

DC Health: Node-Level Online Anomaly Detection in Data Center Performance Data Monitoring

Walter Lopes Neto and Itamir de Morais Barroca Filho(✉)

Metropole Digital Institute, Federal University of Grande do Norte Natal, Natal, Brazil
walter.lopes@ifrn.edu.br, itamir.filho@imd.ufrn.br
http://www.imd.ufrn.br, http://www.ufrn.br

Abstract. Data centers are critical environments for the availability of technology-based services. Aiming at the high availability of these services, performance metrics of nodes such as Virtual Machines (VM) or VMs clusters are widely monitored. These metrics, such as CPU and memory utilization, can show anomalous patterns associated with failures and performance degradation, culminating in resource exhaustion and total node failure. Thus, early detection of anomalies can enable remediation measures, such as VM migration and resource reallocation, before losses occur. However, traditional monitoring tools often use fixed thresholds for detecting problems on nodes and lack automatic ways to detect anomalies at runtime. In this sense, machine learning techniques have been reported to detect anomalies in computer systems with online and offline approaches. Thus, this work aims to propose and evaluate the DC Health application, pursues to anticipate the detection of anomalies in data center nodes. For this, this research was conducted from i) Systematic Literature Review previously performed, ii) problem modeling from real VM data and iii) DCH evaluation using the prequential method in 6 real-world datasets. Preliminary results showed that DCH excelled in constant memory usage and detection accuracy above 75% in the worst of the 5 cases and accuracy of 90% at best when applied in the real node datasets and an accuracy of 85% on the shuttle dataset. As a continuation of this research, it is expected to develop a case study with data center operators and the evaluation of the tool in a large volume of nodes.

Keywords: Anomaly detection · Monitoring · Data center · Fault-detection · Machine learning · Half-Space-Trees · Data mining · Predictive analytics · VM-level anomalies · Node-level anomaly

1 Introduction

Network infrastructure monitoring systems for data centers are essential to maintain observability and high availability of operation in these critical environments. In this sense, existing and widely used systems for this purpose, such as

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13381, pp. 632–649, 2022.
https://doi.org/10.1007/978-3-031-10548-7_46

Zabbix [20], enable detailed monitoring of physical and virtual servers as well as applications. Among these collected data, the health and performance data of the nodes, such as logs and metrics, can present sets of anomalous patterns associated with failures as reported by Ibidunm et al. [14], Guan et al. [11] and [26]. The accumulation of anomalies of different natures can be a significant cause of incremental performance degradation, culminating in resource exhaustion or even the total failure of the host [26]. In this way, techniques that enable the analysis of these patterns aiming at the early detection of performance anomalies have become increasingly present in the literature and the industry as a report for Notaro et al. [25]. Early detection of anomalies in data center nodes, such as performance degradation, can make it possible to apply corrective measures before damage occurs. Examples of corrective measures are virtual machine migration and resource allocation adjustment [23]. The classic machine learning approach to this type of problem is to split the stored data into datasets, train the model on that data, i.e. it is required early access to the entire dataset, and then use it for prediction. Although this classic approach has been used to detect performance anomalies in works in the literature, as presented in Sects. 2 and 3, it faces challenges such as naturally suffering from the aging of the model as reported by Xiao et al. [37]. In this sense, the field of online machine learning has been established to address this problem. The online learning field seeks to carry out the learning process from potentially infinite data streams [29]. Thus, unlike classical methods, the model is exposed only once to each sample and must be ready to perform prediction at any time, in addition, to dynamically adapting to changes in streams, for example, due to concept drift. In this sense, the term concept drift refers to a common challenge in the machine learning area, which occurs when changes in the forecast target variable lead the model to lag, impairing the accuracy over time [36]. Online anomaly detection has been applied in several domains such as network intrusion detection, financial fraud detection, fault detection, and anomaly detection in the context of public health, among others [5]. Among the application domains, we highlight fault detection, which is a field that has attracted interest for more than 30 years and focuses on subareas such as fault tracking, and symptom monitoring, reporting of detected errors, and auditing undetected errors [29]. In this context, online failure prediction refers to identifying whether a failure will occur shortly (during runtime), based on an assessment of the current state of the monitored system [29]. Concerning critical environments such as data centers, online forecasting methods are especially interesting, since they focus on the aspect of short-term forecasts being useful to prevent potential disasters and limit the damage caused by computer system failures. By incorporating system performance metrics at runtime, the prediction of failure occurrence can be on the order of seconds or minutes [29]. However, because they have big data characteristics, dynamically dealing with failure data from monitoring systems in data centers is challenging. Sometimes these failures are unpredictable and it becomes complex and expensive to identify their root cause [26] or individually model the factors that potentially

generate each type of failure, being in some cases unfeasible. Additionally, predicting failures in systems where there is no labeled data (in this case, where each instance used to train the model is previously classified as failure or normal) is a challenge. Real-world computer systems, especially newly deployed systems, do not always have fault-labeled data available or have such data reliably [12]. In this context, generally, the volume of data representing normal behavior is significantly greater than that of failure data so, in that case, it is possible to use unsupervised techniques to reveal hidden patterns in the data, and thus determine also conditions that deviate from the normal operating condition (anomalies). In this sense, the performance degradation of hosts and services can be an early indication of resource exhaustion, which causes failures. In this way, the unsupervised anomaly detection approach is better suited. Regarding node performance and fault identification and performance degradation, current infrastructure monitoring systems such as Zabbix and Cactie generally make use of fixed thresholds [35]. Examples of node performance items monitored with fixed thresholds are CPU or memory usage above a certain fixed value or free disk space below a certain fixed value. This detection format culminates in making monitoring systems overly dependent on manual adjustments by operators. This context becomes especially problematic when it comes to complex and dynamic systems such as data centers, and a greater extent in cloud data centers. In this sense, identifying thresholds related to an anomaly is more complex than using a single value, like 75% CPU usage, for example, as reported by Mendoza et al. [19]. Based on the exposed elements, the following research question will be addressed in this work: **How to early identify anomalies in datacenter nodes, such as virtual machines or virtual machine clusters, from performance measurements, such as CPU and memory utilization, at runtime?** Thus, the objective of this article is to propose and evaluate the DC Health application, which is a solution aimed at anticipating the detection of node anomalies in an online manner. The DC Health application, presented in this work, is a solution aimed at detecting anomalies in an online manner based on performance data from data center nodes, collected as a data stream. The tool is based on the framework skmultiflow [21] (A Python library focused on stream data) and the implementations of Half-Space-Trees, well known and state-of-the-art anomaly detection algorithm for data streams [34]. Introduced by Tan et al. [32] as a fast anomaly detector for evolving data stream.

2 Background

The anticipation of anomaly detection is potentially advantageous for the operation of data centers from the point of view of increasing the availability of nodes and services that depend on them. Anomaly detection can be performed in advance, by taking advantage of intrinsic characteristics of machine learning techniques when training a model and seeking to improve it over time. Thus, as new data is collected (with each new instance of the data stream), the model can be updated automatically.

Data Centers and Cloud Services Infrastructure. Data center (DC) refers to an information technology infrastructure aimed at housing information systems (IS) to minimize all types of interruptions. DCs main components are computers, servers, storage, cooling, and power systems according to Saha et al. [28]. In this sense, the prediction of the failure propensity of data center nodes can be performed based on the analysis of historical performance and failure data and can improve the availability of services in several ways, among them through the allocation of virtual machines in healthy nodes or even migration without disconnecting from the client (live migration) [17]. However, dealing with the volume of performance metrics data can be challenging. Monitoring tools such as Zabbix [20] often collect a large number of performance metrics from the data center and nodes. Among these metrics, we can highlight the level of CPU utilization (at the system or user level), CPU idle time, memory utilization level, and volume of I/O operations, among others [11]. Data centers are usually monitored through network infrastructure monitoring systems, which are essential to maintain the observability and high availability of these environments. Examples of specialized monitoring systems include Zabbix, Nagios, and proprietary systems. In general, these systems collect health and performance data from nodes (logs and metrics) that may present anomalous patterns associated with failures. Which, when accumulated, can cause performance degradation and lead to resource exhaustion and failures.

Anomaly Detection in Data Centers. Anomaly Detection (AD) refers to the problem of identifying patterns in the data that do not conform to the expected behavior according to Chandola et al. [5]. Among the several areas of research that address AD problems, we can highlight the area of machine learning, focusing on the application domain of fault detection from the perspective of performance data, the focus of this work. In this sense, regarding AD in data centers, we considered in this research the hypothesis that in environments designed for high availability such as data centers, anomaly data is significantly rarer when compared to the volume of data of normal operation. Thus, obtaining accurate data, where each instance is clearly labeled as normal or anomalous, and which is representative of all types of behavior, is generally prohibitively expensive [5]. Generally, data labeling to obtain the training dataset is done by human experts or based on the modeling of specific and not very generalizable cases, such as failures in HD's [3,13], and [40]. Additionally, anomalous behavior is dynamic in nature, so new types of uncharacterized anomalies often arise in the model training process. This factor is relevant in the context of reliability, since anomalous events, although very rare, can translate into catastrophic scenarios in certain cases [5]. In this way, the provision of mechanisms that enable the early detection of these events is relevant for the efficient operation of data centers.

Online Anomaly Detection. Online Anomaly Detection (DAO), similarly to online failure prediction, aims to predict the occurrence of failures in a system based on the current state of the system and during runtime [29]. Thus, DAO,

especially applied to sequential data, is a relevant topic in fields such as intrusion and failure detection [6]. In this sense, online learning is superior to offline learning when it comes to applying sequential data, allowing online prediction with low memory usage [8]. In the context of anomaly detection, machine learning can be used to enable the statistical model of the normal behavior of the data to be learned, and so this model can predict future values and identify anomalies, as new data is received online, comparing the predicted with the actual values [19]. In this sense, works like [27,38], and [31] apply online anomaly detection, so that each of the test instances receives an anomaly score, as well as helping to update the model, which evolves incrementally. Approaches and methods aimed at online anomaly detection can be classified into statistical-based, cluster-based, nearest-neighbor-based, and isolation-based. Of these, we highlight the isolation-based methods, which differ from the others in that they do not compute the distance or density of the dataset. Isolation-based methods, like HS-Trees, prove to be scalable and suitable for datastream [33]. In this sense, the sliding window strategy has been used as a way to adapt offline methods for data in-stream format as in [7] and [22]. The application of the sliding window strategy allows online fault detection, for example, to adapt to variations in system behavior. This is because this technique assumes that recent data is more important than old data and thus discards the oldest samples from the window [41].

Unsupervised Anomaly Detection (UAD). Unsupervised Anomaly Detection (UAD) refers to the challenge of applying anomaly detection to data that has not been previously labeled. Unlike standard classification methods, UAD is generally applied to take into account only the internal structure of the data [10]. In this case, each instance can be associated with only one attribute (univariate) or with multiple attributes (multivariate) [5]. In scenarios such as production data centers, it is natural that system metrics change over time due to several factors, such as configuration changes, demand characteristics, software updates, among others. In these cases, models need to identify the new normal behavior in an unsupervised way in order to adapt [1]. Techniques that need pre-labeled data, such as supervised classification-based methods, are generally not suitable for anomaly detection and continuous learning [1]. Among the unsupervised techniques aimed at detecting anomalies in datastream, we can highlight Half-Space-Trees (HS-Trees), introduced by Tan et al. as a fast one-class anomaly detector for evolving data streams [32]. The method used works based on a Haflt-Space-Trees committee in which each HS-Trees is composed of a set of nodes that capture a given set of items. After executing HS-Trees, the identification of an instance as being or not being an anomaly can be done in two ways: **i)** Through the predict function, in which the sample is labeled with a specific value if it is considered anomalous, or **ii)** Based on the visual analysis of the output of the decision function, in which a threshold can be drawn.

Challenges Related to Early Identification of Anomalies in Data Center Nodes. Among the challenges related to the early identification of anomalies in data center nodes, we can mention: **i) Faults of different origins** Nodes of complex systems such as data centers can have their failures of very different origins (hardware, software, bugs, among others). Identifying the root cause or individually modeling each factor that potentially generates the failure can be very complex and costly [39] and **ii) Lack of labeled data in data centers** Predict failures in systems where there is no data previously labeled as failures. That's usually the case for data centers.

3 Related Works

The field of anomaly detection in data streams has been extensively researched and has applications in several domains such as detection of financial fraud, network attacks, in the field of individual and public health and fault detection. In regard of fault detections, several recent works have addressed techniques and tools aimed at detecting anomalies online. The development of DH Health was inspired as a result of some recent work with the approaches described as follows.

A supervised anomaly detection strategy was proposed by Liu et al. [18], applying Self Organizing Maps (SOM). They demonstrate that anomalies within VMs can be detected in certain regions of the infrastructure using unified modeling to detect the node performance. Differently, we propose the application of a different anomaly detector to each node, so each detector can learn from the behavior of an specific node monitored, without being affected by the others.

The work presente by Sirbu et al. [30] is a study of cluster data from Google dataset, collected from a 12K-node cluster with the goal of building and evaluating predictive models for node failures. Although the work presents relevant contributions, the application is designed to be applied in a supervised manner. In contrast to the method proposed by the authors, our work proposes an unsupervised approach, due to the restriction of the data collected in our research. This occurs as a result of the data provided does not have clear markings indicating which samples refer to faulty nodes and which refer to normal behavior.

A method extending HS-Trees was proposed by Wetzing et al. [35] to detect anomalies on IoT-Gateway devices. The authors propose an unsupervised anomaly detection algorithm based on the concept of HS-Trees. The proposed method extends the HS-Trees through online adaptive thresholds for resourced constrain IoT-Gateway devices. In contrast to these, our work is based on the use of virtual machines in the data center, in a context where there is usually no severe restriction of resources.

The Streaming Half-Space-Trees (HS-Trees) was presented by Tan et al. [32] as an anomaly detector with a focus on detection speed for one-class anomalies applied to the data stream. The presented algorithm works in an unsupervised way, so it only needs normal data for training. In the evaluation, the method presented a favorable performance in terms of detection accuracy and runtime performance, when compared to state-of-the-art algorithms. Also applying online

anomaly detection, the work by Togbe et al. [33] presents a quick survey of existing anomaly detection methods for the data stream and provides an implementation of iForestASD, an adaptation of iForest for data streams. Our work, in a similar manner to Tan et al. [32] and Tobe et al. [33], performs the evaluation of the HS-Trees method using the prequential method. However, our work differs in the sense of using real performance data from data center nodes.

4 DC Health

DC Health is proposed as a solution aimed at anticipating the detection, in an online manner, of anomalies in data center nodes.

4.1 DC Health Overview

DC Health (DCH) is a system whose objective is realize early detection of anomalies in data center nodes. For this, the tool consumes data streams composed of performance samples of data center nodes, such as usage level CPU, memory, and network. These samples are collected by the DCH, from network monitoring systems like Zabbix. It is noteworthy that currently, the DCH is in the prototype stage of development. The DCH uses the Half-Space Trees (HS-Trees) algorithm to perform anomaly detection. Thus, within the scope of DCH, anomaly detection is carried out in an unsupervised online manner, that is, without the need to train the system with pre-labeled data. In addition, due to the use of HS-Trees, the DCH has constant memory consumption requirements for each instance of the anomaly detection model. These constant memory requirements are achieved due to the working format of HS-Trees, which processes each sample only once, that is, testing and updating the model for each sample. DCH is designed to perform anomaly detection at a host level in order to be applied to physical servers, virtual machines or virtual machine clusters nodes. In this way, the DCH monitors certain nodes of interest and, by learning from their behavior considered normal, makes it possible to alert datacenter operators in advance if any abnormal events are detected. The application of these techniques makes it possible to learn the behavior considered normal for the data centers performance metrics, in an online manner, making it possible to alert anomalous behaviors in advance, since it makes the use of thresholds more flexible, making them more dynamic, as well as enabling alerts based on abnormal behavior.

The Fig. 1 presents the overview of DC Health. In this overview, it is possible to identify the multiple instances of online anomaly detectors that aim to detect anomalies in advance based on the metrics of each node of interest.

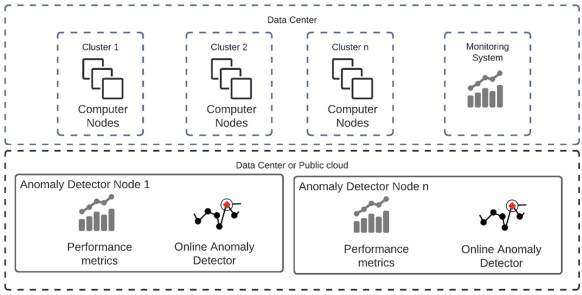

Fig. 1. DC Health overview.

4.2 DC Health Architecture

DC Health is composed of software components called modules. Each component implements a certain set of system functionality. The main modules currently implemented in the DCH prototype are the Stream Collection Module (MCS), the Online Anomaly Detection Module (MDAO) and the Messenger Module (MM). In general, anomaly detection occurs when the Anomaly Detection Module Online understands that there is an anomalous event in a certain node of the datacenter, which is being observed. In this sense, the observed node refers to a host (for example a cluster of virtual machines), which has been previously associated with a Detector. In this way, when using the system, the datacenter operator can create and remove Anomaly Detectors associated with the nodes. Thus, if an anomaly is detected, the operator is alerted via the Messenger Module. Finally, for the DCH to work, it must be hosted on a Linux operating system based on $\times 86$ host. The collection of performance data from the nodes is carried out through the Stream Collection Module (MCS), which communicates with the datacenter monitoring server, such as Zabbix, and starts to receive samples of metrics from performance. Examples of performance metrics are the level of CPU, memory, and network utilization. The Fig. 2 presents the basic architecture of DC Health through an example of a working flow. Observe the data flow where the performance metrics of the nodes are collected and stored by the monitoring server (Number 1). In parallel, the anomaly detection module instantiated for a given node consumes the collected instances one by one from the monitoring server and performs anomaly detection (Number 2). Then, if an anomalous event is detected, the operator is alerted (Number 3). During this process, with each data sample consumed, the model is updated. The operator can perform basic query and configuration operations on the DCH through access using the command line interface (Number 5), as well as query report of anomalies detected by host.

4.3 DC Health Components

Online Anomaly Detector Module (MDO). MDO is the module that performs the main DC Health online anomaly detection function. MDO works online

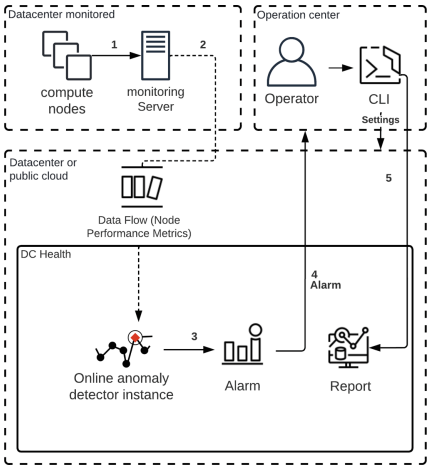

Fig. 2. Basic architecture of DC Health.

and is based on registered nodes. For each node of interest registered with DC Health, a new online anomaly detector is created and keeps track of the flow of performance data streams from this node. Several detectors can be created so that each detector is composed of an instance of HS-Trees that monitors the datastream. For each new data sample associated with a given node, an anomaly score is associated with the instance and a label of normal or anomalous. Additionally, the sample is used for an incremental update of the model. In case the sample is identified as anomalous, the MOD sends, via the Message Module, and identified anomaly alert to the operator, with the related context. The MOD also logs all activities on disk. The entire analysis is considered for a given sample window.

Stream Collection Module (MCS). The MCS communicates with Monitoring Systems through queries via API. After the query, the MCS performs data extraction and transformation and sends the samples in data stream format to the Anomaly Detection Module Online. In the current implementation, MCS supports the Zabbix monitoring system [20].

4.4 Methodology

As a proof of concept, we performed anomaly detection assessment on real data using the DC Health prototype. The phases of the methodology adopted in this research were briefly presented in Fig. 3. Initially, a Systematic Literature Review was conducted (phase 1) to help understand the current state and future trends of the application of machine learning techniques in data center monitoring, aimed at early detection of failures [24]. Based on the techniques and strategies

identified in the SLR, such as unsupervised strategies aimed at detecting anomalies based on the degraded behavior of resource consumption in the [16] and [2] targeting data center, an iterative phase of data analysis and solution development was conducted. In the analysis, historical performance data of nodes from a real data center in production were used to model the problem as a Machine Learning problem. After modeling the problem, during the development of the application, activities such as data collection, exploration and preparation were carried out. In the third phase 3 of development, and evaluation of the detection performance was carried out. The proposed solution was evaluated (phase 3) based on the comparison of anomaly detection results obtained through DC Health against performance metrics of virtual machines from the real data center in production at Instituto Metropole Digital, a unit of the Federal University of Rio Grande do Norte, Brazil (IMD), as well as well-known datasets from the area of machine learning. The evaluation was based on the prequential method (interleaved test-then-train), an alternative to the traditional holdout, commonly adopted in the literature because it is specific to the datastream. The method consists of using each sample to make a prediction (i.e. testing the model), and then the same sample is used to train the model (partial fit). In this way, the model is always tested on samples not yet used in training. Each sample is evaluated in order of arrival and immediately becomes inaccessible (natural from a data stream), serving for testing and training.

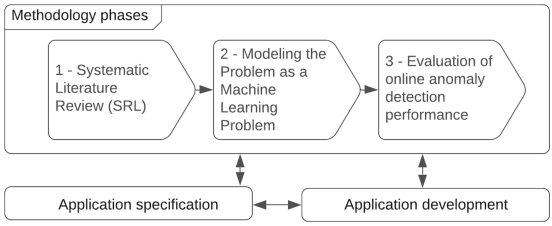

Fig. 3. Methodology phases.

Overview. We propose to approach the problem as an online anomaly detection problem, for that we use the HS-Trees that works in an unsupervised way and can deal with samples from the stream of unlabeled data. In this way, the model is built based on performance data at runtime, collected by the monitoring tool and representing the nodes in their normal behavior (without failures).

In this research we use the implementation of the algorithm HS-Trees presented by [32], as an ensemble of HS-Trees, a fast anomaly detector for data streams that requires only normal data for training. Also, HS-Tree and works well when (as presented in the context of this work), the anomalous data are rare [32]. The method based on HS-Tree was chosen because it is feasible with the use of unclassified data (in an unsupervised way) in addition to being able to use only normal data for training, to create the anomaly detection model.

This choice was made due to the problem of insufficient failure data for training due to natural limitations of the data center data collection process, identified earlier. There are two outputs of the anomaly detection algorithm: The label for each instance, identifying whether the instance is normal or anomalous, and the value of Anomaly Score, which indicates the degree of abnormality.

Methodology Steps. The methodology of this research was developed in 3 steps, as described below:

Step 1 - Systematic Literature Review (SRL). A systematic literature review was conducted following the method of [15]. The following research questions (RQ) were addressed: **RQ1:** What are the main characteristics of the machine learning algorithms and models being applied to monitoring data center infrastructure? **RQ2:** What types of data are being used in the proposed models? (eg sensor data, images, event logs). **RQ3:** What are the main challenges in applying machine learning to data center infrastructure monitoring? **RQ4:** What are the most common types of failures identified in data center infrastructure monitoring?

Step 2 - Modeling the problem as a Machine Learning problem. The approach chosen for the machine learning problem was to consider performance anomalies as potential failures and failure precursors and approach anomaly detection in an online and unsupervised way. With each new instance received, the model is updated and labels the given instance as normal or anomalous. The instances, once labeled, can be used as a source for fault detection through supervised classification methods.

Step 3 - DC Health evaluation. The evaluation was performed by comparing performance metrics during anomaly detection for different existing datasets and for data from a real data center environment. As described in [4], the key criteria for evaluating classifiers in the data stream context (depending on specific requirements) are: processing time, memory usage, adaptability, and prediction performance. In this work, we will use prediction performance. As described by Gama et al. [9], performance measurements on data streams can be performed using prequential evaluation. The evaluation method will be briefly described: **1)** Choice of datasets. As detailed earlier, a reference dataset was chosen along with 5 new experimental datasets. **2)** Choice of evaluation parameters. For the sliding windows, different sizes (32, 64, 128 and 256) were adopted. **3)** Choice of the detection algorithm execution discipline. With each new sample added to the window, the detection algorithm is executed using the contents of the window as a training set. **4)** Choice of main metric for comparison. The chosen metrics were accuracy. Unsupervised methods do not need labeled data, however, for performance evaluation, labeled data and well-known datasets are generally used [10].

5 Experimental Evaluation

In this section, the procedures used in the evaluation experiment, and the characteristics of the data from the experimental environment will be briefly described.

Datasets. The data used in the experiment came from two sources. The first was the Shuttle dataset, which was chosen because it is commonly used in the literature for the evaluation of anomaly detectors in streaming data. Among the works that applied this dataset, we can mention the work by Ding et al. [7], in a witch, the authors proposed the algorithm iForestASD and evaluate this dataset, and also Tan et al. [32], that authors applied the HS-Trees algorithm. Additionally, this dataset contains known anomalies, being used as a baseline for performance reference. The Shuttle dataset is an unbalanced dataset, which contains 80% of the data associated with one of the classes. Additionally, unlike the second data source, has the class attribute available and used during the evaluation. The second data source was the datasets created based on collecting data from hosts in a real data center during the execution of the experiment. For this, we chose to compute nodes that presented known problems related to performance and unavailability (Processor load is too high and Unreachable for more than 10 min). The data collection period was 1 year (01/01/2020 to 12/31/2021). Node 1 to 5 datasets were created containing the following attributes: CPU Load (Average 1 min), Ping (milliseconds), Ping (0 or 1), Memory, and Amount of free space in the file system. Additionally, the data were normalized. Regarding the data nature of the experimental datasets, multivariate data with continuous numerical attributes were considered. Regarding the relationship between the instances, sequential instances were considered.

The Table 1 briefly presents the characteristics of the datasets in terms of a number of instances and attributes. To facilitate the reproducibility of this research, the datasets node 1 to 5 was made available.[1]

Table 1. Summary of the characteristics of the datasets adopted for the experiment.

Summary of the characteristics of the adopted datasets		
Dataset	Number of instances	Number of attributes
Shuttle	49097	9
Node 1 to Node 5	1698	6

Description of the Experimental Environment. Aiming at evaluating the feasibility of the proposal, we defined a controlled environment for the experiment. The experimental environment configuration used to host DC Health was: One server with 2 CPUs: Intel(R) Xeon(R) CPU E5-4610 v4 @ 1.80 GHz, 4 GiB of memory, using de OS CentOS Linux release 7.9.2009 (Core), with a disk of 50 GB (SCSI) and Network interface as Gigabit Ethernet.

[1] https://github.com/walter-ln/datasets_dch.

Execution of the Experimental Evaluation. The experiments aimed at validating the DC Health anomaly detection model were performed as described in the following steps: **1. Experimental environment definition** - the experimental environment was defined as described in this section; **2. Data collection for the experiment** - data collection was performed from the Extraction Module (MET) and Treatment of the DC Health prototype, with the target being the monitoring system (Zabbix) of the data center of the Instituto Metrópole Digital (IMD/UFRN). The MET, according to its configuration, requested all available data for each of the pre-defined hosts. After collection, MET performed preprocessing and created a dataset file for each of the 5 hosts with known issues, as described. In-depth details on how the MET works can be described in Sect. 4.3; **3. Loading of experimental data and conversion into data stream** - the reading of the experimental datasets was performed by the Stream Collection Module (MCS) and this transferred the data in data stream format to the Detection Module of Anomalies Online (MDAO); **4. Model training and evaluation, for each set of parameters** - a total of 24 experiments were performed. For each of the 6 data sets, the model was run considering the sample window value varying between the values 32, 64, 128, and 256 samples. At each sample window value, the model was instantiated, trained over 1000 samples, and performed the prequential evaluation. The prequential method consists of using each sample to test the model and then the same sample is used to train the model, i.e. to perform the partial fit. In this way, the model is always tested on samples not yet used in training. Each sample is evaluated on a first-come, first-served basis and is immediately inaccessible for testing and training. After the evaluation of each model, the result metrics, in this case, in particular, the accuracy, were saved in auxiliary text files.

6 Results and Limitations

In this research, the DC Health system, aimed at detecting anomalies in data center nodes, was presented. Table 2 presents the results obtained for each combination of sample window sizes, with w varying between 32, 64, 128, and 256 samples. The results presented show that DC Health obtained an accuracy of 75% in the worst of the 5 cases and 90% accuracy in the best case. These accuracy values occurred when applied to datasets from real data center nodes (datasets from nodes 1 to 5). As shown in Table 2, the same system obtained an accuracy of 85% in the dataset Shuttle, with window size w = 64 samples. The preliminary experimental results indicate that the proposed solution achieved satisfactory performance, considering the detection accuracy above 75% for the detection of anomalies in real data, both in datasets from nodes 1 to 5, as well as with well-known data of literature. However, considering the natural trade-off between execution speed and accuracy, and in the context of applying early anomaly detection on datacenter nodes, we understand that a faster, even if less accurate, model is usually more desirable than a more accurate model, but slower in terms of detection. This hypothesis is based on the sense that, in critical

Table 2. Comparison of results from the application of the proposed system in different real-world data sets. For each dataset, window size combinations (32, 64, 128, and 256) were tested on 1000 samples.

Dataset node 1			
Window (w)	Accuracy	Memory (kB)	Execution time (s)
32	0.033033	755201	96
64	0.868869	755202	60
128	0.828829	755204	45
256	0.657658	755207	34
Best	**0.868869**	**755201**	**34**
Dataset node 2			
Window (w)	Accuracy	Memory (kB)	Execution time (s)
32	0.034034	755201	95
64	0.837838	755202	61
128	0.825826	755204	41
256	0.688689	755207	37
Best	**0.837838**	**755201**	**37**
Dataset node 3			
Window (w)	Accuracy	Memory (kB)	Execution time (s)
32	0.034034	755201	88
64	0.479479	755202	62
128	0.827828	755204	41
256	0.720721	755207	35
Best	**0.827828**	**755201**	**35**
Dataset node 4			
Window (w)	Accuracy	Memory (kB)	Execution time (s)
32	0.034034	755201	93
64	0.755756	755202	59
128	0.649650	755204	44
256	0.683684	755206	35
Best	**0.755756**	**755201**	**35**
Dataset node 5			
Window (w)	Accuracy	Memory (kB)	Execution time (s)'
32	0.032032	755201	89
64	0.906907	755202	59
128	0.858859	755204	41
256	0.720721	755206	35
Best	**0.906907**	**755201**	**35**
Dataset shuttle			
Window (w)	Accuracy	Memory (kB)	Execution time (s)'
32	0.7888	154677	1313
64	0.8509	116430	1507
128	0.8298	843094	1670
256	0.7197	580721	1572
Best	**0.8509**	**116430**	**1313**

environments, anomalous events, although very rare, can translate into catastrophic scenarios in certain cases [5]. In this way, the system obtained the best execution time of 34 s for the evaluation of 1000 instances. Finally, as shown in Table 2, the memory consumption requirements of the model remained constant for different sizes of windows w. **Limitations** This study has a number of limitations. The set of hyperparameters used to find the best configuration for the model was identified but was restricted so that other configurations can identify better results. Due to time limitations for the research, the prototype was evaluated only through the metrics of accuracy, processing time, and memory usage. Other comparison metrics were not explored, such as Area Under the Receiver Operating Characteristics (AUROC), Area Under The Curve (AUC), Receiver Operating Characteristics (ROC), F1 measure, Precision, and Recall. Due to the availability of data from the real data center, the evaluation was performed based on data from 5 real data center nodes, with known problems in a given time interval. An evaluation with a quantity with a large-scale volume of nodes could reveal different aspects to be considered.

7 Conclusions and Future Work

In this work, we propose and evaluate the early detection of anomalies through DC Health, which used the HS-Trees algorithm as a basis. This method was used to obtain a data anomaly score on each performance sample of nodes operating in a given data center and was chosen due to characteristics such as online and unsupervised detection. The main contributions of this work are listed as follows: **i)** development and availability of the DC Health solution prototype, aiming to facilitate the early detection of anomalies in datacenter nodes for data center infrastructure operators, **ii)** providing a dataset that compiles node performance data from a real industry data center and **iii)** evaluation of the anomaly detection method through *prequential* method in a real industry datacenter. The availability of a specialized tool for early detection of anomalies in data center hosts provides a set of possibilities with the potential to benefit data center operators, users of ICT services, and organizations that finance datacenters and costs relating to the unavailability of ICT services. Among these possibilities, we can highlight the early detection of failures, the possibility of developing advances such as the automation of diagnosis of the root cause of problems, the automation of remediation actions as well as the minimization and grouping of monitoring alerts. In addition, paths such as the possibility of developing specialized tools for early detection in certain fields of application, such as attack detection, and optimization of resource provisioning, among others. As a continuation of the studies carried out in this research, the following proposals for future work are suggested: **i)** conducting a case study involving the application of DC Health in a real data center and thereby collecting the experience and evaluation of data center operators, **ii)** development of new features in DC Health, which facilitate its use and adoption by datacenter operators. To facilitate adoption and use by data center operators, it is suggested to develop a version based on the web for

the anomaly detection tool, **iii)** evaluation of the tool in a large volume of nodes with greater statistical relevance, as well as identifying the best adjustments of *hyperparameters* for the model, as one of the works futures to be developed, it is suggested the evaluation based on datasets representing more nodes of the data center, **iv)** it is also suggested to extend the functionality of the DC Health tool, aiming to evaluate the early detection of anomalies applied to cloud computing nodes, in hybrid datacenters and cloud data centers, **v)** development of automated mechanisms for diagnosis and remediation and **vi)** we also suggest the development of specialized anomaly detectors in certain fields of application, such as in-network attacks.

References

1. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
2. Ahmed, J., et al.: Automated diagnostic of virtualized service performance degradation. In: NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9. IEEE (2018)
3. Alter, J., Xue, J., Dimnaku, A., Smirni, E.: SSD failures in the field: symptoms, causes, and prediction models. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–14 (2019)
4. Brzezinski, D., Stefanowski, J.: Prequential AUC: properties of the area under the ROC curve for data streams with concept drift. Knowl. Inf. Syst. **52**(2), 531–562 (2017). https://doi.org/10.1007/s10115-017-1022-8
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection for discrete sequences: a survey. IEEE Trans. Knowl. Data Eng. **24**(5), 823–839 (2010)
7. Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proc. **46**(20), 12–17 (2013)
8. Fontenla-Romero, Ó., Guijarro-Berdiñas, B., Martinez-Rego, D., Pérez-Sánchez, B., Peteiro-Barral, D.: Online machine learning. In: Efficiency and Scalability Methods for Computational Intellect, pp. 27–54. IGI Global (2013)
9. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Mach. Learn. **90**(3), 317–346 (2012). https://doi.org/10.1007/s10994-012-5320-9
10. Goldstein, M., Uchida, S.: A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS One **11**(4), e0152173 (2016)
11. Guan, Q., Zhang, Z., Fu, S.: Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems. In: 2011 Sixth International Conference on Availability, Reliability and Security, pp. 83–90. IEEE (2011)
12. Guan, Q., Zhang, Z., Fu, S.: Ensemble of Bayesian predictors and decision trees for proactive failure management in cloud computing systems. J. Commun. **7**(1), 52–61 (2012)
13. Huang, S., Liang, S., Fu, S., Shi, W., Tiwari, D., Chen, H.B.: Characterizing disk health degradation and proactively protecting against disk failures for reliable storage systems. In: 2019 IEEE International Conference on Autonomic Computing (ICAC), pp. 157–166. IEEE (2019)

14. Ibidunmoye, O., Lakew, E.B., Elmroth, E.: A black-box approach for detecting systems anomalies in virtualized environments. In: 2017 International Conference on Cloud and Autonomic Computing (ICCAC), pp. 22–33. IEEE (2017)
15. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. **55**(12), 2049–2075 (2013)
16. Lanciano, G., et al.: SOM-based behavioral analysis for virtualized network functions. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 1204–1206 (2020)
17. Lin, Q., et al.: Predicting node failure in cloud service systems. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 480–490 (2018)
18. Liu, J., Chen, S., Zhou, Z., Wu, T.: An anomaly detection algorithm of cloud platform based on self-organizing maps. Math. Probl. Eng. **2016** (2016)
19. Mendoza, M.A., Amistadi, H.R.: Machine learning for anomaly detection on VM and host performance metrics. Technical report, MITRE Corp. Bedford, MA (2018)
20. Mescheryakov, S.V., Shchemelinin, D.A.: Analytical Overview of Zabbix International Conference 2013 (1(188)), pp. 91–98 (2014)
21. Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-Multiflow: a multi-output streaming framework. J. Mach. Learn. Res. **19**(72), 1–5 (2018). http://jmlr.org/papers/v19/18-251.html
22. Mouratidis, K., Papadias, D.: Continuous nearest neighbor queries over sliding windows. IEEE Trans. Knowl. Data Eng. **19**(6), 789–803 (2007)
23. Nagarajan, A.B., Mueller, F., Engelmann, C., Scott, S.L.: Proactive fault tolerance for HPC with Xen virtualization. In: Proceedings of the 21st Annual International Conference on Supercomputing, pp. 23–32 (2007)
24. Neto, W.L., de Morais Barroca Filho, I.: Failures forecast in monitoring datacenter infrastructure through machine learning techniques: a systematic review. In: Gervasi, O., et al. (eds.) ICCSA 2021. LNCS, vol. 12957, pp. 27–42. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87013-3_3
25. Notaro, P., Cardoso, J., Gerndt, M.: A survey of AIOps methods for failure management. ACM Trans. Intell. Syst. Technol. (TIST) **12**(6), 1–45 (2021)
26. Pellegrini, A., Di Sanzo, P., Avresky, D.R.: A machine learning-based framework for building application failure prediction models. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, pp. 1072–1081. IEEE (2015)
27. Pokrajac, D., Lazarevic, A., Latecki, L.J.: Incremental local outlier detection for data streams. In: 2007 IEEE symposium on computational intelligence and data mining. pp. 504–515. IEEE (2007)
28. Saha, S., Sarkar, J., Dwivedi, A., Dwivedi, N., Narasimhamurthy, A.M., Roy, R.: A novel revenue optimization model to address the operation and maintenance cost of a data center. J. Cloud Comput. **5**(1), 1–23 (2015). https://doi.org/10.1186/s13677-015-0050-8
29. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. ACM Comput. Surv. (CSUR) **42**(3), 1–42 (2010)
30. Sîrbu, A., Babaoglu, O.: Towards operator-less data centers through data-driven, predictive, proactive autonomics. Clust. Comput. **19**(2), 865–878 (2016). https://doi.org/10.1007/s10586-016-0564-y
31. Snyder, D.: On-line intrusion detection using sequences of system calls (2001)
32. Tan, S.C., Ting, K.M., Liu, T.F.: Fast anomaly detection for streaming data. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)

33. Togbe, M.U., et al.: Anomaly detection for data streams based on isolation forest using scikit-multiflow. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12252, pp. 15–30. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58811-3_2
34. Togbe, M.U., Chabchoub, Y., Boly, A., Barry, M., Chiky, R., Bahri, M.: Anomalies detection using isolation in concept-drifting data streams. Computers **10**(1), 13 (2021)
35. Wetzig, R., Gulenko, A., Schmidt, F.: Unsupervised anomaly alerting for IoT-gateway monitoring using adaptive thresholds and half-space trees. In: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), pp. 161–168. IEEE (2019)
36. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996). https://doi.org/10.1023/A:1018046501280
37. Xiao, J., Xiong, Z., Wu, S., Yi, Y., Jin, H., Hu, K.: Disk failure prediction in data centers via online learning. In: Proceedings of the 47th International Conference on Parallel Processing, pp. 1–10 (2018)
38. Yamanishi, K., Takeuchi, J.I., Williams, G., Milne, P.: On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Min. Knowl. Discov. **8**(3), 275–300 (2004). https://doi.org/10.1023/B:DAMI.0000023676.72185.7c
39. Yang, Y., Dong, J., Fang, C., Xie, P., An, N.: FP-STE: a novel node failure prediction method based on spatio-temporal feature extraction in data centers. Comput. Model. Eng. Sci. **123**(3), 1015–1031 (2020)
40. Zeydan, E., Arslan, S.S.: Cloud2HDD: large-scale HDD data analysis on cloud for cloud datacenters. In: 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 243–249. IEEE (2020)
41. Zhang, L., Lin, J., Karim, R.: Sliding window-based fault detection from high-dimensional data streams. IEEE Trans. Syst. Man Cybern. Syst. **47**(2), 289–303 (2016)