A Framework for Adaptive QoS of Web Services using Replication

Mohamed-K HUSSEIN, Mohamed-H MOUSA
Faculty of Computers and Informatics, Suez Canal University, Egypt.
{m_khamiss, mohamed_mousa}@ci.suez.edu.eg

Abstract
For large-scale access of web services, such as news web services connected to a database, it is challenging to provide the expected QoS attributes, such as response time and availability. Further, using a single server host is not enough to provide the expected non-functional characteristics of the published web services. Especially, if the server host experience any problem, this will downgrade the reliability and the availability of the web services. Service replication is an important technique to support the non-functional properties of the published web services. In this paper, we propose a framework to adaptively meet the expected Quality of Service (QoS) attributes of web services under high number of requests using service replication. The proposed framework uses Holt’s Linear and Exponential Smoothing (HLES) time series technique to smooth and provide short term predictions of the response times and the requests arrival rates for the accessed web services. Moreover, based upon the predictions using HLES, the framework uses the Queuing Network (QN) to model the impact of the requests load on a particular web service to predict the violation of the expected response time. The QN model takes into account the different classes of the published web services. Upon a prediction of QoS violation, the framework issues a replication decision on another server host. Finally, a load balancing mechanism is used to load balancing the requests between the replicas and the original web service based on the QN model. We present the experimental results which show the effectiveness of the proposed framework.

1. Introduction

Web service is a well-defined software component that provides specific business functionality published over the web [1]. Web services are published, described, discovered and executed using several standard protocols including Web Service Definition Language (WSDL) for service description, Simple Object Access Protocol (SOAP) for intercommunication, and Universal Description Discovery and Integration (UDDI) for publishing and discovering the abstracted and loosely coupled web services [2]. The non-functional attributes of web services, called Quality of Service (QoS), are fundamental to the success of the IT organizations, especially given the ever increasing demand for web resources [3]. The Service Level Agreement (SLA) is the predefined expected QoS attributes standard between the provider and the consumer of a web service, including response time and availability. The response time of a web service is the time required to complete a web service request. Availability is the probability that the system will be able to respond to the client request at any times. However, when the service requests keep growing and the web service will be overloaded, eventually, the SLA will be violated.

Replication technology is one of the useful techniques for supporting the non-functional properties of the published web services, such as availability, reliability and response time, through providing multiple replicas of a specific web service on different web servers. For example, if one service fails, a replica of the failed web service will be possibly created on another web server in order to process the requests [4]. Previous research works, in web service replication, have proved that service replication can basically provide high availability and reduce the response time through broadcasting client requests among a number of replicas [5-7]. However, most of the research mainly focused on the consistency and management of the replicas and hardly considered the adaptation of QoS in the real-time of the web services; in particular, to adapt the performance a web service under large number of requests in order to meet the expected SLA.

In this paper, we propose a framework to provide adaptive replication of web services in a distributed environment in order to meet the expected response times under large number of requests. The adaptive replication decisions must be issued according to the changing conditions of that might occur at the services provider side or within the SLA elements, such as sudden increase of the load on the server hosts or increase of the requests on a particular web service. In order to make an adaptive QoS-based replication framework, the framework must consider the following issues:

- The framework must contain an analytical performance model for response times of the published web services and their replications under different load requests.
- The performance model must take into account the impact of other published web services classes.
- The framework must monitor and provide short term predictions of the future response times and the requests arrival rates of the published web services.
- The framework must be able to predict in advance the future SLA violation, and trigger a replication decision.

For the first and the second issues, Queuing Network (QN) models are widely used in the literature for creating an analytic performance model of a system [8-10]. The QN is used to model a system and estimate the performance impacts of design decisions [10]. Queues include both the resource providing the service and the
waiting line to access that resource. The resource in this paper is a published web service, and the waiting line represents the requests load. Two parameter types are used in creating QN models: workload intensity and service demands. Workload intensity is the load or number of requests per second and the service demands are a measure of how long a device is occupied for a given request type. By identifying the devices in a system, and measuring the service demands imposed by the various request types, we are able to define an upper bound on the number of requests the system can handle before exceeding the expected response time. Many research works have shown that Queuing Network (QN) models provide accurate analytic performance model of a system [8, 9]. In [8], a queuing network model is used to estimate the performance index of a medical information system under different number of requests. In [9], queuing network-based models are proposed to predict the performance of composite Service Oriented Architecture and make them run in an optimal way with limited resources. In [10], a QN model is used to predict how a specific web will respond to various load scenarios. They have shown that QN models are accurate predictors of response times and only require a small effort in order to collect the necessary model data.

For the third and the fourth issues, HLES smoothes the time series and provides a short-term forecast based on the trend which exists over the time series [11]. In this paper, HLES is used to smooth the variations of the recently reported service demands, the time taken to process the request by a particular web service. Further, HLES provides a short term prediction for the future service demands given the current load conditions. On the basis of the predictions, the QN model is used to provide an accurate prediction of the response time, and make replication decisions.

The main contributions of this paper are: 1) we propose a replication-based framework for an adaptive QoS of web services. The main goal of the framework is to achieve an adaptive QoS management in order to meet the predefined SLA. 2) The proposed framework uses a Queuing Network analytical model to predict the response time of the published web services. If the number of requests increases on a specific web service, and the predicted response time using QN exceeds the expected response time in the SLA, the framework initiates a violation of SLA and adaptively replicate the violated web service on another server host. 3) The framework uses Holt’s Linear and Exponential Smoothing technique to monitor and provide a short term prediction of the future requests arrival rates and the response times of the published web services. 4) The framework load balances the requests between the replicated services as well as the original web service in order to handle higher requests loads within the expected SLA using the proposed QN model.

The remainder of this paper is organized as follows. Section 2 presents the related work. An introduction to Queuing Network, HLES and the proposed adaptive replication framework are presented in Section 3. An evaluation of proposed adaptive framework is presented in Section 4. Finally, Section 5 concludes and suggests future work.

2. Related work

There has been lots of research work on web service replication to improve the QoS issues, specially the availability. In [5, 12] the authors has proposed three techniques, called active, semi-active and passive replication, for web service replication process according to only determining which replica could process the incoming requests. In the active replication, the client sends a request to all available replicas, and therefore receives a response from the replica which processed the request firstly. In the passive replication, the client sends a request to the primary replica which basically manipulates the client request, and consequently, sends the response to the client. At the meantime, it notifies all other replicas by the updated state of the primary replica. If the primary fails, another replica will become the primary one instead. Finally, in the semi-active replication technique, all the available replicas process the client request simultaneously. However, non-deterministic service is handled by a single replica appointed as a master. The master replica continuously updates the other replicas. In [13], replication techniques has been proposed based on three techniques named as parallel, serial and composite strategies. The parallel strategy is exactly acting as the active replication in [13]. However, in the serial strategy, the consumer typically maintains an ordered list of the available replicas which can process the request. Initially, the consumer sends the request to the first replica in the list; if the selecting replica fails or dose not respond during a certain period of time, the consumer re-sends the same request to the next replica on the preserved list. Finally, the composite strategy is a combination between the serial and parallel strategies in order to enhance the communication performance among the consumers and the available replicas with taking into the account the network traffic status. In [4], a number of replication strategies have been proposed based on adding time redundancy to a combination of active or passive replication in order to facilitate the process of selecting an appropriate strategy for the different business case. Time redundancy is established to enable the consumer to tolerate faults by sending multiple requests to a specific replica if it the web service fails to respond in a specified period of time. In [14], Directed Acyclic Graph (DAG) is used to model the replication process in order to explore all possible replications. Then, the genetic algorithm is used to select the replication configuration which
improves the performance. However, these replication techniques do not take into account the dynamic environment characteristics of the incoming load request on the different web services or the changing load conditions on the available host. Further, the mentioned works only maintain the consistency of the replicas and to improve the reliability and availability of distributed service-oriented systems, and other QoS issues of service replications were not utilized.

In [15], a stochastic Perti net is used to model the performance of composite services as well as the environment in order to achieve adaptive reselection during runtime. However, this adaptation technique requires expertise to provide the performance models. Further, the adaptation process starts when a failure occurs on a service. It does not consider the dynamic load of the services or the hosts. In [7], a framework has been proposed for consuming a changeable-location replica at runtime. Basically, the suggested framework accomplishes an adaptive reselection of a higher priority service when the failure occurs on the other services or an environment. The service priority order is based on availability of the network and the host machine; and also the execution time and reliability of the services. The main drawback of this framework is that it does not consider some other crucial metrics within the replica reselection process such as the service or host load. In [16], an adaptive replica selection framework has been proposed that takes into account both the characteristics of the replica hosting environment and the consumer requirements. The reselection process is based on calculating a set of hybrid metrics including service failure, host failure and dynamic load as well as the customer business-value degree. However, the main ideas of these proposed adaptive service replication frameworks based on selecting a web server for hosting the needed replica from a predetermined set of available hosts. These frameworks do not take into account the changing load conditions of the service and the web server during runtime.

In this paper, we propose a replication-based framework for an adaptive QoS of web services. The main goal of the framework is to achieve an adaptive QoS management in order to meet the predefined SLA in terms of response time. The proposed framework uses a Queuing Network analytical model to predict the response time of the published web services under large number of requests arrival rates and the response times of the published web services. Moreover, the framework load balances the requests between the replicated services as well as the original web service in order to handle higher requests loads within the expected SLA using the proposed QN model.

3. The proposed adaptive replication framework

This section presents the proposed framework for adaptive QoS management of web services. It starts by describing the Queuing Network model followed by Holt’s Linear and Exponential Smoothing time series technique. Finally, a detailed description of the proposed replication framework is discussed.

3.1. Queuing Network Model

Analytical performance models are used to predict the performance characteristics of a particular system under different load parameters and environment changes [8, 17]. Analytic models can estimate the response times according to the dynamic nature of the environment. Hence, we can issue control decisions and estimate the performance impacts of such decisions on the system. Queuing Network (QN) models are widely used in the literature for creating such analytical performance model of a particular system [8-10]. There are two main parameter are used in creating QN models: load intensity and service demands. Load intensity provides an indication of the requests load, and service demands are the average response times of specific web service on a particular service host.

In a QN model, each new server added to the system as well as other published web services classes impact the performance of a web service $c$. Each web service must have the service demands ($D_{i,c}$) measured for each class of a web service request, where $c$ is a certain web service, and $i$ is the web server which handle the requests. The service demands are represented as the time needed for the device to complete the request. The main goal of the proposed approach is to lower the service demand on a certain web service under heavy loads. By increasing the number of replicas, the average response time on the system for that given requests type will go down by some factor. The average response times $R_c$ are calculated in QN based upon the utilization of each host ($U_i$) and the demand of the service $c$, ($D_{i,c}$), on the server host $i$.

$$R_c = \sum_{i=1}^{K} \frac{D_i \cdot c}{1 - U_i}$$

(1)

where $c$ is a particular web service, and $i$ is a server host, and $K$ is the total number of server hosts which have replicas for the given web service class $c$. The total utilization of a server host $i$, $U_i$, is the sum of the utilizations from all classes of web services. The per class
utilization is a product of arrival rates and service demands.

\[ U_i = \sum_{c=1}^{C} \sum_{c=1}^{C} \lambda_{ic} \ast D_{ic} \]  

(2)

where \( \lambda_{ic} \) the arrival rate of requests of the given web service \( c \) on host \( i \). The first summation in Equation (2) states that the utilization of a server host \( i \) is the sum of utilizations from all web service classes in the system. The second summation, that represents the per class utilization for a server host, is the product of the arrival rates and the service demand.

3.2. Holt’s Linear and Exponential Smoothing and Forecasting

Holt’s Linear and Exponential Smoothing (HLES) is a computationally cheap time series prediction technique. HLES is selected for its capability of smoothing and providing short-term predictions for the measured requests arrival rates and service demand rates. Hence, HLES enables the proposed framework to monitor the arrival rates and service rates and to provide a short-term prediction for the future arrival rates and service rates with low computation time. Using these predictions, we can predict the utilization on each server host using Equation (2), and predict the future response time of the web service using Equation (1).

HLES smoothes the time series and provides a short-term forecast based on the trend which exists over the time series [11]. Suppose \( x_t \) is a time series value at time \( t \). The linear forecast for the \( m \) steps ahead is as follows:

\[ x_{t+m} = L_t + b_t \times m \]  

(3)

where \( L_t \) and \( b_t \) are exponentially smoothed estimates of the level and linear trend of the series at time \( t \):

\[ L_t = \alpha x_t + (1 - \alpha) (L_{t-1} + b_{t-1}) \]  

(4)

where \( \alpha \) is a smoothing parameter, \( 0 < \alpha < 1 \),

\[ b_t = \beta (L_t - L_{t-1}) + (1 - \beta) b_{t-1} \]  

(5)

where \( \beta \) is a trend coefficient, \( 0 < \beta < 1 \).

A large value of \( \alpha \) adds more weight to recent values rather than previous values in order to predict the next value. A large value of \( \beta \) adds more weight to the changes in the level than the previous trend. Initial estimates for \( L_t \) and \( b_t \) are as follows: \( L_1 = x_1 \) and \( b_1 = 0 \).

In this paper, HLES is used to smooth the variations of the measured service demands \( (D_{ic}) \) on each host \( i \) for a given web service \( c \), and to provide a short term prediction for the future \( D_{ic} \).

3.3. The Proposed Framework Architecture

Figure 1 shows the proposed architecture of the adaptive execution of the web service. The framework is based on three services, replicator service, monitoring service and the dispatcher service.

**The monitoring service** monitors the performance of the available servers such as the utilization of the CPU as well as the load on each available host. Further, the local monitoring service on each server host \( i \) monitors the service demand \( D_{ic} \) for each web service \( c \) as well as the arrival rate, \( \lambda_{ic} \), the number of requests joining in the queue of service \( c \) on server \( i \). Then, it uses the HLES to predict the next 60 seconds of the service demand \( D_{ic} \) using equation 3. Finally the local monitoring services report the predicted values to the main monitoring service.

**The dispatcher service** acts as a proxy for web services. It is a software-based layer-7 switch [16]. Such switches perform load balancing using several servers on

![Figure 1. The proposed adaptive replication framework.](image-url)
the back-end and a dispatching strategy using predicted load information about the back-end servers from the monitoring service. The dispatcher service uses queuing network to predict the impact of the load on a particular web service, on the basis of the predicted $D_{i,c}$ using HLES. Once, a dispatcher receives a request, It uses Equation (1) to predict the response time $R_c$ for this request. If the predicted $R_c - SLA(R_c)/SLA(R_c)$ less than a predefined threshold, 10% in the experiments. The dispatcher issues a replication decision. Hence, the replicator installs a new static web service on demand on a particular selected service host, selected by the replicator service. Otherwise, the dispatcher load balances the incoming requests. This is by forwarding the request to the host $i$ which has less utilization $U_i$ using Equation (2). Figure 2 shows the adaptive strategy of the dispatcher.

The dispatcher receives a request for a web service $c$. Use Equation (1) to predict $R_c$.

if the predicted $R_c = \frac{SLA(R_c)}{SLA(R_c)} < 10\%$

install a new replica on server host $i$ which has less utilization using Equation (2).

forward the request to the new server host.

else

forward the request to the server host $i$ which has least utilization $U_i$ using Equation (2).

Figure 2. The adaptive strategy of the dispatcher

The replicator service creates an instance of a particular web service on a selected server upon receiving an order from the dispatcher service. The selection is based on comparing the loads on the available hosts. The current loads on the servers are obtained from the main monitoring service. Where, the main monitoring service collects the loads information from the servers by contacting the local monitoring services every 30 seconds.

4. Experimental Evaluation

In this section, we present the experiments conducted to evaluate the proposed framework. The adaptive mechanism using queuing network is used to maintain response times below that of the predefined SLA under large number of requests.

The evaluation environment consists of 6 servers (Desktop-computers): 1 main server which hosts the adaptive replication framework, 4 secondary servers for hosting the web services replications and 1 database server. Each server is core 2 duo 2.4 GHz, 1 GB RAM, 40G HD and running Fedora 10. Apache Benchmark [18] was used to load test the framework. The Apache Benchmark has the ability to configure the number of requests and the time spent on the test as well as the number of concurrent requests it will make at a given time.

Two different web services were implemented. Each web service is connected to a database server and executes a query. For example, a web service returns the cost of air flight tickets from a specific place to a specific destination. While the other web service returns the hotel prices in a specific destination. For the first web service, it executes an SQL query to obtain 10 records out of 1500 records. The second web service executes an SQL query to obtain 20 records from 7500 records.

We ran a performance test using Apache Benchmark with 10 concurrent users for a period of 2 minutes, 20 concurrent users and so on until 150 concurrent users, each for the same period of time which is 2 minutes. Apache Benchmark measures the requests per second

Figure 3. Response time (in milliseconds) versus number concurrent requests.
during the test and the framework logs the response time for each request.

In the first scenario, we applied the requests only for the first web service. No requests at all were passed for the second web service. Figure 3 shows the mean response times for the corresponding number of concurrent users. Figure 4 shows the corresponding number of replicas which are created dynamically to improve the response time in order to meet the expected SLA. The adaptive replication framework achieves better response time than without adaptation. It kept the response time below the predefined SLA until it installed replicas on all the available secondary servers. Figure 5 shows the cost of the HLES prediction as well as the response time predictions using QN.

In the second scenario, a large number of requests (120 requests) were passed to the second web service for 1 minute. The dispatcher were configured to stop the adaptation for the second web service only, and handles the requests using just one replica on one server host. After 10 seconds, we start sending requests using the Apache Benchmark similar to the first scenario. Figure 6 shows the resulting response times for the corresponding number of concurrent users. Figure 7 shows the corresponding number of replicas which are created dynamically to improve the response. This figure shows that the adaptive replication framework installed only four replicas because the last available server was fully utilized with the requests passed to the second web service.

5. Conclusion and Future Work

In this paper, we have presented an adaptive replication framework to meet the expected QoS attributes, specifically the response time, under large number of requests. The framework uses a Queuing Network model to predict the impact of the load on a particular web service, on the basis of short term predictions of the service demand using the HLES time series. Hence, if the framework predicts a future violation on the specified SLA, the dispatcher issues a replication decision on another server host. Further, a load balancing mechanism is used to load balancing the requests between the existing replicas and the original web service using the QN model. The experimental evaluation shows the effectiveness of the proposed framework.

In our future work, we plan to consider a composite web service in our framework. Complex software product
can be created at runtime by invoking multiple cross organizational web services in an appropriate sequence, known as web service composition. Some web services can be invoked in parallel and others can be invoked in serial. The adaptive QoS for a composite web services is an essential feature of service oriented software. Further, we plan to use the proposed framework for 3D reconstruction on parallel computers in order to achieve high performance of 3D object reconstruction and visualization. A 3D object Reconstruction from a set of points in real time, however, is computationally very expensive task. Further, the use of expensive high performance computing machines is not practical for educational and research applications. The technology of web services opens new opportunities for applications that require high demands on computational resources. The replication technology can be in order to divide the computations over multiple replicas, each on a different geographically distributed computational resource, to speed up the computations.

6. References