Efficient and Optimal Service Component Distribution across Multiple Clouds

https://doi.org/10.3991/ijes.v6i3.9567

Driss Riane and Ahmed Ettalbi(✉)
Mohammed V University, Rabat, Morocco
riane.driss@gmail.com

Abstract—Cloud computing technology is one of the key considerations for business willing to access to different cloud services over the Internet and to benefit from the diversity of IaaS offers and pricing models. Although several solutions are available in the market, there are still some issues to solve. The main important aspect to address is the user's request complexity, the vendor lock-in risk and the SLA fulfillment. In this paper, we propose a Multi-Cloud Broker called MCB that allows an efficient and optimal service component distribution among different clouds in flexible and dynamic infrastructure provisioning environment, in order to achieve better Quality of Service and cost efficiency. The request partitioning is the main step of our approach, this step is performed using Gomory-Hu tree based algorithm. Our simulation results show how our algorithm is better than existing partitioning algorithms in terms of running time.

Keywords—cloud computing, infrastructure-as-a-service, multi-cloud, resource-provisioning, quality-of-service, service deployment

1 Introduction

Cloud computing is a new technology that has facilitated outsourcing enterprises' IT infrastructures on an economic basis. It enables dynamic provisioning of virtual machines and scalable applications to suit their needs thanks to a "pay as you go" pricing model. The lack of common cloud standards and vendor lock-in issues hinder the interoperability across cloud providers. So, in the cloud customer's perspective, selecting the appropriate cloud offers that fits his needs is a difficult task. Thus, several promising approaches for interoperating clouds are present in the literature [4, 2].

A natural evolution in Cloud computing is happening by renting heterogeneous cloud resources and using different services from multiple clouds in order to have a wider range of choices with various cost and quality of services. Improving the QoS, while optimizing the global infrastructure cost; the ability to migrate among several providers; avoiding vendor lock-in; and the need of specific Cloud services which are not provided elsewhere are some of the reasons for using services from multiple clouds. In Multi-Cloud model [3], as the focus of our work, the broker is responsible

to deal with provider API variations and guarantee a layer of indirection, interposed between the customers and the IaaS providers.

As presented in our previous work [1], the request can be considered as an abstract Composite Infrastructure Service characterized by a set of functional and non-functional attributes for each included component. The service deployment on different clouds paves the way for the following advantage:

- Availability and disaster recovery: Failure and low QoS can be compensated quickly with minimal error.
- Geographical coverage: Users from different locations access to service with high QoS.
- No vendor lock-in: Virtual machines can be migrated easily between cloud providers in case of any QoS violation.
- Cost reduction: Different pricing model of the providers can be exploited to reduce infrastructure cost.

In this work, we propose a Broker-Based system to help IaaS users (or service providers) to deploy theirs services in a suitable strategy across multiple clouds automatically. Firstly, the user request is modeled as a graph which contains the detailed requirements about the service to deploy, such as the requested resources for each component of the service, optimization criteria, user objectives and constraints. Secondly, a splitting Algorithm is used to solve the request partitioning problem. This algorithm is based on Gomory-Hu graph transformation [5]. Lastly, cost-aware provider selection is proposed to efficiently map each partition to the most suitable cloud platform while reducing the cost for customers.

The remainder of this paper is organized as follows. In the next Section, we present the problem statement. Section 3 introduces relevant related work in this topic. In the Section 4, we present and detail our proposed multi-cloud broker (MCB) that helps IaaS users to deploy complex services in multi-cloud environment. Section 5 presents the performance evaluation of our partitioning algorithm. Finally, Section 6 concludes the paper and presents an overview about our future work.

2 Problem Statement

Nowadays, the service request can be expressed as a complex topology with nodes and links constraints. Partitioning request over multiple cloud platforms while optimizing the overall service cost is a complex task. In addition, manual service deployment onto the cloud can be complex and fault-prone. Therefore, a third party that acts between cloud providers and customers is required in order to negotiate and allocate resources among multiple clouds.

To study customer requirements and concerns for deploying a composite infrastructure service in multi-cloud environment, we consider a set of virtual appliances modeled as an undirected graph which have communication among them. The good example of real world is the cloud infrastructure services (virtual machines, applianc-

es …) rented from multiple cloud providers for deploying multi-tier applications supporting web-based services (S1, S2, S3and S4) as described in Figure 1.

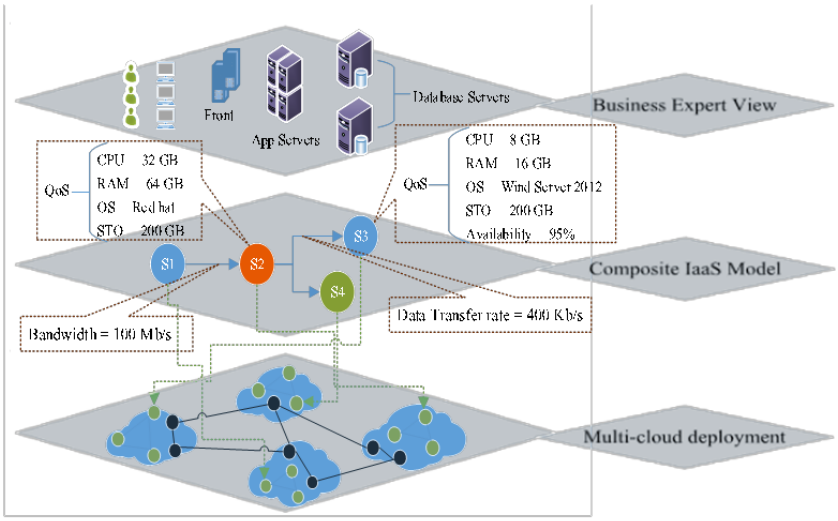

Fig. 1. Motivation scenario

The business expert of the multi-tier application might be interested in cloud deployment in order to minimize infrastructure and maintenance costs, as well as to gain the advantage of on-demand scaling. He prefers multiple clouds deployment to allow disaster recovery and offer service in different geographic zones.

The challenges of such deployment are (i) how to select the best reliable cloud and allocate resource for each component?. (ii) What is the best strategy to decrease latency and data transfer cost between selected clouds?; (iii) What about the total Cloud deployment cost of the composite infrastructure service?.

In this context, the objective of our paper is to formulate our problem as and propose a broker-based system that assists IaaS users in selecting the appropriate cloud platform that best suits their requirements and needs. The broker has to optimally split the request into sub-requests assigned to each suitable cloud provider, offer a service that ensures optimal resource provisioning, SLA enforcement and automatic deployment.

3 Related Work

Due to the growth of cloud computing, there has been a significant amount of research on IaaS clouds, resource provisioning from multiple clouds, cloud brokering and techniques for virtual resource mapping.

The authors in [6, 7] have studied the problem of resource provisioning for a specific type of application across multiple infrastructure providers in cloud or cloud-based data centers. Multiple researches propose algorithms and techniques for cost optimization. As in [8, 9], some provisioning algorithms aim at minimizing the workflow execution cost while meeting the QoS constraints in cloud and grid computing. Reference [10] proposes a new qualitative economic model based optimization approach to compose an optimal set of infrastructure service requests over a long-term period.

Multiple studies deal with the complexity of cloud service selection that while minimizing deployment cost. In [11], a 0-1 integer programming based algorithm is proposed to select the optimal cloud in which a latency sensitive service can be deployed. The programming is subject to multiple criteria, such as resource capacity, load balance and latency request. The authors in [12] discuss how to disperse the different components of the application among multiple clouds and map them to rented cloud resources while minimize the total cost and maintaining an acceptable level of performance. To address multiple objectives (e.g., cost minimization and performance optimization), multi-objective programming based solution is proposed in [13, 14]. The solution scores all kinds of constraints for each cloud service providers, especially on the technology heterogeneity, and then chooses the provider with the maximum score.

To tackle the problem of virtual resource mapping, graph theory based approaches have been proposed [15-17, 21, 28, 29]. The requested virtual machines communicated with each other are viewed as an undirected weighted graph. The infrastructure is viewed as another weighted undirected graph. When communication between VMs and bandwidth are taken into account, it must resort to the networked model, such as graph theory (including graph partitioning, sub-graph matching and the shortest path tree etc…) and virtual network embedding.

Several works [18, 3] have addressed cloud brokering service where the broker acts as a third party between the user and the cloud providers to simplify the service selection and integration from many cloud platforms according to their demand requirements and objectives. The authors in [19] address the problem of multi-cloud resource management that is an optimization problem aimed at reducing the monetary cost and the execution time of consumer applications using Infrastructure as a Service of multiple cloud providers.

Reference [20] tackles the issue of finding an appropriate combination of cloud resources from different cloud providers that satisfies application requirements on the one hand and takes into account general consumer requirements such as budget and time limitation constraints, on the other hand.

However, researches aforementioned are used to address different kinds of problems with different objectives. The majority of these works only consider a single cloud provider. This may not be the most realistic scenario nowadays, since many user requests are complex and need to be provisioned across different infrastructures belonging to multiple providers to deploy and deliver services end to end.

4 The Proposed Approach

In this section, we discuss some challenging aspects of the multi-cloud deployment problem. Specifically, we investigate the question of cloud request partitioning and selecting the most Infrastructure providers for component services hosting and the decision on deployment taking into account user's requirements and QoS constraints.

4.1 Multi-Cloud Broker architecture

Figure 2 presents the architecture of the proposed multi-cloud broker called MCB and depicts a scenario where MCB deals with three cloud providers to offer cloud services.

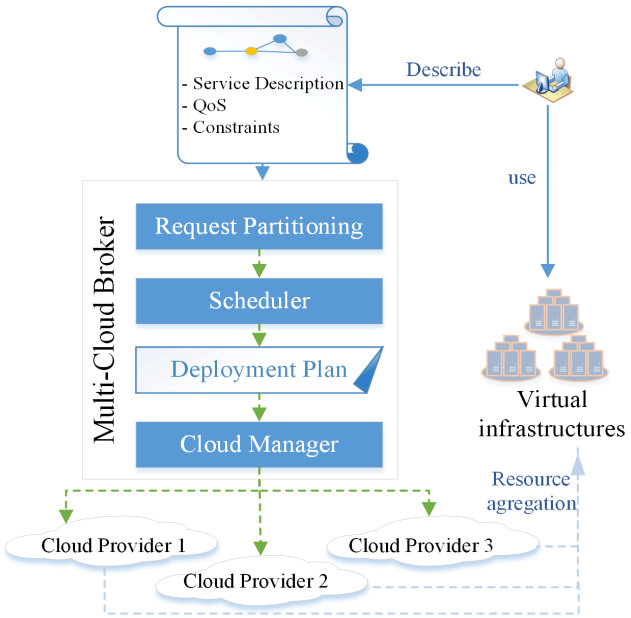

Fig. 2. The MCB Architecture Overview.

The goal is to deploy a complex service composed of a set of components that are executed as interconnected virtual machines in different cloud platforms.

The key process of our system is detailed in the following steps:

Step 1: The user formulates and sends the request to the MCB.

Step 2: The MCB decompose the request into sub-request and try to minimize the number of partitions that can be mapped to at least one cloud provider meeting the user requirements. In this step, the MCB uses the clustering algorithm based on Gomory-Hu transformation [5] to split the graph request. This technique will be detailed in the following.

Step 3: Once the partitions are defined, the MCB allocates each partition from the most suitable cloud using the cost-aware provider selection procedure that will be detailed after. Based on the sub-request requirement, it generates actions and a list of VM templates to be sent to each cloud provider in order to host the service in its own datacenter.

Step 4: The user can access to generic and uniform user interface to manage, deploy and monitor the composite service.

4.2 User request model

As presented in our previous work [1], the service request is modeled as weighted undirected graph GR (VR, ER), where VR is the set of requested nodes, ER is a set of required links. As shown in Figure 3, each node S_i ; $i = \{1,..,4\}$ is associated with a set of resources such as CPU, memory and storage and each link S_{ij} ; $i,j = \{1,..,4\}$ is specified by bandwidth and other parameters (e.g., data transfer rate, delay ...).

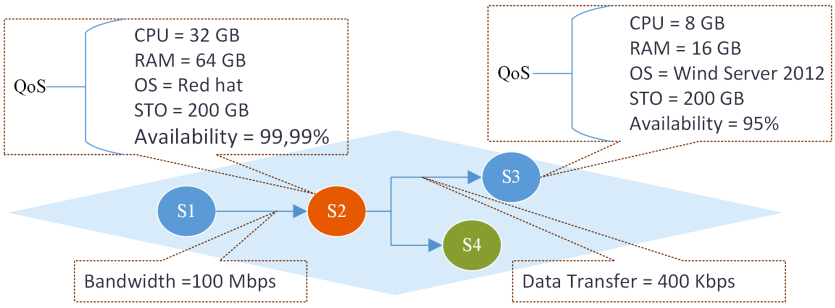

Fig. 3. User Service Request Model.

The request is a file described by the user based on standard language (e.g. XML or JSON). This file contains the detailed requirements about the service to deploy, such as the requested resources for each component of the service, optimization criteria, user objectives and constraints.

4.3 Virtual cloud resources model

Our virtual resource model is based on the model proposed by Amazon EC2 [25]. It classifies the resources (i.e., virtual machines) into certain types, each type having a set of attributes (e.g., virtual CPU, storage, memory, operating system, I/O performance). Each resource type is announced with the associated cost. Table 1 shows the data structure of our virtual resource model.

Table 1. Data structure of virtual cloud resource model

Element	Attributes
InP (IaaS Provider)	Id Name
Resource Type	Id Attributes (CPU, RAM, STO) Location Resource Type cost (per compute unit) Available (true/false)
Peering Nodes	Id Accessible peering node id Inbound traffic cost Outbound traffic cost

4.4 Multi-cloud model

A Multi-cloud strategy is to use multiple cloud service providers to create a solution that is tailored to business needs. Such strategy allows companies to avoid being restricted to a particular vendor and offers more flexibility.

In multi-cloud environment, clouds communicate via a substrate network, a cloud broker is responsible to negotiate with cloud providers on behalf of the user, may also act as a cloud aggregator and provide a unified interface to nonstandard vendor APIs [26, 27]. The broker guarantees that all participating cloud providers disclose their resources information and a set of peering nodes that simplifies the inter-cloud communication.

We model this substrate network as a weighted undirected graph GS (VS, ES). Each substrate node u∈ VS represents a cloud resource (VM, server, router, etc...) and characterized by a set of functional attributes including its type (e.g., node type, OS, virtual environment, etc.) and a set of non-functional attributes (CPU capacity, memory capacity, storage capacity, etc…). Each substrate link (u, v) ∈ ES is characterized by a set of attributes especially bandwidth capacity.

Figure 4 illustrates the view of the broker on the multi-cloud topology where {U, V… Y} is the set of peering nodes and {a, b… g} is the set of offered resources type.

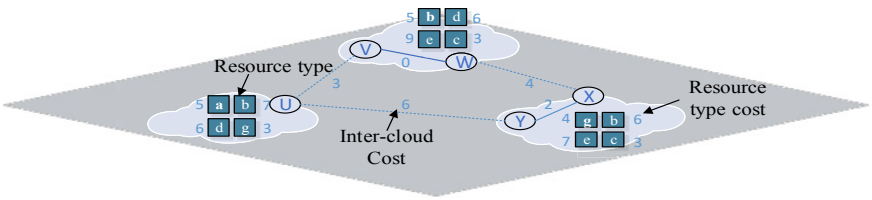

Fig. 4. The Broker's View on the Multi-Cloud Topology.

4.5 Request partitioning

For sake of simplicity, the MCB try to split the request graph GR (VR, ER) into a set of partitions that can be mapped to at least one provider satisfying the QoS constraints. In this section, we propose a new approach based on the clustering of the query graphs.

Gomory-Hu tree graph construction: Gomory-Hu Tree of a graph G is a succinct representation of the edge connectivity between all pairs of its nodes. The tree has the same set of vertices as input graph and has N-1 (N is number of nodes) edges. The Edge capacity function is defined using the minimum cuts [21] between all pairs of nodes of the original graph.

We describe in this section a formal code of the classical Gomory-Hu algorithm [4], used to get the tree representation of the requests. We apply the Gomory-Hu transformation to the request graph GR (VR, ER) in order to get the succinct representation called TGR (TVR, TER). Figure 5 illustrates an example of Gomory-Hu tree construction.

Algorithm 1: Gomory-Hu Tree Construction Algorithm

Input: GR (VR, ER); /* GR is the request graph */;
Output: TGR (TVR, TER); /* TGR is the obtained tree*/;
Init: TVR = VR, TER = Ø, TEMP = VR ;
while (TEMP ≠ ∅) do
$S \leftarrow FirstElement\ (TEMP)$;
$\{S_1, S_2\} \leftarrow minCut\ (S, TG^R)$;
TVR = $\{ TV^R \setminus S \} \cup \{S_1, S_2\}$;
TER = $TE^R \cup (S_1, S_2)$;
If $|S_1| > 1$ then
TEMP = $TEMP \cup \{S_1\}$;
end if
If $|S_2| > 1$ then
TEMP = $TEMP \cup \{S_2\}$;
end if
end while
return TGR

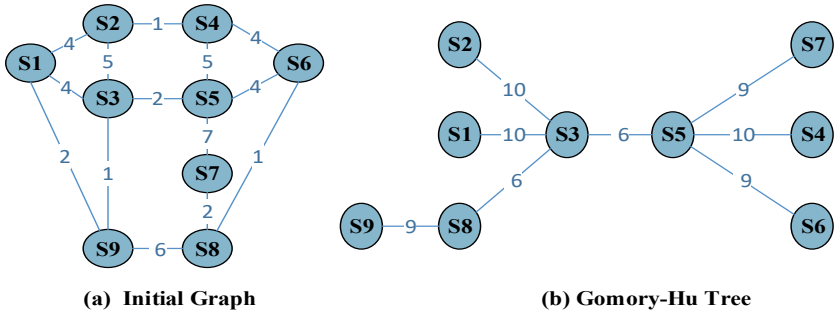

Fig. 5. Gomory-Hu Transformation Example.

Gomory-Hu tree based request partitioning. Our problem is a variant of minimum k-cut of a graph partitioning [21]. A k-cut is a set of edges whose lifting would partition the graph into k connected components. The minimum k-cut finds the cut set with the minimum total weight (i.e., in our case the sum of bandwidth on the edges). The problem of minimum k-cut is N-Complete for non-fixed k. Thus, in our approach we define k according to the number of locations specified in the graph request.

Given a graph request GR (VR, ER) and for a fixed k, we can obtain k partitions by removing (k-1) edges in the resulted Gomory–Hu tree graph TGR (TVR, TER) with the lowest bandwidth. This technique allows, in one hand, to group the nodes with high interactions in the same partition and to minimize the interactions in term of traffic exchanges between partitions in other hand.

Figure 6 illustrates an example of request partitioning into tree partitions based on the proposed approach.

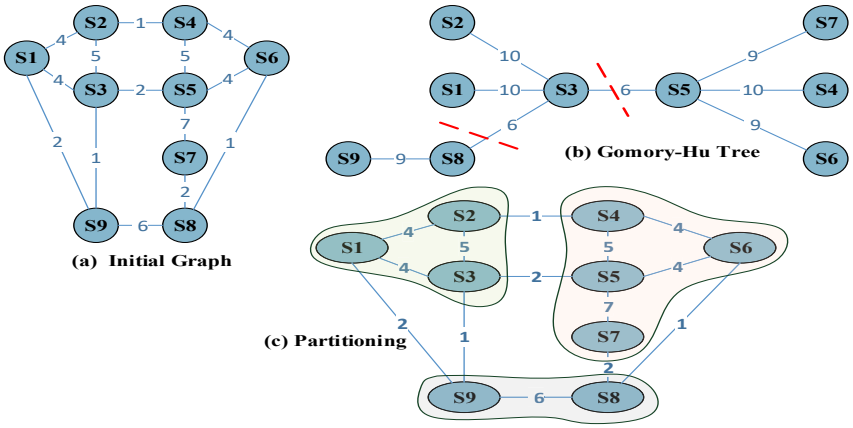

Fig. 6. Gomory-Hu based Graph Partitioning Example.

To obtain the tree partitions, we remove two edges with the lowest bandwidth (i.e., (S3, S8) and (S3, S5) which has $\min(\{10, 9, 6\}) = 6$) from the resulted tree graph.

4.6 Scheduler

Once the request partitioning is done, the scheduler should select for each partition the most suitable provider in order to allocate the required resources. This step is conducted using cost-aware provider selection procedure that aims to minimize the total resource allocation and networking costs. Before detailing this assignment program, we introduce some notations and definitions used in the remainder of this procedure.

Definition 1: Given a graph GR (VR, ER) of the request r, a graph partition P_i identified by $(id = i)$ is a set of l connected nodes (i.e., a group of connected VMs). For each request r, we denote the set of all partitions obtained in the k-cut request partitioning $P_r^k = \{P_1, \dots, P_k\}$.

Where: $\begin{cases} P_i \cap P_j = \emptyset, \forall\, i \neq j \\ \bigcup P_i = V^R,\ i = \{1, \dots, k\} \end{cases}$

Definition 2: The link between two nodes l and l' in the same partition P_i is denoted by $e^i_{(l,\,l')}$ and characterized by a bandwidth $bw_{(l,\,l')}$. We define the set of this links as $L^i = \{\, e^i_{(l,\,l')}\ ;\ l,\ l' \in V^R\}$.

Definition 3: We define the set of links between two partitions P_i and P_j as $e^r_{(P_i,\,P_j)}$. Each link in this set is characterized by a bandwidth $bw_{(P_i,\,P_j)}$. This set is the (k-1) removed edges in the Gomory-Hu tree in order to split the request.

Cost function. Handling complex service requires more attention to the computing and networking requirements, especially the cost metric, to make the decision about the resource allocation for each partition. Equation (1) is the cost $C^{Total}_{c,\,P_i}$ of a given partition P_i when assigned to provider c. It is the sum of computing cost $C^{Computing}_{c,\,P_i}$, defined in (2), needed to serve all partition's nodes during a period T and the networking cost, formulated in (3), $C^{Networking}_{c,\,P_i}$ inside and its connection with other partitions.

$$\mathrm{C}^{\mathrm{Total}}_{\mathrm{c},\,\mathrm{P_i}} = \mathrm{C}^{\mathrm{Computing}}_{\mathrm{c},\,\mathrm{P_i}} + \mathrm{C}^{\mathrm{Networking}}_{\mathrm{c},\,\mathrm{P_i}} \tag{1}$$

$$\mathrm{C}^{\mathrm{Computing}}_{\mathrm{c},\,\mathrm{P_i}} = \sum_{\mathrm{l} \in \mathrm{P_i}} \mathrm{cost(l)}.\mathrm{T} \tag{2}$$

Where $cost(l)$ is the cost of VM instance type l.

Equation (3) is the networking cost for serving partition P_i among provider c. It includes the traffic cost between partition's nodes $C^{in}_{c,\,P_i}$, as in (4), and the communication cost with others partitions $C^{out}_{c,\,P_i}$ as formulated in (5).

$$\mathrm{C}^{\mathrm{Networking}}_{\mathrm{c},\,\mathrm{P_i}} = \mathrm{C}^{\mathrm{in}}_{\mathrm{c},\,\mathrm{P_i}} + \mathrm{C}^{\mathrm{out}}_{\mathrm{c},\,\mathrm{P_i}} \tag{3}$$

$$\mathrm{C}^{\mathrm{in}}_{\mathrm{c},\,\mathrm{P_i}} = \sum_{\mathrm{l,l'} \in \mathrm{P_i}} \mathrm{bw}_{(\mathrm{l,l'})} \cdot \mathrm{C}^{\mathrm{Net}}_{\mathrm{c},(\mathrm{l,l'})} \tag{4}$$

Where $C^{Net}_{c,(l,l')}$ is the cost of connecting two nodes l and l' in the same partition among the provider c.

$$C^{out}_{c,P_i} = \sum_{j=1}^{s} bw_{(P_i,P_j)} \cdot C^{Net}_{c,c'} \quad (5)$$

Where s the number of partitions that are connected to P_i and $C^{Net}_{c,c'}$ is the inter-cloud cost (i.e., a traffic cost between clouds c and c') when partitions P_i and P_j are assigned to providers c and c' respectively.

Cost-aware provider selection. As shown in Figure 7, the scheduler defines for each provider the cost matrix that gives the cost of each partition. To optimize the resource allocation cost, we use cost-aware provider selection procedure detailed in the Algorithm 2 that handles partition mapping based on the cost C^{Total}_{c,P_i}.

Our selection procedure aims to estimate for each partition the cost C^{Total}_{c,P_i} using equation (1) among all the cloud providers to choose the suitable one with the lowest cost.

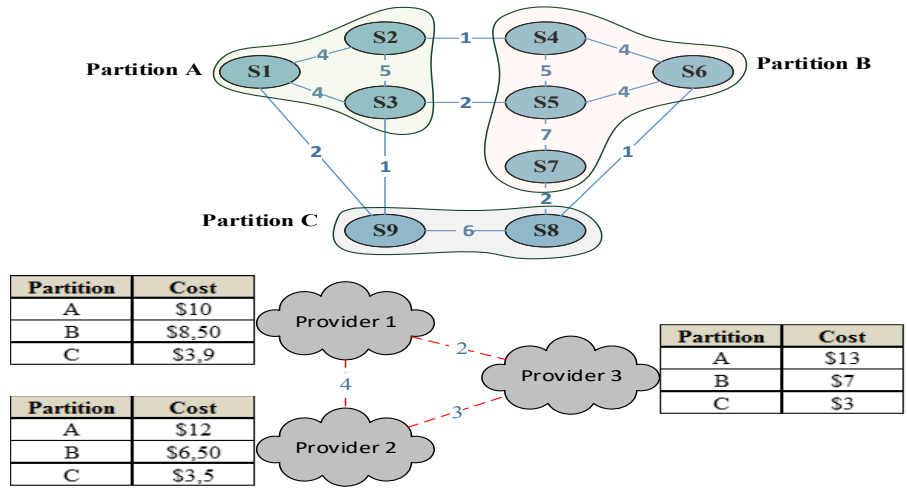

Fig. 7. Cost-Aware Provider Selection Example.

The selection procedure handles the partitions and their total required resources. Based on the equation (1), it calculates for each partition the costs for resource allocation among all cloud providers to select the suitable one (which have the lowest cost). The procedure continues until all partitions are mapped to selected providers, and returns the mapping matrix.

As depicted in Figure 7, by executing Algorithm 2 to the tree partitions {A, B, C}. The partition A has cost_Matrix[A] = [$10, $12, $13] and from where is mapped to provider 1. The partition B has cost_Matrix[B] = [$8.50, $6.50, $7] so is mapped to provider 2. The partition C has cost_Matrix[C] = [$3.9, $5, $3] so is mapped to pro-

vider 3. So the resulted mapping matrix is mapping_plan = {provider1, provider 2, provider 3}.

Algorithm 2: Cost-Aware Provider Selection Algorithm

```
Input: P_r^k = {P_1, ... , P_k} , Cp = {c_1, ... , c_m} ;
Output: mapping_plan[i];
Initialize: satisfied ← 0; mapping_plan[i] ← null ;
      costMat[P_i, c_j] ← null;
For (P_i ∈ P_r^k ) do
For (c_j ∈ Cp ) do
costMat[P_i, c_j] ← calcul_cost (using equation 1);
End for
Sort the list of providers by their costs and capacities;
For (c_j ∈ Cp ) do
If ( c_j can host P_i ) then
mapping_plan[i] ← c_j ;
satisfied ← satisfied + 1 ;
break;
End if
End for
End for
If (satisfied ≠ k) then
mapping_plan[i] ← null ;
end if
Return mapping_plan;
```

4.7 Cloud manager

The cloud manager addresses the management and monitoring actions and provides a uniform and generic user interface to manage services that are executed as virtual machines in heterogeneous distributed data center infrastructures. To provide a multi-cloud interoperability, cloud manager uses specific adapters and interfaces to communicate with other cloud providers. OpenNebula [22] is an example such cloud manager which allows the management of heterogeneous infrastructures and provides API to interoperate with others clouds such as ElasticHosts [23], Amazon EC2 [24].

5 Simulation Results

We present in this Section the results of the performance evaluation of our partitioning approach. The simulation was performed using python based simulator and NetworkX library [30]. Our results are compared to those obtained using Mixed Integer Program (MIP) proposed in [28] and Iterated Local Search (ILS) approach [29].

To evaluate the performance of our approach, we first define the simulation parameters as follows: Each graph request is generated with a random number of connected nodes ([50, 300]). Each link is characterized by a weight representing the resource and networking requirements. This weight is uniform and distributed in [1, 9]. The goal of our simulation is to partition the graph request into k = 4 and k=9 partitions. We run the simulation 40 times and use the average running time metric to compare results.

Figures 8 shows the average running time for k= 4 and k= 9 against the number of nodes. We observe that our approach is better than others algorithm in terms of average running time.

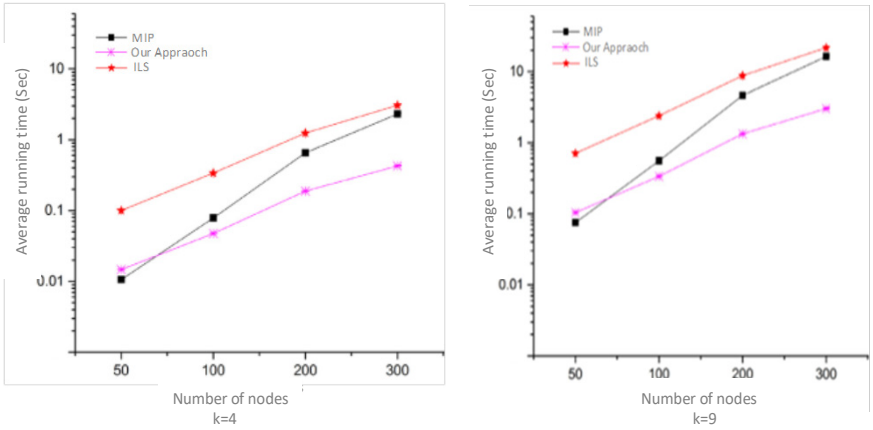

Fig. 8. Average running time for k=4 and k= 9.

6 Conclusion

The diversity of services in a multiple Clouds environment is encouraging more SaaS providers to move towards using the infrastructure services provided by the Cloud providers instead of running their own data centers. However, the lack of an efficient service that maps requested resources, required for composite infrastructure Service Deployment, to offered resources and SLA management approach that minimize the overall deployment cost under QoS constraints impedes this evolutionary process. To tackle these barriers, in this paper, we proposed a Multi-Cloud Broker that allows the deployment of composite infrastructure service across multiple clouds.

The main contribution of our work is the Multi-Cloud Broker architecture which illustrates the key process of our approach. Firstly, we have modeled the user request and virtual cloud resources as a graph, Secondly, we have implemented the request partitioning algorithm implemented based on Gomory-Hu graph transformation. Thirdly, we have formulated the cost function which is used in the proposed cost-aware provider selection algorithm. Lastly, we have conducted experimentation to evaluate the performance of our partitioning approach, the results we have obtained shows how our approach is better than others.

This work is a preliminary schema of the proposed approach, so as future work, there are still many challenges which are needed to be covered during the complete definition and implementation of our Multi-Cloud Broker framework.

7 References

[1] Riane, D. & Ettalbi, A. (2018). A graph-based approach for composite infrastructure service deployment in multi-cloud environment. International Conference on Advanced Communication Technologies and Networking (CommNet 2018) , April 2-4 2018, Marrakech, Morocco, pp. 1-7 https://doi.org/10.1109/COMMNET.2018.8360254

[2] Sotiriadis, S., Bessis, N., Kuonen, P. & Antonopoulos, N. (2013). The inter-cloud meta-scheduling (ICMS) framework. 27th IEEE International Conference on Advanced Information Networking and Applications (AINA), March 25-28 2013, Barcelona, Spain, pp. 64–73 https://doi.org/10.1109/AINA.2013.122

[3] Petcu, D. (2013). Multi-Cloud: expectations & current approaches. ACM The international workshop on Multi-cloud applications and federated clouds, April 22 2013, Prague, Czech Republic, pp. 1–6 https://doi.org/10.1145/2462326.2462328

[4] Cenk Erdil, D. (2013). Autonomic cloud resource sharing for intercloud federations. Future Generation Computer Systems, 29 : 1700–1708 https://doi.org/10.1016/j.future.2012.03.025

[5] Gomory, R.E. & Hu, T.C. (1961). Multi-terminal network flows. Journal of the Society for Industrial and Applied Mathematics, 9: 551–570 https://doi.org/10.1137/0109047

[6] Wei, X., Li, H., Yang, K., & Zou, L. (2014). Topology-aware partial virtual cluster mapping algorithm on shared distributed infrastructures. IEEE Transactions on Parallel and Distributed Systems, 25: 2721–2730 https://doi.org/10.1109/TPDS.2013.224

[7] Toosi, A. N., Sinnott, R. O., & Buyya, R. (2017). Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka. Future Generation Computer Systems, 79: 765-775 https://doi.org/10.1016/j.future.2017.05.042

[8] Leite, A. F., Alves, V., Rodrigues, G. N., Tadonki, C., Eisenbeis, C. & Melo, A. C. M. A. d. (2015). Automating Resource Selection and Configuration in Inter-clouds through a Software Product Line Method. 8th International Conference on Cloud Computing, New York City, pp. 726-733 https://doi.org/10.1109/CLOUD.2015.101

[9] Visheratin, A. A., Melnik, M., & Nasonov, D. (2016). Workflow Scheduling Algorithms for Hard-deadline Constrained Cloud Environments. In Procedia Computer Science, 80: 2098-2106 https://doi.org/10.1016/j.procs.2016.05.529

[10] Alkhanak, E. N., Lee, S. P., Rezaei, R., & Parizi, R. M. (2016). Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. In Journal of Systems and Software, 113: 1-26 https://doi.org/10.1016/j.jss.2015.11.023

[11] Mistry, S., Bouguettay, A., Dong, H., & Erradi, A. (2016). Qualitative economic model for long-term IaaS composition. 14th International Conference on Service Oriented Computing (ICSOC), October 10-13 2016, Banff, Canada, pp. 317-332 https://doi.org/10.1007/978-3-319-46295-0_20

[12] Diaz-Sanchez, F., Al Zahr, S., & Gagnaire, M. (2013). An exact placement approach for optimizing cost and recovery time under faulty multi-cloud environments. 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013), December 2-5 2013, Bristol, UK, pp. 138–143. https://doi.org/10.1109/CloudCom.2013.116

[13] Dubois, D. J., & Casale, G. (2016). OptiSpot: minimizing application deployment cost using spot cloud resources. Cluster Computing, 19: 893–909 https://doi.org/10.1007/s10586-016-0568-7
[14] Weintraub, E. & Cohen, Y. (2015). Cost Optimization of Cloud Computing Services in a Networked Environment. International Journal of Advanced Computer Science and Applications, 4: 148-157 https://doi.org/10.14569/IJACSA.2015.060420
[15] Weintraub, E. & Cohen, Y. (2017). Multi Objective Optimization of Cloud Computing Services for Consumers. International Journal of Advanced Computer Science and Applications, 8: 139-147 https://doi.org/10.14569/IJACSA.2017.080219
[16] Verbelen, T., Stevens, T., De Turck, F., & Dhoedt, B. (2013). Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. Future Generation Computer Systems, 29 : 451–459 https://doi.org/10.1016/j.future.2012.07.003
[17] Aral, A., & Ovatman, T. (2015). Subgraph matching for resource allocation in the federated cloud environment. 8th IEEE International Conference on Cloud Computing (CLOUD 2015), 27 June-2 July 2015, New York, USA pp. 1033–1036 https://doi.org/10.1109/CLOUD.2015.145
[18] Grozev, N., & Buyya, R. (2014). Inter-cloud architectures and application brokering: Taxonomy and survey. Software: Practice and Experience, 44(3), 369–390. https://doi.org/10.1002/spe.2168
[19] Heilig, L., Lalla-Ruiz, E., & Voß, S. (2016). A cloud brokerage approach for solving the resource management problem in multi-cloud environments. Computers & Industrial Engineering, 95:16-26 https://doi.org/10.1016/j.cie.2016.02.015
[20] Coutinho, R. d. C., Drummond, L. M., Frota, Y., & de Oliveira, D. (2015). Optimizing virtual machine allocation for parallel scientific workflows in federated clouds. Future Generation Computer Systems, 46: 51–68 https://doi.org/10.1016/j.future.2014.10.009
[21] Goldschmidt, O. & Hochbaum, D.S. (1988). Polynomial algorithm for the k-cut problem. 29th Annual Symposium on Foundations of Computer Science, October 1988, pp. 444-451
[22] Moreno-Vozmediano, Rafael & Montero, Rubén & M. Llorente, Ignacio. (2012). IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures. Computer, 45: 65-72 https://doi.org/10.1109/MC.2012.76
[23] Elastic Hosts home page, http://www.elastichosts.com/
[24] Amazon Elastic Compute Cloud (EC2) home page, http://aws.amazon.com/ec2/
[25] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/
[26] Barker, A., Varghese, B., & Thai, L. (2015). Cloud services brokerage: a survey and research roadmap. 8th International Conference on Cloud Computing, June 2015, New York City, NY, USA, pp. 1029–1032
[27] Cucinotta, T., Lugones, D., Cherubini, D., & Oberle, K. (2014). Brokering SLAs for end-to-end QoS in cloud computing. 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), April 3-5 2014, Barcelona, Spain, pp. 610–615
[28] Houidi, I., Mechtri, M., Louati, W. & Zeghlache, D. (2011). Cloud Service Delivery across Multiple Cloud Platforms. 2011 IEEE International Conference on Services Computing, July 4-9 2011,Washington, DC, USA, pp. 741-742https://doi.org/10.1109/SCC.2011.107
[29] Leivadeas, A., Papagianni, C., Papavassiliou, S. (2013). Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning. IEEE Transactions on Parallel and Distributed Systems, 24:1077–1086https://doi.org/10.1109/TPDS.2012.204
[30] https://networkx.github.io/documentation/networkx-2.1/reference/index.html

8 Authors

Driss Riane is PhD student at IMS Team, ADMIR Laboratory, Higher National School of Computer Science and Systems Analysis (ENSIAS), Rabat IT Center, Morocco.

Ahmed Ettalbi is Professor at Software Engineering Department of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat, Morocco. His main research interests Object Modeling with Viewpoints, Software Oriented, Cloud Computing and Petri Networks.

Article submitted 25 September 2018. Resubmitted 17 October 2018. Final acceptance 20 October 2018. Final version published as submitted by the authors.