IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

Unsupervised Real-Time Stream-Based Novelty Detection Technique

An Approach in a Corporate Cloud

Anna Vergeles
Cloud Operations
Oracle
Kharkiv, Ukraine
anna.vergeles@oracle.com

Alexander Khaya
Cloud Operations
Oracle
Kharkiv, Ukraine
alexander.khaya@oracle.com

Dmytro Prokopenko
Cloud Operations
Oracle
Kharkiv, Ukraine
dmytro.p.prokopenko@oracle.com

Nataliia Manakova
Cloud Operations
Oracle
Kharkiv, Ukraine
nataliia.manakova@oracle.com

***Abstract*—A highly loaded cloud application environment requires the highest stability and operability, generates large telemetry data streams. These are obvious and actual prerequisites to develop a workload shift detector for the failures prevention aim. Having studied the previous works, the authors developed an approach to the detection of changepoints based on the specific conditions of the streaming telemetry data. The simulation of data center workload has allowed us to generate telemetry data under specific workload, thus we can evaluate the performance of the detector under various conditions. The conducted experiment has shown the viability of the proposed approach as well as directions for further study and improvement.**

Keywords — high-load, cloud, SaaS, telemetry, sensors, logs, real-time, monitoring, streaming data, changepoint, novelty, anomaly detection, unsupervised

I. INTRODUCTION

Oracle Field Service Cloud (OFSC) is a high-load cloud-based mobile workforce management application distributed as Software-as-a-Service with strict Service License Agreement. SLA imposes restrictions to mean time to repair, which is the main reason why early detection of abnormal behavior is crucial for us.

OFSC's environment consists of hundreds of servers with different roles in several data centers across the world. Most common roles are front ends, back ends, databases, storages, etc. Servers within the same cluster with a certain role have similar operating mode while their belonging to different roles almost surely implies working in different modes.

Actually, the cloud architecture is not constant. Increasing number of customers results in scaling and allocating a larger number of hosts in some clusters. Meanwhile, a number of clusters and their role may vary. OFSC as application is being continuously changing and improving.

Whereas OFSC is a cloud-based application, Cloud Operations department performs continuous service monitoring at all the levels of the cloud hierarchy (IaaS, PaaS, SaaS), as well as monitoring of client experience for a variety of functional tasks of the service. Operations produce a significant volume of data: continuous telemetry and application logs both “as-is” and aggregated.

One promising approach to high accessibility and reliability in cloud technology is an analysis of this collected technical and workload parameters in order to identify novel patterns. Once identified, this novel abnormal working behavior should be reported to operations teams that subsequently perform appropriate procedures. In that meaning, abnormal events are events that do not conform to expected patterns. Each discrepancy may be considered an anomaly.

Types and specifics of anomalies do not usually repeat, considering support system of our application working around the clock – they fix and solve emerging incidents. Accordingly, almost each time we deal with a novelty (anomaly), or unseen before operating mode. Supervised learning techniques are not applicable - there is no need in the spot-on detection of expected or scheduled situations like virtual machine restart, and at the same time unplanned situations usually occur only once – each issue results in a set of preventive measures.

The main task of our research is to propose and implement an approach to early detection of novelties and anomalies in a real-working cloud service based on unsupervised learning as a mechanism to watch for shifts in regular workload.

II. REVIEW (RELATED WORKS)

The cloud technologies growth and the necessity of remote hardware monitoring on each of the hierarchical levels as IaaS, PaaS, SaaS, are producing a huge amount of data generated endlessly and in real-time. Processing of such data can be carried out in streaming mode (often called online processing) or in batch mode, which both have its pros and cons [1-5].

One of the important issues of data processing in the streaming mode is the necessity of high-speed computations, especially for machine learning tasks in general and for

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

detecting anomalies in particular, where model training requires several passes and it takes considerable time. Taking into account the peculiarities of our conditions, in order to reduce time delays for learning models in real-time, we focus on partially real-time anomaly detection algorithms, which have initial non-real-time learning phase[6, 7].

Training some models for anomaly detection tasks can be performed in supervised, unsupervised, semi-supervised modes. Since we cannot afford pre-labeling the data, moreover, it makes no sense in constantly changing cloud environment, for our approach, we choose an unsupervised mode.

In scientific papers, we considered the widest range of models suitable for this mode of machine learning, which have specific features for a different type of data stream [8–10].

Besides, it should be highlighted, in many cases, the behavior of the system can change over time, due to well-known problem named concept drift. For example, reconfiguration and upgrading may influence system behavior, so models must adapt to a new definition of "normal" in an unsupervised, automatic way [11, 12].

Individual anomaly rate computed by individual models can be combined in some ensemble technique (e.g., averaging or voting) to form a final score. Such approach presented in many studies [13-17]

Thus, after studying related works, the solution of the studied problem was carried out in the direction of real-time processing of streaming data through formation of an ensemble of models based on non-real-time unsupervised learning techniques.

III. Methodology

This research deals with the data represented as either time series from multiple univariate sensors or aggregated panel data. The main approach is to split the whole process into historic learning part (non-real-time) and real-time pipeline. We mine historical data for expected patterns or states using historic learning module and right after that, in real-time we compare observed mode with the computed expected mode.

A. Data source

Our study is conducted in the field of delivering cloud-based SaaS application, so we have two main types of time series data sources:

• Analogue signals from multiple sensors assembled as snapshots – telemetry, or monitoring data, collected at regular intervals for IaaS, PaaS and SaaS levels.

• Discrete signals without consistent spacing, i.e. data from system and/or application logs. This type of data requires performing pre-processing aggregations.

Those data sources are, in fact, nothing more than infinite data streams of large volume and variety that arrive at the rapid rate – a typical streaming data with its inherent aspects of the concept drift and real-time delivery.

B. Prehandling data

Monitoring checks have different data acquisition intervals depending on a check type. At the historic processing stage, we apply last observation carried forward conversion for these non-uniformly spaced telemetry time series, assuming the last observed value to be valid within the whole time interval. This approach results in alignment of observations on a time grid with step $\Delta\tau$. Generally, at each timestamp in a time grid, τ_i, every metric can be represented as a vector of numeric and factor values:

$$V\tau_i = \langle TIME; OBJECT; GROUP; METRIC; VALUE; TAGS \rangle$$

Furthermore, whereas we have historical data forcibly structured by a time grid, it makes sense to apply the same spacing to future observations to get equidistant points of making decision.

C. Encemble of models and novelty detector

Our cloud-based service is rapidly changing in response to changes in customers' business processes, implementing new application features, etc. Concept drift is a common phenomenon when data distribution changes over time. Any trained model has limited time when it makes accurate predictions. Soon enough, a previously trained model cannot accurately represent data behavior and requires retraining.

The best approach to overcome concept drift is to timely adjust expectations to a current state of a constantly changing system mode. It is important to emphasize that in anomaly detection being more adaptive and flexible should not mean being less sensitive.

In order to adapt to changes in concepts and at the same time to keep an accurate model, we use an ensemble of models to implement novelty detector. With this approach, we calculate anomaly rates independently by each model for each metric for each observation. Among other things, ensemble gives the advantage of uncorrelated errors.

Since we are only discussing a pilot approach to detecting anomalies, our goal is not to select the optimal set of detectors or to find the optimal weights for selected detectors, but rather to focus on a process of detection under certain conditions. We choose several models as inputs to the ensemble, listed below and illustrated in Fig.1:

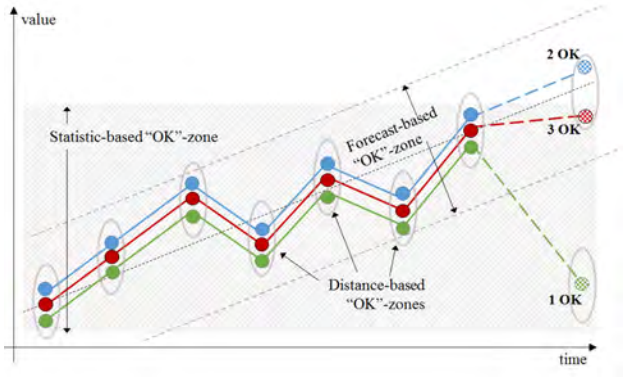

Fig. 1. Basic principle of the proposed novelty detection ensemble

1. Forecast-based (Fb) model. It estimates how big the normalized distance is between the observed data and the value predicted by some forecast model in the non-real-time module for each metric. For our task, we use SSA (Singular spectrum analysis [18]) prediction that is quite accurate for

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

our type of time series. In this model, alert rate rises as soon as an observed value significantly deviates from its prediction.

2. Statistic-based (Sb) model supplements forecast-based model but works independently. The model collects basic statistics from known historical modes and works well for situations when for some metrics graph of the regular has a persistent slope. This model detects trend's permanent growth or recession, which can lead to absolute resource consumption.

3. Distance-based (Db) model takes into account distance from each observed value x_i to every other value in a group for a particular metric in a current time slot. It will alert when there are no other values besides x_i within the certain neighborhood.

D General approach

During the processing of historical data, all models that are part of the ensemble undergo the automatic parameters tuning phase. This set of mutable parameters plays a significant role in the real-time phase. Each parameter is fed in a separate stream and is matched with corresponding observation to calculate anomaly rate. Parallel processing makes our system highly configurable and, what is especially important, fast – there are no complex computations in the real-time phase.

Total anomaly score is calculated in two steps. At first, for each metric we aggregate S(x) as a weighted combination of Fb rate, Sb rate and Db rate. Afterwards, we combine all S(x) into single total anomaly score for the whole system as a harmonic mean. Pre-trained set of parameters expires over a certain period. In our experiment, this period is set to one hour. After expiration, parameters undergo new automatic tuning phase. The general simplified algorithm is presented in Fig. 2.

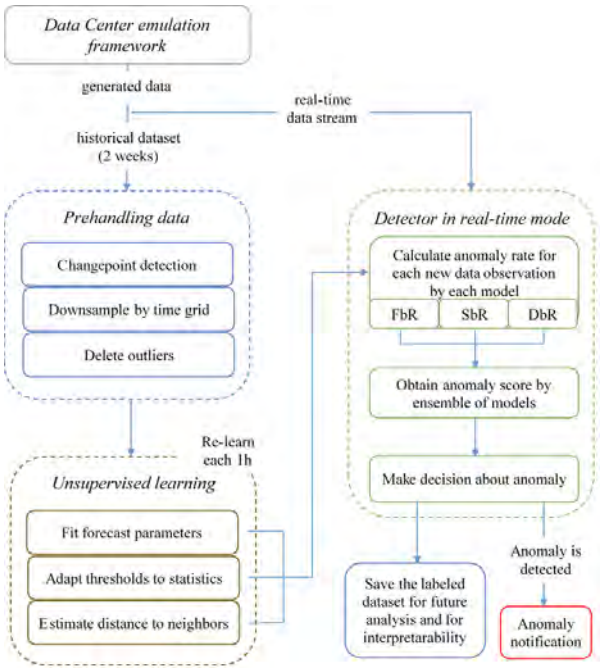

Fig. 2. Simplified algorithm

IV. EXPERIMENTAL STUDY

This section explains the experimental framework for simulating of workload and to approbating proposed novelty detection approach. During the experiment, we generated several data streams of historic and monitoring.

A Experimental framework.

We conducted an experiment within an internal local environment that was created specifically for this study. The environment consists of several virtual machines (VM) to simulate part of data center environment: two front ends (VM1 and VM2) that share single balancer node, several nodes that produce synthetically generated load, and three additional nodes, detailed below, that form an anomaly detection cluster assembled with several open source technologies.

Nodes in the anomaly detection cluster are: queue node, which handles streaming data; real-time and non-real-time modules both run on a compute node; database node serves for storing results. Parallelization for real-time and non-real-time modules has been limited to four threads per node for this experiment.

Queue and database nodes both work in micro batches mode, though this is just a matter of configuration from the point of view of the expected workload.

Virtual machines VM1 and VM2 are frontends that receive emulated workload with synthetically generated API requests. The workload is distributed by working hours to produce a realistic data set. Besides, some synthetic tasks run in a background to produce small load spikes.

We have full control over the API emulation process. Generated workload profile allows us to generate a response stream with a required mode of monitoring data, thus we can evaluate the performance of our detector under certain pre-designed conditions.

For four days prior to our experiment, we have generated a regular profile of disturbing signals to allow history module to adjust to "usual" regime when the experiment will begin. We have split our experiment into two stages to receive two streaming sets of monitoring data. In this way, monitoring data generated under specific workload patterns will represent different workload conditions.

The first scenario assumes both VMs workloads are in stable mode. Numbers of API requests per each period are set according to a regular profile, so all monitoring indicators should remain stable.

The second scenario emulated unstable workload mode is more complex. A number of requests to VM1 and VM2 at first drops to zero for half an hour and for the next half an hour increases in more than two times, if compared to a regular load for this time slot. Besides, part of the time after 17:45 VM2 has been switched off from balancing. This resulted in a slightly higher load to VM1, not anomalous, though. All changes were instant. A number of requests for both scenarios are shown in Fig. 3. Grey area at Stage 2 lies within 16:30-17:30 time interval, when we provoke our VMs to response with a shift in workload as well as expect our system to detect anomalous behavior.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

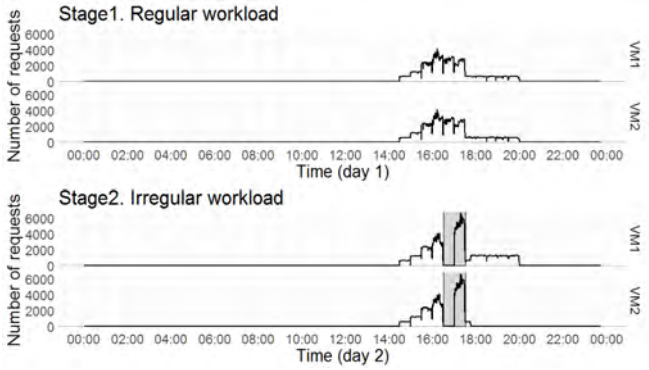

Fig. 3. Number of requests per minute

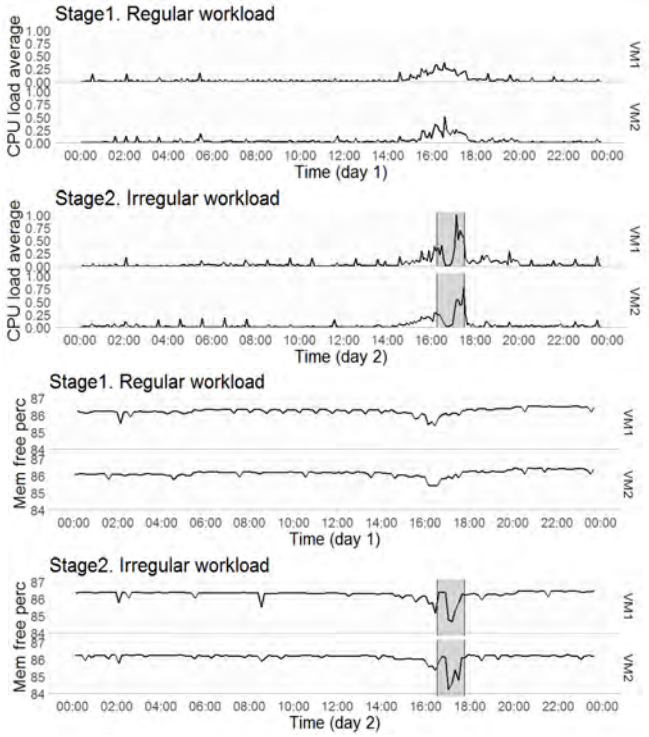

Fig. 4. Corresponding telemetry data

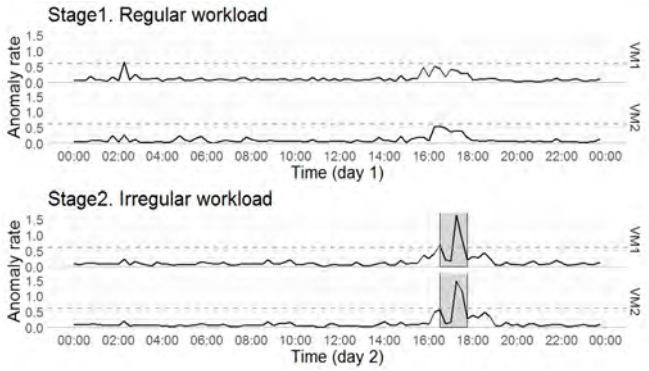

Fig. 5. Convolution of anomaly rate

B Generated data stream.

In our experiment, we limit a list of possible metrics to two telemetry time series, namely CPU load average (CPUla) and free memory percentage (FMp). As mentioned above, our purpose is not to find the best metrics but rather to prove the concept.

The periodicity of observed values is 5 minutes for CPUla and 9 minutes for FMp. For the experiment, we downsampled time series to 15 minutes intervals. The unsupervised historic module produces three streams according to the number of ensemble inputs (Fb, Sb, Db). The fourth stream is the stream of the observed real-time telemetry (monitoring) data. Merged together in a real-time module, all streams form a fast parallel transformation pipeline that calculates anomaly score and forms an output stream of scores along with some interim calculations. This output stream sinks to a database for visualization, further analysis and interpretability.

In such a manner, we obtained data implemented as 4 data streams with numerous instances. Each time historic module processes over 1000 observations to produce forecast and statistics for the next 4 points. Snapshots of observed system behavior and calculated anomaly rates are shown in Fig. 4 and Fig. 5 correspondingly. Grey area on Figure 4 (Stage 2, both metrics) is a period of time when an expert sees some anomalies in CPUla of FMp (16:15-17:30) if compared to regular behavior (Stage 1). Grey area on Figure 5 (16:30-17:45) is a time slot when anomaly is expected to be detected.

C Mannually labeled classes.

Since we are pursuing to develop a tool for conducting an unsupervised anomaly detection, we do not need any previously labeled anomalies to implement our detector. However, to estimate the accuracy of the proposed detection algorithm, two classes of labels were applied: a class label False (no anomaly) is given by default, class label True (some anomalies) is assigned based on expert opinion as described above. It is necessary to emphasize once again that these labels are not used in the run-time of our anomaly detection algorithm.

V. RESULTS AND ESTIMATION OF THE APPROACH

For decision-making purposes whether the changes in mode denote a novelty, we have chosen the threshold to be equal 0.6, which stands for “approximately two-thirds of all diagnostics should point out that anomalous rates have been detected”.

It should be noted, our experiment missed the drop in CPU load caused by the drop in requests designed to be one kind of novelty, but it perfectly revealed unusual behavior expressed in the form of increased resource usage (see Fig. 4).

F1 score estimated for this experiment is 0.625 (precision is 0.833, recall is 0.5). Despite the fact that the estimates are not higher than 0.7, the conducted experiment has shown the viability of our approach and has highlighted directions for further study and improvement. More details on this one can find in the next section.

VI. CONCLUSIONS AND FUTURE WORK

The main contributions of the conducted study are:

(1) A new modular approach to novelty detection has been developed based on the main idea of ensemble model training with splitting into real-time and non-real-time modules

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

(2) Experimental research has been implemented on the data center simulation framework under different scenarios of workload within the proposed method.

(3) Performance assessment, including precision and recall, of the proposed approach under experimental conditions shows an appropriate performance in both indicators.

Nevertheless, it should be noted that the current study has been carried out by a numerical experiment, this causes some limitations. In particular, it is advisable to conduct an advanced research of some theoretical issues:

- What kind of machine learning models should be included in the ensemble detector at the training stage on historical data to improve a quality of real-time detection?
- What kind of anomaly score function will give the best assessment of accuracy and sensitivity for the proposed approach?
- What length of historical data will be the best?
- Which frequency is optimal for retraining the models (once a day, every 3 hours or even dynamically chosen interval, instead of a predefined interval)?

Besides, the proposed approach, suitable for the streaming data anomaly detection, has been successfully validated but has not been compared with any of the existing methods. A comprehensive study of performance of the proposed approach versus other methods would be a good next step in this research.

ACKNOWLEDGMENT

We thank Oracle Corporation for supporting this research project and especially Maryna Chukhray and Andriy Rabochiy for their discussions, feedback and helpful suggestions.

Safe Harbor Statement. The following is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

REFERENCES

[1] M. Dias de Assunção, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," Journal of Network and Computer Applications, vol. 103, pp. 1–17, 2018.

[2] M. Harvan, T. Locher, and A. C. Sima, "Cyclone: Unified Stream and Batch Processing," in 2016 45th International Conference on Parallel Processing Workshops (ICPPW), Philadelphia, PA, USA, pp. 220–229, 2016.

[3] W. Li, D. Niu, Y. Liu, S. Liu, and B. Li, "Wide-Area Spark Streaming: Automated Routing and Batch Sizing," IEEE International Conference on Autonomic Computing (ICAC), Columbus, OH, USA, pp. 33–38, 2017.

[4] K. Vidyasankar, "On Atomic Batch Executions in Stream Processing," Procedia Computer Science, vol. 98, pp. 72–79, 2016.

[5] C. Klein, B. Donnellan, and M. Helfert, Eds., Correlation-Model-Based Reduction of Monitoring Data in Data Centers, Setúbal: SCITEPRESS - Science and Technology Publications Lda, 2016.

[6] P.-Y. Chen, S. Yang, and J. A. McCann, "Distributed Real-Time Anomaly Detection in Networked Industrial Sensing Systems," IEEE Trans. Ind. Electron., vol. 62, no. 6, pp. 3832–3842, 2015.

[7] S. Y. Shin and J. C. Maldonado, Novelty detection algorithm for data streams multi-class problems. Coimbra, Portugal, ACM, March 18-22, 2013.

[8] D. Hong, D. Zhao, and Y. Zhang, "The Entropy and PCA Based Anomaly Prediction in Data Streams," Procedia Computer Science, vol. 96, pp. 139–146, 2016.

[9] C. C. Olson, K. P. Judd, and J. M. Nichols, "Manifold learning techniques for unsupervised anomaly detection," Expert Systems with Applications, vol. 91, pp. 374–385, 2018.

[10] Sajjad Kamali Siahroudi, Poorya Zare Moodi, and Hamid Beigy, "Detection of evolving concepts in non-stationary data streams: A multiple kernel learning approach," Expert Systems With Applications, pp. 187-197, 2018.

[11] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," Neurocomputing, vol. 262, pp. 134–147, 2017.

[12] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," Future Generation Computer Systems, vol. 75, pp. 187–199, 2017.

[13] E. Yu and P. Parekh, "A Bayesian Ensemble for Unsupervised Anomaly Detection," [Online] Available: http://arxiv.org/pdf/1610.07677v1.

[14] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," Information Fusion, vol. 37, pp. 132–156, 2017.

[15] Z. Ding and M. Fei, "An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window," IFAC Proceedings Volumes, vol. 46, no. 20, pp. 12–17, 2013.

[16] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09, Paris, France, p. 139, 2009.

[17] K. Noto, C. Brodley, and D. Slonim, "Anomaly Detection Using an Ensemble of Feature Models," (eng), IEEE International Conference on Data Mining, pp. 953–958, 2010.

[18] N. Golyandina and A. Korobeynikov, "Basic Singular Spectrum Analysis and forecasting with R," Computational Statistics & Data Analysis, vol. 71, pp. 934–954, 2014.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua