PROCEEDINGS OF SCIENCE

Anomaly Detection in Data Center IT Infrastructure using Natural Language Processing and Time Series Solutions

Elisabetta Ronchieri,[a,b,*] **Filippo Pacinelli,**[b] **Luca Giommi,**[a] **Doina Cristina Duma,**[a] **Alessandro Costantini**[a] **and Davide Salomoni**[a]

[a] *INFN CNAF,*
Viale Berti Pichat 6/2, Bologna, Italy
[b] *Department of Statistical Sciences, University of Bologna,*
Via delle Belle Arti 41, Bologna, Italy
E-mail: elisabetta.ronchieri@cnaf.infn.it, filippo.pacinelli@gmail.com, giommi@cnaf.infn.it, cristina.aiftimiei@cnaf.infn.it, davide.salomoni@cnaf.infn.it, alessandro.costantini@cnaf.infn.it

Data centers house IT and physical infrastructures to support researchers in transmitting, processing and exchanging data and provide resources and services with a high level of reliability. Through the usage of infrastructure observability platforms, it is possible to access and analyse data that provide information on data center status enabling the prediction of events of interest. During the last few years, in the context of the main data processing and computing technology research center of the Italian Institute for Nuclear Physics, we have performed a set of studies based on service log files and machine metrics to identify anomalies and define alarm signals. In the present work we aim at validating our previous studies by considering critical scenarios and extending the range and type of monitoring data. With the usage of principal component analysis, clustering techniques, and statistical anomaly detection solutions, we have been able to achieve a faster, almost real-time, detection of anomalies taking into consideration the collection of past events.

As an added value, the relationship between the identified anomalies and the threshold-risk values will be assessed and shown as a dynamic level of risks to be used for predictive maintenance management.

International Symposium on Grids & Clouds (ISGC) 2023
19-24 March 2023
Academia Sinica, Taipei, Taiwan

*Speaker

PoS(ISGC&HEPiX2023)024

1. Introduction

Data centers host IT and physical infrastructures to support researchers in transmitting, processing and exchanging data and provide resources and services with a high level of reliability. Through the usage of infrastructure observability platforms, different kinds of data can be accessed, particularly those related to services that run on both virtual machines and bare mental servers, to predict events of interest. The ability to detect unexpected anomalies, in fact, is of great significance to prevent service degradation, hardware failures, data losses, and complaints from users.

The Italian Institute for Nuclear Physics (INFN) CNAF is a data center that serves more than 40 international scientific collaborations in multiple scientific domains, including high-energy physics experiments running at the Large Hadron Collider in Geneva. CNAF handles a large amount of data with over 1,000 different running services and supports data flowing 24/7. The center already provides a set of pillars, such as connectivity to the cloud and infrastructure for data transmission, and security for data and privacy protection. Within this context, we have performed a set of studies to add the intelligence data pillar to the INFN CNAF main functionalities. This new pillar can be implemented through infrastructure and algorithms in order to extract value from service and physical resources (i.e. logs and monitoring metrics), convert them into useful information (e.g. detecting anomalies), and properly intervene. This might allow for performing predictive maintenance with machine learning (ML) techniques and time series analysis.

Starting from our initial study on log files aimed at building an anomaly dictionary and getting knowledge from files with the application of natural language processing (NLP) solutions and other ML techniques [1], we have continued combining a subset of log files and monitoring data information to detect anomaly patterns involving heterogeneous unstructured data [2]. NLP solutions have been applied to log files to identify anomalies from words and sequences of terms. Good results have been obtained, revealing thousands of anomalies verified by exploiting log-service messages. By defining an ad-hoc clustering algorithm, various types of anomalies at the service level have been identified and grouped together. Furthermore, the adoption of a multivariate time series anomaly detection technique, named JumpStarter [3], enabled us to compute anomaly scores on monitoring data to identify the timeframe where we could overlap services and monitoring data anomalies to perform predictive maintenance analysis.

In the present work, we aim at validating the above-mentioned solutions by considering critical scenarios and extending the range and type of monitoring data. By using error reconstruction techniques based on, but not limited to, principal component analysis (PCA) [4], clustering techniques (such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [5] and K-means [6]), and statistical anomaly detection solutions, we plan to achieve faster and real-time detection of anomalies taking into consideration also the collection of past events. Furthermore, the relationship between the identified anomalies and the threshold-risk values is assessed and shown as a dynamic level of risks to be used for predictive maintenance management. The source code developed for the proposed approach is released as open source, so that it can be easily used by other centers. It has to be considered that training and related inference may vary depending on the amount of data provided by the data center.

The remainder of this paper is organised as follows. Section 2 summarizes papers where both log files and monitoring metrics are used. Section 3 and Section 4 detail data sources and methods,

respectively. Section 5 introduces the defined approach to tackle the anomaly detection problem at INFN CNAF. Section 6 presents and discusses the results. Finally, section 7 provides conclusions and plans for future works.

2. Related Works

Our study aims at performing predictive maintenance by identifying anomalies on the basis of heterogeneous data (i.e., log files and monitoring metrics) that provide us with qualitative and quantitative information about the state of machines and services.

In the literature, there exist few works about the detection of anomalies with the usage of both log data and monitoring metrics. Nti *et al.* [7] develop a multi-source information-fusion stock price prediction framework based on a hybrid deep neural network architecture. They use 6 heterogeneous data sources, including indices values from the stock market, articles from journals, and tweets to develop a matrix of features derived from textual information. Lee *et al.* [8] develop a model to estimate the risk of bidding projects in urban areas by combining numerical metrics and textual reviews. They use the uncertainty information, stored in the unstructured text data. In both of these studies, the data fusion process has been done successfully through the identification of common features: the availability of finance-related textual data is enough to be able to look for data in the required moment in time [7], while every textual information has been assigned to the numerical one [8].

Giommi *et al.* [9] try to develop a predictive maintenance model on the basis of INFN CNAF log files and monitoring metrics related to the StoRM service [10] by exploiting supervised ML techniques. They build a ML model to provide the health status of the system, every 15 minutes, by considering a subset of log files produced by the StoRM service. Instead, our problem is also unsupervised, working with data that are specific to the INFN CNAF data center, whose observations are not anomaly labeled. In our study, we have developed a solution based both on the binary representation of anomaly and non-anomaly observations and, as an added value, on an anomaly score for log messages.

3. Data Sources

In this study, we have worked with different types of data (both textual and numerical), collected in the time range June 6, 2020 - July 21, 2021. The considered data refer to thousands of machines, each one with its own peculiarities in terms of hosted services and its usage within the data center (e.g. user machine, service host, farming, and storage). Data belong to two different categories: log files and monitoring metrics.

3.1 Log files

Log files are important sources of information as they represent the current state of system services and their related processes. Each log file is produced by a distinct service running on a given machine. Each log message contains semi-structured text [11]: it is composed of a sequence of static fields and dynamic fields, i.e. a set of strings that do not change from one event occurrence to another and strings assigned at run-time, respectively. Figure 1 shows an example of a log

message produced by the ALRT service: the ALRT CLI_MAIN Failing Disk and Reason strings represent the static fields, the others are the dynamic fields.

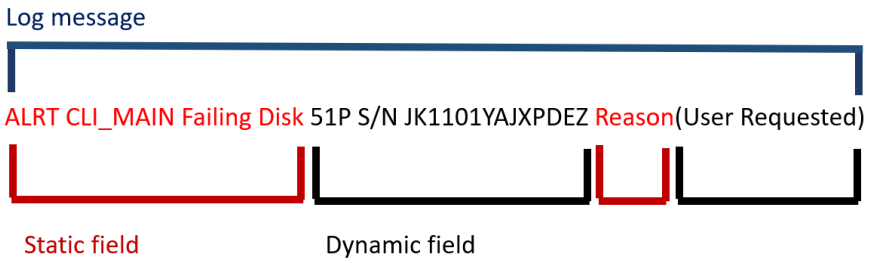

Figure 1: An example of a log message from the ALRT service.

On the same machine, usually, there are several services that store their status in corresponding log files. The occurring events are traced in these files as messages that can be used to analyze and debug system failures. The seriousness of each event is identified by the level of logging included in the message, such as debug, info, and warning: some messages can be more serious than others, but still without blocking the currently running service, like debug; other messages make the service impossible to work, such as alert, fatal error, failure, and alarm. The logs are often highly verbose, making it difficult to understand their meaning. Moreover, the log files have different non-standard formats, making it complex to parse them without customization. In our study, 40 kinds of log files have been considered related to the various services running on 1,258 distinct machines.

Table 1 shows an example of the alert (ALRT) service log file once turned into a comma-separated value (csv) format. The variables are: *date*, *time* and *timestamp* expressing when the instance has been registered in the local time zone; *hostname* and *ip* columns providing information on the machine and network on which the event occurred, respectively; *process_name* representing the name of the service (usually corresponding to the log file name); *msg* reporting the textual message.

Table 1: The ALRT service log file turned into the csv format.

date	*time*	*timestamp*	*hostname*	*ip*	*process_name*	*msg*
2020/08/11	11:57:17	1597139837.0	ddn-04-a.cnaf.infn.it	*	ALRT	ALRT CLI_MAIN Failing Disk 51P S/N JK1101YAJXPDEZ Reason(User Requested)
2020/07/08	14:36:39	1594211799.0	ddn-04-a.cnaf.infn.it	*	ALRT	ALRT AVR_MON Left Power Supply Failure
2020/07/08	14:36:39	1594211799.0	ddn-04-a.cnaf.infn.it	*	ALRT	ALRT AVR_MON Left Side AC Line Low
2020/07/27	16:24:23	1595859863.0	ddn-04-a.cnaf.infn.it	*	ALRT	ALRT CLI_MAIN Failing Disk 19G S/N JK1101YAJSDYXV Reason(User Requested)
2020/07/25	12:24:08	1595672648.0	ddn-04-a.cnaf.infn.it	*	ALRT	ALRT DC_REC Failing Disk 42P S/N JK1101YAKAB69V Reason(IO Time-out)

3.2 Monitoring metrics

Monitoring metrics are numerical data that represent different statistics, typically used to provide information about the health status of machines. They are obtained by using Linux commands that provide statistics belonging to three categories: the actual *load* of the machine, and its average (*load_avg*) over multiple time frames (i.e., 1, 5 or 15 minutes); the *memory usage*; the *central processing unit* (CPU) usage and *input/output* (I/O) (i.e., *io_stat*) rate of data. Load averages [12] are often provided at the last minute, the last five minutes, and the last fifteen minutes. There are many metrics related to the memory usage [13] of a system on which processes are running (see Table 2). IoStat [14] is another command which allows checking more general metrics about the current status of the system (see Table 2). In our study, 18 metrics have been considered.

Table 2: A subset of memory usage and IoStat metrics.

Category	Metric	Description
Memory usage	total	Total installed memory
Memory usage	used	Memory currently in use by running processes
Memory usage	free	Unused memory
Memory usage	shared	Memory shared by multiple processes
Memory usage	buffers	Memory reserved by the operating system to be allocated as buffers when processes need them
Memory usage	cached	Recently used files stored in RAM
Memory usage	buff/cache	Buffers + Cache
Memory usage	available	Estimation of available memory for starting new applications
IoStat	user	CPU% used executed at user level
IoStat	nice	CPU% used executed at the user level with nice priority
IoStat	system	CPU% used while executing at the system (kernel) level
IoStat	iowait	time% of the CPU(s) in idle when system pending requests
IoStat	steal	time% spent on involuntary wait due to another virtual processor
IoStat	idle	time% of the CPU(s) in idle with no system pending requests

Figure 2 shows default information of the Linux *top* command output in kibibytes: total, used, free, and buffers measures are provided for the memory, while total, used, free, and cached are given for the swapped memory.

```
Mem:  1695028k total, 1677560k used,   17468k free, 131036k buffers
Swap:   499704k total,    18164k used, 481540k free, 218792k cached
```

Figure 2: Statistics collected with the Linux *top* command.

Table 3 shows an example of the *memory.used* monitoring metric turned into the csv format. The variables are: *time* expressing when the metric has been registered in the local time zone; *tags* and *domain* providing information about the collected machine values; *name* and *metric* providing the metric name and category; *value* containing the metric value.

Table 3: The memory.used monitoring metric turned into the csv format.

name	*tags*	*time*	*domain*	*duration*	*metric*	*value*
memory.used	host=api-int.cnsa.cr.cnaf.infn.it	1,62643E+18	cnsa.cr.cnaf.infn.it	0.22133	metrics-memory	1,37128E+09
memory.used	host=api-int.cnsa.cr.cnaf.infn.it	1,62644E+18	cnsa.cr.cnaf.infn.it	0.23058	metrics-memory	1,39689E+16
memory.used	host=api-int.cnsa.cr.cnaf.infn.it	1,62644E+18	cnsa.cr.cnaf.infn.it	0.22633	metrics-memory	1,46820E+16
memory.used	host=api-int.cnsa.cr.cnaf.infn.it	1,62644E+18	cnsa.cr.cnaf.infn.it	0.21558	metrics-memory	1,49160E+16
memory.used	host=api-int.cnsa.cr.cnaf.infn.it	1,62645E+18	cnsa.cr.cnaf.infn.it	0.21825	metrics-memory	1,49583E+16

4. Methods

Considering data characteristics and our main goal to identify anomaly, we have used a joint PCA with DBSCAN clustering method and PCA with K-means clustering method. Furthermore, time-series analysis has been performed at the machine level with all the monitoring metrics and log files for the various services. Log data at the service level do not show causal temporal relationships in time series data because the machines show messages as a consequence of the logging process.

4.1 Principal Component Analysis

PCA is a well-known statistical technique. When dealing with a large set of correlated variables, PCA allows creating a set with a lower number of variables that together are able to describe most of the variability of the original set of variables, measured in terms of variance. We observe that PCA properly works when the subset of principal components is big enough to properly represent the variability of the initial set of variables.

The principal component model identifies p principal components, where p refers to a parameter to be defined beforehand. From the result, it builds a new dataset with fewer entries and different values with respect to the starting one. We observe that the matrix multiplication between the transformed data and the n principal components (i.e., n is the number of all the components) would return the original data. Thereby, with a subset of p principal components, PCA returns a matrix of the same dimension of the original one, but with different values from the starting ones. If the selected principal components represent enough variation of the data, this resulting matrix will be very similar to the matrix with the original values. The difference between the original values and the ones obtained after this reconstruction process with a limited number of components is given by the reconstruction error (RE). The observations for which this error takes larger values are those where the PCA model does not fit them properly. We have assumed that anomalies represent rare events, therefore the PCA model has been built on recurring observations, while those for which the reconstruction error is larger are the rarest observations, thus representing potential anomalies [15]. To actually perform dimensionality reduction, only a subset p of the n principal components is kept.

For this method, the *tolerance* parameter identifies the percentage of error observations above which it is possible to consider anomalous a given point.

4.2 Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a traditional density-based clustering algorithm, grouping together data points that are closer to each other than a certain threshold, usually referred to as *epsilon*. This parameter

is just a measure of the distance between points. As a result, DBSCAN creates clusters in areas where points are concentrated, detecting also points that are separated by areas.

This algorithm recognizes the density by counting the number of points in the neighborhood defined by a certain radius of fixed length and defines two data points as connected if they lie inside each other's neighborhood. A point is a core point, if the neighborhood of radius *epsilon* contains at least a minimum number of points indicated with *MinPts*. A point q is directly density-reachable from a core point p, if q is within the epsilon-neighborhood of p, where density-reachability is given by the transitive closure of direct density-reachability. Two points p and q are called density-connected if there is a third point o such that both p and q are density-reachable. A cluster is then a set of density-connected points that is maximal with respect to density-reachability. All the distances have been calculated by means of the Euclidean distance. The noise or outlier is defined as the set of points that are not assigned to any cluster.

Estimating the epsilon has been uneasy when every machine has its own optimal value.

4.3 K-means

The K-means clustering divides data into K distinct, non-overlapping shards. The only parameter is K, the number of clusters, that is specified beforehand. K-means assigns each observation to exactly one cluster.

To better explain how it works, we introduce a mathematical notation. Let $C_1, ..., C_K$ denote the sets containing the indices of the observations in each cluster. These sets satisfy the following two conditions: 1. $C_1 \bigcup C_2 \bigcup C_3$; 2. $C_k \bigcap C_{k'} = \emptyset \ \forall k \neq k'$. The first condition specifies that each observation belongs to at least one cluster, while the second condition requires that the clusters do not overlap, therefore no observation belongs to more than one cluster. The main characteristic of the K-means clustering is to group data minimizing the within-cluster variation.There are multiple ways of formalizing the concept of variation within clusters, but the most common approach is to adopt the Euclidean distance. There are several algorithms to solve such a problem, the one adopted in this work is the one from [16].

K-Means algorithm is very convenient as it allows selecting the number of clusters a priori. This is very useful in the case of anomaly detection, where one can assume the existence of two classes: non-anomalous observations and anomalous ones. We have first used K-means with $K = 2$, and, then, also used $K = 3$, because log messages are not different enough from each other to successfully identify two "poles": a potentially anomalous one and a potentially non-anomalous one. A lot of messages are relatively similar to each other and only a less strict method of separation could bring meaningful results.

Using the K-means clustering technique, data have been partitioned into K clusters, where K could be either 2 or 3, and then the cluster with the smaller number of points has been considered to contain anomalies, assuming that anomalies are supposed to be a minimum or at least a small, part of the data. For most of the services, this approach has been quite effective.

5. Methodology

Predictive maintenance is the main goal of this study that we have tackled by using log files and monitoring metrics from a set of machines. On one hand, log files keep a trace of the events and

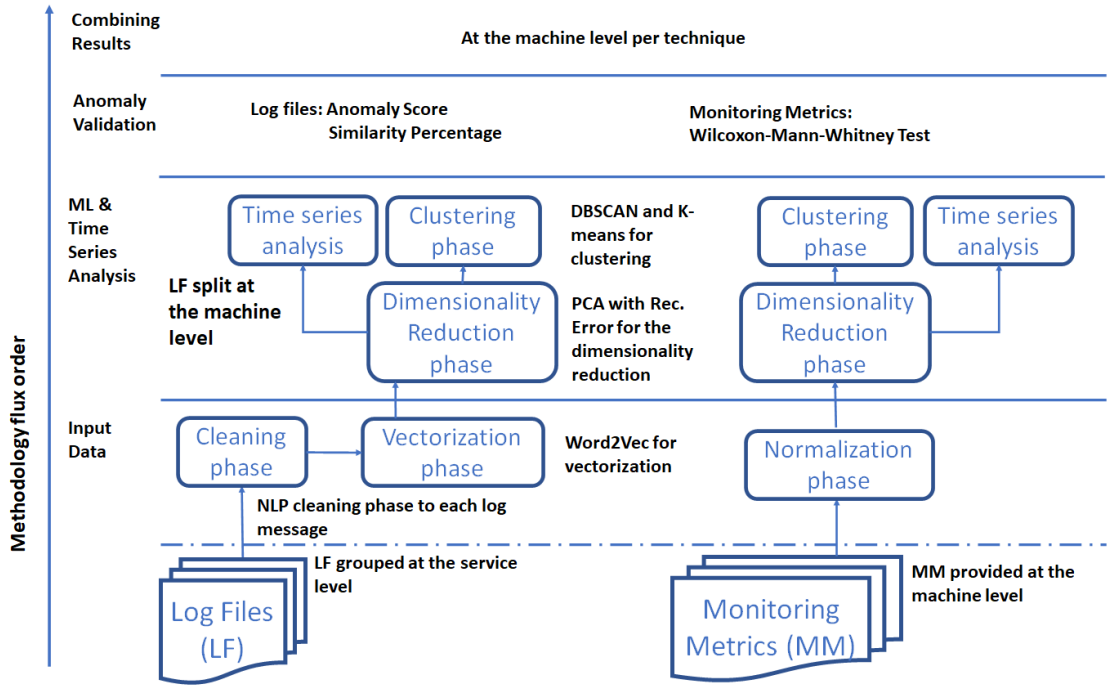

Figure 3: Bottom-up Anomaly Detection Approach with Log Files and Monitoring Metrics.

any likely problems in the corresponding services (e.g. with the usage of logging level information, such as info, error, and warning); on the other hand, the monitoring metrics can show a value variation in the relative trends (e.g. overstepping a threshold value of a specific metric, such as the total memory available on a machine) contributing to tag a likely anomaly in one or more machines. Furthermore, the log files envelop a data type that is not present in the monitoring metrics, identified by a sequence of alphanumeric terms. For data characteristics, we have planned to analyse them separately with unsupervised machine learning (ML) techniques, such as DBSCAN and K-means, and combine results at the machine level with respect to the used technique. Figure 3 shows the bottom-up reasoning adopted in our study whose blocks are explained in the following sections from the input data up to the anomaly validation part.

5.1 Input Data

From the bottom of Figure 3, it is possible to observe the input data, i.e. log files and monitoring metrics. Log data have been already pre-processed and given in structured csv format divided by services. However, various activities have been performed on the log files to turn their texts into features and convert them into numbers, leading to log files with the same data types as the monitoring metrics data. NLP procedures have been adopted in order to remove the unimportant words from each log message and reduce the identified features [17]. During the cleaning phase, each word in the log message has been put in lower case; digit numbers, links, punctuations, and special characters have been removed. The vectorization phase has been performed by applying the word2vec algorithm [18] that encodes words as similar low-dimensional vectors by using cosine similarity and exposes semantic information by linearly transforming these vectors: distances

between vectors are directly proportional to the semantic differences of the words corresponding to the vectors. The usage of NLP allows us to build a dictionary of single words that may identify anomalies in a service, such as *error*, *exception*, *failure*, *low*, *suspending*, *uncorrectable*, and *unrecognized*.

After a first exploration of the entire log files, considering the amount of data and information stored in those files, 11 of them have been identified as the most representative and meaningful for the analysis shown in this work: 1,258 machines have been considered, a total of 14,723,449 log messages have been analyzed with 3,068,479 unique messages, dealing with a total of 3,349 words. Table 4 summarizes a subsets of services considered for the input log files.

Table 4: Logging Services Description.

Service	**Description**	**Reference**
Grafana	It is multi-platform open-source analytics and interactive visualization application. It provides charts, graphs, alerts, and much more.	[19]
HAproxy	It is a free, open-source very fast service, offering high availability, load balancing, and proxy for TCP and HTTP-based applications.	[20]
Octavia	It provides the load balancing API for OpenStack. It is an open-source, operator-scale load-balancing solution designed to work with OpenStack.	[21]
StoRM	StoRM (STOrage Resource Manager) is a storage management service for the generic disk-based storage system.	[10]
Daemon	It is a program running constantly in the background and wakes up to handle periodic service requests, usually coming from remote processes.	[22]
Rsyslog	SysLog is the log messages' standard structure that separates the software generating the messages, the system storing them, and the software that reports them.	[23]
Auditd	It is a service used to log events on Linux systems.	[24]
Smartd	It is a daemon that monitors the Self-Monitoring, Analysis and Reporting Technology system hard drives.	[25]

As monitoring data are the result of an active controlling process, their frequency and therefore the availability of data is clearly different with respect to log data, which are logged without a fixed frequency but rather depend on the occurrence of a certain event produced by the corresponding service (e.g., info, warning, error). Therefore, we have dedicated effort to aggregate log files and monitoring metrics, taking relevant information from each data type at the proper point in time in order to increase the knowledge power and thus the accuracy of the anomaly detection process. For this reason, among all the monitoring data, only the machines for which also log data are available have been selected: 290 machines have been analyzed, for a total of 1,786,283 (1.78 million) data points.

Considering the amount of data and outcomes, to identify if there were statistics to omit in the results, all the monitoring metrics in the various categories have been correlated. Figure 4 shows that the various metrics are correlated both positively and negatively.

This paper shows the results for a subset of metrics, each one belonging to a metric category: *iostat.avg-cpu.pct_idle* that represents the average usage of CPU in the idle state of the machine; *load_avg.fifteen* that represents the average load of work on 15 minutes timespan; *memory-usage* that is the ratio between the memory used and the total memory space.

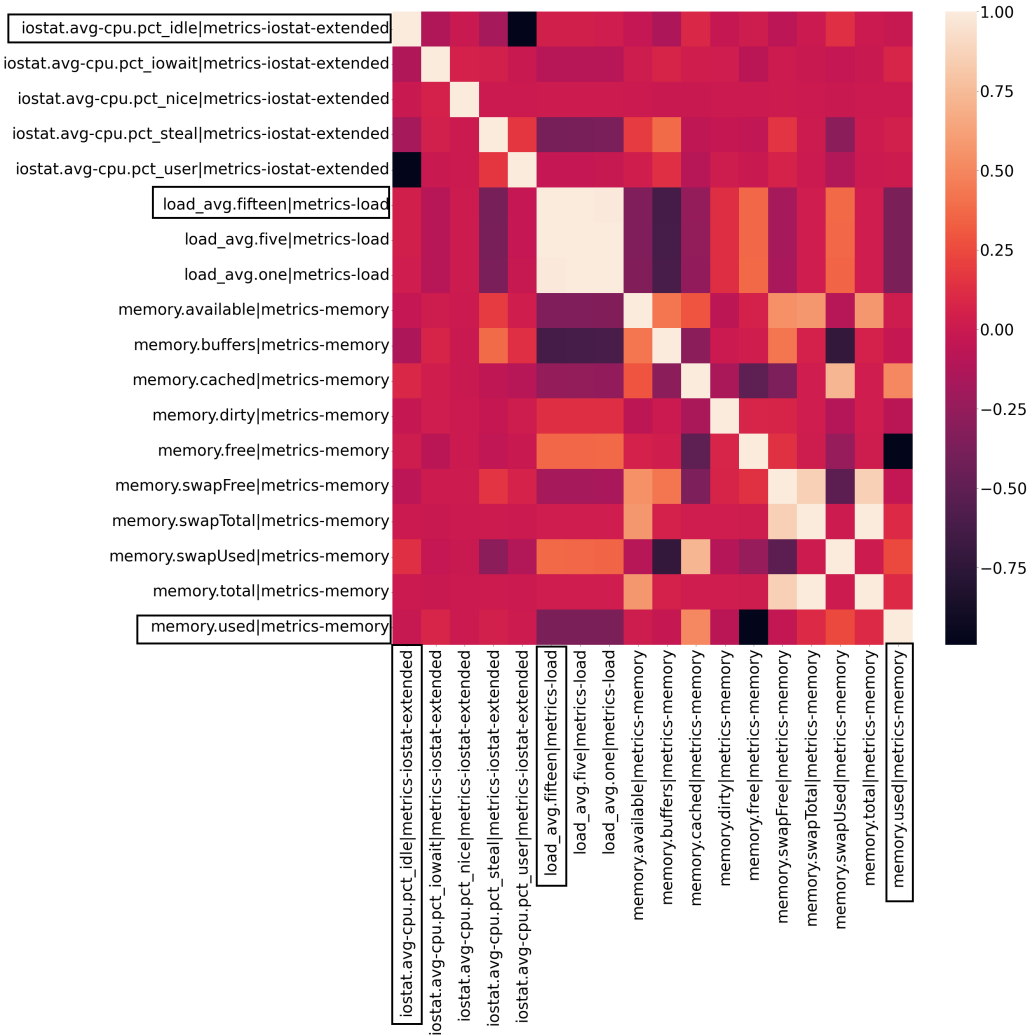

Figure 4: Heatmap of the correlation matrix from the *ce01t-htc* machine.

5.2 Machine Learning Techniques

In Figure 3, the ML techniques have been introduced to identify anomalies in the data. They use the resulting input data, organized at service and machine levels.

With respect to log files, PCA has been applied to the word-vectors obtained during the vectorization phase. Every word corresponds to a vector of 100 elements, therefore reducing the dimension of word-vectors to two dimensions has been useful to decrease the complexity of the problem and the computation time. Furthermore, with the application of the PCA with the reconstruction error (RE) process we have identified words that may be considered anomalies: the larger the RE is for uncommon terms, the more likely anomalies are. For example, the *failed* term has been labelled as an anomaly.

For the monitoring metrics, PCA has been applied to numerical data, identifying as anomalies those observations with a larger reconstruction error with respect to the starting sample. Figure 5 shows an example of anomalies identified for a specific machine with a tolerance of 5% on 3

components once applied PCA with reconstruction error.

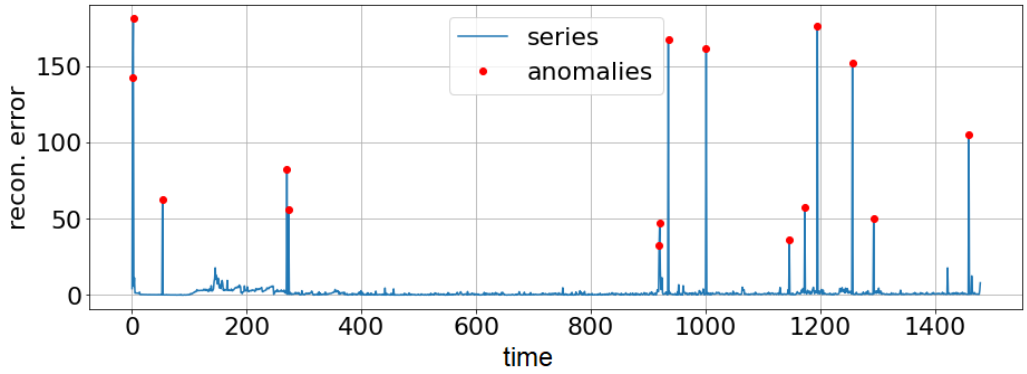

Figure 5: An example of anomalies on the *centos8* machine with a tolerance of 5% on 3 components.

The outputs obtained with the application of PCA have been used to feed DBSCAN and K-Means clustering techniques. DBSCAN is able to work with clusters of different sizes and shapes, so it works with log messages of different sizes. Furthermore, since DBSCAN is based on the density-reachable principle, it can avoid assigning every observation to a cluster, and identify outliers, that can be digested as anomalies. The technique has been used to label words as anomalies whenever possible. Once applied DBSCAN, non-clustered words are flagged with the value 1. The *failed* term has been still labelled as an anomaly. The percentage of epsilon has been referred to the quantile of the series obtained from the distance of all the observations (words) between each other.

In the case of K-Means, the anomaly detection method involved partitioning the observations among K clusters and then considering as potential anomalies those observations belonging to the least populated cluster. With respect to log files, this means the cluster with the smallest number of words. The identified anomaly words are then flagged with value 1. Once more, the *failed* term has been labelled as an anomaly. Furthermore, the *error* term has been also labelled as an anomaly.

5.3 Time Series Analysis

In Figure 3, the time series technique has been also introduced. They are applied to the resulting input data, organized at service and machine levels, with and without the PCA application.

Time series analysis has been performed on log data and monitoring metrics at the machine level. We have excluded analysing data at the service level for two reasons: on one hand every machine can host more services, and temporal causality between messages of the same service but different machines would be completely meaningless; on the other hand log data at the service level are not meaningful from a temporal causality point of view as the machines show messages as they pop up in the logging process.

With logs, messages have been vectorized and word-vectors from each message have been averaged to get the vector belonging to the message. The difference between consecutive vector-messages has been calculated (measured in terms of Euclidean distance between vectors), thus obtaining a time series of the differences between messages. Then, the mean μ and variance σ^2 (consequently, standard deviation σ is also retrieved) of the resulting time series are calculated and an interval of non-anomaly values is defined, where such interval includes all the observations

whose values are at most t standard deviations away from the value of μ. t is an integer value, which in the literature has usually taken values 1, 2 or 3 [26]. Since the treated time series represent distances, they only take positive values and the interval is only upper bounded. Thus, the threshold value is computed, adding μ to $t\sigma$.

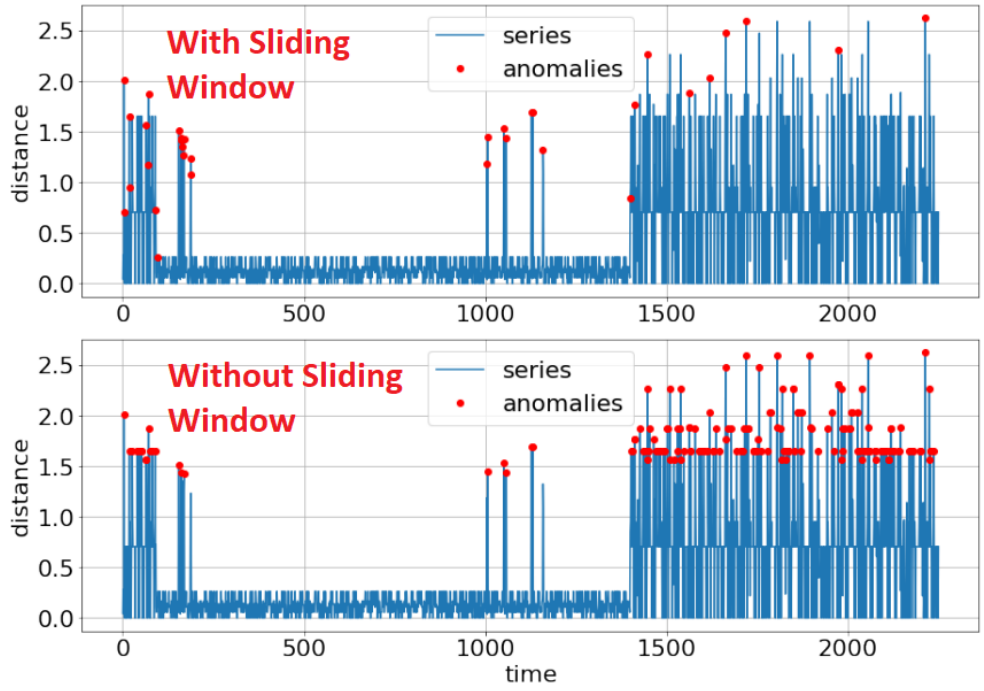

Figure 6: An example of time series analysis with and without the sliding window (a window size of 20 observations) and a tolerance of 2 standard deviations on the *cloud-ctrl01* machine.

The tolerance parameter defines the number of standard deviations from the mean from which a data point can be considered anomalous.

In this work, we have also taken into consideration the sliding window [26], [27]. This process involves taking overlapping subsets of the time series itself and then applying the same procedure described above to the said subset of data to detect anomalies. This process can bring significant improvements as it takes into account more local peaks, therefore being more precise, and at the same time it is more robust against temporary non-anomaly observations whose value is significantly different from other recurrent non-anomalous observations. Figure 6 shows the application of time series analysis to monitoring data with and without the sliding window. The trends are for the *cloud-ctrl01* machine, the cloud controller server one, where the auditd, octavia-housekeeping, rsyslogd, smartd log files are available.

With monitoring data, metrics are already given at the machine level. They do not need to be preprocessed as a function of time, but they have required to be standardized as the orders of magnitudes of the various metrics are really different from each other.

5.4 Validation Methods

For log files, we have considered the following validation methods: anomaly score, qualitative analysis, and similarity percentage.

The application of PCA to DBSCAN or the PCA to K-means produces, for every log service file, a set of words and a binary label, 0 for non-anomaly or 1 for anomaly words. With respect to each log message, an anomaly score is computed as the average value of the labels corresponding to the words that are part of the message, transforming the binary anomaly into a continuous measure. This metric allows attributing a score also to words that are considered both anomaly and non-anomaly: behavior that has been observed due to a limited number of terms present in the overall log messages. Figure 7 shows an example of how the anomaly score is obtained given a log message: the words in red are the words identified as anomalies.

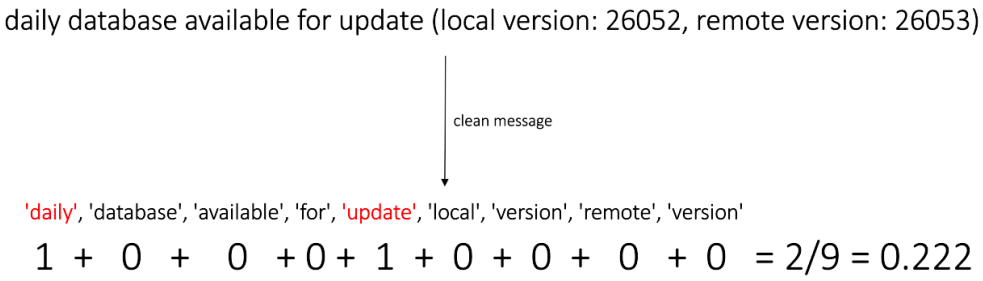

Figure 7: An example of anomaly score computation.

The qualitative analysis considers the most recurring words in messages with high anomaly score and low anomaly score for each technique and every set of parameters, such as the epsilon value. Common words between anomaly and non-anomaly messages have been compared to check if they contain any differences.

A similarity percentage of more than 20%/25% would have been seen as an alarming sign that the partition might not be effective enough. A similarity percentage of more than 20% would mean that the two sets share more than 20% of their words. This might seem a relatively high measure but it should also be considered that log messages share a lot of common words both for anomaly and non-anomaly messages, therefore a too strict measure would have been counterproductive.

For monitoring data, to compare the differences between two samples, the test of hypothesis has been used. Since this problem deals with two samples that can be considered independent, a non-parametric test is required. In this study, the Mann-Whitney test has been used as it has shown to be quite effective in anomaly detection tasks [28].

6. Results and Discussions

This study has produced a huge amount of results for the various techniques. Some results still require a deep investigation. In the following, the most relevant results have been illustrated and discussed.

The anomaly score has contributed to identifying anomaly and non-anomaly log messages, according to the following criteria: *anomaly score* > 0.7 implies an anomaly, while *anomaly score* < 0.3 implies a non-anomaly.

Similarity percentage values have been never above 23% for these three techniques, reinforcing the hypothesis of a good splitting. However, for time-series analysis, their scores per machine have

been most of the time higher than 80%, with low cases close to 50%. Time-series algorithms have been applied at the machine level (rather than service) and the logic of the anomaly detection mechanism is completely different with respect to the service level, therefore this process is not robust and requires further investigation. For example, the results are different for every machine and it is too dependent on the order of messages. Furthermore, results depend too much on the machine under analysis.

Anyway, time-series algorithms have shown interesting results for monitoring metrics.

For many machines, data were available either from log files or monitoring metrics. For some services, monitoring data were not been available. Results have been assembled together when data were provided in log files and monitoring metrics, to increase (or not) the certainty of results obtained.

Correspondence has been verified by checking logs and monitoring occurrences at less than 900 (15 minutes) seconds of difference because log and monitoring events could be registered at a different time. The assumption is that anomaly and non-anomaly events are, at least to some extent, consistent in time and are not single points in time. This can be safely assumed for non-anomaly events, while can lead, at least sometimes, to a small yet likely negligible imprecision for anomaly events. Figure 8 shows the occurrences of the log messages in red and the normalized metrics in blue for the *centos8* machine, where the rsyslogd and smartd services run.

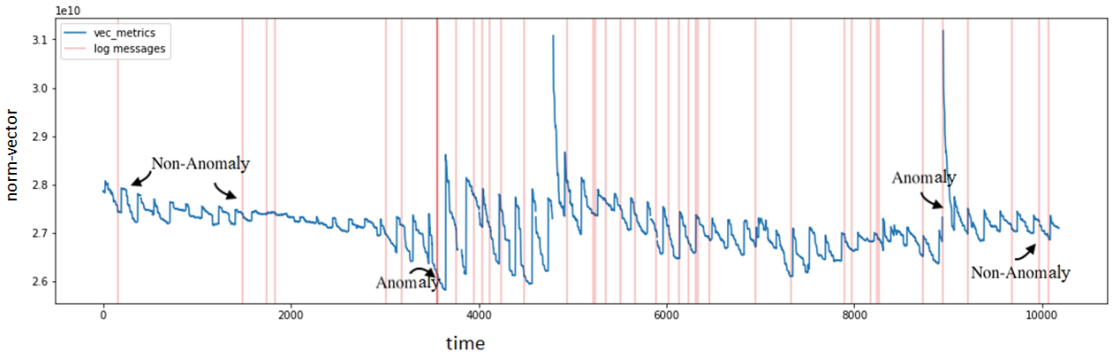

Figure 8: Log files in red versus monitoring metrics in blue for the *centos8* machine: anomaly and non-anomaly occurrences are highlighted.

Table 5 shows results for services that got a corresponding in the monitoring data.

7. Conclusions

In the present work, we have been able to validate an ad-hoc defined clustering algorithm to perform predictive maintenance at the INFN CNAF data center by identifying anomalies on the basis of heterogeneous unstructured data that provide us with qualitative and quantitative information about the state of machines and services. Log files of different services together with monitoring metrics have been considered at the machine level. As an added value, the adoption of a multivariate time series anomaly detection technique enabled us to compute anomaly scores on monitoring data to identify the time frame where we could overlap services and monitoring data anomalies to perform predictive maintenance analysis.

Table 5: A subset of results combining logs and monitoring Metrics

Service	**Log File**	**ML Technique**	**Parameter**	**% of correspondence**
Auditd	auditd	DBSCAN	epsilon = 0.1	50%
		K-Means	n_clusters = 3	97.5%
		PCA	threshold = 0.85%	20%
Smartd	smartd	DBSCAN	epsilon = 0.1	55%
		K-Means	n_clusters = 3	53%
		PCA	threshold = 0.85%	55%
Octavia	octavia-housekeeping	DBSCAN	epsilon = 0.1	~ 100%
		K-Means	n_clusters = 3	99.8%
		PCA	threshold = 0.85%	95%
HAproxy	haproxy	DBSCAN	epsilon = 0.1	51%
		K-Means	n_clusters = 3	~ 100%
		PCA	threshold = 0.85%	~ 100%

We found that for log data, DBSCAN, K-means, and PCA were very effective. For monitoring data, time series analysis provided interesting results, while treating messages as time-series observations did not show very meaningful results. We got over 50% of correspondence between anomalous found in log files and monitoring metrics.

The obtained results demonstrate that the defined pipeline can be exported to other data centers thanks also to the open-source nature of the code adopted for its implementation.

Obviously, the present study can be improved and refined (I) by analysing more in detail the nature of the anomalies to perform a more precise identification of the root causes of anomalies themselves, and (II) by developing an advanced deep learning model that uses different types of data with respect to timestamp, machine name, and network info. Moreover, it has to be taken into account that training and related inference may vary depending on the amount of data provided by the data center.

References

[1] Claudia Cavallaro and Elisabetta Ronchieri. Identifying anomaly detection patterns from log files: A dynamic approach. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blečić, David Taniar, Bernady O. Apduhan, Ana Maria A.C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre, editors, *Computational Science and Its Applications – ICCSA 2021*, pages 517–532, Cham, 2021. Springer International Publishing.

[2] Laura Viola, Elisabetta Ronchieri, and Claudia Cavallaro. Combining log files and monitoring data to detect anomaly patterns in a data center. *Computers*, 11(117), 2022.

[3] Carles Riera, Camilo Rey, Thiago Serra, Eloi Puertas, and Oriol Pujol. Training thinner and deeper neural networks: Jumpstart regularization, 2022.

[4] James Gareth, Daniela Witten, Travor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R. Second Edition*. Springer Texts in Statistics, 2021.

[5] Maria Grigorieva Ivan Zherdev, Konstantin Zhukov and Sergey Korobkov. Parallelizing of the dbscan algorithm in the clusterlogs framework. *International Journal of Modern Physics A*, 37(01), 2022.

[6] Carly L. Clayman, Satish M. Srinivasan, and Raghvinder S. Sangwan. K-means clustering and principal components analysis of microarray data of l1000 landmark genes. *Procedia Computer Science*, 168:97–104, 2020.

[7] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction. *Journal of Big Data*, 8(1), jan 2021.

[8] JeeHee Lee and June-Seong Yi. Predicting project's uncertainty risk in the bidding process by integrating unstructured text data and structured numerical data using text mining. *Applied Sciences*, 7(11):1141, nov 2017.

[9] Luca Giommi, Daniele Bonacorsi, Tommaso Diotalevi, Simone Rossi Tisbeni, Luca Rinaldi, Luca Morganti, Antonio Falabella, Elisabetta Ronchieri, Andrea Ceccanti, and Barbara Martelli. Towards predictive maintenance with machine learning at the infn-cnaf computing centre. In *International Symposium on Grids Clouds 2019, ISGC2019*, 2019.

[10] GitHub. Storm: a manager for storage resource in grid, 2022. http://italiangrid.github.io/storm/documentation/functional-description/1.11.2/.

[11] Diana El-Masri, Fabio Petrillo, Yann-Gaël Guéhéneuc, Abdelwahab Hamou-Lhadj, and Anas Bouziane. A systematic literature review on automated log abstraction techniques. *Information and Software Technology*, 122:106276, 2020. https://www.sciencedirect.com/science/article/pii/S0950584920300264.

[12] Ray Walker. Examining load average. https://www.linuxjournal.com/article/9001, Online; accessed April 16, 2023.

[13] Sandra Henry-Stocker. Making sense of memory usage on linux. https://www.networkworld.com/article/2722141/making-sense-of-memory-usage-on-linux.html, Online; accessed April 16, 2023.

[14] Ravi Savi. Linux performance monitoring with vmstat and iostat commands. https://www.tecmint.com/linux-performance-monitoring-with-vmstat-and-iostat-commands/, Online; accessed April 16, 2023.

[15] Bernhard Schölkopf, John Platt, and Thomas Hofmann. In-network pca and anomaly detection. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, pages 617–624, 2007.

[16] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 147–153. AAAI Press, 2003.

[17] Christophe Bertero, Matthieu Roy, Carla Sauvanaud, and Gilles Tredan. Experience report: Log mining using natural language processing and application to anomaly detection. In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pages 351–360, 2017.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[19] S. Venkatramulu, M.S.B. Phridviraj, C. Srinivas, and V. Chandra Shekhar Rao. Withdrawn: Implementation of grafana as open source visualization and query processing platform for data scientists and researchers. *Materials Today: Proceedings*, 2021. https://www.sciencedirect.com/science/article/pii/S2214785321024160.

[20] Connor Rawls and Mohsen Amini Salehi. Load balancer tuning: Comparative analysis of haproxy load balancing methods, 2022.

[21] OpenStack. Introducing octavia. https://docs.openstack.org/octavia/latest/reference/introduction.html, Online; accessed April 16, 2023.

[22] Daemon. Linux manual page. https://man7.org/linux/man-pages/man7/daemon.7.html, Online; accessed April 16, 2023.

[23] Rsyslogd. Documentation. https://www.rsyslog.com/doc/master/index.html, Online; accessed April 16, 2023.

[24] Auditd. Insightidr - auditd compatibility mode for linux assets. https://docs.rapid7.com/insight-agent/auditd-compatibility-mode-for-linux-assets/, Online; accessed April 16, 2023.

[25] Smartd. Configuration. https://man.freebsd.org/cgi/man.cgi?smartd.conf%285%29, Online; accessed April 16, 2023.

[26] Leticia Decker, Daniel Leite, Luca Giommi, and Daniele Bonacorsi. Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2020.

[27] Vannel Zeufack, Donghyun Kim, Daehee Seo, and Ahyoung Lee. An unsupervised anomaly detection framework for detecting anomalies in real time through network system's log files analysis. *High-Confidence Computing*, 1(2):100030, dec 2021.

[28] Yuhang Wu, Baojing Huang, Xue Li, Yingnan Zhang, and Xinyue Xu. A data-driven approach to detect passenger flow anomaly under station closure. *IEEE Access*, 8:149602–149615, 2020.