Translation of Application-level Terms to Resource-level attributes across the Cloud Stack Layers

George Kousiouris, Dimosthenis Kyriazis, Spyridon Gogouvitis, Andreas Menychtas, Kleopatra Konstanteli and Theodora Varvarigou

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

Abstract— **The emergence of new environments such as Cloud computing highlighted new challenges in traditional fields like performance estimation. Most of the current cloud environments follow the Software, Platform, Infrastructure service model in order to map discrete roles / providers according to the offering in each "layer". However, the limited amount of information passed from one layer to the other has raised the level of difficulty in translating user-understandable application terms from the Software layer to resource specific attributes, which can be used to manage resources in the Platform and Infrastructure layers. In this paper, a generic black box approach, based on Artificial Neural Networks is used in order to perform the aforementioned translation. The efficiency of the approach is presented and validated through different application scenarios (namely FFMPEG encoding and real-time interactive e-Learning) that highlight its applicability even in cases where accurate performance estimation is critical, as in cloud environments aiming to facilitate real-time and interactivity.**

Keywords- Cloud computing, performance estimation, SLA translation, artificial neural networks, real-time Clouds

I. INTRODUCTION

Clouds have become a state of the art ICT solution in the recent years, as the next generation SOA-based [1] technology. In the most recent extension of this computing paradigm, the Software-Platform-Infrastructure (SPI) model [2], everything is offered as a service through suitable infrastructures [3]. The following discrete roles, as identified in [20], are included in this model (Figure I-1):

• Application Developer/Provider: an entity that transforms a regular application into a modular service-oriented one (Software as a Service - SaaS). Besides developing / adapting the application, this role also identifies which are the **application-specific** terms based on which the resources will be assigned and managed:

1. **Workload parameters**: what are the configurable settings with which a client wants to run a service (e.g. number of users, resolution of a streaming video etc.)
2. **Quality of Service (QoS) parameters**: what is the expected output of the service, that is used to express the QoS levels (e.g. response time, frames per second of a video transmission etc). This is also known as Key Performance Indicators (KPIs)

• Platform Provider: an entity, namely Platform-as-a-Service (PaaS) provider, that offers the framework through which SaaS can be executed on an Infrastructure as a Service (IaaS) provider. This role incorporates tasks such as application and performance modeling, enactment and monitoring of application execution, evaluation of events and triggering of corrective actions. In this process it may utilize information provided by the Application Developer for describing the software component (e.g. in XML format)

• Infrastructure Provider: an entity (e.g. [4]) that offers hardware resources, mainly through extensive use of virtualization. This is the Infrastructure-as-a-Service provider (IaaS).

• Consumer/Client: an entity that uses an application offered as a service. The consumer contacts the SaaS provider that has made this SaaS available and requests this software for a specific configuration (**application-level workload** parameters). Furthermore he/she demands **certain levels** of **Quality of Service** (QoS), as measured by specific Key Performance Indicators (KPIs) of the application. The PaaS provider then contacts an IaaS provider and requests **hardware resources** (based on hardware-level parameters) that will satisfy the needs of the consumer. After a successful negotiation, the application (SaaS) is deployed (by the PaaS) on infrastructures provided by the IaaS.

All the aforementioned actions are typically validated through Service Level Agreements (SLAs, [20]) between the involved parties. SLAs describe the agreement on these terms and conditions and legally bind the actors. Typically, there are two SLAs, one between the consumer and the PaaS provider, expressed in **application terms**, and one between the PaaS and IaaS providers, expressed in **resource level terms**.

In this complex and multi-layer ecosystem, one of the most critical steps is the role of the PaaS in **translating** the consumer-defined application parameters (workload and QoS, as these are expressed in the consumer-PaaS SLA) to resource level attributes (as these are needed to be expressed in the PaaS-IaaS SLA). However, the exchange of information between the entities located in different layers is difficult for technical and business reasons. On one hand, it is unlike 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 that competitors access 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, vCloud and OVF. 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 PaaS), the same translation issue exists, with the only difference that now the IaaS provider is responsible for solving it.

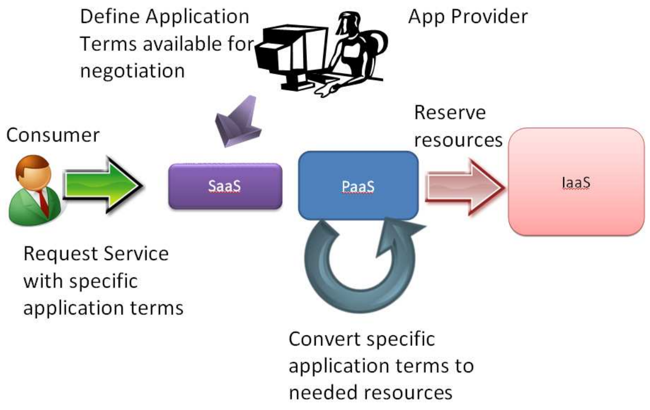
In this paper, a generic approach for performing this translation is presented, based on Artificial Neural Networks (ANNs). We present why ANNs can be applied in order to solve this problem, through **directly converting** the application specific SLA terms to resource attributes. A variety of case studies and different scenarios in which they are applied is presented. The application of this methodology is made on a **per software component** basis and with application-specific terms that vary depending on the examined scenario. Furthermore, different hardware configurations are taken under consideration, depending on what type of Cloud we are considering (**standard or real-time**). The major advantage of the proposed approach is the **limited amount of information** that needs to be passed from one layer to the other in the Cloud stack and the **generic** and **flexible** nature of ANNs that can adapt to different software components. Furthermore, their incorporation of both linear and nonlinear features aids in capturing more accurately the dependencies compared to other methods like multivariate linear regression, as also identified in [5].

In the remaining of the paper, relevant works to the one presented in this paper are listed and analyzed in II, while the core of the translation mechanism appears in III. Indicative case studies of the application of the approach are described in Section IV while detailed evaluation is contained in Section V. Conclusions are included in Section VI.

II. RELATED WORK

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 web service 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.


Define Application Terms available for negotiation
App Provider
Consumer
Reserve resources
SaaS
PaaS
IaaS
Request Service with specific application terms
Convert specific application terms to needed resources

II-1: Discrete roles in the Cloud Stack

Literature [7] proposes 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.

Another interesting work is presented in [8]. The application, whose performance must be measured, is run under a strict 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 is an ideal case 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. 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.

III. TRANSLATION MECHANISM

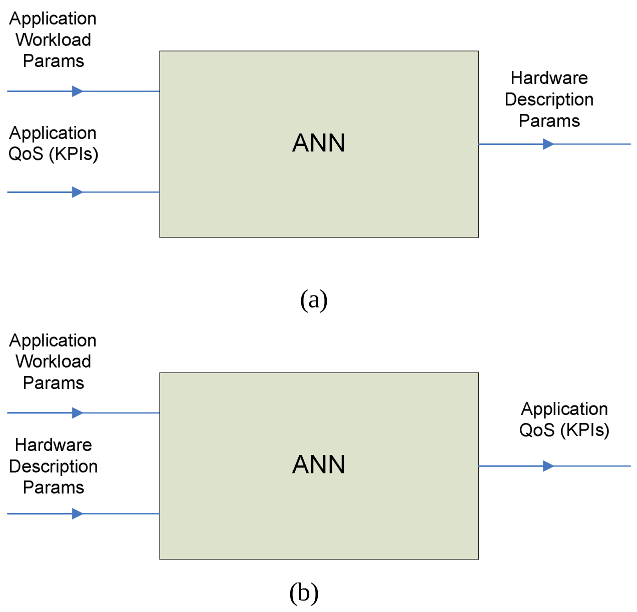
ANNs, due to the fact that they represent a **black box** approach, are perfect for usage in an environment where information is not easily relayed from one layer to the other, mainly for confidentiality purposes. For example, in the SPI model, the SaaS developer (Application Developer/Provider) is a 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 behavior. 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 SLA terms that are used between the client, the PaaS and the IaaS.

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.

A. 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 (in an off-line manner) 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 inputs and outputs of the ANN are determined through proper information provided by the Application Provider in an XML format. More details on this XML schema and the creation of the component descriptions can be found in [27]. 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 tradeoff. In Figure III-1, 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.

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 mean a significant reduction in service costs.


Application Workload Params
Application QoS (KPIs)
ANN
Hardware Description Params
(a)
Application Workload Params
Hardware Description Params
ANN
Application QoS (KPIs)
(b)

III-1: 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

B. *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 SLA between the PaaS and IaaS.

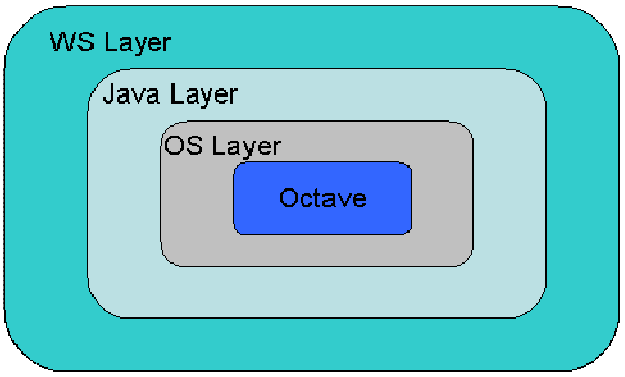
C. *Sample Dataset Creation*

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 IV, 420 different execution configurations were made, 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 III.B. This is necessary in order to avoid extrapolation and have more accurate results in the estimation process.

D. *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 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.



III-2: Layered Implementation of the service ([14])

IV. CASE STUDIES

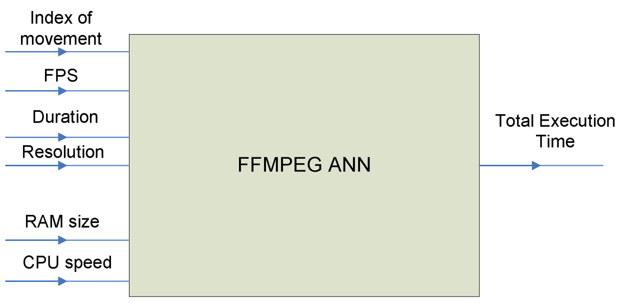
In order to portray the flexibility of the proposed approach, the following case studies are described. These include differentiations on the hardware attributes that are used by the IaaS providers for reserving resources and on the software components that need translation. Furthermore the alteration of the application-level parameters is investigated in order to observe the effect on the model creation.

A. *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 behavior 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, which are the main metrics used by general purpose Cloud providers like [4].

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 IV-1.



IV-1 ANN Model for the FFMPEG component

B. 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 elearning 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 too. 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.

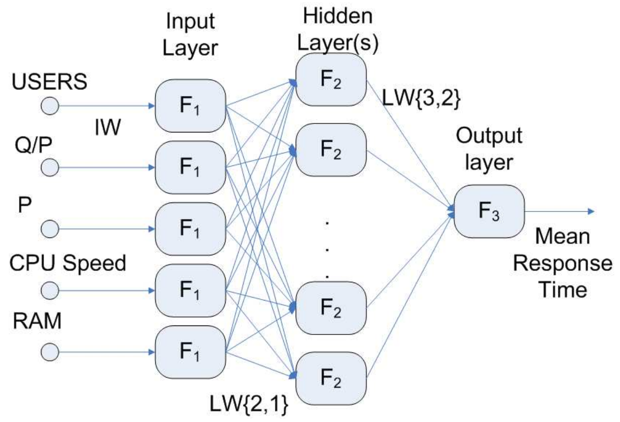
The ANN model used in this case has the aim of translating the consumer-side requested application parameters (in this case the number of users supported by the application) and the hardware resources used (Q/P, P, CPU speed and RAM) to QoS level characteristics (server mean response times and standard deviation of these times). A general form of an ANN as this is produced from software like Matlab or Octave appears in Figure IV-2.

1) Flexibility of the approach (Alteration of model inputs)

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.

So all we have to do is to change the input parameter, from “number of users” to “requests per interval”, preprocess 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 preprocessing 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 preprocessing of the available datasets (like in the case of “requests per interval”).



IV-2: Example structure of the e-learning ANN model for mean response time prediction. For the standard deviation prediction a different ANN model was created

V. EVALUATION

A. Accuracy and structure of the ANN model

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?

In order to investigate this factor, the ability of the networks to predict the performance of the software components was measured. In order to do so, 70% of the dataset acquired through the steps described in III.C was used in order to train the network models, for the applications described in IV. Then, the remaining 30% was used in order to check the accuracy of the networks in predicting the output QoS levels of the application. The Mean Absolute Error (MAE) is depicted in Table 1.

Table 1: Accuracy of the ANN models for the various case studies

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%

The above ANNs are feed-forward back-propagation networks, trained with the Levenberg-Marquardt algorithm [11]. The performance criterion was the mean square error in the training set and it was trained for 150 epochs. Training time is in the range of 1 minute, but this also depends on other factors like size of the training dataset, parameters to calculate etc. 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. More details regarding the ANN structure (weights and biases) follow for the mean response time prediction ANN. In general IW is the input weight matrix, LW{x,y} is the layer weight matrix from layer y to layer x and B{x} are the biases of each layer (as these are depicted in Fig. IV-2):

IW{1,1}=
[0.068042 -3.499239 7.922383 -0.104392 -0.827968
0.639215 0.835052 -0.109065 0.579030 0.296968
-0.375270 -0.665608 -0.258520 0.668154 -1.362266
0.484933 0.461851 3.391634 0.212285 0.029862
12.889114 -1.345095 0.292365 2.349104 -4.364500]

LW{2,1}=
[5.549738 -3.980910 -2.150672 -7.304653 5.629176
0.055164 -1.924175 -0.383149 -0.208614 0.377743]

LW{3,2}= [0.10758 0.80342]

B{1}= [-0.12486 1.33792 1.57375 0.93707 6.83621]'

B{2}= [-2.67678 -0.59151]'

B{3}= [-0.066518]

For the standard deviation ANN, the following matrices detail the weights and biases of the network:

IW{1,1}=
[-0.346696 -30.682570 21.815598 -1.051505 5.334884
-0.540538 17.712117 0.636773 0.760923 -0.773580
-12.470807 9.710534 -24.528460 0.072669 0.248244
5.297675 -15.156324 2.118186 -1.516121 5.218029
-71.191675 83.423849 0.565645 -0.955953 4.114581]

LW{2,1}=
[2.33630 -9.77996 5.33060 14.66727 -5.88447
1.65503 -16.04118 3.40684 0.66283 -2.83369]

LW{3,2}= [-2.1261 2.9647]

B{1}= [-4.48693 -0.24037 -1.81300 -5.60648 -4.87909]'

B{2}= [-5.6650 -3.8961]'

B{3}=[-0.96069]

B. *Robustness of the 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 1:

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

Equation 1

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 III.C 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 before creating the model and thus apply the suitable boundaries to the optimization of the ANN structure.

C. *Time performance of the ANN*

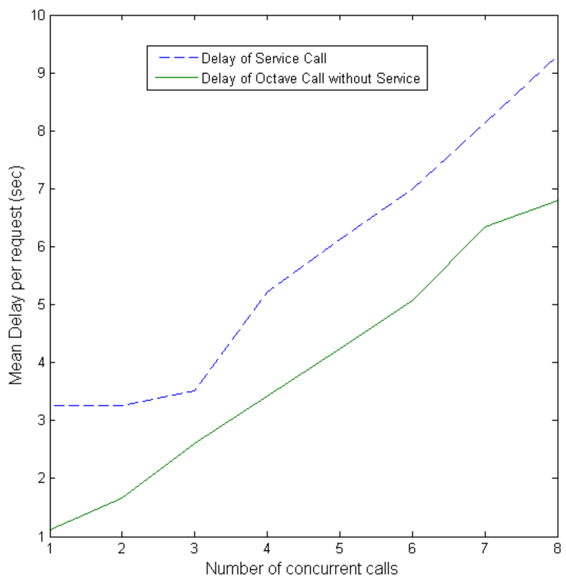
In Figure V-1, the delays for acquiring a response from the ANN model for a given input are shown, for a service oriented and a non-service implementation. As it can be seen from the graphs, the delay of the service layer adds around 2-3 seconds in the estimation process. Furthermore, the delays have an almost linear increase compared to the number of clients issuing requests for estimation. For all measurements the process was repeated for 500 times and the mean times were extracted, in order to gather a statistically robust sample.

The times measured in the figures are for a single request with a single input vector to the models. 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.

At this point it must be stated that if the consumer wants to change the workload that the service is able to handle, the IRMOS platform (in which the service presented in this paper is incorporated) offers renegotiation capabilities. Thus, during runtime, the application may trigger this feature, which in turn will request new predictions from the model in order to accommodate the new application requirements.

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 estimations	1.978 seconds



V-1: Delays for acquiring estimation (through service and direct octave call)

VI. CONCLUSIONS

The translation problem from application-level terms (workload and QoS) to resource-level attributes is a challenging problem 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. In this paper, a generic approach that is based on ANNs was used in order to efficiently be used as a mediator for the PaaS layer, in the latter's attempt to estimate how many resources are needed for an application instance with specific QoS levels to run on a Cloud.

The flexibility of the approach was demonstrated through various case studies with altering demands, it demonstrated significant accuracy and was proven robust. Furthermore, the time requirements for producing the estimations are within the logical time frames expected in a service oriented infrastructure, in both of the examined usage scenarios (standalone estimation or multiple-input estimation). The service call adds ~1 second delay (including xml processing etc.), the GNU Octave initialization 1 second while each estimation takes approximately 10 milliseconds. The needed amount of information from the other layers is limited to already available information regarding the inputs or outputs of the application and the numeric interval of this I/O. Furthermore, 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.

For the future, an aspect that is worth pursuing is the usage of the ANN models during the runtime of the application (and not just SLA negotiation), for dynamically adjusting the allocated resources. This approach, combined with a control loop, would improve the efficiency and automation of the platform.

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.

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 http://doi.acm.org/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 http://doi.acm.org/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 http://doi.acm.org/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.