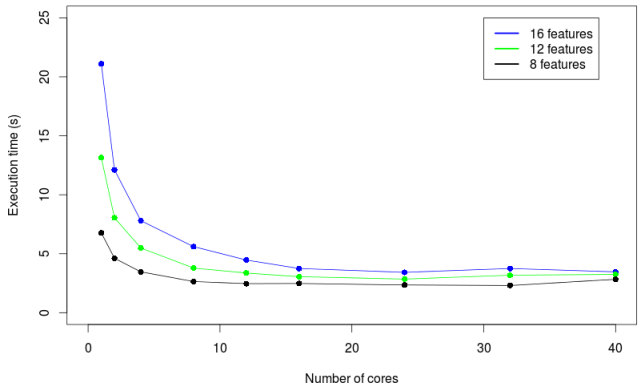

Fig. 4. Execution Time of UNADA

UNADA's deployment has been validated in terms of scalability and execution time. Figure 4 displays UNADA's execution time according to the number of features and cores considered. UNADA's execution time for each feature decreases until reaching a threshold. Furthermore, the difference in execution time of UNADA with different number of features tends to decrease while adding new cores, which implies that a high number of features does not prevent from using the detector. Figure 5 depicts the improvement in UNADA's execution time

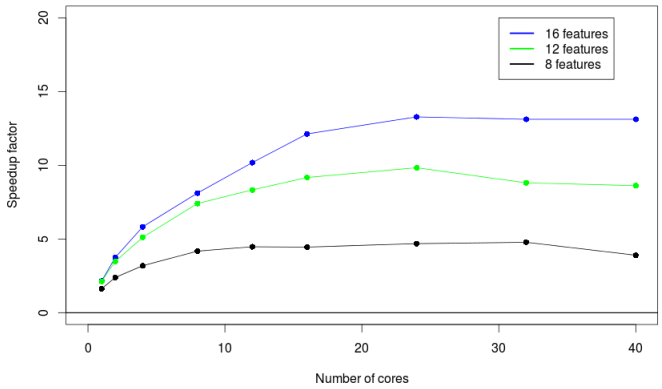

Fig. 5. Speedup of UNADA

with different number of cores compared with a local execution of the detector. The gain in time increases with the number of cores till reaching a threshold . However, for UNADA with 8 and 12 features, from a certain number of cores this gain slightly decreases. This decrease may be due to the fact that serializing the data and sending them to the servers of the cluster may take longer than processing them. Finally, the validation shows that distributing UNADA's computing can improve significantly the processing time till reaching a speedup factor of 13.

5 Future Works

The performance improvement provided by Spark lead us to imagine new ways of detecting anomalies. UNADA detects anomalies over long fixed-length period of time. As this time-bin has a long fixed length, it does not allow an accurate analysis over time. As an example some flows might appear as anomalous simply because they end at the start of the time-bin, consequently having not enough packets analyzed. Therefore, our system would benefit from having a more frequently updated view of the features space. However, because of the algorithm

complexity, we can not perform a clustering over the whole features space more often. Nevertheless, we can think up several solutions. As proposed in [9], we could use an incremental version of the DBSCAN algorithm to incrementally update clusters. When a new packet arrives, it only modifies the features of the corresponding flow within the features space and partially re-compute the cluters. As a per-packet computation is not realistic, aggregating several packets before updating the corresponding flow within the DBSCAN algorithm can be considered. Furthermore, as incremental DBSCAN does not provide an efficient way of updating already clustered points, we also consider updating only points that are enough different from their previous value. This Incremental version of DBSCAN, could allow to detect earlier and more efficiently anomalies. [3] proposes a GPU based implementation of the DBSCAN algorithm which allows to gain a x100 speed-up factor from a typical CPU based implementation. Therefore, as UNADA is based on DBSCAN and is, at least, per-subspace parallelizable, we can expect good results from a GPU based implementation. This speed-up factor could let us perform more frequent updates of UNADA results, moving UNADA closer to real-time efficiency.

6 Conclusion

Unsupervised network anomaly detectors mainly rely on machine learning techniques which complexity are often hight and which can thus hardly be real time or deal with large traffic traces. In this paper, we have proposed to take advantage of new distributed computing systems in order to speed up an unsupervised network anomaly algorithm. Dividing the features space in subspaces allows UNADA to run in parallel multiple DBSCANs and EA algorithms. Spark's high operators distributes the processing of the subspaces over the servers of the cluster and aggregates then the result. UNADA's deployment with Spark can improve the execution time by a factor of 13. This paper is a step forward for detecting network anomalies in real time on large non sampled traffic such as the traffic of an Internet provider's core network.

Acknowledgements

This work has been done in the framework of the FP7 ONTIC project (see http://ict-ontic.eu) funded by European Commission under the Seventh Framework Programme. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

1. Apache spark - lightning-fast cluster computing. https://spark.apache.org/ (Accessed: 2015-04-29)

2. Grid5000. https://www.grid5000.fr (Accessed: 2015-04-29)
3. Andrade, G., Ramos, G., Madeira, D., Sachetto, R., Ferreira, R., Rocha, L.: G-dbscan: A GPU accelerated algorithm for density-based clustering. Procedia Computer Science pp. 369–378 (2013)
4. Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A.: Impact of packet sampling on anomaly detection metrics. In: Proc. of the 6th ACM SIGCOMM Conference on Internet Measurement. pp. 159–164 (2006)
5. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. Computer Communications pp. 772–783 (2012)
6. Celenk, M., Conley, T., Willis, J., Graham, J.: Anomaly detection and visualization using fisher discriminant clustering of network entropy. In: Third International Conference on Digital Information Management. pp. 216–220 (Nov 2008)
7. Dewaele, G., Fukuda, K., Borgnat, P., Abry, P., Cho, K.: Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In: Proc. of the 2007 Workshop on Large Scale Attack Defense. pp. 145–152. ACM (2007)
8. Ester, M., peter Kriegel, H., S, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. pp. 226–231. AAAI Press (1996)
9. Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proc. of the 24rd International Conference on Very Large Data Bases. pp. 323–333 (1998)
10. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. Emerging Topics in Computing, IEEE Transactions on pp. 267–279 (Sept 2014)
11. Fontugne, R., Mazel, J., Fukuda, K.: Hashdoop: A mapreduce framework for network anomaly detection. In: INFOCOM WKSHPS. pp. 494–499 (April 2014)
12. Fontugne, R., Fukuda, K.: A hough-transform-based anomaly detector with an adaptive time interval. SIGAPP Appl. Comput. Rev. pp. 41–51 (2011)
13. Gu, Y., McCallum, A., Towsley, D.: Detecting anomalies in network traffic using maximum entropy estimation. In: Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement. pp. 32–32 (2005)
14. Kanda, Y., Fukuda, K., Sugawara, T.: Evaluation of anomaly detection based on sketch and pca. In: GLOBECOM 2010. pp. 1–5. IEEE (2010)
15. Kriegel, H.P., Kroger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data (2009)
16. Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: Proc. of ACM SIGCOMM 2004. pp. 219–230 (Aug 2004)
17. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. Netw. pp. 3448–3470 (2007)
18. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: In Proc. of ACM CSS Workshop on Data Mining Applied to Security. pp. 5–8 (2001)
19. Wei, X., Huang, H., Tian, S.: A grid-based clustering algorithm for network anomaly detection. In: Data, Privacy, and E-Commerce, 2007. ISDPE 2007. The First International Symposium on. pp. 104–106 (Nov 2007)
20. Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark: SQL and rich analytics at scale. In: Proc. of the 2013 ACM SIGMOD International Conference on Management of Data. pp. 13–24 (2013)