Workload prediction for resource management in data centers

Tuan Le Anh

Tuan Le Anh
VT 2016
Master of Science (one year) in Computational Science and engineering, 15 hp
Supervisor: Johan Tordsson
Examiner: Eddie Wadbro
Computational Science and Engineering, 60 hp

Abstract

Resource management is to arrange and allocate resources for computing operations and applications. In large scale data centers that contain thousands of servers, resource management is critical for efficient operation. To know workload characteristics in advance helps us proactively control resources in data centers, leading to benefits such as power savings and improved service performance. Workload prediction can be used, e.g., to decide how many resources to allocate for each application in a data center in the future. The accuracy of workload prediction varies depending on the used prediction methods and the characteristics of the workload. In this thesis work, we investigate three different methods: Linear Regression (LR), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), and Nonlinear Autoregressive Network with Exogenous Inputs (NARX). These methods are used to build models of resource consumption such as memory, CPU, and disk. Based on these models, future workload resource usage is predicted, and the accuracy of prediction is assessed. We analyze a trace from a production cluster at Google, predict resource consumption for different time intervals, and compute the error between predicted and actual values. The results show that NARX gives higher accuracy than ANFIS and LR when forecasting one-step ahead prediction, and that the ANFIS method provides the best result with multi-step ahead prediction compared to the others. Finally, time to train and re-train LR, ANFIS and NARX are computed. The running times are short, suggesting that the methods can be used in real-time operation.

Acknowledgements

I would like to thank my supervisor Johan Tordsson, who gave me the opportunity to do this nice thesis work as well as valuable feedback throughout this thesis. I would like to thank my fellow cloud researchers Abel, Amardeep, and Mohamad from KTH for nice help at the beginning of the project and thank Eddie Wadbro for his help with my study program. I also thank people in the UMIT research lab for a nice lunch time. Finally, thanks to my family, my wife Oanh and my son Vu, for support me during the thesis time.

Contents

1 Introduction

Workloads in data centers present much more diversity than in traditional scientific and super computing environments [1]. Thus, efficient resource management is a major challenge for such workloads, and each data center needs to find the best way to minimize its resource costs and enhance profits. It also has to ensure that workloads are allocated sufficient resources to make them run fast and have the desired quality of service (QoS). In order to effectively adapt resource allocations, prediction methods are required to forecast the amount of resources needed in advance and make allocations accordingly [2]. The predicted values represent, e.g., the amount of memory and CPU that must be allocated based on coming workload. For example, there could be a large upcoming computational operation that will occupy half of all resources (memory and CPU) of a data center in 5 minutes and this operation needs to be finished in 1 hour. At current time, over half of all resources are allocated for other operations. This means that we need to provide more resources for the data center to adapt to such a workload. However, if the number of resources is predicted incorrectly, consequently, too many or too few resources are allocated. This leads resource wastage and/or poor performance such as high latency or low throughput.

The goal of this thesis work is to analyze and characterize resource usage from a publicly available workload trace for a Google production cluster, to study a set of prediction methods, and to evaluate the accuracy of workload prediction when using different methods. First, data is collected regarding resource usage (in this work, we focus on analysis of RAM and CPU consumption) from the Google trace [3]. The data is smoothed by taking averages of CPU and RAM consumption over different intervals of one, two, and five minutes as well as one hour. Second, the data is investigated and analyzed to build resource consumption models using the LR [4], ANFIS [5] [6], and NARX [8] [20] methods. Based on these models, one-step or multi-step ahead prediction is performed to forecast future resource usage. Third, we use mean average percent error (MAPE) [10] to compute the error of prediction with different data. In addition, a sliding window technique that contains old and current data is used to reduce training time for the methods while keeping the accuracy of prediction as high as possible. Finally, in order to increase the accuracy of workload prediction, the models are re-trained during run time. To trigger re-training, an error tolerance is applied, in which a cumulative sum (CUSUM) test [11] [12] is used to detect when workload prediction error exceeds a threshold.

2 Background

2.1 Data centers

A data center is a facility that contains a large number of computers, up to tens of thousands of servers [13], and associated components such as storage and network equipment. These computers are connected by a network system. A data center commonly includes backup power supplies, redundant communication and environmental control (e.g. air condition, fire suppression, etc). To date, the power consumption of data centers is very large. For example, the data centers in US consumed 91 TWh of electricity in 2013 [14]. We believe that an effect resource management can improve power efficiency, and workload prediction is one of solutions for this. Based on predicted workload, we can allocate resources in data centers in an energy-efficient way, e.g., by turning devices such as physical machines on and off depending on workload.

2.2 Workload characterization

The term workload describes resource allocation, e.g., CPU-core and memory, in data center. Workloads in a data center can be classified based on several characteristics such as burstiness, trend, and periodicity [15]. Mixed workloads are often generated in patterns based on these characteristics. Workload burstiness represents sudden changes in loads, which is a major challenge for efficient resource management. Workload trends describes long-term increases or decreases in loads, while workload periodicity captures any periodically repeating behavior in the workload. For example, Figure 1 shows the workload from the 9th to 29th day of the 1998 FIFA world cup [16], showing a slightly increasing trend, burstiness with workload spikes after 200h and 380h into the trace, and a periodic pattern with repeating peaks every few days (when games were played).

2.3 Workload used in this thesis

In this thesis, the Google cluster trace is used as a workload to evaluate prediction methods. To fulfill the goals of this thesis, we need a workload that contains variation of load as well as characteristics of burstiness, trend, and periodic as mentioned in Section 2.2. The Google cluster trace fulfills this and suits for our work with workload predictions. Other traces, such as from the Umeå University supercomputer center HPC2N, can be found in [17].

2.3.1 Google trace cluster

The Google cluster trace was released by Google in 2011 and contains data collected from approximately 12,500 machines over a period of 29 days. These machines are homogeneous with a variety of memory-to-compute ratios, i.e., some having more memory,

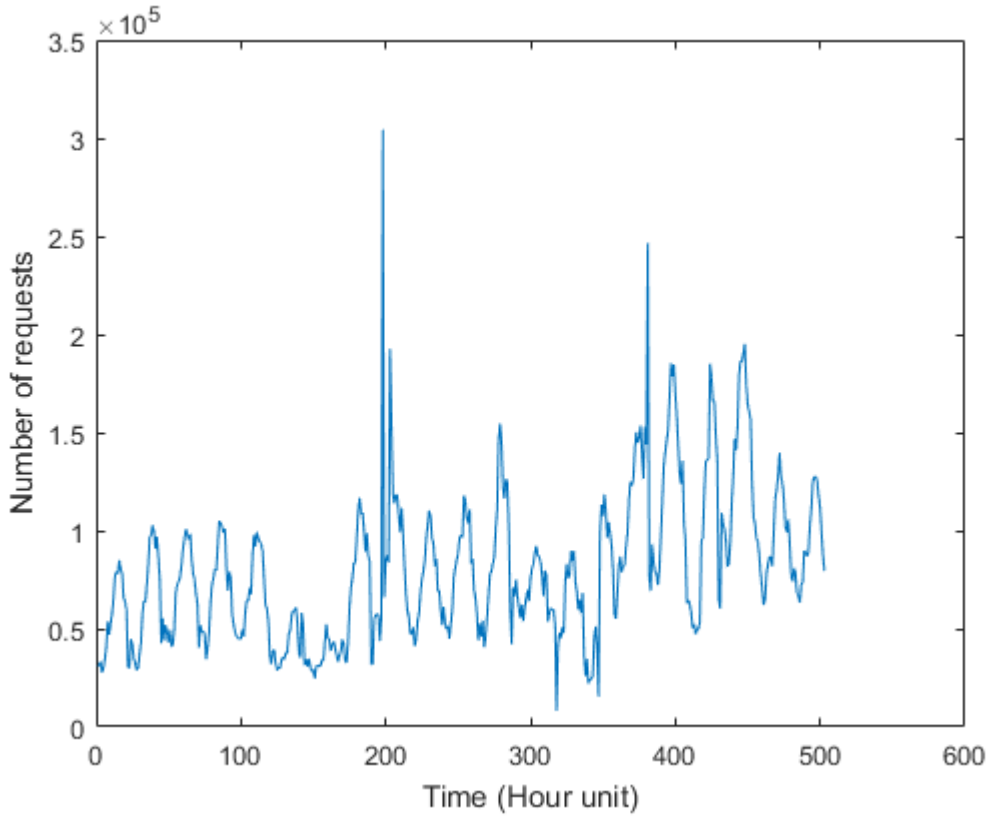


Figure 1: An example workload showing web server load from day 9 to day 29 of the official server for the FIFA World Cup 98. This shows trend, burstiness at the point 200th and 380th, and periodicity of peaks every few days (when games were played).

others more CPU cores, etc. The data is obfuscated, and most of the resource utilization measurements are normalized, including CPU, memory, disk, etc. Most of the machines have half of the CPU and half of memory of the largest machines. The data is collected from two sources: from the cluster schedulers and the individual machines. The Google cluster is a set of machines connected by a high-bandwidth network. A *cell* is a set of machines that share a common cluster manager used to allocate units of work to those machines. One such work unit is called a *job* . A job contains one or more *tasks* that consume resources like CPU, memory, etc. Each task represents a Linux program, which in term may include multiple processes.

The cluster data trace is stored in database tables based on machine information, jobs and tasks, task constraints, and resource usage. In our study, we mainly focus on analyzing memory and CPU usage of all of jobs in the trace. Therefore, all records in the resource usage table are collected to obtain the data for our work, in particular the records of memory and CPU usage.

2.3.2 Characteristics of the Google cluster trace

There are four characteristics of the Google cluster trace described in [18], including cluster utilization, scheduler operation, the priority of tasks, and properties of jobs and

tasks. Total resource allocation is overbooked by a factor of 80% for the cluster's memory capacity and more than 100% for CPU. However, the actual resource usage is much lower, with less than 50% of memory and 60% of CPU being used. A crucial part of data center operations is to improve these numbers to optimize resource utilization and maximize profit. We believe that good workload prediction methods can improve this. In terms of resource requests, the workload is diverse, and about 80% of the total requests come from long running jobs (running time is longer than one day). In this trace, 75% of jobs run only one task. Given these properties, prediction of resource consumption of tasks is not suitable for the LR, ANFIS, and NARX methods as these, and other prediction methods, require more than one data point to build models. As a result, we do not predict resource consumption of individual tasks. Instead, we predict CPU and memory consumption of all jobs in the Google cluster for different time units of one, two, five minutes as well as one hour, which is described in more detail in Section 5.

3 Problem specification

We would like to find a solution to increase overall usage of resource utilization in data centers by using workload prediction for resource management. This thesis mainly focuses on predicting CPU core and memory consumption with different time units in the Google cluster trace and compute prediction error generated by different prediction methods. The main goal is to find suitable prediction methods that accurately predict the workload in the Google cluster trace. In addition, we would find a simple method with fast running time to predict workloads, and adaptive methods to learn the changes in workloads. The other direction of this work is related to what time units are used for prediction. As long as the time units are chosen, respective workloads are then obtained from the Google cluster trace, mentioned in Section 5.

3.1 Problem statement

The problem consists three parts. Each part is associated with a research question that we need to answer. The three questions are as follows:

- How can workload predictions improve power consumption and resource allocation in data centers?
- Short-term resource allocation, e.g., elasticity: How many resources should be added or removed based on short-term fluctuations in demands in data centers?
- Long-term resource allocation, e.g., procurement: How large should a data center be, given how the workloads evolve over time?

3.2 Solutions to the problem

In order to answer the first question, we use workload prediction for CPU-cores and memory consumption. Based on predicted values, we can allocate resources for the upcoming workload by turning on/off physical machines in advance instead of letting the machines running idle. By doing this, we can save power consumption in data centers. The second and third questions are answered depending on the accuracies of short-term and long-term prediction respectively. To address to the second question, we predict CPU-cores and memory consumption with short time units of 1, 2, and 5 minutes, using prediction methods. These time units are chosen to have enough time to successfully turn on physical machines in data centers. For the question related to procurement, we choose the time unit of 1 hour to predict workload. Longer time units could be selected, e.g, 1 day or 1 week, however the Google cluster trace only contains data over 29 day period as mentioned in Section 2.3.1. There are sufficiently many data points to test prediction methods when using 1 hour time unit instead of 1 day or 1 week time unit.

3.3 Expected outcomes

We expect that the accuracy of workload prediction is as high as possible. The more accurate workload prediction, the more efficiently resources can be managed in data centers.. In addition, running time of prediction methods that is further mentioned in Section 5.3 is critical for us if workload prediction is to be applied in real operations. The acceptable execution time for prediction depends on the resource management task. For example, for elasticity, predictions must be made within in a few seconds, whereas for scheduling decisions, execution times up to a minute is commonly acceptable.

4 Prediction methods

Prediction are used for many different problems such as the fuel consumption of automobiles using ANFIS [7], stock price using NARX [20], auto-scaling in service clouds using LR [4], etc. In our strategy, prediction methods are utilized to predict memory and CPU usage of the whole Google cluster in different time units of one, two, five minutes and one hour. As the workload is irregular, we need methods that can adjust models according to the variation of the workload. To this end, we use LR, ANFIS, and NARX to predict memory and CPU consumption. The reason why we choose these methods is that LR is simple method with a fast execution time, whereas ANFIS and NAR are adaptive methods that can learn the changes in workloads. The LR method is used to build linear models, whereas ANFIS and NARX are deployed to build nonlinear models. These models are described in more detail below.

4.1 Linear regression model

Linear regression builds a model of the relationship between a dependent variable (output) and independent variable or variables (input). In cases where there is only one independent variable, the method is called simple linear regression. Otherwise, it is called multi linear regression. In our work, simple linear regression is used to predict resource consumption. The generic form of the LR model is

$$y_i = \alpha + \beta x_i + \varepsilon_i. \tag{4.1}$$

Here, x_i is independence variable, y_i is dependence variable, i is index of the sample of observations, and ε_i is ith noise item. In our case, y is the workload and x is the time. The goal is find the equation of the straight line

$$y = \alpha + \beta x. \tag{4.2}$$

The efficients α and β in Eq 4.2 are computed based on solving the following objective function of a least square minimization problem

$$min \sum_{1}^{n} \varepsilon_i^2 = min \sum_{1}^{n} (y_i - \alpha - \beta x_i)^2. \tag{4.3}$$

By using either calculus or the geometry of inner product spaces to solve Eq 4.3. The results as follows:

$$\beta = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_1^n (x_i - \bar{x})^2} \text{ and} \tag{4.4}$$

$$\alpha = \bar{y} - \beta \bar{x}. \tag{4.5}$$

Here, $\bar{x}$ and $\bar{y}$ are the average values of all observations of x, and y. β can be computed by substituting known previous workload in Eq. 4.4, and then α is calculated based on Eq. 4.5.

4.2 Adaptive Neuro-Fuzzy Inference Systems model

ANFIS is a type of artificial neural network based on the Takagi–Sugeno fuzzy inference system [5]. A inference system basically contains four functionalities: a rule base containing a number of fuzzy *if-then* rules, a database that defines the membership functions used in fuzzy rule, a decision-making unit that performs the interface operation on the rules, and an interface system that transforms a crisp input into a crisp output. ANFIS uses a hybrid learning rule that combines the gradient and the least square estimate methods to identity the parameters in an adaptive network. The architecture of ANFIS [5] is showed in Figure 2.

To exemplify, we assume a fuzzy inference system with two inputs x and y. Let us assume we have a rule base that contains two fuzzy *if-then* rules of the Takagi and Sugeno's type [19]:

- Rule 1: if x is A_1 and y is B_1, then $f_1 = p_1x + q_1\text{y} + r_1$ and
- Rule 2: if x is A_2 and y is B_2, then $f_2 = p_2\text{x} + q_2\text{y} + r_2$,

where p_i, q_i, and r_i are a set of parameters of the rule f_1 and f_2.
The node functions in Figure 2 are described as follows:

Layer 1 : Each node in this layer is a square node with a node function using either *Gauss* or *Generalized Bell* membership function:

$$O_i^1 = \mu_{A_i}(x) = \frac{1}{1 + [(\frac{x-c_i}{a_i})^2]^{b_i}} \tag{4.6}$$

or

$$O_i^1 = \mu_{A_i}(x) = e^{-\frac{x-c_i}{a_i}^2}. \tag{4.7}$$

Here, x is an input to node i, A_i (i = 1,2) is the linguistic label associated with this node function and O_i^1 is the membership function of A_i. In addition, a_i, b_i, and c_i are parameters to be learnt and adjusted. As a result, the shapes of the membership functions on A_i vary accordingly.

Layer 2 : Each node in this layer is a circle node labeled π which multiplies the incoming signal and send it to the next layer:

$$O_i^2 = w_i = \mu_{A_i}(x)\mu_{B_i}(y), i = 1,2. \tag{4.8}$$

Here, $\mu_{A_i}(x)$ and $\mu_{B_i}(x)$ are output of layer 1 and w_i represents the weight of rule i.

Layer 3 : Each node in this layer is a circle node labeled N. The ith node computes the ratio the ith rule's firing strength to the sum of all rule's firing strengths:

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2}, i = 1,2. \tag{4.9}$$

Here, w_1 and w_2 are the weights computed in layer 2.

Layer 4 : Each node in this layer is a square node with a node function:

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i), i = 1,2. \tag{4.10}$$

Here, $\overline{w}_i$ is the output of layer 3, p_i, q_i, and r_i are parameters, and f_1 and f_2 are the defined rules.

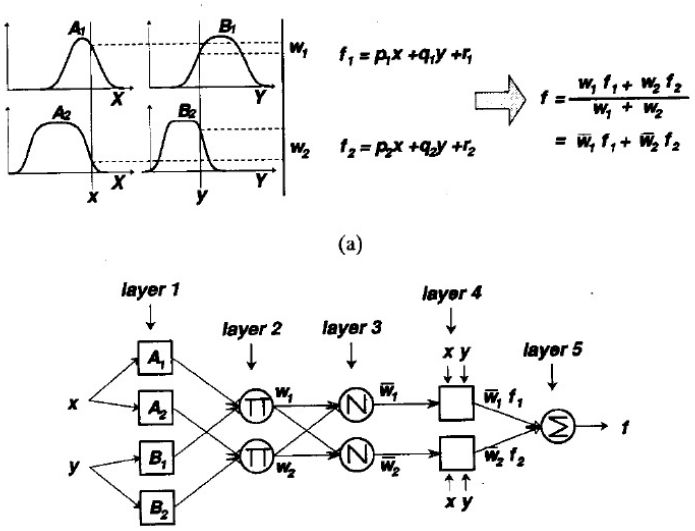


Figure 2: ANFIS architecture [5].

Layer 5 : The single circle node in this layer computes the overall output as the summation of all incoming signals:

$$O_i^5 = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{4.11}$$

The output of layer 5 is summed by all of the output of layer 4. Layer 5 is also the output value of the complete prediction for resource consumption like CPU-cores and memory.

4.3 Nonlinear autoregressive network with exogenous inputs model

In this section, NARX is introduced as a nonlinear autoregressive model including exogenous inputs to deal with chaotic time series. With this method, a model is built based on the past values of output and an exogenous series that represent a driving input. The typical equation of this model is defined as follows:

$$y_t = F(y_{t-1}, ..., y_{t-d}, ..., u_{t-1}, ..., u_{t-d}). \tag{4.12}$$

Here, y is a variable of interest (output), u is an exogenous (driving) input that influences the output y, F is a nonlinear function, and d is the number of previous values of input or output. A simple model can be seen in Figure 3 [20]. The model uses a feedforward neural network with two tapped delay lines that contain previous values of y and u respectively. The first line is used to store the d previous values of the input x, the second one stores the d past values of the output y. These make the model dependent on predicted values and old values of the input. More details about this model can be found in [20] [21].

Additionally, when no exogenous inputs are used in Eq 4.12, the model becomes nonlinear autoregressive (NAR). This is used in our experiments to predict resource consumption.

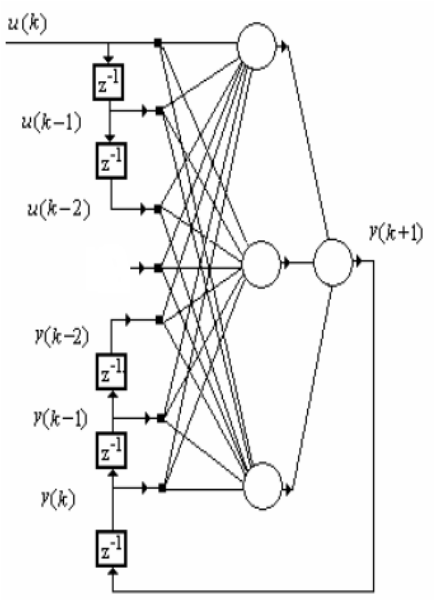


Figure 3: NARX architecture.

4.4 Error estimation

In order to compute the prediction error for the LR, ANFIS, and NAR methods, we use the mean absolute percentage error that is simple to compute and gives reliable results. MAPE is defined as:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{4.13}$$

where A_t is an actual value at time t, F_t is a predicted value at time t, and n is a total number of observations.

4.5 Adaptive training

During our experiments, we noticed that ANFIS and NAR sometimes use too many data points for training. This leads to an over-training problem and a long training running time. The problems can be solved if a sliding window with the size of w is applied, which contains the current data point as well as the last w data points. Adaptive training is a re-training process for ANFIS and NAR to learn the changes of workload recently. When a re-training processes is started, a method is re-trained with the last w data points, and a new model is then built. Additionally, we use the CUSUM test in which the residual between a *threshold* and MAPE is computed, and this number is added to a cumulative sum. The CUSUM test is used to captures the changes of workload, and to trigger a re-training process. Whenever the CUSUM test exceeds a *drift*, re-training is performed. The drift represents the magnitude of changes in MAPE, while the threshold represents the speed of the changes. A combination of them gives different scenarios. For example, a small threshold and drift leads to early triggered re-training, whereas a big threshold and drift can cause a late triggered re-training.

5 Experiments

In our experiments, the Google trace is used to evaluate the LR, NAR, and ANFIS methods. In total, there are 8 different workloads that are generated based on the Google trace. In order to create these workloads, we collect all records in the resource usage table of the Google cluster trace and take averages of CPU-cores and memory consumption of the whole data center over one, two, five minutes and one hour intervals. These workloads are named *workload 1,..., workload 8* respectively, and their characteristics are outlined in Table 1. The resulting workloads can be seen in Figure 4, showing that the number of spikes/fluctuations decrease when the data is smoothed with the different time units, from 1 minute to 1 hour time unit. The workloads have no unit for CPU-cores and memory consumption due to the normalization that all data in the trace are normalized. Here, we clarify that smoothing the workloads depends on how far we need to predict in the future. For example, when CPU-cores and memory consumption are smoothed with one hour interval. This means that for each predicted value, we can conclude that the average usage of CPU-cores and memory in next one hour is the predicted value. Additionally, all the workloads have the same duration of 29 days and starting point in time of the trace.

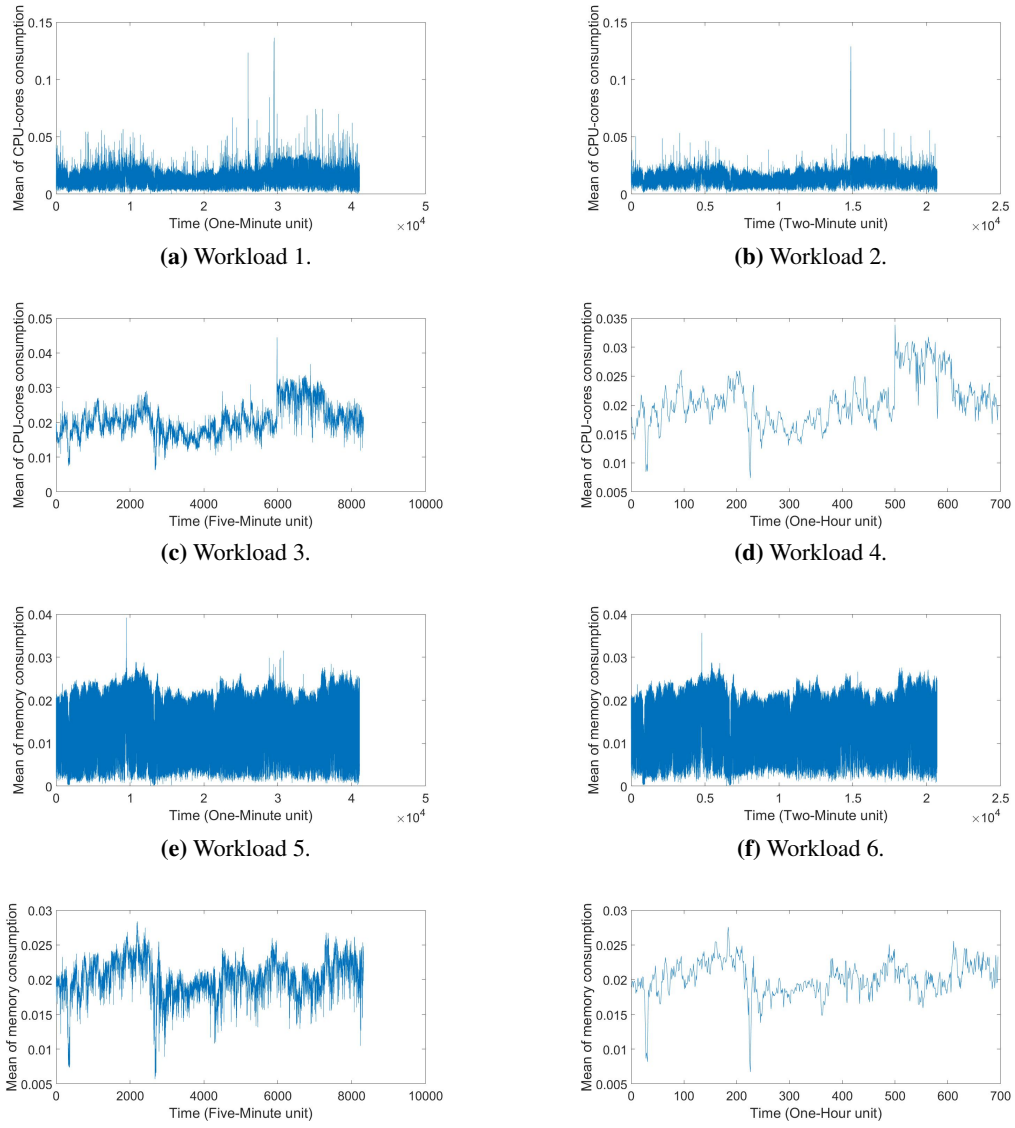
To implement the prediction methods and perform the evaluation, we use the following matlab toolboxes: neuro network toolbox that supports NAR, fuzzy logic toolbox that supports ANFIS, and statistics and machine learning toolbox that supports LR. Matlab scripts are created to perform our experiments. There are four types of experiments described in this chapter. First, one step ahead predictions, time t + 1, are preformed for all of the workloads, using the LR, NAR, and ANFIS methods. Second, the LR, ANFIS, and NAR methods are also used to predict 10 steps ahead, time t + 10, for CPU-cores and memory consumption. Third, we apply re-training using error tolerance for the ANFIS and NAR method to study the impact of retraining on prediction error. The LR method is not based on training data, thus there is no need to retrain it. The time required to re-build models is also investigated. Finally, a correlation analysis is performed to study the relationship between CPU-cores and memory consumption for all the workloads.

In the experiments, we notice that ANFIS and NAR have almost the same performance with high accuracies (MAPE less than 7.1%), followed by LR, when predicting workloads 3, 4, 7, and 8. However, LR, ANFIS and NAR perform worse for workloads 1, 2, 5, and 6, giving hight errors (MAPE larger than 50%). Many spikes in these workloads is the main cause of the larger prediction errors. The execution time of LR is much better than NAR and ANFIS, as LR makes no use of a learning algorithm, where NAR and ANFIS are base on training.

In addition, the ANFIS method requires an input dataset containing 4 inputs and 1 output collected from the workloads, while the LR and NARX methods only use single data series. To model a training data, the ANFIS method uses a hybrid learning algorithm combining the least-squares and back-propagation gradient descent methods [5], while the NAR method uses the Levenberg-Marquardt algorithm [22] for training models.

Table 1 Number of data points and used time unit in each workload.

	CPU-cores consumption			
	Workload 1	Workload 2	Workload 3	Workload 4
Time units	1 minute	2 minutes	5 minutes	1 hour
Number of data points (observations)	41074	20707	8324	695
	Memory consumption			
	Workload 5	Workload 6	Workload 7	Workload 8
Time units	1 minute	2 minutes	5 minutes	1 hour
Number of data points (observations)	41074	20707	8324	695



(a) Workload 1. **(b)** Workload 2.

(c) Workload 3. **(d)** Workload 4.

(e) Workload 5. **(f)** Workload 6.

(g) Workload 7. **(h)** Workload 8.

Figure 4: The eight workloads is used in the experiments, workloads 1,...,4 contain CPU-core consumption and workloads 5,...,8 describes memory consumption, using one, two, five minutes and one hour intervals respectively.

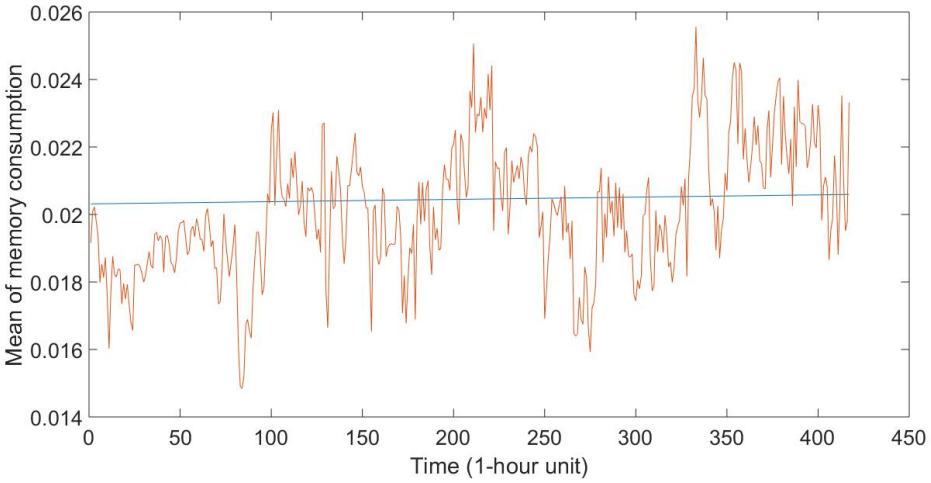


Figure 5: A comparison of actual values and predicted values generated by the LR method when predicting CPU-cores consumption of Workload 8.

Table 2 MAPE of CPU and memory consumption prediction using the LR method.

	Input (CPU-cores consumption)			
	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
MAPE(%)	61.7	61	17.3	15.9
	Input (memory consumption)			
	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
MAPE(%)	99.8	116	9.5	7.8

5.1 One-step prediction

In this section, one-step prediction is performed for the LR, ANFIS, and NAR methods. The main reason is that it creates smaller error than the error created by multi-step prediction. This is explained in Section 6. Additionally, one-step prediction is performed for short-term and long-term resource allocation, e.g, elasticity and procurement as mentioned in 3.1

5.1.1 LR experiments

We use the LR method to build models and forecast CPU-cores and memory usage according to each workload. In every workload, 40% of input data is selected to build the respective models, and the remaining 60% for testing. Based on the predicted values produced by the LR model, MAPEs are computed to evaluate the method. Figure 5 shows that the predicted values of CPU-cores consumption in the blue line do not fit with actual values shown as the red line. As expected, the line simply crosses Workload 8. For this experiments, the MAPE is 17.3 %. Errors of workload predictions of CPU-cores and memory usage decrease significantly when data is smoothed, as shown in Table 2. In particular, the errors of memory consumption predictions are reduced from 99.8% down to 7.8%. The reason is that number of spikes and fluctuations are reduced when the workloads are smoothed.

Table 3 A comparison of MAPE of CPU and memory consumption prediction using ANFIS when changing membership functions, membership numbers, and inputs.

		Input (CPU-cores consumption)			
Membership function	Membership number	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
Gaussian	2	**111**%	81.9%	6.7%	17,1%
Gaussian	3	126%	79.2%	7%	8.3%
Generalized Bell	2	112.4%	**77**%	**6.5**%	**7.3**%
Generalized Bell	3	129.7%	108.8%	6.6%	10.5%
		Input (memory consumption)			
Membership function	Membership number	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
Gaussian	2	285%	143%	5.5%	**4.4**%
Gaussian	3	289%	130%	5.5%	4.5%
Generalized Bell	2	**285**%	139%	5.5%	4.6%
Generalized Bell	3	304%	**108**%	5.5%	4.5%

5.1.2 ANFIS experiments

As the accuracy of workload prediction using the ANFIS method depends on initial parameters of number of membership functions and membership functions. Thus, we evaluate the methods when changing these parameters. To build ANFIS networks, Gauss and Generalized Bell membership functions are used, and the number of the membership functions are changed from 2 to 3 for each input data. We then divide input dataset into two parts: the first part (40%) for training the networks, the second part (60% of the input data) for testing. To start training ANFIS networks, for each training data, the data is then divided into two different parts: 70% of the data for training, 30% for validation. After finishing our experiments, we can see in Table 3 that Generalized Bell with 2 membership functions gives the smallest errors when predicting CPU-cores consumption with workload 1, 2, 3, 4, 5, 6, and 7, while Gauss with 2 membership functions is suitable for consumption prediction of memory with workload 8. Workload 8 predicted by ANFIS gives a good fit with the actual workload in Figure 6. A fraction of Figure 6 can be seen in Figure 7.

5.1.3 NAR experiments

In experiments using the NAR method, we collect 40% from each workload to train NAR networks and 60% of the data for testing. To start training NAR networks, for each training data, the data is then randomly divided into three different parts: 70% of the data for training, 15% for validation and other 15% for testing. As the data division is random, we repeat the training process 10 times and select the best model that contains the smallest error. This hopefully captures the workload characteristics sufficiently. For NAR, there are no rules for how to choose parameters such as the delay d and the number of hidden neurons. Therefore, we build the different architectures of NAR by changing these parameters to find the best results. In Table 4 and 5, the optimal parameters that yield the lowest MAPE are different for each workload. This leads a need to find the best parameters for each workloads when using NAR. For example, when making predictions for Workload 4, the NAR method gives the smallest error of CPU-cores consumption prediction when working with 10 hidden neurons and delays 1:3. Similarly, when we predict for memory consumption of Workload 8 with 7 hidden neurons and delay 1:2, MAPE is 4.6%. In addition, when NAR is used to

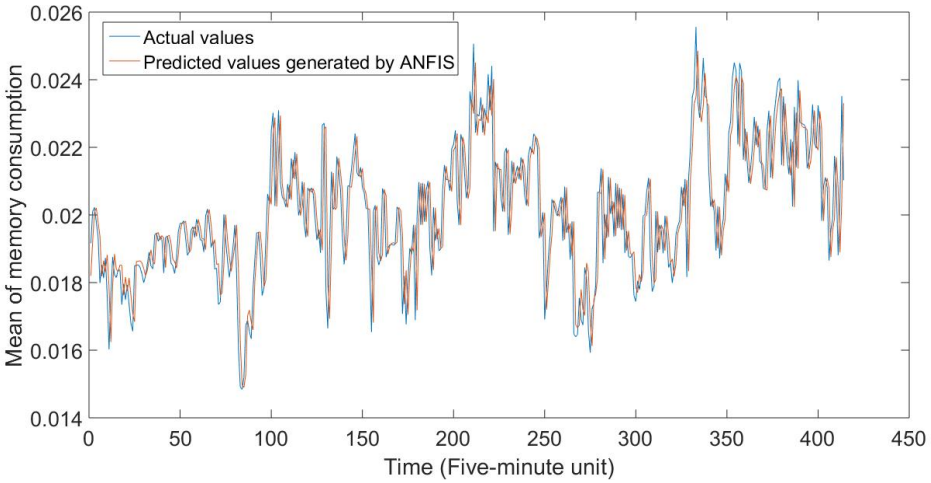


Figure 6: A comparison of actual and predicted values when using ANFIS to predict memory consumption of Workload 8.

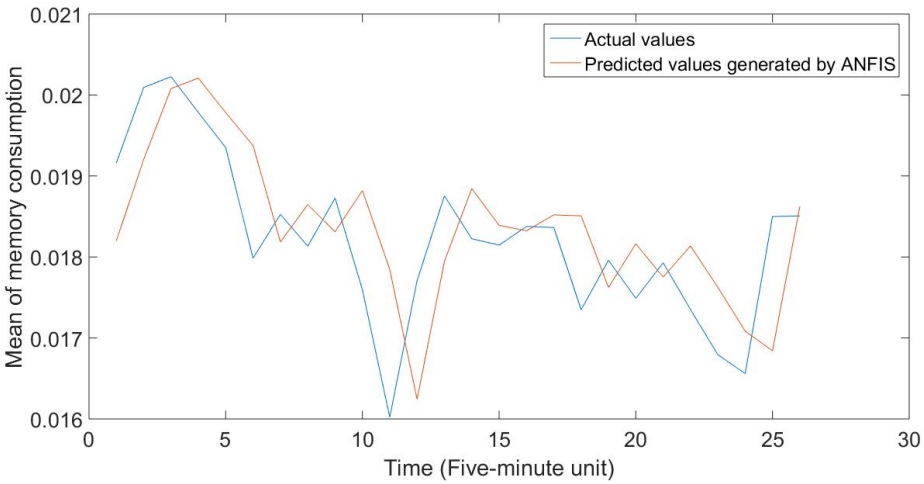


Figure 7: A close up with of a subset of Figure 6, which shows actual values and values predicted by the ANFIS method.

Table 4 A comparison of MAPE of CPU-cores consumption prediction using NAR when changing number of delays, network layers, and inputs.

	# hidden neurons with Workload 1 (1 minute unit)				# hidden neurons with Workload 2 (2 minute unit)				# hidden neurons with Workload 3 (5 minute unit)				# hidden neurons with Workload 4 (1 hour unit)			
Delays	7	8	9	10	7	8	9	10	7	8	9	10	7	8	9	10
1:1	64	64	63.9	64.3	56.3	56.5	56.3	56.5	7.9	7.2	8.4	8.4	8.1	8.9	7.5	9.1
1:2	60.7	61.1	60.8	61.3	52.8	51.8	52.8	51.7	7.3	7.1	7.1	7.10	10	8	8.3	8.6
1:3	**58**	58.4	59.3	59.9	**50.1**	50.3	50.4	51.3	8.3	8.3	**6.7**	7	7.2	9.4	7.4	**7.1**

Table 5 A comparison of MAPE of memory consumption prediction using NAR when changing number of delays, network layers, and inputs.

	# hidden neurons with Worload 5 (1 minute unit)				# hidden neurons with Workload 6 (2 minute unit)				# hidden neurons with Workload 7 (5 minute unit)				# hidden neurons with Workload 8 (1 hour unit)			
Delays	7	8	9	10	7	8	9	10	7	8	9	10	7	8	9	10
1:1	95.8	96.6	95.3	96	84.9	84.8	84,8	84.8	5.8	5.8	5.9	5.9	4.9	4.7	4.9	4.7
1:2	90	89.4	90.7	90	71.6	71.7	71.6	71.5	5.6	5.5	5.7	5.6	**4.6**	4.9	4.7	4.7
1:3	**80**	80	80.5	80.3	70.2	**69.9**	70	70.3	5.5	**5.4**	5.5	5.5	4.8	4.7	4.7	4.8

predict Workload 8, we get a good fit to the actual workload, as illustrated in Figure 8, with a zoomed view presented in Figure 9.

5.1.4 Comparing LR, ANFIS, and NAR

In this section, we collect the best results that were produced by the LR, ANFIS, and NAR methods in the previous experiments that use the same amount of training and testing data. The best results can be seen in Table 6, in which NAR outperforms for almost all of the workloads except Workload 3 and 8. The reason could be that NAR and ANFIS use different learning algorithms.

Based on the table, we return to the research questions discussed in Section 3.1 and note that ANFIS and NAR both provide good predictions for short-term and long-term when we predict CPU-cores and memory consumption using ANFIS and NAR with the time units of 5 minutes and 1 hour. The errors of the predictions are small, MAPE $\leq 7.1\%$. This results demonstrates that short-term and long-term workload predictions (research challenge 2 and 3 in Section 3.1) can be done with good accuracy. The prediction becomes worst when predicting Workloads with the short time units of 1 and 2 minutes due to too many spikes and fluctuations in the workloads.

5.1.5 Sensitivity to training and input data size

We study the error behavior when the lengths of training data and input data are changed by increasing the sizes of them gradually. First, we observe error behavior when changing training data sizes. The length of training data is increased from 5% to 80% for each workload, and predicted values are then estimated from the last 20% of the workload. We find that prediction errors tend to decrease when the size of training data increases, and then remain almost the same level as in Figure 11. This means that we need to stop training to save training time. However, there are spikes in errors in Figure 10 as training set size increases. Thus, the accuracy of workload prediction sometimes does not improve with an increasing training data size.

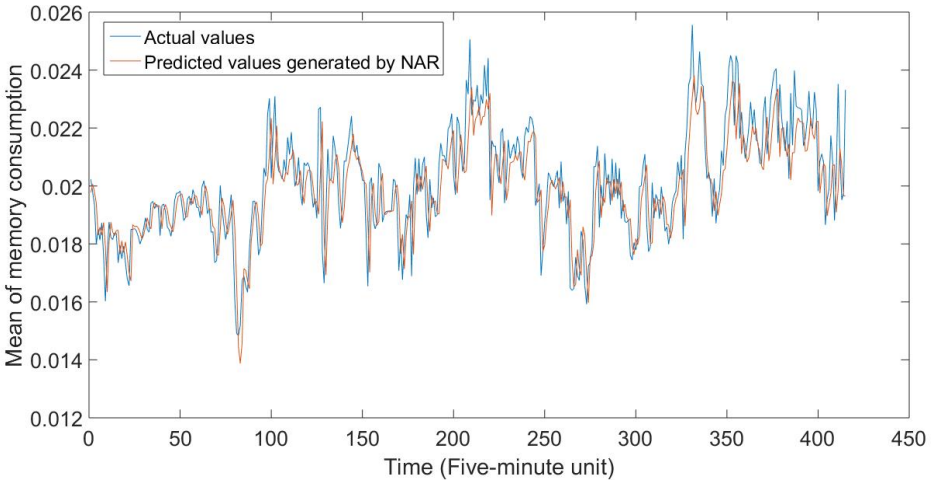


Figure 8: A comparison of actual values and predicted values generated by the NAR method when predicting CPU-cores consumption of workload 8.

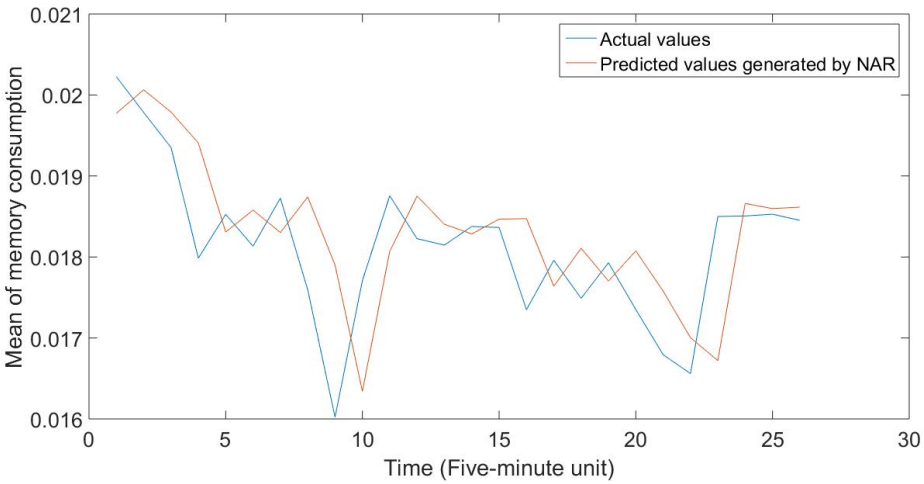


Figure 9: A close up with of a subset of Figure 8, which shows actual values and values predicted by the NAR method.

Table 6 The best results created by the LR, ANFIS, and NAR methods with 8 different workloads.

		Input (CPU-cores consumption)			
Methods	Error	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
LR	MAPE (%)	61.7	61	17.3	15.9
ANFIS	MAPE (%)	111	77	**6.5**	7.3
NAR	MAPE (%)	**58**	**50.1**	7	**7.1**
		Input (memory consumption)			
Methods	Error	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
LR	MAPE (%)	99.8	116	9.5	7.8
ANFIS	MAPE (%)	285	130	5.5	**4.4**
NAR	MAPE (%)	**80**	**69.9**	**5.4**	4.6

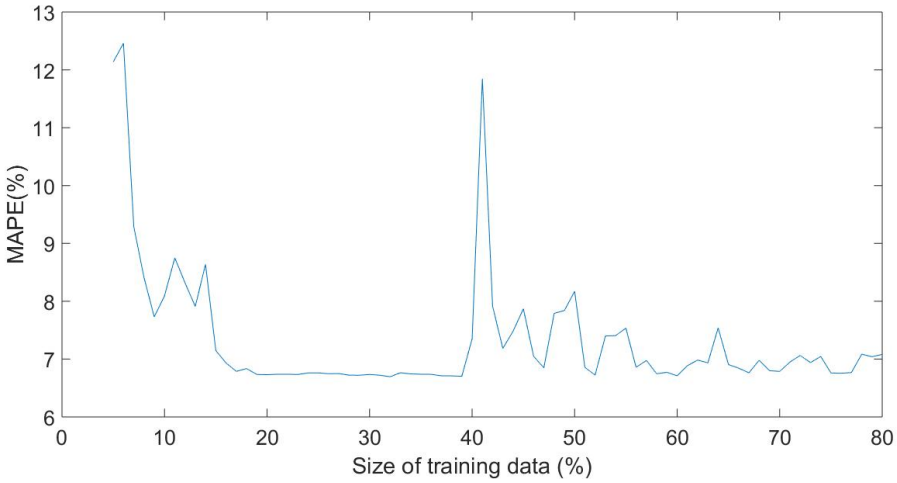


Figure 10: Error of workload prediction using ANFIS with Workload 7 tends to decrease when increasing the length of training data.

In addition, the size of input data is also studied for the LR, ANFIS, and NAR methods. We take 40% of input data to build models, and the remaining part for testing while the size of input data is gradually increased with an increment of 1% from 5% to 100%. The results show that the errors of workload predictions are large at the beginning and tend to decrease before going up gradually. Figure 12 shows that MAPE of CPU-cores consumption predictions produced by the LR, ANFIS, and NAR methods decrease significantly at the beginning and then tend to increase. Therefore, the length of input data affects on the accuracy of prediction. This result demonstrates the need to find a suitable length of each input data to keep the prediction error low.

5.2 Multi-step prediction

In this section, we use LR, ANFIS, and NAR to predict 10 steps ahead for CPU-cores and memory consumption. In order to reduce running time, we only take 10% data (In previous

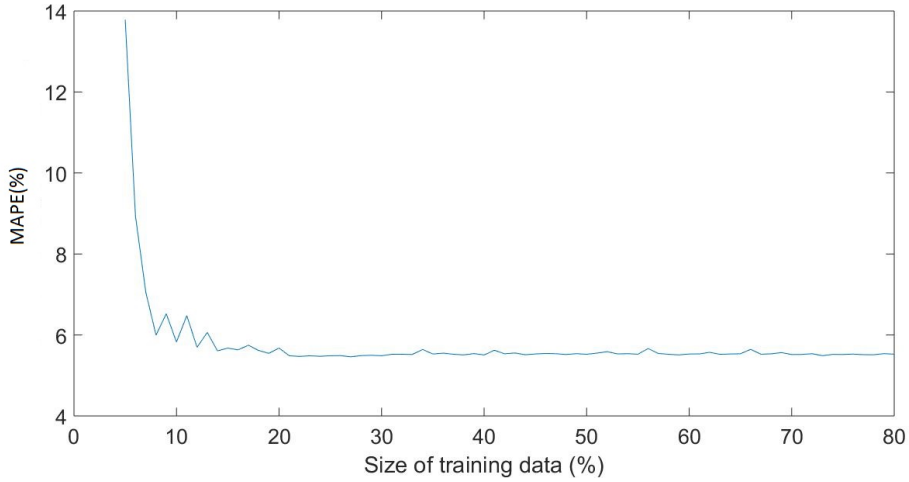


Figure 11: Error of workload prediction using NAR with Workload 7 converges when increasing the length of training data.

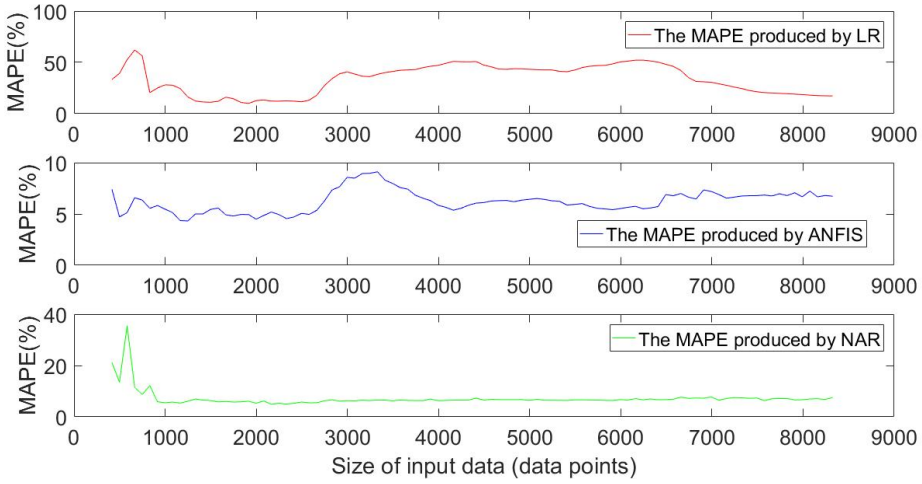


Figure 12: Error behavior of workload prediction using LR, ANFIS, and NAR with Workload 3 when increasing the length of input data.

Table 7 A comparison of MAPE of memory and CPU-cores consumption when predicting 10 steps ahead using the LR, ANFIS, and NAR method.

		input (CPU-cores consumption)			
Models	Error	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
LR	MAPE (%)	**78.9**	55.7	78.1	59.6
ANFIS	MAPE (%)	88.6	64.9	**8.7**	17.3
NAR	MAPE (%)	82	**54.8**	15	**12.6**
		Input (memory consumption)			
Models	Error	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
LR	MAPE (%)	**97.4**	**80.5**	36.2	44.8
ANFIS	MAPE (%)	276	103.7	**6.8**	**6.2**
NAR	MAPE (%)	107.6	118.2	11.3	8.9

experiments, this number is 40%) of each workload to train ANFIS and NAR networks, this number is also used to build the LR model. Here, there are two different ways to build NAR and ANFIS networks based on 10% of input data for training. We build the ANFIS network with 10 different ANFIS models for 10 different time steps accordingly, and the NAR network is deployed to use previous predicted values as an input to predict for next values. To evaluate the networks and the LR model, 1 random point is selected in 90% of the rest of the workloads, 10 future values are then predicted, and MAPE is finally computed. This procedure is repeated 10 times and an overall error is computed by taking average of the MAPEs. Similarly, 20% data of workload 3, 4, 7, and 8 are collected to train and build ANFIS and NAR networks, and LR model, and overall MAPE are calculated accordingly. As shown in Table 7, ANFIS gives the smallest errors when predicting 10 steps ahead for workloads 3, 4, 7, and 8, while LR produces the best predictions for workloads 1, 2, 5, and 6.

5.3 Re-training model and running time

As models could be outdated with the changes of upcoming workload, we employ a re-training approach where these models are re-built to capture new characteristics of the workload. For these experiments, we fix the length of the sliding window, w = 500, for workload 1, 2, 3, 5, 6, and 7, and w = 50 for workloads 4 and 8. The reason for choosing these numbers is that, for example, the number of data points in workloads 4 and 8 are far fewer than the others, and w = 50 is enough to captures the characteristics of workloads 4 and 8 sufficiently. These lengths were determined empirically. To trigger a re-training process, an error tolerance is applied, in which the CUSUM test is used to detect the changes of prediction error, as mentioned in 4.5. To compute a threshold as mentioned in 4.5, we use the following formulation:

$$threshold = m + s * e. \tag{5.1}$$

where m is mean target computed by taking average of all MAPEs produced by the ANFIS, or NAR method from the first w data points of workload. s is a standard deviation computed

Table 8 A comparison of MAPE with and without re-training, using ANFIS methods.

	Input (CPU-cores consumption)			
	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
Parameters	e = 0.2 drift = 1200	e = 0.2 drift = 2000	e = 0.2 drift = 700	e = 0.2 drift = 700
MAPE(%) without re-training	99.1	78.1	15.4	10
MAPE(%) with re-training	141.8 (+42.7)	85.5 (+7.4)	7.4 (-7)	7.4 (-2.6)
	Input (memory consumption)			
	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
Parameters	e = 0.2 drift = 2000	e = 0.2 drift = 2000	e = 0.2 drift = 800	e = 0.2 drift = 800
MAPE(%) without re-training	244	238.6	6.8	29.8
MAPE(%) with re-training	676 (+432)	163.4 (-75.2)	6.6 (-0.2)	15.7 (-14.1)

with the same number of data points (here, data points are MAPEs). In Eq. 5.1, e is an efficient (a positive number) that is changed depending on workload.

To perform experiments, in Eq. 5.1, e and *drift* parameters are manually chosen. We fix the efficient, $e = 0.2$, and change the drift. In these tests, 7% of input data is used to train ANFIS and NAR networks, 93% to test the methods with re-training processes. The results can be seen in Tables 8 and 9. Re-training processes using ANFIS or NAR are not efficient when predicting with workload 1 and 2, while it works efficiently for workloads 3, 4, 6, 7, and 8 for ANFIS, and workloads 3, 4, and 7 for NAR.

To assess the feasibility of using the studied prediction methods in data center operation, we study the execution time needed to train the ANFIS, NAR networks, and LR model. We use different numbers of data points in workload 1 to build the ANFIS, NAR, and LR models and compute the running times accordingly. As seen in Table 10, the running time of LR increase slightly while increasing the size of data. ANFIS consumes more time to build its models when the number of amount of training data increases. We conclude that LR has lowest running times to build its models, followed by NAR and ANFIS. Once the models are build, all methods are efficient when making predictions, the time to predict one step ahead using LR, ANFIS, and NAR models is less than 20ms.

5.4 Correlation study

To perform statistical experiments, we compute the correlation of CPU-cores and memory consumption to see any relationship between them. Each workload pair in which both workloads have the same time unit is chosen to compute the correlation. Table 11 shows that the is quite a strong relationship between CPU-cores and memory consumption. The variance of each workload is also calculated in Table 12, in which the variances tend to decrease when the workloads are smoothed. In addition, the correlation of the variances in the table and the best results, in Section 5.1.4, generated by LR, ANFIS, and NAR is very high, close to 1. As a result, the relationship between prediction error and variance is strong, i.e., the prediction error is small for workloads with low variance and vice versa.

Table 9 A comparison of MAPE with and without re-training, using NAR method.

	Input (CPU-cores consumption)			
	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
Parameters	e = 0.2 drift = 1000	e = 0.2 drift = 1000	e = 0.2 drift = 500	e = 0.2 drift = 700
MAPE(%) without re-training	56.4	48.3	8.6	9.7
MAPE(%) with re-training	59.4 (+3)	48.2 (-0.1)	6.5 (-2.1)	8.8 (-0.9)
	Input (memory consumption)			
	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
Parameters	e = 0.2 drift = 1000	e = 0.2 drift = 1000	e = 0.2 drift = 700	e = 0.2 700 = drift
MAPE(%) without re-training	75.5	68.5	7	6.8
MAPE(%) with re-training	83.6 (+8.1)	73.3 (+4.8)	6 (-1)	7.5 (+0.7)

Table 10 A comparison of running times when building LR, ANFIS, and NAR models.

	Number of data points	LR	ANFIS	NAR
Running time (s)	500	0.016	0.22	0.3
Running time (s)	1000	0.015	0.46	0.4
Running time (s)	2000	0.016	0.98	0.5
Running time (s)	30000	0.024	13	11

Table 11 Correlation between CPU-cores and memory consumption becomes small when data is smoothed.

	Workload pairs			
	Workload 1 and 5 (1 minute unit)	Workload 2 and 6 (2 minute unit)	Workload 3 and 7 (5 minute unit)	Workload 4 and 8 (1 hour unit)
Correlation	0.7302	0.7808	0.4624	0.4372

Table 12 Variances of 8 workloads. The variances tend to decrease when the workloads are smoothed.

	CPU-cores consumption			
	Workload 1 (1 minute unit)	Workload 2 (2 minute unit)	Workload 3 (5 minute unit)	Workload 4 (1 hour unit)
Variance	0.61e-04	0.6027e-04	0.2056e-04	0.1806e-04
	Memory consumption			
	Workload 5 (1 minute unit)	Workload 6 (2 minute unit)	Workload 7 (5 minute unit)	Workload 8 (1 hour unit)
Variance	0.5287e-04	0.6765e-04	0.0762e-04	0.0613e-04

6 Discussion

In this chapter, we discuss the advantages and disadvantages of the methods and what type of predictions can be solved with LR, ANFIS, and NAR. The Google cluster trace and other traces are also discussed.

In the experiments, we see that the exact results of prediction methods will be varied when we use with another workload. However, the overall trend would be the same. LR usually gives high error but has low running time. It is not good for workload with large variation. ANFIS gives small errors with workload containing few spikes and fluctuations, but can take long time to train. NAR produces both good predictions and has short training time, however the running time is still higher than LR. Based on these characteristics of the methods, one can consider to choose the most suitable prediction method the problem at hand.

In this work, we use the Google cluster trace to evaluate the three prediction methods. To test these methods, the eight different workloads are created based on different time units of 1, 2, 5 minutes and 1 hour. The reason that these time units are chosen is explained in Section 3.2. Workload prediction are not the same for different problems. For example, if the problem is to predict resource consumption of jobs that come from only one cluster manager, the workload should probably be created by selecting the historical data of the resource consumption from the cluster manager, and taking either *average* or *sum* of resource usage over a specific time unit. Similarly, when workload prediction is applied for other traces, for example HPC2N trace in [17], we need to determine what data needed to obtain for workload and what time unit for workload prediction.

Another discussion is that when should we choose multi-step prediction instead of one-step prediction? This depends on how big error of prediction for problems that can stand. For example, if workload is predicted by using one-step prediction with Workload 3 to compute CPU-cores consumption in next 5 minutes. The MAPE is 6.5 %. This error could be bigger if 5-step ahead prediction is used to predict CPU-cores consumption in next 5 minutes using Workload 1 (1 minute time unit). As every one-step ahead prediction of Workload 1 gives big errors, MAPE $\geq$ 50%, see Table 6. However, here the benefit of using the 5-step prediction with Workload 1 is that we do not need to smooth workload data like Workload 3 to predict one step ahead of CPU-cores consumption in next 5 minutes.

7 Related works

A lot of research are conducted in workload prediction. Haibo et al [23] use a double exponential smoothing method to predict the future workload of applications. Based on the predicted values, an online self-configuration approach is used for reallocating virtual machines in large-scale data center. They use FIFA World Cup 98 trace to train the method. However, they do not mention the accuracy of their workload prediction. Workload prediction for auto-scaling technique is also mentioned in the paper of Jingqi et al [24]. It uses linear regression model to build a model of workload and predict workload in the future. The result is compared with three other methods such as Autoregressive Moving Average (ARMA), Mean and Max methods, which linear regression method has the highest accuracy. A paper by Roy et al [25] develops a model-predictive algorithm for workload prediction, in which a second order autoregressive moving average method is used. In the paper of M Rasheduzzaman et al [26], ANFIS, NARX, and Autoregressive Integrated Moving Average (ARIMA) prediction methods are also utilized to predict resource consumption such as memory and CPU-cores consumption in the Google Cluster trace. The result shows that the NARX method outperforms the others. However, these methods are not tested with multi-step prediction and re-training processes. The authors do not show training running times for the methods.

The Google trace cluster is analyzed in [18], in which characteristics of resource requests, resource distribution, and actual resource utilization are studied carefully. In the paper of Mansaf Alam at al [27], a statistical study in the Google trace is preformed. The paper shows that jobs in the trace can be classified into three types of jobs: long jobs, medium jobs, and short jobs, and behavior of tasks within a long job type are predictable.

8 Conclusion and future work

In this thesis, we use the LR, ANFIS, and NAR to predict CPU-cores and memory consumption in the Google cluster trace. During our experiments using the methods, we observe that the accuracy of workload prediction sometimes does not improve with an increasing training data size. A big training data causes an over-training and a long training time. A sliding window is a good solution to solve these problems. The length of input data for the three methods affects on the accuracy of prediction. The relationship between the length and the accuracy of prediction is not linear, and we should find a suitable length of each workload for each method. We characterize the Google cluster trace and find that memory and CPU-cores consumption have quite a strong relationship.

We archive the goal of this thesis by answering the last two research questions in Section 3.1 when predicting with workloads 3, 4, 7, and 8. These predictions give high accuracy, MAPE $\leq 7.1\%$.

We assess LR, ANFIS, and NAR based on two factors: training time and precision. NAR predicts more accurately than the others in most of the one-step prediction testing cases and uses short time for training and prediction. Therefore, it is a good method for operations that need high accuracy and low running time. In our views, LR is the best choice for high latency-sensitive problems due to its very low running time. ANFIS, and NAR are fast enough for all type of scenarios due to prediction time $\leq$ 20ms. ANFIS efficiently works with workload containing few spikes/fluctuation.

Currently, we have no rules to select a suitable drift for an adaptive training. An automatic re-training process where the drift is automatically adapted based on the workload is a future goal. Furthermore, this work is mainly focused on workload prediction without involving a real operation. We propose that workload prediction using LR, NAR, and ANFIS should be applied for resource management in a real data center to see how much more efficient resource management can be done. By doing this, we are able to evaluate the methods in a real time. The running time and the prediction errors are also computed in this work. The results is hopefully useful for researchers that would use LR, ANFIS, and NAR to predict other types of workloads, for example when predicting electricity consumption, weather, stock prices, etc. However, they should test the methods with their own workloads to find suitable parameters, etc.

References

[1] C. Reiss, A Tumanov, GR Ganger, and RH Katz. *Heterogeneity and dynamicity of clouds at scale: Google trace analysis*. Proceeding SoCC '12 Proceedings of the Third ACM Symposium on Cloud Computing, No. 7. 2012.

[2] T. Lorido-Botrán, José Miguel-Alonso, José A. Lozano. *A View of Cloud Computing*. Journal of Grid Computing, pp. 1-34, 2014.

[3] C. Reiss, J. Wilkes, and JL. Hellerstein. *Google cluster-usage traces: format + schema*. Google. 2011.

[4] GAF. Seber and AJ. Lee. *Linear regression analysis*. A John Wiley and sons publication. 2003.

[5] Jyh-Shing and Roger Jang. *ANFIS : Adaptive-Network-Based Fuzzy Inference System*. IEEE transactions on system, man, and cybernetics, vol.23, no.3, pp. 665-685. 1993.

[6] J. Soto, P. Melin, and O. Castillo. *A new Approach for Time Series Prediction using Ensembles of ANFIS Models with Interval Type-2 and Type-1 Fuzzy Integrators*. IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pp. 68-73. 2013.

[7] H. Atmaca, B Cetisli, and HS Yavuz. *The Comparison of Fuzzy Inference Systems and Neural Network Approaches with ANFIS Method for Fuel Consumption Data*. Second International Conference on Education. 2001.

[8] Hava T. Siegelmann, Bill G. Horne, and C. Lee Giles. *Computational Capabilities of Recurrent NARX Neural Networks*. IEEE transactions on system, man, and cybernetics, vol. 27, no. 2, pp. 208-215. 1997.

[9] E. Diaconescu. *The use of NARX neural networks to predict chaotic time series*. Wseas Transactions on computer research. 2008.

[10] Tofallis. *A Better Measure of Relative Prediction Accuracy for Model Selection and Model Estimation*. Journal of the Operational Research Society, 66(8), pp. 1352-1362.

[11] E. Page. *Continuous inspection schemes*. Biometrika, vol. 41, no. 1/2, pp. 110–115. 1954.

[12] O A Grigg, V T Farewell, and D J Spiegelhalter. *Use of risk-adjusted CUSUM and RSPRTcharts for monitoring in medical contexts*. Statistical Methods in Medical Research. 2003.

[13] S. Mittal. *Power Management Techniques for Data Centers: A Survey*. Technical Report. 2014.

[14] P. Delforge, J. Whitney. *Scaling up energy efficiency across the Data Center Industry: evaluating Key Drivers and Barriers*[Issue paper]. The Natural Resources Defense Council. 2014.

[15] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. *Workload analysis and demand prediction of enterprise data center applications*. IEEE 10th International Symposium on Workload Characterization, pp. 171-180. 2007.

[16] M. Arlitt and T. Jin. *A workload characterization study of the 1998 world cup web site*. IEEE Network, vol. 14, no. 3, pp. 30-37, 2000.

[17] DG. Feitelson, D. Tsafrir, and D. Krakov. *Experience with using the parallel workloads archive*. Journal of Parallel and Distributed Computing. 2014.

[18] C Reiss, A Tumanov, GR Ganger, RH Katz, and MA Kozuch. *Towards understanding heterogeneous clouds at scale: Google trace analysis*. Intel Science & Technology Center for Could Computing. 2012.

[19] T. Takagi and M. Sugeno. *Derivation of fuzzy control rules from human operator's control actions*. Fuzzy Inform, Knowledge Representation and Decision Analysis. 1983.

[20] P. Amani, M. Kihl, and A. Robertsson. *Narx-based multi-step ahead response time prediction for database servers*. Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on, pp. 813-818. 2011.

[21] S. H. Arbain and A. Wibowo. *Neural networks based nonlinear time series regression for water level forecasting of Dungun river*. Journal of Computer Science, pp. 1506-1513. 2012.

[22] JJ. Moré. *The Levenberg-Marquardt algorithm: Implementation and theory*. Springer. 1978.

[23] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan. *Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers*. Services Computing (SCC), 2010 IEEE International Conference on, pp. 514-521. 2010.

[24] J. Yang , C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, and J. Chen. *A cost-aware auto-scaling approach using the workload prediction in service clouds*. Information Systems Frontiers, Volume 16, Issue1, pp. 7-18. 2013.

[25] N. Roy, A. Dubey, and A. Gokhale. *Efficient autoscaling in the cloud using predictive models for workload forecasting*. In Proceedings of the 2011 IEEE 4th international conference on cloud computing pp. 500-507. 2001.

[26] M. Rasheduzzaman, MA. Islam, T. Islam, T. Hossain, and RM Rahman. *Study of Different Forecasting Models On Google Cluster Trace*. Computer and Information Technology (ICCIT), 2013 16th International Conference on, pp.414-419. 2014.

[27] M. Alam, KA. Shakil, S. Sethi. *Analysis and Clustering of Workload in Google Cluster Trace based on Resource Usage*. Journal of Parallel and Distributed. 2015.