QoS-aware Deployment of Network of Virtual Appliances across Multiple Clouds

Amir Vahid Dastjerdi, Saurabh Kumar Garg, and Rajkumar Buyya
Cloud Computing and **D**istributed **S**ystems (CLOUDS) Laboratory,
Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, VIC 3010, Australia,
[amirv, sgarg, raj]@csse. unimelb.edu.au

***Abstract*—Cloud computing paradigm allows on-demand access to computing and storages services over the Internet. To solve the complexity of application deployment in Cloud infrastructure, virtual appliances, pre-configured, ready-to-run applications are emerging as a breakthrough technology. However, an automated approach for deploying network of appliances is required to guarantee minimum deployment cost, low latency, and high reliability. In this paper, we propose and compare two different deployment approaches: Forward-checking-based backtracking (FCBB) and genetic-based. They take into account Quality of Service (QoS) criteria such as reliability, data communication cost, and latency between multiple Clouds to choose the most appropriate combination of virtual machines and appliances. We evaluate our approach using a real case study and different request types. Experimental results show both algorithms reach near optimal solution. Further, we investigate effects of factors such as latency requirements, and data communication between appliances on the performance of the algorithms and placement of appliances across multiple Clouds.**

Keywords- Cloud computing; Virtual appliance; Quality of service; Service-Level Agreements (SLA);

I. INTRODUCTION

Cloud computing is "a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on-demand to external customers over the Internet" [2]. Clouds can be classified according to their service types and deployment models [3].

As noted by Ian Foster et al. [2], clusters, supercomputers, and partially grid relied on non-Service Oriented Architecture (SOA) application, while Cloud focuses on Web 2.0 and SOA. Although Clouds adopted standard communication protocols such as HTTP, and SOAP, the integration and interoperability of all services and finally service deployment remain as major challenges. To overcome deployment problems such as root privilege requirements and library dependencies, virtual appliance technology is adopted as a major Cloud component [2]. Virtual appliances are a set of virtual images including optimized operating systems, pre-built, and pre-configured, ready-to-run applications which proved to be a better service deployment solution [5].

Cloud deployment includes two main phases: discovery and selection. In the discovery phase, all virtual units and appliances that satisfy users' goals and Quality of Service (QoS) requirements are retrieved. Then, in the selection phase, all the combinations of virtual units and appliances are evaluated and ranked based on user preferences and the top combination in the ranked list are returned as the best composition. A framework for service discovery and deployment in Cloud was introduced in our previous work [9]. In this paper, we focus on QoS-based virtual unit and appliance selection.

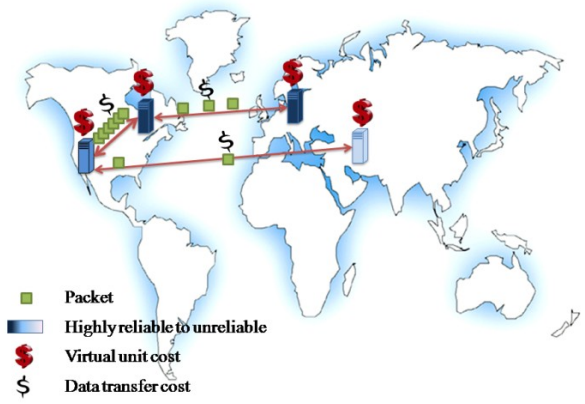


Figure 1. Network of Virtual Appliances Deployment in Multiple Clouds Environment

We investigate a solution considering user's objective and constraints that offers a selection strategy for deploying a group of connected appliances (as shown in Figure 1) in an environment where multiple Clouds are offering their services in the form of virtual units and appliances. In the selection problem, we have users' requests with different latency, reliability and budget constraints, and the objective of minimizing the deployment cost. Moreover, we have various combinations of appliances and virtual units in the Cloud market. The problem is to find a composition that adheres to the user constraints and minimizes the cost of deployment. The deployment problem maps to multi-dimensional knapsack problem due to multiple QoS constraints. The Multidimensional Knapsack problem is classified as NP-hard optimization problem [11]. It consists of selecting a subset of alternatives in a way that the total profit of the selected alternatives is maximized while a set of knapsack constraints are satisfied.

Since by migrating to Cloud, Cloud customers are moving their data and services out of their direct control, they are

978-0-7695-4622-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CloudCom.2011.62

typically concerned about the reliability of Cloud providers' operations. However, this QoS dimension has not been investigated in Cloud provisioning studies [7, 8]. Moreover, since multiple providers are offering different appliances and virtual units with different pricing in the market, it is important to exploit the benefit of hosting appliances on multiple providers to reduce the cost and provide better QoS. However, this could be only possible if high throughput and low latency can be guaranteed among different selected Clouds. Therefore, the latency constraint between nodes has to be considered as another QoS criteria in the selection problem. This paper considers reliability and latency between virtual appliances as the main QoS criteria for designing service selection strategy. In our work, we carefully model the QoS criteria in the problem and then tackle it by two different approaches namely genetic-based and Forward-checking-based backtracking (FCBB) algorithm.

Furthermore, the paper shows effects of data transfer rate between appliances on the performance of algorithms and compares effects of latency requirements and data transfer rates between appliances on their placement across multiple Clouds.

The major contributions of this paper are: 1) modeling relevant QoS criteria, namely as latency, cost (data transfer cost, virtual unit, and appliance cost), and reliability for selection of the best virtual appliances and units in Cloud computing environment, 2) presenting and evaluating two different selection approaches to help users in deploying network of appliances on the multiple Clouds based on their QoS preferences. For that purpose various types of requests (with different network load between appliances) are generated, and data from 12 real Clouds was collected, and 3) investigating effects of factors such as latency requirements, and data communication on the cost of appliance placement and the selection of providers.

II. MOTIVATION SCENARIO AND CHALLENGES

To study user requirements and concerns for deploying a network of appliances on Clouds, we give an example of a real world case study with known network traffics between appliances. A good example of network of virtual appliances (a set of appliances in the form of a connected graph which have data communication among them) is multi-tier applications supporting web-based services. Each tier has communication requirements as characterized in the research conducted by Diniz Ersoz et al. [1]. They considered a data center with 11 dedicated nodes of a 96-nodes Linux cluster and host an e-business web site encompassing 11 appliances: 2 front-end Web-Servers (WS) in its web tier, 3 Databases (DB) in its database tier and 6 Application Servers (AS) in between.

An administrator of the e-Business web site might be interested in migration of the appliances to the Cloud in order to save on upfront infrastructure and maintenance costs, as well as to gain the advantage of on-demand scaling. In addition, to allow disaster recovery and geography-specific service offering, he prefers multiple Cloud deployment. For such deployment, he faces several challenges such as: 1) what is the best strategy for placing appliances across Cloud providers? Should they be placed based on the traffic they exchange, therefore placing those with higher connectivity closer to each other to decrease latency and data transfer cost?, 2) is it economical to do so? (how much is the Cloud deployment cost), 3) if appliances are placed across multiple providers, how the latency between different providers affects the web site performance?, and 4) how can the most reliable Cloud services be selected for the deployment?

In this work, our proposed algorithms help to answer the above questions by selecting the most suitable virtual appliance and virtual unit services based on the application requirements. Our solution minimizes the cost of migration to Cloud while considering user's concerns in regards to reliability and latency. In order to translate the user QoS requirements in terms of reliability, latency and communication costs, we have considered three metrics which are explained in the next section.

III. QOS CRITERIA

The three QoS criteria considered in the selection problem are reliability, cost, and latency.

A. Reliability

For measuring Cloud providers' reliability we introduce SLA Confidence Level (SCL) which is a metric to measure how reliable are services of each provider based on the binding SLAs. SCL values are computed by a third party that is responsible for monitoring the SLA of provider based on the following equation:

$$SCL=\sum_{j=1}^{k}(I_j \times SCL_j) \text{ where: } SCL_j=MSQ_{jt} - PV_{jt}$$

Where the SCL_j is SLA confidence level for QoS criteria *j* of a Cloud service; I_j is the importance of the criteria *j* for user; k is the number of monitored QoS criteria. MSQ_{jt} is a is monitored value of quality of service criteria j in the period of *t*. and PV_{jt} is promised value for the QoS criteria in the period of *t*.

We modeled "availability" for SCL generation, as current Cloud providers only include "availability" in their SLAs. For example, a provider with promised availability of 99% for 365 days and monitored availability of 98% for 365 days has a better SCL compared to a provider with same promised availability and 95% of monitored availability. The reliability in our work is considered as a user constraint for each Cloud service.

B. Cost

Cost is a non-functional requirement of a user who wants to deploy a network of appliances on Cloud. In our problem, minimization of deployment cost is considered as the objective of users. The deployment cost in our selection problem includes monetary cost of leasing virtual units as well as appliances and communication costs [20]. The communication monetary cost for connected virtual appliances depends on how much data they exchange and can be determined by the following factors: 1) One time

communication message size and 2) Communication rate (how often two appliances communicate), which can be calculated based on request inter-arrival rate.

C. Latency

Latency can have a significant impact on e-Business web sites performance and consequently on the end user experience. Therefore, we have considered it in the selection problem as one of the user's constraints. It is assumed that customers have different constraints for the latency between appliances which have to be satisfied with the selection of proper Cloud providers.

IV. ARCHITECTURE

As illustrated in Figure 2, deployment process consists of following phases:

- In phase ① Service requestor specifies requirements for each vertex, connectivity, latency constraints between vertices, and hardware requirements like CPU, storage, and memory.
- In phase ② Software requirements used as an input for discovering the best suited appliances among various repositories of virtual appliance providers which named as virtual market place by VMware. Simultaneously Hardware requirement used by ontology-based matchmaker in discovery component [9] to search for the best available virtual units advertised.
- Phase ③ deals with building the Open Virtualization Format (OVF) package and its metadata based on discovered virtual appliances from external appliance providers. The OVF [33] is a hypervisor-neutral (the OVF doesn't rely on the use of specific hypervisor or virtualization platform), and open specification for the packaging and distribution of virtual appliances composed of one or more VMs.
- During phase ④ the selection component uses user preferences regarding latency, SCL and cost to select the best virtual appliance and virtual unit combination for the group of appliances connected together. Consequently, in this phase the selection component sends a query to the monitoring third party and acquires the associated SCL to each service provider.
- Finally in phase ⑤ SLA will be negotiated and contract will be achieved between the SLA managers and selected virtual unit and appliance providers. Enforceable SLA will be signed by both parties and is kept in a repository and continuously monitored by the third party.

V. PROBLEM FORMULATION

The Cloud selection problem consists of finding the composition of appliances and virtual units for the customers that minimizes the deployment cost and adheres to the reliability and latency constraints. In this section the problem is formally defined.

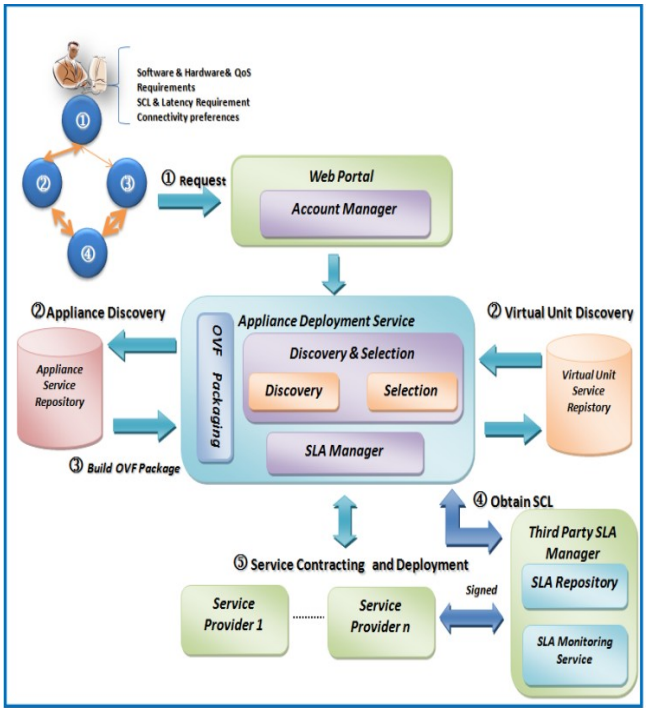


Figure 2. Architecture of Appliance Deployment in Multiple Cloud Environment

- **Provider model**

Let m be the total number of providers. Each provider is represented as

$$P_k: (\{a\}, \{vm\}, Cdata_{in}(P_k), Cdata_{out}(P_k));$$

Where a, vm, $Cdata_{in}(P_k)$ *and* $Cdata_{out}(P_k)$ denotes appliance, virtual machine, Cost of internal data transfer and Cost of external data transfer respectively. A virtual appliance *a* can be represented by a tuple of four elements: appliance type, cost, license type, and size.

$$a: \{ApplianceType; Cost; LicenseType; Size\}$$

A virtual machine *vm* can be formally described as a tuple with two elements as shown below.

$$vm: \{MachineType; Cost\}$$

- **User request model**

The user request for deployment of his application can be translated into a connected graph G (V, E) where each vertex represents a server (virtual appliance running on a virtual unit). Server corresponding to a vertex v is represented as:

$$S_v = \{appliance, virtual\ unit\} = \{a_v, vm_v\}, \forall v \in V$$

Each edge $e\{v, v'\} \in E$ indicates that vertex v and v' are connected. The data transfer between these connected vertexes (i.e., one server to another) is given by D.

The objective of user is to minimize the deployment cost of his whole application on multiple Cloud providers' infrastructure, given a lease period of "T" (unit) and budget b. The users has constraint for reliability (SCL_v) of the provider on which server should be hosted and also latency

constraint ((*L(e{v,v'}) where v,v'* $\in V$) that represents maximum acceptable latency between servers. The cost of renting a server includes the cost of virtual unit and virtual appliance. Let appliance for S_v be rented from provider P_k and virtual unit from provider P_l. The cost of server S_v as shown below is cost of appliance ($cost(a_{v,P_k})$) and virtual unit ($cost(vm_{v,P_l})$ plus cost of transferring the appliance if the appliance and virtual unit providers are not same.

$$Cost(S_v) = \begin{cases} \left(cost(a_{v,P_k}) + cost(vm_{v,P_l})\right) \times T & if\ k = l \\ \left(cost(a_{v,P_k}) + cost(vm_{v,P_l})\right) \times T + & \\ size(a_{v,P_k}) * Cdata_{out}(P_l) & if\ k \neq l \end{cases}$$

Let S_v={ $a_{v,Pk}$, $vm_{v,pl}$ } and $S_{v'}$={ $a_{v',Pk'}$, $vm_{v',pl'}$ } be two connected vertexes (servers) by edge $e\{v, v'\} \in E$; and P_k , P_l, $P_{k'}$ and $P_{l'}$ are the providers using whose resources Servers S_v and $S_{v'}$ are deployed. The data transfer cost between two servers is given by:

$$DCost(e\{v, v'\}) = \begin{cases} DSize(e) \times \left(Cdata_{out}(P_l) + Cdata_{out}(P_{l'})\right) \times T & if\ l \neq l' \\ DSize(e) \times Cdata_{in}(P_l) \times T & if\ l = l' \end{cases}$$

Therefore, the total cost of hosting user's application on the multiple clouds is given by:

$$Total\ Cost = TC = \sum_{v \in V} Cost(S_v) + \sum_{\substack{e \in E \\ v,v' \in V}} DCost(e\{v, v'\})$$

- **Problem Formulation**

The objective of the user is to minimize the deployment cost of his whole application on multiple cloud infrastructure *(P_k.0<k<m)*. Thus, the mathematical model is given by:

$$Min(TC)\ Subject\ to: 0 < TC < b$$

$$Latency(S_v, S_{v'}) < L(e\{v, v'\})\ \ where\ \forall e\{v, v'\} \in E$$

$$SCL(S_v) < SCL_v\ \ where\ \forall\ v \in V$$

Where, $Latency(S_v, S_{v'})$ is the latency between Cloud infrastructures where server S_v and $S_{v'}$ are hosted, and *SCL(S_v)* is the reliability of the Cloud infrastructure where server S_v is hosted.

VI. ALGORITHMS

To tackle the mentioned problem, one may consider a greedy algorithm [8]. However, it cannot be directly adopted to solve the selection problem, as it is not capable of satisfying the budget constraint and latency constraints between vertices. Another approach which can be used to solve the problem is finding all possible compositions using exhaustive search, comparing their overall cost, and selecting the composition with the lowest cost that satisfies budget, reliability, and latency constraints. This approach can find the optimal solution; however, the computation cost of the algorithm is high due to NP hardness of the problem [8,18]. In order to deal with the aforementioned challenges in following we describe two selection algorithms: Forward-checking-based backtracking (FCBB) and the genetic-based Cloud virtual appliance and unit selection.

A. Forward-checking- based -backtracking (FCBB)

In FCBB the process of searching providers begins from a start node (vertex) S_v which has minimum deployment cost (including appliance and virtual unit cost) and for all its children there can be found at least one provider that satisfies all constraints(partial forward checking) [Algorithm FCBB1: lines 14-18]. The partial forward checking on the problem constraints is added to the algorithm to avoid back jumps in the circumstances where latency constraints of the users are comparatively tight.

Algorithm FCBB

```
1: FCBB(Sv,Request G (V, E))
2: if (Sv=theFirstStartNode)
3: |  Sv= getSartNode(Request G (V, E) , processedSet)
4: |  processedSet← Sv ∪ processedSet
5: |  selection(Sv);
6: |  if(selection(Sv')=null) then Backtrack endif;
7: endif
8: connectedNotProcessed=  getConnectedNotProcessed(parentNode,Request G (V, E) , processedSet)
9:  for each Sv'  in  connectedNotProcessed
10: |  selection(Sv');
11: |  if(selection(Sv')=null) then  Backtrack endif;
12: endfor
13: for each Sv'  in  connectedNotProcessed
14: | FCBB(Sv');
15: endfor
```

Algorithm FCBB.1: Function Selection

```
1: function  selection (Sv)
2: mincost=∞;constraintsViloated= false; feasible=true;
3: for  each  Combination in  getAllCombinationsSorted(Sv)// return combination sorted using quick sort.
4:  |  if (SCL(Sv)< SCL(Combination.getVUprovider()) &&  SCL(Sv)<=
       |    SCL(Combination.getApprovider()) ) then
5:  |    connectedProcessed=  getConnectedProcessed(startNode ,Request G (V, E) , processedSet);
6:  |  if (connectedProcessed!=null) then        // Backward Checking
7:  |  for each Sv'  in  connectedProcessed
8:  |  |  if (  Latency(Sv, Sv') > L(e{Sv, Sv'})) then
9:  |  |  |  constraintsViloated= true;
10: |  |  endif
11: |  endfor
12: |  if (constraintsViloated= false) then
13: |  |  connectedNotProcessed= getConnectedNotProcessed(startNode ,Request G (V, E) ,processedSet)
14: |  |  for each Sv'  in  connectedNotProcessed
15: |  |  |  if   ∄ Combination in getAllCombinationsSorted(Sv) that  (Latency(Sv, Sv') < L(e{Sv, Sv'}) )  then
16: |  |  |  |  feasible= false;            // Forward Checking decrease number of backtracking
17: |  |  |  endif
18: |  |  endfor
19: |  |  if  (feasible= true) then
20: |  |  |  cost= communicationCost+Combination.getCost;
21: |  |  |  if (cost < minCost and cost + totalCost < request.getBudget()) then
22: |  |  |  |  minCost = cost;
23: |  |  |  |  selected[Sv]←{ combination.getUnitProvider, combination.getApplianceProvider};
24: |  |  |  endif
25: |  |  endif
26: |  endif
27: |  endfor
28: endfunction
```

Then, S_v is added to the processed node list. After that, the algorithm processes all the children of S_v which are not processed, and for each child of S_v, providers are selected using the selection function [Algorithm FCBB lines:9-12] such that 1) latency and SCL constraints are satisfied with all the connected processed nodes (backward checking) , 2) they can pass forward checking and 3) they have minimum communication (to already processed nodes) and combination cost [Algorithm FCBB1: lines:20-23].

After selection of all the unprocessed children of the start node S_v, the similar search and selection process is done recursively for all the grand children of start node S_v [Algorithm 1 lines: 13-15]. If selection function does not find any set of providers, it moves back and replaces the parent node with the second best set of providers in Combination list (Backtrack) [Algorithm FCBB lines: 6 and 11].

B. Genetic- based Virtual Unit and Appliance Provider Selection

Since genetic approaches have shown potential for solving optimization problems [13], this class of search strategies was utilized in our problem. The adoption of genetic-based approaches for the selection problem involves 4 steps.

The *first* step is to plan the chromosome, which consists of multiple genes. In our problem, each vertex in the graph of request is represented by a gene. The *second* step is to create the population, hence each gene represents a value which pointed to a combination of virtual unit and appliance service (which satisfies requirements of corresponding vertex) in a sorted (based on the combination cost) list. Implementation of fitness function is the *third* step. The fitness values are then used in a process of natural selection to choose which potential solutions will continue on to the next generation, and which will die out. The fitness function as shown in Equation (1) is equal to the total cost of the solution. However, if constraints are violated then the penalty function is applied. Designing penalty function for genetic-based approach is not a trivial task. Several techniques have been applied in our work until a proper penalty function was found that is capable of handling constraints in the problem. The penalty function is constructed as a function of the sum of number of violations for each constraint multiplied by constants as shown in Equation (2). In the penalty function, *Age* is the age of chromosome, *ki* is the constant which differs from a constraint to another constraint, *NVi* is number of cases that violate the constraints and *NNVi* is the number of cases that do not violate the constraints. In addition, to discard the infeasible solutions in early generations, infeasible solutions with lower age are penalized heavier. Finally, the last step is the evolution of the population based on the genetic operator. The genetic operator adopted for our work is Java Genetic Algorithm Package (JGAP) natural selector [14].

$$\textit{Fitness}(\textbf{Chromosome}) = \begin{cases} \left(\sum_{i\in V} Cost(Gene_i) + \sum_{\substack{e\in E \\ i,j\in V}} DCost(e\{Gene_i, Genej\})\right) * \mathrm{T} \\ \qquad if\ cosntarints\ are\ not\ viloated \\ \left(\sum_{i\in V} Cost(Gene_i) + \sum_{\substack{e\in E \\ i,j\in V}} DCost(e\{Gene_i, Genej\})\right) * \mathrm{T} + Penalty(\ ) \\ \qquad if\ cosntarints\ are\ viloated \end{cases} \quad (1)$$

$$Where, Penalty(\ ) = \sum_{i=1}^{n} \left(\frac{NVi}{NVi+NNVi} * ki\right) * \left(\frac{1}{Age}\right) * fitnessvalue \quad (2)$$

VII. Experimental Test Bed Modelling

To evaluate the proposed algorithms and study placement of appliances, essential input data using real experiments was collected. The collected data can be classified either in data for providers modeling, and data for user request modeling.

Providers Modeling: A set of 12 real Cloud providers are selected namely: Amazon, Zerigo, Softlayer, VMware, Bitnami, rpath, Turnkeylinux, Rackspace, GoGrid, ReliaCloud, Lindoe, and Prgmr. Their virtual units and appliances have been modeled in our system. In addition, latency data between Cloud providers and SCL for each of them have been measured. The following subsections describe the data collected in detail.

1) Virtual Unit and Appliance Modeling: We built an aggregated repository of virtual appliance and virtual unit services based on the advertised services by Cloud providers. Services contain information regarding cost, virtual appliance size, and data communication cost inside and outside of Clouds.

2) Latency and reliability (SCL) calculation: The latency data between Cloud providers has been collected over the past three months using the Cloud harmony [23] service. Data collection was conducted twice daily at random times. Tests consist of pinging to determine latency. Table I shows mean latency between EC2 and 3 different virtual unit providers as a sample. Max, min and average of latency between providers are 58.94, 2.51 and 29.88 (ms) respectively. In addition, Panopta (a monitoring tool) is used to supply SCL input data, Table I demonstrates how a sample of SCL input data looks like for 3 Cloud Providers service uptime for the last 365 days.

Generation of requests for experiments: The request generation involves three steps. Firstly, number of servers requested by the user and requirements of each server in terms of virtual unit and appliance types are determined. Next, connected vertices in request are identified. Finally, data transfer rates between connected appliances are identified. For experimental evaluation two classes of requests are used, i.e., a real case study and randomly generated requests.

TABLE I. Latency between Clouds and SCL input data [23]

Cloud A	Cloud B	Latency (ms)	Cloud B Monitored Availability	Cloud B Promised Availability
Ec2	Rackspace	49.8	99.996%	100%
Ec2	GoGrid	8.9	99.999%	100%
Ec2	Lindoe	5.01	99.999%	100%

TABLE II. Request types

Request Type	Request Graph Density	Request Inter-Arrival Rate DB ↔ AS	Request Inter-Arrival Rate WS ↔ As
Strongly Connected	0.85	Log-normal (1.4719, 2.2075)	Weibull (0.70906 ,10.185)
Moderately Connected	0.50	Log-normal (1.1695 1.9439)	Weibull (0.41371 1.1264)
Poorly Connected	0.25	Log-normal (0.8912 1.6770)	Weibull (0.24606 0.03548)

1) modeling user requests using a real case study

For the real case study example, we use the three-tier data centre scenario presented by Ersoz et al. [1]. Required virtual unit and appliance type for each vertex is assigned based on the scenario. They implemented an e-Business web site that encompasses 11 appliances: 2 front-end web-servers (WS) in its web tier, 3 databases (DB) in its database tier and 6 application servers (AS) in between. In their work, three-tier data centre architecture was used to collect the network load between appliances. Each of the nodes have 4 GB of system memory, 64-bit AMD Opteron processors. Two different workloads, RUBiS [25] and SPECjApp-Server2004 [16] are used by them. However, our focus is on the RUBiS which implements an e-Business web site. That web site includes 27 interactions which can be carried out from a client browser. Their analysis of experiments results has been represented by various distributions of request inter-arrival times, and data size between tiers for 15 minutes runs of the RUBiS workload with 800, 1600, and 3200 clients (number of clients causes different request inter-arrival rates which is used to generate different requests types). This data which is shown in Table II is used to calculate the network traffic between connected appliances.

2) modeling user requests for extensive experiment study

Three classes of user requests (network of appliances) namely strongly, moderately, and poorly connected are created as shown in Table II which differs from each other in communicated message sizes, message inter-arrival rates, and graph density (proportion of the number of edges in request graph to total possible number of edges) of the request graph. The reason for building 3 classes of requests is to study the effect of network traffic and request graph density on performance of selection algorithms. For each vertex, we randomly assign a required virtual unit and appliance type, and then we use random graph generation technique to identify which vertices are connected. All generated network of appliances are following the topology presented by Ersoz et al. [1]. Next, based on which types of appliances are connected to each other, data transfer rates are assigned to them according to Table II. For example if one appliance is a database and the other one is application server and the request is in category of strongly connected, then the request inter-arrival rate is Log-normal(1.4719, 2.2075). In addition, to investigate effects of message size on the performance of algorithms, two classes of requests with different message sizes are created using workload 'a' [1] (e-Business application with small message size) and 'b' [24] (98 World cup with large message size).

VIII. EXPERIMENTAL RESULTS

The experiments aim at: a) comparing the proposed heuristics with Exhaustive Search (ES) using the real case study, b) evaluating effects of variation in request types on algorithms performance and execution time, c) analyzing effects of variation in request types and latency constraints on distribution factor.

A. Comparison with Exhaustive Search (ES)

Figure 3 shows how close the proposed algorithms are to the Exhaustive Search (ES) for the case study. Both of them could reach the same solution achieved with ES. As evidenced by Table III, the mean execution time for finding the solution using exhaustive search of solution space is extremely high comparing to our proposed algorithms. The execution time for the ES approach rises further exponentially with the computation effort for larger number of servers and providers; therefore, it cannot be considered as a practical solution for the selection problem with constraints. To further examine the near-optimality of FCBB and the genetic approach, we conducted extensive experiments with 10 different requests (in terms of service requirements, graph density, message size, and request inter-arrival time) for each category of 10, 15, and 20 servers. The results are shown in Table IV, where we can clearly observe that on average, the difference with deployment cost of ES results is just 7% for the FCBB and 1% for genetic approach. Therefore, FCBB and genetic approach can reach the near-optimal solution without much computation cost.

B. Results of variation in request types on algorithms performance and execution time

Figure 4 and 5 depict the performance of our algorithms for different request types (strongly, moderately, and loosely connected) with different number of servers. In the case of workload 'a', as message size is small, differences are not much, except in strongly connected requests (Figure 4a) and especially for the case of 100 servers where genetic-based approach can save approximately 3% of cost. In other cases of workload 'a' when vertices are moderately or poorly connected the genetic-based approach has better or relatively same performance (regarding the cost) compare to the FCBB algorithm. However, when the message size is larger (workload 'b'), as shown in Figure 5a), genetic algorithm in almost all cases outperforms the FCBB algorithm. In Table V, mean execution time for 20 experiments in relation to the number of servers for group of requests is given which shows the execution time of FCBB is negligible compare to genetic. It also shows that adding "forwards checking" feature successfully decreases execution time especially for the requests which require more than 10 servers and therefore it outperform the discard subset algorithm offered in [18] regarding the execution time while they both could result in same objective values for all cases.

Therefore, it can be concluded that the performance of algorithms differs from one workload to another and when there exists a workload with small message size (like the e-Business workload 'a' [1]) performance difference of algorithms is low. In such cases FCBB can be used to save on execution time. However, when the message size increases [24] then they show comparatively higher differences, therefore, when users are looking for the least deployment cost instead of the execution time, the genetic-based approach is more appropriate.

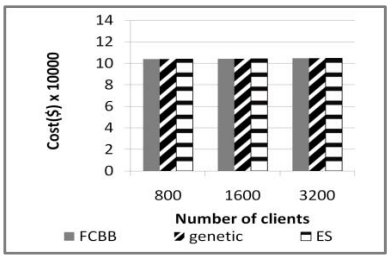


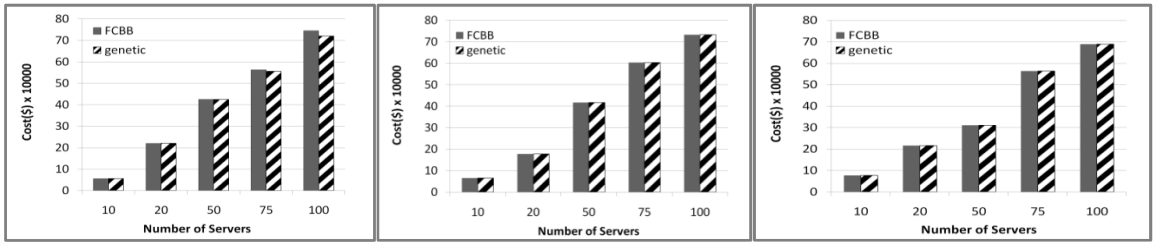
Figure 3. Performance Evaluation for Case Study

TABLE III. MEAN EXECUTION TIME FOR CASE STUDY

Algorithm	Mean Execution time(s)
FCBB	0.102
genetic	36.393
Exhaustive Search (ES)	3248.152

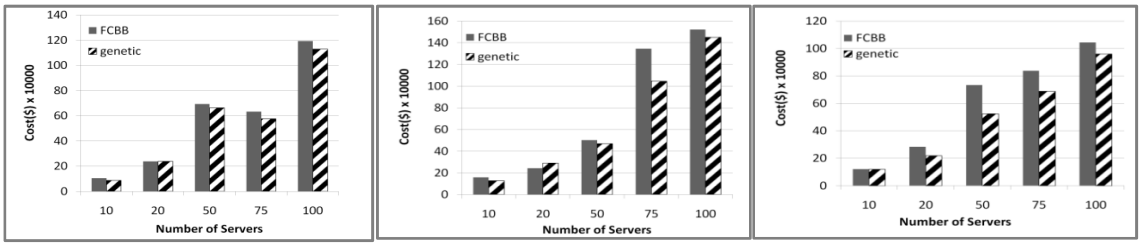
TABLE IV. MEAN EXHAUSTIVE SEARCH(ES) COSTS/ALGORITHMS COSTS

Algorithm	Number of servers		
	10	***15***	***20***
ES/ FCBB	0.9841	0.9175	0.9013
ES/genetic	0.9952	0.9868	0.9923



(a) Strongly connected (b) Moderately Connected (c) Loosely Connected

Figure 4. Change in Connectivity for Workload a



(a) Strongly connected (b) Moderately Connected (c) Loosely Connected

Figure 5. Change in Connectivity for Workload b

C. *Effects of variation in request types and latency constraints on distribution factor*

In this experiment, the objective is to study the possibility of network of appliances placement on different providers rather than one. For this purpose a metric named "distribution factor" is designed, which shows proportion of the number of different providers selected to the total number of providers. Table VI shows how a request type (data transfer rate, and graph density as explained in Table II) affects the distribution factor. For the loosely connected request with loose latency requirement, we can conclude that considering multiple cloud providers decrease the deployment cost while still we can maintain the performance (by adhering to latency constraint). For all cases from 10 to 100 servers when there is higher data transfer and number of connection between vertices the distribution factor decrease dramatically. For majority of cases, it decreases by more than 75 %. It means that FCBB selection algorithms have a tendency to select the same virtual unit provider for all vertices to save on communication cost. The same trend can be observed for the genetic-based approach. When the latency is tight, still if we consider multiple providers for deployment the cost is lower. Nevertheless, the distribution factor decreases by 25%.

Consequently, the experiments show that network of appliances with higher graph densities and data transfer are less likely to be distributed across multiple providers and they are expected to have higher deployment cost.

TABLE V. MEAN EXECUTION TIME (S)

Algorithm	Number of servers				
	10	***25***	***50***	***75***	***100***
FCBB	0.103	0.115	0.288	0.407	0.841
Discarding subset	0.138	0.271	0.849	2.339	6.091
genetic	31.997	144.426	497.377	1288.056	1814.488

TABLE VI. DISTRIBUTION FACTOR

Request type	Number of servers				
	10	***25***	***50***	***75***	***100***
Loosely connected & loose latency	44%	55%	55%	55%	44%
Strongly connected	11%	11%	11%	11%	11%
Tight latency	22%	44%	33%	33%	33%

IX. Related Work

The concept of virtual appliances was originally introduced to simplify the deployment and management of desktop personal computers in enterprise and home environments [26]. Then they have been adapted in Grid and Cluster Computing environments to simplify the deployments [27]. Now with the emergence of Cloud Computing, which utilizes virtualization to provide elastic usage of resources, virtual appliances are becoming the preferred technology to deploy applications on virtual machines with minimum effort.

Sun et al. [5] showed that by utilizing virtual appliances, the deployment process of virtual machines can be made simpler and easier. Wang et al. [29] presented a framework to improve the efficiency of resource provisioning in large data centers using virtual appliances. Similarly, a framework for service deployment in Cloud based on virtual appliances and virtual machines has been introduced in our previous work [9]. That research focused on selecting suitable virtual machines using ontology based discovery model, packaging, and deploying them along with virtual appliances in the Cloud platform, and monitoring the service levels using third parties. In this work, we are concentrating on QoS-based virtual unit and appliance composition where multiple appliances need to be deployed across multiple Clouds with acceptable latency and reliability to achieve users' business objectives.

A single virtual appliance on a virtual unit will not be able to fulfill all the requirements of a business problem. Inevitably we will require more than one virtual appliance and unit working together to provide a complete solution. Hence it is important to develop compositions of virtual unit and appliances. Konstantinou et al. [30] proposed an approach to plan, model, and deploy virtual appliance compositions. In their approach, the solution model and the deployment plan for virtual appliance composition in Cloud platform are developed by skilled users and executed by unskilled users. As cited by them, the contribution has not offered an approach for selection of virtual appliance and machine providers. In our work, however, we consider that users will be only aware of the high level components that are required for the composition to address their business objectives and our solution provides an approach to select the best composition based on their functional and QoS requirements. Similarly Chieu et al. [31] proposed the use of composite appliances to automate the deployment of integrated solutions. However in their work as well, QoS objectives are not considered when building the composition.

Characteristics of the appliance selection and composition in Cloud differ from works done in other contexts such as Grid and web services. Grid Computing aims to "enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations"[2]. Therefore, the QoS management and composition works in this context mainly focus on load balancing (applying queuing theory and market driven strategy [15]) and fair distribution of resources among service requests [21, 22]. Most of these works proposed constraint satisfaction based matchmaking algorithm (CS-MM) and other artificial intelligence based optimization techniques to improve the performance of scheduling. However, In Service Oriented Architecture's (SOA) context the main concern is defining QoS language [6,12] to express user preferences and QoS properties of the service (semantic based [17, 19]). In this context, for automated web services composition, various techniques such as workflow and AI planning have been adapted [32].

However in the context of Cloud computing, selection objective is not a fair distribution of resources between requesters, because in Cloud resources can be considered as infinite [20]. Instead, Cloud customers have emphasized more on QoS dimensions such as reliability, and cost [2]. Therefore, in this work we present a novel way to measure composition reliability and suitability based on Service-Level Agreements (SLA). In addition, the data transfer cost [20] is also included in our deployment cost. The importance of modeling data transfer cost can be realized by the example of deployment in Amazon Cloud where data transfer costs approximately $100 per terabyte. These costs quickly add up and become a great concern for the administrator.

In comparison with our approach for appliance composition, works that applied Analytical Hierarchy Process (AHP) and Multi-Attribute Utility Theory (MAUT) [12], can only perform well when number of explicitly given alternatives is small and number of objectives are limited. In contrast, as shown in section VIII, our approach can deal efficiently with a large number of Cloud services in the repository (1200 services). Similarly another work has utilized Intutitionistic Fuzzy Set (IFS) for ranking service compositions in the context of Grid and SOA [28]. Nevertheless it does not deal with users' constraints such as latency and budget. Moreover, when the problem is NP-hard the execution time is not acceptable. In addition, composition optimization approaches have been categorized by Jaeger et al. [18] to four types namely pattern based, discarding subset, bottom-up selection, and the greedy. Among them discarding subset has the best performance compare to the others when selection problem includes users constraints for QoS criteria. Consequently, we have compared our approach with discarding subset and as shown in section VIII; our approach has lower execution time. In summary, our work is unique in dealing with Cloud specific appliance composition challenges such as placement issues which includes cost of appliance transfer, other ongoing data transfer costs, and inter-cloud latency.

X. Conclusions

In this paper, we investigated Cloud provider selection problem for deploying a network of appliances. We proposed new QoS criteria and the problem of deployment is formulated and tackled by two approaches namely FCBB and genetic-based selection. We evaluated the proposed approaches by a real case study using real data collected from 12 Cloud providers, which showed that proposed approaches deliver near-optimal solution. Next, they were tested with different types of requests. The results show that when message size increases approaches present

comparatively higher differences, and if execution time is not the main concern of users, genetic-based selection in most cases achieves better value for the objective function. In contrast, if the massage size between appliances is small, FCBB can be used to save on execution time. Further, based on conducted experiments, we found out that network of appliances with higher graph density and data transfer are less likely (in contrast to requests with lower data transfer) to be distributed across multiple providers. However, for requests with tight latency requirements, appliances are still placed across multiple providers to save on deployment cost. In future, we plan to investigate integration of SLA-based discovery and selection to our system to further enhance QoS for end users.

REFERENCES

[1] D. Ersoz, et al., "Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center", The 27th Int'l Conf on Distributed Computing Systems , 2007.

[2] I. Foster, et al. , "Cloud Computing and Grid Computing 360-Degree Compared", IEEE Grid Computing Environments Workshop, 2008.

[3] Peter Mell and Tim Grance, The NIST Definition of Cloud Computing, Version 15, 10-7-09.

[4] Amazon EC2, [Online]. Available: http://aws.amazon.com/ec2/

[5] C. Sun, L. He, et al., "Simplifying Service Deployment with Virtual appliances", IEEE Int'l Conf on Services Computing, 2008.

[6] I. Toma, et al., "Modelling QoS characteristics in WSMO", In 1st Workshop on Middleware for Service-oriented Computing, New York, USA, 2006.

[7] M. M. Hassan, et al., "Multi-objective Optimization Model for Partner Selection in a Market-Oriented Dynamic Collaborative Cloud Service Platform", The 21st IEEE Int'l Conf on Tools with Artificial Intelligence, 2009.

[8] W. Zeng, et al., "Cloud service and service selection algorithm research", The 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09), ACM, 2009, pp. 1045-1048.

[9] A. Vahid-Dastjerdi, et al., "An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery", The 10th IEEE/ACM Int'l Symposium on Cluster, Cloud, and Grid Computing, 2010.

[10] H-Q Yu and S. Reiff-Marganiec, "Non-functional property based service selection: A survey and classification of approaches", In Proc. of 2nd Non Functional Properties and Service Level Agreements in SOC Workshop (NFPSLASOC'08), Dublin, Ireland, 2008.

[11] V. Chvatal, "A greedy heuristic for the set-covering problem." Mathematics of Operations Research, 1979, 4(3):233-235.

[12] V. X. Tran, et al. 2009. "A new QoS ontology and its QoS-based ranking algorithm for Web services", Simulation Modelling Practice and Theory 17(8): 1378-1398.

[13] C.A. Coello Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques", Knowledge and Information Systems. An International Journal, 1(3):269–308, 1999.

[14] Jgap [Online].http://jgap.sourceforge.net

[15] X. Wang, K. Yue, J. Z. Huang, A. Zhou, "Service Selection in Dynamic Demand-Driven Web Services", IEEE Int'l Con on Web Services (ICWS'04), IEEE CS Press, 2004.

[16] J2EE. [Online]. Available from http://java.sun.com/j2ee/.

[17] J. M. Garcia, et al., "QoS-Aware Semantic Service Selection: An Optimization Problem", The IEEE Congress on Services, 2008.

[18] Jaeger, M. and G. Rojec-Goldmann, "SENECA – Simulation of Algorithms for the Selection of Web Services for Compositions", In 6th VLDB Workshop on Technologies for E-Services , Norway,2005.

[19] I. Toma, et al., "A Multi-criteria Service Ranking Approach Based on Non-Functional Properties Rules Evaluation", The 5th Int'l Conf on Service-Oriented Computing, Springer, 2007.

[20] M. Armbrust, et al., "Above the Clouds: A Berkeley view of Cloud computing", Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[21] S. A. Ludwig and S. M. S. Reyhani, "Selection Algorithm for Grid Services based on a Quality of Service Metric", The 21st Int'l Symp on High Performance Computing Systems and Applications, 2007.

[22] G. Tapashree, S. A. Ludwig, "Comparison of Service Selection Algorithms for Grid Services: Multiple Objective Particle Swarm Optimization and Constraint Satisfaction Based Service Selection", The 20th IEEE Int'l Conf on Tools with Artificial Intelligence.

[23] Cloudharmony. [Online]. http://Cloudharmony.com

[24] M. Arlitt and T. Jin. A workload characterization study of the 1998 world cup web site. IEEE Network, 14(3), 2000.

[25] E. Cecchet, et al., "Performance and Scalability of EJB Applications", The 17th SIGPLAN Conf on Object-oriented programming, systems, languages, and applications, 2002.

[26] Sapuntzakis, et al., Virtual appliances for deploying and maintaining software, in 'Proceedings of the 17th USENIX conference on System administration', 2009.

[27] K.Keahey & T.Freeman, "Contextualization: Providing one-click virtual clusters", in 'Fourth IEEE International Conference on eScience', Indiana, USA, 2008.

[28] P.Wang, "Qos-aware web services selection with intuitionistic fuzzy set under consumer's vague perception", Expert Systems with Applica- tions 36(3), 4460-4466, 2009.

[29] Wang, et al., "An autonomic provisioning framework for outsourcing data center based on virtual appliances", Cluster Computing 11(3), 229-245, 2008.

[30] Konstantinou et al., "An architecture for virtual solution composition and deployment in infrastructure clouds", in Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing, 2009.

[31] Chieu, et al., "Solution-based deployment of complex application services on a Cloud", in 2010 IEEE International Conference on Service Operations and Logistics and Informatics (SOLI), 2010.

[32] X. Su, J. Rao, "A Survey of Automated Web Service Composition Methods", in Proceedings of First International Workshop on Semantic Web Services and Web Process Composition, 2004.

[33] DMTF (n.d.), `Open virtualization format', http:// www.dmtf.org/standards/ovf.