Optimization and Ranking in Web Service Composition using Performance Index

Pramodh N [#1], Srinath V [#2], Sri Krishna A [#3]
[#] Department of Computer Science and Engineering, SSN College of Engineering,
Kalavakkam- 603110, Kanchipuram District, Tamil Nadu, India.
[1] pramodhn91@gmail.com
[2] srinath.v1991@gmail.com
[3] sri.06.krishna@gmail.com

***Abstract*— Selection of the best service from the existing composite services is an emerging technology that aims at attaining a better performance. The steps involved include selection of relevant web services from the composite service, followed by optimization and ranking that ultimately leads to the execution of the best service. This paper concentrates on combining optimization and ranking based on non-functional QoS parameters to evaluate its quality. The concept of optimization is carried out by ACO (Ant Colony Optimization) algorithm. The optimization principle uses the concept of pheromone deposition and evaporation on the services. Ranking is done using performance index which is calculated dynamically from the non-functional QoS parameters. This ensures that the application based on this approach is efficient and fault tolerant. The result is obtained by measuring the performance of the services for multiple requests.**

Keyword- Web service, Optimization and Ranking, Performance Index, Quality of service/composition

I. INTRODUCTION

The World Wide Web Consortium (W3C) refers to Service Oriented Architecture (SOA) as 'A set of components which can be invoked, and whose interface descriptions can be published and discovered.' SOA consists of principles and methodologies for designing and developing software in the form of interoperable services. These services may be distributed and can be used to perform varied business logic operations. SOA provides Loose Coupling between the services and granularity in the services. Web services exchange information using SOAP protocol. Simple Object Access Protocol (SOAP) is a simple XML-based protocol to let applications exchange information over HTTP. The action between the business process are specified using the executable language called BPEL[5].

Business Process Execution Language (BPEL) is an XML based language that can invoke a web service and provides flexibility by allowing nested calls to web services. In other words, it is an orchestration language which specifies an executable process that involves message exchanges with other systems, such that the message exchange sequences are controlled by the orchestration designer. The message exchange facilities of BPEL depend on the use of the Web Services Description Language (WSDL). The idea of optimization coupled with ranking is a proposal to increase the execution scale of the process that uses it.

Ant Colony Optimization algorithm is used to perform optimization on the selected web services. Ant Colony Optimization (ACO) is a paradigm for designing meta-heuristic algorithms for combinatorial optimization problems [3]. Meta-heuristic algorithms are algorithms which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. Similarly, selection of the best web service involves starting from a null solution and then adding solutions that select the best service one by one. Ultimately, the end result becomes something similar to executing the best service for more number of incoming requests. The concept of pheromone deposition on most accessed paths and pheromone evaporation from least accessed paths can be applied in this case to select the best from the available services such that each one behave like an individual path.

Further, ranking approach asserts values to all the services and enhances the selection obtained from optimization. Coupling ranking along with optimization provides a novel approach to give a fair chance to all the services and also give maximum chance to the best one. The performance index is used as a feature that describes the rank of the web service.

This paper is organized into the following sections: Section 2 talks about the related work and current trends with respect to our problem domain. Section 3 brings out the overall implementation along with the concepts that are involved in all stages. Section 4 consists of the experimental results with graphical analysis. Section 5 concludes the experimental results and Section 6 speaks about the various other further enhancements with respect to our proposal.

II. RELATED WORK

Web Service Composition has been an area of research for quite some time. Different mechanisms of service composition have been used by various researchers. Optimization in service composition has grown into a field of interest of late and many optimization techniques have been developed. The concept of ranking based on non-functional QoS parameters helps in finding the best web service from the existing composite service. In [1], the focus is on semantic web composition by weighing the links based on their semantic quality and finding the optimal composition by using a GA (Genetic Algorithm) based approach. QoS based Dynamic web composition with Ant Colony optimization is discussed in [4], which constructs a path from the request to the response with the existing resources.

Definition 1: Web Service Composition:

Composition refers to combining of the various atomic services together in order to achieve a particular goal.

The services provided by the application are bundled into a single composite web service that invokes only those services required for handling a particular request. This ensures that multiple operations can be serviced by a single composite application.

Definition 2: Optimization:

Optimization is the process of selecting certain services from the existing ones in the composite service based on its optimization parameters.

GA based optimization is used by [1] where optimal semantic links are selected at each level based on their rank along various dimensions. In [4], various paths from the request to the response are generated and the optimal path is selected by the use of ACO algorithm.

Definition 3: Ranking:

Ranking is selection of the best web service from the optimized web services by selecting the web service with highest Performance Index.

In [6], services are ranked based on Web Service Relevancy Function (WsRF) which is measured based on the weighted mean value of the QoS parameters.

III. PROJECT WORK

A. Selection of Relevant Web Services

The best web service is executed from the composite web service after three steps namely selection, optimization and ranking as shown in Figure 1. A composite web service holds all the available services to be worked on. This composite service is created using Business Process Execution Language (BPEL), as explained in [5]. Selection of relevant web services from this composite is first performed by matching the web services with the inputs in the request. These inputs are unique for each operation and so the required services are selected from the composite.

B. Optimization and Ranking

The selected web services are then optimized using the idea behind Ant Colony Optimization algorithm. As shown in [3], this algorithm involves 2 major operations namely deposition of pheromones on the most used path and evaporation of the pheromones from the lesser used paths. Similarly every web service has an initial pheromone density. Based on this value, a few optimal ones are selected.

The best web service processes the request of the client which is selected by ranking of the optimal web services based on the performance index of the non-functional QoS parameters. These parameters are stored in the database and are updated dynamically.

1) Calculation of Optimal Web Services and Performance Index

Let us consider a set ‘S’, of selected web services for execution. If suppose there are ‘n’ services overall, then the number of optimal web services ‘N’ is calculated as,

$$N = \begin{cases} n/2 & \text{if ‘n’ is even} \\ (n/2)+1 & \text{if ‘n’ is odd} \end{cases}$$

The set of optimal web services A is evaluated as shown below.

$$A = \{w : w = nmax(W,N) , \forall W \in \mathbf{S}\}$$

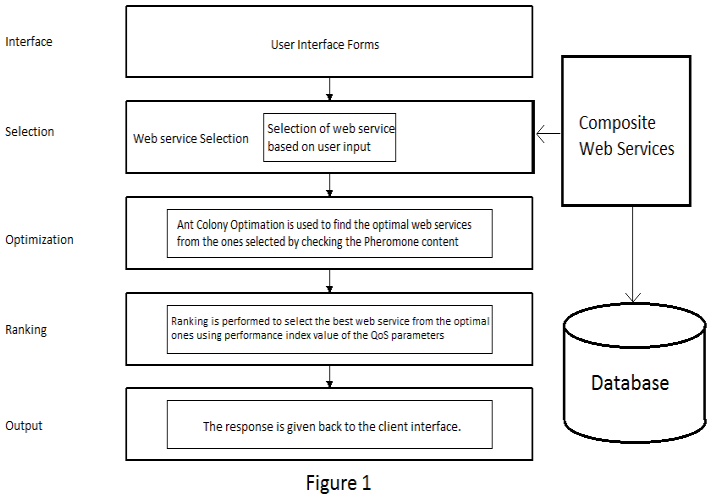


Fig. 1. Project Work

In case a clash occurs with same pheromone density value, then the contents of 'A' are chosen at random from the set of web services that have the same pheromone density. The best web service 'B' which is executed is selected as shown below.

$$B= \{ b \in A : b = \max(a), \forall a \in \mathbf{PI(A)} \}$$

The performance index is a measure calculated from the non-functional QoS parameters based on how the index behaves. The parameters chosen in this paper are Locality of reference, Access Count, Execution Time and Availability. These values individually get themselves updated on each access of their respective service. Let us name these parameters as param 1, param 2, param 3 and param 4 respectively. The Performance Index (PI) is calculated from these parameters as,

$$PI= LoR - AC - ET + AV$$

The Performance Index has both positive and negative effects on the parameters used. Each parameter is individually discussed below. Also, the effect of each parameter along with the range of values they take in affecting the Performance Index is discussed below.

Locality of Reference (LoR): Locality of Reference is a parameter that is used to identify the most recently accessed service. It has a positive effect in the performance index. The positive effect is attributed to the fact that the previous accessed service has a greater probability of being the best service for this request. Locality of reference takes a value of 20 just to add some weight in the PI of the recently used service. For other services, it is set to 0.

Access Count (AC): Access count is the cumulative count of the number of times a service was accessed. It is incremented by a factor of 1 for every access. It has a negative effect because, as the service gets accessed many times, its performance tends to degrade. The value of access count ranges from 0 to 100. When the count reaches the highest value, it is set back again to 0.

Execution Time (ET): Execution time, measured in milliseconds, is the time taken by the web service to process the request. Execution time has a negative effect on the performance index as the service with lesser execution time must be preferred over services with larger execution time. This time, calculated in milliseconds, is the key parameter in analyzing the performance of the application used with and without composition.

Availability (AV): Availability is a measure that increases on each access to a web service. As it is accessed more and more, it becomes more easily available to the requests and thus has a positive effect on the performance index. Its value is incremented by 1% on each access. Availability ranges from 50 to 100. Suppose the service faces a fault, its value is set to 0. This way, its performance index reduces and the chances of this service getting selected is reduced. This way, fault tolerance is handled in this approach.

Each of the values above is normalized so that they don't affect the performance index individually. Table 1 shows the range of values each parameter takes to calculate the performance index as discussed above.

TABLE I Range of Values each parameter takes

Parameter Name	Range of Values
Locality of Reference	{0,20}
Access Count	(0,100)
Execution Time	Time in Milliseconds
Availability	{0, (50,100)}

2) Scenarios

We have considered various scenarios in order to compare the Execution time of the various cases. First, we use a stand-alone web service and gauge its performance. We infer that its fault tolerance is low and while processing a large number of requests, the waiting time for each request increases which leads to high inefficiency. Next, we use a composite Web Service we use a composite web service without any Optimization or Ranking. In this case a web service is randomly chosen from the composite pool and there is no guarantee that the best performing web service is selected. If we use a composite Web Service with optimization alone then Ant Colony Optimization is used to optimize the web services, but fair chance is not provided to the web services and a particular chosen web service selected repeatedly. When we use a composite Web Service with Ranking alone, the web services are merely ranked based on their non-functional parameters and need not be the most optimal web services. So we use a combination of both Optimization and Ranking on this composite Web Service which delivers high performance.

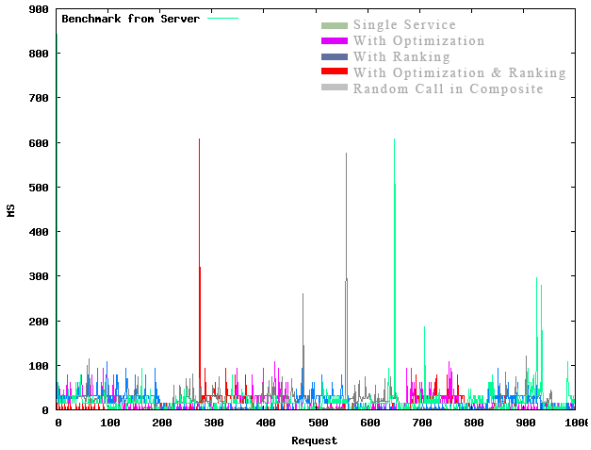
IV. Experimental Results

We analyse the performance of this approach using different methods. These methods are applied on the test application that performs various operations. The test application includes the ones with and without composition. The following section describes the effect of this method on various operations. The subsequent section tabulates the mean value observed in each operation.

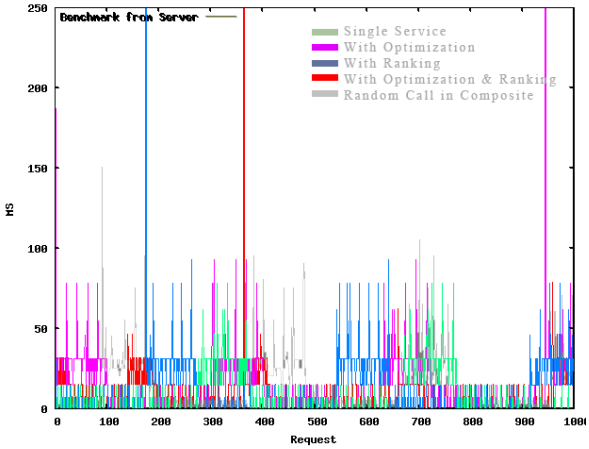
A. Benchmark Results for various operations

To compare the execution time taken in both these cases, we present results obtained for each method and the respective results for 1000 requests. The result for all the operations is shown below.

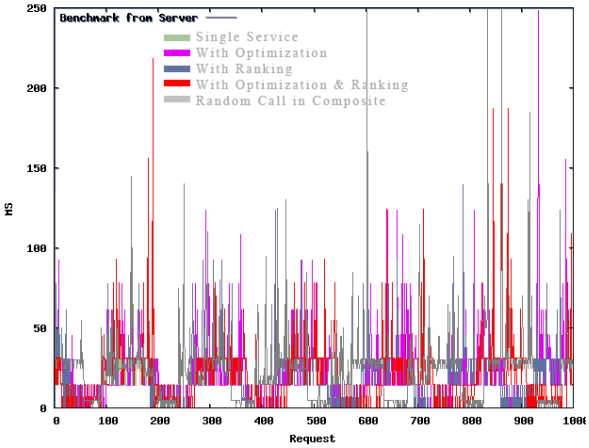
1) Operation 1 (Login)



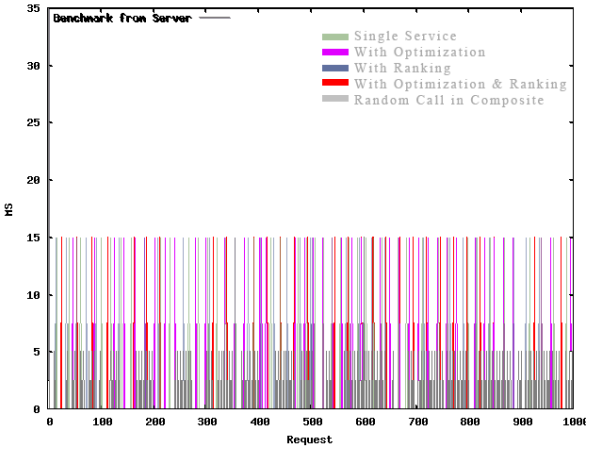
2) Operation 2 (Sign up)



3) Operation 3 (Cancellation)



4) Operation 4 (Booking)



B. Mean Values

The mean values of execution of all the operations for the graphs plotted in section 4.1 are shown in Table 2. The tabulated values show the reduction in execution time observed for the application with and without composition. With increasing number of requests and the size of execution, the mean value is expected to have much more difference.

TABLE II
OBSERVED MEAN VALUES WITH AND WITHOUT COMPOSITION

Operation	Single Service	Random Access from Composite	With Optimization Alone	With Ranking Alone	With Optimization and Ranking
1	17.831	17.460	12.667	13.416	8.143
2	7.613	11.428	13.510	10.390	6.568
3	20.171	23.340	19.469	20.951	18.424
4	1.622	1.355	1.014	0.827	0.624

V. CONCLUSION

For a scalable application, servicing increasing number of requests is critical for its performance. To counter this, we propose a novel approach that couples both optimization and ranking of composite web services. The Web Service Composition is optimized using Ant Colony Optimization (ACO) technique. ACO is a dynamic optimization technique which employs the concept of pheromone density. The pheromone density of the chosen web service increases and that of the other web services are "evaporated" or reduced. Thus, only the most optimal Web Service is chosen during every access. Further, we recommend an approach which endorses the ranking of the optimized web services. We have built a sample application with various operations to check the results. We rank the web services based on 4 non functional QoS parameters - Locality of Reference, Access Count, Execution Time and Availability. These parameters are used to calculate the Performance Index, which describes the total weight each service holds. Thus, through our unique approach, which prescribes the ranking of optimized web services, we ensure that the client is always serviced by the best performing web service at all times. Other significant advantages that add to the robustness of our approach are high fault tolerance, service reusability and loose coupling between the component web services. We have illustrated the benchmark results graphically which further supports our premise that the Composite Web Service followed optimization and ranking performs better when compared to other scenarios.

VI. FURTHER ENHANCEMENTS

Our approach is flexible and presents ample scope for further enhancements. The following can be the possible enhancements to this project. One way to achieve this is by using other optimization techniques. Some of the suggested techniques like Genetic Algorithm (GA), Particle Swarm Optimization (PSO) to perform the optimization. Another enhancement to our idea would be to use a different set of non-functional QoS parameters to evaluate the quality of the web service composition. Further we can vary the number of web services in the underlying composition to improve the performance. Thus, these enhancements further highlight the flexibility of our approach.

REFERENCES

[1] Freddy Le´cue´ and Nikolay Mehandjiev, *Seeking Quality of Web Service Composition in a Semantic Dimension* in IEEE Transactions on Knowledge and Data Engineering (June 2011)

[2] Munindar Singh (*North Carolina State University, USA)* and Micheal N. Huns (*University of South Carolina, USA), "Service oriented computing – Semantics, Processes, Agents"*

[3] Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi, *"Ant Colony Optimization"*

[4] Wei Zhang, Carl K. Chang, Taiming Feng, Hsin-yi Jiang, *QoS-based Dynamic Web Service Composition with Ant Colony Optimization* in 2010 IEEE 34th Annual Computer Software and Applications Conference

[5] Oracle SOA developers guide, "Using BPEL to Build Composite Services and Business Processes" (http://cdn.ttgtmedia.com/searchSOA/downloads/OracleSOADevelopersGuide_05_Final.pdf)

[6] Eyhad Al Masri and Qusay H Mahmoud., *Discovering the Best Web service*, University of Geulph, Geulph (2007)