Accepted Manuscript

Efficient distribution of requests in federated cloud computing environments utilizing statistical multiplexing

Moslem Habibi, MohammadAmin Fazli, Ali Movaghar

PII: S0167-739X(18)30903-8
DOI: https://doi.org/10.1016/j.future.2018.08.032
Reference: FUTURE 4416

To appear in: *Future Generation Computer Systems*

Received date : 14 April 2018
Revised date : 28 July 2018
Accepted date : 16 August 2018

Please cite this article as: M. Habibi, et al., Efficient distribution of requests in federated cloud computing environments utilizing statistical multiplexing, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.08.032

Efficient Distribution of Requests in Federated Cloud Computing Environments Utilizing Statistical Multiplexing

Moslem Habibi, MohammadAmin Fazli, Ali Movaghar

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract

One of the main questions in cloud computing environments is how to efficiently distribute user requests or Virtual Machines (VMs) based on their resource needs over time. This question is also an important one when dealing with a cloud federation environment where rational cloud service providers are collaborating together by sharing customer requests. By considering intrinsic aspects of the cloud computing model one can propose request distribution methods that play on the strengths of this computing paradigm. In this paper we look at statistical multiplexing and server consolidation as such a strength and examine the use of the coefficient of variation and other related statistical metrics as objective functions which can be used in deciding on the request distribution mechanism. The complexity of using these objective functions is analyzed and heuristic methods which enable efficient request partitioning in a feasible time are presented & compared.

Keywords: Cloud Computing, Cloud Federation, Multiclouds, Request Partitioning, Statistical Multiplexing

Email addresses: moslem_habibi@ce.sharif.edu (Moslem Habibi), fazli@sharif.edu (MohammadAmin Fazli), movaghar@sharif.edu (Ali Movaghar)

1. Introduction

Federated cloud computing environments have recently emerged as a trending topic in cloud computing. Here Cloud Service Providers (CSPs) collaborate by delegating some (or all) of their customers' requests to other CSPs. This is done due to various reasons, be it overloaded servers in the federating CSP (i.e. the CSP that delegates parts of its request load to the federated CSP), the need to adhere to customer Service Level Agreements (SLAs) in special circumstances where the cloud provider cannot guarantee quality attributes, etc. Out of the decisions that must be taken in order to operate in such a federated environment, one of the most crucial is which requests to federate and how should this federation take place keeping in mind the CSPs currently participating in the federation. The answer to this question must be one which is efficient and fair for all participating CSPs and incentivizes them to partake in the federation mechanism. How we model and evaluate this based on various objective functions is an important consideration in this area.

In this paper we emphasize a request distribution mechanism that focuses on multitenancy as a key factor that enables the cloud computing paradigm, providing many of the benefits of this paradigm from a cloud service provider perspective. The federation structure used here can be seen in Figure 1. Customer requests are given to a Federation Broker who distributes them between CSPs which are cooperating in a cloud federation. Such a broker must consider multiple criteria when distribution occurs, including those relating to performance, pricing, quality of service, etc. The criteria (and its related objective functions) which we examine is to ensure request partitioning is efficient with regards to utilizing multitenancy. To this end, different objective functions will be considered, including those that impact statistical multiplexing, as we will show throughout the rest of this section.

1.1. Problem Setting

The resource needs of each customer request submitted to the federation broker is modeled as a random variable X_i with mean μ_i and standard deviation

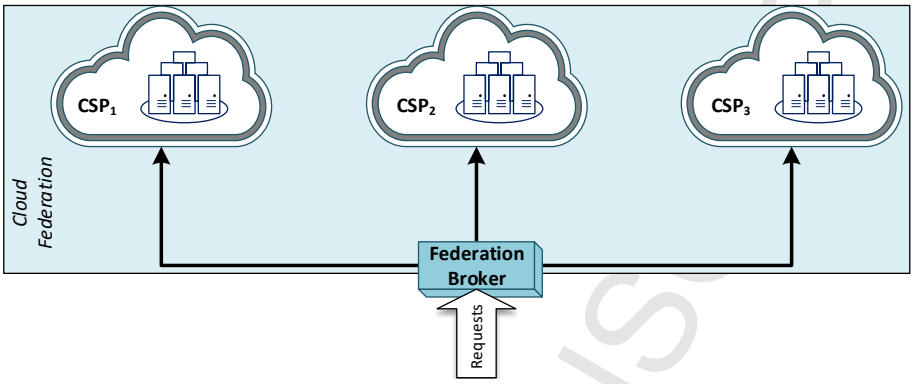
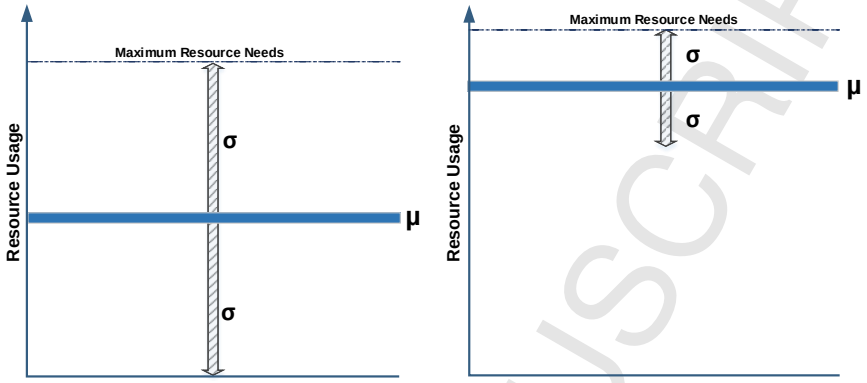


Figure 1: Sample view of a cloud federation setting. In our paper the distribution of customer requests by the federation broker between participating CSPs is done in such a way so as to utilize the multitenancy property of cloud computing. This is done by considering different objective functions, specially those that are related to statistical multiplexing.

σ_i. We must decide on how to partition the complete set of customer requests (random variables $X_1, X_2, ..., X_n$) present in the federation environment between the various CSPs in an efficient way based on various objective functions. One such objective function is the mean (μ) of each subset of requests federated to a single CSP. By ensuring that each of these subsets have comparable means, we can ensure that a larger than normal request load is not suddenly given to a cloud provider, keeping the partitioning mechanism fair. Another important objective function is the standard deviation (σ) of the partitions. If this is too high for a subset of requests federated to a CSP, that CSP will have very contrasting maximum and minimum resource usage, and therefore efficiency decreases as at many times server utilization is low (because of the increased minimum resources required to handle peak usage). This problem becomes more relevant when the mean of a set of requests is small compared to their standard deviation. If a CSP is serving requests that have a large mean value, then small amounts of change in resource usage (based on the standard deviation) is negligible, whilst at the same time requests with a low mean and high standard deviation can incur huge costs on the CSP's data center needs (see Figure 2).



(a) The mean (μ) is small compared to the variance (σ)

(b) The mean (μ) is large compared to the variance (σ)

Figure 2: Here we can see the effect that the mean and variance of resource usage has on CSPs. In (a), the mean of resource usage is small compared to its standard deviation, and therefore allocating the needed hardware to handle peak demand is very costly and wasteful. In (b), the mean is large compared to the standard deviation and therefore only a small amount of over-provisioning is needed, making this scenario more efficient with less resource waste.

Therefore another important objective function here is the ratio of the request partition's standard deviation to its mean, as will be introduced in the next Subsection.

1.2. The Coefficient of Variation Objective Function

One important objective function which we consider here is σ/μ. This ratio corresponds to a measure in statistics called the coefficient of variation (c_v or CoV), which in line with the findings of Weinman [1], can be an important measure of how to efficiently partition requests in a cloud computing environment. To look at this more broadly, using the coefficient of variation (also known as Relative Standard Deviation) we can play on one of the main strengths of the cloud computing paradigm, more specifically economical benefits of server consolidation. Due to economics of scale, CSPs reduce their operating costs by using large data centers and utilizing common infrastructure to consolidate customer requests using VM technology. This is an important method of reducing

the total number of needed hardware, minimizing server sprawl & total data center space requirements and also, by increasing resource utilization, reducing power consumption [2].

An important factor in fully exploiting consolidation is statistical multiplexing among request demand characteristics, i.e. the fact that the peaks and valleys in the resource demands of one customer's requests do not necessarily coincide with other customers [3]. Therefore when one VM is using its peak resource requirements, other VMs on the same hardware may not have much traffic, allowing more efficient resource allocation whilst at the same time adhering to customer SLAs. Here is where the coefficient of variation comes into play as a measure of smoothness or flatness of a set of requests, modeling the efficiency of collocating them on common infrastructure. The correlation between any two of these requests (i.e. two random variables) is the relation between resource usage peaks and valleys when processing the requests together in a specific time frame. When this correlation is high, the two requests need their maximum and minimum resources at around the same time and therefore server consolidation requires more resources. Therefore it is desirable for requests with smaller correlation to be processed on the same hardware.

Taking this into account and using the above mentioned objective functions, we explore how to distribute a set of customer requests between a set of cloud providers in such a way that is efficient, the k-FEOptRP problem. The rest of the paper is organized as follows. We first present some relevant work done in the field of cloud federation and efficient request collocation & distribution. Next our theoretical framework is introduced alongside relevant notations. We then explore the complexity of optimized request partitioning based on the defined objective functions. Heuristics for optimizing the c_v objective function in a federation environment alongside related algorithms are then analyzed and simulation results presented. Finally the paper's findings alongside thoughts on the future direction of work based on these results is showcased.

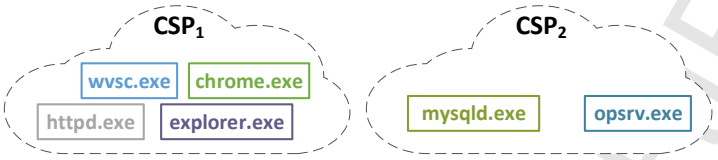
2. Real World Example

Here we provide an example of how the Coefficient of Variation can be used as a metric to decide on process or task partitioning in a real world setting. For this we ran simulations on a server where a real world web application[1] that uses the ubiquitous LAMP[2] stack is run alongside other applications. CPU usage was decided upon as the resource under consideration for this example. Our goal was to have both processes whose CPU usage is dependent on others and processes in which this was not true. By dependent we mean that an increase in CPU use of one such process has a direct correlation with an increase of CPU load in some other processes. This behavior is similar to our context in that we want to partition tasks or processes in clouds in such a way as to have overall minimum CoV for all clouds, i.e. the k-FEOptRP problem, detailed in the next section. The monitored processes were:

- httpd.exe: the Apache2 web server hosting the web application
- mysqld.exe: the MySql database to which the web app connects to
- explorer.exe: the process responsible for the Windows Explorer
- chrome.exe: Google's widely used web browser
- opsrv.exe: a process of the Acunetix[3] web vulnerability scanner
- wvsc.exe: a process of the Acunetix web vulnerability scanner

The CPU usage of the above processes was sampled every 2 seconds, 5000 times. To obtain the CPU usage, the cross-platform psutil[4] module for python was used. Different workloads were run on the above processes during this time in a way that corresponds to their real world usage scenarios. Based on the

[1] The Motoshub social network engine, available at http://shub.ir
[2] Linux-Apache-MySQL-PHP
[3] https://www.acunetix.com
[4] https://pypi.python.org/pypi/psutil



(a) Efficient CoV based partitioning of processes with two CSPs.

(b) Efficient CoV based partitioning of processes with three CSPs.

(c) Efficient CoV based partitioning of processes with four CSPs.

Figure 3: The results of partitioning processes between CSPs in a real world example based on minimizing overall CoV. The results are in line with expectations, as processes whose CPU usage patterns have the most positive correlation are partitioned to different CSPs when possible in each scenario, i.e. (a), (b) and (c).

results obtained from the samples, we can compute the mean and standard deviation for each process and also the covariance between each two processes. The results of the simulation can be seen in Figure 3. Here based on different scenarios where different number of CSPs are available, the efficient distribution of processes in order to minimize CoV was obtained. This was done using a simple brute-force algorithm which checked all partitioning states. As we will later see, running such an algorithm on larger datasets (more CSPs and processes) is infeasible.

The results show that the mysql.exe and httpd.exe processes are partitioned on different CSPs in all scenarios. This is in line with intuition as any heavy workload on the apache server means more database access, therefore they should not be placed together to fully exploit the benefits of statistical mul-

tiplexing. This is also in line with web server and database deployment best practices, where amongst other things performance is one of the reasons that database servers and application/web servers are usually deployed on separate machines. The same is true for the opsrv.exe and wvsc.exe processes of the Acunetix software. There is also a weaker correlation between these two processes and the web application being scanned, and we can see that when there are enough CSPs available, opsrv.exe and wvsc.exe are partitioned independently from mysql.exe and httpd.exe. The CPU usage of the chrome.exe and explorer.exe processes is independent from the other applications sampled and therefore collocation with other processes does not provide a high CoV measurement.

3. Related Work

Research on multicloud environments has increased in recent years with researchers each tackling some aspect of this multifaceted concept [4]. Amongst other things, a subset of this research focuses on mechanisms to distribute customer requests between the CSPs involved in the federation environment, be it based on SLA considerations, profit motives or to establish consistent and stable federation structures amongst the various CSPs. Chang *et al.* [5] present a mathematical model for cloud federation where client requests can be deterministic or random and use this model to show that federation optimizes the number of servers that are needed to service all client requests in comparison to CSPs acting individually.

Chen *et al.* [6] provide scalable algorithms for optimal partner selection & workload outsourcing in cloud federation networks. In order to model both the cooperation & competition that exists among various CSPs, game theory models are used to obtain stable coalitions of cloud service providers. Next mechanisms for optimal workload factoring and resource sharing in these coalitions are introduced. In [7], Mashayekhy *et al.* the partitioning of client requests between cloud providers is studied and a coalitional cloud federation game is used to

decide on how to distribute client request amongst various CSPs. The main objective function here is the total profit obtained by all CSPs involved in the federation mechanism. A cloud federation formation algorithm is also defined which provides an approximate solution to the problem of deciding which subset of CSPs should receive the current client request.

Niyato *et al.* [8] also use coalitional game theory to tackle the challenges that arise in federated cloud computing, mainly how the resources and revenue can be shared by the coalition and also which coalitional structure is desirable, i.e. incentivizes subsets of CSPs to collaborate. Cooperative games are used to decide on the resource and revenue sharing mechanism. Ray *et al.* [9] consider the scenario where CSPs have preference on which federation to join based on their individual satisfaction level defined by two criteria, the profit and availability achieved in the federation. Therefore a hedonic coalition game is formulated whose objective is to maximize these criteria when forming a federation. Here the quality and trust of each CSP is estimated using a beta-mixture model, with results being used to prevent untrustworthy CSPs from joining the federation. Panda *et al.* [10], present online & offline task scheduling algorithms for heterogeneous multicloud environments, which are simulated and compared using various synthetic datasets. Their novel idea is two consider both preprocessing time needed for initializing a VM as well as its processing time in order to maximize resource utilization.

In [11], Breitgand *et al.* focus on ways to distribute client requests between various CSPs in a federated model so as to maximize CSP profit and also comply with service level agreements. For this an integer linear programming model alongside a greedy algorithm is presented which results in a decrease of power consumption in data centers whilst at the same time load balancing requests between different CSPs. Hassan *et al.* [12] also provide a method for distributed resource allocation in a federated environment based on cooperative game theory. Here centralized and distributed algorithms are presented where each CSP's goal is to increase its individual profit.

Das *et al.* [13] present a quality of service and profit aware cloud confed-

eration model based on different VM allocation policies. A Cloud Federation algorithm is used to decide on federation strategies which maximize the total obtained profit of the CSP requesting federation in such a way as to not reduce Quality of Service (QoS) for local client requests of collaborating CSPs. The partitioning of customer requests to CSPs is used by El Zant *et al.* [14] to decide on fair methods of distributing the revenue obtained from cloud federation. Similarly Li *et al.* [15] present a SLA aware model for pricing in federated clouds that make use of a centralized cloud exchange. Thomas *et al.* [16] focus on the question of partner selection when requests need to be federated between CSPs. Here Analytic Hierarchy Process and the Technique for Order Preference by Similarity to Ideal Solutions is used to rank and select an appropriate federation partner based on QoS parameters. Ray *et al.* [17] also utilize QoS alongside price & trust parameters to conduct a multi-criteria decision analysis that selects the best cloud federation in a specific time period in accordance to user preferences for each parameter.

Work on efficient methods of distributing customer requests in a single cloud environment is more numerous, where most proposed mechanisms are either cost based, SLA based, deadline based or profit based [18]. Hao *et al.* [19] consider the resource allocation problem in the context of distributed clouds, i.e. CSPs that operate multiple datacenters at different geographical locations. To this end they propose a generalized methodology for online resource allocation where users specify their resource placement constraints and approximation algorithms are used for joint allocation of these resource in order to instantiate requested VMs.

Phyo *et al.* [2] emphasize the important role of server consolidation in cloud environments, and try to exploit VM multiplexing by proposing a correlation based VM placement approach in cloud data centers which considers the CPU usage of customer requests. In [3], Meng *et al.* also try to place on the concept of economics of scale in cloud data centers by proposing a joint VM provisioning method based on multiplexing which considers consolidating multiple VMs instead of dealing with each individual one, dealing with the peaks and valleys in

the user request workloads. This is done using three design modules, specifically an SLA model, a joint VM sizing technique and a VM selection algorithm.

Moreno *et al.* [20] turn their attention to proposing a dynamic resource provisioning mechanism that deals with another important concept in cloud computing, power efficiency. This mechanism makes use of neural networks to perform resource over allocation with the goal of reducing resource waste in cloud data centers and therefore increasing the efficiency of energy consumption. Power efficiency is also the goal of Portaluri *et al.* [21], where they make use of genetic algorithms for resource allocation to user requests in cloud data centers.

Resource scheduling, which deals with allocating resources to cloud applications, is also another important related research area whose body of work can be of use for request distribution in various multicloud environments [22]. Sotiriadis *et al.* [23] propose the concept of VM scheduling based on resource monitoring data extracted from past resource utilizations. Their proposed VM placement algorithm, which uses machine learning models to predict VM resource usage per server, applies to both normal cloud computing settings as well as intercloud (or federated) environments.

4. The Framework & Theoretical Results

As previously described, we want to decide on how to distribute a set of n customer requests between k cloud providers in a federated environment, optimizing the federation's efficiency based on the definition presented in Section 4. Each customer request is defined as a normally distributed random variable X_i with mean μ_{X_i} and standard deviation σ_{X_i}. The correlation between every two random variable X_i and X_j is defined by their covariance r_{X_i,X_j} denoting the overlap that the two requests' demand needs have with each other (the alignment in the peaks and valleys of their resource consumption in time). We define a request graph G as a weighted complete graph whose vertices are the given customer requests (X_is) and the weight of the edge between X_i and X_j is r_{X_i,X_j}. Hence the problem of distributing request among k cloud providers is

equivalent to the problem of partitioning the request graph into k components.

The efficiency of these partitioned subsets, each denoted by $S \subseteq \{X_1, X_2, \cdots, X_n\}$, is measured by an objective function $f(S)$. Therefore the efficiency of a set of requests S, can be computed by only the objective function values corresponding to the vertices and edges in the partition of the request graph induced by the vertices in S. For $X_S = \sum_{X_i \in S} X_i$ different objective functions, denoted by $f(S)$, will be considered:

- μ_S: The mean of X_S
- σ_S: The standard deviation of X_S
- σ_S^2: The variance of X_S
- $c_v = \sigma_S/\mu_S = \frac{\sqrt{\sum_{X_i \in S} \sigma^2_{X_i} + \sum_{X_i, X_j \in S} r_{X_i, X_j}}}{\sum_{X_i \in S} \mu_{X_i}}$: The ratio of the standard deviation of X_S to its mean, also known as the coefficient of variation
- σ_S^2/μ_S: The ratio of the variance of X_S to its mean. This objective function can be used as an approximation for c_v as it is more tractable

The problem of maximizing efficiency in the request partitioning mechanism when k cloud service providers exist is defined as follows (we call this problem the k-FEOptRP problem):

$$\text{Minimize} \sum_{i=1}^{k} f(S_i)$$

$$\text{subject to} \bigcup_{i=1}^{k} S_i = \{X_1, X_2, \cdots, X_n\}$$

$$S_i \cap S_j = \emptyset \qquad ,1 \leq i \neq j \leq k$$

Based on the various objective functions, different problems of interest can be defined. In Table 1, we briefly document the theoretical results of this paper pertaining to the k-FEOptRP problem. An important part of these theoretical results is the NP-Hardness of k-FEOptRP for the objective function σ_S^2/μ_S, being the best approximation for the smoothness (c_v) of a set of customer requests federated to some CSP.

Table 1: Theoretical results for the k-FEOptRP problem for each objective function.

Objective Function	**k-FEOptRP**
μ_S	Trivial
σ_S^2	NP-Hard
σ_S^2/μ_S	NP-Hard
σ_S	Open Problem
σ_S/μ_S	Open problem

We follow this section by providing details for these theoretical results.

Theorem 1. *The k-FEOptRP problem for every $k > 1$ and for the objective function $f(S) = \sigma_S^2$ is NP-Hard.*

Proof. We reduce this to the Max k-Cut problem where we want to partition a given graph into k partitions such that the number of edges residing between different partitions is maximized. It is known that this problem is NP-Hard for each $k \geq 2$ [24]. We prove the reduction for $k = 2$. For $k > 2$ the justification is the same. To reduce the Max 2-Cut problem to 2-FEOptRP, assume that a graph $G = (V, E)$ with n nodes $\{v_1, v_2, \cdots, v_n\}$ is given and we want to find its maximum cut. To this end, we consider a set of requests $X = \{X_1, X_2, \cdots, X_n\}$ with the following specifications:

- $\mu_{X_i} = 1$ for each $X_i \in X$,
- $\sigma_{X_i} = 0$ for each $X_i \in X$,
- $r_{X_i,X_j} = 1$ for each $X_i, X_j \in X$ and $(v_i, v_j) \in E$ and
- $r_{X_i,X_j} = 0$ for each $X_i, X_j \in X$ and $(v_i, v_j) \notin E$.

Partitioning the request graph of this instance into two sets S and $T = X \setminus S$

and minimizing

$$\begin{aligned}\sigma_S^2 + \sigma_T^2 &= \sum_{X_i,X_j \in S} r_{X_i,X_j} + \sum_{X_i \in S} \sigma_{X_i}^2 \\ &+ \sum_{X_i,X_j \in T} r_{X_i,X_j} + \sum_{X_i \in T} \sigma_{X_i}^2 \\ &= \sum_{X_i,X_j \in X} r_{X_i,X_j} - \sum_{X_i \in S, X_j \in T} r_{X_i,X_j}\end{aligned}$$

is equivalent to maximizing $\sum_{X_i \in S, X_j \in T} r_{X_i,X_j}$ which is equal to the number of edges between the two partitions in G, concluding the proof. □

Theorem 2. *The* k-FEOptRP *problem for every* $k > 1$ *and for the objective function* $f(S) = \sigma_S^2/\mu_S$ *is NP-Hard.*

Proof. First we prove the theorem for $k = 2$. To do this we reduce the Max 2-Cut problem to the 2-FEOptRP problem with objective function σ_S^2/μ_S. Assume that a graph $G = (V, E)$ is given where $V = \{v_1, v_2, ..., v_n\}$ and $E \subset V \times V$ is the set of G's edges. Our goal is to find its maximum cut by solving 2-FEOptRP for an input instance $\mathcal{I}$ constructed from G. In $\mathcal{I}$, the set of requests is $Z = X \cup Y$ where $X = \{X_1, X_2, \cdots, X_n\}$ and $Y = \{Y_1, Y_2, \cdots, Y_n\}$. More than that $\mathcal{I}$ has the following specifications:

- For all $1 \leq i \leq n$, we set $\mu_{X_i} = \mu_{Y_i} = 1$.
- For all $1 \leq i \leq n$, we set $\sigma_{X_i} = \sigma_{Y_i} = 0$.
- For all $(v_i, v_j) \in E$, we set r_{X_i,X_j} and r_{Y_i,Y_j} equal to 1. For all $1 \leq i \leq n$, we set $r_{X_i,Y_i} = L$ (where L is a large number) and for all the other cases we set $r_{P,Q} = 0$ $(P, Q \in Z)$.

Since for each $1 \leq i \leq n$, $\mu_{X_i} = \mu_{Y_i} = 1$, for each $S \subseteq Z$, we have $\mu_S = |S|$. Assume that H is the request graph of this instance. For each subset $S \subseteq Z$, we have:

$$\sigma_S^2 = \sum_{P \in S} \sigma_P^2 + \sum_{P,Q \in S} r_{P,Q} = 0 + \mathcal{E}_H(S),$$

where $\mathcal{E}_H(S)$ is the sum of the weights of H's edges between S's members. Thus for each $S \subseteq X$, σ_S^2 is equal to the number of edges between S's corresponding nodes in G. We have the same equation for each $T \subseteq Y$.

An optimal solution of 2-FEOptRP to this instance gives us two sets S and T for which

$$f(S) + f(T) = \sigma_S^2/\mu_S + \sigma_T^2/\mu_T = \frac{\mathcal{E}_H(S)}{|S|} + \frac{\mathcal{E}_H(T)}{|T|}$$

is minimized. Assume that $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$ such that $S_1, T_1 \subseteq X$, $S_2, T_2 \subseteq Y$, $S_1 \cap S_2 = \emptyset$ and $T_1 \cap T_2 = \emptyset$. Since L is a large number, neither S nor T include both X_i and Y_i for each $1 \leq i \leq n$, therefore if S has m members of X, T must have exactly m members of Y and vice versa. Thus S and T have the same cardinality; more precisely $|S_1| = |T_2|$ and $|S_2| = |T_1|$. Thus, $|S| = |S_1| + |S_2| = |T_1| + |T_2| = |T|$. Thus, minimizing $f(S) + f(T)$ is the same as minimizing $\mathcal{E}_H(S) + \mathcal{E}_H(T)$ which is equal to maximizing the weight of edges between S and T. For maximum weight of edges between S and T is achieved by n edges with weight L plus the maximum number of edges with weight 1 between $S_1 - S_2$, and $T_1 - T_2$ which can both be the max-cut of G.

For $k > 2$, we reduce the 2-FEOptRP problem (which we know its NP-Hardness) to the k-FEOptRP problem. To this end, we add $k - 2$ requests with mean equal to 1 and variance equal to zero. We also set their covariance to every other request equal to a very large number. It is easy to see that solving k-FEOptRP for this instance puts each of these newly added requests in a separate partition and solves the 2-FEOptRP problem for the set of input requests. □

Analyzing the efficiency problem with the CoV set as the objective function is difficult, but by considering the complexity of the above related metrics, we can conjure that it is also NP-Hard. The proof of this is left as an open problem for future work.

Conjecture 1. *The* k-FEOptRP *problem for every* $k > 1$ *and for the objective function* $f(S) = \sigma_S$ *is NP-Hard.*

Conjecture 2. *The k-FEOptRP problem for every $k > 1$ and for the objective function $f(S) = c_v = \sigma_S/\mu_S$ is NP-Hard.*

In the next section we concentrate on heuristic algorithms for the k-FEOptRP problem when using c_v (CoV) as the objective function. This objective function is chosen as it is the one that measures the smoothness of resource consumption for a sub partition of requests, which, as we saw in the Section 1, shows how much we can utilize statistical multiplexing to the benefit of the CSPs.

5. Simulation

To be able to solve the k-FEOptRP problem with the objective function set to the Coefficient of Variation in a feasible time and with results as close to optimal as possible, different heuristics can be considered. In this section we will look at various optimization algorithms and heuristics and explain the simulation environment in which they will be run and which provides the basis for our comparison of these methods. One of the first things we need to do is data generation, so that we have a large enough dataset on which the heuristics can be tested on. This generation must be done in such a way so as to closely replicated the behavior of real world customer requests that are run on today's cloud computing environments. For this we use the results of Morena *et al.* [25] who by sampling publicly released Google Cloud tracelog data [5], show that CPU usage requests of cloud based tasks follow a lognormal distribution.

We consider the autoregressive order-1 model or AR(1)[26], a time series input model which is used to create identically distributed, dependent and covariance-stationary random variables in the following way:

$$X_t = \mu + \phi(X_{t-1} - \mu) + \varepsilon_t, t > 1$$

where ε_t are independent and normally distributed with mean 0 and variance σ_ε^2 and $-1 < \phi < 1$. Also X_1 is generated from the normal distribution with mean

[5] https://github.com/google/cluster-data

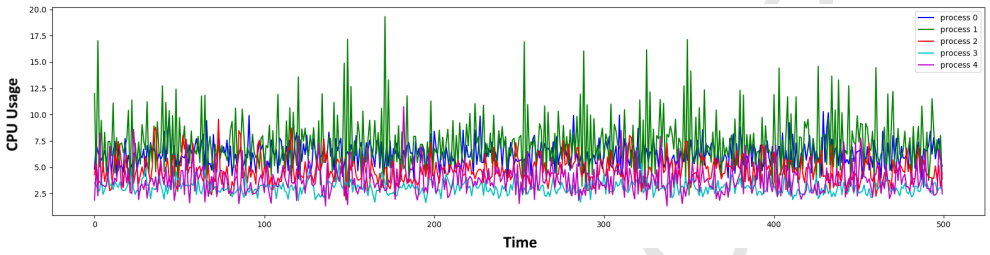


Figure 4: Here we can see an instance of sample CPU usage data generated for five tasks using the lognormal-AR(1) method as described in the text. In this example $1 < \mu < 2$ and $0.1 < \sigma^2 < 0.5$.

μ and variance $\sigma_\varepsilon^2/(1-\phi^2)$. As the AR(1) model creates normally distributed random variables instead of lognormally distributed ones, we use X'_t where:

$$X'_t = e^{X_t}$$

In this paper we call the above random variable generation method the lognormal-AR(1) method. Figure 4 shows generated CPU usage using the lognormal-AR(1) method for 5 requests with 500 samples taken for each one ($t = 500$).

We ran various optimization algorithms on the generated data so as to compare the performance of different heuristics in solving the k-FEOptRP problem. Here we are dealing with a search space whose points are a partitioning of requests to CSPs. Also the set of neighbors of a point p are points similar to p, but in which one task is located on a different CSP than in p. For example the two below points are neighbors in a space comprised of two CSPs and five tasks:

$$p_1 = CSP1\{Proc1, Proc2, Proc3\}, CSP2\{Proc4, Proc5\}$$

$$p_2 = CSP1\{Proc1, Proc2\}, CSP2\{Proc3, Proc4, Proc5\}$$

5.1. Algorithms

In this section each algorithm used in our simulations is described in summary:

- `Alg1`: Here the tasks are randomly partitioned between the CSPs. This process is repeated 2000 times and the best state found among them is chosen. This is used for comparison purposes with other algorithms.

- `Alg2`: This algorithm is based on the hill climbing search method. Starting from a random initial state, at each step a random neighbor of the current state with a lower cost function (in this context meaning lower CoV sum i.e. higher efficiency) is chosen as the new state. This is done for a number of iterations (2000). This process is repeated 10 times from a new random initial state and the best overall state found is chosen.

- `Alg3`: This algorithm differs from Alg2 in that at each step, all neighboring states are checked and we move to the neighbor state with the lowest cost function. This is sometimes called Steepest Ascend hill climbing. The stopping condition for this algorithm is that there are no more neighboring states with a better cost function than the current one. Alg3 is also repeated 10 times from random starting points and the best found state is chosen.

- `Alg4`: This algorithm is based on the family of Simulated Annealing exploratory methods where a temperature variable, T, is used that decreases as time passes. Initially when the temperature is high both neighbors with higher and lower cost functions can be chosen, but the probability of choosing a worst neighbor decreases as the algorithm progresses (i.e. the temperature decreases). This probability is usually based on the expression $e^{\Delta C/T}$ where ΔC is the difference in cost function between the current state and one of its neighbors. In our simulation, the initial temperature is set as $T_0 = 1$ and for each step $T_i = T_{i-1} * 0.999$ until a stopping condition is met ($T < 0.00001$).

- `Alg5`: This algorithm is based on the Late Acceptance hill climbing method, a scale independent search method in which a list of fixed length L_h of previous values of the current cost function is used for comparison when

choosing candidate neighbors [27]. In our implementation we start from a randomly generated point and set L_h to 100. At each step i a random candidate neighboring solution is analyzed. If its cost is lower than the current point we add the cost to the beginning of L_h, removing the list's last value when necessary. If the candidate neighbor has a higher cost value then it is accepted only if its cost is lower than $i \, mod \, L_h$ in step i. We also keep track of idle steps, those in which the candidate solution did not meet any of the above criteria, using it as a stopping condition. When the total number of consecutive idle steps reaches 2000 the algorithm returns the best point (i.e. partition of requests) found so far.

- `Alg6`: For our last algorithm we follow a different strategy. Instead of trying to search the CoV space for an optimal solution, we consider σ^2 as an approximation of this value. This is in line with the fact that $c_v = \sigma_S/\mu_S$, and therefore we want the vertices with the smallest covariance between them to be partitioned to a single CSP in order to obtain a lower σ_S value. σ^2 can be computed for each partition using the covariance matrix $CovMatrix$, showing the covariance between any two processes (i.e. requests). As we showed in 1, this problem can be reduced to the MaxCut problem, where the processes are the graph's nodes which must be partitioned into k partitions. Here k is the number of CSPs that participate in the federation mechanism. As the MaxCut problem is NP-Complete we utilize the multiple search operator heuristic proposed by Ma *et al.* [28]. Here to escape local maximums five different search operators are used, O_i. The O_1 search operator applies single transfer move operations to the best neighbor of the current point. O_2 differs in that it applies double transfer move operations, selecting the one with highest move gain. O_3 is again a single transfer operator which also considers a tabu list [29] in its search approach. O_4 is similar to O_2 but selects the target partitions for transfer at random. Finally O_5 applies completely random moves from the current state in order to diversify the search space. We applying all 5

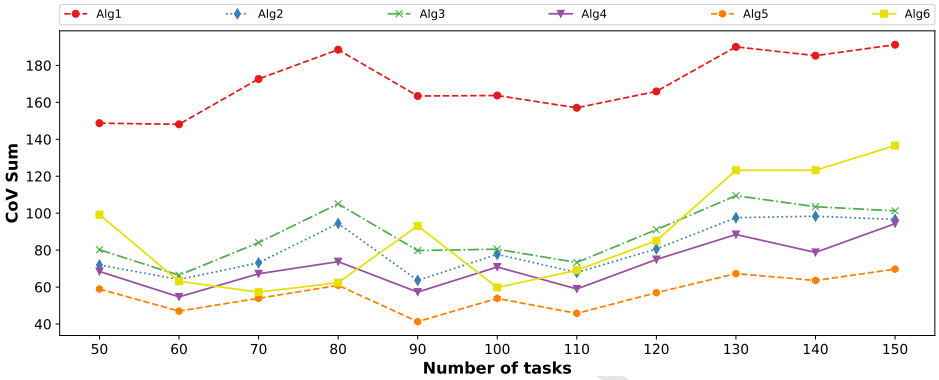


Figure 5: Simulation results for the k-FEOPTRP problem with 10 CSPs when the number of tasks varies between 50 and 150. A lower CoV number indicates higher efficiency i.e. higher utilization of statistical multiplexing.

operators using the method implied by [28].

5.2. Simulation Results

To evaluate the performance of our proposed optimization algorithms, we ran various simulations. The simulation uses input process data obtained from the lognormal-AR(1) method as was described in Section 5 and was run on a typical workstation with a core i7 processor & 16 GBs of ram. In one setup the number of cloud service providers was fixed and we examined the results when varying the number of tasks (user requests) between 50 and 150. In another setup we kept the number of tasks constant whilst the number of CSPs ranged from 5 to 15. The simulation results can be seen in Figures 5 and 6 where the shown cost is the sum of CoV values, i.e. $f(S_i)$ in the definition of k-FEOPTRP. Keep in mind that due to the stochastic nature of our heuristic algorithms each simulation scenario was run 100 times, with each data point shown being the average of these runs.

As can be seen from these figures, all examined algorithms outperformed Alg1, as was to be expected. The results of the simulation can be summarized as below:

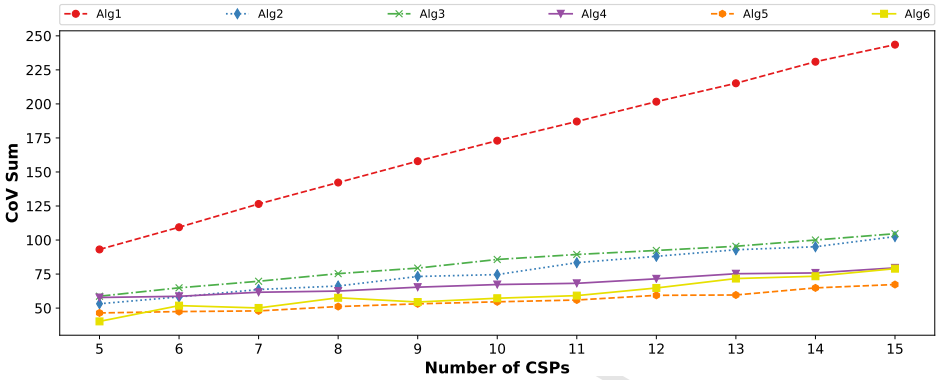


Figure 6: Simulation results for the k-FEOPTRP problem with 70 tasks when the number of CSPs varies between 5 and 15. A lower CoV number indicates higher efficiency i.e. higher utilization of statistical multiplexing.

- The best overall efficiency was obtained using Alg5 which is based on the Late Acceptance Simulated Annealing method, with second best being Alg4 which is more closely related to the classic Simulated Annealing heuristic.

- Alg2 outperformed Alg3. This is interesting as the main difference between these two algorithms is the fact that Alg3 always chooses its best neighbor (lowest CoV sum) as the next state whereas Alg2 randomly selects a neighbor with lower cost than the current step. Alg3's choosing of the steepest slopes to traverse lead it local maximums in which it is stuck.

- The above statement has us believe that our search space is one which includes many local maximums. This can also be seen from the fact that algorithms which diversify their moves and include traversing both better & worse neighbor states outperform others (i.e. Alg4 & Alg5).

- For Alg6 we can see that at some data points this algorithm outperforms most others whilst at the same time there are data points in which its results are lacking. Keeping in mind the fact that Alg6 uses σ^2 as an indirect measure for optimizing c_v, we can see that its output is highly

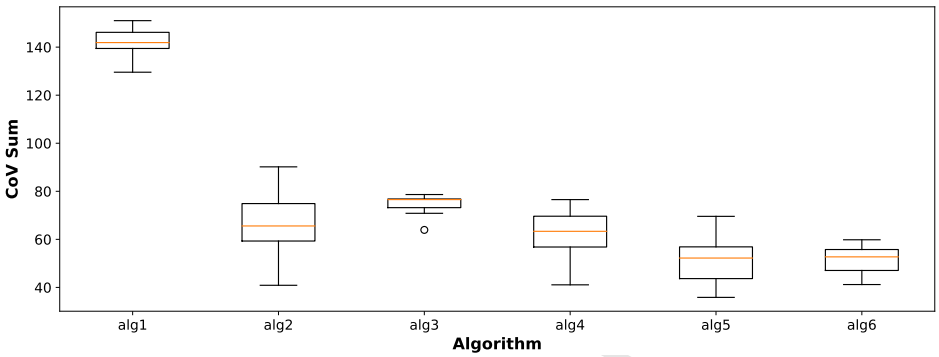


Figure 7: Box plot diagram showing the results of running all six algorithms 100 times on a data point where the number of cloud service providers is 8 and the number of processes is 70.

dependent on the input data ($CovMatrix$ of processes) and therefore using such indirect measures may not always lead to acceptable results. This can further be seen from the fact that in Figure 6 in which the processes are kept constant, the result of Alg6 is consistent for different number of CSPs, mainly between the results for Alg5 & Alg4.

The box plot for a single data point of our obtained simulation results (based on 100 runs) is presented in Figure 7. Here we can see that the variance of Alg2's results is large as the stochastic nature of this algorithm in choosing its next neighbor can lead to different outcomes. For Alg3 the results show less variety, which is the case when the value for the local maximums reachable by traversing the steepest slopes are in the same range. The outlier here, which is close to the mean of Alg2, shows that it is possible to reach the same efficiency that Alg2 can reach, but on the whole Alg2 is more efficient. For Alg4 we can see that more results are in the range between the third quartile and the maximum cost obtained by this algorithm. The situation for Alg5 is the reverse of this, with more results near the minimum cost, showing the fact that on the whole Alg5 is more efficient than Alg4. Also for Alg6 we can see less variety in the results compared to the others, apart from Alg3.

Table 2: Theoretical results for the k-FFOptRP problem for each objective function.

Objective Function	**k-FFOptRP**
μ_S	NP-Hard
σ_S^2	NP-Hard
σ_S^2/μ_S	NP-Hard
σ_S	Open problem
σ_S/μ_S	Open problem

6. The Question of Fairness

Taking in mind the selfish behavior of cloud providers in real world settings, the fairness of the request partitioning mechanism between them is another important factor that future work can address in order to make sure CSPs have an incentive to continue their collaboration in a federated environment. Here we will analyze the theoretical framework for such a mechanism. For this we introduce the k-FFOptRP problem, which attempts to find a request distribution scenario where the worst case efficiency based on the objective function for each CSP is minimized:

$$\begin{aligned} &\text{Minimize } \max\{f(S_i)\}_{i=1\cdots k} \\ &\text{subject to } \bigcup_{i=1}^{k} S_i = \{X_1, X_2, \cdots, X_n\} \\ &\qquad S_i \cap S_j = \emptyset \qquad\qquad , 1 \leq i \neq j \leq k \end{aligned}$$

Like the efficiency problem, different objective functions were considered. The theoretical results can be seen in Table 2, with details provided below.

Theorem 3. *The k-FFOptRP problem for every $k > 1$ and for the objective function $f(S) = \mu_S$ is NP-Hard.*

Proof. Assume that we are given a set $T = \{a_1, a_2, \cdots, a_n\}$ and we are asked to partition this set into k sets $T_1, T_2, \cdots, T_k$ such that for each $1 \leq i < j \leq n$,

$T_i \cap T_j = \emptyset$ and $\bigcup_i T_i = T$ and also for each $1 \leq i \leq k$, $\sum_{a_p \in T_i} a_p = \frac{1}{k} \sum_{a_q \in T} a_q$. This problem is called the k-PARTITION problem and we know that it is NP-Hard [30]. Consider an instance of the k-FFOPTRP in which we have n customer requests $X = \{X_1, X_2, \cdots, X_n\}$ with $\mu_{X_i} = a_i$ for all $1 \leq i \leq n$. It can be easily seen that the partitioning of the set T into k equal sum subsets is possible if and only if the minimum of the maximum μ_{S_i} in the k-FFOPTRP problem is exactly equal to $\frac{1}{k}\mu_X$. □

Theorem 4. *The k-FFOPTRP problem for every $k > 1$ and for the objective function $f(S) = \sigma_S^2$ is NP-Hard.*

Proof. The proof is similar to the proof of theorem 3 with some minor modifications. We reduce the k-PARTITION problem to this problem. We are given a set $T = \{a_1, a_2, \cdots, a_n\}$ and we are asked to partition this set into k sets $T_1, T_2, \cdots, T_k$ such that for each $1 \leq i < j \leq n$, $T_i \cap T_j = \emptyset$, $\bigcup_i T_i = T$ and also for each $1 \leq i \leq k$, $\sum_{a_p \in T_i} a_p = \frac{1}{k} \sum_{a_q \in T} a_q$. We consider an instance of the k-FFOPTRP problem with the request set $X = \{X_1, X_2, \cdots, X_n\}$ with $\sigma^2_{X_i} = a_i$ for each $X_i \in X$ and $r_{X_i, X_j} = 0$ for each $X_i, X_j \in X$. X can be partitioned into k sets $S_1, S_2, \cdots, S_k$ such that $min(max(\sigma^2_{S_k})) = \frac{\sigma^2_X}{k}$ if and only if the answer to the defined k-PARTITION problem is true. □

Theorem 5. *The k-FFOPTRP problem for every $k > 1$ and for the objective function $f(S) = \sigma_S^2/\mu_S$ is NP-Hard.*

Proof. The proof is very similar to the proof of Theorem 2 and Theorem 4. We first prove the theorem for $k = 2$ by reducing the 2-PARTITION problem to 2-FFOPTRP and then for every $k > 2$, we reduce the 2-FFOPTRP problem to k-FFOPTRP which concludes the proof.

For the first part of the proof, assume that we are given a set $A = \{a_1, a_2, \cdots, a_n\}$ and are asked to partition this set into 2 sets A_1 and A_2 such that $A_1 \cap A_2 = \emptyset$, $A_1 \cup A_2 = A$ and $\sum_{a_p \in A_i} a_p = \frac{1}{2} \sum_{a_q \in A} a_q$ for each $i = 1$ and $i = 2$. We construct an instance $\mathcal{I}$ of the 2-FFOPTRP problem with the following specifications:

- the set of requests is $Z = X \cup Y$ such that $X = \{X_1, X_2, \cdots, X_n\}$ and $Y = \{Y_1, Y_2, \cdots, Y_n\}$
- For all $1 \leq i \leq n$, we set $\mu_{X_i} = \mu_{Y_i} = 1$.
- For all $1 \leq i \leq n$, we set $\sigma^2_{X_i} = a_i$ and $\sigma^2_{Y_i} = 0$.
- For all $1 \leq i \leq n$, we set $r_{X_i,Y_i} = L$ (where L is a large number) and for all other cases we set $r_{P,Q} = 0$ $(P, Q \in Z)$.

Assume that the answer to the 2-Partition problem is yes. We prove that solving the 2-FFOptRP problem for $\mathcal{I}$ gives us two sets $S, T \subseteq Z$ such that $\frac{\sigma^2_S}{\mu_S} = \frac{\sigma^2_T}{\mu_T}$. As we saw in the proof of theorem 2, since L is a large number, neither S nor T include both X_i and Y_i for each $1 \leq i \leq n$, therefore if S has m members of X, T must have exactly m members of Y and vice-versa. Therefore S and T have the same cardinality and thus $\mu_S = \mu_T$.

So the only thing remaining is to prove that $\sigma^2_S = \sigma^2_T$. Assume that S_1 is the maximal subset of S which is not a subset of Y. Since for each $Q \in Y$, $\sigma^2_Q = 0$, then $\sigma^2_S = \sigma^2_{S_1}$. Define $T_1 \subseteq T$ in the same way such that $\sigma^2_T = \sigma^2_{T_1}$. Since A can be partitioned into two equal sum sets A_1 and A_2, in the optimal solution for the 2-FFOptRP problem which minimizes the maximum of σ^2_S and σ^2_T, X can be partitioned into two sets S_1 and T_1 with $\sigma^2_{S_1} = \sigma^2_{T_1}$ and thus $\sigma^2_S = \sigma^2_T$

With the same justification, we can prove that if the answer to the 2-Partition is no, then we have $\frac{\sigma^2_S}{\mu_S} \neq \frac{\sigma^2_T}{\mu_T}$ in the optimal solution of the 2-FFOptRP problem.

Now, we prove the theorem for $k > 2$. We reduce the 2-FFOptRP problem to the k-FFOptRP problem. To do this we add $k-2$ requests with their mean equal to 1 and variance equal to zero. We also set their covariance to every other request equal to a very large number. It is easy to see that solving k-FFOptRP for this instance puts each of these newly added requests in a separate partition and solves the 2-FFOptRP problem for the set of input requests. □

Also, we anticipate the below conjectures although their proof is left as an open problem for future work.

Conjecture 3. *The* k-FFOptRP *problem for every* $k > 1$ *and for the objective function* $f(S) = \sigma_S$ *is NP-Hard.*

Conjecture 4. *The* k-FFOptRP *problem for every* $k > 1$ *and for the objective function* $f(S) = c_v = \sigma_S/\mu_S$ *is NP-Hard.*

7. Conclusion & Future Works

In this paper we looked at an efficient method of distributing customer requests among cloud service providers in a federated cloud environment, with the goal of utilizing statistical multiplexing. We showed how the coefficient of variation can be considered as an important objective function in this regard, showing the smoothness of a collection of requests. We described the k-FEOptRP problem for efficient request distribution and examined the complexity of various statistical functions being set as its objective function, with most of them being NP-Hard. Finally we looked at various heuristic algorithms for the k-FEOptRP problem with CoV being set as its objective function and compared them in various simulations scenarios. Our results show that our algorithm based on the Late Acceptance Hill Climbing method outperformed others.

In our work we looked at the behavior of each request in a single time frame, distributing them to various CSPs in one go. In reality each customer request or process will show much varying resource needs and correlation with other requests in its lifetime and therefore we can look at its behavior dynamically, changing the CSP to which it is partitioned to multiple times. Hallac *et al.* [31] use a similar concept in looking at covariance-based clustering of multivariate time series data which can be of use in this regard. This is an important direction our future work can take. We would like to also test out CoV based request partitioning on larger data sets in real world scenarios in which large scale enterprise grade processes are at play.

Fairness is also another import direction that future work can take on, focusing on examining heuristics & approximation algorithms to solve the k-

FFOptRP problem in a feasible way. Lastly, the different cloud technologies & architecture used by a CSP can affect performance & resource consumption patterns[32], which when considered can help us obtain a more robust request distribution model for real world usage. One direction for future work we are working on is to obtain a mathematical model which captures the effect that these difference environments have on resource usage of requests in clouds, i.e. their effect on metrics such as the mean and variance of resource consumption.

References

[1] J. Weinman, Cloudonomics: a rigorous approach to cloud benefit quantification, J. Software Technol 14 (4) (2011) 10–18.

[2] Z. L. Phyo, T. Thein, Correlation based vms placement resource provision, International Journal of Computer Science & Information Technology 5 (1) (2013) 95.

[3] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, D. Pendarakis, Efficient resource provisioning in compute clouds via vm multiplexing, in: Proceedings of the 7th international conference on Autonomic computing, ACM, 2010, pp. 11–20.

[4] A. N. Toosi, R. N. Calheiros, R. Buyya, Interconnected cloud computing environments: Challenges, taxonomy, and survey, ACM Computing Surveys (CSUR) 47 (1) (2014) 7.

[5] B.-Y. Chang, B. Kim, S. Yoon, D.-W. Seo, An evaluation of federated cloud computing effect with service level, in: Computational Problem-Solving (ICCP), 2011 International Conference on, IEEE, 2011, pp. 105–108.

[6] H. Chen, B. An, D. Niyato, Y. C. Soh, C. Miao, Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks, IEEE Journal on Selected Areas in Communications 35 (3) (2017) 557–570.

[7] L. Mashayekhy, M. M. Nejad, D. Grosu, Cloud federations in the sky: Formation game and mechanism, Cloud Computing, IEEE Transactions on 3 (1) (2015) 14–27.

[8] D. Niyato, A. V. Vasilakos, Z. Kun, Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach, in: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Computer Society, 2011, pp. 215–224.

[9] B. Ray, A. Saha, S. Khatua, S. Roy, Quality and profit assured trusted cloud federation formation: Game theory based approach, IEEE Transactions on Services Computing (2018) 1.

[10] S. K. Panda, S. K. Pande, S. Das, Task partitioning scheduling algorithms for heterogeneous multi-cloud environment, Arabian Journal for Science and Engineering 43 (2) (2018) 913–933.

[11] D. Breitgand, A. Marashini, J. Tordsson, Policy-driven service placement optimization in federated clouds, IBM Research Division, Tech. Rep 9 (2011) 11–15.

[12] M. M. Hassan, M. S. Hossain, A. J. Sarkar, E.-N. Huh, Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform, Information Systems Frontiers 16 (4) (2014) 523–542.

[13] A. K. Das, T. Adhikary, M. A. Razzaque, E. J. Cho, C. S. Hong, A QoS and profit aware cloud confederation model for IaaS service providers, in: Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, ACM, 2014, p. 42.

[14] B. El Zant, I. Amigo, M. Gagnaire, Federation and revenue sharing in cloud computing environment, in: Cloud Engineering (IC2E), 2014 IEEE International Conference on, IEEE, 2014, pp. 446–451.

[15] H. Li, H. Lu, Y. Kang, J. Lu, A Research of Pricing for the Cloud Resource based on the Cloud Bank SLA, Journal of Network & Information Security 4 (1) (2013) 69–78.

[16] M. V. Thomas, K. Chandrasekaran, Dynamic partner selection in cloud federation for ensuring the quality of service for cloud consumers, International Journal of Modeling, Simulation, and Scientific Computing (2017) 1750036.

[17] B. K. Ray, A. I. Middya, S. Roy, S. Khatua, Multi-criteria based federation selection in cloud, in: Communication Systems and Networks (COMSNETS), 2017 9th International Conference on, IEEE, 2017, pp. 182–189.

[18] S. Kukreja, S. Dalal, Performance analysis of cloud resource provisioning algorithms, in: Progress in Advanced Computing and Intelligent Engineering, Springer, 2018, pp. 593–602.

[19] F. Hao, M. Kodialam, T. Lakshman, S. Mukherjee, Online allocation of virtual machines in a distributed cloud, IEEE/ACM Transactions on Networking (TON) 25 (1) (2017) 238–249.

[20] I. S. Moreno, J. Xu, Neural network-based overallocation for improved energy-efficiency in real-time cloud environments, in: Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on, IEEE, 2012, pp. 119–126.

[21] G. Portaluri, S. Giordano, D. Kliazovich, B. Dorronsoro, A power efficient genetic algorithm for resource allocation in cloud computing data centers, in: Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, IEEE, 2014, pp. 58–63.

[22] S. Singh, I. Chana, A survey on resource scheduling in cloud computing: Issues and challenges, Journal of grid computing 14 (2) (2016) 217–264.

[23] S. Sotiriadis, N. Bessis, R. Buyya, Self managed virtual machine scheduling in cloud systems, Information Sciences 433 (2018) 381–400.

[24] A. Frieze, M. Jerrum, Improved approximation algorithms for maxk-cut and max bisection, Algorithmica 18 (1) (1997) 67–81.

[25] I. S. Moreno, P. Garraghan, P. Townend, J. Xu, An approach for characterizing workloads in google cloud to derive realistic resource utilization models, in: Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on, IEEE, 2013, pp. 49–60.

[26] J. Banks, Discrete-event system simulation, Prentice Hall, Upper Saddle River, N.J. Singapore, 2010.

[27] E. K. Burke, Y. Bykov, The late acceptance hill-climbing heuristic, European Journal of Operational Research 258 (1) (2017) 70–78.

[28] F. Ma, J.-K. Hao, A multiple search operator heuristic for the max-k-cut problem, Annals of Operations Research 248 (1-2) (2017) 365–403.

[29] F. Glover, M. Laguna, Tabu search, in: Handbook of Combinatorial Optimization, Springer, 2013, pp. 3261–3362.

[30] R. M. Karp, Reducibility among combinatorial problems, in: Complexity of computer computations, Springer, 1972, pp. 85–103.

[31] D. Hallac, S. Vare, S. Boyd, J. Leskovec, Toeplitz inverse covariance-based clustering of multivariate time series data, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 215–223.

[32] S. K. Tesfatsion, C. Klein, J. Tordsson, Virtualization techniques compared: Performance, resource, and power usage overheads in clouds, in: Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering, ACM, 2018, pp. 145–156.

Moslem Habibi received his BSc and MSc degree in software engineering from Sharif University of Technology, in 2010 and 2012 respectively. He is currently a PhD candidate in software engineering at Sharif University of Technology where his research centers on modeling economic incentives for cloud federation. His research interests include cloud computing, cloud interoperability, cloud computing economic models and complex networks.

MohammadAmin Fazli received his BSc in hardware engineering and MSc and PhD in software engineering from Sharif University of Technology, in 2009, 2011 and 2015 respectively. He is currently a faculty member at Sharif University of Technology and R&D head at Sharif's Intelligent Information Center. His research interests include Game Theory, Combinatorial Optimization, Computational Business and Economics, Graphs and Combinatorics, Complex networks and Dynamical Systems.

Ali Movaghar is a Professor in the Department of Computer Engineering at Sharif University of Technology in Tehran, Iran and has been on the Sharif faculty since 1993. He received his B.S. degree in Electrical Engineering from the University of Tehran in 1977, and M.S. and Ph.D. degrees in Computer, Information, and Control Engineering from the University of Michigan, Ann Arbor, in 1979 and 1985, respectively. He visited the Institut National de Recherche en Informatique et en Automatique in Paris, France and the Department of Electrical Engineering and Computer Science at the University of California, Irvine in 1984 and 2011, respectively, worked at AT&T Information Systems in Naperville, IL in 1985-1986, and taught at the University of Michigan, Ann Arbor in 1987-1989. His research interests include performance/dependability modeling and formal verification of wireless networks and distributed real-time systems. He is a senior member of the IEEE and the ACM.

- Statistical Multiplexing based request partitioning for federated cloud environments
- Definition of the efficiency & fairness problem for smooth request partitioning
- Consideration of statistical objective functions such as the coefficient of variation
- Theoretical evaluation of the hardness of efficient & fair request partitioning
- Comparison of heuristic algorithms to solve the efficiency problem