Toward an Efficient Real-Time Anomaly Detection System for Cloud Datacenters

Ricardo Dias*‡, Leopoldo Alexandre. F. Mauricio† and Marcus Poggi*
* Departamento de Informática - Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, Brazil
† Grupo Globo - Globo.com, Brazil
Emails: {rdias,poggi}@inf.puc-rio.br, {ricardo.dias,leopoldo}@g.globo

***Abstract*—Anomaly detection in streaming data of cloud datacenter environments requires efficient real-time systems and algorithms. This paper proposes the Decreased Anomaly Score by Repeated Sequence (DASRS) algorithm, which normalizes time series values and counts each sequence to generate anomaly scores as a function of the number of times they appear. We also propose and implement the Sophia anomaly detection system. Sophia is a big data modular streaming processing system implemented in the Globo.com cloud datacenter. DASRS achieves the best-in-class score calculated by Numenta Anomaly Benchmark (NAB) framework. Besides, it is the fastest and uses the least memory among the state-of-the-art algorithms included in NAB. Results from a live application show that Sophia provides an accurate real-time anomaly detection service.**

I. Introduction

Building an efficient real-time anomaly detection system to monitor a cloud datacenter [1] is a big challenge. Frequent software and hardware upgrades, typical of cloud computing, can cause constant behavioral changes in metrics processed by anomaly detection algorithms. Therefore, efficient algorithms must automatically adapt to pattern changes without requiring human intervention to avoid accuracy problems. Moreover, anomaly detection algorithms must have low complexity and reduced execution time, along with reduced memory and CPU usage, even when analyzing large volumes of data.

Every day, more and more data needs to be analyzed, making the detection time a critical aspect [2]. Security attacks that exploit vulnerabilities in applications, for instance, frequently produce atypical discrete disk I/O, network traffic variations, or changes in memory usage patterns. These malicious anomalies can cause irreparable failure and damage when the time needed to identify them is too long [3]. Consequently, robust anomaly detection architectures are required to collect and process real-time cloud metrics and also provide efficient visualization mechanisms to report anomalies.

This paper proposes the Decreased Anomaly Score by Repeated Sequence (DASRS) algorithm. DASRS is a continuously learning algorithm that normalizes time series values and counts each sequence to generate anomaly scores as a function of the number of times they appear. We evaluate DASRS and, using the Numenta Anomaly Benchmark (NAB) [4], we compare its accuracy with state-of-the-art algorithms [5]. Evaluation results show that the proposed algorithm achieves the best-in-class NAB score. Besides, it is the fastest and uses the least memory among the state-of-the-art algorithms included in NAB. We also propose and implement the Sophia anomaly detection system. Sophia is a modular Big Data streaming processing system implemented in the Globo.com cloud datacenter, which collects and analyzes metrics in real-time using DASRS, providing an accurate anomaly detection service. Results from this live application show that the minimal setup of the proposed system can analyze 2,800 virtual machine metrics in real-time, 14,000 per minute.

The rest of the paper is organized as follows. In Section II, we present the proposed algorithm. Section III details the Sophia anomaly detection architecture. Evaluation results and the use case of Sophia, analyzing Globo.com streaming data, are in Section IV. In Section V discusses related work, and Conclusions are in Section VI.

II. The Proposed Algorithm

The DASRS algorithm, proposed in this paper, identifies and counts the sequences of normalized values that appear in a time series and generates an anomaly score as a function of the number of times it identifies each sequence. The first time DASRS identifies a given sequence, the returned score is as high as possible because the algorithm interprets it as a new behavior. Otherwise, the returned score decreases as the number of times a given sequence is found. Let X_t be a time series with the observations $x_1, x_2, \ldots$. A sequence of X_t is a subset of X_t consisting of consecutive elements, for example, $x_i, \ldots, x_j$, with $i < j$. The normalization applied by DASRS consists of transforming the observation value into an integer between 0 and a normalization factor that we call θ. The equation below represents the operations performed on x_i to get its normalized value (x'_i): $x'_i = \left\lfloor \frac{x_i - min_X}{max_X - min_X} \times \theta \right\rfloor$, (equation 1), where x_i is the input observation ($x_i \in \mathbb{R}$), min_X and max_X are respectively the smallest and highest possible observation values of X_t. θ represents the normalization factor, x'_i is the normalized value of x_i, $x'_i \in \mathbb{N}$ and $0 \le x'_i \le \theta$. We normalize observed values to limit the number of distinct sequences without changing the main characteristics of a time series. As the normalization reduces the number of distinct sequences, we can increase the performance of the anomaly detection algorithm.

This work has been sponsored by CNPq, CAPES and Grupo Globo.

Annex to ISBN 978–3–903176–28–7 © 2020 IFIP

We calculate the anomaly score, taking into account the current normalized sequence and the number of times that sequence appeared in the past. The number of elements in the sequence is defined by a parameter, which we decide to call $sequenceSize$. The last $sequenceSize$ normalized elements of the time series defines the current normalized sequence. We store in a data structure the number of times each sequence appears throughout the time series processing. Then, we calculate the anomaly score using the following equation: $score = \frac{1}{occurrences}$ (equation 2), where $occurrences$ represents the number of times the current sequence appears. At the beginning of the processing of a time series, the algorithm generates high scores of anomalies because, until now, it has found the sequences a few times. For this reason, the initial processing time of a series should be considered a training period or a probationary period. Therefore, the anomalies identified during this period do not trigger alarms. Equation 2 does not generate the final anomaly score. As explained in [6] and [7], many times, a dataset analyzed registers unpredictable behaviors caused by noise or the random nature of some metrics, generating a large number of false positives. To address this, we developed two versions of DASRS. The first, **DASRS Rest**, defines a period after identifying an anomaly, in which the final anomaly score should be smoothed. DASRS Rest understands very close anomalies as part of the same phenomenon. Therefore, after identifying an anomaly, the following scores are decreased for a period defined by the $restPeriod$ parameter. The $getRestScore$ function of Algorithm 1 (Lines 25-31) shows how the $restPeriod$ parameter is used to attenuate the final anomaly score.

On the other hand, we create **DASRS Likelihood** version from the NuPIC library [8]. DASRS Likelihood uses the anomaly likelihood metric, which is a measure of the probability of the current state being anomalous based on the history of the raw anomaly scores calculated by the algorithm. A detailed explanation of the Likelihood score calculation is in [6] and [7]. Algorithm 1, Lines 10-11, shows the $getLikelihoodScore$ that implement the NuPIC library to find anomalies. We set the $useLikelihood$ parameter to false in DASRS Rest and use it as true in the DASRS Likelihood version. Moreover, we use the $pointAnomaly$ function in DASRS Likelihood (Algorithm 1, Lines 12-24) to identify point anomalies [9]. Like the Numenta code, we also set the final score to 1 when we find a point anomaly, regardless of the values calculated for the raw score and the $Likelihood$ score. Thus, we can compensate the $Likelihood$ technique limitation when calculating the probability that the raw score truly is an anomaly.

Table I shows the step-by-step execution of DASRS Rest and DASRS Likelihood while processing a time series. The $Time$ and X columns are the scalar values of a time series over time. X' is the normalized value of X. $Sequence$ is the last 2 ($sequenceSize$) elements of X'. $Occurrences$ is the number of times the $Sequence$ has been found. $RawScore$ is the raw anomaly score calculated by the algorithm. $RestScore$ is the final score returned by the DASRS Rest algorithm after

Algorithm 1: DASRS

```
Input:
  • $minValue$: minimum value between observations
  • $maxValue$: maximum value between observations
  • $\theta$: amount of normalized values
  • $sequenceSize$: sequence window size
  • $restPeriod$: period that the anomaly score is attenuated after detecting an anomaly
  • $useLikelihood$: boolean value indicating whether to use likelihood score
  • $x_i$: current observation
Output: $AnomalyScore \in$ [0,1]: anomaly score
1  Function init($minValue, maxValue, \theta, sequenceSize, restPeriod$):
2      $restWeakenFactor \leftarrow 0$
3      $normInputSequence \leftarrow []$
4      $sequences \leftarrow \{\}$
5      $anomalyLikelihood \leftarrow nupic.algorithms.anomaly_likelihood()$
6      $minVal \leftarrow null$
7      $maxVal \leftarrow null$
8  Function getRawAnomalyScore($occurrences$):
9      return $1/occurrences$
10 Function getLikelihoodScore($currentScore$):
11     return $anomalyLikelihood.computeLogLikelihood(currentScore)$
12 Function pointAnomaly($value$):
13     $isPointAnomaly \leftarrow False$
14     if $minVal! = maxVal$ then
15         $tolerance \leftarrow maxVal - minVal \times 0.05$
16         $maxExpected \leftarrow maxVal + tolerance$
17         $minExpected \leftarrow minVal - tolerance$
18         if $value > maxExpected$ or $value < minExpected$ then
19             $isPointAnomaly \leftarrow True$
20     if $maxVal = null$ or $value > maxVal$ then
21         $maxVal \leftarrow value$
22     if $minVal = null$ or $value < minVal$ then
23         $minVal \leftarrow value$
24     return $isPointAnomaly$
25 Function getRestScore($score$):
26     if $restWeakenFactor > 0$ then
27         $score \leftarrow score/restWeakenFactor$
28         $restWeakenFactor \leftarrow restWeakenFactor - 1$
29     else if $score \geq 1$ then
30         $restWeakenFactor \leftarrow restPeriod$
31     return $score$
32 Function handleRecord($x_i$):
33     $normInpVal \leftarrow \lfloor \theta \times (x_i - minValue)/(maxValue - minValue) \rfloor$
34     $normInputSequence.append(normInpVal)$
35     if $len(normInputSequence) < sequenceSize$ then
36         return 0
37     if $normInputSequence \in sequences$ then
38         $sequences[normInputSequence] += 1$
39     else
40         $sequences[normInputSequence] = 1$
41     $occurrences = count(normInputSequence\ in\ sequences)$
42     $normInputSequence.pop(0)$
43     $rawAnomalyScore \leftarrow getRawAnomalyScore(occurrences)$
44     if $useLikelihood$ then
45         $anomalyScore \leftarrow getLikelihoodScore(rawAnomalyScore)$
46         if $pointAnomaly(x_i)$ then
47             $anomalyScore \leftarrow 1$
48     else
49         $anomalyScore \leftarrow getRestScore(rawAnomalyScore)$
50     return $anomalyScore$
```

mitigating the value of the raw score by the $restPeriod$ period. $LikelihoodScore$ is the final score calculated by the NuPIC library.

TABLE I
DASRS REST AND DASRS LIKELIHOOD PROCESSING STEP BY STEP

Time	X	X’	Sequence	Occurrences	Raw Score	Rest Score	Likelihood Score
0	10,5	0	-	-	0	0	0,030
1	15,3	0	(0, 0)	1	1	1	0,030
2	23,2	1	(0, 1)	1	1	0,5	0,030
3	18,2	0	(1, 0)	1	1	1	0,030
4	27,8	1	(0, 1)	2	0,5	0,5	0,038
5	22,2	1	(1, 1)	1	1	1	0,080
6	20,0	0	(1, 0)	2	0,5	0,25	0,035
7	13,4	0	(0, 0)	2	0,5	0,5	0,051
8	19,0	0	(0, 0)	3	0,33	0,33	0,171
9	24,1	1	(0, 1)	3	0,33	0,33	0,301
10	20,9	0	(1, 0)	3	0,33	0,33	0,167
11	28,1	1	(0, 1)	4	0,25	0,25	0,539
12	22,9	1	(1, 1)	2	0,5	0,5	0,263
13	15,5	0	(1, 0)	4	0,25	0,25	0,492
14	10,4	0	(0, 0)	4	0,25	0,25	0,587
15	16,8	0	(0, 0)	5	0,2	0,2	0,300
16	24,0	1	(0, 1)	5	0,2	0,2	0,207
17	90,0	7	(1, 7)	1	1	1	0,162
18	28,9	1	(7, 1)	1	1	0,5	0,112
19	26,6	1	(1, 1)	3	0,33	0,33	0,112

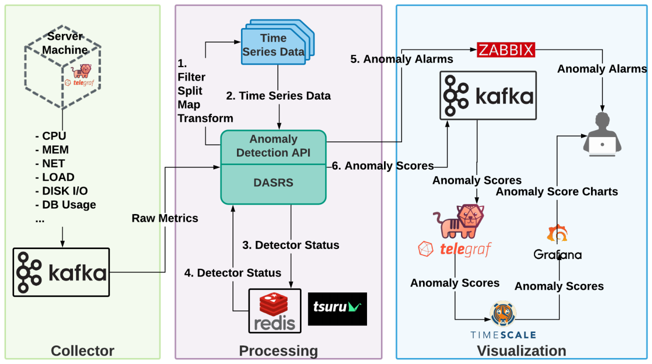

Fig. 1. An agent installed on each monitored machine sends application and operating system metrics to Apache Kafka, which forwards them to the anomaly detection API. The API processes those raw metrics to create the time series that are then analyzed by DASRS. The API sends the detector results to the visualization module, generates security alarms in Zabbix, and stores the state of the detector. Thus the training phase does not need to be repeated. Telegraf stores these results in the TimeScale database, and Grafana displays the results in graphs.

III. The Sophia Anomaly Detection Architecture

Figure 1 illustrates Sophia Anomaly Detection (SAD) architecture. We develop a collector, a processing, and a visualization module. We use Telegraf and Apache Kafka to build the collector module. Telegraf is an agent for collecting, aggregating, processing, and sending metrics. Through plugins, it integrates with a wide variety of systems. Apache Kafka is a distributed streaming platform for a large volume of data flow so that we can use it to publish, store, process, and consume messages in real-time. We install the Telegraf agent on each monitored machine to get metrics such as CPU, networking, memory, disk I/O, disk size, OS load, database connections, among others. The Telegraf agent sends the collected metrics to Apache Kafka that temporarily stores them to be consumed by other applications.

The processing module analyzes metrics, instantiates the anomaly detection algorithm (DASRS), and sends the anomaly scores and alarms to the visualization module. The Anomaly Detection API reads the raw metrics from Apache Kafka. Since the Kafka's messages are complex data structures, the API parses them. As illustrated in interaction 1, the API performs filter, split, mapping, among others operations on the raw metrics to identify each type of metric collected, and assemble the data structure expected for the anomaly detection algorithm, that is, a time series (Interaction 2). If Sophia's anomaly detection API receives CPU and network metrics from three Virtual Machines (VMs), for instance, it will create six different time series: VM1-CPU, VM2-CPU, VM3-CPU, VM1-Network, VM2-Network, and VM3-Network. Besides, to process multiple metrics of several VMs, Sophia's API uses multiple instances of the anomaly detection algorithm DASRS. One DASRS instance for each time series. Therefore, in the example above, the anomaly detection API will instantiate 6 DASRS instances, and each of them will receive the streaming values collected from its time series in an orderly manner.

DASRS algorithms generate an anomaly score for each observation in the time series analyzed. Then, the API compares the anomaly score with a threshold to decide whether to generate an anomaly alarm or not. However, no one anomaly alarm is generated during the training period. Interaction 3 shows that we store the status of each DASRS instance in a Redis database. When analyzing a time series for the first time, Sophia's anomaly detection API creates a new instance of the DASRS algorithm. Contrarily, when analyzing the same time series for a second time, Sophia's anomaly detection API loads the DASRS instance from the previously-stored (Interaction 4) detector's status. This way, it can continue the analyzes of that particular time series, even after a processing interruption for maintenance reasons.

Sophia's anomaly detection API generates anomaly alarms in a Zabbix server (Interaction 5) and sends anomaly scores to the Apache Kafka in the visualization module (Interaction 6). The Telegraf agent in this module reads the anomaly scores from Kafka and sends them to a TimeScale database to be persisted. System administrators can get all anomaly alarm generated by DASRS through Zabbix. They also can analyze the anomaly detections through Grafana. TimeScale is a relational database designed to store time-series data, providing automatic time partitioning; Zabbix is an open-source monitoring software tool, and Grafana is a platform for visualizing and analyzing metrics through graphs. We implement the SAD system using docker containers to increase the architecture portability and flexibility. Thus, using the Tsuru Platform as a Service, we quickly deploy, scale, and manage either the API as well as DASRS.

IV. Evaluation and Results

In this section, we describe the experiments carried out to compare the performance of DASRS Rest and DASRS Likelihood with several anomaly detection algorithms in the literature. We use the Numenta Anomaly Benchmark (NAB), that is an open-source framework, to evaluate the algorithms [5][1]. It also presents the results of the Sophia anomaly detection system when collecting and analyzing metrics in real-time in the Globo.com cloud datacenter.

Nab Score Benchmark - In the first stage, the calculation of the native NAB score is used to measure the performance of the algorithms when searching for anomalies in the NAB and Yahoo-A1Benchmark datasets. The application profiles used to calculate the final score are: standard, reward low FP, and reward low FN. The results, with the final scores of each evaluated algorithm, are illustrated in tables II and III, sorted according to the scores obtained in the standard profile. Analysis of the results shows that both algorithms proposed in this work perform very well. The DASRS Likelihood algorithm has the best score when the NAB dataset is processed while DASRS Rest has the fourth-best score. However, when the Yahoo-A1Benchmark dataset is processed, DASRS Rest gets the best score, and DASRS Likelihood gets the fourth. The reason that the Likelihood version performed better on the NAB dataset and the REST version performed better on the Yahoo-A1Benchmark dataset is due to the characteristics of the time series in each dataset. Most of the time series in Yahoo-A1Benchmark has little variation over time, except in the moments where anomalies occur. On the other hand, the NAB dataset contains many time series with unpredictable behavior, which can make many anomalies to be identified incorrectly by the algorithms. Since the Likelihood function performs well in this scenario, DASRS Likelihood gets better results in the NAB dataset. The same Likelihood function is also used in the Numenta algorithm. For the same reason, this algorithm performed very well in analyzing the NAB dataset, but it was unable to repeat the excellent results in the Yahoo-A1Benchmark dataset.

TABLE II
NAB SCORES USING THE NAB DATASET

Detector	Standard Profile	Reward Low FP	Reward Low FN
DASRS Likelihood	70.3	65.5	73.9
CAD OSE	69.9	67.0	73.2
Numenta	69.7	62.3	74.3
DASRS Rest	66.4	60.2	70.4
Earthgecko Skyline	58.2	46.2	63.8
KNN CAD	58.0	43.4	64.8
Relative Entropy	54.1	48.0	57.9
Windowed Gaussian	40.1	23.9	47.7
Etsy Skyline	35.7	27.1	44.5
Bayesian Changepoint	17.7	5.1	32.3
EXPoSE	16.4	3.5	26.9

TABLE III
NAB SCORES USING THE YAHOO-A1BENCHMARK DATASET

Detector	Standard Profile	Reward Low FP	Reward Low FN
DASRS Rest	67.1	62.6	72.3
CAD OSE	62.7	57.5	67.3
KNN CAD	52.9	41.9	58.8
DASRS Likelihood	52.5	45.1	57.7
Windowed Gaussian	48.9	39.7	54.8
Relative Entropy	48.9	41.4	53.6
Numenta	46.2	41.5	50.3
Bayesian Changepoint	41.1	23.9	53.7
Etsy Skyline	36.5	25.9	43.5
Earthgecko Skyline	35.8	34.6	38.0
EXPoSE	13.8	10.9	26.0

[1]We have also obtained results for Classic Performance Metrics (Precision, Recall, and F1-Score), and runtime of a time series with 10,000 observations, but do not present them due to the number of pages limitation.

Runtime Benchmark - In the second stage, we evaluate the time that each algorithm took to process the NAB and Yahoo-A1Benchmark datasets. The averages of 5 test rounds are in tables IV and V. The execution time is measured in seconds. Evaluation results show that the DASRS Rest algorithm is the fastest in the two scenarios tested. DASRS Likelihood gets the second-best time.

TABLE IV
RUNTIME (IN SECONDS) USING NAB DATASET

Detector	Round 1	Round 2	Round 3	Round 4	Round 5	Mean
DASRS Rest	23	24	25	22	24	23,60
DASRS Likelihood	64	66	68	63	63	64,80
Windowed Gaussian	45	49	49	45	47	47,00
Relative Entropy	62	65	66	63	63	63,80
Bayesian Changepoint	145	152	147	150	145	147,80
EXPoSE	222	230	228	224	218	224,40
CAD OSE	289	295	303	287	282	291,20
earthgecko Skyline	564	568	691	575	547	589,00
Numenta	696	768	738	791	688	736,20
KNN CAD	716	728	812	833	700	757,80
Etsy Skyline	16365	16072	16589	16497	16524	16409,40

TABLE V
RUNTIME (IN SECONDS) USING YAHOO-A1BENCHMARK DATASET

Detector	Round 1	Round 2	Round 3	Round 4	Round 5	Mean
DASRS Rest	7	8	8	8	9	8,00
DASRS Likelihood	10	11	11	11	11	10,80
Windowed Gaussian	18	13	13	12	14	14,00
Relative Entropy	26	19	18	17	18	19,60
Bayesian Changepoint	25	22	20	20	21	21,60
CAD OSE	56	53	52	54	53	53,60
Earthgecko Skyline	74	61	61	61	61	63,60
KNN CAD	79	69	65	70	65	69,60
EXPoSE	81	66	70	62	63	68,40
Numenta	212	188	181	192	178	190,20
Etsy Skyline	787	639	625	684	652	677,40

Memory Benchmark In the third step, we evaluated how much memory each anomaly detection algorithm used to process from 1,000 to 10,000 observations. Figure 2 shows that the CAD OSE, Etsy Skyline, and Earthgecko Skyline algorithms present a constant increase in memory usage that are higher than those of the other evaluated algorithms. DASRS Rest, on the other hand, has no variation in memory usage, which is also the smallest (0.87 MB) among all the anomaly detection algorithms analyzed.

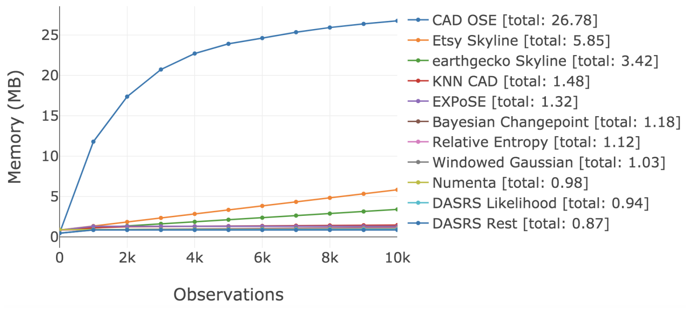

Fig. 2. Memory usage to process 10,000 observations.

The Sophia Anomaly Detection System Evaluation - We install the Telegraf agent in 2,800 machines with a collection interval time of one minute. We configure the Sophia API to analyze five metrics collected by the agent: CPU User, CPU System, CPU Idle, CPU IO Wait, and Used Percent Disk. In this scenario, the Sophia API analyzes 14,000 metrics per minute. The approximate processing time for each observation is only 0.001348 seconds. The system does not produce any delay. The elapsed time between the collection of the metric and the visualization of anomaly alarms in Zabbix is less than 1 minute, including the flush interval of the Telegraf agent. The same goes for visualizing anomaly scores in Grafana. The Redis database consumes only 12 MB of memory to persist the 14,000 DASRS Rest instances. The Sophia anomaly detection system analyzes approximately 20,000,000 metrics in real-time per day, where 0.005% on average are assertively classified as anomalies. Table VI shows the number of observations analyzed and anomalies detected by Sophia System in the period between 2019-06-30 and 2019-07-10.

TABLE VI
OBSERVATIONS ANALYZED AND ANOMALIES DETECTED BY SOPHIA

Day	**Observations**	**Anomalies**
2019-06-30	8123973	0
2019-07-01	19482918	786
2019-07-02	19555331	3039
2019-07-03	19604947	2334
2019-07-04	19651201	2023
2019-07-05	19703618	2139
2019-07-06	19874881	1476
2019-07-07	19874880	1825
2019-07-08	19879536	1191
2019-07-09	19908169	1050
2019-07-10	19942131	1241

V. Related Work

The problem of anomaly detection has a rich literature. [4] implements the Numenta Anomaly Benchmark framework to compare anomaly detection algorithms [7], [10]. NAB dataset has multiple types of time series with labeled anomalies, and it implements a scoring system that rewards the earlier detection algorithms. [7] propose the Numenta HTM detector, which uses Hierarchical Temporal Memory (HTM) for anomaly detection. HTM is a machine learning technology, which uses error prediction calculations to measure how efficiently the model can avoid false positives. HTM models the time series sequences so that, for each instant t, the algorithm makes several predictions for the value of the data in $t + 1$. Then, the algorithm compares such predictions with the actual value to decide whether it can be considered normal or anomalous. Thus, the HTM neocortex-inspired algorithm [11] implements a probabilistic metric defining how anomalous the current state is, based on the prediction history.

[12] develops the open-source contextual anomaly detector CAD-OSE. CAD-OSE's anomaly scores are a function of transformations made in input observations that include normalization and binary representation of metrics in structured data called context. The algorithm identifies anomalies comparing elements of the current context with previous ones to decide how much an observed input value corresponds or not to a new context. [13] implements the EXPected Similarity Estimation (EXPoSE) algorithm, which efficiently calculates the similarity between new data points and the regular distribution of data. The algorithm calculates the probability that a data point is reasonable, based on its similarity to previous points without assuming an underlying data distribution.

In contrast, the HP research team developed an anomaly detection technique based on Tukey and Relative Entropy statistics [14]. They compare observations of a time series data against multiple null hypotheses. When an observation data does not match any of the existing hypotheses, it is considered anomalous. Then, the algorithm creates a new hypothesis to match the identified anomalous observation. Otherwise, it is declared non-anomalous as long as the accepted hypothesis occurs often enough to become a pattern.

VI. Conclusions

This paper proposes the anomaly detection algorithms DASRS Rest and DASRS Likelihood. DASRS Rest has the best NAB score when we analyze the Yahoo dataset, and it has one of the four best scores when we analyze the NAB dataset. Moreover, DASRS Rest is the algorithm that most quickly analyzes all datasets used. Further, DASRS Rest registers the smallest (0.87 MB) memory usage. We also propose and implement the Sophia anomaly detection system that collects and analyzes metrics in real-time in the Globo.com cloud datacenter. DASRS Likelihood ranks first using the NAB dataset, but due to the better overall results, we decide to use DASRS Rest in Sophia to analyze the Globo.com streaming time series in real-time. Sophia, using DASRS Rest, provides an accurate anomaly detection service.

References

[1] G. G. Claps, R. B. Svensson, and A. Aurum, "On the journey to continuous deployment: Technical and social challenges along the way," *Information and Software technology*, vol. 57, pp. 21–31, 2015.

[2] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile networks and applications*, vol. 19, no. 2, pp. 171–209, 2014.

[3] L. A. F. Mauricio, M. G. Rubinstein, and O. C. M. B. Duarte, "ACLFLOW: An NFV/SDN Security Framework for Provisioning and Managing Access Control Lists," in *2018 9th International Conference on the Network of the Future (NOF)*, Nov 2018, pp. 44–51.

[4] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 38–44.

[5] N. Singh and C. Olinsky, "Demystifying numenta anomaly benchmark," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1570–1577.

[6] S. Ahmad and S. Purdy, "Real-Time Anomaly Detection for Streaming Analytics," *CoRR*, vol. abs/1607.02480, 2016.

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

[8] Numenta, Inc., "Nupic API Documentation," 2019, Accessed: 2019-04-27. [Online]. Available: http://nupic.docs.numenta.org/stable/index.html

[9] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278 – 288, 2016.

[10] M. Munir, S. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," *IEEE Access*, vol. PP, pp. 1–1, 12 2018.

[11] J. Hawkins and S. Ahmad, "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex," *Frontiers in Neural Circuits*, vol. 10, p. 23, 2016.

[12] M. Smirnov, "Contextual Anomaly Detector," June 2016, Accessed: 2019-05-01. [Online]. Available: https://github.com/smirmik/CAD

[13] M. Schneider, W. Ertel, and F. T. Ramos, "Expected Similarity Estimation for Large-Scale Batch and Streaming Anomaly Detection," *CoRR*, vol. abs/1601.06602, 2016.

[14] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, May 2011, pp. 385–392.