Scheduling of Composite Services in Multi-Cloud Environment

Faisal Ahmad
Tata Consultancy Services Ltd
Newark, NJ, USA
faisal.nitdgp@gmail.com

Anirban Sarkar
Department of Computer Applications
National Institute of Technology, Durgapur
Durgapur, WB, India
Sarkar.anirban@gmail.com

Abstract—In cloud computing, resource allocation and scheduling of composite web services is very challenging and have many constraints. In a multi cloud environment, nodes can host more than one web service instances. Each web service instance can accept more than one request concurrently. Each node have different load (number of requests being served) at different time. Further, in composite web service, outputs of one service are input to another service which are required to be transferred through the cloud networks, which adds additional substantial data transfer time. The data transfer time among clouds and nodes is very important to depict the practical scenario. This paper presents an efficient scheduling mechanism which can efficiently schedule abstract services of the composite web services considering all above constrains and dynamism of the cloud. In this approach composite services are schedule at cloud, node and then at instance level. In this work, a multi-level chromosome based genetic algorithm is presented to schedule composite web services and to study its effectiveness. The paper also studies the importance of the cloud locality (inter cloud distance) and node locality (inter node distance) in the proposed scheduling of composite services in multi-cloud environment.

Keywords- web-service composition; Scheduling; Algorithm; Cloud computing; Genetic algorithm.

I. INTRODUCTION

"Cloud computing" [1] is an internet-based computing model whereby a pool of computation resources, usually deployed as web services, are provided in a manner similar to those of public utilities. This model provides an economic way for an enterprise to build new composite web services. With the advancement of cloud and service oriented architecture (SOA) technology, large numbers of web services are available in the cloud with diverse objectives. These web services are glued together (orchestration or Choreography) to form composite services. Composite services are designed and represented with the help of abstract web services, rather than concrete services. These abstract services need to be mapped or scheduled to its appropriate concrete service instances running on a node of the cloud. For a given functional requirement, there are very large numbers of candidate web services available in the cloud. There could be many instances of a candidate service, hosted on different nodes spread across more than one cloud. Each of those hosted web service instances can have different processing speed, further at any given time, each running service instance can have different load (i.e. number of requests being served). The complete scenario is presented in the figure 1. In such a dynamic cloud environment there is a need for an efficient scheduling approach, which considers dynamism of the cloud while scheduling composite web services. It is also equally important in a multi cloud environment to study the effect of cloud locality under different load scenarios.

In order to execute efficiently any composite service in a multi cloud environment, each of the participating abstract service needs to be allocated and scheduled to an appropriate service instance running on clouds. This allocation ensures that all participating abstract services of the composite services are assigned to appropriate service instance and the scheduling ensures that the composite services are executed in the minimum time. Some of the important challenges in the scheduling of composite services in multi cloud environment are as follows.

1. The scheduling in the cloud is challenging as it is NP hard problem because of the very large number of nodes present in participating clouds. Further, for each abstract service of the composite service has many candidate services and each candidate service can have many instance services hosted on different nodes spread across different participating clouds.

2. The scheduling of the composite services in a multi cloud environment is multi-level in nature. First, appropriate cloud needs to be selected based on the criteria of cloud locality (i.e. data transfer cost from the previous cloud to current node needs to be considered). Second, appropriate nodes need to be selected based on the criteria of node locality (i.e. data transfer cost from the previous node to current node within a cloud needs to be considered) and node processing speed. Finally, appropriate running service instance needs to be selected based on the processing speed and the load (i.e. number of request currently being processed by the instance) on the nodes.

3. The composite service can have many participating services which can be scheduled in parallel. The scheduler should preserve the precedence constraints of services within composite services at the same time should exploit the parallelism to its maximum.
Figure 1 represents the multi cloud environment for scheduling of the composite service. The composite service $CS_i$ consists of 4 abstract service ($AS_i_A, AS_i_B, AS_i_C, AS_i_D$). There are 2 clouds $C_1$ and $C_2$. In Cloud $C_1$ there are 2 nodes ($n_{11}, n_{12}$) and in the cloud $C_2$ have 3 nodes ($n_{21}, n_{24}, n_{25}$). Each node in the clouds host different number of service instances. For example node $n_{12}$ has 2 service instances while $n_{25}$ have only 2 instances. What is required, then, is to allocate the each abstract service of the composite services to the appropriate service instance. The scheduling determines the appropriate service instance with respect to execution time so that overall execution times of composite services are minimized. Data from one service are transferred to another service instance. This is represented with dashed line. In the multi cloud environment this is very important factor, which should be considered while scheduling. Further, the transfer of data from one node to another node within clouds also adds substantial delay in the execution; simply because there is a possibility that the routers may separate the nodes within a cloud.

II. RELATED WORK

Reference [1] maps web service calls to potential servers using linear programming. Their work is concerned with mapping single workflows using different business metrics and a search heuristic. Reference [2] solved a new multiple composite web services resource allocation and scheduling problem in a hybrid cloud scenario where there may be limited local resources from private clouds and multiple available resources from public clouds. They have presented a random-key genetic algorithm for the resource allocation and scheduling problem. The algorithm handles both problems simultaneously. They tested empirically and the experimental results have demonstrated good scalability and effectiveness. Reference [3] presents a dynamic provisioning approach that uses both predictive and reactive techniques for multi-tiered Internet application delivery. However, the provisioning techniques do not consider the challenges faced when there are alternative query execution plans and replicated data sources. Reference [4] presents a feedback-based scheduling mechanism for multi-tiered systems with back-end databases, it assumes a tighter coupling between the various components of the system.

Typically, schedulers exist at two levels [8]. The first level is that of the local scheduler. At this level, decisions are made for each individual resource. The second level is that of the meta-scheduler. At the meta-scheduler level, several parallel applications are scheduled and undesirable interaction between these applications eliminated [13]. In order to schedule a parallel program, two pieces of information are required: information about resources and information about the application. Considerable work has been done in obtaining information about an application and making it of manageable proportions. There have been many efforts to characterize an application's structure by means of compile-time and run-time analysis of the program [9]. Information about resources has been a subject of interest and there are many existing systems that collect this information. There are tools like NWS (Network Weather Service.) that monitor and report information about network traffic. The Globus [9] project has developed the MDS protocol to store dynamic information about resources (processors and network) and GIIS to provide a query interface to locate resources of interest. [10, 11] defines "superscheduling" to consist of three phases: resource discovery, resource mapping and job startup. Our mechanism for scheduling fits into the resource mapping. For a given requirement, there is large number of candidate web services. Each web services have many instances running on nodes of different clouds. For a given composite web service, we attempt to map all abstract services in the incoming composite service to its appropriate service instances running on different nodes of different clouds such that the execution time of the given composite service is minimum.

Reference [5] proposed a concept of web service families, which are created to capture and manage the dynamic nature of web services in groups. Each web service family corresponds to one or many web service instances.
with a common business objective and same input/output structures. Reference [6] also designed an abstract execution machine, called Web Service Dynamic Execution Machine (WSDE machine). The WSDE machine is capable of executing composite web services having both deterministic and non-deterministic flows, where the composite web services are represented using a graph-based semantic based formal model named WSDE graph. The composite web services are represented with help of web service families as a Web-Service Dynamic Abstract Model (WSDAM) graph [6]. The WSDAM graph is executed in the WSDE machine, which uses the dynamic business web service (DBWS) tree for web service discovery. DBWS tree, which organized all web service families in a tree structure. The WSDE machine lacks the capability of efficient scheduling of web services. This paper proposes a suitable scheduling approach, which can also be used by WSDE machine. It should also be noted that what we propose in this paper could also be used as a general web service mechanism. A quite similar study has been done on computational grid [12], where computational nodes are scheduled for computing but this paper schedule web service instances running on computational nodes.

A lot of similar work has been done in scheduling jobs in grid and cloud but the scheduling of abstract web services of the composite web services in cloud are little different. In job or task scheduling, the nodes are selected whereby a task is executed. It is more related to computational computing. Though the web service scheduling also involved computational but prior to scheduling of abstract web services, web services are hosted on the different cloud nodes dynamically by using some task or job based computational scheduling. Once web services are up and running in different nodes across the cloud, appropriate running instances across the clouds needs to be selected for each abstract service based on performance.

Most of the previous work did not considered the data transfer time between previous service and current service during scheduling, which is very pertinent in multi cloud environment. Further, the previous works lacks the practical approach of the way in which the cloud based web services are hosted. In the practical environment, single node can host multiple service instances and each service instance have different load (i.e. maximum number of concurrent requests a node can handle and number of request being currently served). To best of our knowledge, scheduling in such environment is not considered in previous works.

III. PROBLEM FORMULATION

A composite web service (CWS) is represented by directed acyclic graphs (DAG). Each composite service consists of more than one abstract web service. Consider \((A_{S_1}, A_{S_2}, A_{S_3}, \ldots, A_{S_n})\), are abstract web services for the \(i^{th}\) composite web service \(CWS_i\), where \(A_S\) represent abstract web service and \(n\) is the number of abstract web services participating in the composite service \(CWS_i\).

A set of candidate services for an abstract web service \(A_{S_i}\) is represented as \(\{C_{S_j}, C_{S_{j2}}, C_{S_{j3}}, \ldots, C_{S_{jm}}\}\), where \(m\) is the number of candidates services of \(A_{S_i}\).

A set of web service instances running for each candidate service \(C_{S_j}\) is represented as \(\{I_{C_{S_{j1}}, I_{C_{S_{j2}}, I_{C_{S_{j3}}, \ldots, I_{C_{S_{jm}}}}\}}\), where \(p\) is the number of instance services and \(r\) represents the cloud in which it is hosted where \(1<=r<=t\).

A set of clouds is represented as \(\{C_1, C_2, C_3, C_4, \ldots, C_t\}\), where \(t\) is the total number of clouds participating. A set of nodes in the cloud \(C_r\) is represented as \(\{n_{r1}, n_{r2}, n_{r3}, \ldots, n_{r_y}\}\), where \(y\) is the total number of nodes in the \(C_r\). The set of running instances in the \(s^{th}\) node \(n_{rs}\) in the \(r^{th}\) cloud is represented as \(\{in_{rs1}, in_{rs2}, in_{rs3}, \ldots, in_{rsz}\}\), where \(z\) is the number of instances in the node \(n_{rs}\).

The allocation and scheduling plan for a composite web service \(CWS_i\), is represented as \(X = \{Xi | i = 1; 2; \ldots; n\}\) such that the total cost \(Cost(X)\) is minimal. Let \(Xi\) denote an allocation and scheduling plan for each of the abstract service of the \(CWS\), such that \(Xi = (\{M_{rij1} ; F_{rij1}\}; (M_{rij2} ; F_{rij2}); \ldots (M_{rijk} ; F_{rijk}))\), where \(M_{rijk}\) represents the \(i^{th}\) selected web service instance running on the \(j^{th}\) node \(N_j\) on the cloud \(C_r\) for abstract web service \(A_{S_i}\) of the \(CWS\) and \(F_{rijk}\) represents the execution time of an instance of abstract web service \(A_{S_i}\) on the \(j^{th}\) web service instance of the \(r^{th}\) node of cloud \(C_r\). Equation I gives the definition of total cost for a composite service.

The cost constraint problem-

An allocation and scheduling plan \(X = \{Xi | i = 1; 2; \ldots; n\}\), such that the total response time \(Time(X)\) is minimal.

\[\text{Time}(X) = \sum (F_{rijk})\] (1)

Where \(k = 1\) to \(n\), \(n\) is the number of abstract service.

Cloud Locality- In a multi cloud environment, the selected nodes on which services are hosted can be in different clouds. This involves costly data transfer across the clouds over the networks. For the scalability of composite services, it looks is not always preferable to execute in a single cloud. If the nodes in one cloud are over loaded then it might be preferable to select nodes from other clouds. There is a tradeoff between executing in a single cloud and executing in multiple clouds. For the purpose of executing in a single cloud case, scheduling algorithm can be executed for individual cloud to estimate the execution time of each cloud and then select the cloud, which gives the minimum execution cost. This approach looks theoretically not scalable for very big composite services. To make the scheduling more scalable, it looks multi cloud environment should be used, where nodes of all clouds should be explored to find the optimum scheduling. Therefore it is very important to analyze and study whether the scheduling in multi cloud collectively will be preferable as compared to executing in the most efficient cloud. The cloud locality in a multi cloud environment is the selection of appropriate
cloud based on the data transfer time so that overall execution time is minimize as compared to scheduling in a single cloud.

Maximum number of request which can be handled by a node \( n_i \) in cloud \( C \) is represented as \( MR_i \) and number of request being currently served by a node \( n_i \) in cloud \( C \) is represented as \( CR_i \). The load on the node \( n_i \) in the cloud \( C \) is represented as \( \text{Load}_i \), which is equal as the difference between \( MR_i \) and \( CR_i \).

\[
\text{Load}_i = MR_i - CR_i
\]

The load of the \( r \)th cloud \( \text{CLoad}_r \) is defined as the average of all loads of its nodes.

\[
\text{CLoad}_r = \text{AVERAGE} (\text{Load}_{1r}, \text{Load}_{2r}, \text{Load}_{3r}, \ldots \text{Load}_r).
\]

1- The \( \text{CLoad}_r \) is High i.e. when \( 20\% \leq \text{CLoad}_r \leq 50 \% \)
2- The \( \text{CLoad}_r \) is Low i.e. when \( 50\% \leq \text{CLoad}_r \leq 90 \% \)

With changing load of the clouds, the impact of the cloud locality looks more important. To analyze the cloud locality under different load we run our genetic scheduling algorithm under two loads scenarios. First when load was low and next when load was high.

IV. MULTI-LEVEL CHROMOSOME BASED GENETIC ALGORITHM

We use a multi-level chromosome based Genetic algorithm (GA) to solve the cloud allocation and scheduling problem in a multi cloud environment. Its main features include:

1) Multi-level chromosome is used to present the multi-level solution space requires for selecting cloud, node and service instances. A single value of the gene is not very efficient to represent the solution space in multi cloud environment.
2) The fitness function is based not only on the processing speed of the node but also on the load on the node and the communication cost.
3) The logic for selection of appropriate cloud based on cloud locality.
4) The scheduling algorithm exploits the parallelism of abstract services in composite web services to its full extent.

A. Multi-Layer Genetic Encoding

Computer simulation of GA makes a population of abstract representations (chromosomes) of candidate solutions (individuals) of an optimization problem evolve toward better solutions based on it parametric instances of mutation and crossover. In multi cloud environment, each cloud has many nodes, which host more than one web services instances. A single abstract service can have many candidates services, each candidate service are hosted on different instances of different nodes, which are spread across the clouds. The scheduling in such multi cloud environment needs to selects not only clouds but also the appropriate node instances. The single value chromosome is not very suitable to represent the solution space. A multi-level chromosome is proposed to represent the solution space. The higher level chromosome is used to present the clouds and within each higher level chromosome, nodes instances are represented. The composite web service allocation has the structure as \( (AS_1 (C_i, n_j, in_k), AS_2 (C_i, n_j, in_k), \ldots AS_n (C_i, n_j, in_k)) \) where \( AS_n \) represents the \( n \)th abstract service of a composite service. \( C_i \) represent the cloud selected for the abstract service, \( n_j \) represent the node selected within cloud \( C_i \) and \( in_k \) represent the instance selected with the selected node \( n_j \).

Figure 2 – Multi level Chromosome

Figure 2 represents a chromosome for a composite service having 3 abstract services. It is represented using a chromosome of size 3. Each gene of the chromosome has an inner chromosome of size 3 to represent the cloud, node and instance solution space. In this sample chromosome, the value of gene \( AS_1 \) is represented as \( (I_3, N_1, C_1) \) representing the fact that the first abstract service is scheduled in the 3rd instance of the 1st node of the 1st cloud. Similarly it has values for other 2 abstracts services.

B. Fitness Function

The fitness function is very important in the GA. The fitness function is represented as equation (2). The fitness function used in this GA considered not only the execution time on the node based on the processing speed but also the load and cloud locality factor. The cost in transferring the data among the cloud is considered in the fitness function, which helps in selecting appropriate cloud based on the network.

\[
\text{Fitness(X)} = \text{IET}_{ijklm} + \text{IDT}_{ijklm} + \text{ICT}_{ijklm} \tag{2}
\]

Where

\[
\text{IET}_{ijklm} = \left( I_i \times \left( \frac{1}{f_k} \right) \right)
\]

\[
\text{IDT}_{ijklm} = \frac{\phi (\text{MSR}_{lm} - \text{CSR}_{lm})}{(\text{MSR}_{lm} - \text{CSR}_{lm})} \text{iff} (\text{MSR}_{lm} - \text{CSR}_{lm}) \neq 0
\]

Where \( 1 \leq \phi \leq 2 \)
And

\[ IET_{ijklm} = \alpha \cdot B_{ij} \]

\[ ICT_{ijklm} = \text{Instance communication time from its previous execution node of an } i^{th} \text{ instance web service of } j^{th} \text{ candidate service of a } k^{th} \text{ abstract service running on node } l \text{ of cloud } m. \]

\[ IDT_{ijklm} = \text{Instance delay time of an } i^{th} \text{ instance web service of } j^{th} \text{ candidate service of a } k^{th} \text{ abstract service running on node } l \text{ of cloud } m. \]

This delay is due to the internal scheduling delay of the node.

\[ MSR_{ln} = \text{Maximum service request possible on node } n_{ln}. \]

\[ CSR_{ln} = \text{Current number of request on node } n_{ln}. \]

C. Mutation and Cross Over

The algorithm adopts the one point crossover. The mating takes place by swapping the portion of the chromosomes after a randomly selected point known as crossover point. Thus for the crossover of any two chromosomes, it results in offspring which resembles with one parent till the crossover point and with another one after the crossover point till the length of the chromosome.

The top 2% best chromosome where migrated to next generation without any changes. 83% of new individuals in the next generation are generated by crossover operator. 15% of new chromosomes in the next generation are randomly generated.

V. EXPERIMENTS

Simulation experiments were conducted in matlab 2009a to evaluate the effectiveness and scalability of our algorithm. The experimental settings for our multi-level GA are as follows: The initial population size and the maximum generations were 100 each. The cloud information was stored in data structure named as cloud table which we assume to hold all real time data of cloud. Cloud Table (CT) = \{node_number, clock_frequency, max_request, curr_number_req_served\}. The inter cloud and inter nodes distances were also captured in a data structures. The paper executed two set of experiments.

In the first experiments the multi-level chromosome based genetic algorithm is compared with random approach (RA) and best processing node (BPN) based approach to study the effectiveness of the proposed GA approached. In the (RA), for each abstract service, instances are randomly selected and its execution cost is calculated. In the BPN, among the all possible feasible candidate nodes hosting services instances, the nodes with high processing speed is selected. This set was executed repetitively by increasing the number of abstract services in the composite service and number of nodes in each participating clouds in the multiple of 5. The number of cloud was also increased in multiple of 2.

In the second experiment, the analysis of cloud locality was carried out. It is assumed that each cloud has 50 nodes and maximum numbers of instances available across all clouds are 100. For the given composite service, the Genetic algorithm was executed one by one for each participating clouds and then it is executed for all cloud collectively. This experiment set up was executed under two scenarios. In the first scenario, the load on each cloud was less < 50% and in the second scenario the load on each cloud was more than 50%. The objective was to study execution time of each cloud individually as compared to executing in multi cloud collectively under different loads on the nodes.

VI. RESULTS AND ANALYSIS

The Figure 3 represents the output for the first experiment to determine the efficiency of the genetic algorithm based approach as compared to RA and BPN. The result re-established the facts in context of the scheduling of composite services in multi-cloud environments that the genetic algorithm based approach gives far better result as compared to two other approaches. The RA and BPN results are not consistent. They outperform each other randomly depending on the cloud environments and the loads on nodes.

![Figure 3](image)

Figure 3- Execution time vs number of abstract services.

The Figure 4 represents the study of the cloud locality. The outcome established the fact that the cloud locality has impacts on the overall execution time. The communication cost is considerably high as compared to the processing time. Figure 4 shows the execution time when the composite service was executed individually in each cloud using the GA. The 11th cloud instance on the x axis displays the execution time when it was scheduled and executed in all cloud collectively. In Figure 4, the 5th cloud instance has the minimum execution time when the composite service was executed on it.

![Figure 4](image)

Figure 4- Execution time vs cloud instances.
This second experiments was repeated by increasing the number of clouds in multiple of 5. The table I represents the results for all iteration which was executed under low load and high load scenarios. The number of clouds was increased in multiple of 5 keeping the number of abstract service in the composite service as 100 and number of clouds 50 in each cloud. The column individual cloud represent whether the minimum execution time was achieved when it was scheduled in a single cloud or when scheduled on the multi clouds. All results shows that the scheduling on an individual cloud has always better result as compared to scheduling it on the multi cloud collectively.

<table>
<thead>
<tr>
<th>No of Clouds</th>
<th>low load</th>
<th>high load</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual Cloud</td>
<td>Multi Cloud</td>
</tr>
<tr>
<td>5</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>15</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>20</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>25</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>30</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>35</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

VII. CONCLUSION

The paper initially presented an approach for scheduling composite web services in multi cloud environment using multi-level chromosome based genetic algorithm. Not only did the algorithm consider the processing speed of nodes but also the loads on the nodes in terms of number of request being processing by the nodes. The algorithm preserves the precedence constraints of services within composite services and exploits the parallelism to its full extend. The paper through experiment reestablished the facts in context of the web service scheduling that the genetic algorithm based approach gives far better result as compared to two other approaches based on random and best processing nodes approaches. Further, the paper analyzed and studied the impact and importance of cloud locality in the scheduling of composite services. It established the fact that the scheduling on individual cloud has always better result as compared to scheduling it on the multi cloud collectively. Therefor in a multi cloud environment, it is better to select best performing clouds under the given loads rather than scheduling composite services on multi clouds collectively.

The future study shall include extending the present work for the scheduling of multiple composite services in multi-cloud environment.

REFERENCES