QATAR UNIVERSITY

COLLEGE OF ENGINEERING

TIME-AWARE WORKLOAD CHARACTRIZATION AND PREDICTION FOR PROACTIVE AUTO-SCALING OF WEB APPLICATIONS

BY

NADA MAHMOUD ABOUEATA

A Thesis Submitted to

the Faculty of the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

Masters of Science in Computing

June 2019

COMMITTEE PAGE

The members of the Committee approve the Thesis of
Nada Aboueata defended on 17/04/2019.

Dr. Khaled Bashir Shaban
Thesis/Dissertation Supervisor

Dr. Abdelkarim Erradi
Thesis/Dissertation Co-Supervisor

Prof. Abdelaziz Bouras
Committee Member

Prof. Fethi A. Rabhi
Committee Member

Approved:

Abdel Magid Hamouda , Dean, College of Engineering

ABSTRACT

ABOUEATA, NADA, MAHMOUD, Masters: June: 2019, Masters of Science in Computing

Title: Time-Aware Workload Prediction for Proactive Auto-Scaling of Web Applications

Supervisor of Thesis: Khaled, Bashir, Shaban

Proactive auto-scaling techniques aim to predict the future workload of web applications to provision the required resources, such as virtual machines (VMs), ahead of time. Nevertheless, deciding the optimal number of resources to allocate is a challenging task due to the dynamic nature of workload characteristics and the difficulty of predicting them. Most of the existing workload approaches only consider one workload feature which is typically the volume of requests to characterize and predict the workload. In this thesis, we report the design and development of a time-aware workload prediction model that considers the request time features in order to achieve better workload characterization and prediction. We explore two different approaches, namely *Time-Aware Single-Modeling* and *Time-Aware Multi-Modeling*. The *Time-Aware Single-Modeling* approach builds one model for the entire time-space and has three variations: multivariate regression, univariate Long Short-Term Memory Neural Networks (LSTM), and multivariate LSTM neural network model. While, *Time-Aware Multi-Modeling* approach develops a prediction model for each time partition discovered using a periodicity detection component.

The proposed solutions are evaluated using two real workload datasets: Library portal at Qatar University and NewsLink portal in Pakistan. The results demonstrate that the time-aware approaches achieve more accurate predictions of the workload

patterns compared to other existing approaches. Also, it has been shown that the achieved improvements are statistically different than existing approaches.

DEDICATION

To my parents, for their endless support, love, and faith in me.

To my siblings, and special thanks to my brothers Khaled and Waleed for their endless support, love and for helping me in all the struggles I had with my thesis.

To my friends, my supporters who never let me down, thanks for your love, support and all adventures we had during this journey

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Dr. Khaled and my co-supervisor Dr. Abdelkarim, for their continuous support and encouragement. This thesis would not have been completed without their valuable supervision, patience, and support. Honestly, it was my honor to work with them.

Thank you from the bottom of my heart!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

With the increased hosting of web applications on the cloud, a wide variety of techniques have been developed to improve the scalability of hosted applications while minimizing the cost. One of the main characteristics of cloud hosting is *elasticity* as the application owners can acquire, and release resources as needed. These resources include virtual machines (VMs), computation storage, and bandwidth. Deciding and scaling the appropriate resources to provision to maintain the desired application performance remains a challenging problem. Under-provisioning of resources may lead to performance degradation while over-provisioning incurs unnecessary cost wasted on unused resources. In the case of burst workload, the under-provisioned application leads to costly Service Level Agreement (SLA) violations. Hence, there is a trade-off between meeting SLAs and minimizing the provisioning cost. Therefore, it is necessary to develop effective techniques to automatically allocate resources according to the current application demand. For this purpose, many auto-scaling techniques have been proposed in the literature.

Auto-scaling techniques are classified into two categories namely, reactive and proactive. Reactive techniques are based on a set of predefined rules defined by the system administrator. These rules define thresholds on performance metrics such as the response time and/or on hardware counters such as CPU, RAM, and bandwidth usage. When an indicator crosses the threshold, this trigger scaling to bring the system back to a normal state. However, such scaling takes places after the performance degradation occurs and may lead to SLA violations during the time it takes to provision additional resources. Additionally, reactive autoscaling is incapable of dealing with a rapid increase in workload [21]. On the other hand, Proactive based techniques aim to predict

the required number of resources beforehand to avoid any overhead delay in response time. Generally, scaling can either be done horizontally or vertically. Horizontal scaling refers to adding or removing virtual machines (VMs). While, vertical scaling refers to adapting the number of resources allocated, such as RAM and CPU, to an already running VM. Due to the time needed to boot newly added VMs, proactive based techniques are gaining significant interest compared to reactive based techniques.

1.1. Problem Statement

Most of the existing workload characterization and prediction methods only consider the number of incoming requests (i.e. arrival rate) [3, 5, 8, 15, 24, 28] to characterize the workload. However, web applications workload is a complex multi-facets concept which can be characterized by different properties such as the workload periodicity, burstiness, mean request arrival rate, and variance of the request arrival rate, etc. [6]. For instance, the resources needed to serve 1000 *heavy* load requests are different than serving the same number of requests but generating *low* load. Recently the work proposed in [11] considers both the response document size and the response time to characterize and predict the workload. However, such approach does not consider the temporal characteristics of the workload. Generally, web applications are subjected to dynamic workloads which could exhibit repeated pattern over time. In fact, the time taken by the server to respond can be influenced by both the requested document size and the number of requests submitted to the server at the current time t. Therefore, we believe that considering the workload temporal features can improve the workload characterization and increase the prediction accuracy. There are different ways to consider and include the request time characteristics. In this study, we propose two different models: Time-Aware Single-Modeling and Time-Aware Multi-Modeling. The proposed approach extends the work proposed in [11]. We consider the request

time, in addition to the document size and response time features considered by [11] with the aim of achieving better workload characterization and prediction.

1.2. Thesis Objective

The focus of this thesis is to achieve more accurate predictions of the workload. The following are the main objectives of the thesis:

- Design a time-aware approach that incorporates the request time characteristics in order to achieve better workload characterization and more accurate prediction.
- Investigate and evaluate different ways of including the request time characteristics to improve the workload predictions.
- Conduct a comprehensive performance evaluation to compare the proposed time-aware approach with the state-of-art methods reported in the literature.

1.3. Thesis Significance

We believe that this work is highly important as more accurate workload prediction allows better autoscaling to meet the application desired performance at a minimal cost. This yields improved application performance and the reduction of SLA violations. Additionally, it could also be beneficial for the service providers seeking to optimize their resources utilization.

Although many research works have been done in this area, there are still opportunities for further improvement in workload characterization and prediction accuracy.

The remainder of the thesis is organized as follows. Chapter 2 presents the background and concepts needed for the remainder of this thesis. First, it discusses the general concepts of time series analysis. Then it introduces motif discovery and

Minimum Description Length (MDL) based time series clustering. Chapter 3, presents a review of the related work in workload characterization and periodicity detection area. Our developed solutions and its different variations are presented in Chapter 4. Chapter 5, discusses the experimental setup, datasets, and obtained results. Finally, we conclude the thesis and present some directions for future work.

Chapter 2: Background

In this chapter, we present the background and concepts needed for the remainder of this thesis. Section 2.1. presents the general concepts of time series analysis. Section 2.2. & Section 2.3. introduces motif discovery and MDL-based time series clustering, respectively.

2.1. Time Series Analysis

Definition 1. *A **Time Series** T* of length n is an ordered list of *n* real observations which collected at fixed time intervals, where

$$T = \{ t_1 , t_2 , t_3 , \dots . , t_n \}$$

Definition 2. A ***Time Series Sampling Rate*** is defined as the sampling rate *r* at which the time series observations are collected. Formally,

$$T^r = \{ t_1^r , t_2^r , t_3^r , \dots . , t_n^r \}$$

Definition 3. A ***Subsequence S*** of length *m* of the time series *T* of length *n* is a short *m-length* time series subsets of *T*, which defined as

$$S_i^m = \{ t_i , t_{i+1} , t_{i+2} , \dots . , t_{i+m-1} \} \quad Where \quad 1 \leq i \leq n + m - 1$$

Definition 4. A ***Motif M*** of length *k* is the pair of the two most similar subsequences of length *k*, which defined as

$$M_k = \{ S_i^k , S_j^k \}$$

2.2. Motif Discovery in Time Series

Motif discovery is an important subroutine that has been used in different time series mining tasks and applications such as classification, clustering and anomaly detection [29]. It is used to find the two most similar subsequences in a time series. The most well-known motif discovery algorithm "Mueen-Keogh" (MK) is claimed to be the fastest algorithm. However, MK method is an exact algorithm that runs for a fixed motif length specified by the user, and considered to be intractable (i.e. not scalable) for enumeration purposes due to the massive distance computations needed for all motif lengths [22]. In fact, deciding the optimal length of motif is a challenging task for realistic datasets. Thus, to overcome this problem, a motif discovery method has been proposed called MOEN to enumerate all motifs for a range of lengths. MOEN algorithm searches for all motifs within the specified range of lengths and outputs only the maximally covering motifs, where no other motifs cover it.

MOEN algorithm frees the user from the burden of specifying the exact length of motif, also it has been proved that it is an order of magnitude faster than the native solutions of motif discovery [22]. The speed-up comes from the lower bound (LB) applied on the normalized distances to prune most of the distance computations for the next motif lengths once the best motif of the first length is found. To illustrate the main intuition behind the lower bound in speeding-up the algorithm, we show an example below – taken from [22].

Example

Given the motifs length that ranges between (7, 9). First, the list of top-k motifs of length 7 is generated, as shown in *Figure 1* (A). After the list is generated, the lower bound of the distance for the next length 8 is computed using the max distance in the first list. In fact, this lower bound is used to exclude the motifs that have distances

greater than lower bound in the subsequent lengths, as shown in Figure 1 (C). This is because none of the skipped motifs can have smaller distance than the lower bound for the next length. Therefore, skipping these motifs helps to confine the search space and thus reduce the time complexity.

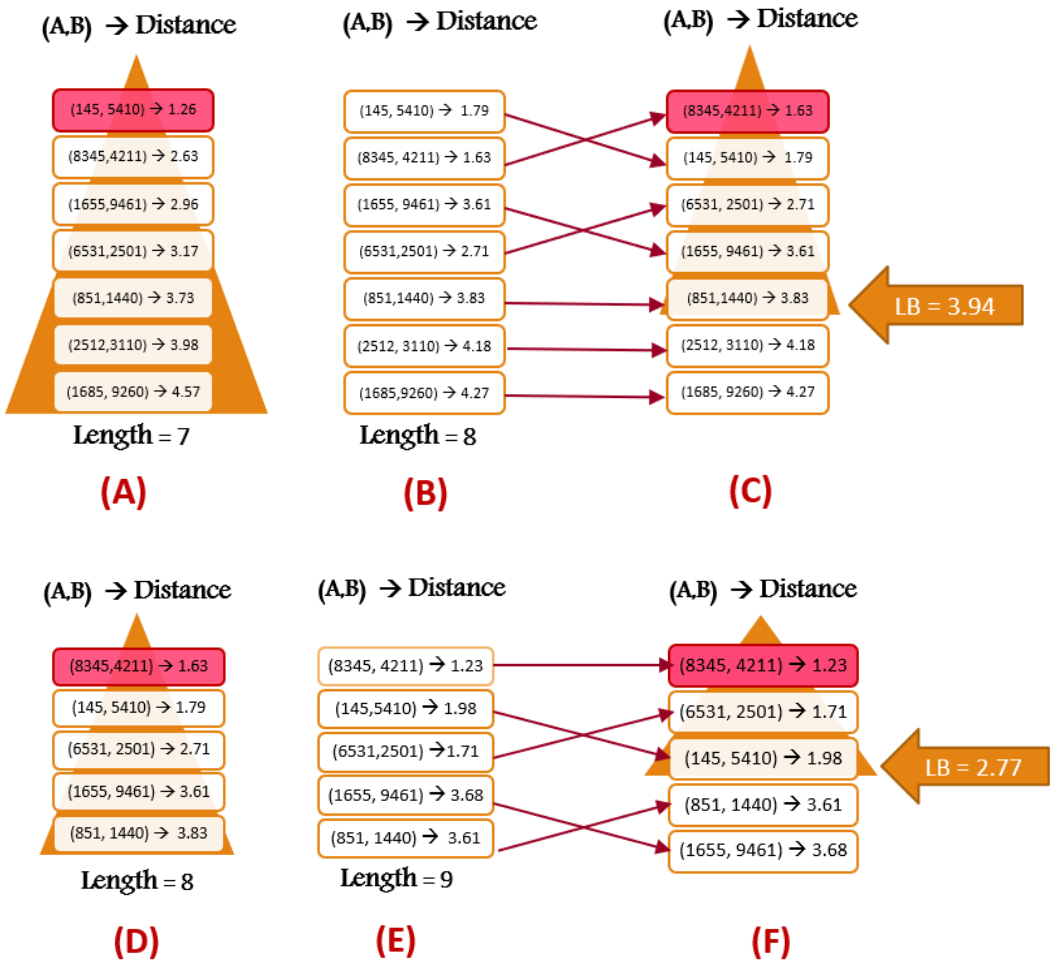

Figure 1: MOEN enumeration example

2.3. MDL-based Time Series Clustering

The description length (DL) of data *m* is defined as the number of bits needed to describe the data *m*. The main intuition behind the Minimum Description Length (MDL) concept is to find a model or hypothesis that has the best ability to compress on a given dataset. In the context of time series clustering, the hypothesis is declared as the cluster center and the more compression ability of the hypothesis, is the more similarity of subsequences would be. To illustrate the MDL principle, we show an example below – adapted from [26]- using discrete data structure.

Example

Given the name "Felix" with its variations in different languages: Felix (English), Felice (Italian), and Félix (French). If it requires 8 bits to store each letter, then the description length for all names → DL(D) = $(5 * 8) + (6 * 8) + (5 * 8) = 128\ bits$. On the hand, with MDL, if "Felix" is determined to be the hypothesis, then we do not have to store all names except the hypothesis and the difference between hypothesis and other names variations. Through MDL, we use only 16 bits to store the last two letters of the 2nd name, and 8 bits to store the second letter of the 3rd name, as shown in *Table 1*. Thus, in total we use only 64 bits to store all names → DL(D|H) = $(5 * 8) + (2 * 8) + (1 * 8) = 64\ bits$. Notice that the bitsave indicates the compression ability of the hypothesis and the similarity degree of other names variations according to the hypothesis.

Table 1. Example of the MDL Principle

D	H = null DL(H) = 0 DL(D)	H = 'Felix' DL(H) = 40 DL(D|H)
Felix	40	0
Felice	48	16
Félix	40	8
Total	128 bits	64 bits

Next, we present the required definitions of MDL in the context of time series clustering.

Definition 5. The ***entropy*** of a time series *T* is a lower bound of the total number of bits required to represent *T*, which defined as

$$H(T) = -\sum_t P(T = t)\log_2 P(T = t)$$

$where$ P is the probability that symbol t will occur

Definition 6. A ***description length (DL)*** of a time series *T* is defined as the total number of bits required to represent *T*, which can be calculated as follows:

$$DL(T) = m \times H(T)$$

Definition 7. A ***Hypothesis (H)*** is defined as the subsequence that used to encode or compress other subsequences.

Definition 8. A ***conditional description length*** of subsequence *A*, given the hypothesis *H*, is defined as follows:

$$DL(A \mid H) = DL(A - H)$$

Definition 9. A ***Description Length of a Cluster C (DLC)*** is the total number of bits required to represent all subsequences belongs to *C*. The center of the cluster is

presented as the hypothesis *H* which used to encode other subsequences. *DLC* of cluster *C* is defined as follows:

$$DLC(C) = DL(H) + \sum_{A \in C} DL(A \mid H)$$

Definition 10. A ***bitsave*** is the total numbers of bits saved after applying a specific action. Formally, ***bitsave*** is defined as follows:

$$bitsave = DL(Before) - DL(After)$$

In the context of time series clustering, bitsave is calculated after applying the following actions:

- Creating new cluster C' using subsequence *A* and *B*

$$bitsave = \big(DL(A) + DL(B) \big) - DLC(C')$$

- Adding subsequence *A* to existing cluster *C*

$$bitsave = \big(DL(A) + DLC(C) \big) - DLC(C')$$

- Merging two clusters, C_1 and C_2 to cluster C'

$$bitsave = \big(DLC(C_1) + DLC(C_2) \big) - DLC(C')$$

Chapter 3: Literature review

In this chapter, we review some of the works that have done in the areas of workload characterization and prediction, and periodicity detection. Because none of the periodicity detection studies have been applied to the web workload domain, the papers that have been reviewed in this section are categorized into:

1. Workload Characterization and Prediction.
2. Periodicity Detection.

3.1. Workload Characterization and Prediction

Generally, workloads can be investigated and characterized from two different perspectives: *Conventional web workload* and *Cloud workload*, as shown in Figure 2.

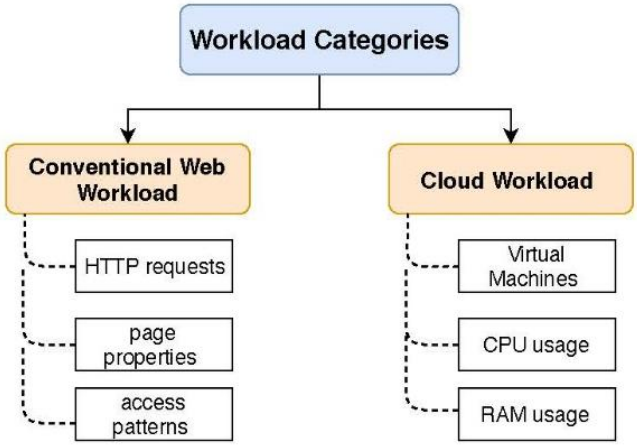

Figure 2. Workload characterization categories

Conventional web workloads are characterized in terms of HTTP requests issued by the clients, page properties, and access patterns [6]. On the other hand, cloud workloads are characterized in terms of resources usage, such as virtual machines, CPU

and RAM utilization [6]. Many research studies have been proposed and experimented with different solutions to characterize and predict the workload. Some of the works characterize the web workloads in terms of HTTP requests arrival rate, as done in [3, 5, 8, 15, 24, 28]. Meanwhile, other works characterize the workload of applications deployed on cloud infrastructures in terms of hardware counters, such as CPU and RAM utilization, as done in [9, 10, 12, 16, 20, 30]. The papers that have been reviewed in this section are categorized into *Web Workload* and *Cloud Workload* characterization and prediction.

3.1.1. Web Workload Characterization and Prediction

Most of the existing works characterize the application workload only in terms of requests arrival, as done in [3, 5, 8, 15, 24, 28]. Generally, time series prediction models are categorized into two classes, namely stochastic and deep neural networks models. Time series data can represent different forms of stochastic processes. The most widely stochastic prediction models used in the literature are, *Autoregressive (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA)* [32]. Also, deep neural networks models have proven their effectiveness in workload prediction. The most common and popular time series neural networks models used in the literature are, *Feed-Forward (FFN), Backpropagation (BNN), Time Delay (TDNN), and Error Correction neural network (ECNN)* [24]. The papers that have been reviewed in this section are categorized based on the technique used in the prediction phase into *Deep Neural Networks Models* and *Stochastic Models* .

3.1.1.1. *Deep Neural Networks Prediction Models*

Nikravesh et al. [24] leverage both machine learning techniques, SVM and back propagation Neural Network, to implement the workload prediction model. The novelty

of this study is by building an adaptive workload prediction which chooses automatically the appropriate prediction model, either SVM or NN, based on the incoming workload pattern. In this work, they focus on three types of workload which are: growing workload, periodic workload and unpredictable workload. The results showed that the SVM trained model performs better with the growing and periodic workload, while NN trained model has better prediction with the unpredictable workload. As opposed to the work done in [24], Kumar et al. [15] make use of feed-forward Neural Network and self-adaptive differential evolution algorithm. The model is trained using the self-adaptive differential evolution algorithm (SaDE) to learn the optimal hyper-parameters for the NN model. The SaDE approach has proven its effectiveness in training feed-forward neural network [17], compared to other population-based approaches such as PSO, GA, etc [1, 13]. The proposed model compared with the well-known back-propagation network algorithm and the proposed model was capable to reduce the prediction error up-to 0.001.

3.1.1.2. *Stochastic Prediction Models*

Differently, studies in [3, 5, 8] use different time series stochastic models for workload prediction. Calheiros et al. [5] make use of ARIMA statistical model for workload prediction. The proposed model applies feedback from the last observed load to update the model on the run. The predicted workload is passed to a scaling decision model to allocate resources accordingly. The experiments showed that the proposed model leads to efficiency in resource utilization with minimal impact in QoS for users. Similarly, Ali-eldin et al. [3] analyze the trend and seasonality of Wikipedia website workload using time series models. Based on the analyzed workload patterns, they propose a simple cubic spline prediction algorithm that can predict workload patterns, in terms of the number of requests (i.e arrival rate), with MAPE of 2%.

As opposed to the work done in [3, 5], Cezar et al. [8] develop an adaptive method that uses different combinations of time series models, instead of developing a single time series prediction model for all cases of workload patterns. The selection of the best prediction model is decided by a genetic algorithm. They use five different statistical models, which are: Naïve model, autoregressive model, autoregressive moving average, and Autoregressive Integrated Moving Average. The proposed approach leads to an accurate prediction of the application workload. In addition, it shows that the proposed method can adapt different pattern of time series data and does not need a huge volume of historical data for prediction models training.

3.1.2. Cloud Workload Characterization and Prediction

Some of the works in proactive auto-scaling consider and characterize the cloud workload in terms of resources usage, such as CPU and RAM usage, as done in [9, 10, 12, 16, 20, 30]. Similarly, to section 3.1.1. , the papers that have been reviewed in this section are categorized based on the technique used in the prediction phase into *stochastic models* and *neural networks models.*

3.1.2.1. *Deep Neural Networks Prediction Models*

Recently, machine learning techniques have proven their efficiency and effectiveness in cloud workload prediction [12, 30]. Tran et al. [30] proposed a forecasting model using Neural Network technique. This work exploits CPU and RAM usages simultaneously. The main intuition behind this idea is to predict each resource usage with considering the relation between all historical resource usages, CPU and RAM usages. So in that way, they avoid missing any relationship among the historical resource usage in the prediction phase. The proposed prediction phase consists of three main sub-components: preprocessor, learning algorithm, and forecasting model. The preprocessor component uses a fuzzy approach to transform

the monitored resource usage into a fuzzy time series, which later used as an input of the learning component. They use the neural networks algorithm to exploit the historical data of CPU and RAM resource utilization and then predict the future values accordingly. The predicted values are then passed to a scaling decision model to make the final decision of VMs resources allocation.

Similarly, Islam et al. [12], utilize Error Correction Neural Network (ECNN) and linear regression algorithms to forecast the future demand of CPU usage. During the training phase, the neural network and linear regression are fed with historical data of CPU usages to predict the future demand. The produced models from the training phase, are validated using sliding window technique, where the input x represents a vector of resource usages over k time intervals and the predicted value y, is r steps ahead of the input window [12]. The results show that the neural network model using the optimal window size outperform the prediction accuracy of linear regression models.

As opposed to the works done in [12, 30], the works in [9, 16] develop an adaptive workload prediction model based on the workload pattern. Both works [9, 16] focus on forecasting CPU resource usage. Gong et al. [9] proposed two prediction model, namely: signature-driven and state-driven prediction model. The signature model is used when historical resource usages exhibit repeated patterns, meanwhile, the state-driven model is used when the historical resource usages does not follow any pattern. Differently, Liu et al. [16] proposed to assign different prediction models according to the structural change of the workload. They classified workloads into two categories: fast time-scale data and slow time-scale data. The results showed that the linear regression model manages to predict the slow time-scale data accurately. In the meantime, SVM model was able to adopt the dramatic changes in fast time-scale

data.

3.1.2.2. *Stochastic Prediction Models*

The most common stochastic prediction model used in the literature is ARIMA, as done in [10, 20]. Huang et al. [10] proposed a Prediction-based Dynamic Resource Scheduling (PDRS) approach for resource allocation of virtualized cloud systems. The proposed system consists of four main components: NodeAget, System Monitor, Resource Demand Predictor, and Elastic Scheduler. The NodeAget is deployed on each cluster server, and it is responsible to collect the resource usages statistics, which represents the CPU usage in this study. These statistics are then collected by the System Monitor component for a predefined measurement interval. Thereafter, the Resource Demand Predictor employs an ARIMA model to predict the usage statistics collected by the System Monitor component. The main novelty of this work is that the resource demand predictor component employs multi-step prediction, first to predict the future usage of each resource type, then analyze the predicted CPU usages to determine the state of the server in the next scheduling interval. Each state has its lower and upper bound of the CPU usage. Predicting the server state is helpful to overcome the issue of over-estimation or under-estimation of resource usages prediction. Based on the predicted information, resource usages, and server state, the Elastic Scheduler component allocates the required number of VMs. The results showed that the proposed system improves the prediction error and allocate resources with an acceptable level of SLA. Similarly, Meng et al. [20] proposed CURPA, a container resource utilization prediction algorithm which based on ARIMA time series model. However, this work leverages the docker container as it is more lightweight than the virtual machines and easier to be deployed. The resources usage considered in this study is the CPU utilization. The CPU usages are used to train a

prediction model based on ARIMA, which later will be predicted by the trained model. The predicted usage is then used by a provision model to allocate the sufficient number of resources by adopting a horizontal auto-scaling approach to increase or decrease the number of containers. The results showed that by taking advantage of ARIMA and Docker container, CURPA managed to achieve high prediction accuracy.

3.2. Periodicity Detection

One way to include the request time characteristic is by considering the workload periodicity. Many works in the literature have been proposed to address the periodicity detection problem in many fields such as finance, e-commerce, medicine, science, image analysis, etc. The papers that have been reviewed in this section are categorized based on the periodicity detection technique that has been used: *Autocorrelation* and *Time Series Clustering*, as shown in Figure 3.

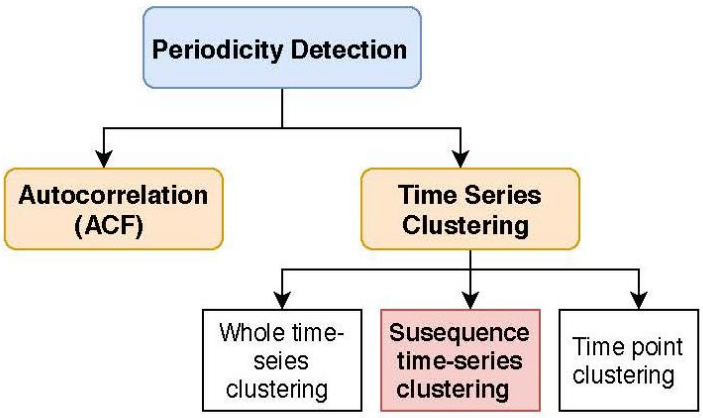

Figure 3. Periodicity detection techniques

3.2.1. Autocorrelation

Autocorrelation function (ACF) is one of the tools used to find repeated patterns in time series data. ACF measures the correlation between the original signal and a shifted version of itself with different time lags. The correlation values range between 1 and -1. Intuitively, if the correlation is near 1 then a repeated pattern is declared at lag t, while an autocorrelation near 1 means that the signal has an opposite direction at lag t. On the other hand, a random signal has an autocorrelation of 0 at all-time lags.

Many studies consider ACF for periodic pattern identification [7, 31]. Vlachos et. al [31] proposed a non-parametric approach which based on ACF and DFT for periodicity detection in time series data. They utilize a two-tier approach by combining both methods, Periodogram and Autocorrelation method, which called AUTOPERIOD. The first step consists of extracting the candidates periods, e.g., hints, by using the Periodogram method. These candidates may be false due to the spectral leakage problem in the periodogram method. Thus, they have developed a verification step by using the autocorrelation function, to verify the correctness of the candidate periods produced by the periodogram. The intuition is that if the candidate periods reside on the hill of the ACF, then this is considered as a correct period. Otherwise, the candidate period is discarded and considered as a false alarm. Similarly, Calzarossa et al. [7] analyzes and predicts the dynamic of web content changes. They proposed system consist of two components: pattern identification and forecasting component. In the pattern identification component, they utilize the ACF and decomposition method to identify temporal patterns. The ACF function computed at different time lags ranges varying from one hour up to one week. The peaks detected by the autocorrelation function identify daily patterns of the time series which

repeated every 24 hours. To identify the temporal patterns more accurately, they decompose time series data into three components: trend, seasonality and remainder components. The weekly and daily patterns are captured by the trend and seasonal competent, respectively. After that, the forecasting component uses numerical fitting techniques to identify the parameters of the trigonometric polynomial that best fit the trend and seasonal patterns. On the other hand, it uses ARIMA model to fit the remainder part.

Despite the success of ACF function in periodic pattern extraction, ACF doesn't give the details about when exactly each identified pattern occurs. For example, ACF might give you that there is a pattern which is repeated every 5 hours, however without any insights when exactly this pattern is repeated within the day (i.e., the starting and ending hour of the repeated pattern). For this purpose, Time series clustering techniques are discussed in the next section to address this limitation.

3.2.2. Time Series Clustering

As mentioned in the previous section that ACF doesn't declare clearly the time boundaries of the discovered periodic patterns. One way to address this gap is by using time series clustering techniques. Clustering is the most common solution to uncover hidden repeated patterns in time series data. As stated in [2], time series clustering related works are classified into three categories as following:

a) **Whole Time-series clustering:** This clustering is similar to the conventional ways to cluster discrete objects. In the context of time series, discrete objects are presented as a set of individual time series, with the objective to group similar time series into the same cluster/group.

b) **Subsequence Time-series clustering (STS):** As opposed to the Whole time-series clustering, subsequence clustering is performed on a single stream of time

series data. This category consists of two steps: first extracting set of subsequences from the original time series stream using a sliding window, and then cluster similar subsequences into same groups.

c) **Time-point clustering:** The notation of clustering here is clustering time points based on a combination of their temporal proximity and similarity between time points values.

Given all time-series clustering categories discussed above, STS clustering category is the best fit for this thesis because the web applications workload is presented as a single stream of time series data. Web workload time series data should be partitioned into meaningful subsequences, and then cluster these subsequences into groups. By this way, periodic patterns can be declared using these clusters.

3.2.3. Subsequence Time Series Clustering

Nevertheless, extracting the most meaningful subsequences for clustering from a single stream of time series data is a challenging task. As stated in [14], all studies conducted before 2003 tried to cluster every single subsequence in the original time series which leads to the well-known issue of meaningless clustering results in STS clustering. Therefore, many works have been proposed later to address this issue. The papers that have been reviewed in this section focus mainly on the time series clustering algorithms which based on the minimum description length (MDL) distance measure.

3.2.3.1. *MDL-based Clustering Method*

The most recent works on STS clustering (STS) [18, 19, 26, 27] address the problem of meaningless clustering results by the following ideas. First, STS clustering requires ignoring some data from the time series. In fact, the observations proved that any attempt on trying to cluster every single subsequence of a time series might lead to

meaningless results. The meaningless results are due to the noisy parts included in the clustering process. Thus, ignoring some subsequences is needed to avoid any meaningless results. Second, clustered subsequences are not allowed to overlap with each other. By using these two ideas, [18, 19, 26, 27] resolve the meaningless problem in STS clustering. All studies in [18, 19, 26, 27] leverage motif discovery methods to extract meaningful subsequences. Motif discovery method aims to find the two most similar subsequences in a given time series. The most common algorithm used in the literature to find the most similar pair of motifs is MK algorithm [23].

Both works in [26, 27] use MK motif discovery algorithm to subsequence the original time series and find out the most meaningful list of motif pairs. Given the list of motif pairs discovered by MK, authors in [26] make use of the Minimum Description Length (MDL) similarity measure to find the best hypothesis that has the highest ability to compress the data. The compression ability indicates the degree of similarity between the model and subsequences. For example, if the compression ability achieved by the model is high, this indicates that there is a high similarity between the model and subsequences. The proposed clustering algorithm defines three operations, which are: creating a cluster, adding to cluster, and merging clusters. The proposed approach iteratively adopts the idea of MDL to calculate the number of bits saved for each operation and then chooses the operation that maximizes the number of bits saved. Differently, the work proposed in [27] uses data encoding distance measure to choose the optimal operation in each step of clusters construction.

Despite the success of both approaches [26, 27] in achieving meaningful clustering results, a predefined parameter is still needed by the MK algorithm to define the approximate length of subsequences. Nevertheless, defining the optimal length of the subsequence is a challenging task. Therefore, many works in the literature have

been proposed to address this issue by automating the process of finding the optimal length of subsequences.

Madicar et al. [18] proposed a parameter-free STS clustering which eliminates the burden on the users to set the subsequence length parameter. The proper length of the subsequence is found by running the MK motif discovery algorithm [23] iteratively at different lengths starting from length 2 up-to-half of the time series length. The discovered motifs are then ranked based on the Euclidean distance similarity measure and their frequency. After that, all k^{th}-best motif are passed to the clustering algorithm. Similarly to [26, 27], the clustering process in [18] defines three operations: add, create and merge. The error for each operation is calculated and then choose the operation that minimizes the error. The error per cluster is defined as the summation of Euclidean distance between each subsequence in a cluster.

Madicar et al. [19] proposed an enhanced version of [18] to improve the quality of the discovered motifs and to address the high variability in the subsequences widths. Similar to the work in [18], [19] consists of two main phases: selection of the proper subsequence length process and clustering process. In the first phase, they combine both MDL and MK motif discovery to improve the quality of the discovered motifs. The MDL is used to rank the priority of the motif results. For the clustering process, it uses the same idea of the clustering process used in [18, 26, 27]. The results proved the robustness of the proposed approach in handling the high variability of subsequences length.

3.2.4. Discussion

In this section, we present a comparison between the proposed approaches under autocorrelation and subsequence time series clustering. The strengths and weaknesses of the proposed works under each method are presented in *Table 2*. The strength is assessed in terms of two dimensions, namely, parameter-free and scalability on large datasets. On the other hand, weaknesses are evaluated in terms of lack of time boundaries. It can be noticed from Table 2, subsequences time series clustering approaches address the gap of declaring the time boundaries of the discovered pattern. However, those algorithms are not scalable on large datasets due to the motif discovery method used in these studies. More details on an alternative scalable method of motif discovery, is already presented in section 2.2.

Table 2.Strengths and Weaknesses of Periodicity Detection Researches.

Article	Method	Strength		Weakness
		Parameter-free	Scalability	Lack of time boundaries
[31]	ACF	✓	✓	✓
[7]		✓	✓	✓
[19]	Subsequence Time Series Clustering	✓		
[18]		✓		
[26]				
[27]				

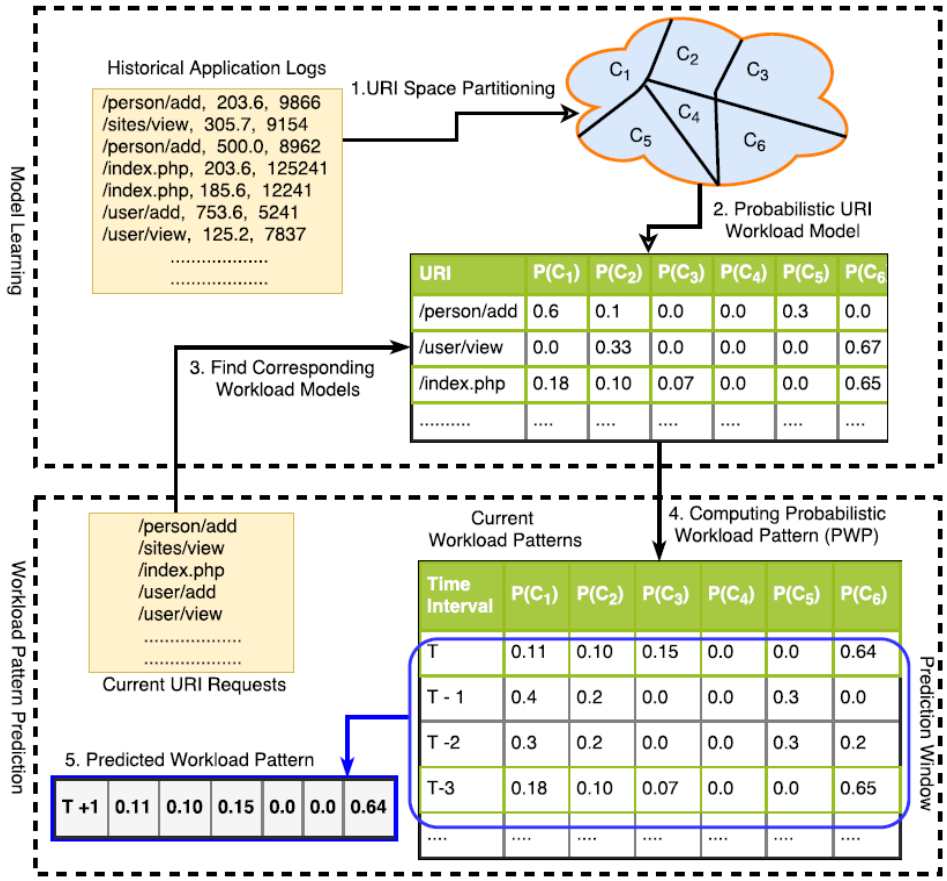

Figure 4. The Baseline system architecture – taken from [11]

3.3. Baseline System

The proposed approach extends the work presented in [11]. In this study, we consider [11] as the baseline system that is not using *time-aware* approach. The baseline system, shown in Figure 4, composed of two models namely, workload characterization model and workload pattern prediction model. The detailed components of each model are described in the following subsections.

3.3.1. Workload Characterization Model

The workload characterization model consists of three main components:

a) URI Space Partitioning:

The application URIs are partitioned into different partitions based on two features: response time and document size. The partitioning model uses the unsupervised clustering algorithm, K-means, to partition the application URIs.

b) Probabilistic URI Workload Modeling:

Given the URIs space partitions, each URI is modeled by a probabilistic representation vector (v_p). The probabilistic workload modeling measures the probability of the URI to lie in each cluster. The URI probability shows the distribution of URI along each partition, which given by:

$$v_{p_{(i)}} = \frac{q_i}{q_t}$$

where q_i are the URI occurrences in the i-th partition and q_i is the total number of the URI occurrences over all partitions.

c) Workload Pattern Identification

Given the URIs distribution in each cluster, this component computes the representation vector (i.e., workload pattern) of the received URIs in a given time interval. This vector shows the distribution of URIs along each cluster, and the summation of all distributions yield the total number of incoming requests in that time interval. Later in this thesis, we refer to the time interval by a sampling rate.

3.3.2. Workload Pattern Prediction Model

The workload prediction model considers two methods of time series forecasting namely, moving average and non-negative least square regression (NNLS) methods. The prediction model will be used later to predict the workload patterns.

Chapter 4: Methodology

The proposed approach extends the work presented in [11] by adding a temporal dimension that considers the request time, in addition to the document size and response time features in order to deliver better workload characterization and prediction. The request time characteristic is included using two different approaches: *Time-Aware Single-Modeling* and *Time-Aware Multi-Modeling*. The upcoming sections present the details of how the proposed approaches extend the baseline system models.

4.1. Workload Characterization Model

This step involves the process of generating the workload patterns (i.e. URIs distribution) given a predefined sampling rate, as described in Section 3.3. Two models were investigated: Time-Aware Single-Modeling and Time-Aware Multi-Modeling.

4.1.1. Time-Aware Single-Modeling

In this approach, the workload characterization model generates the workload patterns for the entire time-space. Thus, the entire application logs are passed to the characterization model to generate the workload patterns.

4.1.2. Time-Aware Multi-Modeling

Contrary to the *Single-Modeling* approach, the multi-models approach generates the workload patterns per each time partition as illustrated in Figure 5. The time partitions are identified by a periodicity detection component. The next subsections present the details of the proposed periodicity detection component.

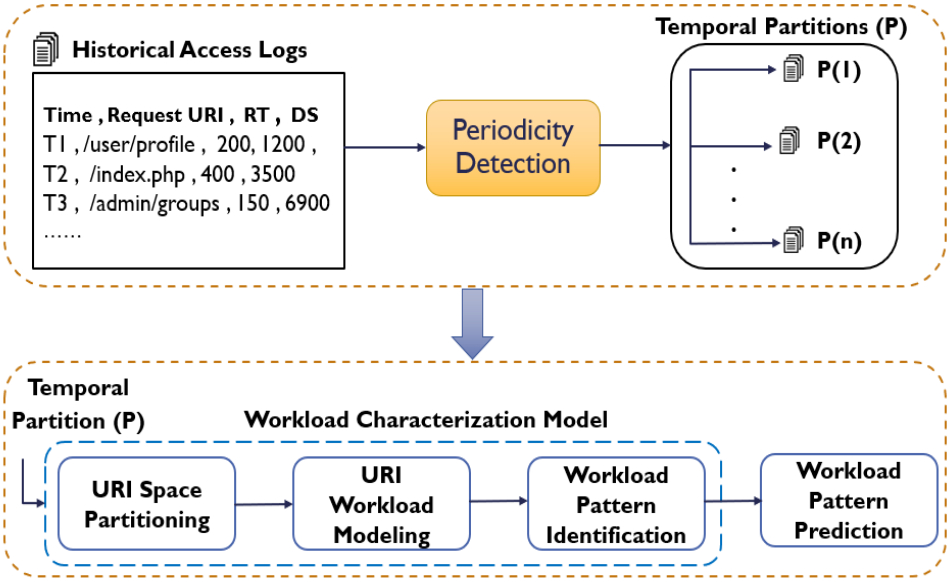

Figure 5: Time-Aware workload prediction using periodicity detection

4.1.2.1. *The Proposed Periodicity Detection Component*

The periodicity detection component consists of two main models: *Intensity-based Workload Partitioning* and *Hierarchical Time partitioning*, as shown in Figure 6. The *Intensity-based Workload Partitioning* model partitions the entire URIs space into different partitions based on the intensity of the workload. The workload is considered as the number of requests. Then, the discovered partitions are passed to the *Hierarchical Time Partitioning* model to extract the periodic patterns. The detailed components of each model are described in the next subsections.

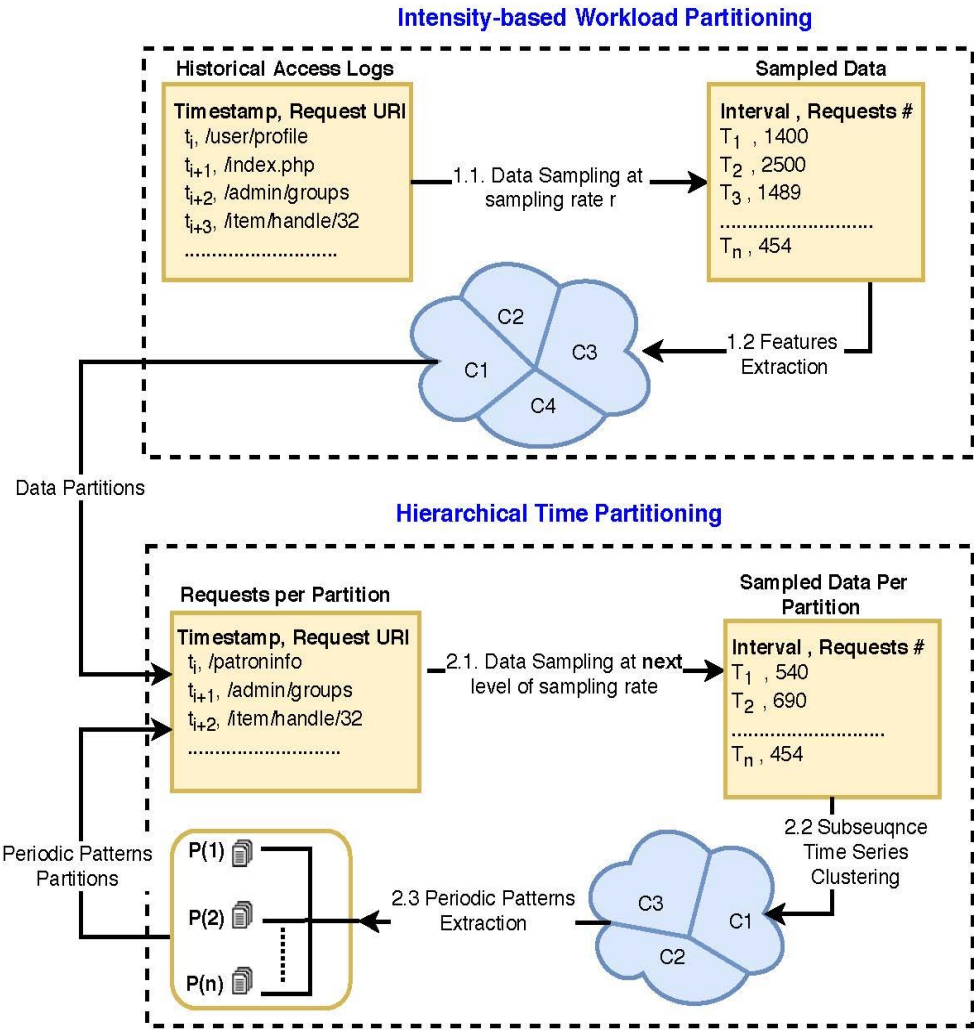

Figure 6: Architecture of the proposed periodicity detection component

Figure 7. Two workload patterns that similar in structure but different in intensity

a) Intensity-based Workload Partitioning

This model is responsible for partitioning the URIs space based on the workload intensity, before passing it to the hierarchical time partitioning model. The workload pattern intensity is measured as the number of requests received at time t. As mentioned earlier in Section 3.2.3. , the subsequence time series clustering methods are used to extract periodic patterns. A popular approach of the time series clustering is structure-based clustering, which looks for similar patterns in change (i.e. structural similarity) regardless of the pattern's intensity (i.e. amplitude of signal) [2]. For example, two workload patterns might be identified as similar patterns because they have similar structure although they may have different intensity as illustrated in Figure 7. In fact, workload patterns with similar structures and different intensity should be considered different as each one requires different amount resources based on the intensity. Therefore, to avoid this problem we propose an intensity-based workload partitioning

which is responsible to partition the URIs space based on the workload intensity, before performing the hierarchical time partitioning for periodic patterns extraction. The proposed model is composed of two main components: Data Sampling and Features Extraction component. The proposed model is composed of two main components: *Data Sampling* and *Features Extraction* component. The *Data Sampling* component samples the historical logs using a predefined sampling rate. For example, if the sampling rate is hour, then the sampled data (S) is expressed as the total number of requests received per hour. The *Features Extraction* components uses k-means clustering algorithm to partition the sampled data using four statistical features, which are: mean (S_{mean}), standard deviation (S_{std}), minimum (S_{min}) and maximum (S_{max}).

b) Hierarchical Time Partitioning

Given the data partitions produced by the previous model, this model extracts periodic patterns in a top-down hierarchical fashion. Generally, the periodic patterns are identified using a subsequence time series clustering method (STS) given a predefined sampling rate. The levels in this model are computed recursively, such that each periodic pattern identified at level n, is the input to the next level to extract the periodic patterns from each. In addition, the sampling rate used at each level is reduced as we go down the hierarchy. The main intuition behind this idea is that zooming-in into small partitions at each level of the hierarchy might help in extracting more meaningful periodic patterns. For example, if *k* periodic patterns are extracted at level *n*, then at level (*n+1*) our approach will zoom-in into each pattern declared at the previous level *n* to extract periodic patterns from each, as illustrated in Figure 16 and Figure 17. This model is composed of three main components: Data sampling, Subsequence time series clustering, and Periodic patterns extraction. The details of these components are explained in the next subsections.

i. Data Sampling

Given the URIs space per partition, these partitions are sampled using a sampling rate which gets reduced as we go down the hierarchy. The main intuition behind this idea is that the sampling rate is directly proportional to the size of partitions at each level. For example, monthly sampling rate would be used to extract patterns from a yearly dataset, meanwhile, daily sampling rate would be used with monthly datasets. Therefore, in this component, if data partitions at level n are sampled using a monthly rate, then subsequent partitions at level n+1 are sampled using a daily rate.

ii. Subsequence Time Series Clustering

Afterward, the sampled data is passed to a subsequence time series clustering (STS) method to extract similar patterns. For this purpose, we use the MDL-based clustering method proposed in [26]. This method [26], consists mainly of two stages: subsequences extraction and clustering stage. In the subsequence extraction phase, we extract the most meaningful subsequences from the entire series of time series data. After that, the extracted subsequences are passed to the clustering stage.

The STS clustering algorithm proposed in [26], makes use of MK [23] motif discovery algorithm for subsequences extraction stage. However, MK is not scalable for enumeration purposes, as mentioned in section 2.2. Therefore, to overcome this problem, we replaced the MK method used in [26] by MOEN motif discovery method [22] to enumerate all motifs for a range of lengths.

Algorithm 1 shows the pseudocode of the modified version of the MDL-based clustering approach. The algorithm takes the time series and minimum

and maximum length of motif as an input, and the final clusters of similar subsequences as an output. First, the algorithm enumerates all motifs given the minimum and maximum length of motif. Then, the enumerated motifs are then passed to the next stage “clustering stage”. The clustering stage is done in a bottom-up hierarchical fashion. It defines three operations, which are: create a cluster, add to cluster and merge clusters. At each level of the hierarchy, it tries all operations and chooses the best operator. The algorithm uses the minimum description length distance measure (MDL) to calculate the number of bits saved by each operator and then chooses the operator that maximizes the number of bits. The algorithm terminates in two ways: if the best possible operator cannot save more bits, or all data are used.

iii. Periodic Patterns Extraction

Given the subsequences clusters produced by the STS clustering component, not all similar subsequences in each cluster can be declared periodic. As we might have similar subsequences that occur at random places of the time series. Therefore, the proposed model declares periodicity in each cluster if at-least two members occur within the same place of the time series. For example, a periodicity is declared if the similar subsequences in a cluster occur between 1:00 pm and 2:00 pm. Meanwhile, if the similar subsequences occur at totally different places, let us say one occurs at 6:00 am, and other occur at 2:00 pm, then this case is not declared as periodic patterns. The extracted periodic patterns are then passed again to the “Hierarchal Time partitioning model” to extract periodic patterns from each pattern partition identified at the previous level of the hierarchy.

Algorithm 1: SubsequenceTimeSeriesClustering(TS,m,mx)

Input : $TS \leftarrow$ Time Series, m,$mx \leftarrow$ minimum and maximum length of time series motif

Output: Final cluster of subsequences

```
   /* 1.  Subsequences Extraction Stage                              */
1  MotifsperLength ← MOENEnumartion(TS, m, mx)

   /* 2.  Clustering Stage                                           */
2  clusters ← {}
3  bitsave ← ∞
4  while bitsave >0 do
5      bitsave ← −∞
6      costs ← {}
7      tmpClusters ← {}
       // Create new cluster
8      foreach motifLength ∈ MotifsperLength.keys() do
9          (A, B) ←MotifsperLength.getKey(motifLength).getNextMotif()
10         C′ ← CreateCluster(A, B)
11         Cost ← ComputeBitSave(C′, A, B)
12         costs.append(Cost)
13         tmpClusters.append(clusters ∪ {C′})
14     end
       // Add subsequence to an existing cluster
15     foreach C ∈ clusters do
16         A ← NearestNeighbor(ts, C)
17         C′ ← AddToCluster(C, A)
18         Cost ← ComputeBitSave(C′, C, A)
19         costs.append(Cost)
20         tmpClusters.append(clusters ∪ {C′} − {C})
21     end
       // Merge Two clusters
22     foreach C1 ∈ clusters do
23         foreach C2 ∈ clusters do
24             C′ ← MergeClusters(C1, C2)
25             Cost ← ComputeBitSave(C′, C1, C2)
26             costs.append(Cost)
27             tmpClusters.append(clusters ∪ {C′} − {C1} − {C2})
28         end
29     end
       // Pick the best operation for the current iteration and
          update clusters
30     bitsave ← max(costs)
31     indexMax ← costs.index(bitsave)
32     clutsers ←tmpClusters(indexMax)
33 end
```

After extracting periodic patterns partitions, each partition is passed to the *Workload Characterization Model* to generate the workload patterns per each partition.

4.2. Workload Pattern Prediction Model

Given the workload patterns generated by the *Workload Characterization Model*, this model is responsible for predicting the workload patterns at the next time interval. The following subsections present the steps taken to develop a single and multi-time-aware prediction models, respectively:

4.2.1. Time-Aware Single-Modeling

In this approach, we develop one prediction model for the entire time-space. Basically, the steps taken to develop a single time-aware prediction model, are as follows:

1. Features Engineering

Given the workload patterns generated by the *Workload Characterization Model*, the features engineering process is applied to extract different features from the workload patterns. The features could be categorized into two categories based on the modeling type:

a) **Univariate Modeling:** uses one feature to construct the feature matrix and target value, as shown in Figure 8. In this study, we consider the number of requests as a feature to construct the model.

b) **Multivariate Modeling:** incorporates different time features to construct the feature matrix. The time features considered in this study are: month, day of week, hour of day, minute of the day, is weekend, and is holiday. This method includes the extracted time features and number of requests at time (t) as the features matrix to predict the next number of requests, as shown in Figure 9.

Figure 8: Univaraite modeling

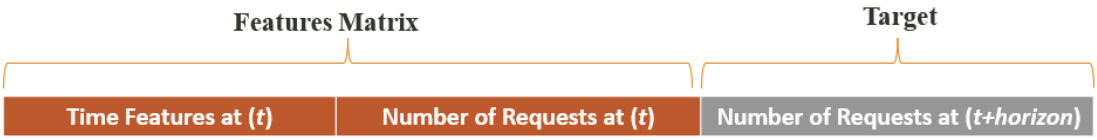

Figure 9: Multivaraiate modeling

2. Time Windowing

Time windowing is one of the transformation techniques used in time series prediction. Since the web applications workload data is a time series data, we consider the window size and time horizon to transform the workload data. Figure 8 illustrates the concept of window size, by selecting the last k number of requests from time (t) to (t+window) as the feature matrix. On the other hand, the time horizon defines which step ahead to be predicted. For example, if the horizon is 2, then the second step ahead will be predicted. Figure 8 and Figure 9 illustrates how time horizon concept is applied to the target value.

3. Prediction Models

We used two different methods to build the prediction models: NNLS Linear Regression and Long Short-Term Memory Neural Networks (LSTM).

a) Non-Negative Least Square Regression (NNLS):

The NNLS is one type of least square methods that are used to estimate the coefficients of the fitted linear regression model. A simple linear regression model equation is shown in the below equation:

$$Y = = \alpha + \beta X$$

The coefficients (β and α) are estimated by minimizing the sum of squares of the differences between the actual values and the predicted values. NNLS apply a constrained least square such that the estimated coefficients are not allowed to be less than 0.

b) Long Short-Term Memory Neural Networks (LSTM)

Recurrent Neural Network has received great attention for modeling time series data due to its effectiveness in remembering patterns for long durations of time. This is achieved by the time dependency property in RNNs. Figure 10 shows how the time dependency property is achieved through the cyclic connections between the units in RNNs. Each unit in RNN considers the previous predictions and information learned from them to predict the future ones. In fact, RNNs are only effective with short term dependencies. However, it turns out to be ineffective when the predictions need to remember larger range of dependencies. This is due to a well-known problem RNN suffers from, which is *Vanishing Gradient*.

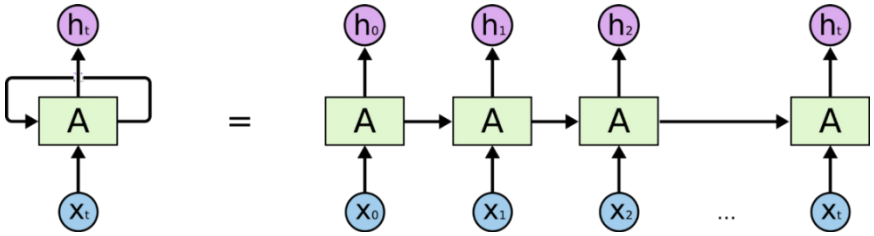

Figure 10: RNN architecture

Long Short-Term Memory (LSTM) is one variant of RNNs that was designed to solve the problem RNN suffers when dealing with long term dependencies. The *Vanishing Gradient* issue is resolved by including a memory cell that consists of three main gates: forget gate, input gate and, output gate. The LSTM updates the memory cell using these gates by controlling what information to forget or keep (forget gate), what information to update (input gate), and what information to output (output gate).

Due to the effectiveness of this method in modeling and predicting time series data, LSTM is used in this work to model and predict web applications workload. LSTM hyperparameters that can be tuned to improve the LSTM performance include the activation function, number of layers, number of neurons, number of epochs, batch size, to name a few. In this thesis, we have varied the activation function, the number of layers, and the number of neurons in each layer. LSTM method is sensitive to the scale of data. Therefore, we have used the Min-Max scaling method, which scales the data to a range of [0,1] or [-1,1]. The equation of Min-Max scaling method is shown below:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

4.2.2. Time-Aware Multi-Modeling

Given the workload patterns generated by the *Workload Characterization Model* per time partition, a prediction model is developed for each partition. In this approach, we have used a simple univariate regression-based model to predict the workload per time partition.

Chapter 5: Experimental Evaluation

This chapter presents the experiments that we conducted to evaluate the proposed solutions. We start by presenting the evaluation setup in section 5.1. We then present the experiments and discuss the results in section 5.2.

5.1. Experimental Setup

This section describes the datasets collection, training and testing data splits, and the evaluation measures used to evaluate the proposed approach.

5.1.1. Datasets Collection

To evaluate the proposed approach, we obtained the web server access logs from two real workload datasets generated by the Library Portal at Qatar University[1] and News Portal (NewsLink[2]) in Pakistan. Qatar University Library portal is a publicly available site used to access the library resources and services. The NewsLink portal is a news portal that collects the news from different resources. The workload access logs store information about the client requests and server response. This log entries include client IP address, resource requested URL, request timestamp, the status code of the server response, number of bytes transmitted by the server (e.g., document size), elapsed time of the requested file (e.g., response time), client user agent, and others. Figure 11 shows five records of the Library portal access logs collected by an Apache web server. The IP addresses of the clients are anonymized to ensure their privacy. In this work, we only consider three features namely, document size, response time and request timestamp for workload characterization and prediction.

The Library Portal access logs are collected from March 2018 to July 2018.

[1] http://mylibrary.qu.edu.qa
[2] http://newslink.pk

Meanwhile, the NewsLink access logs are collected across one day, 30 Sep 2017. Since the obtained News Link access logs are small, we emulate a periodic behavior of arrival rates over five days. This is done, by replicating the behavior of one day over five days. In addition, we added a normal random noise with a scale factor of 50 to the periodic behavior.

```
7154396679 - - [02/Apr/2018:05:24:11 +0300] "GET /apple-touch-icon.png HTTP/1.1" 301 - 173625
7154396679 - - [02/Apr/2018:05:24:11 +0300] "GET /favicon.ico HTTP/1.1" 301 - 180427
7154396679 - - [02/Apr/2018:05:24:12 +0300] "GET /en/favicon.ico HTTP/1.1" 404 1499 229153
9506378910 - - [02/Apr/2018:06:07:47 +0300] "GET /en/ HTTP/1.1" 200 28145 253439
5982669013 - - [02/Apr/2018:03:11:37 +0300] "GET /en/collection/special-collections HTTP/1.1" 200 19434 510217
```

Figure 11: Example of an anonymized apache access log.

5.1.2. Training and Testing Data Splits

We have used the multiple train and test approach that used specifically with time series data [4], to split our datasets. This approach trains different models to predict the test data split as shown in Figure 12. The main idea is that once the test dataset becomes available, the model is recalibrated using the previous train and test datasets. For the Library Portal dataset, we start by using three weeks of data for training and one-week data for testing. On the other hand, for News Link Portal dataset, we start with one-day for training and one-day data for testing

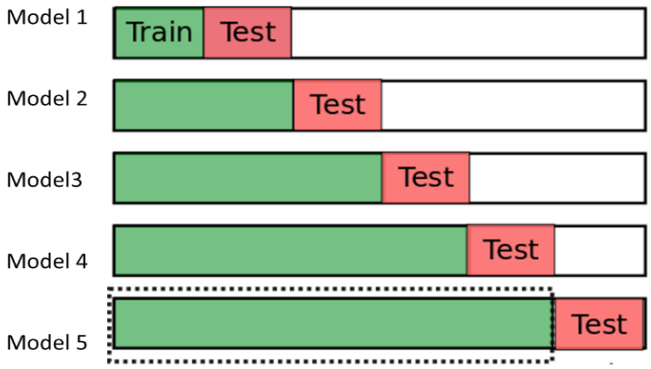

Figure 12: Multiple train and test approach

5.1.3. Evaluation Measures

In this work, the prediction error is calculated as the difference between the actual and the predicted workload pattern. Given the predicted workload pattern $p_t = [\,d'_1, d'_2, d'_3, \ldots, d'_k]$ and actual workload pattern $p_t = [d_1, d_2, d_3, \ldots\ldots, d_k]$ at time t, where k represents the number of partitions and d_i is the distribution over partition i, the prediction error is calculated as follows:

$$Prediction\ Error = \sum_{i=1}^{k} |(d'_i - d_i)|$$

5.2. Results

In this section, we present and compare the obtained results of the baseline system and proposed approaches. We start by presenting the workload characterization evaluation in section 5.2.1. We then evaluate the workload pattern prediction model in section 5.2.2.

5.2.1. Workload Characterization Evaluation

Before evaluating the prediction model, we first need to generate the workload patterns, which represents the URI distributions at a given sampling rate. This step is done using the workload characterization model proposed in [11] . For the proposed time-aware single-modeling approach, the workload patterns are generated over the entire time-space. On the other hand, with the multi-modeling proposed approach, the periodic time partitions are first extracted using the periodicity detection component. Then, these time partitions are passed to the workload characterization model to generate the workload patterns per each time partition. The upcoming sections present the results of generating the workload patterns for both proposed approaches.

5.2.1.1. *Time-Aware Single-Modeling*

Given the application URIs space of Library Portal and News Link datasets, the URIs are partitioned into different clusters based on two features: response time and document size. To identify the optimal number of clusters (k), we used the elbow method as an indicator of the appropriate number of clusters. *Figure 13* shows the residual error at different number of clusters which ranges between 2 and 26 for the library portal datasets, and ranges between 2 to 10 for the news link portal. It can be clearly seen that the residual error starts to flatten significantly at $k = 12$ and $k = 7$ for the library and NewsLink portal, respectively. Thus, from this experiment, we identified $k = 12$ and $k = 7$ as the optimal number of clusters for the library and news link portal, respectively.

Given the optimal number of URIs clusters, the workload patterns are then generated using the *Workload Characterization Model* [11]. The workload patterns are generated using two sampling rates: one-hour rate and 10-minutes rate.

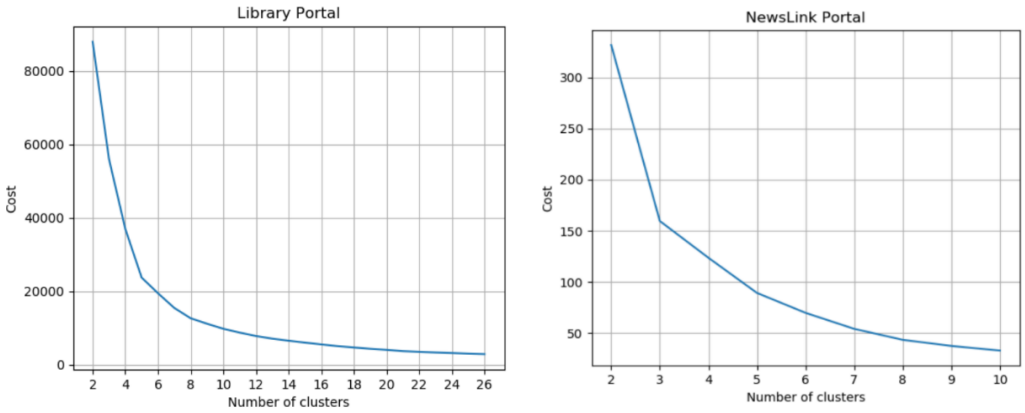

Figure 13: Residual error with increasing number of clusters over all datasets

5.2.1.2. *Time-Aware Multi-Modeling*

In this approach, the periodic patterns are first extracted by the periodicity detection component. Then, the workload patterns are generated per time partition.

Evaluation on Library Portal Dataset

We present how the proposed periodicity detection is used to extract periodic patterns from the library portal dataset. First, we start with the *Intensity-based Workload Partitioning Model*, followed by the *Hierarchical Time Partitioning Model*.

a) Intensity-based Workload Partitioning

As shown in Figure 14, the library portal has similar patterns over months but with different workload intensities. Therefore, we run the intensity-based workload partitioning model on this dataset with day sampling rate. Then, we extract four statistical features, S_{mean} , S_{std} , S_{min} , and S_{max} per month. The months are then clustered based on the statistical features shown in Table 3 using k-means clustering method. We run k-means across different number of clusters which ranges between 2

and 5. Results show that the optimal number of clusters is k=2. Table 4 shows that k-means splits the months into two clusters: March, April, and May in one cluster, and June and July in another cluster. In fact, these splits show the workload behavior at Qatar University, as the first three months represent a busy period at the university compared to June and July period.

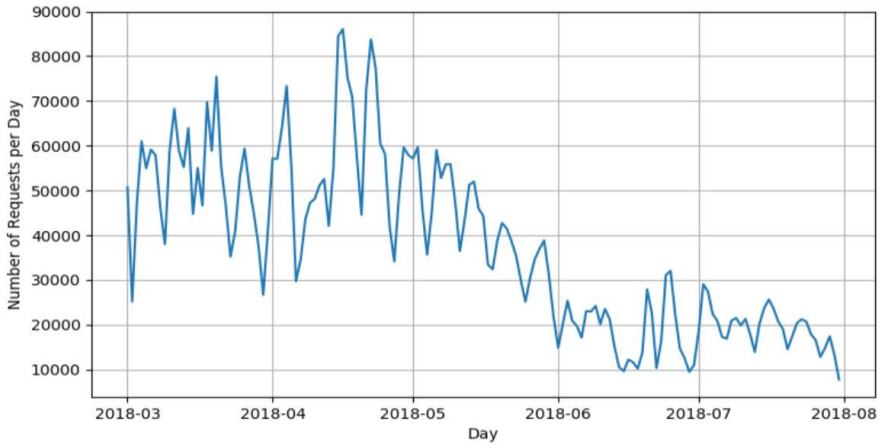

Figure 14: Number of request per day for the library portal

Table 3: Statistical Features Extracted per Month

Month	S_{mean}	S_{std}	S_{min}	S_{max}
March	51224.5	11782.8	25189	75465
April	57516.3	15178.7	29734	86084
May	41949.9	10109.2	21915	59714
June	18251.4	6514.6	9484	32079
July	19229.6	4416.5	7755	29066

Table 4: Intensity-Based Workload Partitioning Results of the Library Portal

Month	Cluster ID
March	1
April	1
May	1
June	2
July	2

b) Hierarchical Time Partitioning

Given the time partitions produced by the *Intensity-based Workload Partitioning model*, each partition is passed to the *Hierarchical Time Partitioning model*. For simplicity, we have shown below the levels of the hierarchy.

Level # 1: Hierarchical Time Partitioning

Given the partitions extracted by the *Intensity-based Workload Partitioning model*, we sample these partitions using *day* sampling rate. Then, the sampled data is passed to the STS to extract periodic patterns. *Figure 15* shows the periodic patterns extracted at this level. The highlighted boxes show the partitions that did not show-up as periodic patterns. Meanwhile, the others show the discovered periodic patterns.

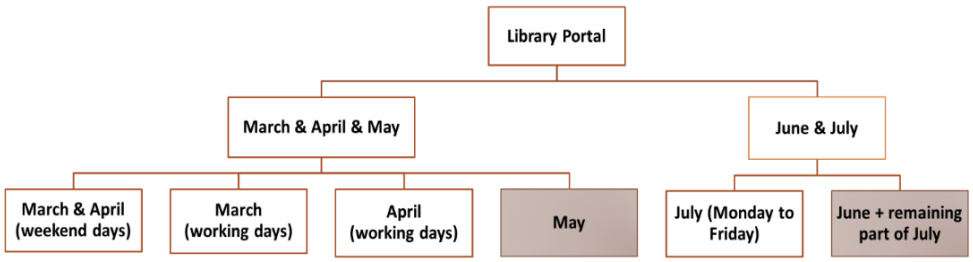

Figure 15: The 1st level periodic patterns extracted from the library portal

Level # 2.1: Hierarchical Time Partitioning

Given the periodic patterns extracted at level # 1, we sampled these partitions using a smaller one-hour sampling rate. Then, each sampled data is passed to the STS to extract periodic patterns. *Figure 16* shows the periodic patterns extracted at this level. It can be clearly seen, that the extracted periodic patterns mostly start at the early morning hours and ends before noon. In fact, this shows how the proposed periodicity detection component can extract patterns that reflect a realistic behavior that represents a low activity period in the morning hours which increases gradually in the later hours.

Level # 2.2: Hierarchical Time Partitioning

Differently than level 2.1, level 2.2 samples the periodic patterns extracted at level # 1 using a smaller 10-minutes sampling rate. This shows the periodic patterns which appear when the data is sampled using a smaller sampling rate. *Figure 17* shows the periodic patterns extracted at this level. Like level 2.1, most of the patterns extracted at this level happens to start at the early morning hours and ends before noon.

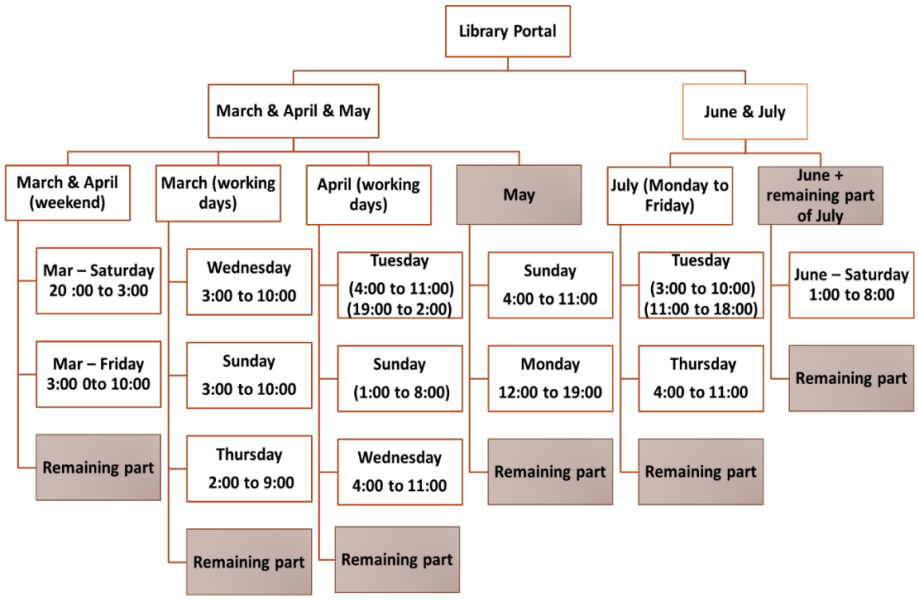

Figure 16: Level 2.1 periodic patterns extracted from the library portal dataset

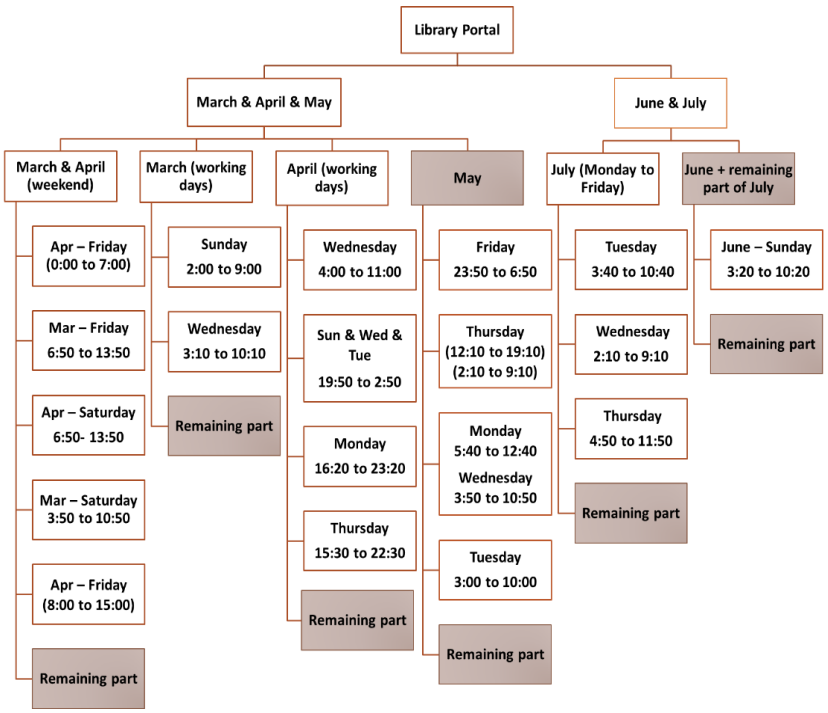

Figure 17: Level 2.2 periodic patterns extracted from the library portal dataset

Workload Characterization Model: After extracting the periodic partitions, the workload characterization model performed on each time partition to generate the workload patterns at one-hour and 10-minutes sampling rate. For the time partitions extracted with one-hour sampling rate, the experiments show that the optimal number of clusters to cluster the application URIs per time partition is k = 10. Meanwhile, For the time partitions extracted with 10-minutes sampling rate, the experiments show that the optimal number of clusters to cluster the application URIs per time partition is k = 8.

Evaluation on the NewsLink Portal Dataset

Similarly, we present how the proposed periodicity detection is used to extract periodic patterns from the NewsLink dataset. First, we start with the *Intensity-based Workload Partitioning Model*, followed by the *Hierarchical Time Partitioning Model*.

a) Intensity-based Workload Partitioning

As shown in *Figure 18*, the NewsLink portal has similar patterns over days without any differences in the workload intensities. Therefore, this step is skipped, and the original historical logs are passed immediately to the *Hierarchical Time Partitioning Model*.

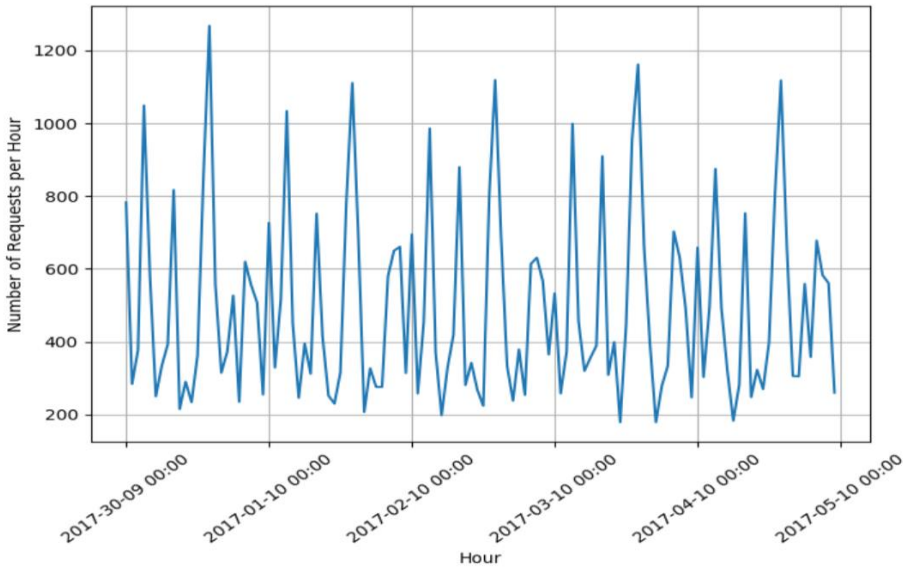

Figure 18: Number of requests per hour of the newsLink portal

b) Hierarchical Time Partitioning

Since the size of NewsLink portal is very small (only five days), we run the *Hierarchical Time Partitioning Model* only for one level with two sampling rates: one-hour rate and 10-minutes rate. The level with both sampling rates is shown below.

Level # 1.1: Hierarchical Time Partitioning

The historical logs of NewsLink are passed to the *Hierarchical Time Partitioning model* and sampled using one-hour sampling rate. Then, the sampled data is passed to the STS to extract periodic patterns. *Figure 19* shows the periodic patterns extracted at this level.

Figure 19: Level 1.1 periodic patterns extracted from the newsLink portal

Level # 1.2: Hierarchical Time Partitioning

Differently than level 1.1, level 1.2 sample the historical logs of NewsLink using a smaller sampling rate: 10-minutes rate. This just to show the periodic patterns that appear at a smaller sampling-rate, as shown in *Figure 20*.

Figure 20: Level 1.2 Periodic patterns extracted from the newsLink Portal

Workload Characterization Model: After extracting the time partitions, the workload characterization model performed on each time partition to generate the workload patterns at one-hour and 10-minutes sampling rate. For the time partitions extracted with one-hour sampling rate, the experiments show that the optimal number of clusters

to cluster the application URIs per time partition is k = 5. Meanwhile, For the time partition extracted with 10-minutes sampling rate, the experiments show that the optimal number of clusters to cluster the application URIs space per time partition is k = 5.

5.2.2. Workload Pattern Prediction Evaluation

This model is evaluated by predicting the workload patterns generated by the *Workload Characterization Model* at two sampling rates: one-hour and 10-minutes rates. The workload patterns at one-hour rate, are predicted at different time horizons which ranges between 1 to 3 with a step of size. While, the workload patterns at ten-minutes rate are predicted at 1,6, 18 horizons.

The prediction models are tuned with different parameters to achieve the best performance. For the univariate modeling, the window size is tuned using two ranges. With 10-minutes sampling rate, the window sizes have been chosen from range 5 to 20 with a step of size 5. Meanwhile, with one-hour sampling rate, the window sizes have been chosen from range 1 to 5 with a step size of 1. In addition to that, we have chosen four parameters to tune the LSTM model, which are: (1) Activations functions: Sigmoid function (Sigmoid), Hyperbolic tangent function (tanh), and Rectified linear unit function (Relu) (2) Number of hidden layers have been chosen from range 1 to 3 with a step of size 1 (3) Number of neurons in hidden layers have been chosen from range 10 to 20 with a step of size 5. The following sections present the experiments conducted to evaluate all proposed solutions: *Time-Aware Single Modeling* and *Time-Aware Multi-Modeling*

5.2.2.1. *Time-Aware Single Modeling*

We have developed three variations of time-aware single-modeling, namely: Multivariate Regression, Multivariate LSTM, Univariate LSTM. The features considered in both, multivariate and univariate models, are already described in Section 4.2.1. The models have been tuned using different parameters to achieve the minimum prediction error. Table 8 in the appendix shows the tuned parameters values that produce the minimum prediction errors.

We experimented the prediction models at different time horizons and sampling rates. *Table 5* shows the minimum prediction error achieved by the single modeling approaches. It can be noticed that the Multivariate Regression approach outperforms all other approaches across all time horizons, when the data is sampled using one-hour sampling rate. This improvement is consistent over both datasets. On the other hand, with 10-minutes sampling rate, none of the improvements are achieved consistently by one approach over all time horizons and datasets. Therefore, we cannot draw a clear conclusion for predictions with 10-minutes sampling rate. Meanwhile, we can conclude that the Multivariate Regression approach is the best among the time-aware single modeling approaches in handling the predictions with one-hour sampling rate. This indicates that the features extracted using the multivariate modeling are indicative features which help the prediction model to predict the workload patterns at one-hour rate.

Table 5: Workload Pattern Prediction Error for all Time-Aware Single-Modeling Approaches. The Best Score per Horizon is **Boldfaced**

			Time-Aware Single Modeling		
Datasets	Sampling rate	Horizons	M_ Regression	M_ LSTM	U_ LSTM
Library Portal	one-hour	1	**17.3**	19.1	18
		2	**19.3**	20.6	20.1
		3	**21**	22.3	21.8
	10-minutes	1	39.2	39.9	**35.8**
		6	40.5	41.3	**37.7**
		18	43.4	42.5	**41**
NewsLink Portal	one-hour	1	**26**	27.3	27.3
		2	**26.5**	27.5	27
		3	**26.7**	27.6	27
	10-minutes	1	**36.3**	42.2	38.8
		6	43.7	**43.3**	45
		18	44.3	**43.3**	45.9

*** M** stands for Multivariate and **U** stands for Univariate modeling

5.2.2.2. *Time-Aware Multi-Modeling*

This approach develops a prediction model per time partition which is extracted by the proposed periodicity detection component. As is the case in Time-Aware single modeling, the prediction models have been tuned using different parameters. We experimented the prediction models at different time horizons and sampling rates. Table 6 presents the minimum prediction errors achieved by thee multi-modeling approach.

Table 6: Workload Pattern Prediction Error of Time-Aware Multi-Modeling Approach.

Datasets	Sampling rate	Horizons	Time-Aware Multi-Modeling Data_Partitioning
Library Portal	one-hour	1	18.2
		2	21.3
		3	24.2
	10-minutes	1	31.5
		6	34.3
		18	39.8
NewsLink Portal	one-hour	1	29.2
		2	28.4
		3	30.1
	10-minutes	1	23.6
		6	33.6
		18	33.8

5.2.2.3. *Summary and Comparison of the Results of all Experiments.*

To validate the effectiveness of all proposed approaches, we compare the obtained results with the baseline systems. As explained earlier in Section 3.3.2. , w have developed two variations of the baseline system: the first is based on the moving average method for prediction, and the other is based on NNLS regression method.

Table 7 shows the best approach per time horizons and sampling rates. It can be noticed that both baseline systems are almost achieved similar performance. In addition, the

results show that Multivariate Regression approach outperforms all proposed approaches and baseline systems, when the data is sampled using one-hour sampling rate. On the other hand, the data partitioning proves its effectiveness when the data is sampled using 10-minutes sampling rate, over all proposed approaches and baseline systems. This indicates that the sampled workload at the 10-minutes rate, is not predictable by the single-modeling approach. In the meantime, partitioning data, make it more predictable and thus helps the prediction model to predict it.

Also, it can be clearly noticed that the longer the horizon, the higher the prediction error. This indicates that the last k observations of the workload are more correlated with the next one (k+1), compared to the further ones.

We used a two-tailed paired t-test, with a significance level $\alpha = 0.05$, to indicate statistically-significant improvements of the best approach over all other approaches. The results of the significance test indicate that the achieved improvements are statistically different on all datasets.

Table 7: Workload Pattern Prediction Error of Proposed Approaches and Baseline Systems. The Best Score per Horizon is **Boldfaced**

Dataset	Sampling rate	Horizons	Baseline systems		Time-Aware Single Modeling			Time-Aware-Multi-Modeling
			Moving Average	U_ Regression	M_ Regression	M_ LSTM	U_ LSTM	Data_Partitioning
Library Portal	one-hour	1	17.8	18.2	**17.3***	19.1	18	18.2
		2	20.9	21.7	**19.3***	20.6	20.1	21.3
		3	23.7	24.6	**21***	22.3	21.8	24.2
	10-minutes	1	35.7	35.6	39.2	39.9	35.8	**31.5***
		6	37.4	37.4	40.5	41.3	37.7	**34.3***
		18	41.8	42.4	43.4	42.5	41	**39.8***
NewsLink Portal	one-hour	1	29.3	28	**26***	27.3	27.3	29.2
		2	30.3	29.2	**26.5***	27.5	27	28.4
		3	30.5	31.5	**26.7***	27.6	27	30.1
	10-minutes	1	37.3	32.1	36.3	42.2	38.8	**23.6 ***
		6	50.2	45.3	43.7	43.3	45	**33.6***
		18	48.3	48	44.3	43.3	45.9	**33.8***

* Indicates significant improvement of the best approach over all other approaches using two-tailed paired t-test, with α=0.05 **M** stands for Multivariate and **U** stands for Univariate modeling

Chapter 6: Conclusion and Future Work

This thesis focuses on improving the workload characterization and prediction. We considered the request time in addition to the response time and document size, to develop a time-aware workload prediction model. We proposed two approaches to make it time-aware: (1) Time-Aware Single Modeling that develops one prediction model for the entire time-space, and (2) Time-Aware Multi-Modeling that develops a prediction model for each time-partition discovered by the periodicity detection competent.

We have conducted different experiments on two realistic workload datasets and presented the results. The experiments were conducted at two sampling rates: one-hour and 10-minutes rate. The experiments showed that the time-aware approaches are sensitive to the sampling rates. The results demonstrated that the multivariate regression model developed for the entire time-space, was the most effective approach in predicting the workload when sampled using one-hour rate. While, the multi-models approach is more effective for predicting the workload sampled using 10-minutes rate. Additionally, we leverage the LSTM Neural Networks for predicting the workload. Despite the success of LSTM in handling time series data, LSTM models failed to produce good results compared to other approaches. We believe that this caused by the size of training dataset that leads to poor generalization of predictions.

We contributed to the workload characterization and prediction research area in the following ways: (1) Propose time-aware workload characterization and prediction model (2) Study the sensitivity of the proposed time-aware approaches at different sampling rates (3) Conduct comprehensive experimental evaluation on realistic datasets and compare the results with the state-of-art methods.

6.1. Future Work

There are many research directions that can be considered to further improve this research. The important recommended future research directions are summarized below:

- In this thesis, we rely on one distance measure for time series clustering, which is the Minimum Description Length (MDL). We plan to try other distance measures, such as the Dynamic Time Warping (DTW) [25], due to its strength in dealing with temporal drifts by measuring one-to-one and one-many similarity distance. We believe that DTW might affect the outcomes of the periodicity detection component and thus the prediction model.
- We plan to improve the LSTM Neural Network developed in this thesis by tuning other hyperparameters such as the number of epochs, batch size, optimizer and others. In addition, we plan to train and test the model on larger datasets to discover its limitations and address them to improve the model.
- We used the simple grid search method to select the best combination of LSTM hyperparameters. We plan to try other methods, such as greedy and dynamic programming approaches. We believe that these methods will confine the search space and thus enable us to tune more hyperparameters.
- We plan to extend this work to predict the workload for multi-tiered web applications. This will be done by predicting the workload per each tier.

REFERENCES

[1] Abdul-Kader, H. and Abdul Salam, M. 2012. Evaluation of Differnial Evolution and Particle Swarm Optimization Algorithms at Training of Neural Network for Stock Prediction. *International Arab Journal of e-Technology*.

[2] Aghabozorgi, S., Seyed, A. and Wah, T.Y. 2015. Time-series clustering – A decade review. *Information Systems*. 53, (2015), 16–38.

[3] Ali-eldin, A., Rezaie, A., Mehta, A., Razroev, S., Sj, S., Tordsson, J. and Elmroth, E. 2014. How will your workload look like in 6 years ? Analyzing Wikimedia ' s workload. (2014).

[4] Bergmeir, C. and Benítez, J.M. 2012. On the use of cross-validation for time series predictor evaluation. *Information Sciences*. 191, (2012), 192–213.

[5] Calheiros, R.N., Masoumi, E., Ranjan, R. and Buyya, R. 2015. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications ' QoS. *IEEE Transactions on Cloud Computing*. 3, 4 (2015), 449–458.

[6] Calzarossa, M.C., Massari, L. and Tessera, D. 2016. Workload Characterization : A Survey Revisited. 48, 3 (2016), 1–43.

[7] Calzarossa, M.C. and Tessera, D. 2015. Modeling and predicting temporal patterns of web content changes. *Journal of Network and Computer Applications*. 56, (2015), 115–123.

[8] Cezar, J., Ricardo, E., Jose, M., Ehlers, R. and Reiff-marganiec, S. 2016. Combining time series prediction models using genetic algorithm to autoscaling Web applications hosted in the cloud infrastructure. (2016), 2383–2406.

[9] Gong, Z., Gu, X. and Wilkes, J. 2010. PRESS : PRedictive Elastic ReSource Scaling for cloud systems. *2010 International Conference on Network and Service Management*. (2010), 9–16.

[10] Huang, Q., Shuang, K., Xu, P., Li, J., Liu, X. and Su, S. 2014. Prediction-based Dynamic Resource Scheduling for Virtualized Cloud Systems. 9, 2 (2014), 375–383.

[11] Iqbal, W., Erradi, A. and Mahmood, A. 2018. Dynamic workload patterns prediction for proactive auto-scaling of web applications. *Journal of Network and Computer Applications*. 124, February (2018), 94–107.

[12] Islam, S., Keung, J., Lee, K. and Liu, A. 2012. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*. 28, 1 (2012), 155–162.

[13] Karaboga, N. and Cetinkaya, B. 2004. Performance Comparison of Genetic and Differential Evolution Algorithms for Digital. *Evolution*. (2004), 482–488.

[14] Keogh, E. Clustering of Time Series Subsequences is Meaningless : Implications for Previous and Future Research.

[15] Kumar, J. and Singh, A.K. 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*. 81, (2018), 41–52.

[16] Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B. and Chen, J. 2017. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications*. 80, June 2016 (2017), 35–44.

[17] Liu, W., Song, H., Liang, J.J., Qu, B. and Qin, A.K. 2014. Neural network based on self-adaptive differential evolution for ultra-short-term power load forecasting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 8590 LNBI, (2014), 403–412.

[18] Madicar, N. 2013. Parameter-Free Subsequences Time Series Clustering with

Various-width Clusters. (2013), 150–155.

[19] Madicar, N., Sivaraks, H. and Rodpongpun, S. 2014. An Enhanced Parameter-Free Subsequence Time Series Clustering for High-Variability-Width Data. (2014), 419–429.

[20] Meng, Y., Rao, R., Zhang, X. and Hong, P. 2016. CRUPA : A Container Resource Utilization Prediction Algorithm for Auto-Scaling Based on Time Series Analysis. *2016 International Conference on Progress in Informatics and Computing (PIC)*. (2016), 468–472.

[21] Miguel-alonso, T.L.J. and Lozano, J.A. 2014. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. (2014), 559–592.

[22] Mueen, A. Enumeration of Time Series Motifs of All Lengths.

[23] Mueen, A., Cash, S. and Westover, B. 2002. Exact Discovery of Time Series Motifs. (2002).

[24] Nikravesh, A.Y. and Ajila, S.A. 2015. Towards an Autonomic Auto-Scaling Prediction System for Cloud Resource Provisioning. (2015).

[25] Oates, T., Firoiu, L. and Cohen, P.R. Clustering Time Series with Hidden Markov Models and Dynamic Time Warping.

[26] Rakthanmanon, T., Keogh, E.J., Lonardi, S. and Evans, S. 2011. Time Series Epenthesis : Clustering Time Series Streams Requires Ignoring Some Data. Mdl (2011), 547–556.

[27] Rodpongpun, S., Niennattrakul, V. and Ratanamahatana, C.A. 2012. Knowledge-Based Systems Selective Subsequence Time Series clustering. *Knowledge-Based Systems*. 35, (2012), 361–368.

[28] Roy, N., Dubey, A. and Gokhale, A. 2011. Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting. *2011 IEEE 4th International*

Conference on Cloud Computing. (2011), 500–507.

[29] Torkamani, S. and Lohweg, V. 2017. Survey on time series motif discovery. 7, April (2017), 1–8.

[30] Tran, D., Tran, N., Nguyen, G. and Nguyen, B.M. 2017. ScienceDirect A Proactive Cloud Scaling Model Based on Fuzzy Time A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness Tran and SLA. *Procedia Computer Science*. 108, (2017), 365–374.

[31] Vlachos, M., Yu, P. and Castelli, V. 2005. On Periodicity Detection and Structural Periodic Similarity. *Proceedings of the 2005 SIAM International Conference on Data Mining*. (2005), 449–460.

[32] An Introductory Study on Time Series Modeling and Forecasting Ratnadip Adhikari R. K. Agrawal.

APPENDIX

Table 8:The Tuned Parameters Values that Produce the Minimum Prediction Error

Dataset	Sampling rate	Horizon	Time-Aware Single Modeling											
			M_Regression				M_LSTM				U_LSTM			
			window	#neurons	#layers	Act.	window	#neurons	#layers	Act.	window	#neurons	#layers	Act.
Library Portal	one-hour	1	NA				NA	20	2	tanh	2	20	1	tanh
		2						20	2	tanh	2	20	1	tanh
		3						20	3	tanh	2	20	1	tanh
	10-minutes	1	NA				NA	20	2	relu	15	20	1	tanh
		6						10	2	tanh	10	15	1	tanh
		18						20	2	tanh	5	20	1	relu
NewsLink Portal	one-hour	1	NA				NA	10	3	tanh	3	10	2	sig
		2						20	3	tanh	3	10	3	relu
		3						20	2	tanh	2	20	2	sig
	10-minutes	1	NA				NA	20	2	sig	5	20	1	relu
		6						15	3	sig	5	20	3	sig
		18						15	3	sig	5	20	2	sig

* NA indicates that the parameter is not applicable for the given approach.

*** M** stands for Multivariate and **U** stands for Univariate modeling