Real time anomaly detection of NoSQL systems based on resource usage monitoring

Spyridon Chouliaras and Stelios Sotiriadis

***Abstract*—Today, with the emergence of the industry revolution systems such as Industry 4.0, Internet of Things and big data frameworks pose new challenges in terms of storage and processing of real time data. As systems scale in humongous sizes, a crucial task is to administer the variety of different sub-systems and applications to ensure high performance. This is directly related with the identification and elimination of system failures and errors, while the system runs. In particular, database systems, may experience abnormalities related with decreased throughput or increased resource usage, that in turn affects system performance. In this work, we focus on NoSQL database systems, that are ideal for storing sensor data in the concept of Industry 4.0. This typically includes a variety of applications and workloads that are difficult to online monitor, thus making anomaly detection a challenging task. Creating a robust platform to serve such infrastructures with minimum hardware or software failures is a key challenge. In this work, we propose RADAR, an anomaly detection system that works on real-time. RADR is a data-driven decision-making system for NoSQL systems by providing process information extraction during resource monitoring and by associating resource usage with the top processes, to identify anomalous cases. In this work, we focus on anomalies such as hardware failures or software bugs that could lead to abnormal application runs, without necessarily stopping system functionality e.g. due to a system crash, but by affecting its performance e.g. decreased database system throughput. Although, different patterns may occur through time, we focus on periodic running workloads (e.g. monitoring daily usage) that are very common for NoSQL systems, and Internet of Things scenarios where data streams are forwarded to the Cloud for storage and processing. We apply various machine learning algorithms such as autoregressive integrated moving average (ARIMA), seasonal ARIMA and long-short-term memory recurrent neural networks. We experimentally analyse our solution to demonstrate the benefits of supporting online erroneous state identification and characterisation for modern applications.**

***Index Terms*—Anomaly detection, Real time analytics, Cloud Computing, NoSQL systems.**

I. Introduction

TODAY, data streams from Internet of Things and Industry 4.0 modern factories are becoming massive in terms of volume, variety and velocity that are generated and stored. A common practice for storing semi-structured dataset is the use of NoSQL systems that provide flexible storage ways, enough to deal with massive amounts of data [1], while at the same time are easy to scale up and manage, comparing to monolithic database systems. A key challenge is to support robustness and reliability of cloud systems with respect to hardware and software failures, especially when the system scales up in massive cluster sizes. Therefore, efficient ways on detecting abnormalities are more important than ever as anomalies can affect performance and provoke unreasonable energy consumption, that in turn results to false scaling estimation. In addition, abnormal patterns are a strong indicator of cyber-attacks from unauthorised users [2].

In this work, we develop RADAR, a system for real time anomaly detection in NoSQL databases based on monitoring resource usage on real time. To achieve it, we analyze historical data for resource usage metrics (CPU, memory, disk usage etc.) and we train different machine learning models to generate patterns, such as (a) Autoregressive Integrated Moving Average (ARIMA), (b) Seasonal ARIMA and (c) Long-Short-Term Memory Recurrent Neural Networks. RADAR uses a pattern recognition model further allow us to support a signal similarity process to identify future system abnormalities on the fly. We rank the various models, to ensure we select the most accurate model for various cases. The predicted signal (generated from the previous mentioned models) is used as a representative wavelet to detect future abnormalities. To move a step forward, we compare new signals using the Dynamic Time Warping (DTW) method as in [3] and in [4]. The algorithm maps the representative and the new wavelet (that is real time generated) in order to classify it as normal or abnormal based on a threshold.

We developed an experimental system based of NoSQL databases to generate real time workloads using the MongoDB system. To create a real-world case scenario, we use the Yahoo! Cloud Serving Benchmark (YCSB) [5]. At the same time, NoSQL applications are monitored by collecting different information such as CPU percentage, Virtual Memory percentage, Disk usage and information for each process separately. The process information associates abnormal runs with the cause of event and provides an added feature that is key to increase models accuracy.

The contribution of this work is on the *comparison of different machine learning models to design patterns for NoSQL system runs and characterise them according to resource usage monitoring and process information extraction*. We experimentally demonstrate that the LSTM-RNN model (that is a sequence learner) provides the best performance by offering better accuracy in terms of the root mean square error. Our experimental analysis is based on a combination of read and write operations of YCSB workloads to demonstrate

S. Chouliaras is with the Department of Computer Science and Information Systems, Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, UK e-mail: s.chouliaras@dcs.bbk.ac.uk.

S. Sotiriadis is with the Department of Computer Science and Information Systems, Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, UK e-mail: stelios@dcs.bbk.ac.uk.

Manuscript received July 31, 2019; revised October 25, 2019.

realistic runs.

II. Motivation

Industry 4.0, Internet of Things and data streaming systems are usually related with semi-structure datasets and require fast and efficient data storage such as NoSQL systems. Modeling periodic workload executions of NoSQL systems is a challenging task, mainly because of the variety of workloads and varied user requests. Our method based on the the assumption that workload types of applications are known and NoSQL systems can generate resource usage patterns that are recurrent due time, thus to support this we present a preliminary experimental analysis. The experiment includes the execution of the YCSB workload in a MondoDB VM to explore its behaviour. Figure 1 illustrates the use of a YCSB workload and the variations created on the VM's CPU usage percentage over 70 minutes period of time. We used a YCSB workload that includes 200 thousand records with 50% read and 50% write operations in order to create a single wavelet. Thus, generating 14-th wavelets over the 70 minutes period needed about 2.8 million of records to be loaded and executed in the NoSQL application. As shown in Figure 1, we observe that patterns (also called predicted wavelets) are repeatable and modelling such a signal could support future predictions on unseen wavelets for anomalous runs. In our case, the predicted wavelet has been used as a representative wavelet of all the training signals. A model that can map representative wavelet to a new unseen wavelet can classify it as normal or anomalous, according to the "difference" (in terms of signal similarity) among wavelets. This demonstrates the importance of creating a contiguous run of normal patterns in our system.

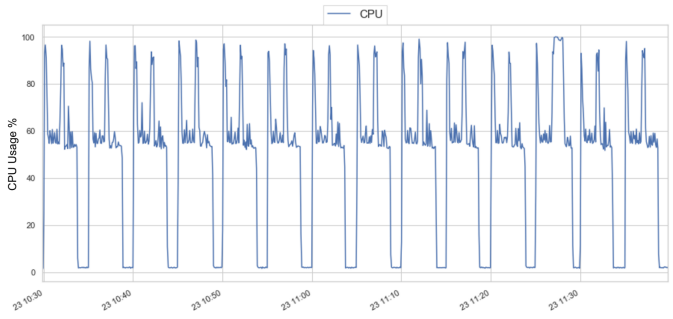

Fig. 1. CPU Usage percentage over YCSB workload

Figure 2 shows both normal and abnormal signals over 90 minutes period of time. After the 14-th wavelet, four new abnormal wavelets have occurred in the system and caused a sharp increase to CPU usage that surpasses 95% percentage point for a significant period. To demonstrate this, between the $11:40$ and $12:00$ time instances, we ran the YCSB workload simultaneously with a stress workload (using CPU-bound tasks) in order to further overload the system and create synthetic abnormal signals. We can easily observe that the last 4 signals are significantly different than the previous, however there is no clear definition whether the signals are abnormal or not. Especially, from the perspective of the application, a common abnormality could be an affection on the throughput, however this classification cannot happen with the current experiment.

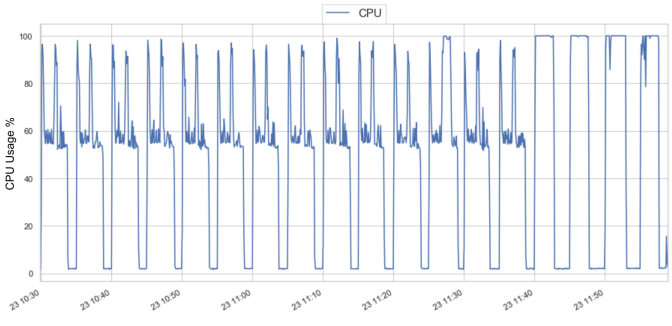

Fig. 2. CPU Usage percentage over YCSB workload and stress package

To further identify abnormalities, we ran an experiment by monitoring the CPU percentage of a VM while running the YCSB workload. As shown in Figure 3, a NoSQL application has been deployed on Cloud and measured based on the resource usage, process information and data throughput. At a point (time instance between 11:54 - 11:56) an abnormal run has been introduced to the system, by stressing the system with a highly CPU intensive daemon process. We executed a YCSB workload with 200 thousand records by running in a small size VM (2 CPU Core,40 HD Disk, 4GB RAM). The YCSB configuration includes a 50% reads and 50% writes to create CPU usage variation in different time periods. The CPU usage pattern can be identified in the first part of the wavelet. To further demonstrate the process information extraction, we label the process with the maximum CPU percentage that specific moment. The latter points out the process that contributed the most in that period of time and can be a further criterion to understand what caused abnormal behaviours.

While the system is under stress, the wavelet pattern changes and the $stress$ process (as shown in the graph) generates the maximum CPU usage. In particular, 11:41 - 11:54 demonstrates normal runs while 11:54 - 11:56 shows an abnormal run that caused a substantial increase to CPU usage. The labels can show that $stress$ process generates this abnormality. We expect that this analysis will provide an important feature for the machine learning model.

Fig. 3. CPU percentage over YCSB labeled by maximum process CPU

To explore further how abnormal run affects the application performance, we measure the throughput of a MongoDB

Fig. 4. YCSB Workload Average read operations

storage system. Figure 4 shows the average operations in MongoDB, at the same time period as the data shown in Figure 3. We can observe that the stress package does not create abnormalities only in the CPU usage but also in the average throughput (named as AVG). The average read operations in the database decreased sharply between 11:54 and 11:56, as in that time both YCSB and stress were running simultaneously (e.g. a drop on the throughput from an average of 450 to an average of 300). We summarize our motivation findings as follows.

- NoSQL workloads can generate repeatable patterns, that over time can be similar, thus pattern recognition mechanism can support advanced anomaly detection analytics.
- Process information extraction can further support the analysis process. By identifying possible abnormal runs, process name is an important feature.
- Anomaly detection requires constant understanding of application metrics, e.g., throughput and process data to define abnormalities with higher accuracy.

Based on this discussion, we present RADAR, a system to benefit abnormal system identification and characterization by automatizing the resource usage and process information extraction on the application side.

III. RADAR: Reactive Anomaly Detection Analytics on Real Time Resource usage monitoring

RADAR provides is designed as a reactive component, that can collect data, analyze, on the fly, and generate patterns accordingly. Figure 5 demonstrates the flowchart of RADAR deployment that includes the following components.

- The Monitoring Engine: This component includes the application and the workload execution. Data are collected and submitted to the storage system.
- The Storage Node: This component includes the NoSQL system (MongoDB) for data storage. Data can be populated by other components.
- The RADAR analyser: This component includes the models for analysis, including RNN and ARIMA. In addition, patterns generated by these models are forwarded to the signal similarity method to identify anomalous runs.

The flow of events is as follows.

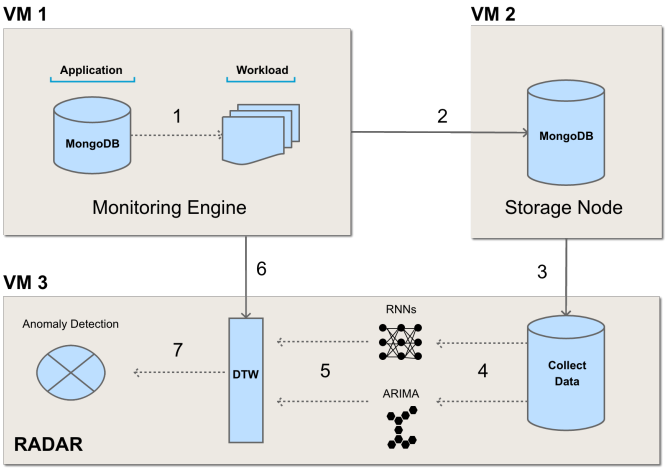

Fig. 5. RADAR: System information flow chart

- Line 1, it links the ”Application” (NoSQL systems of MongoDB) monitored by the engine with the YCSB workload. The monitoring engine collects different resource usage data such as CPU usage, Memory usage, Disk utilisation as well as process information such as CPU usage per process, Memory usage per process and other. Then, the application is benchmarked using the YCSB workloads and the monitoring engine sends status information every interval (in our case every 5 seconds) to the storage node.
- Line 2, it shows the connection of the application with the storage node that collects the historical data (JSON format). The storage node continuously receives data from the monitoring engine while RADAR retrieves data every specific time interval to be analysed and processed by the user.
- Line 3, it illustrates the connection of MongoDB node with RADAR system. At this event, the system is ready to generate analytics as shown in Figure 1 and illustrate abnormalities in the application as shown in Figure 2.
- Line 4, it illustrates how the collective time series data used as an input to different models, such as ARIMA and LSTM-RNNs models and output predictions about future abnormalities that may occur. Both models are evaluated based on the root mean square error (RMSE).
- Line 5, it illustrates how the predictive wavelet, generated by the models (LSTM-RNNs and ARIMA models), has been used as the representative wavelet from the dynamic time warping algorithm.
- Line 6, it demonstrates the way in which RADAR is detecting abnormalities on-the-fly. After the LSTM-RNNs and ARIMA models have been trained and generated a representative wavelet, a new test wavelet enters in RADAR in real time follows a mapping process with the representative wavelet.
- Line 7 it shows a signal similarity algorithm, such as the dynamic time warping (DTW) calculates the distance between the representative and the test wavelet and clas-

sifies the latter to be normal or anomalous based on a specific threshold has been set on by the administrator

IV. Experimental Setup

This section describes the Cloud deployment and the datasets used for the experimental analysis. We deployed a MongoDB system for benchmarking and evaluating the performance of different models.

The experimental system includes three virtual machines, each of 6GB RAM, 40HD Disk, 2 CPU Core. The first virtual machine contains the MongoDB storage system that initialises the application to be monitored. In order to expose application on real time workloads, YCSB workloads is executed inside application, as YCSB offer specific workloads in order to test and evaluate various databases including MongoDB. For this work, a YCSB workload with 200 thousand records executed with 50% read and 50% write operations respectively. The aforementioned technique caused a realistic work flow on the system leading to CPU variations, as in Figure 1.

The dataset stored in MongoDB can be grouped into two categories:

1) Monitoring data for resource usage such as CPU, memory and disk usage.
2) Valuable information about each process that is running inside the application. This will allow information extraction for different scenarios, such as the process with the maximum CPU utilisation, the working directory of each process as well as the virtual memory used percentage.

RADAR extracts data from MongoDB storage node into the system for analysis and visualisation purposes.

V. Experimental scenarios

A. Autoregressive Integrated Moving Average Modelling

Autoregressive Integrated Moving Average (ARIMA) is an advanced method in order to model time series sequences and predict unseen future events. There are three parameters in ARIMA(p,d,q) model, where p is the past values that need to be used in order to predict future events or the total autoregressive terms, q is the past forecast errors or the total moving average terms in order to predict the future values and d is the order of differencing that needs to be applied to the sequence to guarantee stationarity. Parameter tuning of the ARIMA model has been followed in order to find the best model that produce the least the lowest Root Mean Square Error (RMSE).

The Autocorrelation Function plots (ACF) and Partial Autocorrelation Function plots (PACF) are necessary graphs in order to identify the parameters of the ARIMA model in stationary series [6]. An ACF plots or Correlogram has been visualised using all lag values with 5% significance limits for the autocorrelations. Figure 6 shows that lags result in successive negative and positive auto correlation values. The latter indicates that time series sequence does not need differencing as it is sufficiently stationary for modelling. ACF also demonstrates a periodicity over time series sequence. A seasonal fluctuation over the period of 58 lags has been observed where autocorrelations have larger values for the seasonal lags. Thus, the seasonal component may enhance model's performance and accuracy.

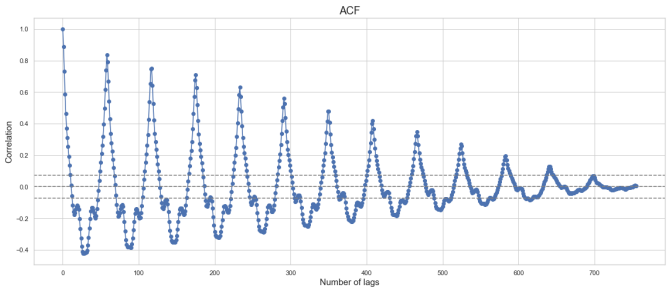

Fig. 6. Autocorrelation Function Plot

A PACF plot has been used with all lag values and 5% significance limits for the partial autocorrelations. The PACF plot shows the amount of correlation between the variables that is not explained by their mutual correlations. As for example, if there is a set of variables Y_1 Y_2 and Y_3 and a variable X is being regressing over those variables, the partial correlation between X and Y_1 is the amount of correlation between X and Y_1 that is not explained by their common correlations Y_2 and Y_3. As the PACF plot shows the partial autocorrelations of all the lags it can point out how many Autoregressive terms needed to be explained in the Autoregressive AR(p) model and in ARIMA(p,d,q) model in extend. In this research, different value of orders have been considered in ARIMA(p,d,q) model in order to minimise $RMSE$. Figure 7 demonstrates correlation of the residuals with 5% significance limits for the partial autocorrelations.

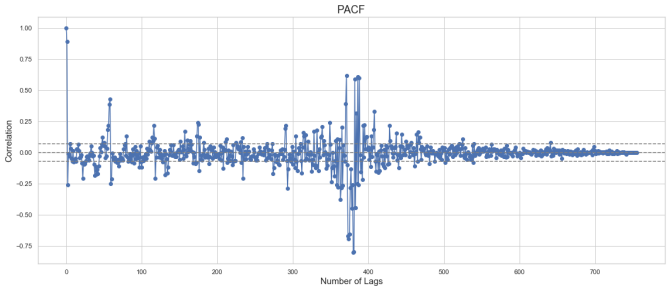

Fig. 7. Partial Autocorrelation Function Plot

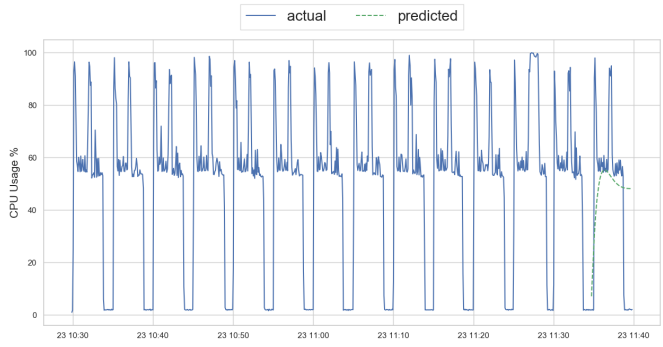

Fig. 8. Autoregressive Integrated Moving Average (ARIMA)

In addition, the Augmented Dickey Fuller (ADF) is unit root test that has been used to examine the null hypothesis of an ARIMA (p,1,0) process being non stationary against the stationary ARIMA (p+1,0,0) alternative [7]. The Null hypothesis has been rejected and ADF indicates a stationarity in time series. The latter has increased the efficiency of the ARIMA hyper-tuning process while the differencing level has been set to 0 as the ADF test indicates stationarity in time series. After a combination of all parameters, a final seasonal ARIMA $(3, 0, 1)$ model has been selected in order to be trained over the collective data. The ARIMA $(3, 0, 1)$ model has been evaluated over the standard deviation of the residuals with an $RMSE = 28.26$. The model has been trained over 13 wavelets as the 14-th wavelet has been used as the validation wavelet to evaluate model's predictive accuracy.

Figure 8 shows the predicted line (green dot line) that has been generated from ARIMA(3,0,1) model over the actual wavelet (blue line). As the graph illustrates, ARIMA model prediction tries to explain as match variation over the new unseen wavelet and follows the increased trend of the pattern. However, the ARIMA could not demonstrate a good fit, which is maybe a result of the time series sequence size.

B. Seasonal Autoregressive Integrated Moving Average Modelling

$ARIMA(p, d, q)$ model has been used in order to predict an unseen wavelet of CPU usage and produced an $RMSE = 28.26$. The ACF graph revealed a seasonal fluctuation over 58 lags. Thus, a seasonal component has been added into ARIMA (p,d,q) in order to decrease the RMSE and increase the accuracy of the model.

After careful considerations and combination of all the parameters, a final $SARIMA(1, 0, 3)(1, 1, 1)_{58}$ model has been been trained and produced a significant improvement over the standard deviation of the residuals with an $RMSE = 9.95$. The model has been trained over the 13-th wavelets and tested over the last 14-th wavelet. It is worth mentioning the SARIMA modelling was time efficient as the training phase was sort.

Figure 9 shows the predicted line (green dot line) over the actual wavelet between time period . The graph also illustrates the SARIMA model improvement over the non seasonal ARIMA model as the green dot line approaches the actual wavelet more precisely. The seasonal component and the use of Seasonal ARIMA has decreased significantly the root mean squared error and predicted accurately the test wavelet as it shown in Figure 9. However, this research is about to explore further models for time series data in order to achieve even better results in CPU usage prediction.

C. Long Short-Term Memory Recurrent Neural Networks Modelling

LSTM networks have been used in order predict the CPU usage of an unseen wavelet and create a representative wavelet to be tested with new unseen patterns. Firstly, time series have been normalised within the range of -1 and 1 in order to fit LSTM-RNN model while the activation function that has been

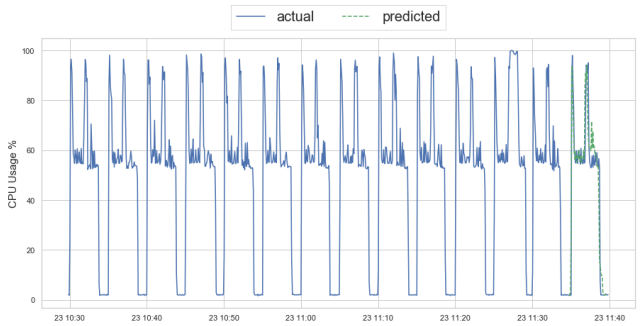

Fig. 9. Seasonal Autoregressive Integrated Moving Average (SARIMA)

used is the hyperbolic tangent $tanh$ function. The LSTM-RNN model has used as a training set 13-th wavelets and tested in a new unseen wavelet and evaluated over the Root Mean Square Error. The model has used Mean Squared Error (MSE) as the loss function through the training phase. Mean Squared Error of an estimator is a commonly used regression loss function that measures the sum of squared distances between the targeted variable and the predicted variable is being calculated.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{1}$$

where n is the number of the total predictions generated by the model, Y_i is the target variable and $\hat{Y}_i$ is the predicted variable.

The LSTM-RNN model has used the adaptive moment estimation (Adam) as an optimisation algorithm in order to iteratively optimise the objective function. Adam is an algorithm for first-order gradient based optimisation of stochastic objective functions based on lower-order estimates that are adaptive through the training process [8]. Thus, Adam optimiser promises less memory requirements as it computes only first and second order of gradients with individual adaptive learning rates. Table 1 shows different experiments with LSTM-RNN model where L indicates the number of layers, N indicates the number of nodes, RMSE the test Root Mean Square Error and N the total number of parameters that have been used through training phase. We can observe that as the number of epochs increases the RMSE is decreasing. We can observe that as the number of epochs increases, the error reduces accordingly.

TABLE I
EXPERIMENTING WITH NUMBER OF NODES AND EPOCHS OF THE LSTM-RNN MODEL

Layers	Nodes	Epochs	N	RMSE
1L	10	20	491	22.2
1L	50	20	10,451	17.86
1L	50	100	10,451	13.43
2L	50	20	30,651	14.12
2L	50	100	30,651	12.02
2L	50	200	30,651	6.87
2L	50	400	30,651	4.96

The best $RMSE = 4.96$ has been achieved after using a two-layer architecture with 50 nodes and training epochs set to

400. The total number of the parameters that have been used through the training phase is 30,651. It is worth mentioning, that training time has been sharply increased proportionally with the number of Nodes and Epochs. For that reason, a batch size of 50 training samples has been used in order to increase the efficiency of the training phase. The batch size is the number of samples that are processed through the network before the model is updated. Figure 10 illustrates the effectiveness of the LSTM-RNN model to predict a new unseen wavelet.The 14-th wavelet that has been produced as shown in Figure 10 (green dot line), approaches the actual wavelet (blue line) on the desired level.

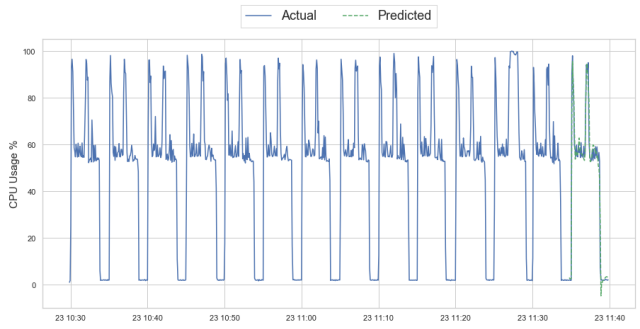

Fig. 10. Long Short-Term Memory Recurrent Neural Networks prediction

LSTM-RNN, ARIMA and SARIMA models trained based on resource usage data and generate predictions of future abnormalities in the system. As the LSTM-RNN model generates the least RMSE over the ARIMA and SARIMA models, it has been used as the primary model for this work. The predicted wavelet that has been generated by the LSTM-RNN model has been used as the representative wavelet over all the tested workloads. This representative wavelet has been used in the mapping process that tests for anomalies between the new unseen wavelets.

D. Signal similarity with Dynamic Time Warping

Since the machine learning models have been trained and can make future predictions, it is important to classify the new unseen wavelets as normal or abnormal. It has to be underlined, that LSTM networks have been trained on YCSB workload and learnt the repeated normal wavelet pattern over the time as it is shown in Figure 1. Thus, LSTM created a predicted wavelet that has been used as the representative wavelet to be test with new unseen wavelets. As the new pattern reaches the collection data inside RADAR from the data storage node, a signal similarity model calculates the difference between the test wavelet and the predicted wavelet. A mapping process between the predicted wavelet P_i and the test wavelet T_i is necessary in order to detect abnormalities to the system. For that purpose, RADAR uses a widely recognised signal similarity technique called Dynamic Time Warping to find the relation between the test and the predicted wavelet.

Figure 11 shows the predicted wavelet (green dot line) produced by the LSTM model; a model which has been trained over 13 normal wavelets. Beyond that, a new wavelet in the system has been classified by DTW algorithm as normal (blue line) since the distance between the predicted wavelet is lower than the $normal\_threshold$ with the . On the contrary, another wavelet has been detected as abnormal (red line) as the distance between the predicted wavelet is higher than the $normal\_threshold$. The normal threshold is calculated to 280, while the normal distance is 170 and the anomalous distance is 1554, that clearly shows that the latest (anomalous signal distance) is over the normal threshold, thus is classified as abnormal.

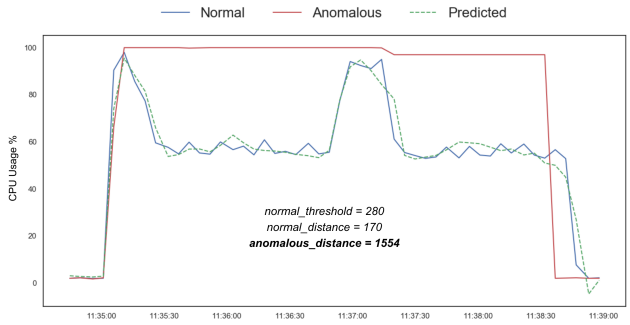

Fig. 11. Dynamic Time Warping classifies a wavelet as normal (blue line) or abnormal (red line) based on a $normal\_threshold$.

It has to be mentioned, that RADAR enables the user to customise the $normalthreshold$ based on system's functionality. In this research, the $normal\_threshold$ equals to 280 thus wavelets above that limit has been classified as anomalous.

E. Process information on Anomaly Detection

Detecting abnormalities in NoSQL systems enables system's administrators to audit the system more efficient. However, identifying the problem that has caused the abnormal behaviour it is still challenging. Application metrics (e.g. Operations per/sec) and system metrics (e.g.CPU usage, Memory usage) gives the auditor information about the error but in higher level. RADAR focuses on detecting abnormalities based on system metrics and at the same giving information to system's administrators about potential processes that have caused the anomalous run. While the system is being monitored on-the-fly by gathering important information about the resource usage, it is also collecting information about all the running processes in that time interval. Then processes have been sorted by CPU usage, so that the first process in the array is the process that needs the most CPU resources in order to run, the second process needs the second most CPU resources in order to run and so on. Consequently, data stored in the database as a a historical record.

After the collection stage, RADAR request data through specific time intervals for further analysis and visualisation. Figure 12 shows the CPU usage over 90 minutes and visualise also process information in that period of time. The graph gives the ability to the auditor not only observe CPU fluctuations over the time but also inspect which process needed the maximum CPU resources that specific moment. The latter explores the causality of abnormal CPU usage behaviour, thus

while the CPU usage exceeds the CPU threshold, RADAR points out the process that contributed the most. In this example, RADAR used a $CPU\_threshold$ of 85%, that means that process information has been visualised only if CPU usage exceeds that limit, however, auditors have the ability to customised the $CPU\_threshold$ accordingly.

Fig. 12. CPU percentage over YCSB labeled by maximum process CPU

Figure 12 shows labels of different process that caused the CPU to surpasses the customised CPU threshold. YCSB loads the workload using Java programming language thus java is on the major processes that has been pointed out from RADAR. MongoDB, that has been used as the primary database application in this research, has also drained out a significant amount of CPU while it performs reads and write operations. The graph demonstrates both, normal and abnormal runs over a 90 minutes period. Between $10:28-11:38$, YCSB workload was running in order to create normal wavelets that followed a realistic scenario with normal CPU usage fluctuations. However, between $11:38-11:58$, YCSB was running in parallel with Stress package that created abnormal wavelets and changed the pattern of a normal run. Stress package can stress the system based on user's pre-selected options while in this case Stress raised CPU usage above the $CPU\_threshold$ for a significant period of time. As the CPU usage has surpasses the $CPU\_threshold$, all processes (Java, Mongod and Stress) have been continuously visualised from RADAR.

VI. Literature review

In this section, we discuss approaches and techniques related with the anomaly detection in Cloud computing systems by focusing on supervised and unsupervised models by monitoring system information.

Large scale distributed systems generate massive amounts of data; thus anomaly detection is a challenging task, in terms of computational and storage efficiency. In order to tackle such issues, in [9] authors proposed a Principal Component Analysis (PCA) based method that identifies deviated data instances and in parallel reduces the dimensionality of the generated data. PCA is an advanced dimensionality reduction technique that maps the data in a lower dimension by keeping the maximum of the explained variance-covariance of a set of possibly correlated variables [10]. As their monitoring system collects time series stream data (such as TCP connection requests per second from local monitor nodes), a central coordinator node targets to detect abnormalities in the system. Their solution tries to tackle the trade-off between the accuracy of anomaly detection model and the amount of data that need to be analysed in order to improve the efficiency of the model as well as enable accurate detection.

In [11] authors have proposed a network anomaly detection technique to prevent the exhaustion of network's bandwidth over various denial-of-service attacks (DoS attacks). Firstly, they point out the importance of analysing bandwidth-utilisation in both types of intrusion detection systems (IDS) such as signature-based detection IDS or anomaly-based detection IDS. They used collective data from a known commercial web server in order to train and evaluate their model. The training data from the collective dataset have been used to train both Autoregressive Integrated Moving Average (ARIMA) and Seasonal Autoregressive Integrated Moving Average (SARIMA) models and the testing data have been used for model evaluation based on the normalised root mean square error (NRMSE) a normalised version of the root mean square error (RMSE). A threshold parameter has been obtained over the upper bound of the NRMSE of the testing data. The anomaly detection technique is based on SARIMA model prediction over the actual usage. If the NRMSE produced by the SARIMA model is greater than the threshold, an abnormality has occurred in the system. Their work has successfully detected attacks based on network bandwidth information. However, the performance of the anomaly detection system drops when the bandwidth usage is quite small over long periods of time as SARIMA model learns to adapt into the attacks.

In [12], [13] authors propose a novel technique for detecting anomalies in the cloud based on both application metrics such as Tweets per Second (TPS) and system metrics such as CPU utilisation etc. They also used time series production data in order to evaluate their model performance and accuracy. It is also important to identify underlying patterns in Twitter's time series production data in order to increase the efficacy of detecting abnormalities. For that purpose, they used time series decomposition technique in order to decompose time series into three components, seasonal, trend and residual respectively. Although such techniques are powerful, long term time series can degrade learning algorithms efficiency and accuracy as the volume of the data increases linearly through time. The latter indicates the necessity on finding different techniques in order to deal with long time series data. In [14] authors have developed a platform that monitors the system for HTTP traffic or resource utilisation levels across multiple instances of inter-cloud applications. Authors analysed resource usage features and traffic values to trigger autoscaling and avoid system overloading. Their load balancing framework improved the elasticity in the IaaS level and highlighted key requirements of inter-cloud autoscaling platforms.

In [3] authors propose a semantic aware technique based on both application metrics (e.g. database throughput) and resource usage data (e.g. CPU, memory etc.) that have been collected and tagged with the application context. After the collection stage, Recurrent Neural Networks (RNNs) trained

on the pre-labeled data in order to learn different patterns and predict new abnormal wavelets. They deployed both Apache Cassandra and MongoDB NoSQL systems in different Virtual Machines (VMs) and monitored the VMs under intensive workload periods. To demonstrate a real world scenario for anomaly detection, the authors executed YCSB workload in conjunction with different experiments such as fault injection and resource interference to generate synthetic abnormal wavelets. Finally, Dynamic Time Warping algorithm has been used to determine if a new wavelet is normal or abnormal based on administrator's threshold (thres_norm). If the distance pointed by the DTW is bigger than thres_norm, an alarm is generated and all the intuitive information about the anomalous pattern is used to understand the characteristics of an anomalous flow and prevent future abnormalities.

A lot of research has been contacted in order to improve energy efficiency in Data Centers around the world [15], [16], [17]. Amongst other solutions, auto scheduling recommendation systems have been initialised to improve the efficiency on cloud systems and reduce unreasonable energy consumption. Such systems can be based on various parameters such as application metrics, system metrics, HTTP traffic etc. In [18] authors focus on forecasting CPU usage of machines in datacenter using Long-Short-Term Memory (LSTM) Recurrent Neural Networks as well as autoregressive integrated moving average (ARIMA) models. In their findings, after training, hyper-tuning and evaluating both models, LSTM non-linear time series model outperformed ARIMA model for almost all combination of its hyper-parameter.

Although, all these techniques have been effectively worked towards detecting abnormalities on Cloud systems, are mainly focused on data mining and storage methods. In contrast, our method focus on the real time anomaly detection by monitoring resource usage (inspired by the works of [3] and in [4]). We further explore the information from process executions to further support the analysis phase. We aim to associate system runs, with top processes in order to identify processes that could cause abnormal runs. The proposed method is developed upon a flexible architecture that supports information extraction and storage outside of the application side (e.g. the NoSQL system).

VII. Conclusion

We proposed an anomaly detection and prediction system to identify abnormalities in NoSQL systems by monitoring resource usage and process information. Our solution provides useful insights in the information extraction process and the association of resource usage statistics with the process that cause increased usage patterns. We explored various solutions using the YCSB workload and datasets generated by running different workload configurations (data read and update) in NoSQL applications to demonstrate how anomaly detection can identify problems and failures on modern platforms. We also explored the accuracy between different machine learning models such as ARIMA, Seasonal ARIMA and LSTM-RNNs, to identify and predict of future abnormalities based on historical system usage data. The experimental scenarios are prosperous, as we observed that the forecasting of CPU usage using the LSTM-RNN model has shown significant improvement over the ARIMA and SARIMA models when assessing the accuracy of models using the root mean square error. In future, we plan to integrate log data in order to relate abnormal runs with the cause of event. This will help us improve the accuracy of our model by classifying abnormal runs by categories as defined by the application.

References

[1] V. N. Gudivada, D. Rao, and V. V. Raghavan, "Nosql systems for big data management," in *2014 IEEE World congress on services*. IEEE, 2014, pp. 190–197.

[2] T. Garfinkel, M. Rosenblum *et al.*, "A virtual machine introspection based architecture for intrusion detection." in *Ndss*, vol. 3, no. 2003. Citeseer, 2003, pp. 191–206.

[3] A. Bhattacharyya, S. A. J. Jandaghi, S. Sotiriadis, and C. Amza, "Semantic aware online detection of resource anomalies on the cloud," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016, pp. 134–143.

[4] A. Bhattacharyya, S. Sotiriadis, and C. Amza, "Online phase detection and characterization of cloud applications," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2017, pp. 98–105.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 143–154.

[6] L.-M. Liu, G. B. Hudak, G. E. Box, M. E. Muller, and G. C. Tiao, *Forecasting and time series analysis using the SCA statistical system*. Scientific Computing Associates DeKalb, IL, 1992, vol. 1, no. 2.

[7] Y.-W. Cheung and K. S. Lai, "Lag order and critical values of the augmented dickey–fuller test," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] L. Huang, X. Nguyen, M. Garofalakis, M. I. Jordan, A. Joseph, and N. Taft, "In-network pca and anomaly detection," in *Advances in Neural Information Processing Systems*, 2007, pp. 617–624.

[10] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[11] A. Hanbanchong and K. Piromsopa, "Sarima based network bandwidth anomaly detection," in *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2012, pp. 104–108.

[12] O. Vallis, J. Hochenbaum, and A. Kejariwal, "A novel technique for long-term anomaly detection in the cloud," in *6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, 2014.

[13] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," *arXiv preprint arXiv:1704.07706*, 2017.

[14] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Vertical and horizontal elasticity for dynamic virtual machine reconfiguration," *IEEE Transactions on Services Computing*, 2016.

[15] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," 2001.

[16] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu, "Greendcn: A general framework for achieving energy efficiency in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 4–15, 2013.

[17] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing*. IEEE Computer Society, 2010, pp. 826–831.

[18] D. Janardhanan and E. Barrett, "Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2017, pp. 55–60.