Early Detection of Fraud Storms in the Cloud

Hani Neuvirth[1(✉)], Yehuda Finkelstein[1], Amit Hilbuch[1], Shai Nahum[1], Daniel Alon[1], and Elad Yom-Tov[2]

[1] Azure Cyber-Security Group, Microsoft, Herzelia, Israel
{haneuvir,t-yehudf,amithi,snahum,dalon}@microsoft.com
[2] Microsoft Research, Herzelia, Israel
eladyt@microsoft.com

Abstract. Cloud computing resources are sometimes hijacked for fraudulent use. While some fraudulent use manifests as a small-scale resource consumption, a more serious type of fraud is that of fraud storms, which are events of large-scale fraudulent use. These events begin when fraudulent users discover new vulnerabilities in the sign up process, which they then exploit in mass. The ability to perform early detection of these storms is a critical component of any cloud-based public computing system.

In this work we analyze telemetry data from Microsoft Azure to detect fraud storms and raise early alerts on sudden increases in fraudulent use. The use of machine learning approaches to identify such anomalous events involves two inherent challenges: the scarcity of these events, and at the same time, the high frequency of anomalous events in cloud systems.

We compare the performance of a supervised approach to the one achieved by an unsupervised, multivariate anomaly detection framework. We further evaluate the system performance taking into account practical considerations of robustness in the presence of missing values, and minimization of the model's data collection period.

This paper describes the system, as well as the underlying machine learning algorithms applied. A beta version of the system is deployed and used to continuously control fraud levels in Azure.

1 Introduction

The adoption of the public cloud as an agile model for computational resources consumption is continuously increasing. The high scalability of these services offer many opportunities, as well as new challenges. Examples include failure detection [1, 2], resources optimization [3–5], and security [6, 7]. However, a common challenge to all is the efficient and effective analysis of large quantities of data that is continuously accumulated by such cloud platforms.

A significant portion of the data collected at the cloud is in the form of time series data, e.g., signals from the monitoring of continuous resource use. Therefore, machine learning algorithms performing time series analysis and forecasting are commonly applied. Numerous algorithms have been developed for this purpose over the years [8]. The most established ones are auto-regressive models, integrated models and moving average

A. Bifet et al. (Eds.): ECML PKDD 2015, Part III, LNAI 9286, pp. 53–67, 2015.
DOI: DOI: 10.1007/978-3-319-23461-8_4

models, modeling the signal value at a certain time-point using linear dependencies on preceding data points. Various extensions to these algorithm for multivariate analysis where the underlying entity is a vector of co-evolving signals also exist [9].

A common analysis setting of these time series in the cloud is anomaly detection [10]. Wang et al. developed EbAT, an entropy based anomaly testing for detecting anomalies in cloud utilization patterns [11]. In a more recent paper they describe a lightweight online algorithm for this task based on the Tukey and Relative Entropy statistics [12]. Dean et al. created the Unsupervised Behavior Learning (UBL) algorithm which leverages self-organizing maps to detect performance anomalies [13]. Vallis et al. focused on long-term anomaly detection analyzing data from twitter [14].

With the cloud environment being noisy and dynamic, short periods when reliable data is only partially available often occur. Therefore, in order to be applicable in practice, the algorithms must be able to operate in the presence of missing values. For univariate time series, linear and spline interpolation methods are a common approach [15]. An advanced approach is provided by the DynaMMo algorithm, which aims to enhance the information from the correlations among multiple dimensions, through modeling co-evolving time series through the use of latent variables [15]. Wellenzohn et al suggest a method imputing the missing values from the k most similar patterns in historical data [16]. Xie et al. developed the MOUSSE algorithm which is based on submanifold approximation, which can handle some amount of missing data, and demonstrated it on theft detection and solar flare detection on video streams [17].

Among the many challenges in the cloud, of primary importance is ongoing services availability. In parallel to the genuine users, these services attract fraudsters trying to utilize the resources afforded for more nefarious means. In order to avoid decreases in service quality due to fraudulent activity, cloud providers have to either maintain excessive resources or detect, as soon as possible, consumption by fraudsters and take relevant action.

Here we address the early detection of significant peaks in fraudulent activity in cloud systems, which we term “fraud storms” (FS). In order to avoid a reduction in the quality of service (QoS) as a result of this kind of indirect attacks [18], the service provider has to maintain large amount of excessive resources, which has significant cost implications. We examine two different possible approaches for the problem – a supervised approach aiming to predict the continuous fraud levels, and an unsupervised approach searching for anomalies in multivariate capacity related signals. The requirement for prompt detection poses a constraint on the resolution of data available for the analysis, which is collected at a datacenter level, as opposed to subscription level data, which is too big to collect within the required time constraints.

2 Methods

2.1 The Problem Setting

Microsoft Azure is one of the largest public-cloud platforms available, spanning several large data centers around the world. The size of each data center varies depending on local needs, economic considerations, etc. From a fraud protection perspective, this means that the effect of fraud peaks on a certain data center depends on its size.

In addition, Azure has a wide variety of offer types through which users can subscribe to the system. For example, free trial accounts that offer limited resources to new subscribers, and Pay-as-you-go offers allowing users to pay at the end of the billing period for specific resources consumed during this time. Each of these offer types requires a different verification process on registration to the system, designed to address the associated fraud risk.

The final true labeling of a certain subscription as fraudulent is based on post-usage information, for example, through information that certain credit cards were reported stolen. These labels necessarily arrive much later than the fraudulent use, typically in the order of a month. However, if a specific subscription is suspected to be fraudulent, the cloud operator has the ability to manually investigate the user further, and if found to be fraudulent, the subscription is closed. In the case where a group of fraudulent subscriptions is identified, a bulk shutdown process can be employed.

The type of resources consumed by fraudsters highly depends on the specific fraud type. Bitcoin mining would usually require high compute resources, while spamming and click-frauds might have a greater effect on the bandwidth use.

To achieve a high-resolution control over fraud storms, supplying adequate reliability for the different data center regions and offer types, similar offer types have been grouped together by a domain expert, and the analysis is performed independently for each pair of region and offer group, which we term here a sample-set. Effectively, in the available data there is some correlation between FS events on different regions, however, in this study, the analysis of each sample-set is independent of the others.

2.2 The Labels

The objective of this work is to detect fraudulent peaks affecting compute resources. To obtain FS labels, the ongoing number of fraudulent cores in use was extracted, and a domain expert used this information to assign labels to hourly data on an eight months period, spanning from January 6 to September 10. Note that this procedure was performed on past events, once the full information about fraud was available. The labels indicate the start date of the FS event, the date of the first bulk-shutdown of fraudulent subscriptions employed to discard the fraudulent accounts, and the date in which the fraudulent cores consumption returned to its limited regular level.

2.3 The Analyzed Signals

We use two signals sampled at hourly intervals in this version of the FS detection component: average of the total number of cores used in the past hour in a region and offer group, and the total number of new subscriptions belonging to a certain offer group. These signals are extracted at the level of the datacenter, and thus available at a short latency. Here we use data gathered between January 6 and September 10, 2014. Some of the regions and offer groups became operational only later within this period, and thus their analysis sequence is shorter. Note that the breakdown of the new subscriptions signal to the different regions is not available, and thus the exact same signal is shared across all regions. The design of the algorithms is such that new signals can be easily added.

2.4 Learning Approaches

Two very different learning approaches can be utilized for this problem. One is a two-layer supervised approach, analogous to the labeling process, where the first layer predicts the continuous signal of the number of fraudulent cores that was used to derive the classification, and the second layer uses a classification algorithm over this signal, to predict the binary FS labeling. An alternative approach follows the observation that each FS is an outlier of the system and hence applies an anomaly detection approach.

The Supervised Analysis Approach

This analysis includes two independent layers: using regression to predict the number of fraudulent cores (FC) at each time-point from the available new-subscriptions and total-core-usage signals, followed by a classification of this prediction to retrieve the binary FS prediction.

The input to the present version of the system are two signals, sampled hourly: the number of cores in use, and the total number of new subscriptions created for this offer group. The number of cores at each region and offer group was first processed to define their analysis starting point, discarding data with no sufficient variation, by requiring a minimum standard deviation of 3 on a 7-days window. This is due to the fact that data starts to flow from the data centers and offer types already at their testing phase, before they actually become operational.

The regression analysis in the first layer was performed independently for each sample-set (region and offer group). The features used include the values at the past 1,2,4 and 8 hours, 1, 2, 4, 7 and 14 days, and averages of the signal in windows of 2, 4, 8, 12, 24 and 48 hours, as well as their polynomial products of degree 2. These features were first filtered based on low variance ($<10^{-10}$). Then, they were further filtered using a false discovery rate (FDR) procedure on the p-values obtained from the correlation coefficient of each feature with the outcome setting alpha=0.05. Finally, they were standardized, and fed into a ridge regression model counting over ridge values ranging from 1 to 10^{12} at a logarithmic scale. Model selection was performed using generalized cross validation with the python scikit-learn package.

In the training phase, the regression, predicting the number of fraudulent cores is performed in 5-fold cross-validation in order to obtain signal with the noise resulting from the prediction model. Another regression model is trained on the full training data to be used on the test set. Three features were extracted from this noisy signal for each time-point t, relative to the preceding time point (t-1): their difference (diff=FC(t)-FC(t-1)), their relative difference (diff/mean(FC)), and their ratio (FC(t)/FC(t-1). These features coming from all the sample-sets are then fed into a logistic regression model to train the classification. For the labeling of the classification layer, two schemes were explored: a strict labeling considering only the time period until the bulk shutdown as FS, named “bulk shutdown” (see section 2.2), and a looser definition representing the whole FS period as positive.

The Outlier Detection Approach

The outlier detection algorithm processes the time series of the same selected signals. It is composed of two parts: prediction of the value at the next time point, and evaluation of the error. Our signals show typical seasonality patterns (An example is shown in Figure 6), and thus, an analysis similar to the one employed by Bay et al. was used [19]. Specifically, we utilized a sliding window approach, in which the values of both signals in a recent time window are used to predict the value of each signal at the next timepoint. The selected features include the values at the past 1,2,4, and 8 hours, 1, 2, 4, 7 and 14 days, and averages of the signal in windows of 2, 4, 8, 12, 24 and 48 hours.

Several regression models have been explored for the prediction, including random forest regression, k-nearest neighbors and SVM with polynomial and RBF kernels, however all gave a similar performance to the one obtained by a ridge regression model, thus, this simpler model was selected. The ridge in this model was selected using cross validation over a log-scale set of values between 1 and 10^{12}.

In the training phase, prediction was performed using a sliding window starting from a 60 days history, and in gaps of 30 days. The history length used for training each window was set to a minimum of 60 days, and a maximum of 100 days, in order to have data that is both sizable and relevant. For each window, we used linear regression on the features listed above to predict the value at the next time-point in 5-fold cross validation (CV). One additional training is performed on the full data window.

The CV data was used to estimate the error distribution. A multivariate Gaussian distribution is assumed, and the mean vector and covariance matrix are calculated using a robust estimation procedure available in the python scikit-learn package [20]. These are then used to calculate a p-value for each time-point. There exist two possible definitions for the p-value of the multivariate normal distribution (MVN). The first is analogous to the one-dimensional counterpart, and is based on the Mahalanobis distance [21]. The second considers the distribution in a vectorized manner, and estimates the weight of the cumulative distribution function in a rectangle of values higher than the observed values [22]. The latter demonstrated better performance (data not shown). This observations is aligned with the fact that FS by definition refers to anomalies at the higher cores and subscription values, and thus this p-value estimation method was selected. The p-value bound was optimized by counting over a few selected values.

There are two sources for randomization in this approach: in the sampling of the CV partition, and in the robust covariance estimation [20]. All the experiments were repeated three times, with different seeds, to estimate the associated standard deviation. Unless indicated otherwise, the standard deviations of all the points in the figures in this paper fell below 1%.

2.5 Performance Evaluation

To compare the binary predictions with the true labels, all time-points identified as a FS that are less than 36 hours apart were clustered. Following, each cluster was mapped to the earliest overlapping FS, if any.

Two measures of interest were calculated over these clusters in order to compare the models: the number of FS events detected, and the median time to detect in hours, which is the difference between the start time of the predicted cluster to the start time of the FS label. This measure may take negative values in case the predicted cluster starts before the manually labeled FS start date.

In addition, we calculated FPDays, the fraction of days in which a false FS alert was raised for the sample-set, and a summary measure indicating the fraction of sample-sets having FPDays < 0.05. Note, that high values of this measure correspond to low FP rates. Naturally, the aim is to maximize the number of FS events detected and the fraction of sample-sets having a low FP rate, while minimizing the detection time.

3 Results

This section begins with comparing the different analysis approaches, showing that in our setting the anomaly detection approach is superior to the supervised analysis. Then, we compare the performance of our algorithm, named FraudStormML, with the Seasonal Hybrid ESD algorithm, a state of the art anomaly detection algorithm [14]. Further, we address two key challenges that stem from the practical nature of the problem: making the analysis robust to missing and corrupted data, and enabling fast utilization of the system on newly available sample-sets.

3.1 Analysis Approaches Comparison

Two main analysis approaches have been explored in this study. The input to both is the two hourly signals providing the core usage and count of new subscriptions per region and offer group. Recall that the signal used to derive the manual labels is the number of fraudulent cores. The first approach performs supervised analysis to predict the number of fraudulent cores. Following, a binary classification stage is used to map the continuous prediction to the binary FS label. The second approach performs multivariate anomaly detection on the two input signals.

In the initial analysis using the supervised approach, no FS has been detected. To further investigate this, we explored the errors of each of the layers independently. Figure 1 shows the performance of the classification layer that is based on the true number of fraudulent cores that was used for the manual labeling (gray line with circle markers). These are extremely good results, with a false-positive rate lower than 5% in nearly all sample sets while providing full detection of all FSs. This implies that despite the scarcity of the labels, the logistic regression model can easily capture the manual labeling process, and the failure is related to the regression layer or to the combination of the two layers. Note that these results were obtained only with the "bulk shutdown" classification labeling scheme. When the full FS duration was labeled as positive, no FS has been detected. This can be ascribed to the use of a linear model for the classification, however, the overall good performance suggests that the steep rise in the number of fraudulent cores was the main factor in the manual classification.

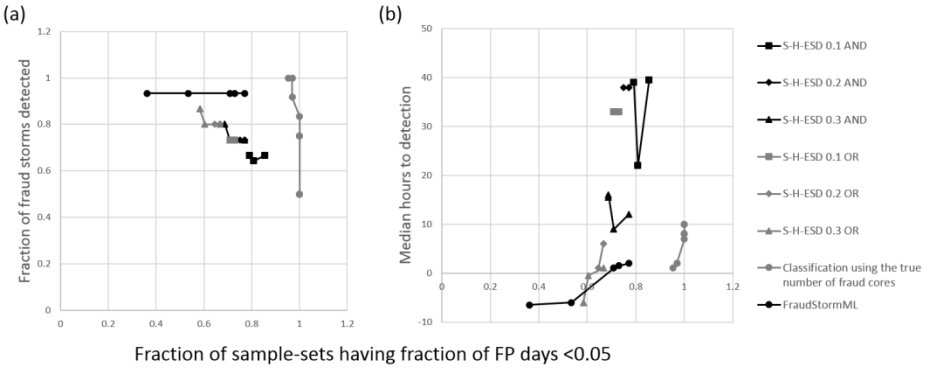

Fig. 1. Performance comparison of the fraud storm detection approaches

Investigating the error of the regression layer, we observed good prediction power in general, but high RMSE values specifically in regions where a FS occurred (Figure 2, Pearson correlation coefficient=0.53, p-value=4e-4). This suggests that the number of fraudulent cores cannot be easily modeled in our context, since the behavior during FS is significantly different from normal, and thus an anomaly detection scheme would be more appropriate.

(a)

Offer Groups

regions	A	B	C	D	E
1	0	0.037	0	0	0.037
2	0	0.037	0.074	0	0.037
3	0	0	0.037	0	0.037
4	0	0	0	0	0.074
5	0	0.111	0	0	0.074
6	0	0.074	0	0	0.037
7	0	0	0	0	0.074
8	0	0	0	0	0.074
9	0	0	0	0	0.074
10	0	0	0	0	0.111

(b)

Offer Groups

regions	A	B	C	D	E
1	8.22	1066.85	50.34	25.98	1738.46
2	7.76	3121.29	3391.17	29.52	1035.49
3	16.95	535.71	1656.73	110.33	446.35
5	12.07	965.95	38.24	26.11	392.26
6	43.34	868.81	471.86	10.06	523.73
7	2.77	0	0	2.39	180.85
8	0.67	55.69	0	0.56	473.42
9	4.61	633.47	44.76	19.64	487.6

Fig. 2. (a) The distribution of FS over regions and offer groups. (b) The RMSE obtained on the regression analysis for each region and offer group. A green-to-red color scale is used to represent the values ranges depicted in the boxes. The correlation between the figures is evident (Pearson correlation coefficient=0.53, p-value=4e-4), proving the regression model fails on FS events, which suggests that an anomaly detection approach might be more suitable. The data for regions 4 and 10 was not available early enough for them to qualify for this analysis.

Turning to the anomaly detection approach, we compare our algorithm, named FraudStormML, to the Seasonal Hybrid ESD (S-H-ESD) algorithm, a state of the art algorithm recently published for anomaly detection in the cloud, specifically designed for time-series data [14]. This algorithm is similar to the one we employed in that both algorithms account for the daily and weekly seasonality in the data. It differs from our algorithm in the specific statistical analysis applied, utilizing the generalized Extreme Studentized Deviate test, and by the fact that the analysis is a univariate one.

To apply this algorithm to our data, we ran the code provided by the authors of this algorithm on each of our signals independently, and combined the outputs using two functions: the "OR" function classifies a time point as FS if an anomaly has been detected in any of the signals, while the more strict "AND" function only classifies as FS the time points in which both signals indicated an anomaly. The parameter controlling for the maximum fraction of anomalies was used with values of 0.1, 0.2 and 0.3, and the sensitivity bound alpha received values of 0.05, 0.1, 0.2 and 0.3, when this value was lower than the corresponding maximum anomaly fraction allowed. All the remaining parameters were assigned their default values.

Figure 1a compares the results of FraudStormML to the various runs of the S-H-ESD algorithm. One can observe that the detection rate of the FraudStormML algorithm is higher at the same FP rates (Figure 1a), and the median time-to-detect is lower (Figure 1b).

3.2 Handling Missing and Corrupted Data

With the cloud environment being a highly dynamic one, it is not uncommon for data to become temporarily corrupted or unavailable. From our experience, those vulnerable times are often associated with fraud storms. Hence, allowing a fast recovery of the system from such events is highly important. Furthermore, when training an anomaly detection model, it would be wise to discard past FS events, as these may reduce the sensitivity of the model. Therefore, an important aspect of a FS detection system is its ability to be trained and applied in the presence of missing data.

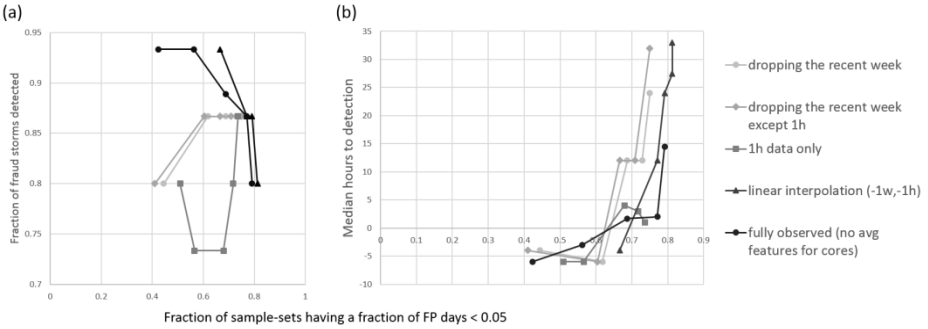

Fig. 3. Performance comparison of the anomaly detection in the presence of missing values

In this analysis we assume a broad missing data model in which the cores-usage data is missing for a period of one week where the same hour at the previous week is the most recent time-point that is available. For the simplicity of this analysis, the features calculating past averages were completely dropped for this signal. The missingness pattern was applied at the feature extraction stage for all the data points in the sample set, and the prediction performance was evaluated.

Figure 3 presents a comparison of five different models: a fully observed model (excluding the averaging features), a model with the most recent week missing, a model where the most recent week except the preceding hour is missing, a model that

is based only on the preceding hour, and a model deriving all the features from a linear interpolation between the value at the preceding week and the value at the preceding hour. We can observe that dropping the recent week has a significant effect of about 10% on the FP rate, and a similar effect on the fraction of FS detected. The effect is somewhat reduced in the model which adds in the preceding hour (1h) feature, simulating the case when the information on the recent hour has been accumulated, however, not to a significant level. Dropping the longer history features, and counting on the 1h feature alone provides similar performance, yet also a noisier one. In contrast, the linear interpolation model manages to recover the performance to a level very similar to the fully observed model.

3.3 The Effect of the Data Accumulation Period Length

As the cloud keeps evolving, new offer groups are created from time to time. Therefore, it is important to be able to estimate the minimal time required for the system to become effective. Note, that as a measure for the system performance on new offer types, this is merely an approximation, since new offer types might have different behavior from the longstanding offers for which we have data.

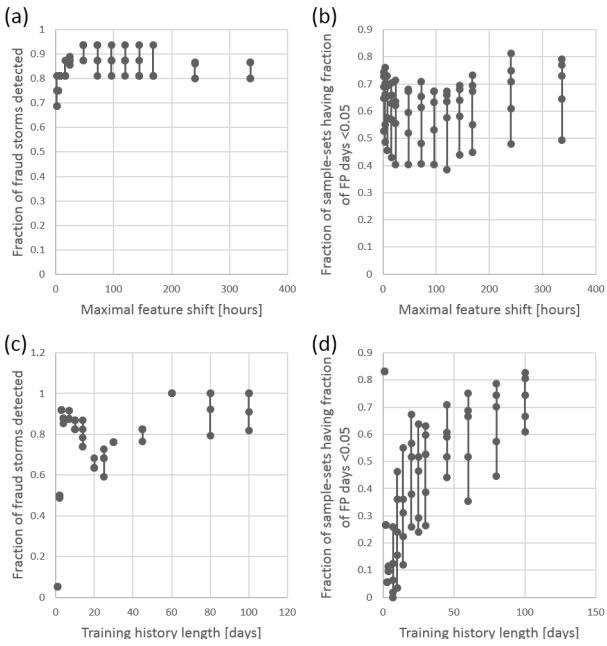

Fig. 4. The performance achieved as a function of the data accumulation period.

The length of the data accumulation period is a sum of two parameters in our model: the longest shift used to derive features, and the amount of data required for training. Figure 4a presents the effect of reducing the extent of the features used. It plots the detection achieved for different feature shifts ranging over values in [2, 4, 8, 16,

24, 48, 72, 96, 120, 144, 168, 240, 336] hours, where in each experiment a single shift was added to all the smaller ones. All dots connected with a line are the results of the same model configuration, with different p-value thresholds ranging from 10^{-5} to 10^{-15}. The experiments for this plot were performed with full training data as was finally used by the FraudStormML system, requiring a minimum of 60 days and at most 100 days. As before, each dot is the mean of three experiments with different randomization seeds. Figure 4a presents a significant decrease in the sensitivity of the model below 48 hours. Then, the detection rate is stabilized. For feature shifts of 1–2 weeks there is some improvement in the noise of the model, making it less sensitive to the p-value threshold. The FP rates are stable for all values, which might be related to the large training set used (Figure 4b).

We evaluate the performance of the model given different history lengths of data used for training. Figures 4b, 4c show the performance as a function of the history length, ranging between 1 to 100 days, with the sliding window gaps set to the history length in case it is smaller than 30, and 30 days otherwise (as in the final model). Experiments have been repeated with three different seeds. The standard deviation was lower or equal to 1% for FP, and lower or equal to 4% for the fraction of FSs detected. Figure 4c shows initial good performance, which is misleading, because, as Figure 4d shows, it is associated with very high FP rates (low fraction of the sample-sets having FP rate < 5%). Then, as the sensitivity of the model increases, the FP rates on (d) decrease, and so does the detection rate on (c). Finally, when sufficient training data is accumulated, the results on both measures reach the desired performance. The results imply that the model can achieve similar performance with feature shift of 1 week, and training history of 60 days. Shorter periods will result in some performance degradation.

4 The System

The algorithm described here is a component of a bigger system controlling the ongoing fraud levels in Azure. This system is based on a map-reduce architecture collecting telemetry data from the cloud around the world, analyzing the high-resolution, subscription-level data. It is based on a large collection of signals, and thus suffers from delays in data latency. The FraudStormML component described here aims to control the fraud at the data center level, and to be thinner in the aspect of signals, in order to significantly reduce the time-to-detect on significant fraud peaks.

The FraudStormML component, is backed-up by a rule-based analysis component. This component is important to support for possible failures of the system, which might result from failures in data collection, for example in mapping of new subscriptions to their offer types, etc. On top of this system, we employ the more sensitive machine learning component. This component scans the main data centers and offer groups on an hourly basis, and feeds the resulting evaluation into a database. A Power BI dashboard reads from this database and provides an overview of the fraud storm state of the system. A snapshot of this system is presented in Figure 5. A detailed figure for each of the regions and offer groups suspected to be affected by a FS is also produced, to allow a deeper manual investigation. The model for this component is trained weekly. Time periods where data was corrupted or missing are being manually fed into a database, and are interpolated during training.

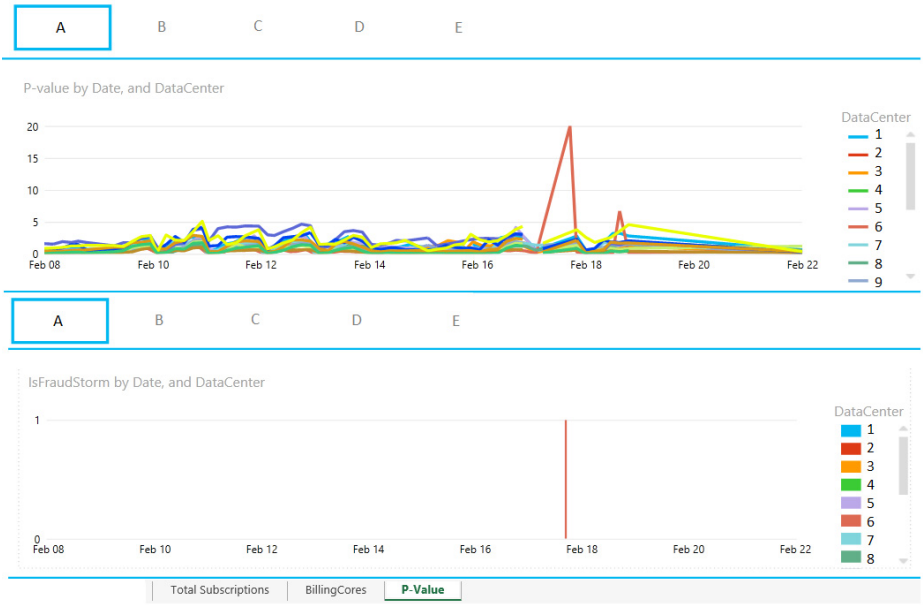

Fig. 5. A snapshot of the FraudStormML component dashboard.

5 Discussion

This paper presents a machine leaning based fraud storm detection system. We examined the performance of a supervised approach, and compared it to an unsupervised anomaly detection approach. Selecting between the two approaches a-priori is not straight forward, as each of them has its advantages. On one hand, the cloud environment is a very noisy one supplying many outliers that are not necessarily FS events. On the other hand, a supervised approach is challenging due the scarcity of FS events. Moreover, we observed that the regression prediction accuracy is especially low at regions where FS occurred, as they are outliers of the model. The supervised approach might gain from use of more sophisticated, non-linear models. However, in a big data system designed to continuously monitor the cloud, the advantage of light-weight models is evident. With an underlying such model, the anomaly detection approach performed significantly better than the supervised alternative.

There are many sources for false positive detections in this system. First is the use of manual labeling, which is a common practice in many machine learning domains, albeit a noisy one. Figure 6 presents a few examples for FP detections. All the panels on Figure 6 share the same x-axis, indicating the timeline. The first (top) panel shows the core-usage signal, the second panel shows the new subscriptions signal, the third panel plots the p-values obtained by the FraudStormML system, with red bars indicating a detection. The panel on the bottom is the continuous signal used for labeling, i.e. the number of fraudulent cores. The purple background indicates regions that were manually labeled as a FS.

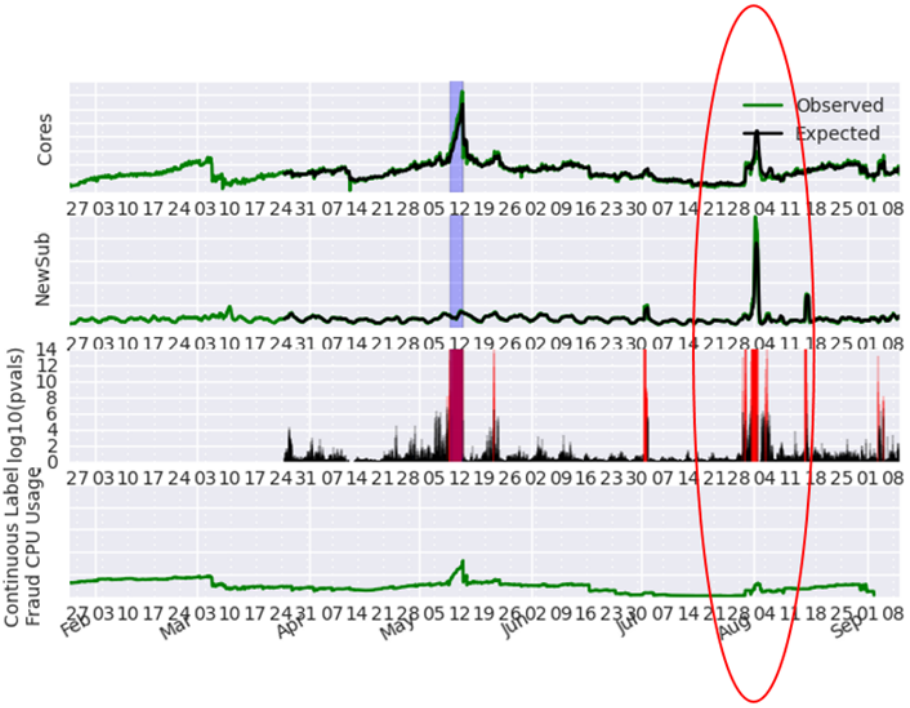

Fig. 6. The input and output signals of the system for one of the sample sets. All the panels share the same x-axis indicating the timeline. The two top panels present the values of the analyzed signals (cores and new subscriptions). In these panels, the green curve presents the observed values, while the black curve represents FraudStormML's predicted values. The third panel presents the $-\log_{10}$(p-value) of the prediction obtained by the FraudStormML component, with red bars indicating a detection. The bottom panel presents the continuous label, namely, number of fraudulent cores in use. The time window that was manually labeled as a FS is depicted in purple background at the top three panels. There appear to be a few anomalies that are FP detections of FS. The FP anomaly marked with the circle is also apparent at the continuous label panel at the bottom, thus representing a limited but true event.

For this region and offer group we observe a single FS event labeled. This event was successfully detected. However, there are a few other anomalies detected by the system. The one designated by the red circle demonstrates one FP detection. Even though this event was not manually labeled as a FS, the corresponding small anomaly in the number of fraudulent cores used (bottom panel) is evident.

Other anomalies detected are true anomalies that are not FS. In practice, the information about their cause is not always available on past data. Since the system became operational, we encountered several such events resulting from normal but rare events such as switching the default region of the system. Each event leads to a manual investigation, which would eventually allow us to identify more relevant signals that could be added to the system in order to reduce the FP rate. Another way to reduce FP rate is to use information coming from different regions, as FSs often affect more than one re-

gion, however, we observed that the migration of a FS between regions is slow, and utilizing this comes at the expense of the detection time of the system.

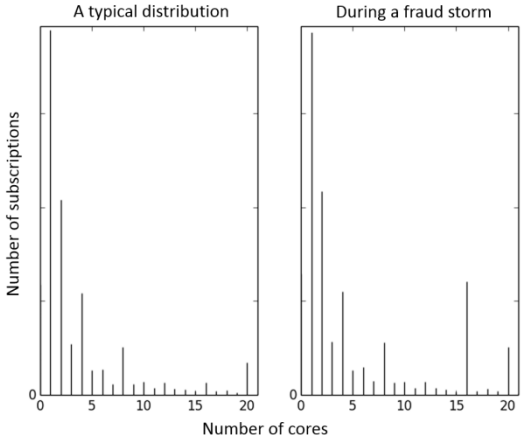

Fig. 7. A qualitative comparison of the distributions of the number of cores used typically, and during a specific fraud storm event on one of the data centers. Typical users aim to minimize their costs, and thus they mostly utilize the minimal required compute power, while fraudsters in this case aimed to maximize their utilization of the system, resulting in a peak at the high cores during a fraud storm.

Modeling fraud is a challenging task as fraudsters constantly keep learning the detection systems, and searching for new ways to bypass them. Features that distinguish between fraudulent and genuine activity can be divided to two types: those monitoring the resource consumption, and those monitoring the fraudsters' breaking-in method. While both are valuable, the latter is not necessarily steady and could very easily change. Conversely, the fraudsters' resources utilization is steadier, usually changing only with the specific fraud purpose, and thus basing a system on these features, as in our analysis, would result in a system with a longer life expectancy. Figure 7 presents an example for such feature, by comparing the typical distribution of the core usage per subscription to the one observed during a FS in one of the data centers. While the genuine user aims to minimize their cost though minimizing the resources consumption, a fraudster that managed to break into the system usually aims to maximize its utilization as much as possible. Therefore, on the right panel we observe a large peak at the high cores consumption during a FS. This information can be modeled into our system through a few possible features: the total cores consumption (as present in the above analysis), the mean core consumption per subscription, with the high values truncated to reduce the noise, and the entropy of the cores-per-subscription histogram. Note that the entropy feature has the unsteadiness disadvantage described above. In our analysis, both alternatives did not improve the prediction over the total number of cores.

Ongoing improvement of the systems is an important aspect of future research. The FraudStormML algorithm can be easily enhanced with new signals, like bandwidth and disk usage. This is important, since the typical fraudster's motivation is dynamic

and changes over time, which affects the type of resources they consume. In the course of system operation, some anomalies will be detected, those that would turn out to be FP can provide new insights that would further be integrated into the system. Having a design which supports this evolutionary process is a fundamental challenge in this field, and a key aspect of the system.

References

1. Bhaduri, K., Das, K., Matthews, B.L.: Detecting abnormal machine characteristics in cloud infrastructures. In: 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW), pp. 137–144. IEEE (2011)
2. Zhu, Q., Tung, T., Xie, Q.: Automatic fault diagnosis in cloud infrastructure. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 467–474. IEEE (2013)
3. Hormozi, E., Hormozi, H., Akbari, M.K., Javan, M.S.: Using of machine learning into cloud environment (A Survey): managing and scheduling of resources in cloud systems. In: 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 363–368 (2012)
4. Beloglazov, A., Buyya, R.: Energy efficient resource management in virtualized cloud data centers. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 826–831. IEEE Computer Society (2010)
5. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**, 755–768 (2012)
6. Hashizume, K., Rosado, D.G., Fernández-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. J. Internet Serv. Appl. **4**, 1–13 (2013)
7. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Inácio, P.R.: Security issues in cloud environments: a survey. Int. J. Inf. Secur. **13**, 113–170 (2014)
8. Bontempi, G., Ben Taieb, S., Le Borgne, Y.-A.: Machine learning strategies for time series forecasting. In: Aufaure, M.-A., Zimányi, E. (eds.) eBISS 2012. LNBIP, vol. 138, pp. 62–77. Springer, Heidelberg (2013)
9. Aggarwal, C.C.: Outlier analysis. Springer Science & Business Media (2013)
10. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. CSUR. **41**, 15 (2009)
11. Wang, C., Talwar, V., Schwan, K., Ranganathan, P.: Online detection of utility cloud anomalies using metric distributions. In: 2010 IEEE Network Operations and Management Symposium (NOMS), pp. 96–103. IEEE (2010)
12. Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., Schwan, K.: Statistical techniques for online anomaly detection in data centers. In: 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 385–392 (2011)
13. Dean, D.J., Nguyen, H., Gu, X.: Ubl: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In: Proceedings of the 9th International Conference on Autonomic Computing, pp. 191–200. ACM (2012)
14. Vallis, O., Hochenbaum, J., Kejariwal, A.: A novel technique for long-term anomaly detection in the cloud. In: Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, Berkeley, CA, USA, pp. 15–15 (2014)

15. Li, L., McCann, J., Pollard, N.S., Faloutsos, C.: Dynammo: Mining and summarization of coevolving sequences with missing values. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 507–516. ACM (2009)
16. Wellenzohn, K., Mitterer, H., Gamper, J., Böhlen, M.H., Khayati, M.: Missing Value Imputation in Time Series using Top-k Case Matching
17. Xie, Y., Huang, J., Willett, R.: Changepoint detection for high-dimensional time series with missing data. ArXiv Prepr. ArXiv12085062 (2012)
18. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M.: A survey of intrusion detection techniques in Cloud. J. Netw. Comput. Appl. **36**, 42–57 (2013)
19. Bay, S., Saito, K., Ueda, N., Langley, P.: A framework for discovering anomalous regimes in multivariate time-series data with local models. In: Symposium on Machine Learning for Anomaly Detection, Stanford, USA (2004)
20. Rousseeuw, P.J., Driessen, K.V.: A fast algorithm for the minimum covariance determinant estimator. Technometrics **41**, 212–223 (1999)
21. Multivariate normal distribution (2015). http://en.wikipedia.org/w/index.php?title=Multivariate_normal_distribution&oldid=651587942
22. Genz, A., Bretz, F.: Computation of multivariate normal and t probabilities. Springer Science & Business Media (2009)