A Service-Oriented Framework for GNU Octave-Based Performance Prediction[1]

George Kousiouris
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
gkousiou@telecom.ntua.gr

Dimosthenis Kyriazis
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
dkyr@telecom.ntua.gr

Kleopatra Konstanteli
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
kkonst@telecom.ntua.gr

Spyridon Gogouvitis
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
sgogouvitis@telecom.ntua.gr

Gregory Katsaros
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
gkats@telecom.ntua.gr

Theodora Varvarigou
National Technical University of Athens
9 Heroon Polytechniou str,
157 73 Zografou, Athens
+30 210 772256
dora@telecom.ntua.gr

ABSTRACT

Cloud/Grid environments are characterized by a diverse set of technologies used for communication, execution and management. Service Providers, in this context, need to be equipped with an automated process in order to optimize service provisioning through advanced performance prediction methods. Furthermore, existing software solutions such as GNU Octave offer a wide range of possibilities for implementing these methods. However, their automated use as services in the distributed computing paradigm includes a number of challenges from a design and implementation point of view. In this paper, a loosely coupled service-oriented implementation is presented, for taking advantage of software like Octave in the process of creating and using prediction models during the service lifecycle of a SOI. In this framework, every method is applied as an Octave script in a plug-in fashion. The design and implementation of the approach is validated through a case study application which involves the transcoding of raw video to MPEG4.

Innovation Area

Services Delivery Platform and Methodology

Keywords : Service Oriented Infrastructures, Performance Estimation, GNU Octave, Quality of Service, Service Delivery

1. INTRODUCTION

Service Oriented Architectures (SOAs) [1] refer to a specific architectural paradigm that emphasizes implementation of components as modular services that can be discovered and used by clients. Infrastructures based on the SOA principles are called Service Oriented Infrastructures (SOIs). A basic principle that governs the operation of such environments is that the relations between the SOA initiators and requesters are driven by Service Level Agreements (SLAs) [2]. This means that the roles of the parties involved in a transaction in a SOA-based environment (service provider and service customer), the relation itself and the relation details are defined by SLAs. In most cases, SLAs define the service that will be offered, the Quality of Service (QoS) parameters (expressed with specific terms) as well as other terms related to the involved parties (e.g. compensation in case of SLA violation). QoS provisioning and management in general is a very important and challenging field of research in distributed environments. In [3] an approach is presented for managing QoS in SOA that focuses on the categorization of the quality characteristics and uses a specific XML-based language for applications and service providers to express QoS requirements and contracts.

Taking into account that the QoS requirements submitted by the end-user are stated into the SLA, what is essential for the service providers prior to signing the specific SLA refers to an estimation of the resources needed to fulfill the user requirements for every application that is offered as a service from his/her platform. To this direction, performance prediction and modeling is considered of major importance since the service providers need to determine the resources in order to fulfill the QoS requirements of the application while at the same time resource utilization must be maximized.

[1] This work has been supported by the IRMOS project and has been partly funded by the European Commission's IST activity of the 7th Framework Programme under contract number 214777.

To this end, a number of methods have been implemented, from graph based ([4],[6] and [7]) to probabilistic ([8]) and machine-learning ones ([9][9]). In general, a model is built in order to predict the output from the input, usually by taking into account suitable parameters or by analyzing past historical data. Models can be based for example on linear functions, neural networks, classification and regression trees or statistical properties. However, in general there is no specific framework for implementing such methods, especially in a service-oriented environment. Useful and widely adopted tools, such as GNU Octave ([21][21]) make the implementation of such methods much easier than with standard programming languages such as Java or C++. Octave is an open source, community-driven high level numerical computation software, very similar to Matlab. It can be used for implementation of advanced methods like the aforementioned ones, used in the context of performance prediction or analysis. However the automated adaptation of software like Octave on distributed environments is far from perfected. A very thorough analysis on various schemes of this sort can be found in [17].

The major aim of this paper is to present the IRMOS project[24] Mapping Service, a service framework, in which all the aspects of the SOA paradigms will be taken into consideration and the resulting solution will be able to implement whatever estimation method as an Octave script. To this end, all functionalities that are required by such a service are addressed, from XML-based information extraction regarding critical information for an application, embedding of Octave software for use in the service lifecycle and up to retrieval of historical data from databases, a key feature of interest since all methods use the latter.

The service-oriented implementation approach that is presented is generic and can be applied in service oriented architectures. The layered architecture that has been followed for the implementation of the mechanism can be used for adopting every performance prediction technique that can be written in Octave and focuses on the decoupling of the framework from the method used. The latter is used as an Octave plug-in, which can be altered at any time, without further additions. Furthermore, due to the layer approach, the dependencies from specific OS or distributed computing software toolkit used are minimized.

The remainder of the paper is structured as follows: Section 2 presents related work in the field of service oriented performance estimation and enablement of mathematical software through Web/Grid services. The next section describes in detail the functionality of the service along with the specific supportive actions whereas Section 4 introduces the service layer approach and the per layer implementation. In Section 5 we present a case study based on the application of encoding raw video in MPEG4 format. In order to demonstrate and evaluate the operation of the implemented mechanism a set of simple approximation techniques such as multivariable regression and polynomial fitting, are used, in order to produce approximating functions that connect the application input characteristics (as predictors) with the execution end time of the application (as the QoS level offered by the SP). The aim of these methods is just to demonstrate the abilities of the framework and not to provide efficient performance predictions. Finally, Section 6 concludes the paper with a discussion on future research and potentials for the current study.

2. RELATED WORK

There are various approaches for service oriented performance prediction mechanisms that aim to estimate the resources needed for the execution of specific applications in the SOI realm or for offering mathematical software in distributed environments.

For example, in [12], the Network Weather Service includes a set of forecasters and sensors in a distributed environment, in order to retrieve data dynamically and apply them to modules that implement advanced prediction techniques. While ahead of its time, this implementation was heavily based on C language, thus making the implementation of the techniques used more difficult and time consuming.

A framework for incorporating QoS in Grid applications is discussed in [5]. In this paper, a performance model to estimate the response time and a pricing model for determining the price of a job execution are used. In order to determine whether the client's QoS constraints can be fulfilled, for each QoS parameter a corresponding model has to be in place. The main disadvantage is that Vienna Grid Environment (VGE) does not prescribe the actual nature of performance models. It specifies only an abstract interface for performance models, taking as granted that these models will be provided from analytical modelling or historical data. But analytical modelling in general requires a thorough knowledge of the application, in order to develop the equations that depict its performance. However, in the case of SOIs, this knowledge may not be available in the wanted level of detail.

In [14], the authors present GridSolve, which emphasizes the ease-of-use for the user and includes resource monitoring, scheduling and service-level fault-tolerance. In addition to providing Fortran and C clients, GridSolve enables scientific computing environments (such as Matlab) to be used as clients, so domain scientists can use Grid resources from within their preferred environments. This effort attempts to go the other way around and "gridify" applications such as Matlab and Octave, in order for them to be able to exploit distributed resources. This is also done for Maple in [15] and for a variety of computer algebra packages in [18]. While these works are significant, it is questionable how the produced services can be used in an integrated service oriented environment during the service lifecycle. They can be considered mainly as a successful effort to improve the performance of software like Octave through distributed infrastructures.

In [13], an architecture very similar to the design presented here is followed, in order to provide Maple-based mathematical web services. While very promising, this approach is not incorporated in a Grid/Cloud environment and focuses mainly on the offering of mathematical services. In the GENSS project [16], the main focus lies on the matchmaking techniques for advertisement and discovery of mathematical services, from a semantic point of view. The authors of [22] demonstrate the use of Octave for creating performance models for network interfaces based on queueing models, while in [23] a limited part of Octave is offered through Web Services interfaces for use of the signal processing functions by mobile phones.

The design presented in this paper aims at providing the service oriented framework for performance prediction methods in a generic way without requiring detailed knowledge of the internal parts of the infrastructure or the application, but only by creating an Octave script of the proposed method. The latter is especially important for use by people who may be experts in the performance estimation realm but are not directly involved with the issues and development of SOIs. Furthermore, the designed service can be used in real life automated environments, e.g. during the SLA negotiation process between a Service Provider and the client.

3. Mapping Service Functionality

In the IRMOS framework, which is a platform offering real time execution on service oriented infrastructures, in order for an application to be included in the Service Provider's list of enabled services, a number of steps must be implemented in order to adapt it to the framework. These steps are described in detail in [19]. In a nutshell, the application developer needs to create an XML description of his application, including a set of characteristics for its functional and non-functional requirements. This description is called Application Service Component Description (ASCD) in the IRMOS language. Afterwards, the Service Provider needs to go through the process of creating a number of mapping rules, that ,in IRMOS, are considered as the first level of performance prediction and are used for the mapping between user requirements, application characteristics and the resources needed in order to meet the QoS levels that will be exposed in an SLA. The aim of this paper is to describe the framework for the creation of these mapping rules/models, however the design and implementation that was followed is generic, thus enabling the Mapping Service to be used even outside the aforementioned platform. The overall platform architecture of IRMOS can be found in [25]. At this point it must be stressed that while the interests of the Application Developer and the Service Provider seem contradictory, this is not the case due to the value chain followed. From this point, the Application Developer adapts his application to the IRMOS platform, in order to be used by other parties (customers) , through the Software-as-a-Service model. So it is actually his/her interest to aid the platform to correctly predict the necessary resources, so that the application runs smoothly with the minimum resources, in order to be competitive in comparison to similar applications. Furthermore, there are more than one IRMOS platform providers, so for the same reason (competitiveness) the latter want to achieve accurate predictions.

In order to design such a service, one must take into account all the different functionalities that need to be implemented in order to have a fully operational platform component. The high level functions that must be provided are a) model creation based on the performance prediction method b) model exposure for the estimation process. Furthermore, these processes must be fully automatic in order to fit in the service oriented infrastructure.

3.1 Model Creation

For a model to be created, a number of steps must be implemented:

- Triggering of the service, including a specific identifier for the application whose model is going to be created
- Acquisition of information regarding the application that needs to be modeled (this includes predictors-inputs, estimated outputs, possible values of the parameters etc.)
- Acquisition of historical data from a database that are going to be used as the data set for the method
- Passing of the parameters to Octave
- Octave method script and its execution in the service lifecycle
- Storing of the rules for future use

One key point here is the extraction of information regarding the application. This can be done via an XML-based schema, which is instantiated by the application developer and contains critical information regarding the inputs and outputs of the software module. What is considered as critical information are the inputs and outputs of the application, mainly the inputs or characteristics that influence application performance and are regarded as predictors and the outputs that can be considered as the QoS metrics. Furthermore, for these elements, ranges of possible values must be identified or enumerations regarding their acceptable values. This is mainly done because in a variety of methods, normalization of the data into common intervals is needed. In order for the conversion to take place, the allowed values must be described in a descending or ascending order of importance. The schema used is described in [19]. To this end, the main service must be able to process the XML description and extract all the necessary info for the application.

From the previous steps, the necessary components can be identified. These include the core service (Mapping Service) which is responsible for the coordination of the entire process and the main processing tasks. The Application Description Repository contains the XML descriptions of the components, while the Historical Database is used to store data from previous executions. These may originate from already existing historical data or from benchmarking. Finally, the Model Repository contains the stored rules after the finalization of this operation (Model Creation) The sequence diagram for the CreateModel operation of the service appears in Figure 1. The interfaces that need to be implemented after the triggering of the service by the external world (in our case the other services of the IRMOS Framework) are (1) the retrieval of the XML description from the application repository, (2) the retrieval of the historical data from the Database and (3) the storing of the model functions in the Model Repository. For the internal operations, the processing of the XML description and the actual creation of the models are needed.

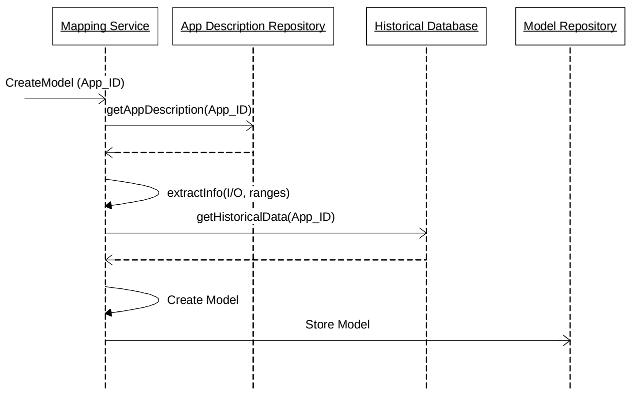

Figure 1: Sequence Diagram for CreateModel Operation

This phase is envisioned to be included during the application adaptation to the service oriented platform, for creating the models of the application component. Whether this can be included directly in the runtime negotiation mainly depends on the implemented method and if it can create models on the fly in a logical time interval. In many cases these methods are time consuming, involving costly numerical computations. This time is deteriorated by the fact that Octave is slower than traditional programming languages. However given its diversity and ease of implementation and the fact that the models do not need to be created on the fly during negotiation but in an earlier (in any case automated) phase, this delay is not a hindering factor.

.Furthermore, while it is generally considered as an interactive environment, Octave can also be run via command line, in an automated fashion, thus enabling its variety of tools and functions to be used in a SOA approach.

3.2 Model Usage

For a model to be used, a number of steps must be implemented:

- Request for an estimation based on the model predictors (such as application inputs/characteristics/)
- Acquisition of information regarding the application (this includes inputs, outputs, possible values of the parameters etc.)
- The model is retrieved from the repository and run with the current set of predictors
- The response is passed back to the Service Provider

The sequence diagram of this phase appears in Figure 2. Interfaces must be implemented towards the external framework (in this case the IRMOS framework described in [25]) for accepting estimation requests and towards the Model Repository for retrieving during runtime (in SLA negotiations for example) the models. Also, a similar to the previous phase interface towards the Application Description Repository is necessary in order to obtain the ranges of the inputs for normalization purposes. The main internal operation, other than the XML extraction described previously, is to execute these models through proper Octave scripts in order to get the estimated response that is passed back to the client.

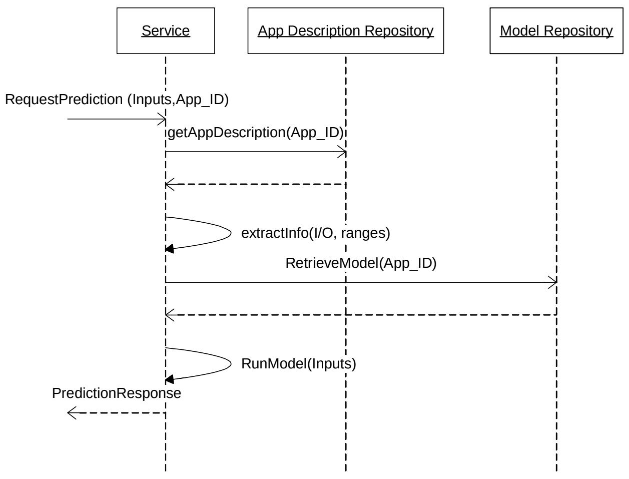

Figure 2: Sequence Diagram for RequestPrediction Operation

This phase is in all cases lightweight and is used during runtime negotiations due to the fact that the model is already created during phase 1 and the time to execute it is negligible (for example it may involve the appliance of the input parameters to a model function in order to estimate the output based on a mathematical function).

4. Service Oriented Implementation

The different layers in the service oriented architecture of the estimation mechanism are depicted in Figure 3. The reason for having these layers is because there are software solutions (such as Octave and Matlab) for easily implementing various estimation methods, but in order to pass information and arguments from the client to the bottom layer it was considered safer to include intermediate steps that would minimize the interdependencies from one layer to the next. This also increases the decoupling between the different layers, making it possible to change at any point one layer, with minimum interventions to the rest of the components. For example, for OS, the main element affected is the Shell script and for Octave a few commands in the Java core class relating to the passing of the dataset and the execution command in the shell script.

The resulting implementation is also generic and fully decoupled from the underlying algorithm, since at any time the developer may switch the methods of estimation by simply replacing the underlying scripts (A, B, C or D). In our case, with the use of such a decoupled solution, it was feasible to easily implement the mechanism with more than one method as mentioned in Sections 4.4 and 5. These methods can, at any time, be increased or combined in order to have an even more robust estimation approach. Following, details are presented with regard to each layer.

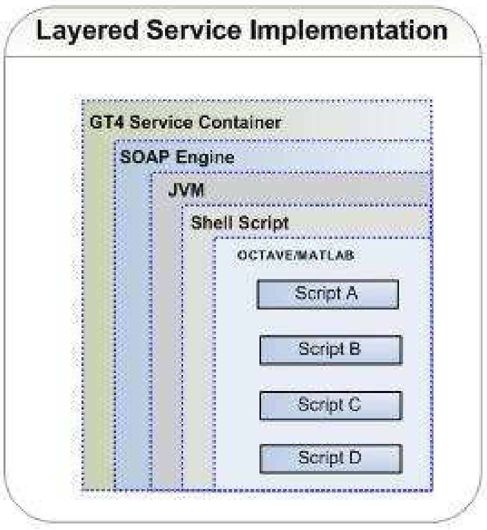

Figure 3: Layered Implementation of the Service

4.1 Middleware Level (GT4)

The service-oriented version of the mechanism presented in this paper was implemented using the Globus Toolkit version 4 (GT4) and runs under the standalone container that the toolkit offers. GT4 [11] is an open source Grid middleware that provides the necessary functionality required to build and deploy fully operational Grid services. It implements the WS-Security and other specifications relating to security, as well as the Web Services Resource Framework (WSRF), WS-Addressing and WS-BaseNotification specifications.

The WS interface in front of the mechanism acts as a gateway offering a single-point of access to its functionality while at the same time “hiding” its details and complexity from the clients. In more detail, the client of the service sends a SOAP request to the operation exposed by the WS interface. The SOAP message includes the input needed, i.e. the ID of the application software module for which a model needs to be created. This message is captured by the SOAP Engine running within the GT4 container which serializes the included information into Java objects before passing it to the service’s core Java class for further processing.

The operations described in Section 3 are exposed to the outside world through standard WSDL descriptions.

4.2 Connecting Level (Java Core)

The connecting component (Java class) that is in the center of the framework is the most critical part. It is responsible for taking the ID of the application component whose model needs to be created, retrieving the XML description (ASCD) of the Application Service Component and processing it. This processing is performed based on the tag names inside the ASCD, regardless of the depth level where the information is stored. This aids the developer in decoupling the structure of the ASCD from the rest of the framework, with only dependence the tag names. Based on the information extracted from the ASCD, the SQL interface is enabled and the data from the Historical Data Repository (MySQL DB) are retrieved and normalized, by taking into consideration ranges or enumerations of the predictors and parameters that were previously extracted from the XML description. The transformation of the data set into normalized values in a common predefined interval is necessary for internal processing reasons of the various estimation methods. For the ranges, this is quite easy, taking only the minimum and maximum range declared in the ASCD. For the enumeration it was considerably more difficult, since the list of possible values is dynamic for each case of inputs, so proper assignment mechanisms had to be implemented. These mechanisms are deployed in both phases of the service (CreateModel and RequestPrediction) due to the fact that the inputs received during the second must again be normalized in order to be used in the resulting model functions of the first phase.

Then the Java2Octave interface takes advantage of a suitable library [20] for storing these data as Octave readable data files. Afterwards the Java2System interface executes the shell script, which in turn initializes the Octave environment which is responsible for actually creating the models, after reading the dataset.

Thread locking mechanisms have been implemented during this stage so that overlapping requests to create models do not interfere with one another. The reason for not using this Java2Octave interface to also execute the Octave scripts is that the Octave engine that is provided in the library does not support the use of extra packages that are very helpful in Octave. Afterwards, a Java2System interface is executed in order to initialize the Octave method script execution as a system command (shell script). The most critical part in this process is to effectively capture the I/O produced by the system command so that the system does not halt. This manipulation is also achieved through proper use of threads for capturing input, output and error streams. These streams are transmitted back to the client, so that the progress of the process is observed. The time that Octave can execute in order to produce the model is bounded by the WS request timeout and is configurable.

When the execution finishes, the output of Octave, model, function etc., is transferred to the Rule Repository, so that it can be retrieved in the future for the Model Usage phase. The internal structure of the service appears in Figure 4.

4.3 OS level (FC9)

For the OS level layer, Fedora Core 9 was used. The main functionality of this layer, as seen in the introduction of this Section is to enable the decoupling of each layer from the previous. In terms of functionality, the shell script used is mainly for launching the Octave environment, plus a number of necessary secondary actions such as directory manipulation.

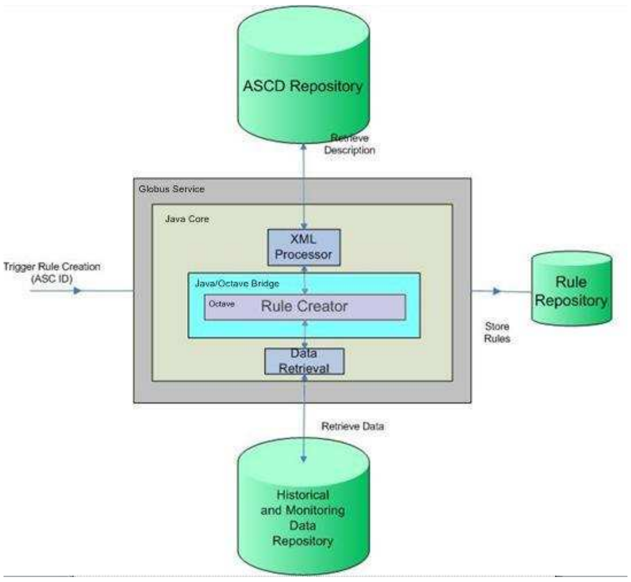

Figure 4: Mapping Service Internal Structure

4.4 Numerical Software Level (GNU Octave)

For the core implementation of the estimation methods GNU Octave [21] was selected. Octave is an open source, community-driven scripting language. It includes a number of add-on packages, in addition to the main version, that significantly aid in the implementation of advanced algorithms more efficiently and in a more abstract way than standard programming languages such as C and Java. Its scripting style is very dense and helps in the easier and faster implementation of complicated methods. It also provides built-in functions, that either alone or in combination can be used to create performance models of an application. Other solutions such as Matlab were also investigated, but were abandoned due to the fact that while the latter supports in many cases the compiling of the scripts into executables, a step which is necessary for the automated execution of the methods in the presented service oriented context, this does not happen for all available functions, thus limiting the underlying algorithms. Octave on the other hand can be run from the command line in all cases.

In our case, due to the fact that the main focus of this paper is not to investigate efficient performance prediction methods, simple built-in functions were used that implemented the multi-variate linear regression and polynomial fitting through very simple commands.

Two Octave scripts were used, one for the creation of the models (in the form of mathematical rules or algebraic functions connecting the predictors with the estimated output) and one for the execution of them with the according inputs during the runtime estimation phase. During this phase, the inputs of the predictors were applied to the estimating functions, in order to receive the prediction of the outputs.

5. Case Study

In order to validate the functionality of the proposed approach in terms of efficiency and ease of use, an application was selected for predicting its QoS level. The methods used were simple, since their main usage was to evaluate the framework and not actually predict with accuracy. As predictors, specific characteristics of the application were chosen, that in the IRMOS framework are described in advance by the application developer and included in the ASCD, as described in [19]. The identification of the predictors lies on the latter, but they do not require extensive knowledge of the internal code of the application, only experience gained from using the component. The developer does not need to be an expert on the application or on performance prediction to identify them.

5.1 Application

The application that we used in order to measure the ability of the methods was the encoding of raw video in MPEG4 format with the FFMPEG encoder [10]. In order to predict the execution time of the encoder (which is regarded as the QoS level offered in the SLA- how fast the encoding service will finish), four parameters of the input video which have a direct effect on the execution time were considered as predictors. 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). The method selected was multiple linear regression through least squares fit, which in Octave is given through only one command (**regress**). In Figure 5, the XML description appears, based on the IRMOS schema used. From this description, the Mapping Service is able to process and understand that the model should have four inputs and one output, along with the necessary intervals for the normalization.

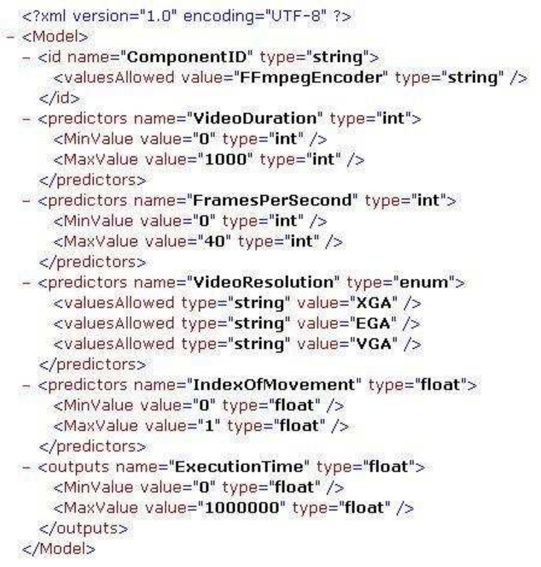
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <Model>
  - <id name="ComponentID" type="string">
      <valuesAllowed value="FFmpegEncoder" type="string" />
    </id>
  - <predictors name="VideoDuration" type="int">
      <MinValue value="0" type="int" />
      <MaxValue value="1000" type="int" />
    </predictors>
  - <predictors name="FramesPerSecond" type="int">
      <MinValue value="0" type="int" />
      <MaxValue value="40" type="int" />
    </predictors>
  - <predictors name="VideoResolution" type="enum">
      <valuesAllowed type="string" value="XGA" />
      <valuesAllowed type="string" value="EGA" />
      <valuesAllowed type="string" value="VGA" />
    </predictors>
  - <predictors name="IndexOfMovement" type="float">
      <MinValue value="0" type="float" />
      <MaxValue value="1" type="float" />
    </predictors>
  - <outputs name="ExecutionTime" type="float">
      <MinValue value="0" type="float" />
      <MaxValue value="1000000" type="float" />
    </outputs>
  </Model>
```

Figure 5: XML Description for the FFMPEG Encoder

During the experiments it was noticed that the size of the raw video could be correlated with the total execution time for the encoding. This is logical since 3 out of 4 parameters have a direct relation to the size (fps, duration and image resolution). In order to further test the framework, single variable approximation techniques were used that correlated the size of the video with the execution time. In this case the observed error is due to the fitting error of the curve to the experimental data along with the lack of consideration for the index of activity of the video. For the single variable approximation, a number of functions were considered: linear, quadratic, and higher order polynomials up to the 5th degree. For all these methods to be implemented, one command was used (**polyfit**) inside a for loop that changed the order of the fitted polynomial in each iteration.

In the experiments, the platform and the conditions of execution (CPU load) were kept the same. This is a simple approach, but the main focus was to validate the framework From the acquired data set, about 70% was used for the creation of the functions and the remaining 30% was used for the validation of each method's accuracy. Automated division in the required subdatasets is performed through the Octave command **subset**.

5.2 Results

The results from the applied methods appear in the following table (Table 1). In this, all the aforementioned variations are presented, along with the mean absolute error and the maximum absolute error of estimation.

Method	Mean Absolute Error of estimation (%)	Max Absolut Error of estimation (%)
Linear	10.46	20.97
Quadratic	27.84	91.64
Cubic	8.11	14.84
4th degree polynomial	8.16	13.01
5th degree polynomial	15.4	42.51
Multivariate Linear Regression (outliers removed)	15.5	34.5

Table 1: Comparative results of function approximation techniques

For higher order polynomials the results deteriorated and were similar to the ones for the quadratic case of fitting. In Figure 6, a graphical representation of the % error of each method for every validation case is presented.

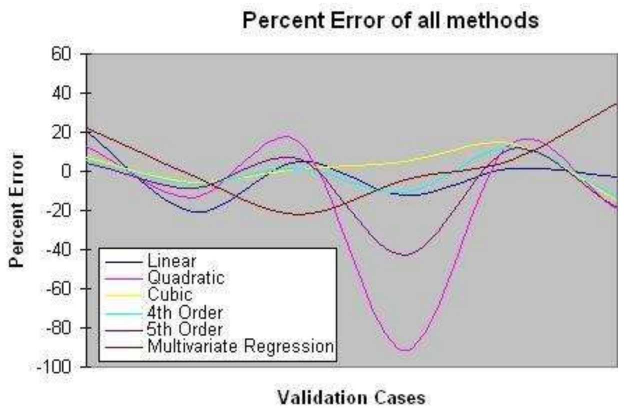

Figure 6: Comparative graphs of the different methods in terms of % error

While the end estimation results are not the primary concern, the performance of the cubic and 4th degree polynomials is promising, as a low-cost, easy to implement method. In any case, the framework was able to extract the necessary models through very small octave scripts. These scripts at any time may be replaced by more sophisticated algorithms and methods, without affecting the remaining components.

6. Conclusions

In this paper we have presented a service oriented framework for implementing Octave based performance prediction methods. This framework can be applied in a real life production environment, in order to aid in the SLA negotiation process between consumers and service providers in SOIs. In this context the proposed framework is generic and loosely coupled, so that at any time specific layers can be removed and replaced by more advanced ones. The performance estimation methods have been reduced to Octave level scripts, a fact that aids in abstracting the service oriented aspects from the actual estimation processes.

For the future, one goal to pursue is to implement more advanced methods through Octave scripts and test their performance in the service oriented context. Furthermore, due to the fact that now the methods can be applied easily and that Octave offers a large set of statistical tools, the combination of different methods for the improvement of the overall estimation is worth investigating.

Concluding, SOIs have not yet adopted a specific approach for performance estimation and modeling although a set of approaches have been presented in the research community. In that rationale and based on the reviewed literature, the essence of such mechanisms is shown, the ease of use and incorporation of which is considered to be of major importance for efficient service provisioning.

7. ACKNOWLEDGMENTS

This work has been supported by the IRMOS project and has been partly funded by the European Commission's IST activity of the 7th Framework Programme under contract number 214777.

8. REFERENCES

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

[2] Debusmann, M.; Keller, A., "SLA-driven management of distributed systems using the common information model," Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on , vol., no., pp. 563-576, 24-28 March 2003.

[3] G. Wang, A. Chen, C. Wang, C. Fung, and S. Uczekaj, “Integrated Quality of Service (QoS) Management in Service-Oriented Enterprise Architectures”, In Proceedings of the 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC), Monterey, California, September 2004.

[4] 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

[5] Siegfried Benkner, Gerhard Engelbrecht, "A Generic QoS Infrastructure for Grid Web Services," aict-iciw, p. 141, Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06), 2006

[6] Peer Hasselmeyer, Bastian Koller, Lutz Schubert, Philipp Wieder: Towards SLA-Supported Resource Management. HPCC 2006: 743-752

[7] Jarvis, S. A., Spooner, D. P., Keung, H. N., Cao, J., Saini, S., and Nudd, G. R. 2006.Performance prediction and its use in parallel and distributed computing systems.Future Gener. Comput. Syst. 22, 7 (Aug. 2006), 745-754.

[8] Oana Florescu, Menno de Hoon, Jeroen Voeten, and Henk Corporaal. Probabilistic modelling and evaluation of soft real-time embedded systems. In Proceedings of SAMOS VI, LNCS 4017,2006.

[9] Singh, K., İpek, E., McKee, S. A., de Supinski, B. R., Schulz, M., and Caruana, R. 2007. Predicting parallel application performance via machine learning approaches: Research Articles. Concurr. Comput. : Pract. Exper. 19, 17 (Dec. 2007), 2219-2235.

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

[11] http://www.globus.org/toolkit/

[12] R Wolski, N Spring, J Hayes “The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing - Future Generation Computer Systems, 1999

[13] E. Smirnova, C.M. So, S.M. Watt, Providing mathematical Web services using Maple in the MONET architecture. In Procs. MONET Workshop (2004)

[14] YarKhan, A., Dongarra J., Seymour K., 2007, in IFIP International Federation for Information Processing, Volume 239. Grid-Based Problem Solving Environments, eds. Gaffney, P. W.. Pool, J.C.T., (Boston: Springer), pp. 215–224.

[15] http://www.hpcgrid.com

[16] http://genss.cs.bath.ac.uk/index.htm

[17] Petcu, D. 2006. BetweenWeb and Grid-based Mathematical Services. In Proceedings of the international Multi-Conference on Computing in the Global information Technology (August 01 - 03, 2006). ICCGI. IEEE Computer Society, Washington, DC, 41. DOI= http://dx.doi.org/10.1109/ICCGI.2006.52

[18] A. Al Zain, K. Hammond, P.W. Trinder, S.A. Linton, H.-W. Loidl, and M. Costanti. SymGrid-Par: Designing a Framework for Executing Computational Algebra Systems on Computational Grids. In Proc. International Conference on Computer Science (Workshop on Practical Aspects of High-level Parallel Programming), Beijing, China, May 27-30, 2007, 2007

[19] IRMOS Project: “Irmos Application Blueprint”, December 2009

[20] http://kenai.com/projects/javaoctave/downloads

[21] http://www.gnu.org/software/octave/

[22] Paola Laface, Damiano Carra and Renato Lo Cigno, “A Performance Model for Multimedia Services Provisioning on Network Interfaces”, in Lecture Notes in Computer Science, Volume 3375/2005, Springer Berlin / Heidelberg

[23] Roger J. Castaldo Michael A. McKay Vladimir Tosic, “Exposing GNU Octave Signal Processing Functions as Extensible Markup Language (XML) Web Services”, in Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on

[24] http://www.irmosproject.eu/

[25] IRMOS Project D3.1.2: “IRMOS Overall Architecture”, NTUA and other partners, February 2009