Paper

Quality Aware Virtual Service Delivery System

Mariusz Fraś and Jan Kwiatkowski

Wrocław University of Technology, Wrocław, Poland

Abstract—The problem of providing support for quality of service (QoS) guarantees is studied in many areas of information technologies. In recent years the evolution of software architectures led to the rising prominence of the Service Oriented Architecture (SOA) concept. For Web-based systems there are three attributes that directly relate to everyday perception of the QoS for the end user: availability, usability, and performance. The paper focuses on performance issues of service delivery. The architecture of Virtual Service Delivery System (VSDS), a tool to serve requests for synchronized services is presented. It is proposed suitable monitoring technique used for estimation of values of service parameters and allocation of communication and execution resources by means of service distribution. The paper also presents results of experiments performed in real environment that show effectiveness of proposed solutions.

Keywords—quality of services, service virtualization, service request distribution.

1. Introduction

For Web-based systems there are three attributes that directly relate to everyday perception of the quality of service for the end user: availability, usability, and performance. The performance issues of information systems are very widely explored in different contexts. For SOA-based systems solutions concerning the quality of services have been generally developed in the context of Web services, usually proposing useful standards for quality of service mechanisms, such as WS-Policy [1] and WSLA [2]. To support the quality of service delivery some selection of service algorithms are also proposed. For example in the work [3] the service selection based on utility function on attributes assigned to services (such as price, availability, reliability and response time) has been proposed. Most of these works assume that values of service parameters does not change dynamically.

On the other hand the quality of services can be considered in the context of the quality of the resource utilization. Among the others the virtualization is already being used as a common and proven way to decrease the overall hardware needs and costs, however still the hardware utilization is around 20% and storage utilization does not go above 60% [4]. Using virtualization gives very promising results, but as stated in [5] it is still not enough. Virtualization stopped and is not pushing forward. Mission critical services are used as before due to the easier maintenance, controlling and monitoring. What is more, reduced budgets made it much more complicated for real virtualization adaptation since - especially at the beginning – costs of implementation are higher than those of keeping everything as is.

In the paper the Virtual Service Delivery System (VSDS), a tool for efficient allocation of communication and execution resources to serve requests for synchronous services and service monitoring during its execution is presented. The service requests are examined in accordance to the SOA request description model. The functional and non-functional requirements in conjunction with monitoring of execution of services and communication links performance data are used for requests distribution and for resource allocation. At the lower layer virtualization is used to control efficient resource allocation to satisfy service requests.

The paper is organized as follows. Section 2 briefly describes the main ideas used during designing and developing presented in the paper Virtual Service Delivery System. In the Section 3 the main service quality issues are discussed. The architecture and functionalities of the VSDS are presented in Section 4. Section 5 describes the ways how service monitoring and evaluation the values of service parameters is done by the Broker and Virtual Server Manager (VSM), two components of VSDS. In the next section results of the first experiments performed on the implemented system are presented. Finally, Section 7 outlines the work and discusses the further works.

2. The Concept of Quality Aware Service Delivery

The concept of effective and quality-aware infrastructure is based on the idea of Virtual Service Delivery System capable to handle client's requests taking into account service instance non-functional parameters.

The main components of the system are network service broker (further called Broker) and Virtual Server Manager. They are built as a component of a SOA. The main assumptions for operation of both modules are:

- the Broker delivers to clients the set of J services (so called atomic services) $as_j, j \in [1,J]$,
- the Broker knows execution systems $es_m, m \in [1,M]$, where real services (service instances) are available,
- the Broker monitors execution of client's requests and collects the monitoring data,

– the Broker acts as a service proxy – it hides real service instances, and distribute client's requests for services to proper instances according to some distribution policy,

– the VSM is responsible for creation of service instances,

– the VSM is responsible for service execution,

– the VSM offers the access to hypervisor actions that is independent on any used virtualization system by using *libvirt* toolkit,

– the VSM offers information about particular physical servers as well as running virtual service instances.

– the VSM is responsible for monitoring of executed services and execution environments (servers) including running of servers virtual machines.

The Broker implements the Virtual Service Layer (VSL). The VSL (Fig. 1) virtualizes real services available on service execution systems (servers). The VSM manages virtualized computational resources. Both layers are defined as the tuple $< ES, CL, AS, IS >$. $ES = \{es_1, \ldots, es_m, \ldots, es_M\}$ is the set of execution systems es_m, where: $m \in [1, M]$, M – the number of execution systems. The execution systems can be placed at different geographic locations. $CL = \{cl_1, \ldots, cl_m, \ldots, cl_M\}$ is the set of communication links cl_m from the Broker to execution systems. The Broker delivers the set of J atomic services $AS = \{as_1, \ldots, as_j, \ldots, as_J\}$. Each atomic service as_j available at the Broker is mapped to one or more known instances that form instance subset IS_j. Instances of given atomic service can be localized at different execution systems es_m. $IS = \{IS_1, \ldots, IS_j, \ldots, IS_J\}$ is the set of all instances of services, where: IS_j is the subset of instances of service as_j, $is_{j,m}$ is the m-th instance

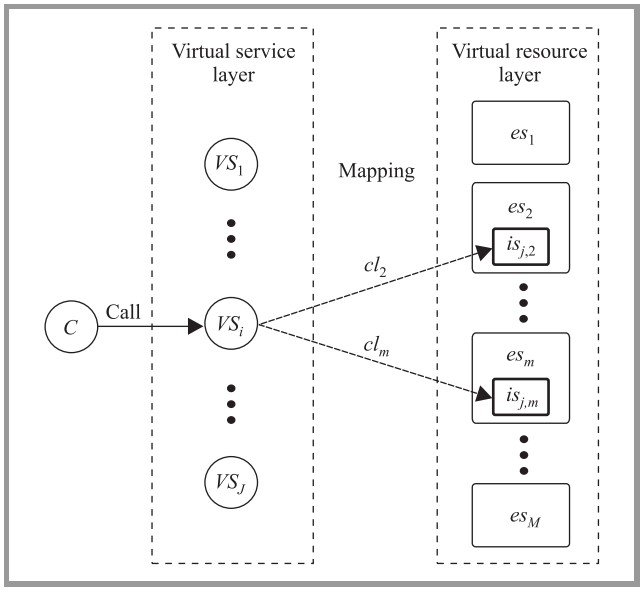


Fig. 1. The layers of Virtual Service Delivery System.

of j-th service as_j localized in given execution system and M_j is the number of instances of j-th service.

The real services are hidden from client point of view. The Broker advertises virtual services VS_j in accordance with SOA paradigm, and handles client's request for services. The client deals with virtual service (virtualized atomic service as_j) that can be executed at different locations.

The Broker collects essential data about service execution. It also monitors values of parameters of execution environment, i.e., communication links cl_m and execution systems es_m. The main advantage of virtualization of services is that according to values of service instance parameters some quality based policy of service delivery can be applied. The client of the system C calls the Broker for a service, and the Broker distribute the request to one, chosen service instance to ensure proper values of service quality parameters.

At the VRL the management of available execution systems at the lowest level is performed. VSM that implements VRL is responsible for efficient allocation of execution resources to services using virtualization techniques [6]. There are two aims of using VRL, which can, and in most cases would be, mutually exclusive. First of all the manager shall provision the instances of services with proper resources to ensure the fulfilment of requirements for requested service. Secondly it shall increase the utilization of the available resources, so that overall capacities are used to the highest possible degree. Managing the resources can be described in three distinct steps: provisioning of resources, adjusting and freeing the resources.

The largest difference between VSM and other similar solutions is coming from another targets standing behind our proposition. While most of other solutions are strictly devoted to manage the infrastructure, VSM is devoted to properly dispatch the requests, placing the virtualization management on the second place. Nonetheless one can point a number of similarities starting from common modular architecture with possibilities to customize the software easily. Furthermore just like other solutions *libvirt* is used to overcome the problem with communication with various hypervisors.

The role of VSM as a dispatcher means that some of the functionalities are redundant. Under such situation one may put offering Amazon compatible API, billing integration, number of control panels and so on. On the other hand the functionality is extended to understand the SOAP messages, identify which services are capable of performing them and finally running those services and dispatching the requests. Analogical software is found as an addition on top of OpenNebula and offers service orchestration and deployment or service management as a whole [7].

The instances of given atomic service are functionally the same and can differ only in the values of non-functional parameters $\psi(is_{j,m}) = \{\psi^1_{j,m}, \ldots, \psi^f_{j,m}, \ldots, \psi^F_{j,m}\}$, where $\psi^f_{j,m}$ is f-th non-functional parameter of m-th instance of j-th atomic service. Two kinds of service parameters may be distinguished: static parameters - constant in long period

of time (i.e., service price), and dynamic parameters – variable in short period of time, e.g. the completion time of execution of the service instance may be the case. From the client point of view, the very important service parameter is response time, which is usually quite variable parameter. In the network environment it consists actually of two components: data transfer time and execution time on the processing server (later called execution time for short).

3. Service Quality Issues

In order to assure proper quality of service delivery the three requirements are to be considered: effective and suitable service parameters monitoring and estimation, proper service request distribution according to current service parameters estimation and resource utilization, and service execution resources management.

The quality of network services depends on communication link properties and effectiveness of request processing on the server. Both affect the quality of each service instance separately and can be expressed by values of service instance non-functional parameters. To satisfy requested service parameters distribution of the request to proper service instance must be performed. The problem of service request distribution can be stated using criterion function Q

$$is_{j,m^*} \leftarrow arg_m(\psi^1_{j,m}, \ldots, \psi^f_{j,m}, \ldots, \psi^F_{j,m}). \qquad (1)$$

It is the task to select such instance is_{j,m^*} to serve request for service as_j that criterion Q is satisfied. In the particular case it is the task of finding extreme of the criterion function.

To satisfy proper service instance selection the current values of each instance parameters should be known. This requires methods of estimation and/or forecasting of values of such parameters. The Broker uses two approaches: statistical methods based on time series analysis and method based on artificial intelligence approach – using fuzzy-neural network [8], [9] and monitoring of parameters characterizing execution environment. For both approaches the estimation of values of previous executions is required what is described in the subsequent sections.

On the basis of forecasted and/or monitored values of parameters several approaches to service distribution algorithms can be adopted. Generally, the fully controlled environment case and not fully controlled environment case can be distinguished. The first one refers to the use of dedicated links and VSMs in all execution systems. The second one is when there is no full control of communication links. It is the most common condition for delivery of service in the Internet according to SOA paradigm.

For the VSDS the best effort based algorithms for the uncontrolled environment are implemented by now. Two most commonly considered service parameters are used: data transfer time and completion time of service execution in the processing server. The selection of service instance for requested service as_j is performed according to criterion (2)

$$is_{j,m^*} \leftarrow arg_m min(T^{j,m}_{PROCESS} + T^{j,m}_{TRANSFER}), \qquad (2)$$

where $T^{j,m}_{PROCESS}$ is execution time of service instance $is_{j,m}$ and $T^{j,m}_{TRANSFER}$ is data transfer time for fulfilling request for service instance $is_{j,m}$.

The distribution algorithms that takes under consideration the completion time of service execution require estimation and forecasting of values of these parameters. As mentioned above, the Broker uses time series analysis based forecasting or fuzzy-neural network based forecasting shortly described later.

4. The VSDS Components

The two main VSDS components, the Broker and Virtual Server Manager, are built as a components of a SOA-based system. The architecture of VSDS is very flexible and gives opportunity to compose the processes from services publicly or privately available.

The Broker handles SOAP requests for services. The virtual services provided by the Broker are described using WSDL (Web Service Definition Language) standard and are published in accordance to SOA paradigm.

The client's requests are analyzed and checked versus the information about possible places of execution as well as values of non-functional parameters of service execution at each location. In order to support evaluation of values of service instance parameters the Broker performs active monitoring of execution environment, i.e., values of parameters of communication links to execution systems (servers) and server state parameters. Execution system state monitoring is done with use of SOAP messages.

The above functionality is performed by the following modules of the Broker (Fig. 2):

- Controller – the main control unit performing service request distribution. It makes the decision on the basis of values of service instance parameters derived from Estimator/Predictor module;
- Service Monitor – monitors the execution of services at the TCP session level, and records the values of executed service parameters;
- Environment Monitor – makes active measurement of values of execution environment parameters – the server state and values of communication link parameters;
- Estimator/Predictor – the module which estimates values of essential parameters characterizing the service and instances with use of TCP session level data, and performs prediction of values of parameter on the basis of historic data and current values of environment parameters.

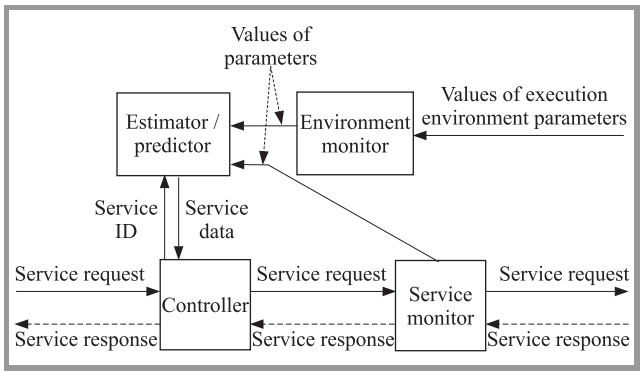


Fig. 2. The architecture of the Broker.

VSM offers two interfaces to interact with the virtualized environment. One is XML-RPC based that is used mainly for communication between internal VSM modules. More important is the possibility to direct SOAP calls to services to be handled by the VSM. Each and every request is then redirected to proper service instance based on the requirements it has. Proper instance is either found from the working and available ones or the new one is started to serve the request.
Such approach gives the possibility to manage the virtualization automatically with minimal manual interaction. The architecture of the VSM is presented in Fig. 3, as for now there is a number of independent modules offering the XML-RPC interfaces to interact with them.

- Manager – manages all other modules and routes the requests to the services,
- Virtualization Unit – offers the access to hypervisor actions, uses *libvirt* to execute commands what gives the independence from particular hypervisor,
- Database – module used to store monitoring data, images of available services (capsules) and information about available execution systems,
- Monitoring Unit – offers information about particular physical servers (execution system) as well as about available virtual service instances,
- Matchmaker – module responsible for the properly match the requirements of the request with capabilities of the environment and current state of it.

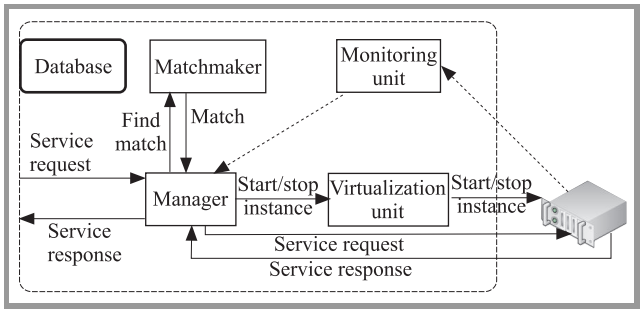


Fig. 3. The architecture of the Virtual Server Manager.

Each SOAP request coming to the system is directed to the Manager module which extracts requirements passed in the header section of the message to properly handle the request. Process of request handling starts with SOAP message coming from outside through the Broker that is an actor initiating the process service execution. This is in accordance to the general idea of placing VSM inside Service Oriented Architecture where Broker is common module for such purpose. The Manager module has a role of being the gateway to the system and hides all of the heavy lifting from outside world. It extracts the requirements passed in the SOAP message, in its header section. The requirements are extracted and converted to simple text form which is used by the Matchmaker module.
Handling the request can lead to one of three situations. There is a running service instance, which can perform it and it will be returned as the target to which the SOAP request shall be forwarded. There is no running service instance, but there is an image which satisfies the conditions. In such a case the image will be instantiated and it will be used as the one to perform the request. Last possibility is the lack of proper service and image in which case the error will be thrown and finally returned as a SOAP Fault message to the client.
As it was already mentioned, virtualization management is based on open source *libvirt* toolkit. It offers the virtualization API supporting number of the most popular hypervisors. From the point of view of this paper it is less important how technically the management is performed. It is more important to note what are the capabilities of the management and how it is understood here. The virtualization management does not mean simply to start or stop the virtual machine, it is much more complex problem. The complexity is coming first of all from the decision making problem. Firstly the correct service instance or service image should be found, by means of fulfilling specified in the service request requirements. In the case when the new instance of service has to be created the proper resources should be allocated and finally make processing as minimal footprint as possible.
Currently, using VSM the following features are available:

– mechanism for the creation and use of services – using the SOA paradigm and virtualization; this makes services independent from the available hardware architecture, and ensures the efficient use of hardware resources;

– method of delivery of services in a virtual machine environment – are taken into account the performance parameters of the virtual machine, service and equipment on which a virtual service instance is installed;

– tool architecture and its constituent modules is open, communication takes place via defined interfaces using XML-RPC for internal communication and the SOAP protocol for external communication.

5. Evaluation of Values of Service Parameters

5.1. Service Monitoring and Estimation in the Broker

The Broker performs request distribution based on the actual values of service instance parameters forecasted from the monitored and collected data of previous executions. The two basic service parameters, the data transfer time and completion time of service execution in the processing server, are obtained in two ways: with use of SOAP based cooperation between the Broker and the system which executes the service instance, and with use of the monitoring of TCP session which handles service request to the processing server.

In the first case the execution system must be able to interpret the specific additional data in Broker calls for service, and include additional specific data in service response. The execution systems controlled by VSM has such ability.

The Broker records the arrival time of each request for the service, the start time of call for service to the server which executes the service instance, and the time of end of processing of the service response from the server. The difference of the last two times establishes the total time of the request processing. The execution time of the service instance (processing time in the server) is delivered in the service response SOAP message. The service data transfer time is assumed as a difference between the total time of the request processing and service instance execution time. This time includes the time of resolving DNS address of processing server and all pre-transfer operations. However, these components of request intervals are measured by the Broker and can be excluded as described latter.

When cooperation between the Broker and the execution system is not possible, the values of essential service parameters are obtained with use of the analysis of the TCP session that handles the Broker's request for the service to the execution system.

The client request arrives at the moment t_{RA} (Fig. 4). The interval T_{DM} is the time of choosing the service instance (or server) that will process the request. Starting from this point the Broker measures the following time intervals of TCP session of call to processing server:

- the time of resolving DNS name address T_{DNS},
- the time of establishing TCP connection (TCP Connect time) T_{TCPC},
- the time to receive the first byte of transferred data from the server executing the service T_{FBYTE},
- the total time of the request processing $T_{SUM} = T_{FBYTE} + T_{DTRANS}$.

The Broker also records the number of sent bytes B_S and the number of received bytes B_R during the session.

It is assumed, that services are delivered using SOAP standard and the server responds after receiving all necessary

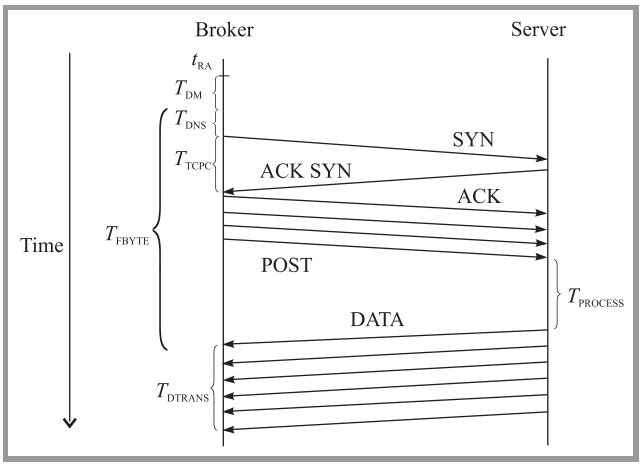


Fig. 4. The TCP session of handled client request.

data from the Broker. If we assume that transfer rate to and from the server are similar (what can be not true in general case), the service execution time $T_{PROCESS}$ can be evaluated with Eq. (3):

$$T_{PROCESS} = T_{FBYTE} - T_{DNS} - \frac{B_S}{B_R} \cdot (T_{SUM} + -T_{FBYTE}) - 2 \cdot T_{TCPC}. \quad (3)$$

Very often the request for service does not transmit other data in addition to those that fully identifies the service, so the time of this transmission can be neglected. Because the processing servers are registered in the broker earlier their IP addresses can be known, and the T_{DNS} time can be usually also neglected. In such case the service execution time $T_{PROCESS}$ is calculated according to Eq. (4):

$$T_{PROCESS} = T_{FBYTE} - 2 \cdot T_{TCPC}. \quad (4)$$

The data transfer time is the difference between the total time of the request processing and service execution time $T_{PROCESS}$. The service delivery time (for the client which requests the service from the Broker) includes also the decision making time T_{DM}, which according to distribution algorithm may be neglected or not.

In more general case some pre-transfer operations (i.e., SSL connect/handshake) must be also taken into account. The Broker can measure such operations too. It must be noted, that in case of lack of cooperation between the Broker and the execution system the estimation procedure is possible only when data transfer time from the Broker to the server is negligible, or data transfer rates to and from the server can be compared, or the data transfer time to the server can be measured separately.

Forecasting of values of service execution time and data transfer time for incoming requests is also performed in First, with use of time series analysis, i.e.:

- moving average of recorded times of previous executions:

$$\hat{t}^n_{j,m} = \frac{1}{L} \sum_{k=n-1}^{k-L} w_k \cdot t_k,$$

– moving median of recorded times of previous executions:

$$\hat{t}^n_{j,m} = med(t_{k-1}, t_{k-2}, \ldots, t_{k-L}),$$

where: $\hat{t}^n_{j,m}$ – forecasted time for n-th request served by service instance $is_{j,m}$, L – the length of the observation window, w_k – window function, t_k – the times of previous requests served by instance $is_{j,m}$, n – the index of current request.

When cooperation with the server is possible, i.e., when VSM is applied, the evaluation of service transfer and execution times can be performed with use of the concept of fuzzy-neural controller built with use of 3-layered fuzzy-neural network [8]–[11].

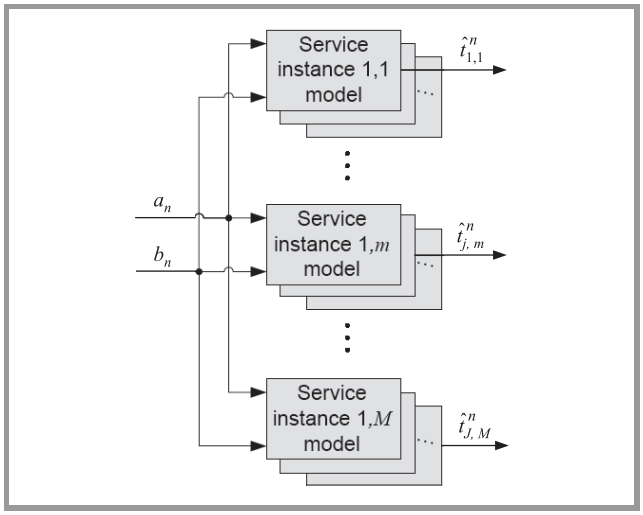


Fig. 5. Modeling service instances as fuzzy-neural controllers.

The fuzzy-neural controllers model each communication link and each service instances separately (Fig. 5). The adaptive model, described in detail [9] evaluate the output value (transfer or execution time) using two input values of parameters characterizing communication link or execution system. The input of the communication link model is, by now, link throughput of sample value of data downloaded from execution system, and link latency (namely TCP Connect Time), both derived with use of measurements and time series analysis. The input of service instance model can be any of monitored parameter by VSM or the counted number of being processed calls in the server.

5.2. Service Execution Monitoring in Execution System

The VSM is equipped with Monitoring Unit that is able to collect different parameters related to service execution and state of execution systems, depending on Broker request. Monitoring agent shall accompany with any currently active service. The frequency of measures or agreed values of attributes are present in the contract thus the agent shall simply check if the operations are done accordingly.
To ensure efficient resource utilization, incoming requests to VSM are attributed to execution classes. The functional and non-functional requirements are considered. The VSM exploits the combining the service orientation with automatic management using constraints attached to the requests what increases overall reliability, response time and constraints fulfilment, reducing the need for manual work in the same time.
Currently two different ways of performing monitoring can be in use. First solution based on using Munin, a tool, which is used to monitoring service execution and state of execution systems. Unfortunately this approach although highly efficient does not allow direct monitoring of virtual machines used for service instance execution. It's why it was decided to introduce an alternative way of monitoring. The second available solution based on using Xen-stat, which is an integral part of the package Xen virtualizer. Using Xen-stat allows to monitor the server and each virtual machine at intervals specified by the administrator. In particular, it is possible to monitor:

– CPU consumption by each virtual machine individually,

– CPU usage on the server,

– RAM memory usage of each virtual machine individually,

– RAM usage on a server.

For storing monitoring data simple MySQL database is used. Implemented database consists of three tables:

– measurements – contains information about the virtual machine load,

– capsules – contains data about virtual machines,

– servers – contains information about the server and its current load.

To collect and record information about the system load special module implemented in Perl is used. The module is run every minute and takes measurements at intervals set by the administrator. Data from the monitoring of virtual machines and servers available are displayed by Xentop command, and then the results are parsed and stored into a database. Then it is possible to visualized the results of the monitoring of resource usage by running virtual machines using Xen Graph tool. Monitoring data collected by the VSM in the local database are also available to the outside through accepted by the VRM SOAP messages.
Concluding current solution is very flexible because depending on the needs of monitoring the system usage and service execution gives the opportunity of using two different tools – Munin and Xen-stat.

6. Effectiveness Tests

In the preliminary experiments the selected proposed solutions were tested in real environment – in the Internet.

The broker has been implemented as fully operational tool in Java technology. It supports all described functionalities and serves its services in accordance with SOA standards. The experiments has been focused on testing usefulness of monitoring and evaluation of service instance execution time, however data transfer time was also tested.
The Broker served a number of clients requesting fixed set of network services from five servers located in different countries of Europe. There were established six test services. Each service was running on each server giving a total of 30 service instances. The services generated different amount of data to transfer from 50 to 200 kilobytes.

Service instances were set different values of non-functional parameters. On each server machine the services run in www server with established maximum number of parallel threads for serving clients requests. Each service instance was implemented in that way that have had minimal time of processing not including any server service handling overhead (e.g., queuing delay). The service differed in basic execution time with one another and the service instances of the same service differed in basic execution time depending on instance location. The times varied from 2 to 6 seconds.
The clients which requested services and the Broker were located at Wroclaw University of Technology campus. The servers that run service instances were located in five different countries:

- planetlab2.rd.tut.fi, (193.166.167.5), Finland,
- ple1.dmcs.p.lodz.pl, (212.51.218.235), Poland,
- planetlab1.unineuchatel.ch, (192.42.43.22), Switzerland,
- planetlab4.cs.st-andrews.ac.uk, (138.251.214.78), UK,
- planet1.unipr.it, (160.78.253.31), Italy.

The research scheme was the following:

- the clients requested all six services in a round-robin fashion, each client in a different order,
- the number of clients increased from 0 to 80 during 3 hour test,
- the requests were distributed by the Broker according to round-robin algorithm,
- it were measured essential moments of each TCP session handling requests for services, and recorded service instance execution times received in server responses,
- for each request the fuzzy-neural controller calculated forecasted value of service instance execution time and data transfer time,
- estimated and forecasted times were compared against measured ones.

All requests transmitted no additional data to the server. It was assumed that the true real values of service parameters were: service instance execution time measured in the server $T_{PROCESS-REAL}$, and the difference of the total time of request processing T_{SUM} and the service instance execution time measured in the server $T_{SUM} - T_{PROCESS-REAL}$, assumed as real value of data transfer.
The test showed that none of five servers were overloaded by requests for services. Figure 6 shows Mean Absolute Percentage Error (MAPE) calculated for estimation of service execution time $T_{PROCESS}$, using monitoring of TCP session only. The figure shows MAPE for all 30 instances grouped in the following manner: first six instances from first server (each of different service), next six instances from second server, and so on.

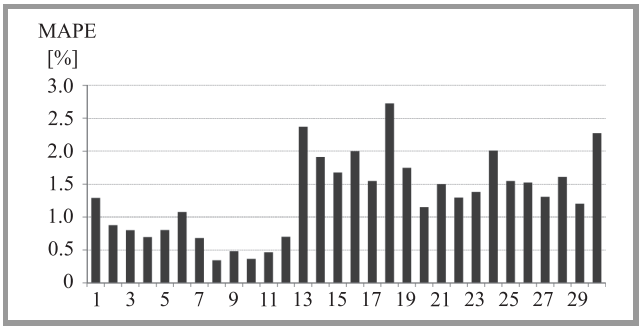


Fig. 6. The MAPE of $T_{PROCESS}$ time estimation for all service instances.

The total MAPE for all estimation is very good, and is equal 1,31%. It is interesting to see the visible difference of error level for particular servers. For server 2 (instances 7 to 12) the total MAPE is the smallest and is 0,51%. For server 3 (instances 13 to 18) the total MAPE is the largest and is 2,04%. This could be caused by different server queue thresholds (the number of requests processed in parallel), however should be thoroughly examined.
Figure 7 shows effectiveness of forecasting service instance execution times $T_{PROCESS}$ with use of fuzzy-neural controller. The total error of prediction MAPE is equal 1,32%. This is very good value. However, it must be noted that experiment was performed for stable server operation. In this case no obvious dependency of error level on particular server is visible.

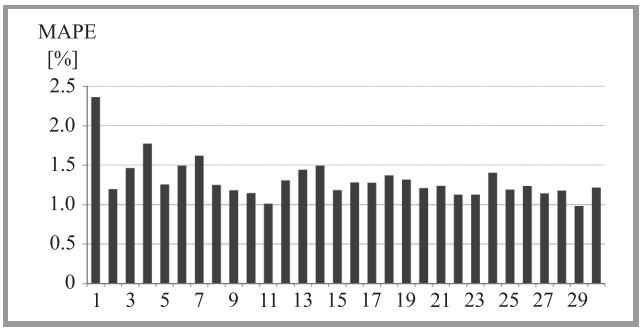


Fig. 7. The MAPE of fuzzy-neural forecasting of $T_{PROCESS}$ time for all service instances.

The forecasting of data transfer times were performed using moving average method and with use of fuzzy-neural controller. The forecasting errors were much greater than forecasting execution times. The total MAPE for fuzzy-neural controller was 13.5% (for weighted moving average even greater, about 19.3%).

It was due to the fact, that very small amount of data was transmitted and there were short transfer times – they varied from about 50 ms to hundreds of milliseconds. At the same time it was found that one of the link (to server 2 – service instances 7 to 12) was apparently problematic and significantly expanded total error as shown in Fig. 8. It must be noted that preliminary experiment was not focused on testing fuzzy-neural controller for data transfer time forecasting, and learning parameters of the controller were not tuned too. When this conditions will be met the better forecasting is expected.

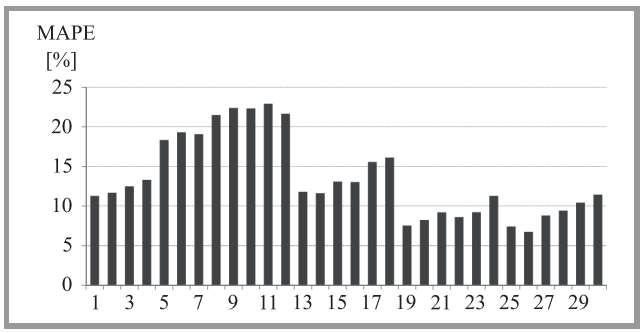


Fig. 8. The MAPE of fuzzy-neural forecasting of data transfer time for all service instances.

To test the performance of the VSM the test environment has been created. Most of the code has been written in Python (modules), sample services as well as sample requests are generated using Perl. Tests in current state has been limited to the CPU usage. The services are dummy and all they do is to utilize certain capacity of the CPU during certain amount of time. The solution to make this happen is somewhat not exact and limits the usage properly when set above 5%. It uses simple method to increase the CPU usage to 100% forking Perl processes and starting never ending loops. Such process would always consume all of the processing power so it is being limited using CPUlimit tool.

The number of CPU's to be stressed is fixed in the code, but it is not a problem to pass this number as an argument to the script. Time and CPU usage are limited in the directly in the dummy services implementation. It is done by first starting the script as a new process. It's ID is passed to the CPUlimit tool with desired usage, e.g., 50% and as the last step after desired number of seconds the process is killed to free the resources.

The first performed experiments using testing environment, in case when VSM is lack of automatic resource freeing are very promising. During these experiment called service is configured to consume 80% of CPU for 60 seconds what simulates some relatively exhausting operation. The requests are incoming almost in the same time and they require at least 30% CPU capacity to be free. The outcome of such a simple simulation is increasing number of instances of the service to handle sudden peak in requests, yet there is no mechanism to limit the number of the instances afterwards.

7. Conclusion

Quality of services in Service Oriented Architectures yields a number of issues which involves suitable monitoring and estimation of values of service parameters, distribution of service requests to selected execution systems running instances of services, forecasting values of service parameters and virtualization management. Automation of such process requires well designed architecture and procedures of service quality aware system.

The presented solution for solving all mentioned problems are still under study. First experiments are very promising. The effectiveness of service request distribution algorithms depends on precise evaluation of values of service instance parameters. The two basic ones, execution time and data transfer time, are the key to satisfy Quality of User Experience (QoE). The experiments showed that the evaluation of execution time is very good. However, the case of sending large amount of data from the client must be also tested. The evaluation of data transfer time is to be more explored and requires well prepared extensive experiments.

It is worth to note that presented solutions are implemented as fully operational tool ready for use in real environment that applies SOA standards and Simple Object Access Protocol (SOAP).

Acknowledgment

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund programs no. POIG.01.01.02-00-045/09-00 and POIG.01.03.01-00-008/08.

References

[1] D. Box *et al.*, "Web Services Policy Framework (WS-Policy)", 2003 [Online]. Available: http://public.dhe.ibm.com/software/

[2] A. Keller and H. Ludwig, "The WSLA framework: specifying and monitoring service level agreements for web services", *J. Netw. Sys. Manag.*, vol. 11, no. 1, 2003.

[3] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang H, "QoS-aware middleware for Web services composition", *IEEE Trans. Softw. Engin.*, vol. 30, no. 5, 2004.

[4] P. Sargeant, "Data centre transformation: How mature is your it?", 2010 [Online]. Available: http://www.gartner.com/it/

[5] B. Snyder, "Server virtualization has stalled, despite the hype", InfoWorld, 2010 [Online]. Available: http://www.infoworld.com

[6] D. Rosenberg, "Analyst: Virtualization management key to success", 2010 [Online]. Available: http://news.cnet.com/8301-13846_3-10468343-62.html

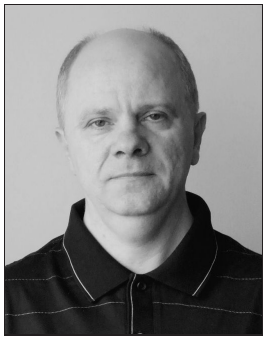
[7] P. Sempolinski and D. Thain, "A comparison and critique of Eucalyptus, OpenNebula and Nimbus", *C*, in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2010*, Indianapolis, USA, 2010.

[8] L. Borzemski, A. Zatwarnicka, and K. Zatwarnicki, "Global distribution of HTTP requests using the fuzzy-neural decision-making mechanism", in *Proc. 1st Int. Conf. Comp. Collective Intelligence*, Lecture Notes in AI, Springer, 2009.

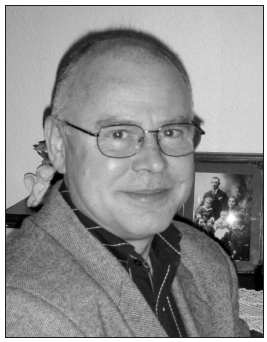
[9] M. Fras, A. Zatwarnicka, and K. Zatwarnicki, "Fuzzy-neural controller in service request distribution broker for SOA-based systems", in *Proc. Int. Conf. Computer Networks 2010*, A. Kwiecien, P. Gaj, and P. Stera P., Eds. Berlin, Heidelberg: Springer, 2010.

[10] L. C. Jain and N. M. Martin, *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications*. CRC Press LLC, London, 1999.

[11] E. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis", *IEEE Trans. Comp.*, vol. C-26, iss. 12, 1977.



Mariusz Fraś received M.Sc. in Electrical Engineering in 1989 and in Computer Science in 1991, both in Wrocław University of Technology. In 2004 he received Ph.D. in Computer Science in Institute of Informatics, Wrocław University of Technology. Since 2004 he works as an Assistant Professor at the Faculty of Computer Science and Management, Wrocław University of Technology. His area of scientific interest includes distributed processing and Internet services. His main area of interest is parallel and distributed processing in computer network environment, the quality of network services, and Internet research and the performance of Web services.
E-mail: mariusz.fras@pwr.wroc.pl
Wrocław University of Technology
Wybrzeże Wyspiańskiego st 27
50-370 Wrocław, Poland



Jan Kwiatkowski received M.Sc. and Ph.D. in Computer Science from the Institute of Technical Cybernetics, Wrocław University of Technology at 1977 and 1980, respectively. Since 1980 he works as an adjunct at the Faculty of Computer Science and Management, Wrocław University of Technology. In the years 1987–1998 he acted as a deputy director responsibly for education. From 2002 to 2004 under sabbatical leave, he worked as associate Visiting Professor at Math and Computer Science Department at the University of Missouri, St. Louis. Since 2007 he acts as Computer Science and Management Faculty Dean representative responsible for foreign students, currently as a Dean's Plenipotentiary for International Relations. His area of scientific interest includes software engineering and parallel processing. He is mainly interested in parallel and distributed software design process, performance evaluation of parallel programs and cluster/grid computing.
E-mail: jan.kwiatkowski@pwr.wroc.pl
Wrocław University of Technology
Wybrzeże Wyspiańskiego st 27
50-370 Wrocław, Poland