Research Journal of
Applied Science
MEDWELL PUBLICATIONS

Cost Effective Approach of Complex Web Service Composition

B. Muruganatham, Sahil Babel and Neha
Department of Computer Science and Engineering, SRM University, Chennai, India

Key words: Failure probability, Quality-of-Service (QoS), web services, response time, execution duration

Corresponding Author:
B. Muruganatham
Department of Computer Science and Engineering, SRM University, Chennai, India

Page No.: 17-22
Volume: 15, Issue 1, 2020
ISSN: 1815-932x
Research Journal of Applied Sciences

Abstract: Web services are gaining importance and are often used as a standard approach for integrating diverse miscellaneous services often distributed over the network. Hence, it necessitates attaining high levels of reliability and availability without being affected by any service, network or infrastructure failures. To accomplish this, we propose an efficient mechanism which will narrow the plausibility of fault detection by dynamic web service selection among similar services. The Quality-of-Service (QoS) can have an important effect on web service composition service reliability. In this study, we perceive the rationale for the failure of the web services and resolve it by considering the QoS attributes like response time, execution time, etc. The proposed modeling approach indicates a possible reduction in the failure probability as well as a significant improvement in the execution duration of the complex web services composition. This has been proved through the results achieved with an experimental setup.

INTRODUCTION

Web services composition are used to manage business processes. It is dynamic in nature. Complex web service composition can be framed by identifying and integrating diverse web services distributed over the network. Web services is a service offered to facilitate communication between the service provider and service consumer over the web. Web services can be defined and published by the service consumer and it has been discovered and invoked for completing the task by the service consumer. Two or more web services combined to complete a task is called composite web services or web services composition. Reliability achievement is a big issue in service oriented computing; it is not like traditional standalone machine.

Web services are spread across the web which is remotely developed and hosted making it difficult to locate. It may also happen that the service becomes unavailable without prior notice. Also, its performance can dynamically change according to various reasons like change in workload of servers, unstable network connectivity or communication links, etc.

Faults may occur in service oriented systems due to various reasons. There can be fault in its framework like power fluctuations or any flaws made during the designing and implantation phase such as any software bug. The defect can also be introduced manually by the administrators. There can also be an issue that the service requested is not available or there is a very high response time.

In this study, we present a methodology which will detect the fault in the web service composition and then optimize the results obtained by minimizing the faults. For example, if a client requests for a set of services from the server, for instance, a user wants to buy a book online.

He will make a request and try to contact one of the various servers available (like amazon.com, ebay.com, flipkart.com).There can be a failure in the web service invocation, for instance, the server may not be available at that moment due to various reasons like maintenance or server is down or the server might be taking a lot of time to reply to the request or the particular requested web service is not offered by the server. Hence, in such cases to minimize this tendency of failure, we can redirect the request to another server.

Service availability indicates whether a service is available or not, i.e. Whether a response to the service invocation has been successfully received. The response time is the duration from the time the request has been sent till the time the response has been received.

Literature review: In this section, first will rundown and go through some of the related works and investigations on QoS management and fault tolerance mechanism prior to examining the affiliation amidst our research work and standardization efforts for the web services composition technique.

There are some previous researches on reliability and fault tolerance and the evaluation of the existing web service composition. Researchers[1] scrutinize the predicament of employing the optimal and best strategy for fault tolerance to build and establish a reliable service oriented system by mapping the user requirements and prerequisites. It also examines the fault tolerance policy selection for the business processes which are semantically homologous.

Various research experiments and investigations have been carried out on the subject of QoS aims selection of web services and composition. A QoS-aware middleware platform[2] proposed and it reports the issues of web services discovery from the repository for composition to maximizing the satisfaction of users by considering the QoS properties.

QoS management has been extensively studied and analyzed in the horizon of web service composition system[3, 4]. The primary concern of this research is on the subsequent matters: QoS specification to acknowledge characterization of functional performance and QoS attributes. Researchers[5, 6, 2] discusses the QoS properties providing the non-functional characteristics for web service selection when more than one service is present with similar and comparable properties.

In Menasce[6] various QoS issues on web services (Availability, response time, throughput, security properties like authentication, confidentiality, data integrity and non-repudiation) are discussed from the point of view of the service provider and service user. It states that the service users and providers need to be able to engage in QoS negotiation. The providers should monitor the load they receive from the users and should check whether they meet the Service Level Agreement (SLA). The user must also check the quality-of-service they obtain.

In the fault tolerant infrastructure for web services is studied which can be used for interoperability among applications and for redistributing on the web with high reliability and availability demands. It must provide a crystal-clear fault tolerance for the clients. The principle objective of this is to assure transparency in fault tolerance for end users utilizing the active replication technique.

In contemporary, there has been a growing interest in processes which can be implemented by invoking multiple web services. The client can cite their preferences and constraints and based on this the selection of services are dynamic in nature for service composition and enforced by discovering the suitable services available at the runtime[2, 5]. Negotiation techniques have also been used to attain and come up with a practical and reasonable solution.

A hybrid model is proposed to enhance the performance of complex composite web services. With the help of this the rate of failure can be computed and also the reliability can be determined. However, this does not work towards achieving the minimal error.

The works mentioned above do not aim towards minimizing the probability of failure in web services which has been taken into consideration in this study.

MATERIALS AND METHODS

Proposed model: In this section of paper, we begin by first describing the architecture and its various entities. The different QoS attributes for the web services are discussed followed by the required algorithms.

Architecture: The service provider is accountable for the providing the web services to the end users. The service consumer is the client who is the end user or web application making the request for the web services and utilizing these services created by the service provider. All the communication taking place will be tabulated in the service register which will be later used for analyzing purpose (Fig. 1).

The request analyzer will be calculating the amount of time taken to transfer the request from the consumer to the provider and to process this request. The response analyzer will be calculating the time required to send the requested web service from service provider to the service consumer. The WS status will mention the status of the service provider concerning the web service request.

The request will be sent to all the service providers one at a time and this data will be stored in the service register which will later forward its data to the fault detector. It will be detecting the fault in the web service

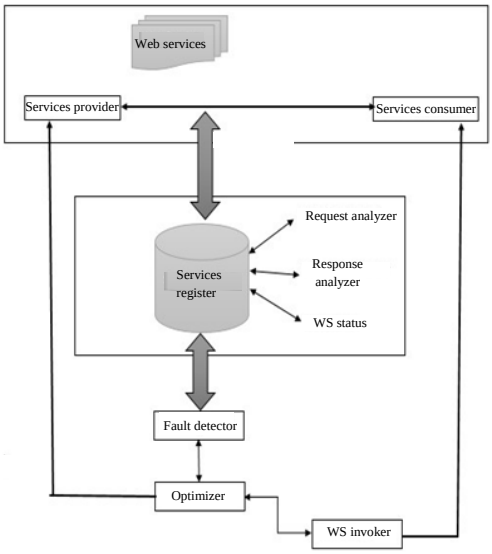

Fig. 1: Architecture diagram

invocation by various service providers which will be used to calculate the failure probability. This result will then be examined by the optimizer which will try to minimize this failure probability previously obtained by the fault detector. This can be done by redirecting the request to another nearby web service provider which will provide the similar homogeneous web services.

The WS Invoker is used for invoking the web service from the web service provider as instructed by the optimizer and will then send this service to the web service consumer.

Quality criteria for web services: In the present scenario, there are multiple candidates available for web services with similar functionalities. In such cases, QoS attributes impart some non-functional features for dynamic web service discovery. Based on the previous research methods and experimental results[2, 5, 6], we exemplify the following QoS attributes for the web services:

Service availability: The service availability determines the possibility of the end user to access the web service. The value of service availability is calculated as the total duration for which service is available during the last t seconds.

Service execution duration: The service execution duration is computed as the expected duration in seconds required by the service consumer to send the request and receive its results.

Service response time: The response time measures the duration receiving the response from the server.

Service failure probability: Failure occurs if either at the server side the request was not successfully executed or when the end user does not receive its corresponding response. Hence, the service failure probability is computed as the ratio of the number of services which were not successfully executed to the total number of web services.

Algorithm 1; Sending request to the service provider:

Input: SP: service provider.
WS: web service.
Output: request sent.

```
int SP Number = |SP|;
int WS Number = |WS|;
int ser=0;
for (j=0; j<SP Number; j++) do
ser++
for (i = 0; i<WS Number; i++) do
sendurl (ser, SP Number[j],WS Number[i]);
end
end
```

In Algorithm 1, the web service consumer is sending the request for the web service to the web service provider one at a time. The ser is the service index for the service provider which is being requested by the service consumer.

Algorithm 2; Finding response time:

Input: SP: service provider.
WS: web service.
Output: RT: response time

```
Open connection
int code: get response code
string response: get response message
long start time: current time of system
long response time: current time-start time
long x1: response time%1000
long x2: response time/1000
return response time
```

In Algorithm 2, the consumer bind with provider to invoke the functionalities of web services. The response, i.e., response code, response message from the network is obtained. The response time is calculated as the duration of the connection being established till it sends the corresponding requested web service back to the service consumer.

Algorithm 3; Fault detection:

Input: RT: response time
Output: F: failure

```
int lim: Time limit of RT
if (RT>lim) then
code = 408
response = Timeout
flag = true
else if (code = 404) then
flag= true
end
if (flag==true) then F = true
else F = false
return F
```

In Algorithm 3, the failure can occur if the response time exceeds time limit or if the web service is not available. The 408 is the request timeout error. It is an http status code which implies that the request the service consumer sent to the service provider took longer than the web service provider was prepared to wait.

The 404 is the Not Found Error Http status code. This error is shown when the requested web service by the service consumer is not available at the service provider.

Algorithm 4; Minimizing failure probability:

Input: SP: service provider.
WS: web service.
Output: request sent.

```
int SP Number = |SP|
int WS Number = |WS|
for (i=0; i<WS Number; i++) do
for (j=0; j<SPNumber; j++) do
curService = WS_i
curServer = SP_j
Send request of curService to curServer
if (code == 200) then
break
end
end
```

In Algorithm 4, the request for the web service made by the service consumer is sent to the service provider. If due to any reason like the service was unavailable on that service provider or its response time is exceeding the time limit etc. another request will be made to send the same service but this time to the next service provider. This will ensure that the availability of web services increases and hence there will be significant reduction in failure probability.

Experimental setup: The experimental setup has various computers or nodes acting as the server. The client will send a request for a web service. This request will be henceforth forwarded to the servers nearby one at a time. There can be various causes of failure now based on the Quality-of-Service (QoS) attributes as discussed in section 3 of this paper.

When the web service is unavailable on the server: In this scenario, it may happen that the particular web service requested by the service consumer is not available with the service provider. This can be due to various reasons like the server is down, network connectivity issues, server maintenance is taking place or it may be permanently shut down.

Service availability is the possibility that a service invocation will be executed successfully with a response. The value of service availability can be determined from the former data of service invocations as the ratio of the number of successful execution of web service against the total number of request made. Hence, this unavailability of the requested web service on the service provider will eventually lead to an error message received by service consumer hence acting as the failure.

When the web service has exceeded its response time: Here, the web service provider is available unlike in the first scenario but is taking time way too long to respond to the request made by the service consumer than it was supposed to. Hence, due to this increase in response time, it will increase the execution duration of the web composition and the service consumer will have to wait for long which is unfavorable.

Response time is the total amount of time it takes to respond to a request for web service. It is the duration from the point of time the request has been sent till the time its response has been received.

Hence, in such cases to reduce the risk of failure the client's request for that particular web service will be forwarded to other nearby servers providing similar web service as per request. This will thus lead to the successful invocation of web services and is efficient to implement for the execution of web services. Hence, it increases the success probability for invocation of web services by reducing the failure probability.

RESULTS AND DISCUSSION

Our proposed model and algorithm have been tested on a wide range of the web service request which was randomly generated. Experiments have been carried out to examine and compare the solutions which were obtained after applying the proposed methodology.

The following Table 1 given below states some of the response status codes based on the request made by the service consumer to the service provider where 200 represents the successful execution of web service and others represent a failure.

Table 2 and 3, presents the experimental results for only one request made for five web services with two service providers available. It returns the response time along with the status for each request indicating the outcome as success or failure. The web service index identifies the web service, and the server hit name identifies the service provider to which the request for the web service has been sent.

Table 2 shows the experimental result before implementation of fault reduction. Each service here is sent to a single service provider whereas in Table 3, the web services have been redirected to another service provider on the occurrence of a failure.

Table 1: Response status codes and message

Response code	Response message
200	OK
400	Bad request
404	Not found
408	Request timeout
429	Too many request
502	Gateway is not good
503	Unavailability of service
04	Timeout of gateway

Table 2: Experimental results of web service for one request; before implementing failure reduction

Web service index	Status	Response time (sec)	Outcome	Server hit name
WS1	408	2.179	Fail	Ser1
WS2	200	0.454	Success	Ser1
WS3	404	1.199	Fail	Ser1
WS4	403	1.157	Fail	Ser1
WS5	200	0.604	Success	Ser1

Table 3: After implementing failure reduction

Web service index	Status	Response time (sec)	Outcome	Server hit name
WS1	408	1.658	Fail	Ser2
WS2	200	0.454	Success	Ser1
WS3	200	0.568	Success	Ser2
WS4	403	0.166	Fail	Ser2
WS5	200	0.889	Success	Ser1

Table 4: Experimental results of web services for 100 request; Before implementing failure reduction

Web service index	Mean response time (sec)	Number of fail outcomes	Number of success outcomes	Failure probability
WS1	1.456	42	58	0.42
WS2	0.961	24	76	0.24
WS3	0.872	21	79	0.21
WS4	2.256	55	45	0.55
WS5	0.567	19	81	0.19

Table 5: After implementing failure reduction

Web service index	Mean response time (sec)	Number of fail outcomes	Number of success outcomes	Failure probability
WS1	1.032	26	74	0.26
WS2	0.211	18	2	0.18
WS3	0.755	15	85	0.15
WS4	2.116	37	53	0.37
WS5	0.232	11	89	0.11

For instance, it can be observed that in Table 3, the services which had previously failed as shown in Table 2 have now been redirected to server 2. It can also be noted that the number of web services successfully executing has increased.

Further, the experiment was extended to a different number of requests and then their aggregated values for response time was used to calculate mean response time. The number of failure and success outcome for all requests is being noted, and hence the failure probability for each web service is being calculated.

Table 4 and 5, presents aggregated data for 100 requests for five web services before and after applying the proposed methodology respectively. It can be observed that there is an increase in the number of successful execution of the web service. Hence, there is a significant reduction in failure probability of web services by increasing the availability of services in the event of failure.

Figure 2a, b and 3a, b presents the result of experiments conducted for 100, 200, 300, 400, 500 requests for five web services. Figure 2a, b compare the failure probability and Fig. 3a, b compares the response time before and after applying the proposed methodology. Hence, it can be clearly observed from the graph that the failure probability is getting reduced with better response time. Thus, the web services are being executed efficiently with optimum execution duration.

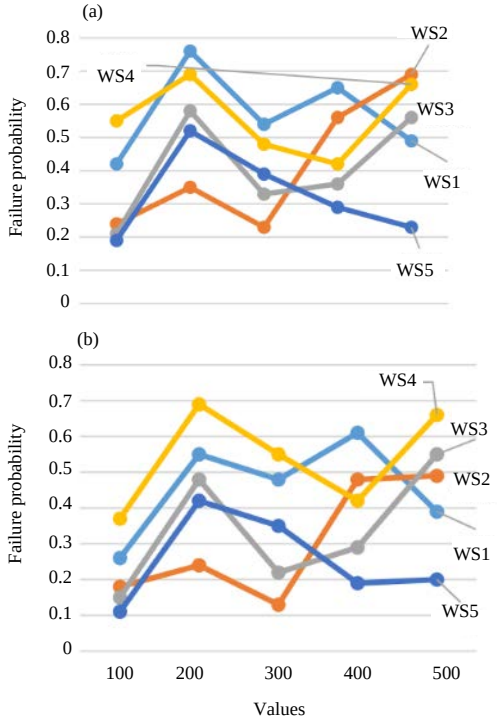

Fig. 2(a-b): Failure probability versus number of requests: a) Before implementing failure reduction and b) After implementing failure reduction

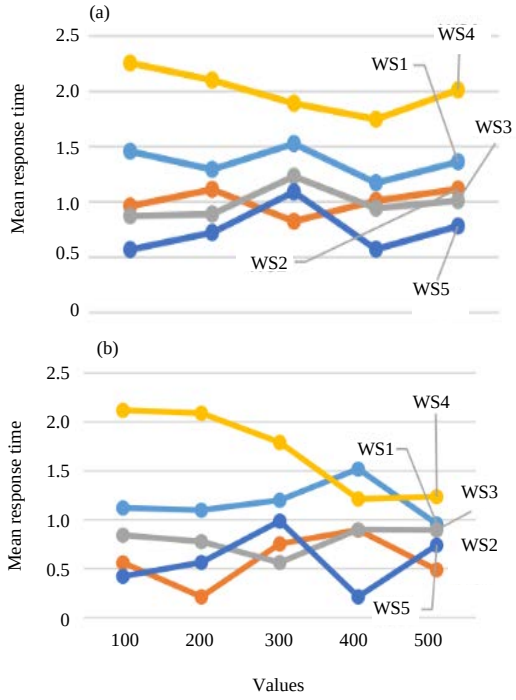

Fig 3(a-b): Response time versus number of requests: a) Before implementing failure reduction and b) After implementing failure reduction

CONCLUSION

In this proposed approach, we have conferred an optimized approach for the composition of web services by dynamically selecting the web services which allows specifying constraint on quality prerequisites. We have investigated the problem of minimizing the risk of failure of web services by reducing the failure probability as well as significant improvement in the execution duration of web services, or the composition is observed.

In future, more exhaustive and extensive research will be done on other QoS values for efficient execution of web services. This work can be extended to more complex web services dealing with sensitive data like in medical diagnosis, bank transaction, etc.

REFERENCES

01. Zheng, Z. and M.R. Lyu, 2013. Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints. IEEE. Trans. Comput., 64: 219-232.
02. Zeng, L.Z., B. Boualem, A.H.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, 2004. QoS-aware middleware for web services composition. IEEE Trans. Software Eng., 30: 311-327.
03. Aurrecoechea, C., A.T. Campbell and L. Hauw, 1998. A survey of QoS architectures. Multimedia Syst., 6: 138-151.
04. Nahrstedt, K., D. Xu, D. Wichadakul and B. Li, 2002. QoS-aware middleware for ubiquitous and heterogeneous environments. IEEE. Commun. Mag., 39: 140-148.
05. Ardagna, D. and B. Pernici, 2007. Adaptive service composition in flexible processes. IEEE Trans. Software Eng., 33: 369-384.
06. Menasce, D.A., 2002. QoS issues in web services. IEEE Internet Comput., 6: 72-75.