Accepted Manuscript

Modelling and prediction of resources and services state evolvement for efficient runtime adaptations

Dimosthenis Kyriazis

PII: S0167-739X(18)31317-7
DOI: https://doi.org/10.1016/j.future.2018.09.035
Reference: FUTURE 4470

To appear in: *Future Generation Computer Systems*

Received date : 29 May 2018
Revised date : 23 July 2018
Accepted date : 12 September 2018

Please cite this article as: D. Kyriazis, Modelling and prediction of resources and services state evolvement for efficient runtime adaptations, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.09.035

Modelling and prediction of resources and services state evolvement for efficient runtime adaptations

Dimosthenis Kyriazis

Dept. of Digital Systems, University of Piraeus,

80, Karaoli & Dimitriou Str, 18534 Piraeus, Greece

email: dimos@unipi.gr

Abstract: Regardless of their implementation aspects and distribution elements, i.e. centralized or distributed, service-based environments such as cloud computing and edge/fog infrastructures, enable the provisioning of services addressing a wide range of application domains. The key requirement for users and consumers of such services refers to the corresponding levels of quality, which is affected both by the real-world dynamics - given the non-deterministic use of services, and by the underlying resources state - given the typically virtualized sharing nature of the resources. In this paper, an approach is presented that aims at estimating the evolvement of services and resources state in order to provide insights for runtime adaptations, as required to ensure services quality. The state refers to different metrics / parameters such as memory, number of users, throughput, etc, and can be extended and applied to different ones. The proposed approach exploits polynomial regression and prediction to identify the aforementioned state evolvement by mapping the two first monitoring data points for each metric / parameter to the corresponding function that depicts their evolvement. The latter provides added value in different cases, including among others the adaptation of monitoring time intervals, the estimation of the potential breach of quality thresholds, and the prediction of the time for runtime adaptations and scaling decisions. The effectiveness of the implemented approach is demonstrated and evaluated through a set of different scenarios.

Keywords: service oriented infrastructures; cloud computing; quality of service; monitoring; runtime adaptations; polynomial regression; polynomial prediction

1. Introduction

The continuous-changing landscape of the services provisioning space as well as the realization of advanced communication and networking paradigms - such as SDN/NFV and 5G, drives the emergence of new holistic environments. While these holistic environments integrate the aforementioned advanced communication paradigms, they also exploit computing infrastructures such as cloud, edge and fog, in order to provide added-value services to a wide set of users and consumers. Currently, service providers go beyond the SPI model (Software, Platform, and Infrastructure as a Service) [1], aiming at the delivery of different assets as a service. Furthermore, new patterns of mobility and the wide deployment of Internet of Things (IoT) environments contribute towards the compilation of new services and products through edge / fog computing models. In this context, cloud infrastructures actually reflect a baseline utility for any IT-based service delivery environment. As a result, the infrastructure space of emerging service-based environments includes different computing, storage and communication elements that serve the needs of applications and users in a holistic way, expressed as "complete computing" [2].

On the applications space, the aforementioned communication and IoT environments act as enablers for the provision of added-value services in combination with data management and computing "backbone" infrastructures. All in one, they allow the realization and offerings of composite applications that consist of application service components (i.e. micro-services) – often of different nature. These application service components provide specific functionality, contributing to the overall

application's one, and may be offered by different providers. It has to be noted that this composite application paradigm is also applied across the service stack since besides service components (on the application layer), different infrastructure services (e.g., networking or storage resources) may also be offered within the overall applications by exploiting the concept of containerization [3].

In this context, service provisioning is shifted from the concept of a delivery of services "on top" of resources to a "blended" service and infrastructure elements delivery: composite application graphs include application components, data management and processing mechanisms (e.g. data cleaning or aggregation), and communication elements (e.g. virtualized network functions). This creates a number of interdependencies among these elements in the service lifecycle and raises several challenges. One of these refers to availability and its impact in consumers and businesses – as highlighted an incident to top three cloud provides for 3-6 days would result in ground-up loss between $6.9 and $14.7 billion [4]. Another challenge refers to dependability - as also one of the main Future Internet Architecture Design Principles [5], which has been characterized as fundamental for the user needs. In general, the requirement for services quality in these application and infrastructure blended environments is clear and has been identified as a main requirement in both the edge [6] and the cloud computing [7] domains.

To address the quality requirement, several novel approaches have been proposed and realized, ranging from the analysis of historical data for proactive decisions to runtime adaptations based on real-time analytics and control loops in different levels of the service-oriented environments. The fundamental block in all these approaches is the monitoring framework collecting, storing and processing monitoring data to drive runtime adaptation decisions for the provisioning of quality guarantees [8]. However and given the scale, structure and complexity in IoT, (mobile) edge, cloud and application environments, monitoring frameworks need to be efficient in terms of the corresponding monitoring time intervals / periods in order to minimize their footprint, performance overhead and cost on the overall environment. For example, an average of 18% of the total running cost for cloud computing is consumed by monitoring tools, as in the case of Amazon EC2 CloudWatch [9]. What is more, the time at which runtime adaptations take place is critical given the non-deterministic usage of the applications (reflecting usage behaviours) in combination with the fact that these applications have specific strict quality requirements. Therefore, there is a need to "know when to act" to ensure the provision of resources in time, while minimizing the cases of unrequired overprovisioning. To this end, this paper introduces an approach based on polynomial regression and prediction to model the evolvement of the services and resources metrics in order to estimate the time at which these services will reach a specific threshold. This threshold may refer to a service level objective, a parameter in a service level agreement, a resource threshold set by an infrastructure provider, an application-specific threshold such as number of sessions or users, etc. The goal of the estimation is dual: firstly to provide insights on when to act and trigger runtime adaptations (e.g. scaling decisions), and secondly to adjust monitoring time intervals accordingly given that the pace and the direction at which a metric evolves can be estimated.

The remainder of the paper is structured as follows. Section 2 presents related work in the field of adaptation of monitoring time intervals as a means for the estimation of the time for runtime decisions towards the provision of service quality guarantees. Section 3 introduces the overall architecture of the proposed approach, while Section 4 presents the incorporated algorithms to model the evolvement of different application and resources metrics / parameters and estimate the triggering time for runtime adaptations. The evaluation results for different experimentation scenarios are cited in Section 5, while Section 6 concludes with a discussion on future research and potentials for the current study.

2. Related Work

In the infrastructure domain and more specifically related to cloud computing services, monitoring for managing service clouds is essential for the health of cloud systems and is significant for both providers and consumers [10], [11], [12], [13]. Generally, monitoring is a key tool for both managing software and hardware resources, and for offering information for those resources and the consumers' hosted applications. On the other hand, in the cloud computing field, monitoring consists of two types: high-level and low-level monitoring [14]. The high level monitoring focuses on the virtual environment status. In addition, the low-level monitoring deals with information collected for the status of the physical infrastructure. Based on these types, three kinds of monitoring function can be provided: (i) monitoring of physical resources, (ii) monitoring of virtual resources, and (iii) monitoring of application services. A cloud monitoring system is a self-adjusting and typically multi-threaded system supporting monitoring functionalities [15]. It comprehensively monitors pre-identified instances/resources on the cloud for abnormalities. On detecting an abnormal behaviour, the monitoring framework attempts to auto-repair this instance/resource if the corresponding monitor has a tagged auto-heal action [15]. Recently, many vendors associated with cloud services tend to introduce cloud commercial and open source platforms, and monitoring systems for cloud monitoring services in order to take advantage of keeping their resources and applications operating at peak efficiency, to detect variations in resource and application performance, to account the Service Level Agreement (SLA) violations of certain quality parameters, and to track the leave and join operations of cloud resources due to failures and other dynamic configuration changes. The most spread platforms include commercial (e.g. CloudWatch [9], AzureWatch [16], CloudStatus [17], Nimsoft Monitor [18], LogicMonitor [19]) and open source ones (e.g. Ganglia [20], MonaLisa [21], GridICE [22], Nagios [23], PCMONS [24], DARGOS [25], Hyperic-HQ [26], Sensu [27], etc). These monitoring frameworks consist of a set of agents that collect data from the entities to be monitored and are typically collocated with them (e.g. in the same virtual machine). The (centralized) monitor collects the data from all agents and aggregates them producing the monitoring output.

Recent interesting approaches move beyond the ones described previously by aim at estimating and adapting the monitoring time intervals according to different metrics. Such an approach is presented in [28], on which authors perform correlation analysis for different monitoring metrics in order to monitor the most important ones and thus minimize the monitoring footprint, while also predicting the possibility of faults through Principal Component Analysis in order to adapt accordingly the monitoring intervals. Moreover, with the recent proliferation of cloud service brokerage, monitoring federated service clouds have been the most noteworthy and influential. When service clouds are federated to accept other's workload there needs to be a consideration of how monitoring will behave in the presence of the federated cloud infrastructures. In this context, an approach for multi-cloud monitoring by adapting the corresponding cross-cloud intervals has been proposed [29], while a similar approach has been proposed in [30] with however the drawback of the usage of the same monitoring tool across all federated cloud systems. The Cloud Adoption Toolkit focuses on cost prediction and allows modelling the application requirements over time and predicting migration and future costs across multiple cloud providers [31]. In the same context, i.e. the adaptation of the infrastructure by also considering the corresponding cloud services, authors in [32] propose a mechanism for estimating the virtualized resource requirements of the application by defining the minimum amount of required resources to avoid performance degradation of the running services. For applications also exploiting edge computing environments, an adaptive mechanism is discussed in [33]. Authors propose a distributed agent-based mechanism that utilizes edge nodes to perform aggregation and runtime decision taking on an edge level towards effective and timely decision

making. Efficient resource optimization and runtime adaptations following an agent-based monitoring system has been proposed in [34]. The system also includes an alarm feature, which is triggered if a value breaches a threshold, which is however pre-defined. A quite interesting approach for autonomous monitoring and management, namely iOverbook, has been introduced in [35]. The proposed framework enable online overbooking based on a feed-forward neural network model by taking into account historical information regarding the usage of resources in order to forecast the mean hourly resource usage one step ahead.

The differences between the existing approaches and the research outcomes presented in this section and our proposed approach are summarized in the following: (i) adaptable monitoring time intervals for approaches that work with fixed ones, given the highly dynamic environments, (ii) identification of time intervals – or more precisely of the time at which adaptations need to take place – based on runtime information and not based on the analysis of historical data that can only occur offline, (iii) a distributed architecture with distributed decision points that are all self-adaptable towards more autonomous systems as required in the case of edge / fog computing, (iv) applicability to both resource- and service- level metrics since the proposed approach is agnostic to the parameters being analysed.

3. Overview of the proposed approach

The approach introduced in this paper aims at modelling and estimating the evolvement of the services and resources in terms of their state – i.e. different metrics and parameters. Based on this modelling, it is feasible to identify the time as an input to monitoring mechanisms (i.e. monitoring time interval) as well as the time at which runtime adaptations need to take place in order to ensure quality of service. The overall approach has been developed as a service, namely "Runtime analyser and estimator", incorporating the developed algorithms to enable estimation of the services and applications state evolvement. The developed services has been evaluated (corresponding results are cited in Section 5) in a cloud environment as shown in Figure 1. The key aspects of the proposed approach are the following:

- The Runtime analyser and estimator follows an hierarchical architectural approach: instances of the service are deployed on a cluster-level, while another instance is deployed on a multi-cluster (e.g. cloud) level. The latter enables adaptations regarding monitoring time intervals on a per-cluster case given that different applications and resources may require different time intervals given their state [36].
- While monitoring time intervals are updated on a cluster level, adaptation actions are triggered on the cloud level. Even though these adaptations may only refer to a specific cluster, they are triggered on a cloud level given the need to ensure that the overall state of the cloud environment is considered before for example triggering scaling decisions for a specific cluster.
- The Runtime analyser and estimator is not coupled to a specific monitoring framework but can be utilized as an external "add-on" mechanism for optimization of runtime decisions and of monitoring time intervals.

Based on the above, the proposed deployment is depicted in Figure 1, with the main information flows summarized as follows:

- The underlying resource management system provides the corresponding resources to different applications. In this case, Mesossphere DC/OS [37] has been selected in order to abstract resources and enable the provision of containers on a per-cluster case.

- The monitoring framework obtains information from different containers based on the agents that have been deployed in the containers. Prometheus [38] has been chosen as a monitoring framework. Prometheus gathers monitoring parameters at specified intervals, shows the results, and triggers alerts based on rule expressions. Moreover, it follows an hierarchical architectural approach, with Prometheus servers and Push gateways being deployed in different levels of the hierarchy. In this case, these have been deployed in each cluster and enable the collection of information on a per-cluster case by obtaining the monitoring data from the Exporters deployed in each container. Exporters are probes and can be either provided by the application developer or obtained by the open source community supporting Prometheus (additional information on the Exporters used in this case is cited in Section 5).
- Based on the collected monitoring data from the Prometheus server of each cluster, the Runtime analyser and estimator performs modelling for the obtained parameters / metrics and according to this modelling it esimates the monitoring time interval. The latter is provided to the Prometheus server (as a scrape interval) in order to update the monitoring intervals for the containers of the cluster.
- On the next level of hierarchy (i.e. multi-cluster level), the Prometheus server obtains information from the deployed servers in different clusters. This information is modelled by the Runtime analyser and estimator, which in turn triggers a scheduler that performs scalability decisions. In this case, Fenzo scheduler [39] has been deployed on top of Mesosphere. However, Mesosphere allows the deployment of different or custom schedulers.

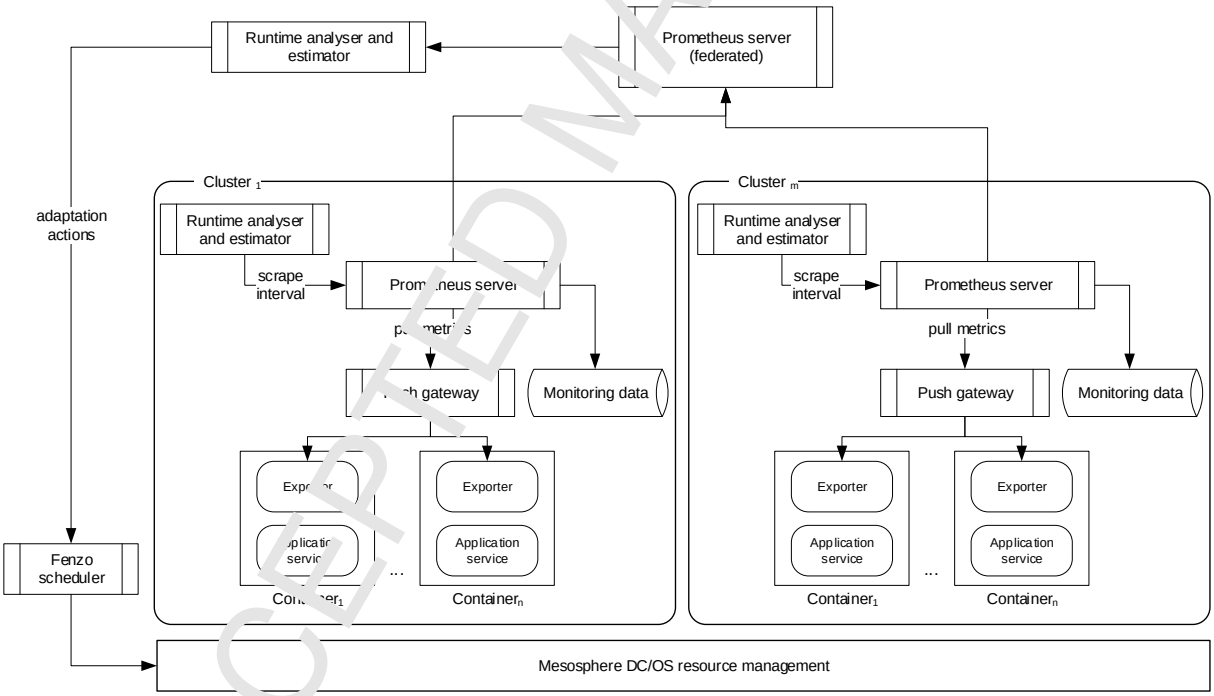


Figure 1. Approach overview

4. Triggering time for runtime adaptations based on resources and services evolvement

In this section, the algorithms that have been implemented in the Runtime analyser and estimator service are presented. These algorithms aim both at modelling the evolvement of the state of the

resources and services and at estimating the time for triggering runtime adaptations and for adjusting monitoring time intervals.

4.1 Modelling of the resources and services evolvement

By denoting as S_i the *i*-th application service deployed in a container, each service is characterized by a list of *M* service attributes $S_i = [sa_1 sa_2 \Lambda\, sa_M]^T$. In a similar way, R_i is the *i*-th resource (e.g. container) deployed in a cluster. Each resource is characterized by a list of *N* resource attributes $R_i = [ra_1 ra_2 \Lambda\, ra_N]^T$. The *M* attributes correspond to the quality parameters of the application service (e.g. users, sessions, response time, etc.) and the *N* attributes to the quality parameters of the resource (e.g. CPU, memory, throughput, etc.). It should be noted that the proposed approach is agnostic to the actual metrics / parameters, meaning that given that it focuses on the evolvement of the attributes values, these attributes could refer to any metric / parameter (e.g. memory or CPU). This way, an application service or a resource are represented as a vector. In the context of this paper, these attributes are considered to be numerical of real values and will be the ones collected by the monitoring mechanism.

Given that all service and resource parameters change in time, the goal of the modelling phase is to identify a function that corresponds to the actual changes of the attributes values in time. This function is denoted as $p(t)$, where *t* denotes the time. The main underlying concept of the proposed modelling approach is to utilize interpolants in order to perform polynomial fitting and as a result approximate the evolution of the parameters (i.e. service and resource state) in time, thus identifying the $p(t)$ function. Considering that there will be *n+1* points obtained by the monitoring mechanism for a given attribute *a* (e.g. memory usage), these refer to the following:

$$(t_o, a_o), (t_1, a_1), (t_2, a_2), \ldots, (t_{n-1}, a_{n-1}), (t_n, a_n)$$

Based on these set of points, there will be a polynomial $p(t)$, with a maximum degree of *n*, for which:

$$p(t_0) = a_0, p(t_1) = a_1, \ldots, p(t_{n-1}) = a_{n-1}, p(t_n) = a_n$$

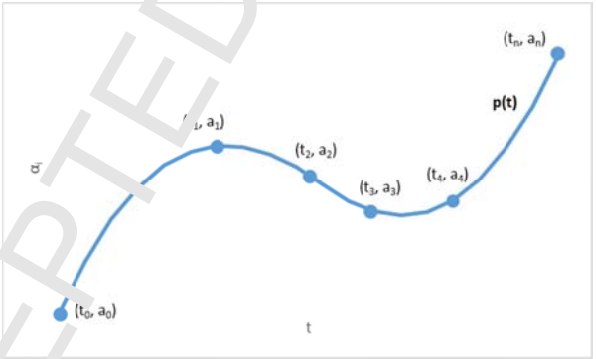


Figure 2. Interpolant p(t)

The key aspect is that these points may have been produced by a function $f(t)$ that represents the actual evolvement of an attribute *a* in time. Thus, it will be:

$$f(t_i) = p(t_i), \qquad i = 0, 1, 2, 3, \ldots n$$

Therefore, by collecting a set of monitoring data in different time frames, it is feasible to identify and approximate a function that represents how a specific service or resource will behave in time in terms of different attributes evolvement, such as CPU, memory, throughput, etc.

Furthermore, instead of linear approximation, the implemented modelling approach exploits polynomial approximation up to the 5th degree. Higher degrees actually refer to a service or resource

for which the changes are such that even if modelled, the overall behaviour will trigger continuous adaptations and thus any modelling or estimation does not add value to monitoring time intervals adaptation or triggering time of corrective actions. In order to perform interpolation, the set of monitoring data is collected following the initial configuration of the monitoring framework (e.g. per 1 minute). Based on the above and since the proposed modelling approach addresses up to 5th degree polynomials, the required number of points to perform interpolation would be polynomial degree + 1,thus in total 6 points (i.e. monitoring data) are required.

Through the proposed modelling approach of the Runtime analyser and estimator component, a number of potential changes in the attributes of the resources and services can be addressed - as also depicted in the following figure (Figure 3).

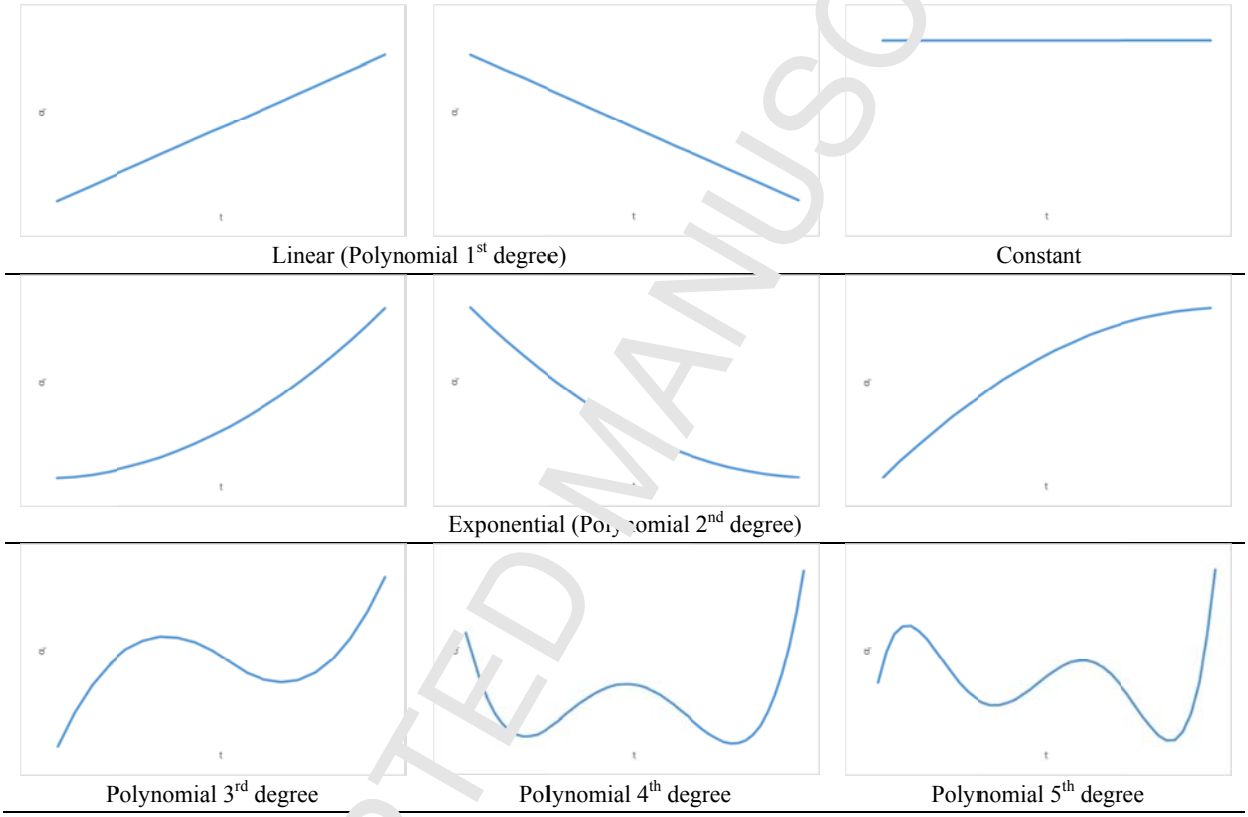


Figure 3. Potential changes of the attributes of resources and services in time

As described previously, the polynomial $p(t)$ crosses all given points (collected monitoring data). However, it might be possible to develop the corresponding prediction by not crossing all given points, a method known as least squares method. The evaluation of this method has shown higher error rates (cited in Section 5) for polynomials of 3rd degree and higher and therefore is not considered in the approach presented in this paper.

4.2 Estimation and optimization of the triggering time for runtime adaptations

The modelling outcome - reflected in the polynomial $p(t)$, is associated with an error, expressed through the criterion Chebyshev:

$$\varepsilon = \max_{t \in [a,b]} |f(t) - p(t)|$$

Based on the aforementioned criterion, in the case of non-polynomial, there could be a function that represents the data (in our case the change in the parameters / metrics of a service or resource) in a

better way – i.e. with a smaller (or even zero) error. Such a case could be for example an exponential function (which is typical in the case of cloud applications and services). Thus, the proposed estimator mechanism should also account for such cases before concluding to the actual function $p(t)$ that represents the evolvement of the state of services and resources. This is considered during the estimation phase as described below.

The next step of the mechanism is to estimate the time at which a specific threshold might be reached in order to trigger the corresponding runtime adaptations on the infrastructure level (e.g. scaling). This threshold is denoted as th.

Given that polynomials of 5th degree are considered in the proposed approach, it is not feasible to solve the equation regarding the variable *t*. However and given that the services and resources behaviour is non-deterministic and usually reflected in such polynomials, there is the need to estimate the time t_x at which a threshold (e.g. memory usage) is reached. Thus, it is required to estimate the:

t_x for which $p(t_x) = k$, where k is the threshold value.

To estimate the aforementioned time t_x, the following algorithm has been implemented in the Runtime analyser and estimator:

Optimization and estimation of the triggering time for runtime adaptations

1. Collect 6 monitoring data points (to enable polynomials of 5th degree to be considered) for a given attribute a (e.g. memory usage) in the initially configured intervals of the Prometheus server (every second):
$$(t_o, a_o), (t_1, a_1), (t_2, a_2), (t_3, a_3), (t_4, a_4), (t_5, a_5)$$
2. Model the evolvement of a based on the polynomial regression and prediction by calculating the function $p(t)$.
3. Utilize the function $p(t_x)$ for x = 7, 8, 9, … 16. These refer to subsequent points of the initial 6 collected monitoring points as estimates (i.e. perform 10 predictions of the corresponding attribute a through the $p(t)$.
4. Store all the results in an array and perform binary search for the threshold th. If the value being searched doesn't match a specific one, consider as the triggering time the smallest closest neighbor.
5. While $th > p(t_{15})$ return to Step 3 and perform 10 additional predictions since the threshold for a specific attribute has not yet been predicted from the identified polynomial.
6. Store the additional results from Step 5 in the array and perform binary search for the threshold th in order to obtain the estimation / prediction on the time at which the threshold will be reached.
7. Obtain the corresponding time t_x.
8. Perform polynomial fitting for an exponential function based on the 6 data points collected in Step #1 and perform the number of predictions required as known by Step #5 (through a global variable).
9. In time t_{x-1} compute the error ε both for the $p(t)$ and for the exponential function. According to the smaller error, disregard one of the 2 functions for future cases.
10. For each new monitoring result / data, restart the overall process.

Table 1: Triggering time estimation algorithm

Based on the algorithm above, the scaling decisions are triggered through the Fenzo scheduler presented in Section 3.

5. Evaluation

5.1 Experiment setup

The experiments used to evaluate the proposed approach were performed in a real-world setting in terms of the utilized frameworks and tools. An infrastructure consisting of eight (8) nodes has been exploited as a testbed for the validation of the mechanism results. The nodes have the following characteristics: 16GB RAM, Intel i7-4790 @ 3.60 GHz x 8 CPU Cores, 2TB Storage. Mesosphere DC/OS 1.11 has been deployed and in total 24 containers have been made available for experimentation, presenting a small scale representative cloud environment.

Regarding the application services that have been used, WikiBench [40] has been selected as the benchmark since it addresses a wide range of typical services (including application servers, load balancers, and databases) and thus different parameters can be monitored, modelled and estimated. It should be noted that the purpose of this study is not to stress-test application servers or databases, but to demonstrate through a benchmark the non-deterministic behaviour of services (and as a result resources). To this end, WikiBench is an optimum benchmark given that the load on the services (e.g. web servers due to number of concurrent users) and on the resources (e.g. CPU or memory) follows the usage patterns of visitors, which is non-deterministic.

To realize the evaluation environment and exploit Wikibench for load generation, a load balancer has been deployed, namely HAPProxy [41]. Moreover, Apache has been used as an application server and MySQL as the database. In terms of monitoring, as already described in Section 3, Prometheus has been selected and deployed, as well as the corresponding exporters (acting as probes for the different services) being available from the Prometheus community [42]. These exporters are the following: Apache exporter, HAProxy exporter, MySQL exporter. Furthermore, the initial scrape interval in the corresponding configuration file of Prometheus has been updated to 1sec (the default is 1min). The data that have been collected during the experimentation phase are summarized in the following table.

Metric	Type	Description
cpu_usage	float	CPU usage percentage
memory_usage	float	Memory usage percentage
io_bytes_read	bytes/second	Number of bytes read from block devices
io_bytes_write	bytes/second	Number of bytes written to block devices
rx_packets	packets/second	Number of received packets
rx_bytes	bytes/second	Number of received bytes
tx_packets	packets/second	Number of sent packets
tx_bytes	bytes/second	Number of sent bytes

Table 2: Published service attributes / QoS parameters

5.2 Evaluation results

A number of experiments has been performed to evaluate different aspects of the proposed approach. The main aspects under investigation include the following: (i) Evaluation of the choice regarding polynomial regression and prediction comparing to the least square method, (ii) Evaluation of potential change from polynomial approximation to exponential approximation as proposed by the algorithm presented in Section 4.2, (iii) Evaluation of the numerical-driven approach to perform search on the data space in order to identify the time for triggering corrective actions, (iv) Evaluation of the actual predictions comparing to the ground truth and as a result the actual values.

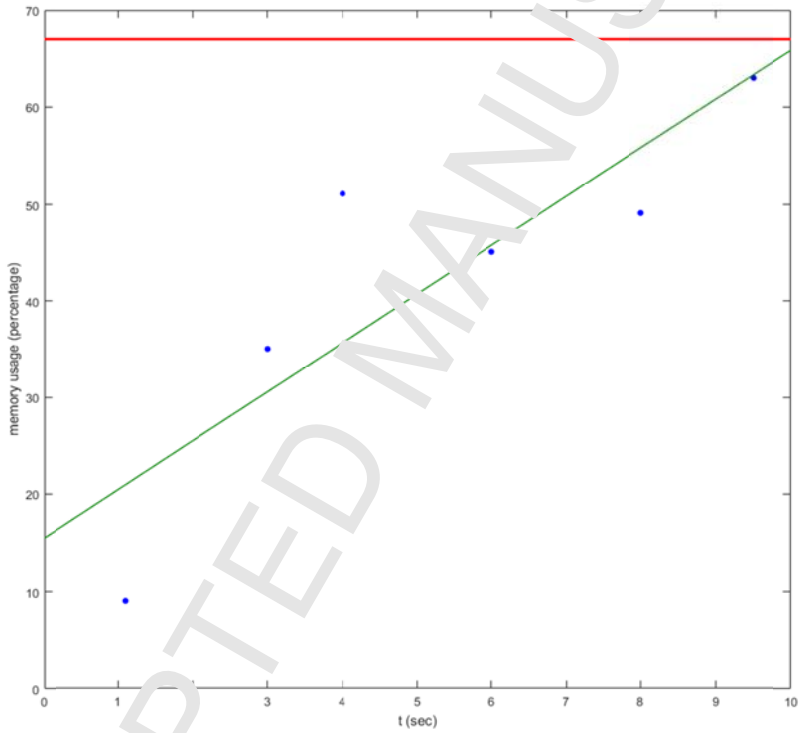
An additional aspect that was taken into consideration refers to the evaluation of the presented mechanism both for service-related metrics / parameters and for resource-related ones. To ensure that both are being addressed, the experiments referring to aspects under investigation (i), (iii), (iv) above

have been conducted with a resource-related metric: the memory usage. The service-related metric has been addressed by the aspect (ii) under investigation.

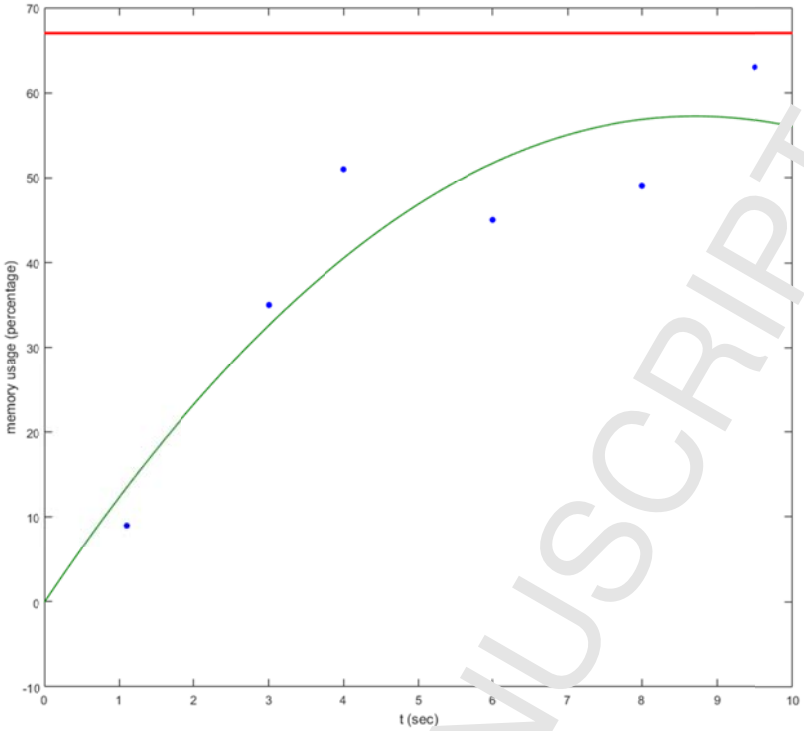
The following paragraphs showcase the results for all identified aspects under investigation.

Evaluation of the choice regarding polynomial regression and prediction comparing to the least square method

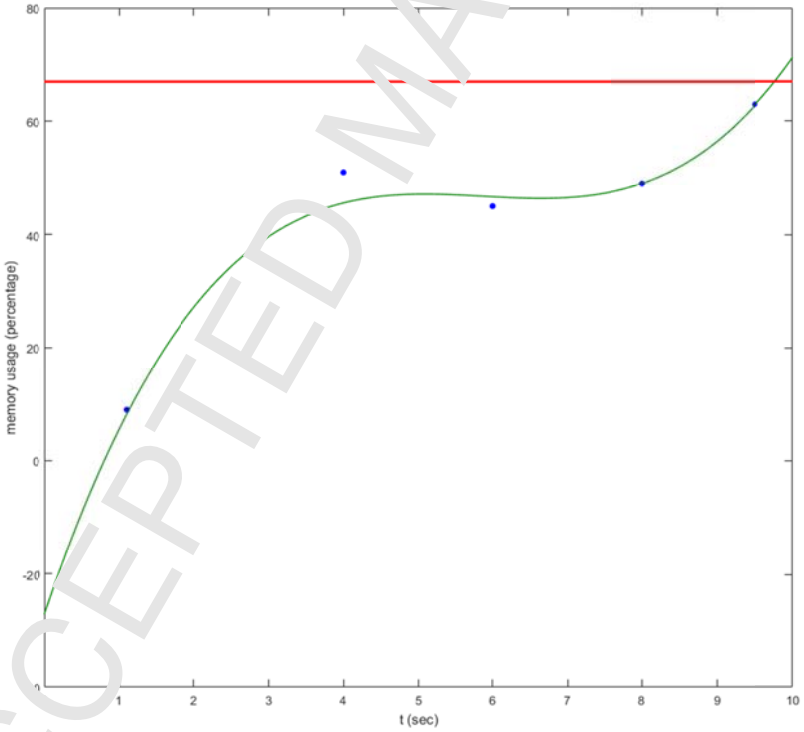
To perform the specific evaluation 6 data points – monitoring data values – have been collected and the corresponding polynomial regression took place. The regression was performed for different polynomial degrees to showcase the outcomes when not all points are considered (least square method) and when all points are considered (5th degree polynomial). The results are depicted in Figure 4 and Figure 5 respectively. It should also be noted that a specific threshold has also been considered (e.g. obtained through a service level agreement) in order to showcase whether the different approaches have an impact on the runtime actions with respect to that threshold. The threshold has been set to 67% memory usage for the resource being modelled.



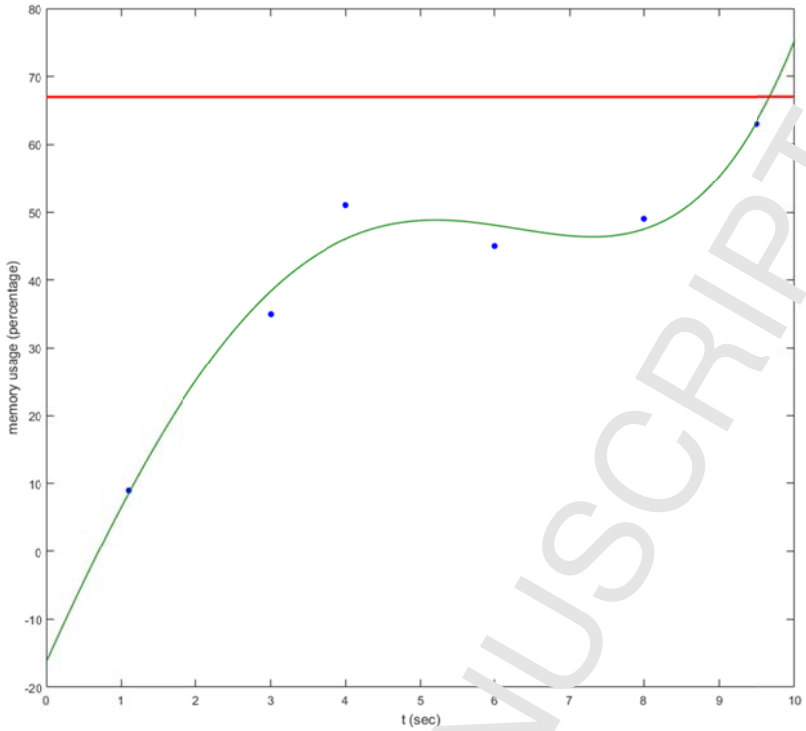
(a) Polynomial (1st degree) approximation



(b) Polynomial (2nd degree) approximation



(c) Polynomial (3rd degree) approximation



(d) Polynomial (4th degree) approximation

Figure 4. Polynomial approximation with the least square method

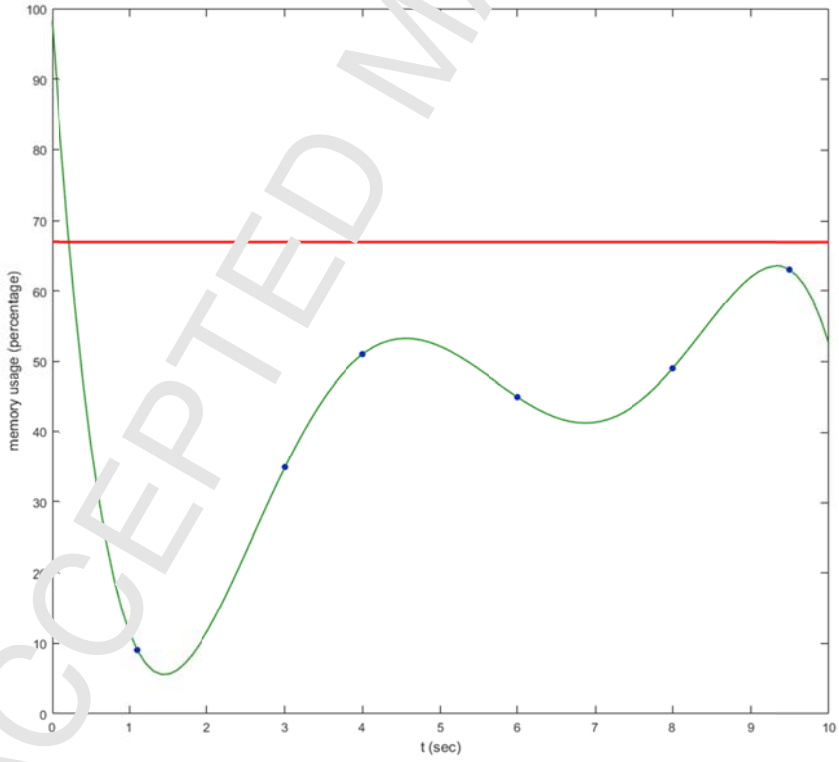


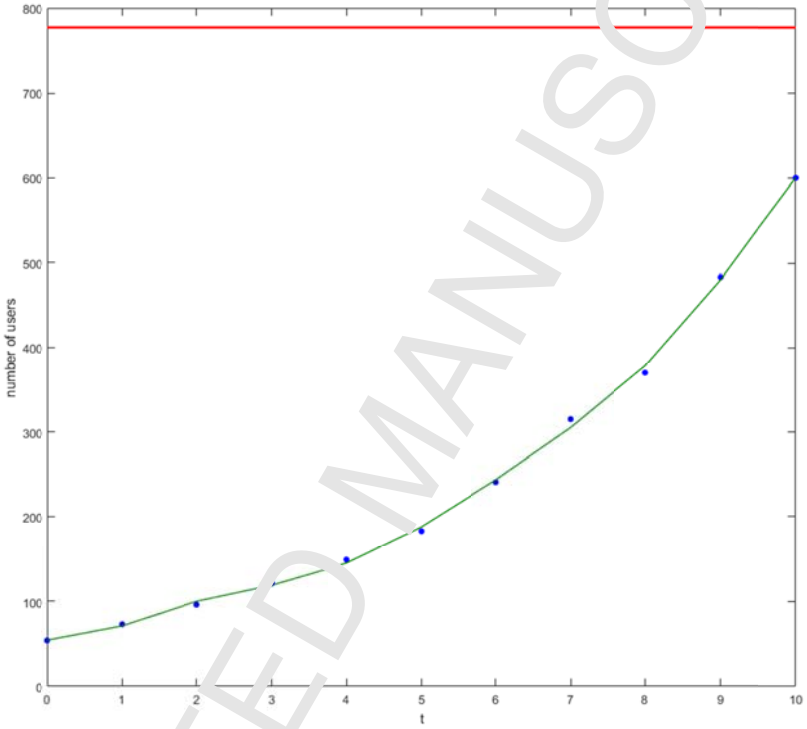
Figure 5. Polynomial (5th degree) approximation with interpolant

As depicted in the figures above, the least squares method would lead to overprovisioning of resources given that the estimations either already cross the specified threshold or will lead to it within the next 2 seconds. The only case of least squares that does not reach the threshold is the 2nd degree

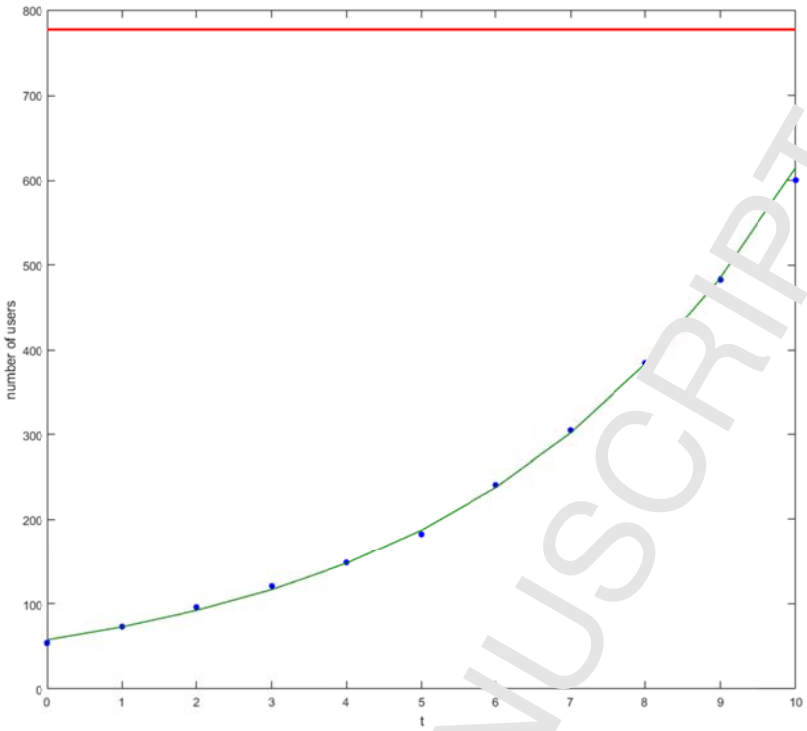
approximation. However, the error in that case is very high (~10%) and thus cannot be considered as a trustworthy estimation. The 5th degree polynomial performs quite efficiently regarding the estimation and prediction (error rate < 2%) and thus it has been selected for experimentation with additional metrics and cases.

Evaluation of potential change from polynomial approximation to exponential approximation

The aim of this experimentation is to evaluate whether the approach followed in the algorithm presented in Section 5.2 to utilize exponential approximation instead of polynomial adds value to the overall prediction. To this end, an experiment has been performed considering this time as a metric the number of concurrent users, and a threshold set to 780 users. The exponential approximation outperforms the polynomial one as shown in the following figure.



(a) Polynomial approximation



(b) Exponential approximation

Figure 6. Polynomial approximation vs exponential approximation

Evaluation of the numerical-driven approach to perform search on the data space in order to identify the time for triggering corrective actions

The next experiment aims at evaluating the aspect of the algorithm to search on the data space in order to conclude to the time at which corrective actions to ensure quality of service need to be taken. Given that for >3rd degree polynomials there is no unique solution, binary search has been selected as a means to search on the data space. With the initial threshold (set on 67% of memory usage) and following the polynomial regression (performed until t_6), a set of 10 estimations is produced and the algorithm searches within this space. Thereafter, the initial experiment has been updated to include a second threshold - set to 90% of memory usage. The reason for the latter is to evaluate whether the approach to perform every time a set of 10 sequential predictions and search in that space yields acceptable results. The prediction and the aforementioned thresholds are cited in the next figure.

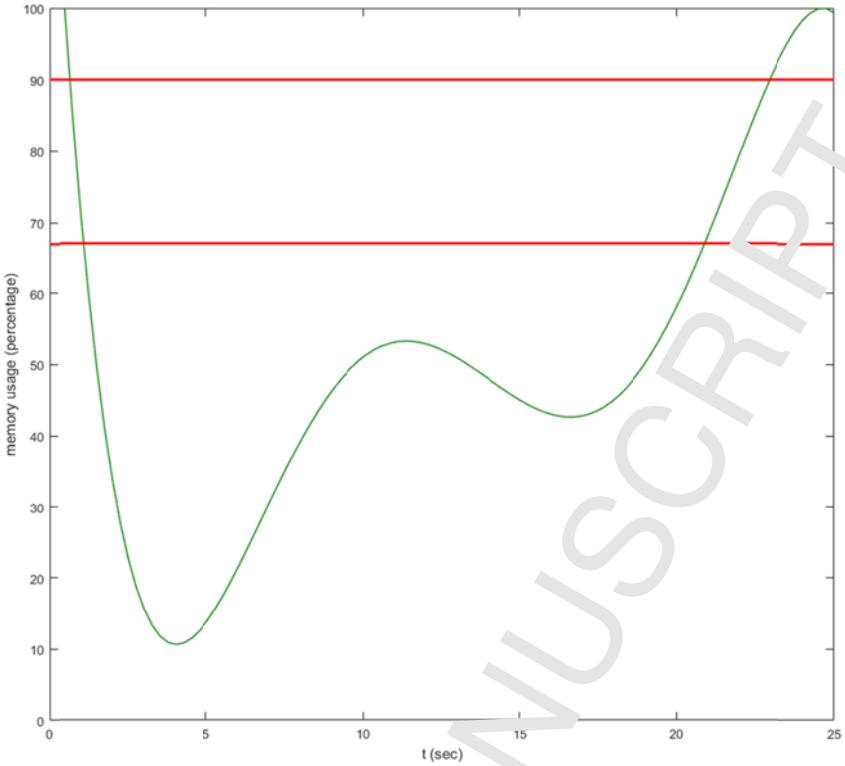


Figure 7. Polynomial (5th degree) regression and prediction for t_0 - t_{26}

Based on the above and as depicted in the following table (Table 3), t_{15} is chosen as the time to trigger corrective actions (i.e. before reaching the threshold).

Time	Estimation	Threshold
t_7	51.00	67.00
t_8	53.21	67.00
t_9	52.01	67.00
t_{10}	46.69	67.00
t_{11}	45.00	67.00
t_{12}	42.75	67.00
t_{13}	43.57	67.00
t_{14}	48.56	67.00
t_{15}	58.01	67.00
t_{16}	71.03	67.00

Table 3: Estimated memory utilization based on the interpolant for t_7 - t_{16}

In the case of the updated threshold, given that it is higher than the current set of estimations – shown in Table 2, an additional set of estimations is compiled. In this case, the threshold value is lower than the highest estimated value and thus binary search is performed within the data space. As shown in the next table (Table 4), corrective actions will be triggered in t_{22}.

Time	Estimation	Threshold
t_{17}	73.89	90.00
t_{18}	76.77	90.00

t_{19}	79.67	90.00
t_{20}	82.53	90.00
t_{21}	85.32	90.00
t_{22}	88.01	90.00
t_{23}	90.55	90.00
t_{24}	92.89	90.00
t_{25}	95.00	90.00
t_{26}	96.81	90.00

Table 4: Estimated memory utilization based on the interpolant for t_{17} - t_{26}

Evaluation of the actual predictions comparing to the ground truth

For the previous experiment the monitoring data have been collected in order to analyse the error rate of the estimations of the polynomial regression and prediction comparing to the ground truth measurements. As shown in the figure below the error rate is less than 4% (absolute value), while the average error rate is ~ 1.5%.

Figure 8. Error percentage of polynomial prediction comparing to ground truth measurements

6. Conclusions

Service provisioning on top of IT infrastructures including cloud, edge / fog computing and Internet of Things environments leads to value creation for different entities in the ecosystem [43]. However, this value is amplified only in the case of services that are provided with specific quality of service guarantees. In this context, this paper introduced a complete solution that addresses the case of decision taking regarding adaptations on the infrastructure level, necessary to ensure quality of service. A mechanism has been developed and integrated with a well-established monitoring framework (i.e. Prometheus), enabling modelling and estimation of the evolution of different parameters. These parameters could be application and / or resource metrics that are collected by monitoring frameworks. The mechanism presented in this paper utilizes polynomial regression and prediction in order to model and predict the aforementioned evolution of parameters in time. The latter also contributes towards minimizing the overhead of monitoring, since by predicting the evolution of an attribute's value and estimating the time at which it will reach a specific value (e.g. a threshold) the monitoring time intervals are adapted accordingly. For example in the case of a constant value the

monitoring intervals will be increased (i.e. non-required monitoring will be avoided), while in the contrary in the case of an exponential increase the intervals will be decreased since a threshold could potential be reached in a short time window. Furthermore, according to the prediction, it is feasible to identify when a threshold will be reached and thus take proactive actions. The threshold can be set by cloud administrators or obtained by service level agreements. Key aspect in the proposed approach is that polynomial fitting provides a high degree of accuracy and minimizes cases of potential overprovisioning of resources in contrast with the least square method. Moreover, the presented mechanism utilizes polynomial approximation up to the 5th degree (and not linear approximation), which represents a broad change in parameters values and deals with potential spikes and bursts. An initial prototype has been finalized and is fully functional in a real-world setting on top of Mesosphere, enabling the management of containers and services in an optimum way.

Notwithstanding, it is within future plans to further improve the proposed approach. To this end, Netwon polynomial will be utilized as a basis in order to enable new points to be added during the polynomial fitting. The final outcome of the compilation of the polynomial will not change (comparing to the approach presented in this paper), however it is more efficient since not all points need to be re-calculated when a new point is considered for the compilation of the polynomial. Furthermore, the approach will be enhanced to include weight factors that will represent the importance of services (in the cases of composite applications consisting of several services), as well as the importance of different parameters / metrics compared to others (e.g. CPU vs memory). The latter will provide accurate estimations for runtime adaptations considering the overall application as well as the overall underlying infrastructure in a holistic way, while cross-services and cross-resources optimization will also be feasible. The overall predictions will also be considered during the resource allocation phase in order to address cases of resources contention in the case of sharing resources and concurrent (potentially conflicting) requests.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation computer systems, 25(6), pp. 599-616, 2009.

[2] D. Kyriazis, K. G. Jeffery, "Cloud Forward: From distributed to complete computing", Future Generation computer systems, pp. 87-88, 2018.

[3] R. Dua, A. R. Raja, D. Kakadia, "Virtualization vs containerization to support PaaS", IEEE International Conference on Cloud Engineering (IC2E), pp. 610-614, 2014.

[4] Lloyd's of London, "Cloud Down - Impacts on the US economy", https://www.lloyds.com/~/media/files/news-and-insight/risk-insight/2018/cloud-down/aircyberlloydspublic2018final.pdf

[5] Future Internet Architecture (FIArch) Group: Future Internet Design Principles. European Commission, http://ec.europa.eu/information_society/activities/foi/docs/fiarchdesignprinciples-v1.pdf, 2012.

[6] S. Shahzadi, M. Iqbal, T. Dagiuklas, Z. U. Qayyum, "Multi-access edge computing: open issues, challenges and future perspectives", Journal of Cloud Computing, 6(1), p.30, 2017.

[7] K. Jeferry, G. Kousiouris, D. Kyriazis, J. Altmann, A. Ciuffoletti, I. Maglogiannis, P. Nesi, B. Suzic, Z. Zhao, "Challenges emerging from future cloud application scenarios", Procedia Computer Science, 68, pp.227-237, 2015.

[8] A. M. Fahad, A. A. Abdulghani, M. K. Nizam, "The importance of monitoring cloud computing: An intensive review", IEEE Region 10 Conference, pp. 2858-2863, 2017.

[9] Amazon CloudWatch, http://aws.amazon.com/es/cloudwatch/

[10] S. A. Chaves, R. B. Uriarte, "Towards an architecture for monitoring private clouds." IEEE Communications Magazine, vol. 49, pp. 124-131, 2010.

[11] B. Grobauer, T. Waloschek, "Understanding cloud-computing vulnerabilities." IEEE Security and Privacy, 2010.
[12] I. Brandic, D. Music, P. Leitner, S. Dustdar, "Vieslaf framework: Enabling adaptive and versatile sla-management." Grid Economics and Business Models, pp. 60-73, 2009.
[13] J. Moses, R. Iyer, R. Illikkal, S. Srinivasan, K. Aisopos, "Shared Resource Monitoring and Throughput Optimization in Cloud-Computing Datacenters." Parallel & Distributed Processing Symposium (IPDPS), pp. 1024-1033, 2011.
[14] K. Alhamazani, et al., "An overview of the commercial cloud monitoring tools: research dimensions, design issue, and state-of-art." Computing, vol. 97, pp. 357-377, 2014.
[15] M. Anand, "Cloud Monitor: Monitoring Applications in Cloud." IEEE International Conference on Communication, Networking & Broadcasting, pp. 1-4, 2012.
[16] CloudMonix AzureWatch, http://www.cloudmonix.com/aw/
[17] Cloud Status, https://cloudstatus.eu/
[18] Nimsoft Monitor, https://www.solvitnetworks.com/nimsoft-monitor-free-infrastructure-monitoring/
[19] Logic Monitor, https://www.logicmonitor.com/
[20] M.L. Massie, B. N. Chun, D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience." Parallel Computing, vol. 30, pp. 2004, 2003.
[21] H. Newman, I. Legrand, P. Galvez, R. Voicu, C. Cirstoiu, "MonALISA: A distributed monitoring service architecture." In proceedings of CHEP03, 2003.
[22] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, "GridICE: A monitoring service for grid systems." Future Generation systems, vol. 21, pp. 559-571, 200-
[23] Nagios Monitoring Framework, http://www.nagios.org
[24] Private cloud monitoring system pcmons, https://github.com/pedrovitti/pcmons
[25] J. Povedano-Molina, J. M. Lopez-Vega, J.M. Lopez Soler, A. Corradi, L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds", Future Generation Computer Systems, 29(8), pp.2041-2056, 2013.
[26] Hyperic Application & System Moniroting, https://github.com/hyperic/hq
[27] Sensu, https://sensu.io/
[28] T. Wang, X. Jiwei, Z. Wenbo, G. Zeyu, Z. Hua, "Self-adaptive cloud monitoring with online anomaly detection", Future Generation Computer Systems, 80, pp. 89-101, 2018.
[29] M. Miglierina, E. Di Nitto, "Monitoring in a Multi-cloud Environment", Model-Driven Development and Operation of Multi-Cloud Applications, Springer, pp. 47-52, 2017.
[30] Y. Al-Hazmi, K. Campowsky, T. Magedanz, "A monitoring system for federated clouds", 1st IEEE International Conference on Cloud Networking (CLOUDNET), Paris, France, pp. 68–74, 2012.
[31] A. Khajeh□Hosseini, D. Greenwood, J.W. Smith, I. Sommerville, "The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise", Software: Practice and Experience, 42(4), pp.447-465, 2012.
[32] T. Wood, L. Cherkasova, K. Ozonat, P. Shenoy, "Profiling and modeling resource usage of virtualized applications", 9th ACM/IFIP/USENIX International Conference on Middleware, Springer, Berlin, Germany, pp. 366–387, 2008.
[33] S. Taherizadeh, A. C. Jones, I. Taylor, Z. Zhao, V. Stankovski, "Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review", Journal of Systems and Software, 136, pp. 19-38, 2018.
[34] A. Meera, S. Swamynathan, "Agent based resource monitoring system in IaaS cloud environment", 1st International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA), Elsevier, Kalyani, India, pp. 200–207, 2013.
[35] F. Caglar, A. Gokhale, "iOverbook: intelligent resource-overbooking to support soft real-time applications in the cloud", IEEE 7th international conference on Cloud computing (CLOUD), Anchorage, USA, pp. 538–545, 2014.
[36] Evolving from Machines to Services, https://grafana.com/blog/2016/01/12/evolving-from-machines-to-services
[37] Mesosphere DC/OS, https://docs.mesosphere.com//
[38] Prometheus, https://prometheus.io/

[39] A. Leung, A. Spyker, T. Bozarth, "Titus: Introducing Containers to the Netflix Cloud", Queue 15, no. 5, pp. 30, 2017.
[40] WikiBench, http://www.wikibench.eu/
[41] HAProxy Load Balancer, http://www.haproxy.org/
[42] Prometheus Exporters, https://prometheus.io/docs/instrumenting/exporters/
[43] S. Gebregiorgis, J. Altmann, "IT service platforms: Their value creation model and the impact of their level of openness on their adoption", Elsevier Procedia Computer Science, 68, 173-187 (2015).

Author Biography

Dimosthenis Kyriazis is an Assistant Professor at University of Piraeus (Department of Digital Systems). He received his diploma from the school of Electrical and Computer Engineering of the National Technical University of Athens (NTUA) in 2001 and his MSc degree in "Techno-economics" in 2004. Since 2007, he holds a PhD in the area of Service Oriented Architectures with a focus on quality aspects and workflow management. His expertise lies with service-based, distributed and heterogeneous systems, software and service engineering. Before joining University of Piraeus, he was a Senior Research Engineer at the Institute of Communication and Computer Systems (ICCS) of NTUA, having participated and coordinated several EU and National funded projects (e.g. BigDataStack, CrowdHEALTH, MATILDA, ORBIT, LeanBigData, CoherentPaaS, COSMOS, VISION Cloud, IRMOS, etc.) focusing his research on issues related to quality of service provisioning, fault tolerance, workflow management, performance modelling, deployment and management of virtualized infrastructures and platforms.

Author Photograph

Dimosthenis Kyriazis

Highlights

- Prediction of the evolution of application services and resources parameters / metrics based on polynomial approximation
- Identification of time intervals for triggering runtime adaptations on the infrastructure level towards the provision of quality guarantees
- Distributed architecture with distributed decision points evaluated in a real-world setting on top of a container-oriented resource management system and with a well-established monitoring framework