Subgraph Matching for Resource Allocation in the Federated Cloud Environment

Atakan Aral, Tolga Ovatman
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
{*aralat, ovatman*}*@itu.edu.tr*

***Abstract*—Federated clouds and cloud brokering allow migration of virtual machines across clouds and even deployment of cooperating VMs in different cloud data centers. In order to fully benefit from these new opportunities, we propose a heuristic that outputs a matching between virtual machine and cloud data centers by taking resource capacities, VM topologies, performance and resource costs into account. Results of our initial evaluation using the CloudSim Framework indicate that, proposed heuristic is promising for a better optimized placement of networked VM groups onto the federated cloud topology.**

***Keywords*-federated clouds; resource allocation; resource mapping; virtual machine placement; subgraph matching;**

I. Introduction

Federated Clouds are quickly gaining popularity in both academia and industry since they provide better interoperability and scalability then divergent cloud data centers [1], [2]. Clouds constituting the federation can be public, private or community owned. In this paradigm, scaling and migrating virtual machines (VMs) across multiple IaaS providers in different geographical locations is possible. Moreover, cloud brokering architecture allows hosting different components of a service as VMs in separate clouds [3].

Abovementioned advancements in cloud systems introduce new challenges and open research directions. In this study, we focus on the problem of choosing cloud data centers from a federated cloud to deploy VMs requested by the user with a specific topology. In our model, a user request is not a single VM or multiple isolated VMs, but it is a group VMs that are interconnected via input/output relations. This model represents a multi-tier architecture. Data centers are similarly interconnected based on a given topology. Matching VMs to data centers is not trivial since there exists multiple criteria to optimize such as computing resources, bandwidth, performance and cost.

We propose a heuristic that implements subgraph matching from graph theory to determine an allocation for each VM in the request. It takes resource capacities, VM topologies, performance (in terms of latency and running time) and resource costs into account during the decision process. The aim is to benefit IaaS provider by serving more users without damaging QoS as well as benefit user by reducing the cost for a certain computation.

II. Related Work

Various optimization and approximation algorithms are suggested for the problem of placing VMs on a substrate network of data centers. These heuristics and algorithms are based on linear programming, artificial intelligence, nature inspired computing and game theory [4]. For brevity, we mention two of the most relevant studies here. A more detailed list of studies is available in [5].

Given an application to run on the cloud, a recent study [6], aims to decide at which data centers to host it and which software component should be hosted at each data center. The objective function minimizes traffic delay, energy consumption, CO_2 emission, bandwidth and server costs, simultaneously. Authors represent both the network and application as graphs and try to find a mapping that achieves the objective. Results demonstrate the efficiency of their tabu search algorithm in comparison to mixed integer programming (MIP). In another study [7], authors aim to optimally assign capacity requests to physical machines (node mapping phase) and connectivity requests to network entities (link mapping phase). The two phases are optimized in a coordinated way by using MIP approximation for the former and either shortest path or minimum cost flow algorithm for the latter. Evaluation metrics used in this study are mapping cost/revenue, acceptance ratio and hop count.

The unique aspect of our approach is the explicit consideration of VM performance as well as utilization of graph theory algorithms. While most studies measure bandwidth usage and latency, they do not demonstrate the effect of them on VM performance in terms of running time. We define input and output relationships among VMs to realistically model the performance. Subgraph matching algorithm, on the other hand, is expected to be more efficient than linear programming and yield better results than approximation algorithms.

III. Problem Modeling

A. Topology Modeling

We model both cloud topology and requested VM topology as weighted, undirected, simple graphs. Vertice represent cloud data centers or requested VMs and have attributes for computing resources (i.e. CPU, memory and storage).

Edges, on the other hand, represent the network connections between them, while their weight represent bandwidth and latency. This representation is convenient for applying graph theory algorithms. Figure 1(a) demonstrates a graph of data centers (white circles) with the network connections between them (black lines). Similarly, another graph of VMs (black circles) and desired connections between them (double lines) are displayed.

B. Broker Modeling

A data center broker is attached to each node in the cloud topology to represent the user base on that location. It is broker's responsibility to receive user VM requests and find good locations to deploy them. If the broker deploys the VMs on the data centers that are distant in terms of latency, deployment delay increases.

C. VM Life-cycle Modeling

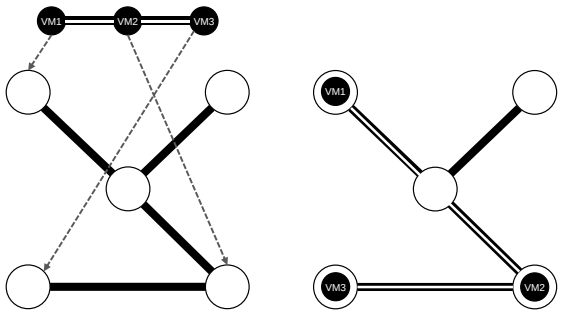
VM Life-cycle starts when a user requests a group of VMs with a certain topology. Then, a broker communicates with known cloud data centers in the federation and submit requested VMs to the data center(s) it selects. An example selection is represented with grey dashed arrows in Figure 1(a). When each VM in the requested group is deployed, they start executing their computing tasks. In addition to task size and VM computation power, execution duration is also affected by the communication delay. This delay is directly proportional to the input/output size of the VM and the latencies to the connected VMs in the group.

D. Bandwidth Modeling

Since we model the bandwidth request between two VMs in edges, bandwidth capacity of both cloud data centers that are allocated to these VMs (i.e. two end-points of the edge) are utilized. More generally, our model utilize the bandwidth capacities of all the nodes that are on the shortest path (in terms of latency) between the two vertice. Only exception to this is the case where both VMs are deployed on the same cloud so there is no need to utilize inter-cloud bandwidth. We do not implement path-splitting at the moment, so the bandwidth request between two VMs is nonbifurcated. Figure 1(b) shows an example bandwidth utilization. Since VM1 and VM2 are not deployed in neighbour locations, some bandwidth from another data center on the shortest path also needs to be utilized.

E. Cost Modeling

Instead of fixed pricing for computing resources, we employ yield management that is studied in [8] as "Trough filling". Our implementation is to simply increase the price of the resources that are running low in a cloud data center. Following the common strategy of IaaS providers, cost of a VM is measured only by its memory and bandwidth consumption.



(a) Matching (b) Allocation

Figure 1. An example topology and bandwidth model.

IV. Topology Based Matching Heuristic

As mentioned earlier, suggested heuristic considers the topology of both cloud federation and VM request to suggest a data center for each VM. It employs an algorithm based on LAD subgraph isomorphism solver [9] and finds a matching between the requested topology and a subgraph of the cloud topology. Main and alternative steps of the heuristics are given below where G_{req} is the graph representing the desired topology and G_{cloud} is the graph representing cloud federation topology.

1) All subgraphs of G_{cloud} that are isomorphic to G_{req} are identified.
    a) If none exist, VMs are processed individually.
2) For each subgraph, average latency of data centers to the requesting broker node is calculated.
3) G_{req} is matched to the subgraph with the lowest average latency (G_{match}).
4) Each VM in G_{req} is submitted to the matching data center in G_{match}.
    a) Each VM that fails to deploy in the matching data center is processed individually.

In the ideal case, VMs are deployed in the designated locations. However, exceptions in steps 1a) and 4a) require processing VMs individually ignoring the topology. The following steps are executed in such cases.

1) Locations of the already deployed VMs that belong to the same request are identified.
2) Data centers in G_{cloud} are probed in the increasing order of average latency to these locations.
3) If all candidates are already probed, the process is restarted in case a data center is now underloaded.

Main objective of the heuristic is to decrease both deployment delay (by placing VMs close to the broker) and communication delay (by placing connected VMs to the neighbour data centers). In addition, it aims to balance load and reduce resource costs by avoiding placements in a single data center which would overload the data center and increase prices.

V. Evaluation

A. Experimental Setup

1) CloudSim Improvements: All simulations are carried out using the CloudSim Framework [10]. Being the most powerful general-purpose tool for modeling and simulation of cloud computing infrastructures, CloudSim lets us carry out basic simulation tasks such as data center and VM creation, custom cloud topologies, load generation, simple scheduling and logging. However, for the specific needs of the problem at hand, quite a number of additional features are required. New features that are added to CloudSim can be grouped under the following topics.

- VM group support
- Communication and data transfer between VMs in the same group
- VM lifecycle
- Advanced bandwidth allocation
- Graph visualization (for verification purposes)

2) Load Generation: In order to evaluate the suggested approach in a realistic way, a real-world cloud topology is used for our experiments. Topology data is taken from the FEDERICA project [11]. It implements an experimental network infrastructure for trialing new networking technologies. Final version of the infrastructure includes, 14 Point of Presences in 14 National Research and Education Networks in Europe, dedicated 1 Gbps channels among them and up to 4 virtualization servers at each location.

To decide the number of VM request from each location, we followed a similar way to [6] by using the population density around the location. Number of requests is in the range $[2, 32]$ depending on population density. Number of VMs at each request is specified based on a Poisson distribution with $\lambda = 3$. We run our simulation for a 50-hour duration so the arrival times of requests are uniform random in the range $[0, 50)$. Data center memory capacity is $64x$ and available bandwidth is $80y$. Each VM requires memory allocation between $1x$ and $8x$; and also bandwidth allocation to/from other VMs between $1y$ and $8y$.

B. Baseline Methods

Suggested topology based matching heuristic (TBM) is evaluated against the following baseline methods.

1) Arbitrary First Fit (AFF): Default selection algorithm in the CloudSim framework which probes data centers in an arbitrary predefined order and allocates the first available data center.

2) Latency based First Fit (LFF): Same as AFF except that the data centers are probed in an increasing order of latency to the user rather than arbitrary.

3) Load Balancing (LBG): This method aims to assign equal load to all data centers relative to their resource capacities. To that end, a VM request is always assigned to the data center with the lowest resource utilization.

C. Initial Results and Discussion

Simulation is repeated 30 times for each heuristic and configuration in order to collect significant values for the performance criteria. Due to space limitation, results from 8 out of 64 configurations are presented. The results are presented in Figure 2 where horizontal axis is the memory request of a single VM ranging from $1x$ to $8x$. In these configurations, bandwidth request is fixed at $4y$. Vertical axis unit is hours for Figure 2(a-d), millions of instructions for 2(e), percent for 2(f,h) and USD for 2(g).

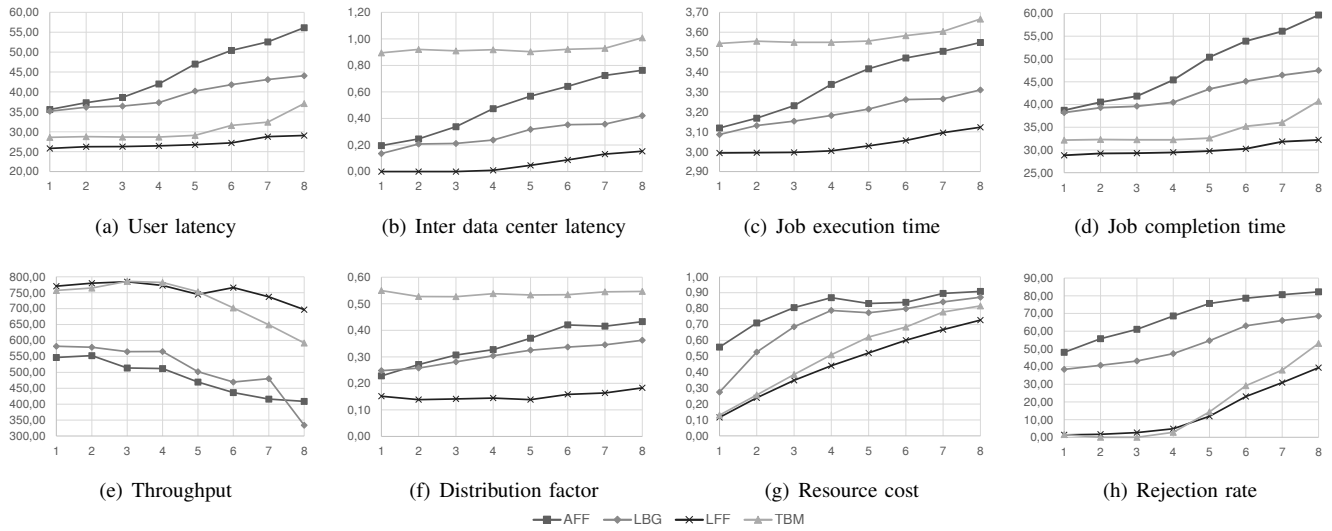
In Figure 2(a), average latency between brokers and their VMs is given. LFF achieves the best user latency since it places all VMs to the closest location as long as enough resources are available. TBM is a close competitor although it does not generally deploy VMs on the same location but distribute them to the neighbourhood. For all heuristics, average latency increases as the VM memory size increases because larger VMs quickly fill up nearby locations. Similarly, average latency between the VMs in the same group is demonstrated in Figure 2(b). LFF, AFF and LBG all allow placement of group VMs in the same location so they have better inter data center latency values compared to TBM. As the VM size increases, it gets harder to deploy all group VMs in the same location.

As explained earlier, execution time of a job (Figure 2(c)) in a VM increases as the latency to the connected VMs increase (communication delay). There is no increase if they are in the same location. Job completion time (Figure 2(d)) involves waiting duration (if a data center is not available immediately) and deployment delay in addition to execution time. Due to the increase in latencies, execution and completion time also increase as VMs grow.

As demonstrated in Figure 2(e), throughput of LFF and TBM are quite close while AFF and LBG suffer significantly. As data center utilization rates increases, TBM has difficulty in finding available subgraphs on the cloud topology. That's why it faces a steeper decrease of throughput.

In Figure 2(f) the results for an evaluation criteria suggested by the authors called distribution factor is presented. This criteria is useful for measuring to what extent VMs of a request are distributed over the cloud. It is the percentage of VMs in the same group but not deployed in the same data center. As expected, TBM has the greatest distribution factor independent of VM size while the other heuristics start with focused placements but forced to scatter as VMs grow.

Since yield management strategy is used for resource pricing, cost of a unit resource is variable. Figure 2(g) clearly present that, AFF and LBG heuristics result in up to 6 times more expensive resources than LFF and TBM, especially for smaller VMs. Finally, Figure 2(h) is for the rejection rate comparison. When a heuristic matches a VM to a location but the data center does not have enough available resources, that VM is rejected and a new matching is required.



(a) User latency (b) Inter data center latency (c) Job execution time (d) Job completion time

(e) Throughput (f) Distribution factor (g) Resource cost (h) Rejection rate

Figure 2. Evaluation results for variable VM memory request.

VI. Conclusion and Future Work

A novel graph based heuristic is proposed for better placement of networked multiple VM requests onto the federated cloud topology at hand. Initial evaluation shows that it performs significantly better than heuristics such as load balancing and arbitrary first fit. However, proposed approach struggles to outperform latency based first fit heuristic and performs roughly the same in terms of throughput and cost.

Since the heuristic is still under development, authors are optimistic about its performance in the future. Some planned improvements for the algorithm are listed below.

- Relaxation of the isomorphism search by allowing homeomorphic subgraphs.
- Allowing deployment in a single location when resources are available.
- Employing different strategies for different VM groups.
- Defining a hyper-heuristic.
- Variable bias towards cost or speed optimization.
- Formal modeling of the problem and scaling support.

References

[1] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 1–11, 2009.

[2] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*. Springer, 2010, pp. 13–31.

[3] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Key challenges in cloud computing: Enabling the future internet of services," *IEEE Internet Computing*, vol. 17, no. 4, pp. 18–25, 2013.

[4] P. Endo, A. de Almeida Palhares, N. Pereira, G. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs, "Resource allocation for distributed cloud: Concepts and research challenges," *IEEE Network*, vol. 25, no. 4, pp. 42–46, 2011.

[5] A. Aral and T. Ovatman, "Improving resource utilization in cloud environments using application placement heuristics," in *Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER)*, 2014, pp. 527–534.

[6] F. Larumbe and B. Sanso, "A tabu search algorithm for the location of data centers and software components in green cloud computing networks," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 22–35, 2013.

[7] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1060–1071, 2013.

[8] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[9] C. Solnon, "Alldifferent-based filtering for subgraph isomorphism," *Artificial Intelligence*, vol. 174, no. 12, pp. 850–864, 2010.

[10] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[11] M. Campanella, "The FEDERICA project: creating cloud infrastructures," in *Proceedings of the First International Conference on Cloud Computing (CLOUDCOMP)*, 2009, pp. 19–21.