NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

PEDRO MIGUEL MATOS SILVA

BSc in Computer Science

BETTER MONITORIZATION AND PREDICTABILITY FOR COMPUTATION INFRASTRUCTURES THROUGH DATA FUSION

NOVA University Lisbon
02, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

BETTER MONITORIZATION AND PREDICTABILITY FOR COMPUTATION INFRASTRUCTURES THROUGH DATA FUSION

PEDRO MIGUEL MATOS SILVA

BSc in Computer Science

Adviser: João Leitão
Assistant Professor, NOVA University Lisbon

Co-adviser: Cláudia Soares
Assistant Professor, NOVA University Lisbon

NOVA University Lisbon
02, 2022

Abstract

Cloud computing provides a wide variety of services that allow its users to develop and maintain applications at a relatively low cost. The services offered by a cloud system also provide the ability to flexibly scale the resources of any application. Because of these benefits, cloud data centers are extremely popular. However, although generally considered secure, for circumstances involving the management of sensitive data some companies opt for a hybrid solution that involves storing some or all of their resources in a private data center, and occasionally, when system utilization reaches higher levels, the execution of some components is assigned to a public cloud data center, ensuring the integrity and security of that data without affecting service performance. Its however essential that such a migration services happens before the system is overloaded, as to avoid further negative impact to system performance, while avoiding services running in the cloud needlessly due to its increased costs. To mitigate this problem it is possible to apply methods that aim to predict periods of high system utilization. The prediction results can later be utilized to migrate resources before usage spikes occur, in a time window where it is possible to deal effectively with them.

In this work we intend to explore new strategies to monitor and extract metrics from different components of a private data center, and combine this information in order to characterize the level of utilization of the system.

It is also intended, based on the information extracted, to investigate solutions that predict future system utilization and to develop a scaling mechanism that allows certain resources to be sent to and removed from a the private data center to a public cloud provider. The results of each solution will be compared for their accuracy, execution speed, and the improvement in response time that can be achieved in the managed system. This thesis is conducted in collaboration with NOVO BANCO.

Keywords: Cloud Computing, Infrastructure Monitoring, Workload Prediction

Resumo

A computação em nuvem fornece uma grande variedade de serviços que permitem aos seus utilizadores desenvolver e manter aplicações a um custo relativamente baixo. Os serviços oferecidos por um sistema em nuvem proporcionam também a capacidade de escalar de forma flexível os recursos de qualquer aplicação. Devido a estes benefícios, os centros de dados em nuvem são extremamente populares. Contudo, embora considerados seguros, em circunstâncias que envolvem a gestão de dados sensíveis, algumas empresas optam por uma solução híbrida que envolve o armazenamento de alguns ou todos os seus recursos num centro de dados privado, e ocasionalmente, quando o sistema atinge níveis de utilização superiores, a execução de alguns componentes é atribuída a um centro de dados público em nuvem, garantindo a integridade desses dados e sem afectar o desempenho do serviço. É no entanto essencial que tais serviços de migração aconteçam antes do sistema estar sobrecarregado, de modo a minimizar o impacto negativo no desempenho do mesmo, ao mesmo tempo que se evita a utilização de serviços na nuvem desnecessariamente de modo a evitar custos. Para mitigar este problema, é possível aplicar métodos que visem prever períodos de elevada utilização do sistema. Os resultados da previsão podem ser utilizados mais tarde para migrar recursos antes da ocorrência de picos de utilização, num intervalo de tempo em que é possível lidar eficazmente com os mesmos.

Neste trabalho pretendemos explorar novas estratégias para extrair métricas de diferentes componentes de um centro de dados privado, e combinar esta informação de modo a caracterizar o nível de utilização do sistema. Pretende-se também, com base na informação extraída, investigar soluções que prevejam a utilização futura do sistema e desenvolver uma ferramenta de dimensionamento que permita o envio e remoção de certos recursos de um centro de dados privado para um serviço em nuvem. Os resultados de cada solução serão comparados pela sua precisão, velocidade de execução, e melhoria no tempo de resposta que pode ser alcançado no sistema gerido. Esta tese é conduzida em colaboração com NOVO BANCO.

Palavras-chave: Computação em nuvem, Monitorização de Infraestruturas, Previsão de utilização

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context

Cloud computing [43, 58] provides an enormous share of digital services that provide more flexible and fast services for users at a relatively cheap price. It is one of the major technologies of computation infrastructures because of how quicker and easier it is to develop and maintain applications, while reducing the initial cost of hardware. Adding to all of this, Cloud Computing provides the potential for high scalability and elasticity which is relevant to this case, since when the load increases services in clouds can allocate more resources to handle more requests. This is called auto scaling and it allows cloud infrastructures to adapt to the application needs while also ensuring reliability and quality of service.

Because of this, cloud data centers are one of the most popular ways to host or serve software or application data, however, even though cloud services are widely considered to be safe it is still not advisable for storing critical or sensitive data. Companies that deal with such data (for instance **banking companies**), have the option of storing all the sensitive information in private data centers, minimizing the risk of exposing such sensitive information to the cloud operator or third parties.

However private data centers also have some relevant weak points when comparing to cloud services, they are typically more costly, they do not scale nearly as well and when an update needs to be done, some private data centers might need to interrupt or delay some processes for certain period [8].

To ensure security of critical data, and to prevent leakage of such data, while also maintaining performance and reliability, some services use an hybrid solution, where applications have a part of their data (or all of it) stored in private data centers, and occasionally, when demand for some resources increases, they delegate some internal components to a public cloud data center in such a way that sensitive information exposition is minimized.

1.2 Motivation

The importance of monitoring infrastructures comes into play when there is a need to predict an increase of user requests or for example predicting when a machine is going to fail (although often machines fail unexpectedly). Cloud services internally monitor their machines and, using their own prediction algorithms, anticipate these spikes in usage as well as inherent machine failures and start allocating new resources so that the availability and efficiency of the whole application/service is not compromised. Of course, in order to control the cost increase of allocating new machines its possible to employ a maximum threshold of machines that can run at the same time.

Infrastructure monitoring and load prediction are especially important for the previously mentioned hybrid approach. To handle usage peaks in this situation, it is possible to, as previously mentioned, move some of the application components usually executed on the private data centers to the public cloud. However, this dynamic migration of components takes some time, hence the importance of forecasting this spikes in workload in a useful time window that allows the resources to be moved to minimize the negative effects in the users experience. Naturally, such mechanisms can also be relevant to decommission public cloud resources when they are no longer necessary to minimize costs.

1.3 Proposal

In this thesis, solutions to better monitor computation infrastructures and predict spikes in usage will be explored. Research will be conducted in finding ways to select, collect and combine different metric data from different sources in order to first characterize the workload of a system. Then, various prediction approaches will be examined and tested. To automatically deal with periods of higher load will also be looked upon. Finally, an in depth evaluation of the applied prediction algorithms will be carried on, testing their accuracy and if possible, their impact in quality of service for the whole system.

This research will be focused on applications that operate on the previously stated hybrid approach, meaning they mainly function using resources from a private data center while occasionally utilizing services from a public cloud provider.

1.4 Expected Contributions

The main contributions that are expected from this thesis work are the following:

- The implementation of mechanisms to collect metrics from different sources of the data center infrastructures as well as novel methods to combine those metrics to infer additional information.

- The investigation and analysis of different prediction methods for forecasting workload of the whole system.
- The evaluations of the solutions proposed, by experimentally comparing the accuracy results of the different algorithms and, at a later stage, if possible compare the quality of service improvements offered by each method.

1.5 Research Context

This dissertation is done under the context of a research project conducted in collaboration with NOVO BANCO DSI.

NOVO BANCO [42] is a banking company established in 2014 with 358 active branches, 20 Corporate Centers, 4582 employed professionals, more then 1.4 million customers and a digital banking system.

NOVO BANCO online banking service [36] is free, safe and available. The research that will be conducted also has the objective of matching these standards by exploring new approaches to deal with load peaks, while guaranteeing the safety of the data and preventing its misuse by third parties.

1.6 Document Structure

The rest of this document is organized as follows:

Chapter 2 introduces some of the research regarding ways to monitor infrastructures and represent the workload of a system. It also presents the state of the art of the currently used workload prediction schemes, explaining briefly the main concepts of some of the most popular approaches.

Chapter 3 presents some important software and protocols that are currently used in managing, securing, and monitoring data centers which will be relevant for the work to be developed on this thesis.

Chapter 4 provides details about the research plan and describes each step of future work.

2

Related work

In this chapter we firstly discuss in Section 2.1 the concept of autonomic computing is introduced along with its relevance for resource computing then, in Section 2.2 we present different ways to monitor infrastructures and acquire metrics. Furthermore, in Section 2.3, we provide a brief characterization of workload and all its representations (the metrics utilized, ways to fuse them). Finally, in Section 2.4, we survey some of the current relevant research regarding the field of workload forecasting and regarding the comparison between the different forecasting methods .

2.1 Autonomic Computing

As the development of software progresses, more complex systems are built, and the need for intercommunication between these systems increases. Making the management of an entire system more and more complicated. Systems need to be capable of managing themselves in an automatic way, reducing the complexity of manually managing software continuously and allowing faster responses[22].

Autonomic computing aims to achieve the previous goal. An autonomic system must be able to manage itself by adjusting to different varying factors, by predicting what future need might be and by searching ways, that take into account monitoring metrics and act upon that information in the most optimal way possible.

Achieving better monitoring of infrastructures, involves of course the design of an autonomic system, since manually monitoring all the metrics would be more difficult and costly. Additionally, the system needs to be able to automatically forecast future conditions based on the metrics observed and ensure that there is enough time for eventual adjustments.

To be able to call a system autonomic, that system must have some self management capabilities which can include:

Self-Configuration, meaning that the system is able to change itself automatically in a way that will adjust to new requirements

Self-Healing, meaning that the system has the ability to find problems within itself and correct those problems by reconfiguring itself.

Self-Optimization, which represents the ability of the system to look for the optimal solution to a demand automatically, by having feedback mechanisms within itself that make it possible to take action.

Self-Protection, which is the capability of the system to recognize attacks and security breaks and protect itself from them.

The concept of Autonomic computing has been an objective for multiple studies regarding workload prediction and auto scaling of resources in cloud systems.

In [21], autonomic choosing of metrics that represent the workload behavior is explored, and, by the choice of those metrics, autonomic computing is also applied in the choosing of a forecasting method that best fits the workload behavior represented by those metrics. Similarly, in [6] a way to automatize the representation of the workload trend (based of the resource measurements taken) of a web-based system is explored, as well as the automatic prediction (in real time) of future workload.

In [48], the concept of autonomic computing is applied in the automatic monitoring of cloud system metrics, automatic workload prediction based on those metrics and an automatic self-configuring mechanism that decides, based on the prediction computed, if its necessary to scale up the number of machines or if the system should scale down its computing power in order to save money.

To map the concepts of Autonomic computing onto "real" systems and to model them into autonomic managing systems there is a commonly used blueprint, called the **MAPE-K management control loop**[24]. Each letter in MAPE stands for a different function that represents a step in the loop. Each step will now be shortly explained in the context of this study:

In the first **Monitoring** (**M**) step, resource metrics are collected from the different devices. These metrics are then aggregated in a way that represents the behavior of the system from recent events. In this study, this function will correspond to the topics Metrics and their Acquisition and Workload Characterization and Data Fusion.

In the **Analysis** (**A**) step, the previous view of the system offered by the representation of the aggregated metrics is observed and analyzed. This function has the task of predicting future behavior patterns by employing prediction techniques on the representative aggregation of the metrics. This function will correspond to the topic of Workload Prediction.

The next function is the **Plan** (**P**) function. In this operation, a plan is made to choose a procedure to act upon the prediction made. In this case it could be deciding if it is necessary to delegate some of the resources to a public cloud and which of those resources are worth extracting.

The last step is the **Execute** (**E**) function. In this operation all the changes necessary that were scheduled in the plan operation are performed.

To follow this blueprint, not every two steps could for example alert a manual manager that could then devise a plan of execution and perform it. To allow this approach the time gap between the acquired metrics and the prediction generated would have to be larger so that the manual manager could devise a new execution plan and execute it.

The **K** stands for **Knowledge Source**, this source essentially saves all the information and policies obtained by the autonomic manager and the rest of the components.

The main goal of this study should not be to make every step of infrastructure management completely automatic but rather to reduce human intervention significantly on each of the previous described steps. Since the **Plan** and **Execute** steps will mainly depend on the details of the system itself, the focus of the next sections will be mainly on the first two steps.

The first monitoring step of the control loop involves the acquisition of metrics and the characterization and representation of the workload by those metrics. The Analysis step will be represented by the topic of workload prediction.

2.2 Metrics and their Acquisition

The metrics taken from a Data Center are all the raw measurements that are possible to pull from all its allocated physical machines and/or virtual machines. These resource measures are useful for knowing, in real time, the performance, behavior and health of a system. There are numerous metrics that can be relevant to estimate the current utilization of a system, being the current CPU usage, the amount of memory being used, the number of requests of a particular service, disk throughput, number of raised exceptions and many others. Some of these metrics can be classified as lower level (hardware level) while others as higher level (service/application level). It is possible (using the technologies that will be described) to get these raw measurements from different origins, they can be pulled from the load balancers, from the firewalls and obviously from the different physical machines.

An important distinction to make is related with how metrics are obtained, they can be continuously pulled by the monitoring server from the various targets (using a specific protocol or API) or an agent can be installed at the endpoints of the devices that will collect metrics from those devices and then send all the information back to the main server.[26] This two types of monitoring are referred as agentless monitoring and agent-based monitoring respectively.

Agentless monitoring is typically easier to deploy, maintain and extend while being lightweight and not as intrusive as agent-based monitoring, however it provides more restricted (shallow) metric data analysis because it is limited to what metrics the protocol or device that is being used shows by default. This monitoring technique can also pose a security problem because it generally requires global access of the entire network to pull metrics.

Agent-based monitoring on the other hand is typically a bit more complex in terms of deployment and management but provides deeper data metric analysis as well as increased security (since the agents to not require global access to the network).

The choice between this two types of monitoring is dependant on what level of intrusiveness is the system going to allow as well as the level of monitoring depth that is necessary to accurately represent its workload, meaning if all the information needed for analysis of the resources is in the device protocols, then the installation of an agent might not be necessary.

What metrics should be acquired and the ways to acquire them will clearly depend on factors such as the complexity of the infrastructure and the level of security and visibility that is wanted for the monitoring portion of the system

2.3 Workload Characterization and Data Fusion

The workload of a Data Center is conceptually the "work being done" for a time duration considering all the different machines currently employed. It can be considered and interpreted in many different ways, it can be represented as an individual resource measurement value (e.g. user requests or CPU utilization) and as a multidimensional representation of multiple metrics. Although there is no agreed classification method, workloads can be differentiated by [28, 30]:

- Their resource demands, the workload can be I/O intensive, compute intensive and bandwidth intensive. These demands may change randomly and continuously in the same workload and define the application or machines in which the workload was taken from.

- Their Architectural structure, multi-task applications can have different models of architecture such as a parallel model, pipeline model or a combination of both.

- Their nonfunctional requirements such as the behavior patterns that a specific machine shows when dealing with any request.

- Their processing models, which can be online/interactive or offline/batch, this is a broader distinction since the processing model will impact the resource demands and behavior of a the workload, for instance, an interactive workload might have shorter tasks, while the batch ones consist longer tasks that tend to be more resource intensive.

The first crucial task is to choose which and how many metrics will help to characterize the workload, the number of the metrics chosen is important since because many measurements might compromise the efficiency of the whole monitoring process, while choosing less measurements might provide insufficient information. Some metrics might

not provide any benefit in representing the workload and in the prediction of future conditions, and therefore should not be considered.

The choice of these metrics can be done manually based on prior knowledge of the system (for example, the effects that an increased number requests has on the system), or as [45] suggests, by utilizing feature filtering techniques such as PCA (Principal Component Analysis) for an automatic way to select the metrics that have the most impact in the workload prediction.

The process of aggregating different monitoring information and combining it into a single improved coherent representation of the workload is often referred as **Data Fusion**. The term **Data Fusion** is widely employed in diverse fields of research (e.g. sensor networks and database warehousing) and for that reason its difficult to define it and provide a strict classification of each of its techniques[14].

Making an aggregation of the resource measurements from the different systems (such as Firewalls, Load Balancers and Physical Machines) can increase efficiency in analysing and predicting workload, and can also filter out outliers and uncorrelated measurements that arise from the different devices within a system[44].

There are some challenges caused by combining information from different sources. One first problem that may arise, especially in predictions that need to be done in a shorter time-frame, is that when systems have a higher variance in short periods of time, it is not viable to just represent workload as a conjunction of raw measures[6]. These measures might not be linearly correlated and only provide a strange view of the system that is instantaneous which does not directly help with the prediction of future spikes, so in these cases it is of more interest to work with a representation of workload behavior rather than a directly using raw metrics. Also its important to note that as systems become more complex, the number of metrics that are important to take into account tend to increase, which means that algorithms that will later predict future load based on this metrics will also have higher complexity.

In order to solve this problem and to get a representative view of the load of each resource measurement, its suggested in [6], to utilize functions called **load trackers**. These functions will make a representation of the load by filtering out noises that arise from a sequence of uncorrelated resource measures so that this representation can be analyzed later (by load predictors). Since they allow easier analysis, it will also be possible to use predictors that are less complicated and therefore make the whole prediction more time efficient.

In the previously mentioned study, a distinction is made between two types of load trackers, linear load trackers and non-linear load trackers. In linear load trackers **SMA** (Simple Mean Average) or **EMA** (Exponential Moving Average) can be utilized.

SMA is the unweighted mean of all the taken resource measures evaluated at a certain time. They tend to introduce a certain delay when the sample size of resource measures used increases, to put a threshold to this delay EMA models can be employed.

EMA is the weighted mean of all the taken resources where these weights decrease exponentially using a smoothing factor. This makes it so that the last resource measures will have a higher impact on the calculation when comparing to the first measures.

When the sample size of resource measures increases by a considerable margin, the limitations of the linear load trackers might surface, and therefore it might be worth to contemplate the option of using non-linear load trackers. The reason being that the bigger the resource metric set, the bigger the delay caused by the both mentioned linear load trackers.

As an example of a non-linear tracker there is the **Cubic Spline** function. The **Cubic Spline** is a cubic function that interpolates a set of data points[53]. For the resource workload representation to fit the data with increased accuracy, it's possible to increase its polynomial degree. However, higher-order curves tend to be more complex to process in real time and can cause overfitting, by introducing strange outlier wiggles as the data changes. Cubic interpolants are a great tool to avoid these oscillations and get a smoother end function. Although cubic splice is heavier to compute than the previous linear load tracker models, it is independent of the resource metrics set, and it is more reactive to changes in the workload.

The accuracy of the workload representation of each method will of course depend on the system in which they are being employed. It should also be noted that accuracy alone shouldn't be the only factor to consider, because as previously mentioned it is necessary to consider the delay introduced by each method as well has the heaviness in processing the algorithms (e.g. autoregressive models could necessitate frequent updating of their parameters, even though they could provide a more accurate load representation [6]).

After making a representation of the load trend in each resource measurement, the main goal is to make a global representation of the entire system workload based on the trend of each resource metric. The prediction algorithm that is employed (which as later explained will depend on the infrastructure characteristics) will mainly determine if this global combination is necessary and what techniques should be used to do so.

2.4 Workload Prediction

A Prediction is essentially the output of an algorithm that as been trained on a certain dataset, and then can be applied to new data to forecast the probability of certain outcomes or to classify that new data[38]. In this work we are mostly interested in the ability to predict how the workload will evolve in the (near) future.

The concept of prediction is widely employed in multiple areas of computer science, and it is associated with an automatic approximation of a future reality. It can be used to classify new data by different features, or to get a fitting model that represents the real data in the best way possible.

As previously mentioned, in the context of making sure a good quality of service for customers is provided, a data center should be flexible in the management of its resources.

Hence the importance of prediction algorithms that can, based on previously monitored data, determine if and which new changes need to be made to support future probable events.
In order to forecast future workload, there are several methods, involving very distinct research fields. It is not feasible to perfectly compare the different prediction methods, but it is possible to take into account some of the pros and cons of each one in different datasets (with enough detailed research on the approach).

In the next topics firstly some of the different schemes employed in the area of workload forecasting will be presented. Then some of the general topics about a prediction algorithm will be briefly explained. Finally some ways to compare different workload prediction solutions will be disclosed.

2.4.1 Workload Prediction Schemas

It's important that the prediction model adopted is proactive, meaning it must predict future workload fast enough so there is sufficient time to allocate new resources (or to put these resources in a cloud environment). For this purpose, it should also not be too complex.

It must adapt to changes in behavior of the workload and consider its historical data in order to estimate future demand more accurately. There are various schemes that can be employed to forecast future workload, even so, it's important to note that flash workload produced artificially, for example a DDOS attack, will be very hard to predict, since this events are probably not derived from any historical data.

The schemes that are more statistical tend to perform better on more stable, low variance datasets because they employ predictions linearly, while machine learning and deep learning approaches tend to be more adaptable.

There is a wide variety of methods that can be used to forecast future workload, represented in the next figure:

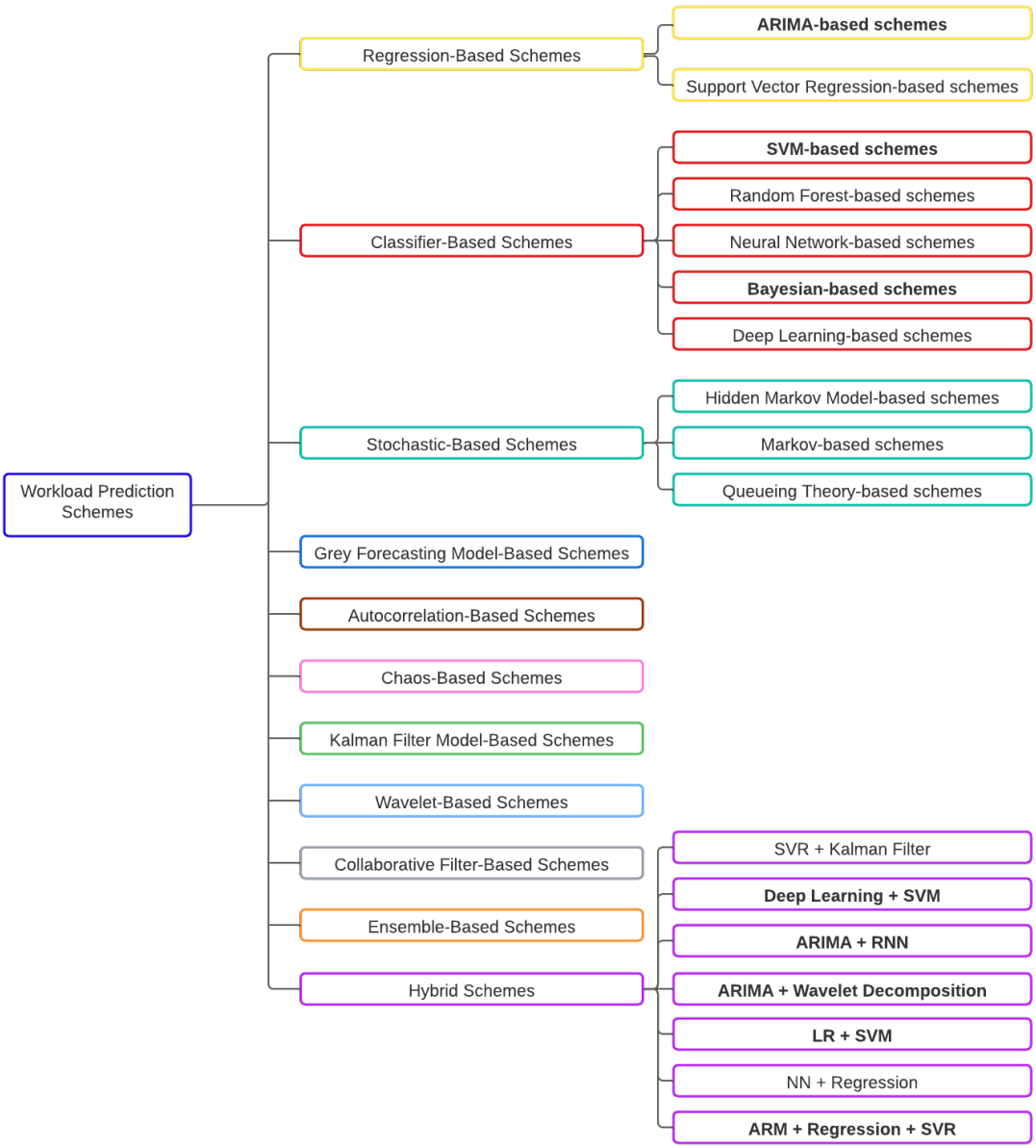

Figure 2.1: The different Workload Prediction Schemes, adapted from [30]

Now considering the different fields of study of workload prediction, a brief description of some of the most popular prediction schemes will be given:

ARIMA

Auto Regression Integrated Moving Average (ARIMA) is a statistical analysis model that utilizes data from a time series to predict future points and data trends. [7] [57]

A standard ARIMA model has 3 parameters [18], the first one is the number of autoregressive terms, the second one is the number differences employed (the differentiation) and the third is the number of moving average components. The model can be further explained by outlining each process:

The **Auto Regression** component (**AR**) refers to a model with a changing variable that regresses on lagged values, using this to predict the future values.

$$y_t = \alpha_1 y_{t-1} + \epsilon_t. \tag{2.1}$$

y_t is the representation of the preceding values of the problem where the model is being used.

The left side of the addition corresponds to a product between a self-regressive coeficient (α_1) and the previous observations, the right side (ϵ) corresponds to a random component.

The **Integrated** component (**I**) is basically the observation of the difference between raw data measurements and the previous values allowing the time series to become stationary.

$$y_t = y_{t-1} + \epsilon_t. \tag{2.2}$$

It takes into account the accumulated effect of several processes, because they might affect the behavior of the time series. To allow the time-series to become stationary the model undermines the short-time fluctuations of the system, by, at the extreme (with a differentiation of order 1), assuming that the difference between to consecutive values is constant.

The **Moving Average** component (**MA**) is the dependency between an observed value and the residual error (by applying the same model to previous observations). The number of previous time periods specified in the current value is designated in the number of moving average components.

$$y_t = \epsilon_t - \theta_1 \epsilon_{t-1} \tag{2.3}$$

To extend the practical usage of this model the Box-Jenkins [10] method was developed. The Box-Jenkins precept is composed of three steps, model identification, the assessments of parameters and the examination step.

For the model identification step some auto-correlation methods such as the ACF function and the PACF function can be applied to identify the order of the model. Besides that, the model should be applied on a stationary time-series meaning some data

transformation will need to be applied to reduce the amount of variance in the data before using the model.

The parameters are then estimated based on the previous step in a way that the error is minimized.

In the final examination or diagnosis step, the residual errors are measured in order to test the adequacy of the model employed. Conceptually this is done until an adequate model is obtained.

As previously explained **ARIMA** should not be applied in raw data if the dataset is unstable, there should be first a data transformation to make a more stable representation of the load (as previously stated in the load trackers section). This can be computationally expensive in real time high demand computing environments [6]. Adding to that, it can be difficult to constantly estimate the parameters of the model since they require a large amount of observations [18]. The application of this model tends to perform better on more stable datasets [20]

SVM

Support Vector Machines (SVM) encompass classification and regression techniques of supervised learning [2]. It is applied in a multiple variables feature space and depends heavily on the training data.

They are suitable in large scale systems because of their effectiveness with dealing with high dimensional spaces, meaning if a large amount of metric features is required, the model could still function. However as the number of different features is increased the sample window has to increase as well, otherwise the model will have the tendency to over-fit.

SVM based schemes are employed mostly in this multi attribute cloud environments, [35, 3] are examples of support vector regression models employed in cloud data environments in order to estimate future workload.

Bayesian Models

Bayesian Classifiers are models based on probabilities employed for several prediction problems. [37, 34]

The classification of a workload is done by taking into account the previously observed probability distribution of a set of metrics. Bayesian models as the name suggests are derived from a mathematical formula called the Bayes' Theorem:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} \tag{2.4}$$

Equation 2.4 denotes the probability of A given that B (the evidence parameter) is true, is equal to the product of the prior probability of A (P(A)) and the posterior probability of B (P(B|A)).

The most basic type of Bayesian Classifiers, is the Naive Bayes Classifier where the assumption is made that all the input resource measurements are have conditional independence between each other, this might not be applicable to some combinations of metrics since resource measurements tend to have some sort of correlation between each other.

[16, 47], are examples where the Bayesian model is used to predict future workload in cloud systems. It is stated generally that the utilization of this model involves these steps:

- Choosing a set of target states and assuring that the metrics chosen are independent.
- Calculating the probability distribution of the previously taken metrics for the chosen target states.
- Computing the joint probability for each state.
- Predicting the future state probability for the load.
- Evaluating the model and its accuracy.

In [47], it is declared that using the proposed Bayesian Model, approximately 75–80% of the data center servers could have their workload predicted with accuracy percentages greater than 80%.

There are some general limitations of Bayesian methods [46]. There is no objective way to choose the prior metrics, this has to be based on prior knowledge of the system, there needs to be an examination on what type of prior is the the best for a prediction that is not misleading. Bayesian Analysis can have also high computational cost, especially when considering multiple metrics from multiple sources.

LSTM

Long Short-Term Memory (LSTM) is a type of neural network model that utilizes short-term memory processes in order to build long-term memory [20]. An LSTM network [61] extends the concept of Recurrent Neural Networks (RNN). RNNs are deep learning algorithms frequently used for time based data [54]. They function (in a summarized way) by taking information from previous inputs to influence the output and current input of the network.

The problem with Recurrent Neural Networks is that their memory is short-termed, so a LSTM can be employed to solve this issue.

LSTM produces a result based on 3 factors, the cell state (the long-term memory of the network), the previous hidden state (meaning the result of the previous point in time and the information given at the current point in time.

An LSTM is composed of 3 parts [17], a Forget gate, an Input Gate and an Output Gate. The Forget Gate decides if a certain information should be kept or forgotten, the

input gate calculates for each new information how important that information is for the system and the output gate takes updated cell state ,the previous hidden state and the input data, and decides a new hidden state.

LSTM models [20] are a fairly complex field of deep learning and are good for predictions that can have delays of arbitrary duration in important events of the time series.

There are other multiple deep learning approaches, however, based on [50], in the two real-world traces chosen by this study (Google Cluster Data and a traditional Linux load trace) the LSTM model provided better results the other traditional models.

[27, 51, 62] several studies that utilize this model in order to forecast future workload in large systems.

One issue of LSTM models is their performance issues when executing on large-scale computing systems, in [51] a parallel improved LSTM is proposed to improve the prediction speed in those environments.

2.4.2 Prediction Gap

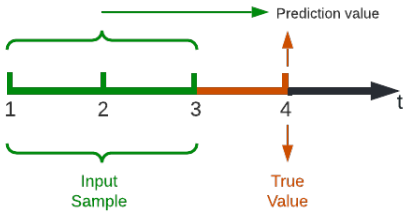

Figure 2.2: Illustration of 0-gap Prediction, adapted from [20]

One concept important to understand is the notion of a gap between the input values and the predicted values, meaning the number of points or the period of time that is between the input sample and the prediction window[20]. In 0-gap prediction there is no time gap between the input training values and the predicted values.

Using 0-gap prediction there is no time to allocate or migrate new resources. For this purpose, it is necessary to adopt a m-gap methodology, where m represents the time interval between the training set and the predicted output values.

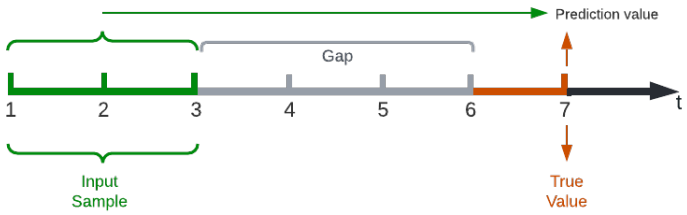

Figure 2.3: Illustration of m-gap prediction, adapted from [20]

This time gap will depend on how much time does it take for the system to respond to a sudden increase of workload. A trade-off to consider is that the bigger the window size the less accurate the model will be, however the prediction will leave more time for the system to react.

2.4.3 Input Sample Size and Predicted Window Size

Figure 2.4: Representation of the input sample and prediction

Another important topic to consider is the size of the input sample that is used. For example, in the identification of patterns on the dataset if the intention is to determine which hours of the day have a higher number of requests, an input window of one to several days might be useful. However, if the objective is to predict sudden increases in server load, then smaller input samples might be better.

Other factors for the choosing of a larger or smaller sample include the efficiency of the prediction algorithm and how heavy it is to constantly compute workload over a larger period of time.

Besides the sample input size there needs to be some pondering to determine the window size produced as output from the prediction algorithm. It represents the number of data points that are displayed in the output of the algorithm or the time interval of such prediction.

Evidently, multiple factors come into play when choosing the "optimal" prediction window size mainly the portion of the future load should be large enough to represent a symptom in the data center performance that needs fixing, the performance of the employed prediction algorithm, and the size of the sample input utilized.

2.4.4 Evaluation of Prediction Models

To evaluate prediction models [28, 11] applied on the same computation load, there are multiple measures to consider when comparing between true load and predicted load:

- **Mean Absolute Error** (**MAE**): The average between the values computed in the prediction and the actual values[23].
- **Mean Square Error** (**MSE**): The average of the squared distances between the prediction values and the actual values[31].
- **Root Mean Square Error** (**RMSE**): The squared root of the mean square error.
- **Mean absolute percentage error** (**MAPE**): The average of the percentage errors between the prediction values and the actual values.

The mere calculation of these metrics to evaluate the performance of a prediction scheme might not suffice, there is also other factors to consider, such as the size of the input window that is being utilized and the gap that is being considered between the input load points and the predicted points[20]. If a larger sample window is being considered, that means more input points of load representation are considered by the predictor and therefore more accurate results will be extracted, however, as the input sample becomes wider, the delay introduced by those predictors tends to increase (especially considering more complex predictors).

Consequently, its important to test the previously mentioned metrics in different sample sizes, if the size of such samples introduces a considerable performance difference in the algorithms that are being employed.

Besides the input window size, there is the predicted window size to consider when comparing the different prediction schemes because some might perform better in smaller window sizes.

It could also be relevant to compare the results considering different gaps (m-gap prediction) between the measured input load values and the predicted values. Some prediction algorithms might have more accurate results on narrower gaps but, in comparison to other prediction models, produce less accurate results on wider gaps.

On the assumption that the time it takes to allocate resources from a private center to a public one is not substantially variable, the gap considered can be a fixed value and all the prediction schemes can be compared using the same gap value.

At a later stage it is also viable to, instead of just measuring the accuracy of the prediction schemes using the previous metrics, analyze the impact on the quality of service of the whole system and take that new information into account when evaluating a prediction algorithm[11].

The quality of service metrics will of course depend on the service itself, but analyzing generic metrics such as average response time or the number of rejected requests might provide additional information that can be used to compare the effects produced by different prediction approaches.

3

Important Technologies

As previously stated, metrics can be taken from various levels within a service, from the hardware (single device) to the higher levels of a whole application, firewall and/or load balancers. In this Section each of these concepts will be explained and some popular examples of monitoring software will be provided. Note that the main idea of this work is not to add additional probing or monitoring mechanisms (which has its associated costs and might affect the performance of the system). But instead leverage as much as possible on metrics which are already collected.

3.1 Load Balancers

In order to efficiently respond to all the millions of requests from the various users at the same time, multiple servers (potentially replicated) are necessary, this allows the system to scale and deal with higher demand. However there needs to be something that that balances the requests between the servers in a way that maximizes the performance and space efficiency and makes sure that no server is being overworked. Load balancers serve this purpose.

A **Load Balancer** is a component that distributes network load (for example, requests)across the different servers by choosing, for each request, which server that request is going to be routed to[59]. This component must also ensure high availability by sending requests primarily to servers that are working while being flexible enough so that new servers can be added and removed at runtime.

In order to effectively distribute the network traffic between servers, multiple techniques can be employed. The effectiveness of these algorithms will depend on the needs of each service (hybrid approaches between some of this techniques are also utilized)[29]:

Round Robin/Weighted Round Robin – Requests are routed to a group of servers in a repeated sequential loop. Its possible to turn a simple round robin into a weighted round robin by simply giving priority to some specific servers that will be repeated more often and thus receive more requests.

Least Connections/Weighted Least Connections – Requests are routed to the server

with the least number of connections to clients. If for example some servers have more computing power and can handle more requests than others, its possible to give priority to this servers (weighted least connections).

Least Time/Response time – Requests are routed to the server that responds the fastest. Hash – Requests are routed based on the has of a key defined (such as the client IP address or the request URL).

Resource Based – Requests are routed based on the availability status and resources of a server (using an agent that reports this information to the load balancer).

Random – Requests are routed randomly, employing no specific criteria.

At the Load Balancer level [5] its possible to monitor "High level" metrics that might be useful to forecast future conditions, for example, the total number of requests sent to the application by the users, the average response time of the service, the number of rejected connections, the number of current connections to the service and others.

3.2 NGINX and Traefik

NGINX and **Traefik** are examples of widely used load balancing software. They both rely on a reverse proxy to provide scalability, flexibility and increased security [56, 1].

A reverse proxy is fundamentally a public mask, meaning it is a server (with a public address) that is on the edge of a network that intercepts the client requests and then will communicate with the "real web server".

One benefit of utilizing a reverse proxy is **improved security**. Since the clients do not contact the IP address of the original server its easier to change the number of backend servers (scale up or scale down), this also offers a defense against diverse security problems such as DDoS (Distributed Denial of Service) attacks by not accepting requests from some sources or by putting a limit of requests accepted per client.

Another asset of utilizing a reverse proxy is speed up in response time (web acceleration). This happens as a result of techniques such as **compression**, **SSL termination** and **caching**.

By the **compression** of server responses its possible to reduce the bandwidth necessary, speeding up the network.

By **SSL termination** the reverse proxy spares the server of the task of decrypting and encrypting responses, these tasks are now performed on the proxy server so the main server can focus totally on responding to the client requests.

By **caching** client responses, making frequent requests perform faster.

NGINX is fairly simple to manage and tends to have better performance than Traefik [9], on the other hand, Traefik is very well documented and it is suited for dynamic environments (such as micro-service deployment and cloud management) [15], where it is easier to configure.

When comparing these two technologies for their monitoring features, the free version of NGINX does not make some monitoring information available (such as error rates)

while Traefik shows it for free. NGINX Plus offers a lot more monitoring features, but it is a paid version.

3.3 Firewalls

A firewall is a program (software) or device (hardware) that ensures the safety of a network by monitoring and restricting access from incoming and outgoing traffic. It represents the idea of a safety barrier that filters out unauthorized requests inside a network. Even though there are multiple types of firewalls in general a firewall will, using their predetermined rules if any incoming traffic is allowed to enter a network or if any traffic is allowed to leave, this rules are known as an access control list. This access control list rules can be customizable and are based usually on IP addresses, applications, domain names, key words, network protocols and ports.
There are several types of Firewalls that are employed for specific situations, but by comparing their approach to filtering data its possible to distinguish them by these types [55, 52]:

Packet Filtering Firewalls

Packet filtering firewalls operate inside the network, and instead of routing packets, this firewall type functions by examining IP addresses, port number and other packet options inside the packet header, if this information fits a predetermined set of rules the packet is allowed.

Circuit-level Gateway

Circuit-level gateways do not observe the packets, instead they work between the local and remote hosts and they function by monitoring TCP handshakes and other network protocol session initiation messages across the network. This is done in order to decide if the session initiated is legitimate. One major disadvantage of this type of firewall in the context of this study is that they provide no application layer monitoring, meaning metrics possible to gather from this approach will be fewer and less intuitive.

Application-level Gateway

Application-level gateway (often referred as proxy firewall) operates between the users and the edge of the network. Application-level gateways also filter packets according to multiple characteristics. The main and important advantage of this firewall type is that more complex metrics can be shown since, the firewall acts between the devices and the network monitoring every traffic that goes in, out or gets filtered.

While these services provide considerable data security, they can affect network performance and can be hard to manage.

Stateful inspection firewall

These types of services examine each packet and also monitors the state of every packet, discerning for example if the packet is from an already established TCP or another session. Providing more security than either packet filtering or circuit monitoring but at the greater cost on network performance.

Next-generation firewall

Next-generation firewall (NGFW) basically adds new features to stateful inspection firewalls, including intrusion prevention,deep-packet inspection, SSL and SSH inspection.

NGFW additionaly monitor intricate information about the traffic that is passing by it, meaning more monitoring data is provided in this firewalls than in traditional approaches.

3.4 Iptables and Check Point

Iptables and **Check Point** firewalls are two very popular firewall systems that are currently used by many provisioners.

Iptables [25] is a stateful inspection firewall that is used to define several tables with IP packet filter rules (that are normally stored in the kernel). Each table has a group of chains which are essentially a set of rules. For a packet to pass through the firewall it has to match one of the rules in the internal tables managed by iptables. The main tables of this solution are:

- The **filter** table, which is the default table, has the main role of packet filtering, meaning it has a set of rules that will decide if a packet is authorized to continue to its destination or if it should be dropped
- The **NAT** or Network Address Translation table has a set of translation rules that will modify the packet source or destination addresses in order to route the packet. It has features to alter the packet as soon as they enter, to alter local packets before routing and to alter packets that are about to go out. It is continuously used to allow access to machines in private networks to public networks.
- The **MANGLE** or Modify IP Headers table, is used as the name suggests to define rules that will alter the IP headers of each packet. It has features to alter incoming packets before routing, to alter local packets that were generated before routing, to alter packets when they enter in the box, to alter that are being routed through the box and to alter packets that are about to go out.

Iptables is a powerful tool that is widely used, for example for blocking IP addresses that can be used to secure a network.

Check Point Firewall[55] on the other hand is a next-generation firewall (NGFW), that supports features such as:

- VPN connectivity
- Identification and prevention of threats
- Filering network access

- Application control
- Data loss regulation

Check Point Firewall [52] works by enabling IP forwarding right after services start running. It denies all incoming packets while allowing outbound traffic by loading a default filter during the booting process. Every connection is added to the firewall table in order to track which traffic is authorized, this can be very efficient and increase scalability but can introduce problems when the aim is highly-available system.

3.5 SNMP

Starting with the technologies that focus on the collection and monitoring of information on a network, **SNMP** is the most barebones (hardware level) of the three discussed monitoring technologies.

It stands for Simple Network Management Protocol, and it is used to collect and organize, in a standardized way, device information on a network (via UDP ports). [13, 19, 12]

The great majority of devices that are connected somehow to a network support SNMP. Currently there are three versions of SNMP:

- **SNMPv1** which corresponds to the original version of the protocol
- **SNMPv2c** which corresponds to a revise version of SNMP that introduces community strings (which is basically a password).
- **SNMPv3** which is the last version and adds new authentication and encryption features.

An SNMP device (that is SNMP enabled) is called an agent, and it has several objects that can be interacted with (some objects will be industry standard, and some will be specific to the device).

All the objects that are specified in a device have an identifier that is used to address them (Object Identifier). The Object Identifiers are simply a sequence of numbers that are stored in a file called Management Information Base (MIB). This MIB objects are organized hierarchically in a tree Structure.

In order to interact and access the objects inside a MIB of a specific device on a network a Network Management System (NMS) is needed. This software component communicates with the agent using specific commands. The commands used by the NSM to communicate with the agent are:

- Get Requests that include Get, GetNext and GetBulk, which are used to actively request information from the agent.

- Set Requests that are used to change the value of an Object inside the MIB of a certain device.
- Trap and Inform Requests, that are normally utilized to monitor critical events. The difference between inform and trap requests is that inform waits for an acknowledgement, and therefore is more reliable.

3.6 Nagios

Nagios is an open source monitoring software that can be used to monitor applications, server infrastructures, networks and operating systems. It can be also used to generate alerts to the admins about some met conditions or exceptions[60].

Nagios offers monitoring of several services and protocols, including SNMP, and despite the fact that it was designed for Linux, Nagios can run in multiple operating systems such as Windows and Unix. Nagios is also easily extendable for each system via user written **Plugins**.

The **Nagios Architecture** is based on a server-client architecture and it consists of the following components: [33]

- The **Scheduler** which corresponds to the Nagios Server, periodically checks the employed plugins and acts in accordance to its results.
- The **GUI** which corresponds to the Nagios Interface, it is a web page that can send notifications to its users by changing the colors of the buttons.
- The **Plugins** are the components of Nagios that can be configured by its users. They make Nagios a software monitoring tool that can work in almost any system, service or infrastructure they also enhance monitoring capabilities in these fields. Plugins can be configured to check actively or passively for certain metric results and return those results back to the server.

Nagios provides both **agent-based** and **agentless monitoring**: [4]

In **Agentless monitoring** Nagios does not need to actively contact the system. It simplifies management since there is no need to install and manage a external agent.

SNMP monitoring is considered as agentless monitoring since it doesn't require a separate Nagios agent (even though internally, as previously explained, SNMP uses agents and therefore can be considered agent-based monitoring), its done by configuring Nagios to receive and compute SNMP traps[49].

Some of the cons of agentless monitoring in Nagios include the limitation of metrics to those that are supported by the protocol employed and also lower scalability since agentless methods tend to be heavier on the server.

Agent-based monitoring in Nagios, like other agent-based approaches, requires installation of an agent on the target system, this approach allows the addition of new monitoring features via diverse free available plugins which provides increased monitoring

depth, however (as previously mentioned) installing new agents on the systems increases management difficulty, and most agents require manual definition of every target system. Agent-based monitoring is more used by Nagios users than agentless monitoring because of all the monitoring capabilities that plugins offer.
Besides the previous distinction, Nagios has a free and open source version, **Nagios Core**, as well as a paid version which needs a software licence, **Nagios XI**.

Nagios XI offers new interface customization options such as custom dashboards and views, as well as new reporting and visualization capabilities. Users of Nagios XI do not necessarily need to know how to work with the command line, because this version has a more user-friendly interface.

Nagios XI also has enterprise edition that provides further sophisticated tools including reports that predict when devices reach their maximum capacity and the possibility to modify an enormous number of machines at the same time.

3.7 Prometheus

Prometheus is an open source monitoring system used primarily to monitor highly dynamic container environments, but it can be used in a more traditional non container infrastructure where there are just bare servers with applications deployed directly.

Prometheus functions by collecting and storing system metrics as time series data. Similarly to Nagios, it offers automated monitoring and alerting to its users.

Prometheus [39] provides an innovative pull approach where instead of the server being responsible for for pushing the metric data to a collection platform, there is a **Prometheus server** that periodically retrieves metrics from the systems or devices and stores it in a time series format. Multiple instances can pull metrics solving the problem of a single point bottleneck and relieving the clients from the extra monitoring load, these instances are called **node exporters**.

Node Exporters essentially convert the multiple metrics from the sources, to metrics that will later be scraped by the Prometheus server. They are configured to work for specific systems, for example, to monitor an SNMP device an SNMP node exporter needs to be installed, however manual reconfiguration of exporters is also possible.

Besides the Prometheus server and Node exporter, other important Prometheus components include[32]:

The **Push Gateway** that has the objective of allowing shorter lived jobs to push their metrics to something similar to a metrics cache. The need for this component is justified by the fact that for shorter lived jobs the server might not have time to pull metrics from them.

The **Alert Manager** where a user can define and customize the generation of alerts based on the metrics collected, it can also be configured to group alerts by their type and make some alerts have effects on each other.

The **Prometheus Query Language**, that enables users to query the data that exists in the database that can be shown then on the Prometheus web page or fed into an external API (for example Grafana).

Prometheus is fit for most infrastructure monitoring situations and cloud monitoring environments. It is also very suited for micro services because of its multi-dimensional querying capabilities.

3.8 Nagios and Prometheus Comparison

Nagios and **Prometheus** are two very different monitoring services, hence it is not easy to linearly compare them. The focus will be on comparing the enterprise version of Nagios XI (which costs 3495 US dollars) with Prometheus which is a free open source software that is continuously in development.

In terms setting up the environment [40, 41], Prometheus has the upper hand, Nagios initial set up requires the manual configuration of files while Prometheus is faster and easier to run. The number of integrations on Prometheus is also larger in comparison to Nagios, which has a limited number available. However, Nagios has community free plugins that make the system very flexible in contrast with Prometheus which is not as easy to customize and relies on extending itself with other applications by, for example, utilizing its query language.

One of the most important distinctions between the two is that Prometheus is more fit for cloud-based environments (mainly because of its easy Kubernetes integration) while Nagios is more suited for monitoring smaller an static environments, this distinction probably heavies the most when deciding between the two.

In terms of visualization capabilities, Nagios XI provides more custom options for graphs and dashboards, however Prometheus is more easily extendable with other tools like Grafana that provide a wide set of visualizations and graphs, and can manage historic data.

Summing up, the choosing between Nagios and Prometheus depends largely on the system environments on which they are employed. Prometheus is more recent and has a easier initial set up phase and more integrations with other applications, Nagios on the other hand is extremely flexible and can very easily scale out of the box while keeping maintenance easy. They can both be configured to monitor high level metric data (such as number of requests from a service) and low level metric data (such as cpu utilization of a specific machine), however the latter is easier in Nagios since, as previously mentioned this software was specifically designed for monitoring static infrastructures.

In our case study (NOVO BANCO) nagios is employed, hence it should be easy to access low level metrics from the machines in the deployment as well as networking measures.

4

Research Plan

In the previous chapters, some solutions regarding infrastructure monitoring, workload representation and workload forecasting were presented. It was also provided a brief depiction of the currently used technologies to manage and monitor large scale infrastructures. In this chapter the research plan of this thesis will be proposed. In Section 4.1, we refine the goals of the project and present the steps to achieve those goals. In Sections 4.2,4.3,4.4 and 4.5 further details will be provided about each step presented in Section 4.1. Finally, in Section 4.6 the scheduling of the overall research will be proposed.

4.1 Revisiting the Work Objectives

The goal of this thesis work is to research ways to combine metric data from different sources within a data center infrastructure and (in a reactive way) generate a prediction automatically based on that data. That prediction will then be utilized to generate an execution plan, where the options of migrating some resources to a public cloud will be explored.

As suggested, this research will then be divided in several steps:

- Firstly, there will be a step to collect and define the system requirements, so that a concrete plan can be devised, this entails modelling the monitoring metrics already available in the private data center of our case study (NOVO BANCO).
- Secondly, an implementation of a simple reconfiguration tool, that will be used to migrate resources to the cloud (or remove them) will be designed.
- Thirdly, there will a step to select and incorporate the different monitored metrics that will be used as input for some current methods of workload prediction. In this step, those viable prediction methods will be also tested on the system. This third step will happen partially in parallel with the previous one.
- Finally, an experimental evaluation will be conducted, comparing each considered method.

4.2 System Requirements and Constraints

Before starting to monitor the system, it will be crucial to understand what can be tampered with and what are the requirements to do so. Furthermore, it will be essential to know what technologies are being utilized on the use case architecture or what technologies are being employed to monitor the different components of the infrastructure.

The research will be bounded by the constraints defined by the use case, those will include:

- **Performance Constraints**, which represent the restrictions on how quickly should the system react to changes, including what is the time interval for the execution of the possible solutions.
- **Intrusiveness Constraints**, which represent, how intrusive can the monitoring be, including how much delay can the monitoring of the system introduce to its overall functioning.
- **Security Constraints**, which are the restrictions of security imposed on the system. Based on this constraints it will be possible to discern what resources can be migrated to the cloud as well as have an idea of the time requirements to migrate those resources, since this can heavily impact the time interval that is available for the next steps.

Based on the previous constraints, there will be an assessment of monitoring mechanisms. This will be the first task since it will be necessary to discern which technology will be used to monitor the system components as well as the way this monitoring process is conducted, for example, if installation of data probes onto the components is possible.

In addition to defining the monitoring mechanisms, it will be necessary to select which metrics better characterize the workload of the system. This metric data will be utilized in each workload prediction approach employed and therefore, could be adapted to each algorithm.

4.3 Implementation of a reconfiguration tool

In this step, a reconfiguration tool that will be responsible for the resource changes in the cloud will be implemented. This mechanism will be necessary to execute the necessary resource migrations to the cloud as well as the removal of such resources when they are no longer needed (to minimize the cost of the running the system).

More details to the implementation of this tool require a more in depth understanding of the technical characteristics of the NOVO BANCO infrastructures namely knowledge about which public cloud provider is currently being utilized.

This mechanism will be based on mostly ready to use technology and serves the purpose of informing the rest of the work on the available time window for reconfiguration.

4.4 Workload Prediction

In this step, an implementation of some workload prediction methods will be made. There will be an initial preference in utilizing schemes that are more popular and tested upon, since those methods are more likely to be reliable in the efficacy of their results. The plan is to implement three different workload prediction algorithms, which can be for example the ARIMA, Bayesian, and LSTM based model approaches. However, since we did not have the opportunity to access details related with the concrete monitoring data already collected by the bank, nor details about the access patterns to the bank infrastructure (due to lack of a signed non-disclosure agreement), we might need to revisit this decision, which will be done in the first task. Furthermore, the system requirements that include the security and time constraints inspected in the previous step, are expected to streamline the number and variety of prediction schemes tested.

Each of the previously employed solutions will be tested by their execution performance and how intrusive they are to the overall functioning of the system. The accuracy of each solution will be measured mainly by error measuring metrics that compare the predicted values to the actual values registered by the system.

4.5 Experimental evaluation

In this task, by utilizing the previously implemented reconfiguration tool, an experimental validation and evaluation of the offered strategies to predict workload as to guide migration of resources to the cloud based on the prediction results of each method will be conducted. To do this, it will be necessary to first identify what deployment adaptations, which can and need to be made on the output of our of our runtime prediction system, and when those adaptations should be reverted

Finally, if possible, the impact of each prediction on the system efficiency will be further evaluated by testing specific quality of service measurements, such as the response time of requests, after the migration of resources is executed.

To conduct these experiments we can resort to a (partial) replica of the servers that will be managed by the solution, and inject over this replicas (executed on a laboratory environment) artificial workloads based on historical data provided by NOVO BANCO. This experiment will allow us to study the actual behavior of different alternatives of the proposed solution.

4.6 Planning

In this section the scheduling of the work is presented, taking into account the previously mentioned phases of the work and the writing of the dissertation.

Table 4.1: Schedule Table

Task	Start	End	Weeks
Analysis of System Requirements and Constraints	March 1st	March 31st	4
Assessment of monitoring mechanisms			2
Selection of metrics			2
Implementation of the reconfiguration tool	April 1st	April 30th	4
Implementation of Workload Prediction schemes and accuracy Measuring	April 1st	June 30th	12
ARIMA Model implementation			4
Bayesian Model implementation			4
LSTM based Model implementation			4
Experimental Evaluation	July 1st	August 31st	8
Experimental migration of resources to the cloud			3
Response time comparison between all solutions			2
Thesis Writting	April 1st	September 30th	24

Figure 4.1 summarizes the work plan as a Gantt chart. Note that some taks due overlap in time since they can be done in parallel, for example while implementing the reconfiguration tool it will probably be possible to know the time interval for executing workload prediction algorithms, meaning this two tasks can be done in parallel. The writing of the thesis should also be done when each step is finished, meaning it should be a continuous process.

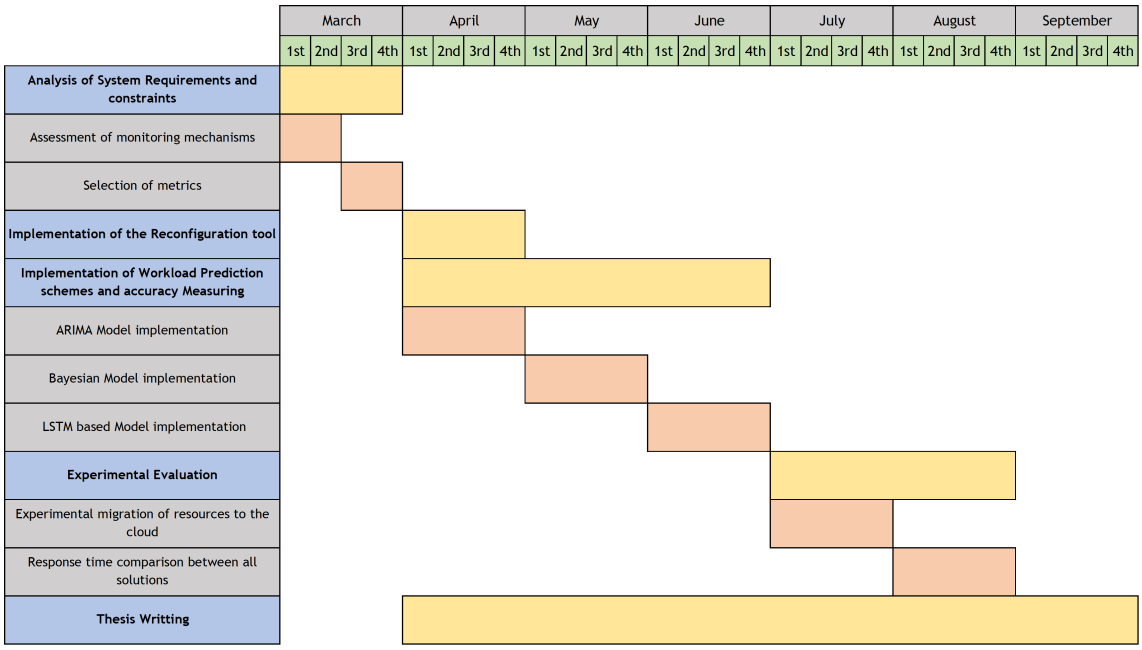

Figure 4.1: Work Schedule represented in a Gantt Chart

4.7 Final Summary

This document has presented our plan to conduct research on ways to extract metric data from different components of a system and combine it in order to infer additional and more precise information that will be used to predict the future workload of a system. We then explored workload forecasting techniques that can be applied to predict system utilization in real time.

We expect to implement some of the most used workload forecasting techniques and test their accuracy results by comparing the predicted values to the real values registered by the system. The choice of this techniques might be adapted to better fit our use case (NOVO BANCO).

We also expect to develop a scaling mechanism that will allow resources to be sent to a public cloud provider, as well as ensuring the removal of those resources when they are no longer needed in order to minimize costs. This mechanism will allow us to, by scaling the system based on each prediction results, compare the results of the solutions implemented by comparing not only their accuracy measures but also the improvement in request response time introduced by each one.

Bibliography

[1] https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/. visited on: 02-17-22 (cit. on p. 20).

[2] *1.4. Support Vector Machines*. en. URL: https://scikit-learn/stable/modules/svm.html (visited on 02/17/2022) (cit. on p. 13).

[3] L. Abdullah et al. "Predicting Multi-Attribute Host Resource Utilization Using Support Vector Regression Technique". In: *IEEE Access* 8 (2020), pp. 66048–66067. DOI: 10.1109/ACCESS.2020.2984056 (cit. on p. 13).

[4] *Agent-Based vs. Agentless Monitoring with Nagios*. en-US. May 2019. URL: https://www.nagios.com/news/2019/05/agent-based-vs-agentless-monitoring-with-nagios/ (visited on 02/17/2022) (cit. on p. 24).

[5] *An In-Depth Guide to Load Balancer Monitoring*. en-US. Feb. 2019. URL: https://blog.appoptics.com/an-in-depth-guide-to-load-balancer-monitoring/ (visited on 02/17/2022) (cit. on p. 20).

[6] M. Andreolini and S. Casolari. "Load prediction models in web-based systems". In: *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*. 2006, 27–es (cit. on pp. 5, 8, 9, 13).

[7] *Autoregressive Integrated Moving Average (ARIMA)*. en. URL: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp (visited on 02/17/2022) (cit. on p. 12).

[8] T. V. Batista and M. G. Carvalho. "Component-Based Applications: A Dynamic Reconfiguration Approach with Fault Tolerance Support". In: *Electronic Notes in Theoretical Computer Science* 65.4 (2002). SC 2002, Workshop on Software Composition Affiliated with ETAPS 2002 (Satellite Event of ETAPS 2002), pp. 13–21. ISSN: 1571-0661. DOI: https://doi.org/10.1016/S1571-0661(04)80434-1. URL: https://www.sciencedirect.com/science/article/pii/S1571066104804341 (cit. on p. 1).

[9] *Benchmarks - Træfik | Traefik | v1.4*. URL: https://doc.traefik.io/traefik/v1.4/benchmarks/ (visited on 02/17/2022) (cit. on p. 20).

[10] G. E. Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (cit. on p. 12).

[11] R. N. Calheiros et al. "Workload prediction using ARIMA model and its impact on cloud applications' QoS". In: *IEEE transactions on cloud computing* 3.4 (2014), pp. 449–458 (cit. on pp. 17, 18).

[12] D. J. D. Case et al. *Introduction and Applicability Statements for Internet-Standard Management Framework*. RFC 3410. Dec. 2002. DOI: 10.17487/RFC3410. URL: https://www.rfc-editor.org/info/rfc3410 (cit. on p. 23).

[13] D. J. D. Case et al. *Introduction to version 2 of the Internet-standard Network Management Framework*. RFC 1441. Apr. 1993. DOI: 10.17487/RFC1441. URL: https://www.rfc-editor.org/info/rfc1441 (cit. on p. 23).

[14] F. Castanedo. "A review of data fusion techniques". In: *The scientific world journal* 2013 (2013) (cit. on p. 8).

[15] B. Catanese. *From NGINX to Traefik (with Docker on DigitalOcean)*. en. July 2021. URL: https://medium.com/geekculture/from-nginx-to-traefik-with-docker-on-digitalocean-fcaeea3c6a4e (visited on 02/17/2022) (cit. on p. 20).

[16] S. Di, D. Kondo, and W. Cirne. "Host load prediction in a Google compute cloud with a Bayesian model". In: *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE. 2012, pp. 1–11 (cit. on p. 14).

[17] R. Dolphin. *LSTM Networks | A Detailed Explanation*. en. Dec. 2021. URL: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9 (visited on 02/17/2022) (cit. on p. 14).

[18] J. Fattah et al. "Forecasting of demand using ARIMA model". In: *International Journal of Engineering Business Management* 10 (2018), p. 1847979018808673 (cit. on pp. 12, 13).

[19] M. Fedor et al. *Simple Network Management Protocol (SNMP)*. RFC 1157. May 1990. DOI: 10.17487/RFC1157. URL: https://www.rfc-editor.org/info/rfc1157 (cit. on p. 23).

[20] J. Gao, H. Wang, and H. Shen. "Machine learning based workload prediction in cloud computing". In: *2020 29th international conference on computer communications and networks (ICCCN)*. IEEE. 2020, pp. 1–9 (cit. on pp. 13–17).

[21] N. R. Herbst et al. "Self-adaptive workload classification and forecasting for proactive resource provisioning". In: *Concurrency and computation: practice and experience* 26.12 (2014), pp. 2053–2078 (cit. on p. 5).

[22] P. J. Horn. "Autonomic Computing: IBM's Perspective on the State of Information Technology". In: 2001 (cit. on p. 4).

[23] *How to Calculate Mean Absolute Error in Python?* en-us. Nov. 2021. URL: https://www.geeksforgeeks.org/how-to-calculate-mean-absolute-error-in-python/ (visited on 02/17/2022) (cit. on p. 17).

[24] *An Architectural Blueprint for Autonomic Computing*. Tech. rep. IBM, June 2005 (cit. on p. 5).

[25] *iptables(8) - Linux man page*. URL: https://linux.die.net/man/8/iptables (visited on 02/18/2022) (cit. on p. 22).

[26] S. Karimi. *Agent vs. Agentless Monitoring: Which is best for your infrastructure?* en_US. URL: https://www.panopta.com/resources/agent-vs-agentless-monitoring/ (visited on 02/17/2022) (cit. on p. 6).

[27] J. Kumar, R. Goomer, and A. K. Singh. "Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters". In: *Procedia Computer Science* 125 (2018). The 6th International Conference on Smart Computing and Communications, pp. 676–682. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.12.087. URL: https://www.sciencedirect.com/science/article/pii/S1877050917328557 (cit. on p. 15).

[28] K. Kumar et al. "Forecasting of Cloud Computing Services Workload using Machine Learning". In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.11 (2021), pp. 4841–4846 (cit. on pp. 7, 17).

[29] *Load Balancing Algorithms and Techniques*. en-US. URL: https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques/ (visited on 02/17/2022) (cit. on p. 19).

[30] M. Masdari and A. Khoshnevis. "A survey and classification of the workload forecasting methods in cloud computing". In: *Cluster Computing* 23.4 (2020), pp. 2399–2424 (cit. on pp. 7, 11).

[31] *Mean squared error loss function | Peltarion Platform*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-squared-error (visited on 02/17/2022) (cit. on p. 17).

[32] MetricFire. *What is Prometheus? | MetricFire Blog*. en-US. URL: https://www.metricfire.com/blog/what-is-prometheus/ (visited on 02/17/2022) (cit. on p. 25).

[33] *Nagios Tutorial - Javatpoint*. en. URL: https://www.javatpoint.com/nagios (visited on 02/17/2022) (cit. on p. 24).

[34] *Naive Bayes Classifiers*. en-us. Mar. 2017. URL: https://www.geeksforgeeks.org/naive-bayes-classifiers/ (visited on 02/17/2022) (cit. on p. 13).

[35] P. Nehra and A. Nagaraju. "Host utilization prediction using hybrid kernel based support vector regression in cloud data centers". In: *Journal of King Saud University - Computer and Information Sciences* (2021). ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.04.011. URL: https://www.sciencedirect.com/science/article/pii/S1319157821000975 (cit. on p. 13).

[36] *novobanco Online | novobanco*. URL: https://www.novobanco.pt/particulares/novobanco-online (visited on 02/17/2022) (cit. on p. 3).

[37] D. L. Poole and A. K. Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010 (cit. on p. 13).

[38] *Prediction*. en-US. URL: https://www.datarobot.com/wiki/prediction/ (visited on 02/17/2022) (cit. on p. 9).

[39] Prometheus. *Overview | Prometheus*. en. URL: https://prometheus.io/docs/introduction/overview/ (visited on 02/17/2022) (cit. on p. 25).

[40] *Prometheus vs Nagios*. en-US. July 2020. URL: https://logz.io/blog/prometheus-vs-nagios-metrics/ (visited on 02/17/2022) (cit. on p. 26).

[41] *Prometheus vs Nagios | Learn Top 7 Comparisons with Infographics*. en-US. Sept. 2020. URL: https://www.educba.com/prometheus-vs-nagios/ (visited on 02/17/2022) (cit. on p. 26).

[42] *Quem somos | novobanco*. URL: https://www.novobanco.pt/investidores/sobre-nos/quem-somos (visited on 02/17/2022) (cit. on p. 3).

[43] V. Rajaraman. "Cloud computing". In: *Resonance* 19.3 (2014), pp. 242–258 (cit. on p. 1).

[44] P. Rattadilok et al. "A data fusion framework for large-scale measurement platforms". In: *2015 IEEE International Conference on Big Data (Big Data)*. 2015, pp. 2150–2158. DOI: 10.1109/BigData.2015.7364000 (cit. on p. 8).

[45] B. Raza et al. "Autonomic performance prediction framework for data warehouse queries using lazy learning approach". In: *Applied Soft Computing* 91 (2020), p. 106216 (cit. on p. 8).

[46] *SAS Help Center*. URL: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.4/statug/statug_introbayes_sect015.htm (visited on 02/17/2022) (cit. on p. 14).

[47] G. K. Shyam and S. S. Manvi. "Virtual resource prediction in cloud environment: A Bayesian approach". In: *Journal of Network and Computer Applications* 65 (2016), pp. 144–154. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.03.002. URL: https://www.sciencedirect.com/science/article/pii/S1084804516300091 (cit. on p. 14).

[48] P. Singh et al. "Research on auto-scaling of web applications in cloud: survey, trends and future directions". In: *Scalable Computing: Practice and Experience* 20.2 (2019), pp. 399–432 (cit. on p. 5).

[49] *SNMP Monitoring*. en-US. URL: https://www.nagios.com/solutions/snmp-monitoring/ (visited on 02/17/2022) (cit. on p. 24).

[50] B. Song et al. "Host load prediction with long short-term memory in cloud computing". In: *The Journal of Supercomputing* 74.12 (2018), pp. 6554–6568 (cit. on p. 15).

[51] X. Tang. "Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network". In: *IEEE Access* 7 (2019), pp. 40525–40533. DOI: 10.1109/ACCESS.2019.2905634 (cit. on p. 15).

[52] *The 5 Different Types of Firewalls Explained*. en. URL: https://www.techtarget.com/searchsecurity/feature/The-five-different-types-of-firewalls (visited on 02/17/2022) (cit. on pp. 21, 23).

[53] S. Venkateshan and P. Swaminathan. "Chapter 5 - Interpolation". In: *Computational Methods in Engineering*. Ed. by S. Venkateshan and P. Swaminathan. Boston: Academic Press, 2014, pp. 213–254. ISBN: 978-0-12-416702-5. DOI: https://doi.org/10.1016/B978-0-12-416702-5.50005-3. URL: https://www.sciencedirect.com/science/article/pii/B9780124167025500053 (cit. on p. 9).

[54] *What are Recurrent Neural Networks?* en-us. URL: https://www.ibm.com/cloud/learn/recurrent-neural-networks (visited on 02/17/2022) (cit. on p. 14).

[55] *What is a Firewall and Why Do I Need One?* en. URL: https://www.techtarget.com/searchsecurity/definition/firewall (visited on 02/17/2022) (cit. on pp. 21, 22).

[56] *What is a Reverse Proxy vs. Load Balancer?* en-US. URL: https://www.nginx.com/resources/glossary/reverse-proxy-vs-load-balancer/ (visited on 02/17/2022) (cit. on p. 20).

[57] *What Is ARIMA Modeling?* en-US. URL: https://www.mastersindatascience.org/learning/what-is-arima-modeling/ (visited on 02/17/2022) (cit. on p. 12).

[58] *What is Cloud Computing?* en-us. URL: https://www.ibm.com/cloud/learn/cloud-computing (visited on 02/17/2022) (cit. on p. 1).

[59] *What Is Load Balancing? How Load Balancers Work*. en-US. URL: https://www.nginx.com/resources/glossary/load-balancing/ (visited on 02/17/2022) (cit. on p. 19).

[60] *What is Nagios? - Definition from WhatIs.com*. en. URL: https://searchitoperations.techtarget.com/definition/Nagios (visited on 02/17/2022) (cit. on p. 24).

[61] Y. Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270 (cit. on p. 14).

[62] Y. Zhu et al. "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–18 (cit. on p. 15).