Dynamic, Behavioural-based Estimation of Resource Provisioning based on High-level Application Terms in Cloud Platforms

George Kousiouris[1], Andreas Menychtas[1], Dimosthenis Kyriazis[1,2], Spyridon Gogouvitis[1], Theodora Varvarigou[1]

[1]Dept. of Electrical and Computer Engineering, National Technical University of Athens,
9, Heroon Polytechniou Str, 15773 Athens, Greece
E-mail: {gkousiou, ameny, dimos, spyrosg, doravarv}@mail.ntua.gr

[2]Dept. of Digital Systems, University of Piraeus
80, Karaoli & Dimitriou Str, 18534 Piraeus, Greece

***Abstract*— Delivering Internet-scale services and IT-enabled capabilities as computing utilities has been made feasible through the emergence of Cloud environments. While current approaches address a number of challenges such as quality of service, live migration and fault tolerance, what is of increasing importance refers to the embedding of users' and applications' behaviour in the management processes of Clouds. The latter will allow for accurate estimation of the resource provision (for certain levels of service quality) with respect to the anticipated users and applications requirements. In this paper we present a two-level generic black-box approach for behavioural-based management across the Cloud layers (i.e., Software, Platform, Infrastructure): it provides estimates for resource attributes at a low-level by analyzing information at a high-level related to application terms (Translation level) while it predicts the anticipated user behaviour (Behavioural level). Patterns in high-level information are identified through a time series analysis, and are afterwards translated to low-level resource attributes with the use of Artificial Neural Networks. We demonstrate the added value and effectiveness of the Translation level through different application scenarios: namely FFMPEG encoding, real-time interactive e-Learning and a Wikipedia-type server. For the latter, we also validate the combined level model through a trace-driven simulation for identifying the overall error of the two-level approach.**

Keywords- Cloud computing, performance estimation, SLA translation, artificial neural networks, workload prediction

1. INTRODUCTION

Virtualization of hardware, rapid service provisioning, scalability, elasticity, accounting granularity as well as various cost allocation models enable Clouds to efficiently adapt service provisioning [1] to the dynamic demands of Internet users. In such Cloud environments that provide an infrastructure able to facilitate the needs of added-value services [3], several providers deliver services according to the Software, Platform, Infrastructure (SPI) service model [2]. The contractual relationships between these providers are reflected to the corresponding Service Level Agreements (SLAs) [20], which legally bind the actors and incorporate the information on each level / layer.

- The "Application SLA", between the Software-as-a-Service (SaaS) and the Platform-as-a-Service (PaaS) providers, includes high level application parameters. These can be broken down to workload parameters (e.g., number of users) and QoS terms (e.g., response time, frames per second during a video transmission, etc.) – also known as Key Performance Indicators (KPIs). They are defined by the Application Developers in the SaaS layer, included in the A-SLA and made concrete by the clients of the service for each individual application instance.
- The "Technical SLA" between the PaaS provider and the Infrastructure-as-a-Service (IaaS) provider, includes low-level hardware parameters (i.e., resource attributes) that should be provided in order to fulfil the requirements expressed in the Application SLA.

In this complex and multi-layer ecosystem (Figure 1), one of the most critical steps is the role of the PaaS in translating the high-level application parameters (workload and QoS, as these are expressed in the Application SLA) to resource level attributes (as these are needed to be expressed in the Technical SLA).

However, the exchange of information between the entities located in different layers is difficult for technical and business reasons. On the one hand, it is unlikely for these entities, especially for the application providers and the infrastructure providers, to expose/reveal the internal operational parameters and processes to other layers due to the risk of competitors accessing this information. On the other hand, exchange of such information is not always technically feasible. Each entity focuses on a particular set of parameters that can be interpreted by this layer and based on these parameters the respective mechanisms for service provisioning and QoS enforcement are implemented. Lack of cloud standards across the cloud layers is another big issue that disallows the effective share of information between different entities. Some prospective attempts towards this

direction include OCCI [42] and OVF [44]. At this point it must be noted that if there is a direct liaison between the consumer and the IaaS provider (case where SaaS is offered directly through IaaS), the same translation issue exists, with the only difference that now the IaaS provider is responsible for dealing with it.

Furthermore, a major challenge for SaaS providers wanting to exploit the benefits of cloud computing is to manage QoS commitments to customers throughout the service lifecycle. The complexity of this problem increases considering the non-deterministic behaviour of the users. While targeted tools and services are provided in the PaaS layer (e.g., application and performance modeling, enactment and monitoring of application execution, evaluation of events and corrective actions, etc.), behavioural-based management of the services lifecycle requires mechanisms that abstract the aforementioned complexity by providing on-demand usage patterns and estimates for high-level application workload parameters. These will allow for optimum and automated management of QoS commitments by proactively detecting potential SLA violations and thus allowing for triggering the appropriate preventing actions (i.e., resource provision to efficiently facilitate the estimated requirements).

In this paper, we present a generic black-box approach for correlating resource attributes at a low-level with information at a high level related both to application terms and to users' behaviour. The approach includes a time series analysis mechanism for identifying and estimating high-level parameters with respect to the workload of the application (e.g., number of users, frequency of requests, geographical distribution, etc.), and a translation mechanism to convert these high-level parameters to resource attributes, which can be negotiated with IaaS providers in order to provide the corresponding resource. Both of these mechanisms are based on Artificial Neural Networks (ANNs), which are being used on a per software component basis and with application-specific terms (that vary depending on the examined scenario) and different hardware configurations. In one of our previous works [38], a translation mechanism was presented, that took as prerequisite the specification of the workload parameters by the client. In this approach through the addition of the Behavioural Layer, the clients may insert only the needed KPI levels.

The major advantages of the enhanced approach are the following: (i) enabling environments to facilitate real-time and interactive applications through the provision of accurate estimates during runtime with regard to the anticipated workload - leading to the required resource provision, (ii) cost minimization and provision of QoS guarantees through proactive SLA violation detection, (iii) increased business agility enabled by the decreased time at which additional capacity can be offered, (iv) optimized infrastructure capacity and utilization through the exploitation of the anticipated workload information, (v) limited amount of information that needs to be passed across the Cloud stack layers and (vi) applicability to different software components due to the generic and flexible nature of ANNs.

Finally we must stress that these methods were selected due to the specific Cloud characteristics of entity separation in the SPI model (that limits the exchange of information between the layers) and the ability to regulate resources automatically through available interfaces. Furthermore, they were selected in an effort to create a generic framework that can handle different types of applications (but with application specific characteristics) in an automated way as would be expected by a large scale PaaS provider. In more general web scenarios where these limitations may not exist, the proposed methods may also be applied, given that suitable elastic interfaces exist. However there may be other methods too, like analytical modeling, which may perform better. Analytical modeling in general needs significant amount of information regarding the internal structure of the applications and a sufficient amount of time spent by an expert to design and fine-tune it. If the general web scenario is for a specific application for example, then this type of modeling would be more efficient. However it cannot be realistically applied in a PaaS provider that has to model numerous services.

Following, relevant works to the one presented in this paper are listed and analyzed in Section 2, while the core of the proposed approach appears in Section 3. Indicative case studies of the application of the approach are described in Section 4, while detailed evaluation is contained in Section 5. The paper concludes with a discussion on future research and potentials for the current study.

2. RELATED WORK

Numerous interesting works have been identified with respect to the aforementioned issues. In [5], the importance of applying ANNs in the service oriented field is demonstrated. In this approach, the main goal of the ANN is to set up technical targets of design attributes of web service systems in terms of quality goals. In this process it takes under consideration the relationships between quality of service requirements and design attributes of a web service system. A number of critical requirements from perspectives of both web system platform development users and application users and their related design attributes are identified in order to guide the design and development process.

In [6], the LoM2HiS framework is presented. In this case the authors provide a framework for the reverse process of the one that we suggest; this is the translation of low level metrics to high level terms that are used in Cloud Service Level Agreements. Furthermore, they focus on more generic characteristics and terms such as availability, that are not application specific.

Authors in [7] propose a two-step analysis for multi-tier applications in order to decompose the SLA into low level objectives. In the first step, a resource demand estimation scheme is implemented through a profiling process. Profiling creates detailed resource profiles of each component in the application. A resource profile captures the demand of each transaction type at that resource, through historical data or benchmarking. Regression analysis is then applied to the

data, to derive the per transaction-type resource demands. The resulting resource profile for the application is then stored in a repository. In the next step, analytical models are used; a factor that is against the confidentiality issues mentioned in the introduction and hinders flexibility.

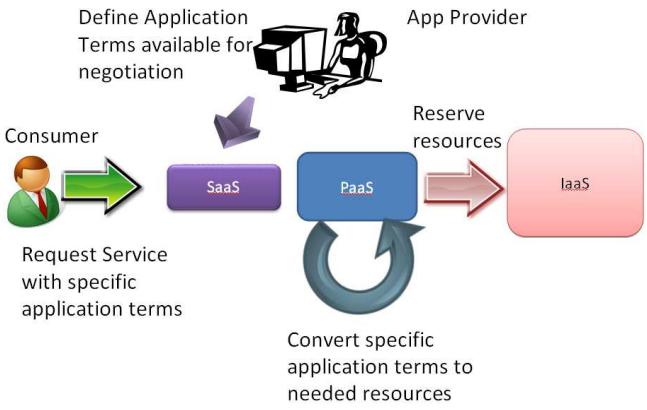

Figure 1: Discrete roles in the Cloud Stack

Another interesting work is presented in [8] where applications to be measured run under a strict enforcement of reservation of resources in order to determine if the given set of reservation parameters satisfies the time constraints for execution. If this is not the case, then these parameters are altered accordingly. If there is a positive surplus, the resources are decreased and if it is negative they are increased until a satisfying security margin is reached. This method succeeds in achieving high utilization rates, in the cost of test executions for every deployment with different application-level parameters.

A promising code analyzing process that allows for the simulation of system performance is described in [9]. It models the application by a parameter-driven Conditional Data Flow Graph (CDFG) and the hardware (HW) architecture by a configurable HW graph. The execution cost of each task block in the application CDFG is modelled by user-configurable parameters, which allows highly flexible performance estimation. The simulator takes the application CDFG and HW graph as the input and carries the simulation at a low level to catch the detailed HW activities. Source code analysis is also performed in the case of [10]. Analyzing the source code in order to conclude to resource requirements is probably the optimal solution. However, in the context of Service Oriented Infrastructures (SOIs), the limited amount of information passed from one layer of the Cloud stack to the other due to the discrete roles of the involved parties hinders the applicability of such solutions.

An interesting work, comparing Bayesian Networks (BNs) with ANNs is presented in [24]. According to this conclusion, ANNs have better model response times but worse model creation times. In our work it is considered that model response times are more significant than model creation times. The model is constructed once and can be refreshed at regular intervals but without tight timing constraints. Furthermore, our approach aims at identifying the dependency of the performance of each service from its inputs, thus leading to a per specific SLA estimation. In [25], an online system based on closed loop feedback from application KPIs is used in order to drive the resource management. This can be used only when SLAs are not needed for reserving a priori the resources, but the application can adjust its own reservations, without consulting first with the IaaS provider. However, any control-based method has the disadvantage that the anomaly must first occur, then be recognized and then be corrected through a suitable action (like increasing resources). If this delay is higher than the frequency of changes in the application resource needs it will lead to an oscillatory behaviour and the resources will always be one step behind from what is actually needed. In [26], a performance framework based on benchmarking and statistical inference is introduced, for message-oriented services.

A two-layer SLA approach between the different entities in a Cloud environment is detailed in [22]. The SLAs are divided into application SLAs (between the Consumer and the PaaS) and technical SLAs (between the PaaS and IaaS). An interesting standard for achieving interoperability between different approaches regarding model usage is found in [23]. Furthermore, this framework can be used as a schema for defining the inputs and outputs of the models.

With respect to on-demand resource provisioning at runtime, an interesting approach is presented in [31]. Authors introduce a mechanism for dynamic re-configuration and allocating of resources to virtual machines according to the application requirements. The approach is based on a reinforcement learning agent (namely VCONF) for scalability and adaptability, which enable auto-configuration of VMs on the IaaS layer. On the same subject, authors in [37] combine reinforcement learning (for offline training) with queuing models (for online management of the environment). Besides, the presented approach for resource estimates highlights that reinforcement learning is optimum for high performance in training.

Another interesting approach for workload prediction based on Kriging surrogate models to provide an autonomic controller is discussed in [33]. Multi-dimensional surrogate models are used to approximate the performance profile of applications through a utility function, being used afterwards to support the controller decision making. While thorough in the estimation part, this approach mentions the idea of incorporated future workload prediction but does not go as far as combining the two levels. Within the framework of task decomposition and scheduling, machine learning techniques are used in [35] to generate ANN-based performance models for prediction based on historical performance data. Machine learning through regression classifiers is also described in [36], in an approach that enables prediction of SLA violations at runtime based on measured information and facts originating from QoS and process instance (of composite services) data. This approach uses also workflow languages (BPEL) and checkpoints together with the regression models in order to identify at runtime the completion of a specific service.

An infrastructure that scales according to the workload is presented in [32]. The workload prediction is performed through an autoregressive model, while enhancements are made based on predictive data grouping according to historical information. Similar to that, authors in [34] discuss on a prediction model (namely trend-aware regression model) of time series that can be used during runtime using filtered data points of the resources.

What can be seen from the related work, is that while numerous extremely worthwhile approaches exist, they are commonly restricted to one of the two layers (Translation and Behaviour) that are considered in this paper. Another interesting conclusion is that ANNs are gaining more and more in popularity for these types of problems.

3. TWO LAYER ESTIMATION MECHANISM

The major goal of Clouds at the moment is to offer infrastructures that are self-manageable and QoS aware. In order to achieve this, various approaches have been followed, as seen in the related work. From these we have followed the approach discussed in [22], which divides the SLA process in two stages (Application-SLA and Technical-SLA), as it has been portrayed also in the introduction.

A. IRMOS Cloud Delivery Model

The two layer SLA has been followed in the IRMOS framework, in which also the mechanism presented in this paper was initially developed. According to this framework, which abides by the SPI model, an Application Developer, in order to adapt his/her application on a PaaS framework, needs to create an XML description of the components, following a specific template (available in [27]). This template contains critical information for the framework to understand the structure and details of the components. The Application Developer is the most suitable entity for creating this, since this role has full knowledge of the application and its characteristics. Furthermore this description contains details from which the Application SLA template will be created, for the application clients to populate. Regarding the work presented in this paper, the most interesting are the application specific terms that are configurable by the client (e.g., maximum number of users, resolution etc.) and affect application performance. These are characterized as workload parameters. In addition, the QoS outputs or KPIs (e.g., server response time) are also defined, which again are configurable by the client in the A-SLA. Another requirement from the application developer is that monitoring information regarding the KPI values is sent back to the platform during operation, through an interface and process defined in [44]. Relevant detailed information on the process of the creation of the description and the A-SLA templates may be found in [45].

Whenever a new A-SLA instance is created by a client, the PaaS layer has this available information (workload parameters and QoS levels), based on which it must choose the low level characteristics of the resources. These resources will be reserved on the IaaS layer, in order for the specific application instance to utilize them. The negotiations between the PaaS and IaaS layers will be performed upon the hardware metrics that the IaaS advertises and utilizes (like number of cores and RAM in the case of [30], or predefined instances like in the case of AWS). Thus what the PaaS needs in this process is a mathematical function that can be used to correlate the necessary parameters (workload, KPIs and hardware metrics) and thus decide on the suitable amount of resources for each individual application instance that is based on an individual A-SLA.

Through the aforementioned framework, the PaaS layer may perform this translation in a generic and standard way, due to the templates used, but in a fashion that is adapted to the characteristics of each individual component.For aiding the PaaS layer in this task, in this paper a two-level approach is followed:

- Translation Layer: a function approximation model is created for the cloud-based service/application that has the goal of correlating the A-SLA terms (workload characteristics and KPIs of the application) to the T-SLA ones (the hardware configuration on the IaaS). In order to identify these parameters, the mechanism takes advantage of the XML descriptions provided by the Application Developer, for defining the inputs and outputs of the model. Furthermore, due to the generic framework described, each application may define its own characteristics, thus leading to a per case model creation.
- Behavioural Layer: a model is created in order to capture the different patterns of usage of the service and be able to predict the future workload parameters of the latter. With the addition of this layer, the client now must specify only the needed KPIs and not the workload parameters.

Both layers are based on Artificial Neural Networks (ANNs), but of different type and architecture. ANNs are application-independent and self-learning black box models. Their greatest advantage is that they are an inherently non linear method, a characteristic that is critical in the context that we are investigating. Trying to predict application output from key inputs will at many cases include such non linear behaviour due to the software nature (for example if-then-else cases in the internal structure of the code). Other black box machine learning approaches such as Support Vector Machines may also be used, provided that non linear kernels are considered. One aspect of this approach is the global optimization achieved by the SVM training. However, in some cases and for some kernels this optimization problem may not be solved, at least in polynomial time. So a dilemma exists as to whether a non-optimal approach should be followed, such as ANNs, that may be applied in all cases, or an optimal one that may result in no models at all for some exceptions. Due to the fact that this approach is based around a service oriented framework, the need for automation was considered more important than a (usually) small deterioration of prediction capabilities. Furthermore, there

are indications ([46][47]) that ANNs may perform better in the case of noisy data. Given that in our computing model, we (the PaaS layer) have no information regarding the conditions of execution of the VM in the IaaS layer (for example how overloaded the specific physical host may be), it is safe to expect noise in the datasets. This may appear as significant deviations in the response times for the same load (number of users) and same VM used, simply because the IaaS provider has decided to start new VMs on the physical host.Another strong point for ANNs was the existence, at least from our experience, of more mature and easy to use software solutions (such as GNU Octave and Matlab toolboxes) with a variety of implemented functions and algorithms. In the next sections, details regarding the layers are presented.

B. Translation Layer

ANNs, due to the fact that they represent a **black box** approach, are perfect for usage in an environment where information is not easily relayed from one layer to the other, mainly for confidentiality purposes. For example, in the SPI model, the SaaS developer (Application Developer/Provider) is a different entity from the PaaS provider. Thus it will be reluctant to share critical information regarding the application, like the source code or internal structural behaviour. ANNs do not need any knowledge regarding the internal structure of the software components. They only need the inputs and outputs of the model, which are available since they are the A-SLA terms that are used between the client, the PaaS and the IaaS. Thus they are an ideal candidate for depicting the relationship between the high level application parameters and the low level indicators. In such a relationship (or function), the influence of the application workload parameters and the ability of the hardware metrics must be able to reflect the changes in the outputs of the service (KPI). By having such a correlating function, we are able to observe the effect of one parameter over the other.

Furthermore they need a representative dataset of executions that is used from a macroscopic point of view. Through the use of these data, the network weights are determined in order to identify complex, linear or non-linear dependencies of the output from the inputs. In this layer, we focus on function approximation ANNs.

1) Core of the mechanism

The core of the translation mechanism consists of an ANN factory, based on the GNU Octave tool [18]. Through this mechanism, an ANN is created for each software component that is offered through the PaaS provider. More details regarding the creation of the models can be found in [19]. In a nutshell, a Genetic Algorithm selects the fittest parameters for each network after an extensive optimization process in order to adapt it to each software component's needs (more suitable transfer functions, number of neurons per layer, total hidden layers etc.). The GA creates solutions based on these parameters and passes them to the ANN function that initializes and trains the networks based on these values. Then the evolution criterion (training error of the networks) is returned as a measure of the performance of a particular solution. The GA then retrieves these metrics and creates the next generation of solutions, based on evolutionary operators like elitism, mutation and crossover. In each generation the algorithm also saves the networks that pass a certain generalization test (prediction error in an intermediate validation set) In our case we utilized the standard GA methods offered by toolboxes like GNU Octave and Matlab. Thus the final network selection is directed towards solutions that have better generalization capabilities.

The inputs and outputs of the ANN are determined through proper information provided by the Application Provider in an XML format, as seen in Section 3.A. Then the ANN is stored and can be used in an online mode (for example during SLA negotiation) from the PaaS provider. The latter can use this model to correlate the application level terms (workload and QoS) with the hardware parameters and come up to a decision regarding the optimal trade-off. In Figure 2, the two ways the ANN can be created and used are portrayed. The most straightforward way is shown in (a), where the application parameters (workload and QoS) are considered as inputs and the predicted output is the necessary hardware descriptors. This gives one **definitive** answer to the translation question. Another solution, which is shown in (b) is to have the application workload parameters and resource level parameters as inputs, in order to predict their combination on the application KPI levels.

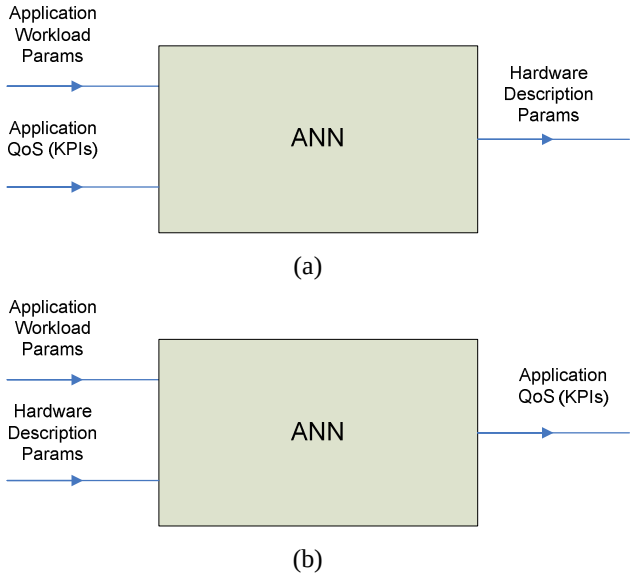

Figure 2: General Form of the models and variations a) Prediction of needed hardware configuration for meeting KPI levels b) Prediction of KPI levels for a given hardware configuration

In our case, due to the insertion of another optimization layer, as was done in the IRMOS project [17], the second approach was followed. Due to this, the ANN should be queried many times, in order to find the optimal hardware allocation for meeting the consumer's KPI levels with the minimum resources consumed. For example, slightly reduced KPI levels could result in a significant reduction in

service costs. Furthermore, through this form, evaluation is possible because it can be directly used with available hardware on which measurements are conducted. If it was used in the other way around, then each resource that was predicted would have to be found in order to validate the estimation (e.g., 1.45 Ghz of CPU)

2) Information Needed By the Other layers

At the introduction, it was stated that the major advantage of this approach is the limited amount of information that needs to be shared across the different layers of the Cloud stack. This information consists of non-confidential, already available information. In detail, what is necessary from the Application Developer at this stage is the following:

a) to be able to describe the application parameters of the software component which influence its performance requirements and are used in the SLA between the client and the PaaS provider (e.g., number of users, resolution of a streaming server etc.). This includes enumerated types or numerical intervals of possible values

b) to identify the QoS metrics (KPIs) that will be used to measure the component's performance levels

c) to provide means of changing the parameters mentioned in a) and monitor the metrics mentioned in b)

With regard to this information, there is no confidentiality issue since these are the direct terms that are used in the SLA.

For the IaaS provider, what is necessary to provide is the way it describes its hardware capabilities, which again is not confidential. Such information may be for example the metrics by which the computational performance of a hardware resource is measured (e.g., CPU frequency, specific benchmark score, number of cores needed, RAM size etc.). This information is also available and is the basis of the T-SLA between the PaaS and IaaS.

3) Sample Dataset Creation

In order to acquire this necessary dataset, a series of indicative executions must be performed. These must represent different conditions of execution by varying the application workload and the hardware parameters, in a parameter sweep fashion. The effect of each execution on the resulting QoS levels (KPI scores) is observed through the component's self-monitoring mechanisms. More details on this approach can be found in [11]. This is in general a time consuming process that is also dependent on the type of application and how it needs to be sampled. Indicatively, for the e-learning scenario described in Section 4, 420 different execution configurations were conducted, each of which had a 300 second duration, thus resulting in around one and a half day of sampling. Alternatively, available historical data sets may be used.

What is critical at this stage is to have samples for the edges of the numerical intervals of the parameters, as these are identified in Section 3.B.2. This is necessary in order to avoid extrapolation and have more accurate results in the estimation process.

4) Service Framework

For any approach to be applicable in a service oriented environment, a service oriented implementation must exist in order to embody it. In our case, this is described in detail in [14]. In this work, we implemented a layered, decoupled approach (Figure 3) in which any proposed method can be inserted as a GNU Octave or Matlab script. At the time a simple model based on multivariate linear regression was used in order to validate the framework. In the current work, this simple approach has been replaced by ANN models, through a mere replacement of the GNU Octave script at the last layer of implementation. This service is not installed on every node. A central instance may accommodate the necessary requests from a higher level component, such as a runtime evaluator. The evaluator may enquire the estimation framework on demand or on a periodical way, regarding all the running services for which the PaaS layer is responsible. More details regarding the scalability boundaries of the framework are included in Section 5.A.2.

This estimation framework is part of the PaaS platform. Thus one instance is necessary for all the services running on a PaaS platform. The deployment of such an estimation server may be either internally or externally to the platform, if the PaaS provider wishes to utilize IaaS resources, as a client.

Another issue is whether this role may be also undertaken by an external entity that acts as a specialized performance prediction provider. This entity may implement the described framework in the paper, acting thus as a consultant to PaaS providers. However for such an approach to be successful, the PaaS providers would need to send also historical data regarding the previous executions (and application SLA instances that correspond to them), that serve as the training set for the models. It is our view that while this may be feasible (and very interesting from a business model point of view), PaaS providers will most probably be reluctant to share this kind of information with external entities. Furthermore, for the workload prediction (TS-ANN layer), the PaaS layer would need to send monitoring information to the external entity regarding the usage conditions of a specific application running on their infrastructures. However this might not be desirable by the owner of the application instance due to potential confidentiality issues. Nevertheless, it is a very interesting opportunity for an added value service, in case the aforementioned limitations are lifted.

C. Behavioural Layer

If any approach needs to be proactive in order to achieve optimized management, it must be able to act before a critical change occurs and influences the response of the system. This change for example may be a peak in workload that deteriorates the system's response times. As seen in the related work, approaches based on control for example must first identify the deterioration on the system performance and then act to improve the resources and thus the response time of the service. However such an action may lead to non optimized usage of resources and oscillatory behaviour,

especially in cases of rapid fluctuation of a service. If for example the frequency of changes is higher than the system's response (time needed to identify the change but mainly time for raising extra VMs), this will lead to cases where the latter is always one step behind.

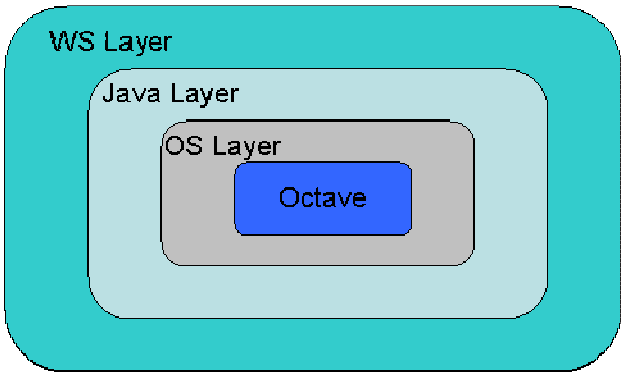

Figure 3: Layered Implementation of the service ([14])

In order to deal with these circumstances and act in a proactive manner, a new component has been added to the design of the translation framework presented in [38]. With the addition of this component we may achieve dynamicity in the service behaviour and automated management, without the service owner having to specify in each execution interval the high level workload inputs, but only the KPI levels. This component consists of a behaviour/workload prediction mechanism (Time Series ANN), that takes as input the time series of the varying workload features or inputs, like the number of users. By predicting accurately in advance the anticipated workload, we can then consult the Translation ANN regarding the predicted QoS of the service with the given or different resources. By regulating the latter in advance, we can adapt to varied workloads **marginally before they appear**, thus optimizing the resource management and profit.

The complete mechanism appears in Figure 4.

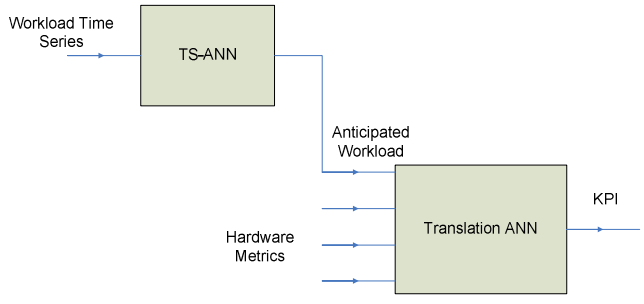

Figure 4: Combination of Behavioural Prediction (TS-ANN) and Translation Mechanism (Translation ANN) for regulating in advance the necessary resources

The TS-ANN module is based on a non-linear autoregressive network with exogenous inputs (NARX) network ([28]). This utilizes the formula of Equation 1 to predict future accesses, approach successfully used in the time series prediction field ([29]).. It mainly utilizes past values of the original signal (y(t)) in addition to a potential exogenous input (u(t)). In our case y=u, so that we can predict the future accesses based only on the pattern of the past workload. Furthermore, the non-linear functions of the neurons of the ANN help in adapting to sudden changes in the pattern, like peaks in activity.

$$y(t) = f(y(t-1), y(t-2,..., y(t-n_y),$$
$$u(t-1), u(t-2), ...u(t-n_u))$$

Equation 1: Formula of the NARX ANN

Through the training process, the network learns how the previous values affect the following by identifying patterns in the past and arranges the coefficients of each factor (weights and biases). Afterwards, the validation set is used in order to extract the error of the approach. The network is given the initial feed of time series values (the last of the training set) and starts predicting the values of the validation set. These are compared against the actual ones in the evaluation section.

In order to optimize the structure of the NARX ANN and improve its performance, we again utilized a Genetic Algorithm (GA) that defined the most critical parameters of the network through an iterative evolutionary process. This is critical since in an automated framework, this process must be performed without any human intervention for deciding the fittest configuration of the ANN. With relation to the previous layer and the function approximation ANN, in this case we considered also extra features as parameters for the GA like the following:

- Type of training function used during the training process
- Size of the initial feed required for a prediction (how many previous time intervals of activity must be inserted)
- Number of epochs used for training (iterations of the training process in order to reduce the error)

For each different setup, the performance of the network on the validation set was returned, in order to characterize the fitness of the specific solution. As the iterations (generations) of the GA proceed, the network's characteristics are refined so that the error of the latter is minimized. More information on this can be found in [39].

4. CASE STUDIES

The case studies presented in this section are separated again in two levels. We consider three cases with different inputs and outputs, to show the flexibility of our proposed black box approach. These include differentiations on the hardware attributes that are used by the IaaS providers for reserving resources and on the software components that need translation. For the Behavioural level, we consider joining one of the Translation case studies with the time series

analysis in order to create the framework for evaluating the overall approach.

A. *Translation Layer Case Studies*

1) FFMPEG Encoder Application on a general purpose Cloud

The first application was **an encoding of raw video in MPEG4 format with the FFMPEG encoder** [15]**.** In order to investigate the behaviour of the code, four application-level parameters/characteristics which have a direct effect on the **workload** produced by the application were altered. These were the duration of the video, the frames per second (FPS) used in the capture, the resolution of the images and an index of movement inside the video (0 for still images, 0.5 for mediocre movement and 1 for intense movement). This index is important in cases of video processing algorithms which compare consequent frames and perform a number of operations based on the differences between them. By having great differences between frames, such as in an action movie for example, computation time will increase. For the hardware-level parameters, two characteristics were chosen, the CPU speed of the node, plus the number of available cores.

The overall execution time until the encoding is finished was selected as the KPI metric. Depending on how fast the client wants his application to finish, the PaaS provider will choose a better hardware node and will charge the former accordingly. The overview of the FFMPEG ANN model is depicted in Figure 5. The specific application was chosen since there are numerous application parameters (4 in number) that affect its completion time. Furthermore, the KPI metric (completion time) represents an application area that consists mainly of a task-job (batch) nature.

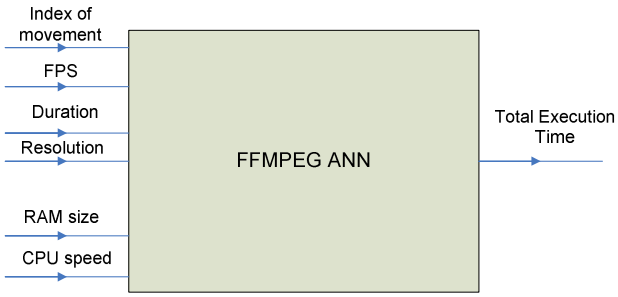

Figure 5: ANN Model for the FFMPEG component

2) E-learning Application on a Real-Time Cloud

This application (more details can be found in [11]) is a Tomcat-based server that is supported by a MySQL database. Its main goal is to receive requests with GPS location and return pointers to e-Learning objects that are located nearby. It is deployed on a Cloud that supports real-time features [16], so that the inner server response times have guaranteed levels of QoS. The main difference of this Cloud from the mainstream ones is that it uses a real-time scheduler [13] in order to guarantee the CPU percentage of a Virtual Machine (VM) over a time period. Furthermore, it uses network control and storage access control policies in order to guarantee these metrics. So, in the negotiation request from the PaaS to the IaaS, the PaaS must also specify which are the real time critical parameters (computational budget Q over a time period P), in addition to normal Cloud hardware parameters like the CPU frequency or the number of cores. The reason for choosing this case study was that it represents a different area of applications (interactive multimedia), with different KPI needed (distribution of server responses) from case study 1. Furthermore, the real time attributes may significantly affect the performance of the server due to the interactivity aspect, so it was another differentiator with regard to case study 1.

The ANN model used in this case has the aim of translating the consumer-side requested application parameters (in this case the number of users supported by the application) and the hardware resources used (Q/P, P, CPU speed and RAM) to QoS level characteristics (server mean response times and standard deviation of these times). A general form of an ANN as this is produced from software like Matlab or Octave appears in Figure 6.

Now let's consider the case where the aforementioned model is not sufficient for describing the application. One could argue that it is not the number of users that connects to the application, but their interactivity levels, for example how fast they are moving. A fast moving user would generate more requests to the server than a more slowly moving one. The particular application under investigation monitored and documented the response time of the server to each individual request, in addition to the timestamp.

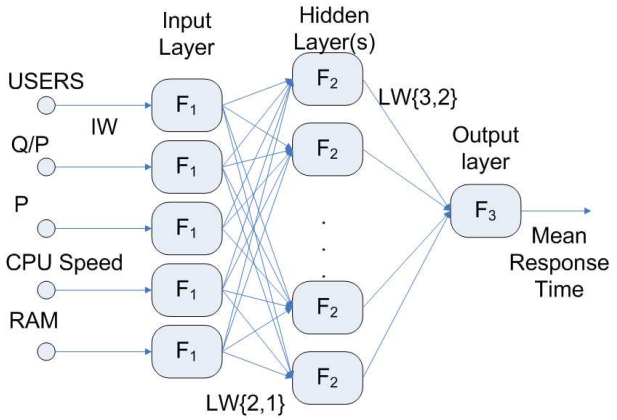

Figure 6: Example structure of the e-learning ANN model for mean response time prediction. For the standard deviation prediction a different ANN model was created

So what is needed is to change the input parameter, from "number of users" to "requests per interval", pre-process the existing dataset in order to extract the needed information and retrain the ANN model. Then it can be used during the online operation of the application. This means that actually no SLA negotiation will be performed, but direct estimations, depending on the interactivity levels portrayed by the users, and according reservations will be made towards the IaaS provider, through a more direct resource reservation process. However it should be noted that in order

for the pre-processing to be this flexible, the documented information that is necessary should be a priori included in the collection of the measurements (in this case the timestamp).

All the aforementioned variations indicate the greatest advantage of the ANNs, which is the flexibility in the model creation. In order to alter the model, all we need to do is replace the column order in the dataset that is provided for training, or alter the pre-processing of the available datasets (like in the case of "requests per interval").

B. Behavioural Layer

1) Combined Workload Forecasting and Translation Mapping of a Wikipedia Server

For the Behavioural layer case study, it was considered that in this case there is no single entity that wants to reserve a service with a specific high level parameter (like for example a school principle would do before a classroom trip to a museum where the e-Learning server of the aforementioned use case is deployed). The owners of the server offer it to the general public and they just need the KPIs of the server to be under a specific limit. Thus it is the responsibility of the PaaS layer to maintain this limit, by self-managing the application. In order to do this, it must be aware of the anticipated traffic beforehand, so that it can marginally provision the necessary resources. For this to happen it must integrate the TS-ANN functionality explained in Section 3 with the Translation ANN..

As a use case for the combined mechanism we considered a Wikipedia-like server (based on [40]). For this application 3 inputs were considered, the number of users accessing the server, the cores and the RAM of the VM. For the VM we utilized the Flexiscale Cloud ([30]). As a KPI output we considered the average response time of the server. This scenario was chosen since we needed to demonstrate the efficiency of the combined two-level approach described in Figure 4. Thus we needed a realistic data set that would come from the actual usage of a similar service on the Internet and would depict the variations in demand. These variations should be predictable by the proposed Behavioural Layer. Then we could deploy a similar service and simulate the real traffic data in order to get the server response times (for the Translation Layer). This combined data set (incorporating user demand and application response to that demand under different hardware resources used) was not available through numerous investigated trace sites. Through its acquisition, we validated the runtime operation and behaviour of the service when the proposed two-level mechanism was deployed. Through this it was feasible to check the end to end error of our approach. Furthermore, it is a typical case study in which the owner of the application instance may not be aware in advance how many users will access the service.

All the aforementioned use cases did not need alterations in the model creation, other than the different number of inputs and the new dataset. This would not happen if for example we followed an approach based on analytical modeling instead of a black box one. At this point it must be stressed that the change of hardware characteristics between the case studies could be minimized (with the exception probably of the real time characteristics in Case study 2) if Cloud providers followed a more homogeneous and standardized way of describing the VM instances in their infrastructures. This can be achieved through the usage of a standardized set of benchmarks, with which the IaaS entities may declare the capabilities of their virtualized resources and can greatly enhance Cloud QoS management.

5. EVALUATION

A. Accuracy and structure of the Translation ANN models

As it has been described in the previous sections, ANNs can be used to directly translate application terms to resource attributes with very little information provided by the Application Developer. But is the accuracy of these translation rules satisfactory? Do the reserved resources meet the expected QoS?

For investigating this factor, the ability of the networks to predict the performance of the software components was measured. The available datasets were obtained through sampling. For example, for the Wikipedia-like server we followed a sampling process in order to get response times for the server for different numbers of users. The original sampling set was in the range of 10-1150 users, with a step of 10. Three virtual resources on the Flexiscale Cloud Platform were used, one 2 core-2 GB of RAM virtual machine, one with 4 cores and 4 GB of RAM and one with 8 cores and 8 GB of RAM. This was performed also because historical data tend to gather around the specific interval of usage so it is in general easier to predict due to the oversampling in the particular area. Through a more generic sampling process the proposed method can be better validated. The results (percentage of time outs, average response delay and standard deviation of delays) appear in Figure 7 for all hardware setups.

A similar sampling process was followed for the other case studies too. More info on the e-learning sampling can be found in [11]. After the sampling, 50% of the dataset acquired through this process was used for training the network models, regarding the applications described in Section 4, and 20% for an intermediate validation check during the GA execution. Then, the remaining 30% was used as a final validation check of the networks accuracy in predicting the output QoS levels of the application. The Mean Absolute Error (MAE) is depicted in Table 1, along with the structural characteristics of the networks.

In our case, the GA tried out many different combinations of ANN structures, each of which was trained, thus resulting in about half an hour of total training. The GA settings for the Translation ANN optimization were set to 20 generations, with each one investigating 20 solutions. The GNU Octave tool and its optimization package were used. The GA decided on the number of layers, number of neurons per layer and type of transfer function available. The intervals of investigation for these parameters were 3-5, 1-30 (except for the inputs and outputs which are determined by

the model) and 1-3 respectively. The ANN initialization functions were 'initwb' for the layers and 'rands' for the weights and biases. The produced ANNs are feed-forward back-propagation networks, trained with the Levenberg-Marquardt algorithm [12]. The performance criterion was the mean square error in the training set and it was trained for 150 epochs. The parameter goal was set to 0.0000001 and maximum training time for each ANN to 500 seconds. Training time was usually in the range of 1 minute, but this also depends on other factors like size of the training dataset, parameters to calculate etc.

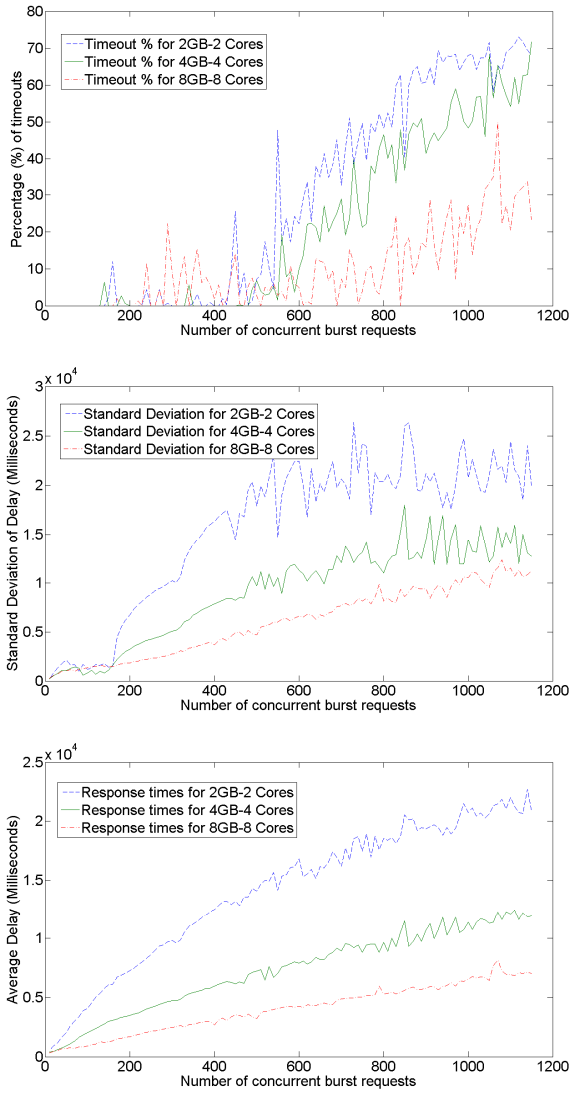

Figure 7: Comparative results for sampling Wikipedia server experiment response times, percentage of timeouts and standard deviation

The approach was compared with the multivariate regression method for the Wikipedia Server, with the same inputs and on the same dataset. This method was selected since it represents a simple, fast and computationally non-intensive method. The results, which were very discouraging, appear also in Table 1.The individual errors in the validation set for the same method (mvregress) appear in Figure 8. From these it is evident that the compared method has very significant errors, even in the validation, and cannot be used with safety during any runtime operation.

Table 1: Accuracy of the models for the various case studies in the validation test

Type of ANN Model	Neurons per Layer	Transfer Functions per Layer	Mean Absolute Error
FFMPEG Encoder Execution Time	6-13-1	Tansig-Softmax-Tansig	8.21%
E-Learning Server Mean response Time	5-2-1	Tansig-Tansig-Purelin	2.51%
E-Learning Server Standard Deviation	5-2-1	Logsig-Logsig-Purelin	2.75%
Wikipedia Server	3-3-2-1	Tansig-Tansig-Tansig-Purelin	3.46%
Multivariate Regression for Wikipedia Server	-	-	66.06%

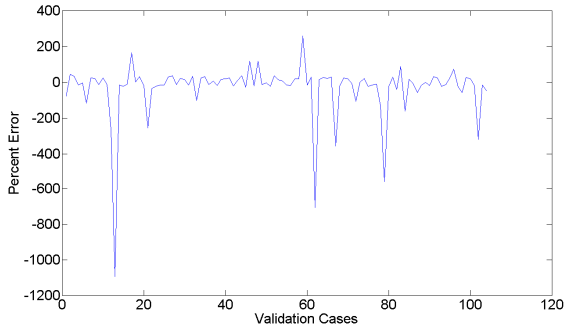

Figure 8: Validation cases percent error for the multivariate regression (Wikipedia Server)

More information regarding the ANNs is available (e.g., specific weights and biases of the networks) but was not included here due to space limitations.

1) Robustness of the Translation ANN model

An ANN model's robustness depends on the relationship between the number of available data points for training and

the parameters of the network (weights and biases) that must be calculated through the training process. If the number of training points is higher than the number of network parameters that need to be calculated during the training process, this means that each variable is calculated by taking the mean from more than one points. Thus, no single data point (that may be influenced by random noise during the collection of the data set) can significantly affect the resulting values. In detail, for a 3-layer ANN, with *N* inputs, *K* number of neurons in the hidden layer and *L* outputs the number of unknown network variables *V* (connection weights between the neurons and bias of each neuron) is given by Equation 2:

$$V = 2 * N + (K * N + K) + (L * K + L)$$

Equation 2

In our models (with *N*=5, *K*=2, *L*=1), this results in 25 variables. The available data set in our case (e-learning server) consisted of 420 data points. From this, 50% was used for direct training of the network, resulting in 210 data points, which is much higher than the needed 25 values. Another 20% was used during training for validation purposes. The overall network prediction capability (as this is depicted in Table 1) was measured on the remaining 30% of the data set that **was not used in any way during the training process**, thus simulating a real testing situation.

At this point it must be noted that a "chicken and egg" problem arises. How can we know how many samples to take (during the process described in Section 3.B.3) when the structure of the network is not known a priori but is decided by the GA. This can be solved if we consider some maximum values for the network parameters and demand that the dataset has the necessary size. On the other hand, given that the dataset may be time consuming to create, or not available at all, we can check its size, or the size of the available historical data, before creating the model and thus apply the suitable boundaries to the optimization of the ANN structure.

2) Time performance of the Translation ANN

Given that the proposed framework is included in a service oriented infrastructure, an evaluation must be performed on the delays inserted by the latter for acquiring the necessary estimations. As seen in Section 3, the ANNs are created and enquired through GNU Octave, which is a specialized open source numerical software tool, similar to Matlab, embedded on a service framework. So what is critical to observe is the time behaviour of the service under extensive workload (requests for performing estimations). And while the creation of the models is mainly an offline process, that does not have significant timing constraints, the enquiry for estimation occurs during the online SLA negotiation or during service operation. Thus it may be constrained on the delay of the response.

In order to stress the service framework, we used a multithreaded client that targeted the estimation service with the specific request based on different model IDs. We changed the model IDs for which the client requested a prediction for a given input vector so that no caching phenomena could influence the delays. Then we consecutively launched different numbers of concurrent threads (from 1 to 200 with a step of 10) so as to simulate different number of users. Each thread made 30 consecutive calls so that the delays in thread creation would not influence the actual concurrent ones. The first and last 10% values were discarded in an attempt to avoid any transitional stages.

From the individual response delays the average delay values were extracted along with their standard deviation and percentage of time outs. These appear in Figure 9. It is evident from the sampling process that these delays reflect 200 actual users hitting simultaneously the server in a **burst** mode fashion. From these graphs it can be concluded that the delays have a linear behaviour with regard to the concurrent burst requests. However, after a specific limit (around 160 users) there is an almost exponential increase in the standard deviation of the response times and the percentage of time-outs, thus indicating the operational limits of the framework for the specific host that was used. This was a dual core (@1.6Ghz) with 2 GB of RAM, running Ubuntu 10.04 and Glassfish 3, in which the service was deployed. Thus in conclusion, 160 concurrently executing requests is the operational limit, with an average delay of 30 seconds. In case lower times are necessary, the maximum allowed requests may be regulated accordingly by utilizing the relationship depicted in Figure 9 and utilizing a load balancing design with multiple estimation server instances. If the PaaS provider wishes, it can also externalize these worker servers to an IaaS provider, acting as a client of virtualized resources.

At this point it must be stressed that the server configuration is also critical for the behaviour of the service. By limiting the server threads to a number around 20-25, the requests are queued up until a worker thread is available. Due to the fact that the maximum worker threads are constrained (and in a low value), thrashing phenomena are avoided. These could be caused by large numbers of threads executing at the same time but wasting all of the CPU resources in their task switching. Thus it is expected that the linear behaviour will remain for higher numbers of users too, since the only difference is the time spent in the queue. However, as seen above, when the time-out limit of the calls is reached, then these will be practically useless and the delays will be in vain. Depending on the trade-off between the maximum waiting time and the hardware assigned to the service itself, this usage limit may be configured.

The times measured in the figures are for a request with a single input vector to the models (thus a prediction for a single workload and hardware parameter set). However it was mentioned that because of the optimization layer that investigated numerous different hardware configurations before deciding, the estimation process must be performed for more than one times. This means that the ANN is fed with large arrays of multiple inputs to which it should respond with an array of multiple outputs, after calculating each individual output from each individual input. This does not mean that the delays showed in the figure will be multiplied by the size of the input array. The main part of the delays is the service call and the Octave environment initial

setup call. Inside this environment, each individual execution is performed much faster, in the range of **10 milliseconds** per estimation, as seen in Table 2.Thus by merging all the requests for a specific service in one call we can achieve better times for the optimization phase.

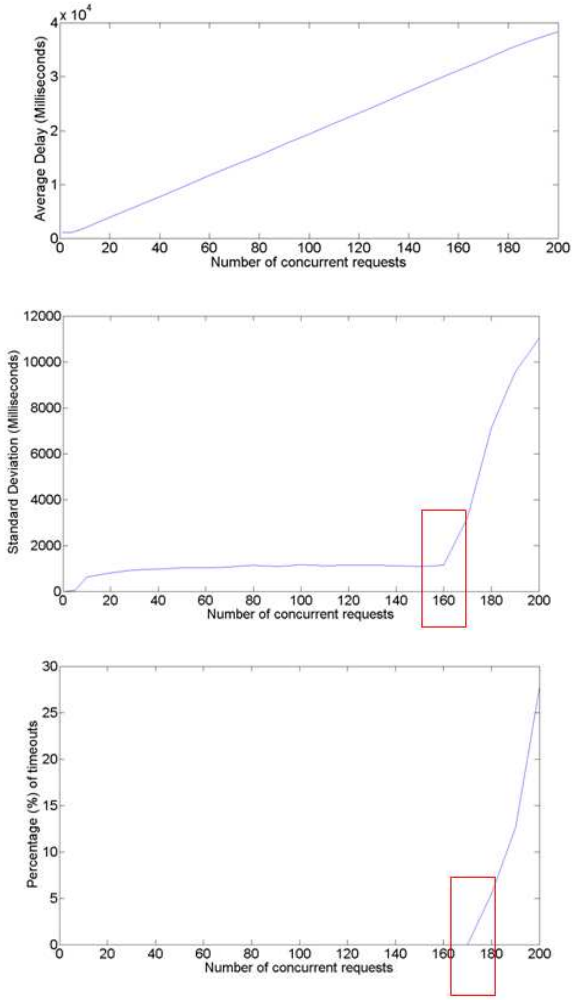

Figure 9: Delays, Standard Deviation and Percentage of time outs for the service estimation framework

Table 2: Difference in execution times depending on number of estimations (1 client)

Octave execution for 1 estimation	1.104 seconds
Octave execution for 100 batch estimations	1.978 seconds

B. Behavioural Layer and Combined Evaluation Through Trace Driven Simulation of Runtime Operation

For evaluating the proposed combined approach, a real world dataset was needed, that would give the information that the models require. This information consisted of a realistic traffic pattern that is fed in the TS ANN so that the future activity of the server is predicted. Then based on this prediction of anticipated usage, the Translation ANN can perform a second level prediction for the QoS based on the used resources. This overall prediction must be compared to the actual values of the response times for the actual number of the arrived workload.

So during this runtime operation, the evaluation is based initially in the forecasting of the next values of hits for the Wikipedia-like Server. Based on these expected values, we predict through the Translation ANN the response time for the given infrastructure. Finally we evaluate our overall prediction framework through comparing the predicted response time for the predicted traffic with the actual response time of the real traffic.

This combined dataset, to the best of our knowledge, did not exist. For creating it, the traffic data from the English Wikipedia hits per hour were collected and processed to represent a realistic series of hits over time. The samples involved a 3-month period, from July to September 2011and were divided by 100000 so as to reduce the time needed for the sampling process and the need for resources for the experiments. This does not change the form of the patterns therefore the measured ability of the TS ANN to identify it. The form of the overall sample appears in Figure 10.

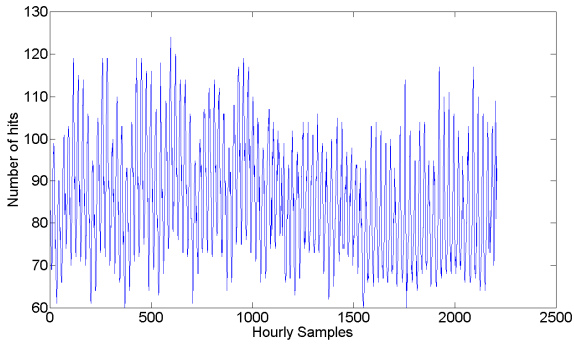

Figure 10: Form of the English Wikipedia hits per hour (divided by 100000) for the observed period (July-September 2011)

In order to have also the response times for the runtime operation evaluation, we proceeded in a trace-driven simulation. In this, the Wikipedia trace served as input to a multithreaded client that had the goal of simulating the amount of users requesting pages from the site. However, the way the data are reported is per hour, so there is no information as to how they are distributed over this time interval. In order to represent the worst case scenario, which is the arrival of this traffic in burst mode, we created a client that generated the amount of requests equivalent to the hourly original trace through thread creation. Thus an equivalent number of threads was created simultaneously and launched requests towards the server. This was also done in order to speed up the simulation process. Each client collected 5 samples of calls towards the server, for gathering a sufficient dataset. The top and bottom 10%.of the samples were discarded in an attempt to eliminate any client thread bottleneck phenomena as suggested in [33]. This is necessary for removing values of response times that do not represent the correct number of users (because for example not all

threads have been started in the beginning or some threads have finished in the end of the measurement interval). During the sampling time, the overall delay for each call was measured in the client side and was logged in a common file. The pseudo-code for the client side appears in Figure 11.

Furthermore, different hardware setups were used by creating different virtual servers on the Flexiscale Cloud [30], for getting the response times for different resources.

```
1.while traceline!= null
2.      hitSize=str2int(traceline)
3.      for i=1:hitSize
4.              start separate Client threads
5.              for each thread//parallel processes
6.                      for samples=1:max_samples
7.                              Get page
8.                              Log delays
9.                      end
10.             end
11.     end
12.     wait for all threads to finish
13.end while
```

Figure 11: Pseudocode for Client Multithreaded Simulator

As stated earlier, the traffic pattern collected from Wikipedia served as input for the TS-ANN. For this, we broke down the available 2208 value set in two parts. The first 1500 values served as the training and validation set. The final 708 values were used for the **runtime operation** test. This served two purposes:

- Evaluate the final selected TS-ANN for a time period that had not been used in any way during training, representing a period of normal operation of the latter in case it was deployed in a running infrastructure
- Offer the predicted values for this period to the Translation ANN for acquiring the predicted outputs based on the anticipated usage.

The GA-based optimization for the TS ANN was performed in Matlab. The optimization parameters were the neurons per layer (1-30), training functions (1-6, from the 6 available in Matlab: 'trainlm', 'traingd', 'traingdm', 'traingda', 'traingdx', 'trainbr'), initial feed (2-10), learning rate (0-1), training epochs (50-800) and transfer functions (tansig, logsig, purelin, radbas). The GA ran for 50 generations, with a population size of 30. Elite members were set to 8, crossover fraction to 0.8, the migration interval to 10, migration fraction to 0.2, initial penalty to 10 and penalty factor to 1000. The training goal for the networks is set to 0.01 and the maximum training time for each ANN to 2000 seconds.

The produced network appears in Table 3, along with its accuracy in the runtime operation test. This is compared against the open source Java library OpenForecast [41]. OpenForecast provides a variety of methods and a general method to obtain the best model from all of them. This function was used. The comparison between the two approaches also appears in Table 3.

Table 3: Characteristics and performance of the TS-ANN in the runtime operation set

Method	Past values used	Mean Absolute Error (%)	Maximum Absolute Error (%)
TS-ANN (Logsig-Purelin)	4	3.70	22.01
OpenForecast Best Method	10	5.32%	36.77%

The accuracy of the TS-ANN for the runtime operation appears in greater detail in Figure 12. From this it is evident that it is able to capture the actual values with increased accuracy for a long look-ahead interval. We avoided plotting the OpenForecast method in Figure 12a) for better visibility of the graphs. The individual percent errors for both approaches are combined in Figure 12b). From this the enhanced distribution of values for the TS-ANN is evident.

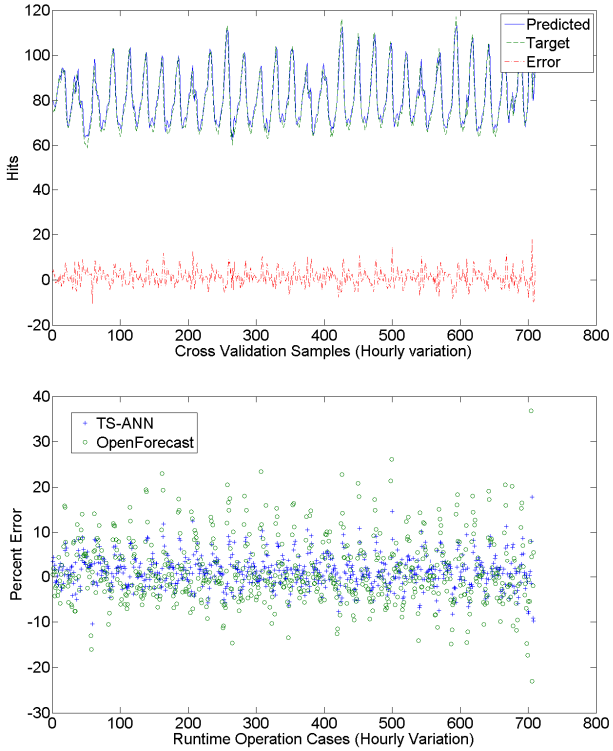

Figure 12: Look ahead prediction of anticipated traffic from the TS-ANN for the runtime operation: a) target and predicted values b) percent error in each case for the TS-ANN and the OpenForecast model

The predicted values of the TS-ANN obtained from this step were fed at the Translation ANN for extracting the anticipated response time of the server. Then these values were compared to the actual ones obtained during the trace

driven simulation for the last 708 values. These values were not used at all during the training and selection of both types of ANNs. Thus we compared our overall approach, by observing in this comparison both the errors from the TS-ANN and the Translation ANN. The results for all three hardware configurations appear in Figure 13 . The accuracy of the Translation ANN that was used during runtime operation appears in Table 4., compared to the validation accuracy that was presented in the previous sections. It is evident that while this overall error (combined TS-ANN and Translation ANN) is higher than the validation case (which is normal since it also incorporates the TS-ANN error and an unknown data set), it is still very accurate with a mean absolute value of less than 10%.

Table 4: Response Time Translation ANN Accuracy in Runtime Operation in comparison to the validation phase

Mean Validation Error	Max Validation Error	Mean Runtime Operation Error	Max Runtime Operation Eroor
3.46%	14.7%	9.61%	45.03%

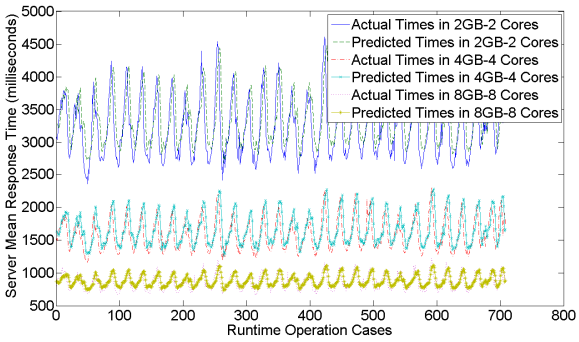

Figure 13: Overall approach error for the runtime operation trace driven simulation cases (including anticipation error and translation error)

In Figure 14 the percent errors of the runtime operation evaluation are portrayed, independently of type of machine used. One interesting conclusion from this figure is that the proposed method mainly overestimates the amount of resources needed, as seen from the fact that the majority of errors are positive and the positive maximum values are higher than the according negative ones. This leads to safer provisioning, especially in cases of soft real-time applications, in comparison to an under-provisioning approach. Furthermore, it reduces the size of a safety coefficient based on a probabilistic approach for avoiding SLA violations on a given percentage (e.g., 95%).

6. CONCLUSIONS

The translation problem from application-level terms (workload and QoS) to resource-level attributes is a challenging issue in the modern Cloud computing stack, mainly due to the discrete roles between the layers (SaaS, PaaS, IaaS) and the limited amount of information available from one layer to the other. Furthermore, the management of a service running on a virtualized infrastructure requires a priori knowledge of the anticipated future traffic. In this paper, a generic approach that is based on a two level prediction (future workload behaviour and translation to KPIs and needed resources) was used as a mediator for the PaaS layer, in the latter's attempt to regulate the resources needed for an application instance with varying demand and specific QoS levels on a Cloud. Furthermore, the owner of the service instance does not need to pre-calculate the expected workload, since this can be identified by the Behavioural Layer of the mechanism. The responsibility of the owner is simply to declare the necessary KPI levels.

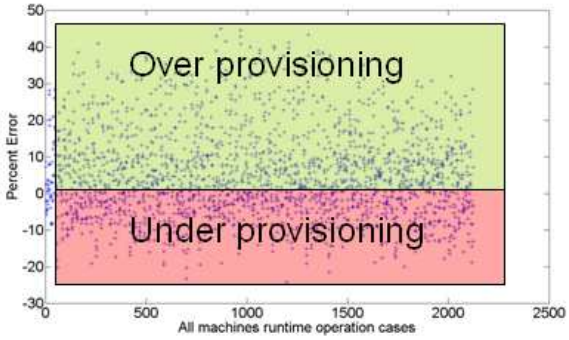

Figure 14: All Machines Percent Error for the runtime operation trace driven simulation cases for the combination of TS-ANN and Translation ANN

The flexibility of the approach was demonstrated through various case studies with altering demands, incorporating different types of applications, KPI outputs and hardware descriptors. It demonstrated significant accuracy and was proven robust, in a variety of cases regarding standard clouds, real time clouds and different types of applications. What is more, the combined approach was evaluated through a thorough trace-driven simulation from the real world, in a runtime operation manner, giving promising results for the overall ability to predict and pre-configure the system's behaviour. The combined mean error of this validation was less than 10%.

In addition, the time requirements for producing the estimations are within the logical time frames expected in a service oriented infrastructure. The confidentiality issue across the *aaS layers is satisfied due to the need for limited information regarding the inputs or outputs of the application. This was feasible due to the black box approach followed. Changing the inputs and outputs of the model is as simple as changing the column sequence in the data matrix that is used to train the ANN model. Finally, the use of a Genetic Algorithm aided in avoiding issues with ANNs such as small learning rates, fine-tuning considerations and generalization capabilities.

For the future, an aspect that is worth pursuing is the incorporation of more modelling parameters. These may include energy efficiency for example, directly in the Translation phase, for optimizing application behaviour. Indirectly the time series module may be used to predict future server utilization and thus suitable consolidation

decisions at the IaaS layer for the minimization of active physical nodes without violation of the KPIs. Another metric that can be predicted is the production of energy from renewable sources at the infrastructure layer. This feature may be subject to great variations and can lead an IaaS provider in optimal resource management with regard to energy consumption. Another feature that will be investigated is the potential usage of alternative machine learning techniques such as Support Vector Machines, since there are indications that they perform more efficiently than ANNs in a variety of cases.

ACKNOWLEDGMENT

This research is partially funded by the European Commission as part of the European IST 7th Framework Program through the projects IRMOS under contract number 214777 and OPTIMIS under contract number 25715. The authors would also like to thank Prof. Anastasios Doulamis for his valuable experience and suggestions with regard to the machine learning methods used.

REFERENCES

[1] T. Erl, “Service-oriented Architecture: Concepts, Technology, and Design”, Upper Saddle River: Prentice Hall PTR. ISBN 0-13-185858-0, 2005

[2] Youseff L, Butrico M, Da Silva D (2008) Toward a unified ontology of cloud computing. In: Grid computing environments workshop, 2008. GCE ‘08grid computing environments workshop,2008, GCE ‘08, pp 1–10P

[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A view of cloud computing. Commun. ACM 53, 4 (April 2010), 50-58.

[4] Amazon Elastic Cloud: http://aws.amazon.com/ec2/

[5] Lianzhang Zhu; Xiaowing Liu; , "Technical Target Setting in QFD for Web Service Systems Using an Artificial Neural Network," Services Computing, IEEE Transactions on , vol.3, no.4, pp.338-352, Oct.-Dec. 2010

[6] Emeakaroha, Vincent C.; Brandic, Ivona; Maurer, Michael; Dustdar, Schahram; , "Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," High Performance Computing and Simulation (HPCS), 2010 International Conference on , vol., no., pp.48-54, June 28 2010-July 2 2010

[7] Chen, Iyer, Liu, Milojicic, Sahai, "SLA Decomposition: Translating Service Level Objectives to System Level Thresholds," icac, p. 3, Fourth International Conference on Autonomic Computing (ICAC'07), 2007

[8] Jae W. Lee, Krste Asanovic, "METERG: Measurement-Based End-to-End Performance Estimation Technique in QoS-Capable Multiprocessors," rtas, pp. 135-147, 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), 2006

[9] Zhengting He, Cheng Peng, Aloysius Mok, "A Performance Estimation Tool for Video Applications," rtas, pp. 267-276, 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), 2006

[10] A. Strube, D. Rexachs, E. Luque, “Software probes: Towards a quick method for machine characterization and application performance prediction”, Proceedings of the 7th International Symposium on Parallel and Distributed Computing, 2008

[11] George Kousiouris, Fabio Checconi, Alessandro Mazzetti, Zlatko Zlatev, Juri Papay, Thomas Voith, Dimosthenis Kyriazis, “ Distributed Interactive Real-time Multimedia Applications: A Sampling and Analysis Framework ”, in 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010), July 2010, Brussels, Belgium

[12] Moré, J.J.: The Levenberg-Marquardt algorithm: implementation and theory. In: Watson, G.A. (ed.) Numerical Analysis, Dundee 1977. Lecture Notes in Mathematics, vol. 630, pp. 105–116. Springer, Berlin 1978

[13] Fabio Checconi, Tommaso Cucinotta, Dario Faggioli, Giuseppe Lipari, "Hierarchical Multiprocessor CPU Reservations for the Linux Kernel," in Proceedings of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009), Dublin, Ireland, June 2009

[14] Kousiouris, G.; Kyriazis, D.; Konstanteli, K.; Gogouvitis, S.; Katsaros, G.; Varvarigou, T.; , "A Service-Oriented Framework for GNU Octave-Based Performance Prediction," Services Computing (SCC), 2010 IEEE International Conference on , vol., no., pp.114-121, 5-10 July 2010

[15] Fabrice Bellard, FFMPEG multimedia system: http://www.ffmpeg.org, 2005

[16] Dimosthenis Kyriazis, Andreas Menychtas, George Kousiouris, Karsten Oberle, Thomas Voith, Michael Boniface, Eduardo Oliveros, Tommaso Cucinotta, Soren Berger, "A Real-time Service Oriented Infrastructure," accepted on the GSTF International Journal on Computing, ISSN 2010-2283

[17] M. Boniface, B. Nasser, J. Papay, S. Phillips, A. Servin, K. Yang, Z. Zlatev, S. Gogouvitis, G. Katsaros, K. Konstanteli, G. Kousiouris, A. Menychtas, D. Kyriazis, “Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds”, he Fifth International Conference on Internet and Web Applications and Services, ICIW 2010, May 9 - 15, 2010 - Barcelona, Spain

[18] GNU Octave tool: http://www.gnu.org/software/octave/

[19] George Kousiouris, Tommaso Cucinotta, Theodora Varvarigou, The Effects of Scheduling, Workload Type and Consolidation Scenarios on Virtual Machine Performance and their Prediction through Optimized Artificial Neural Networks, Journal of Systems and Software, In Press, Accepted Manuscript, Available online 14 April 2011, ISSN 0164-1212, DOI: 10.1016/j.jss.2011.04.013.

[20] Markus Böhm, Galina Koleva, Stefanie Leimeister, Christoph Riedl and Helmut Krcmar, “Towards a Generic Value Network for Cloud Computing”, Economics of Grids, Clouds, Systems, and Services Lecture Notes in Computer Science, 2010, Volume 6296/2010, 129-140

[21] Alhamad, M.; Dillon, T.; Chang, E.; , "Conceptual SLA framework for cloud computing," Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on , vol., no., pp.606-610, 13-16 April 2010

[22] Georgina Gallizo, Roland Kübert, Karsten Oberle, Klaus Satzke, Spyridon V. Gogouvitis, Gregory Katsaros, and Eduardo Oliveros: A service level agreement management framework for real-time applications in cloud computing environments. Proceedings of the 2nd International ICST Conference on Cloud Computing, Barcelona, Spain October 26 – 28, 2010

[23] Alex Guazzelli, Kostantinos Stathatos, and Michael Zeller. 2009. Efficient deployment of predictive analytics through open standards and cloud computing. SIGKDD Explor. Newsl. 11, 1 (November 2009), 32-38. DOI=10.1145/1656274.1656281

[24] Rui Zhang and Alan J. Bivens. 2007. Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems. In Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches (SOCP '07). ACM, New York, NY, USA, 67-74. DOI=10.1145/1272457.1272467

[25] Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah. 2010. Q-clouds: managing performance interference effects for QoS-aware clouds. In Proceedings of the 5th European conference on Computer systems (EuroSys '10). ACM, New York, NY, USA, 237-250. DOI=10.1145/1755913.1755938

[26] J. Happe, D. Westermann, K. Sachs, and L. Kapova, “Statistical inference of software performance models for parametric

performance completions," in Lecture Notes in Computer Science, 2010, Volume 6093/2010, 20-35, DOI: 10.1007/978-3-642-13821-8_4

[27] Dimosthenis Kyriazis, Ralf Einhorn, Lars Furst, Michael Braitmaier, Dominik Lamp, Kleopatra Konstanteli, George Kousiouris, Andreas Menychtas, Eduardo Oliveros, Neil Loughran, Bassem Nasser, "A METHODOLOGY FOR ENGINEERING REAL-TIME INTERACTIVE MULTIMEDIA APPLICATIONS ON SERVICE ORIENTED INFRASTRUCTURES ,", in Proceedings of the IADIS International Conference Applied Computing, Timisoara, Romania 14-16 October 2010.

[28] Mathworks NARX Network (narxnet, closeloop): http://www.mathworks.com/help/toolbox/nnet/ug/bss36dv-1.html

[29] Maria P. Menezes, Jr. and Guilherme A. Barreto. 2008. Long-term time series prediction with the NARX network: An empirical evaluation. Neurocomput. 71, 16-18 (October 2008), 3335-3343. DOI=10.1016/j.neucom.2008.01.030

[30] Flexiscale Cloud Computing Platform: http://www.flexiscale.com

[31] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, George Yin, "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration", 6th international conference on Autonomic computing, 2009

[32] Juan M. Tirado, Daniel Higuero, Florin Isaila, Jes´us Carretero, "Predictive Data Grouping and Placement for Cloud-based Elastic Server Infrastructures", 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011

[33] Giovanni Toffetti, Alessio Gambi, Mauro Pezzè and Cesare Pautasso, "Engineering Autonomic Controllers for Virtualized Web Applications", 10th international Conference on Web engineering, 2010

[34] Sara Casolari, Mauro Andreolini, Michele Colajanni, "Runtime prediction models for Web-based system resources", IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008

[35] Jiangtian Li, Xiaosong Ma, Karan Singh, Martin Schulz, Bronis R. de Supinski and Sally A. McKee, "Machine Learning Based Online Performance Prediction for Runtime Parallelization and Task Scheduling", IEEE International Symposium on Performance Analysis of Systems and Software, 2009

[36] Philipp Leitner, Branimir Wetzstein, Florian Rosenberg, Anton Michlmayr, Schahram Dustdar, and Frank Leymann, "Runtime Prediction of Service Level Agreement Violations for Composite Services", ServiceWave 2009

[37] Gerald Tesauro, Nicholas K. Jong, Rajarshi Das, Mohamed N. Bennani, "On the use of hybrid reinforcement learning for autonomic resource allocation", Springer Cluster Computing, 2007

[38] George Kousiouris, Dimosthenis Kyriazis, Spyridon V. Gogouvitis, Andreas Menychtas, Kleopatra Konstanteli and Theodora A. Varvarigou, "Translation of Application-level Terms to Resource-level attributes across the Cloud Stack Layers ,", Computers and Communications (ISCC), 2011 IEEE Symposium on , vol., no., pp.153-160, June 28 2011-July 1 2011 doi: 10.1109/ISCC.2011.5984009

[39] George Kousiouris,George Vafiadis and Theodora Varvarigou, "A Front-end Hadoop based Data Management Service for Efficient Federated Clouds", to appear in 3rd IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2011), Athens, Greece, Nov 29- Dec 1, 2011

[40] Mediawiki web-based wiki software application: http://www.mediawiki.org/wiki/MediaWiki

[41] OpenForecast, general purpose forecasting model library: http://www.stevengould.org/software/openforecast/index.shtml

[42] Open Cloud Computing Interface Working Group: http://occi-wg.org/

[43] Open Virtualization Format Specification, available at: http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.0.0.pdf

[44] Gregory Katsaros, George Kousiouris, Spyridon V. Gogouvitis, Dimosthenis Kyriazis, Andreas Menychtas, Theodora Varvarigou " A Self-adaptive hierarchical monitoring mechanism for Clouds, , Journal of Systems and Software, Available online 8 December 2011, ISSN 0164-1212, 10.1016/j.jss.2011.11.1043

[45] Luís Costa, "Modeling Interactive Real-time Applications on Service Oriented Infrastructures", 3rd IRMOS Public Seminar, available at: http://irmosproject.eu/Files/02_IRMOS_Oslo_Seminar_Modelling-PartI.pdf

[46] Bridges M, Heron EA, O'Dushlaine C, Segurado R, Morris D, et al. (2011) Genetic Classification of Populations Using Supervised Learning. PLoS ONE 6(5): e14802. doi:10.1371/journal.pone.0014802

[47] J. Li, An empirical comparison between SVMs and ANNs for Speech Recognition,April 05 2010 bhttp://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/li.pdfN.