



A Cost-Aware Management Framework for Placement of Data-Intensive Applications on Federated Cloud

Moustafa Najm¹ · Rakesh Tripathi¹ · Mohammad Shadi Alhakeem² · Venkatesh Tamarapalli¹

Received: 18 September 2020 / Revised: 9 January 2021 / Accepted: 11 February 2021 /
Published online: 5 March 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract

Modern big data analysis and business intelligence applications come with high resource demands, along with a requirement for large data transfer between storage and compute nodes. Since the available resources of a single data center might not be sufficient to host these applications, federated cloud systems present a promising solution. The objective of each cloud service provider in a federation is to maximize its own profit and to minimize its total operating cost. To achieve these objectives, spatial variation in the energy cost and bandwidth cost could be leveraged while allocating the workload. We propose a hierarchical approach for resource management in a federated cloud, catering to the requirements of each provider in the federation. We formulate the placement of a data-intensive applications as an optimization problem to minimize the total operating cost, including the energy and communication costs. We propose an algorithm to allocate the virtual components of a data-intensive application, in two phases; *partitioning* and *mapping*. While *partitioning* creates clusters of correlated nodes, *mapping* allocates the created clusters to data centers that minimize the total cost. Through extensive experiments in different scenarios, we demonstrate that the proposed algorithm achieves a significant reduction in the total operating cost.

Keywords Data-intensive applications · Energy cost · Federated cloud data centers · Resource Management

1 Introduction

The world is witnessing an enormous growth in data generation, and it is expected that the data generated in 2020 will be 50 times the data generated in 2011 [1, 2]. With cloud computing becoming the clear future of computing, data-intensive

✉ Moustafa Najm
moustafa.najm@iitg.ac.in

Extended author information available on the last page of the article

applications are increasingly being deployed in the cloud to address the resource requirements [3, 4]. It had been reported that 71% of the cloud applications are data-intensive applications, bringing in an income for cloud providers (CPs) to the tune of \$126 billion [5], and this income is increasing as the cloud computing market is expected to touch \$300 billion by 2021.¹ Accordingly, it becomes increasingly challenging to manage scheduling the data-intensive applications in the cloud [6].

Data-intensive applications often involve bulk data transfer between storage and computing nodes, which poses a major challenge for small CPs [7]. Federated cloud data centers offer a promising solution for handling such applications in a scalable and cost-efficient manner, as they can avoid dimensioning for peak workload and avoid the associated resource wastage via shifting the workload to other data centers (DCs) in the federation [8]. Federated cloud systems aggregate the resources provided by the infrastructure of collaborating CPs into a single platform [9, 10]. They can increase the revenue of a CP by renting out free resources to other CPs and also enable elasticity in serving the workload [11, 12]. Further, federation allows a small CP to offer services at large and global scale without establishing new DCs, by leasing resources from the other CPs [13, 14]. In addition, the geographic diversity of the CPs in the federation may present many other advantages, like high availability, low latency and cost efficiency [15].

Data-intensive applications are typically scheduled on physical servers using two different virtual components that are data blocks (DBs) as storage nodes, and virtual machines (VMs) as compute nodes. Based on the data access patterns between these components, the transferred data size could well exceed 500 TB per day, which presents a major challenge in scheduling data-intensive applications [16]. The placement of communicating virtual components on a single data center is a potential solution to avoid latency and bandwidth cost. A single DC may not be able to accommodate all the virtual components of an application with large resource requirements [17]. Therefore, these components may be allocated across multiple DCs in a federation, resulting in inter-DC communication overhead, wherein the data transfer cost may lead to a drastic increase in the total cost of operation (TCO) [7]. A recent report predicted the annual traffic between DCs to be 20.6 ZB by 2021, resulting in bandwidth costs to the tune of \$4.45 billion [18].

Besides the bandwidth cost, the cost of energy consumed is an additional important issue for large-scale DCs. It is reported that DCs across the globe would consume 20% of the total energy in the world by 2025 [19], and it is estimated to account for 30% to 50% of the TCO for CPs [20]. In the U.S. alone, it is expected to rise up to 140 billion kWh by 2020, costing the CPs about \$13 billion in electricity bills.² Since the electricity price varies remarkably across different countries; ranging between \$60 and \$250 per MWh,³ leveraging the spatial variation in the electricity price would be advantageous for the CPs in a federation to reduce their operating

¹ <https://www.gartner.com/en/newsroom/pressreleases/2018-08-15-gartner-says-cloud-computing-remains-top-emerging-business-risk>.

² <https://www.nrdc.org/media/2014/140826>.

³ https://en.wikipedia.org/wiki/Electricity_pricing.

costs by intelligent routing of workload [8]. While the literature focuses on only either the bandwidth cost or the energy consumption cost, we show that both should be considered to optimize the cost and performance.

In addition, the workload distribution in DCs is typically handled in a centralized manner, which is not scalable and not compatible with the federation requirements [21]. The semi-autonomous nature of each member in the federation leads to diverse objectives in the placement policy. To deal with this issue, we design a hierarchical framework for resource management and workload distribution. We also propose an efficient algorithm for placement of data-intensive applications in a federated cloud. The proposed algorithm considers the correlation of virtual components to reduce inter-DC communication and leverages the electricity price variations across DCs of the federation to minimize the cost of energy consumption.

Motivation and contributions This research is motivated by the following observations. First, data-intensive applications require a large amount of resources that could span across multiple DCs, and in this case exploiting federation is a good choice for small CPs. However, existing cloud management approaches do not consider the semi-autonomous requirements of the federated system. Second, CPs are mainly interested in maximizing their profit and minimizing their operating costs. Energy consumption contributes a significant fraction of CPs' operating costs, and it could be reduced by taking advantage of the electricity price diversity across the DCs. In addition, data-intensive applications may trigger large data transfer over WAN, wherein the cost of communication could surpass the cost of the energy consumption. This requires maintaining data locality to minimize inter-DC traffic by carefully allocating highly communicating virtual components. Third, VM placement usually consider the allocation of homogeneous VMs in isolation, which is not suitable when heterogeneous VMs collaborate in an application.

To address these issues, we propose a heuristic algorithm for minimizing the operating cost of big data applications in a federated cloud and showed preliminary results to demonstrate the benefits of federation [22]. The major contributions of this work are as follows:

- We propose a framework for resource management and workload distribution that satisfies the requirements of federated cloud systems, like semi-autonomy, privacy and scalability.
- We formulate the placement of collaborating virtual components of a data-intensive application as an optimization problem to minimize the TCO, that includes costs of energy consumption and inter-DC communication.
- We propose an algorithm for the allocation that clusters the virtual components of the application based on their communication requirements and maps these clusters to DCs of the federation to minimize the total operating cost of the CP that is hosting the application. Evaluation of the algorithm using real-world data shows the advantages of the proposal in a federated system compared to the traditional geo-distributed cloud.

The rest of this paper is organized as follows. Section 2 discusses the literature that addresses VM allocation as a constrained optimization problem. We present the

hierarchical framework for resource management and workload distribution for federated clouds in Sect. 3. In Sect. 4, we present the optimization model and the algorithm to solve it is presented in Sect. 5. Simulation results are discussed in Sect. 6 followed by the conclusions in Sect. 7.

2 Related Work

2.1 VM Placement in a Single DC

The problem of VM allocation is mostly explored for the data center at a single site with minimizing the energy consumption as an objective [23]. VM consolidation is widely used to allocate VMs on a minimum number of physical servers and switch off idle servers [24, 25]. In [26], VMs that have a complementary temporal traffic requirements are packed in a group. Then, VMs of the same application or the same group are placed in nearby servers to maximize the resource utilization and minimize the number of active servers. In [27], VMs of a virtual cluster are classified into two groups; one group where VMs are always busy, and another group where VMs may change to sleep mode. Accordingly, tasks are migrated between these two groups for trading off the energy saving and the response time. All these works aim to reduce the energy consumption within a single DC, where there is no diversity in the electricity price, and the network bandwidth is not a big issue.

2.2 VM Placement in Large-Scale DCs

In context of large-scale DCs, the problem becomes interesting because of spatial variation in the electricity price across DCs. Kumar et al., in [28], proposed a meta-heuristic approach for VM scheduling in a large-scale DCs with the objective of optimizing energy consumption, time, and execution cost of heterogeneous tasks. However, authors did not consider the energy cost variation. The work in [29], considered distributed DCs powered by different energy sources with different prices. The problem of VM allocation was modeled with an objective of reducing energy cost and carbon taxes. In [30], an algorithm was proposed to reduce the total energy cost in a federated cloud platform by migrating VMs to a DC where the electricity price is lower while considering migration cost. However, this work did not study large data migration. In [20], the authors proposed two algorithms for initial VM allocation, and for VM migration in distributed cloud DCs to minimize the energy cost. They also reported the trade-off between load balancing and energy cost. However, they did not consider inter-DC communication, and the cost of VM migration. Note that inter-DC VM migration is a costly operation [31], especially for data-intensive applications where a large amount of data needs to be migrated [20, 32]. The work in [20] proposed an algorithm for VM allocation using a simple cost of running workload in the different sites of the distributed data center. The algorithm selects the data center with the lowest energy cost. However, it completely ignored the communication between VMs which affects the operating cost of the cloud

provider. The approach in [20] requires communication between all the data centers, while the information is collected in a hierarchical manner with our proposal. If the number of cloud providers and data centers in the federation are L and M , respectively, then, the number of messages for the work in [20] would be M^2 whereas it would be only $L + M$ ($L \ll M$) in our architecture. In our proposal, the status messages are sent only on demand based on request/reply protocol, whereas they are exchanged periodically in [20], thereby increasing the communication overhead. On the whole, the literature focuses on the energy consumption cost of independent VMs without considering the traffic demand between them and the cost of inter-DC communication, that would largely affect the operating cost of a CP in federated platform, and the performance of user applications [33].

2.3 Placement of Data-Intensive Applications in a Single DC

Placement of virtual components (data blocks and compute VMs) for data-intensive applications should consider the traffic demand between the components of the application. To deal with this issue, Zhang et al., proposed a VM placement algorithm for data-intensive applications in a single DC to minimize the data access latency [34]. They assumed that the placement of data nodes is given and allocate only the compute VMs. However, the allocation of both data blocks and compute VMs for data-intensive applications should be handled during the application deployment. Ferdaus et al., addressed the problem of traffic-aware placement of multi-tier applications in a single DC, considering the communication pattern between the virtual components [35]. A greedy heuristic was proposed to map the virtual links between the components to physical links, while satisfying the bandwidth demand and minimizing the network distance. It minimizes the network traffic within a DC, without considering resource utilization and energy consumption cost.

2.4 Placement of Data-Intensive Applications in Large-Scale DCs

There are a few works that addressed the problem of deploying data-intensive applications in a distributed cloud environment. The work in [36] addressed the placement of data blocks and compute VMs in distributed DCs. They proposed a meta-heuristic approach based on Ant Colony Optimization (ACO) to minimize the network distance by selecting a subset of servers that are physically closer for VM allocation. They assumed that all the VMs have the same configuration and that each server can accommodate the same number of VMs. Gu et al. addressed big data processing in geo-distributed DCs, considering data and task placement along with data transfer [37]. The problem has been formulated as MILP problem with the objective of minimizing the number of active servers and communication cost. However, it has been solved using Gurobi optimization solver which is time consuming and not feasible for large-scale problems. Furthermore, they assumed that all the DCs have equal number of servers and allocated the VMs without considering cost variation across the different DCs. Xiao et al., studied the distributed deployment of MapReduce workload across geo-distributed DCs to minimize costs and latency [38]. The

problem was divided into three sub-problems including data loading, VM provisioning and reducer selection. Lyapunov optimization was applied for solving these sub-problems independently. The allocation of data blocks and compute VMs was done separately, and the proposed solution is useful only for MapReduce applications. In [39], the authors proposed an algorithm that schedules data and associated tasks in the same DC to reduce the latency and the data transfer across the DCs, while achieving load balancing. However, they focused only on reducing inter-DC network latency to improve the performance of data-intensive applications, while they did not consider the cost-efficiency of resource allocation.

In the literature, most of the works focused on either minimizing the energy consumption or the network traffic. We argue that deploying a data-intensive application in a federated cloud requires that both these factors should be considered to minimize the total operating cost of the CP.

2.5 Design of Resource Management Framework

Cloud systems are mostly managed using centralized approaches [40]. In [29], it was assumed that the CP consists of several DCs with the information in a central database and the cloud broker performs the allocation based on this information. However, this requires a centralized view of all the resource utilization, which is not compatible with a federated system, where the autonomy of different CPs for internal management makes it difficult to maintain a central database. Similarly, in [41], authors introduced a central third party that is responsible for resource management and the communication with the CPs in the federation. Again, this model suffers from the problem of single point of failure where the entire federation fails if this central manager fails. In [20], a distributed approach for workload management across multiple DCs in different geographical areas was proposed. Every data center has a manager and all the managers exchange information for dynamic VM allocation. Though the idea of resource management framework is similar to our work, we note that in a federation, each cloud provider is autonomous with its own set of objectives for resource allocation. Hence, we added another layer to the management platform and adapted it to be compatible with the federated environment. The Federated Cloud Manager (FCM) in our proposal takes care of the privacy and semi-autonomy requirements of the cloud providers. Hence the information exchange is only done between FCMs (fewer number compared to DCMs).

From the literature survey, we observed that the VM allocation was usually handled in a centralized manner mostly in a single DC or on a limited scale. The semi-autonomous nature of federated cloud environment warrants that a CP cannot directly access the resource availability in the other CPs. Therefore, we design a hierarchical resource management and allocation framework for federated cloud systems, wherein we address the problem of cost-aware deployment of data-intensive applications, taking into account both energy and communication costs. Based on the proposed architecture, we design an algorithm to allocate clusters of virtual components, instead of individual VMs, considering the inter-component traffic

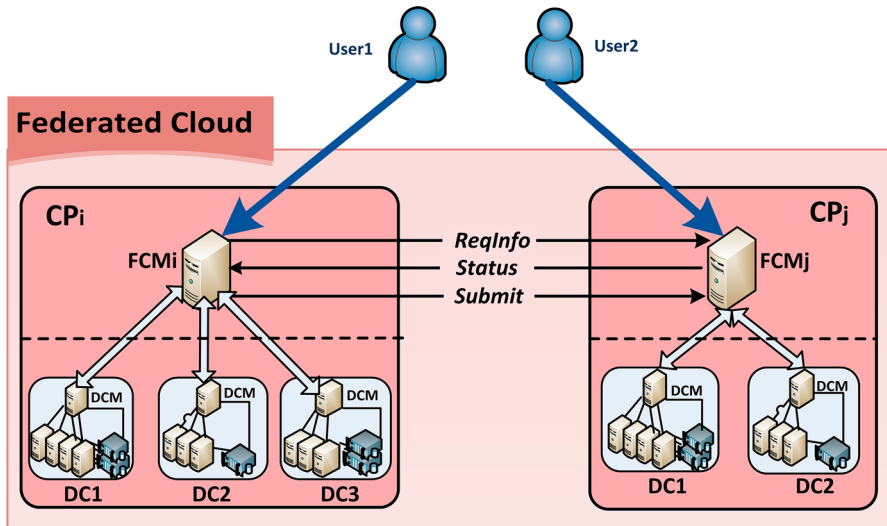


Fig. 1 Proposed hierarchical management framework for federated cloud system with multiple cloud providers where each cloud provider (CP) has a federated cloud manager (FCM) and each data center has a data center manager (DCM). On the federation level, the communication happens between FCMs only

pattern and electricity cost variation across the different DCs in the federation to minimize the total operating cost.

3 Hierarchical Management Framework for Federated Clouds

In this section, we propose a framework for resource allocation that is compatible with the requirements of a federated cloud system. Each CP is considered to be autonomous and has its own objectives regarding resource allocation, like cost minimization, latency reduction or load balancing [40]. It may also have different allocation strategies, insourcing and outsourcing strategies for renting resources from other CPs in the federation. All this needs to be done transparently without affecting the resource allocation policy. The proposed framework minimizes the information exchange required for workload distribution and allows each CP in a federation not to reveal the resource allocation strategy followed.

The federated cloud system considered here consists of multiple CPs, with each CP having a different number of DCs [10]. As illustrated in Fig. 1, the proposed framework uses two different layers for each CP in the federation. The components in the framework are: (i) a Data Center Manager (DCM) which serves as a local management layer for each DC of a CP in the federation, and (ii) a Federated Cloud Manager (FCM) which serves as a federation management layer, that is implemented by each CP in the federation.

The DCM in each DC is responsible for managing the resources and the workload distribution within the DC. It communicates with the FCM of its CP to handle the local DC resource allocation. The functions of the DCM include: (i) sending

updated information on the DC's status to the FCM, (ii) receiving the workload requests from its CP's FCM, and (iii) placement process inside the DC executing a specific consolidation algorithm. We do not discuss the placement algorithm at this level, as it is well studied in the literature.

The FCM of each CP in the federated cloud is responsible for communicating with (i) FCMs of other CPs in the federation, (ii) DCMs of the DCs that are operated by its CP, and (iii) the end users of the associated CP as well. Accordingly, FCM acts as an interface for its CP's users. It receives resource allocation requests and assigns the required resources from the DCs of its CP or from the DCs of other CPs in the federation, that is transparent to the end user.

Consider a federated cloud with two CPs as illustrated in Fig. 1. FCM_i and FCM_j are the federated cloud managers of CP_i and CP_j , respectively. Three messages are used in the information exchange among the two FCMs as defined below.

- (i) *ReqInfo*: FCM_i requests the status information of CP_j 's data centers from FCM_j . Since any FCM in the federation could allocate the DC resources, an FCM should request the status information of a specific DC whenever it needs to assign some workload to it. This can be done by sending *ReqInfo* message with a specific DC identifier.
- (ii) *Status*: This is a reply message from FCM_j to FCM_i , triggered in response to a *ReqInfo* message with the current status information, say $\{Status_{j_1}, Status_{j_2}, \dots, Status_{j_n}\}$ of the data centers $\{DC_{j_1}, DC_{j_2}, \dots, DC_{j_n}\}$ of the CP_j .
- (iii) *Submit*: It is initiated by FCM_i while assigning some workload to a DC of CP_j .

These messages can be implemented using Simple Network Management Protocol (SNMP). The exact content of the status information exchanged between FCMs depends on the agreement between the CPs in a federation, but typically it should provide the minimum information required for the allocation algorithm. To suit the proposed algorithm in this work, we define the format of the status message as:

$$Status_{jk} = \{DC_{jk}, Cpu_{jk}, Mem_{jk}, Str_{jk}, E_{jk}\} \quad (1)$$

where DC_{jk} is the identifier of the k th data center of the j th cloud provider, Cpu_{jk} is the number of CPU cores available in DC_{jk} , Mem_{jk} and Str_{jk} are the amount of memory and storage available in DC_{jk} , respectively, and E_{jk} is the electricity price at DC_{jk} .

On receiving a request for resource allocation, the FCM collects the status of all DCs across different CPs in the federation, by contacting the DCMs (for its own DCs) and the FCMs (for other CPs). After receiving the status information, the FCM coordinates with other FCMs to distribute the workload efficiently considering the resource availability and the operating cost.

The proposed framework can be implemented without changing the architecture of the existing cloud platform. As the framework is hierarchical, it can be extended incrementally as more CPs join the federation. Since the status messages are sent only on demand, the framework minimizes the communication

overhead. Additionally, the framework is stateless, since it uses request/reply paradigm and there is no state stored at the FCM. If an FCM fails, the CP can simply create a new FCM without any state transfer. The framework gives great flexibility to CPs allowing them to change their resource allocation and workload distribution policies autonomously and transparently.

4 Problem Formulation

In this section, we discuss the proposed optimization model for cost-aware placement of data-intensive applications in federated cloud DCs. Table 1 summarizes the notation used. First, we state the system model and the assumptions made, before presenting the optimization model.

4.1 Assumptions

- We consider the initial placement of virtual components assuming that the resource requirement of an application is less than the total available capacity of the system.
- The requirements of virtual components and the capacity of the DCs are heterogeneous.
- The bandwidth cost within a DC is negligible, and there is no restriction on the bandwidth usage. The inter-DC bandwidth cost depends on the location [39, 42].
- The cost of a resource used is based on its energy consumption. We use the electricity price at the location of a DC as the pricing factor for CPs [37].
- FCMs exchange information that includes the DC identifier, the available resources in the DC, PUE, and the current electricity price (see Eq. 1).

4.2 System Model

Let \mathbb{P} denote the set of collaborating CPs in a federated cloud. Each cloud provider CP_l in the federation has a set of data centers \mathcal{D}_l and an FCM F_l . Let $\mathbb{F} = \{F_l, l \in \mathbb{P}\}$ be the set of all FCMs and $\mathbb{D} = \{\mathcal{D}_l, l \in \mathbb{P}\}$ be the set of all DCs in the federated cloud. Each data center in the federation $\mathcal{D}_{lk} \in \mathbb{D}$ is characterized by its CPU capacity Cpu_{lk} , memory capacity Mem_{lk} , storage capacity Str_{lk} , power usage effectiveness PUE_{lk} , and its operating cost in terms of electricity price E_{lk} . We assume that the supply of these resources is controlled and can be changed by the CP. Accordingly, an FCM, say F_l has a freedom to announce the resource capacity available for sharing at each of its data centers, say \mathcal{D}_{lk} . We define the network bandwidth between data centers by a matrix $BW_{|\mathbb{D}| \times |\mathbb{D}|}$, where $BW_{lk,nh}$ represents the bandwidth of the link between a data center \mathcal{D}_{lk} and another data center \mathcal{D}_{nh} .

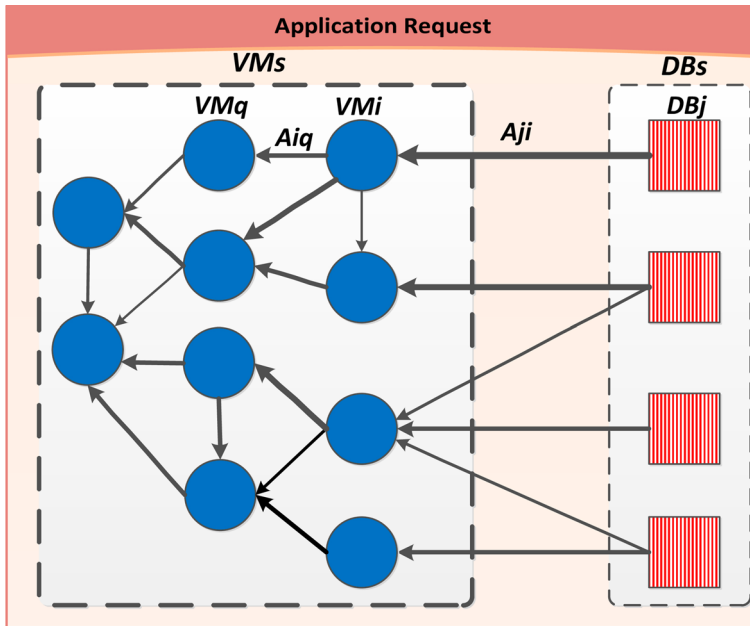


Fig. 2 Graph representation of a data-intensive application request where red squares represent data blocks (DBs) and blue circles represent compute VMs. The thickness of an edge reflects the variation in traffic demand between the virtual components (Color figure online)

4.2.1 User Request

A CP receives a request for multiple virtual components. Let \mathbb{C} be the set of all virtual components needed by the application, which is a union of two subsets; \mathbb{M} , the set of compute virtual machines and \mathbb{B} , the set of data blocks. Each virtual machine $VM_i \in \mathbb{M}$ demands a number of CPU cores, VM_i^c and memory, VM_i^m (per GB), while each data block $DB_j \in \mathbb{B}$ comes only with a storage demand, DB_j^s (per GB).

The application request is represented as a weighted directed graph $G(\mathbb{C}, W)$, where \mathbb{C} is the set of virtual components, and W is the set of weighted edges that represent the traffic demand between the virtual components. Figure 2 depicts a typical application request with squares indicating the data blocks and circles indicating the compute VMs. A weighted edge is directed from a source VM (or DB) to a destination VM, and its thickness reflects the variation in the traffic demand. Data-intensive applications are usually characterized by high traffic between DBs and VMs, low traffic among VMs and no traffic between DBs [39]. We define \mathbb{C} as a multi-dimensional vector, where each dimension represents a type of resource; CPU, memory and storage. We represent the traffic demand by an adjacency matrix $A_{|\mathbb{C}| \times |\mathbb{M}|}$, where A_{ji} is the data volume between DB_j (or VM_j) and VM_i . We define also a binary variable x_{lk}^i to indicate whether a virtual component $VC_i \in \mathbb{C}$ is placed in the data center \mathcal{D}_{lk} ($x_{lk}^i = 1$) or not ($x_{lk}^i = 0$).

Table 1 Notation used in the model

Symbol	Description
\mathbb{P}	Set of cloud providers in the federation
\mathbb{F}	Set of federated cloud managers in the federation
\mathbb{D}	Set of data centers in the federation
\mathcal{D}_l	Set of data centers in the l th cloud provider
\mathcal{D}_{lk}	The k th data center in the l th cloud provider
Cpu_{lk}	CPU capacity of data center \mathcal{D}_{lk}
Mem_{lk}	Memory capacity of data center \mathcal{D}_{lk}
Str_{lk}	Storage capacity of data center \mathcal{D}_{lk}
E_{lk}	Electricity price at data center \mathcal{D}_{lk} (in \$/KWh)
PUE_{lk}	Power usage effectiveness at data center \mathcal{D}_{lk}
m_{lk}	The number of servers at data center \mathcal{D}_{lk}
P_{lk}	Power consumption of data center \mathcal{D}_{lk} (KWh)
$d_{lk,nh}$	Cost of data transfer between \mathcal{D}_{lk} and \mathcal{D}_{nh} (\$/GB)
$BW_{lk,nh}$	The bandwidth of the link between \mathcal{D}_{lk} and \mathcal{D}_{nh} (Gbps)
\mathbb{C}	Set of virtual components
\mathbb{M}	Set of virtual machines
\mathbb{B}	Set of data blocks
VM_i^c	CPU demand of i th virtual machine
VM_i^m	Memory demand of i th virtual machine
DB_j^s	Storage demand of j th data block
x_{lk}^i	Virtual component i assigned to data center \mathcal{D}_{lk}
A_{ji}	Traffic demand between two components j and i (GB/h)
SP_v	Power consumed by server v (in KWh)
BP_v	Base power consumed by server v when idle
FP_v^r	Power consumed by resource r in server v at full utilization
U_v^r	Utilization of resource r in server v
C_v^r	Capacity of resource r in server v
y_v	binary variable that indicates if server v is used or not
t	The running time of the application (in h)

4.2.2 Power Consumption

Though power consumption is usually modeled as a function of only the CPU utilization, it was noted that the storage may consume up to 40% of the total energy of a data center for data-intensive applications [43]. Accordingly, we consider CPU and storage to consume a significant portion of power. The total power consumed by a server v is expressed as [44]

$$SP_v = BP_v + \sum_r FP_v^r \times \frac{U_v^r}{C_v^r} \tag{2}$$

where BP_v is the base power consumption of a server v , and FP_v^r is the power consumed by a resource r of a server v at its full utilization. $r \in \{CPU, Storage\}$ is the

concerned resource, U_v^r and C_v^r are the utilization and maximum capacity of the resource r in a server v . The power consumption of an entire data center \mathcal{D}_{lk} also includes the power consumed by the other data center facilities, such as cooling system, that might account for 50% of the total energy based on the location [43]. We define the total power consumed (KWh) at a data center \mathcal{D}_{lk} housing m_{lk} servers as

$$P_{lk} = \sum_{v=1}^{m_{lk}} SP_v \times y_v \times PUE_{lk} \tag{3}$$

where y_v is a binary variable set to unity when the server v is being used and PUE_{lk} is the power usage effectiveness of a data center \mathcal{D}_{lk} .

4.3 Optimization Model

Next, we define the cost components considered in the formulation of the optimization problem.

4.3.1 Energy Cost

The cost incurred by power at a data center \mathcal{D}_{lk} can be obtained by multiplying power usage P_{lk} of \mathcal{D}_{lk} (Eq. 3) with the electricity price E_{lk} (\$/KWh). The energy cost incurred by the federation per hour (\$/h) is defined as

$$EC = \sum_{\mathcal{D}_{lk} \in \mathbb{D}} P_{lk} E_{lk} \tag{4}$$

4.3.2 Communication Cost

Comparing to inter-DC bandwidth cost, the communication cost is negligible for virtual components within a DC [39]. We define the communication cost due to data-intensive applications deployed within the federation CC as

$$CC = \sum_{\mathcal{D}_{lk} \in \mathbb{D}} \sum_{\substack{\mathcal{D}_{nh} \in \mathbb{D} \\ \mathcal{D}_{lk} \neq \mathcal{D}_{nh}}} \sum_{i \in \mathbb{C}} \sum_{j \in \mathbb{M}} x_{lk}^i x_{nh}^j A_{ij} d_{lk,nh} \tag{5}$$

where A_{ij} is the data transfer requirement between VC_i and VM_j (in GB/h) and $d_{lk,nh}$ is the data transfer cost between \mathcal{D}_{lk} and \mathcal{D}_{nh} (\$/GB).

Using all the cost factors considered above, we formulate the optimization model as

$$\text{Minimize } (EC + CC) t \tag{6}$$

subject to

$$\sum_{\mathcal{D}_{lk} \in \mathbb{D}} x_{lk}^i = 1, \forall i \in \mathbb{C} \quad (7)$$

$$\sum_{j \in \mathbb{M}} x_{lk}^j VM_j^c \leq Cpu_{lk}, \forall \mathcal{D}_{lk} \in \mathbb{D} \quad (8)$$

$$\sum_{j \in \mathbb{M}} x_{lk}^j VM_j^m \leq Mem_{lk}, \forall \mathcal{D}_{lk} \in \mathbb{D} \quad (9)$$

$$\sum_{b \in \mathbb{B}} x_{lk}^b DB_b^s \leq Str_{lk}, \forall \mathcal{D}_{lk} \in \mathbb{D} \quad (10)$$

$$\sum_{i \in \mathbb{C}} \sum_{j \in \mathbb{M}} x_{lk}^i x_{nh}^j A_{ij} \leq BW_{lk,nh} \forall \mathcal{D}_{lk}, \mathcal{D}_{nh} \in \mathbb{D} \quad (11)$$

$$x_{lk}^i \in \{0, 1\}, \forall i \in \mathbb{C}, \forall j \in \mathbb{M}, \forall b \in \mathbb{B}, \forall \mathcal{D}_{lk} \in \mathbb{D} \quad (12)$$

Equation 6 minimizes the cost of both the energy consumption and the communication for time t (h); since we consider *pay-as-you-use* pricing model. The constraint in Eq. 7 makes sure that every virtual component is assigned to one and only one DC. Constraints in Eqs. 8 and 9 describe that the CPU and the memory used by all VMs allocated to a data center \mathcal{D}_{lk} should not exceed its CPU and memory capacities, respectively. In the same way, Eq. 10 shows that the storage used by all DBs allocated to a data center \mathcal{D}_{lk} should not exceed its storage capacity. Equation 11 represents the limited bandwidth constraint. It describes that the cumulative traffic requirement between the set of virtual components placed at a data center \mathcal{D}_{lk} and the set of virtual components placed at another data center \mathcal{D}_{nh} cannot exceed the bandwidth of the link between these two data centers. Equation 12 defines the domain of variables of the problem.

This problem is a variant of the vector bin packing problem where the items (i.e., VMs) are packed into bins (i.e., data centers), such that the vector sum of the items received by any bin does not exceed the bin's (resource) limit. The vector bin packing problem and its variants are NP-hard problems [45, 46]. Therefore the problem considered in this paper is NP-hard [7]. In the next section, we propose a heuristic to find a near-optimal solution.

5 Proposed Algorithm

In this section, we propose a cost-aware algorithm for allocating virtual components of a data-intensive application in a federated cloud. It minimizes the total operating cost by taking advantage of the electricity price diversity across various DCs in a federation to minimize the cost of energy consumption. It assigns correlated

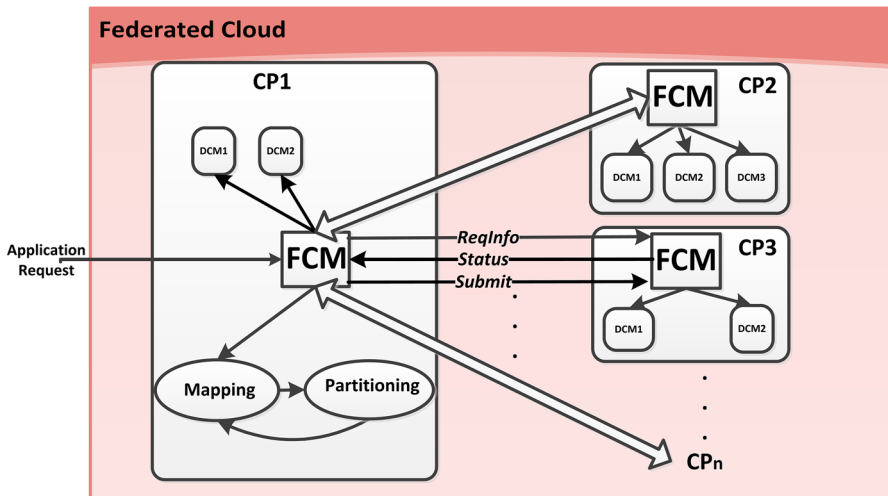


Fig. 3 The Data Flow Diagram of the proposed algorithm which allocates the virtual components of a data-intensive application, in two phases; partitioning and mapping. The algorithm is executed by the Federated Cloud Manager (FCM) of the Cloud Provider (CP) which provides the service of data-intensive applications deployment to users (e.g. CP_1), after receiving an application request

virtual components to the same DC, reducing the WAN traffic. In most of the real-world applications a group of VMs communicate to perform a specific task, where intra-task communication is higher than inter-task communication. Based on this, the algorithm considers the traffic pattern of the application and allocates clusters of virtual components instead of individual components. This algorithm runs in two phases; *Partitioning* and *Mapping*, as illustrated in Fig. 3. *Partitioning* phase partitions the request graph, shown in Fig. 2, to clusters based on the traffic density between the nodes, that reduces inter-DC communication and bandwidth cost. The *Mapping* phase allocates clusters of virtual components to a DC with a minimum energy cost to minimize the operating cost. The algorithm is executed by the federated cloud manager that receives the application request.

Algorithm 1 lists the steps in the proposed heuristic, named **Cost-Aware Virtual Components Allocation (CAVCA)**. Without the loss of generality, let us consider that the cloud provider $CP_n \in \mathbb{P}$ is providing the service of data-intensive applications deployment to users, and its FCM F_n runs the algorithm to allocate the required resources for an application request. The algorithm takes as an input \mathcal{D}_n ; the set of local data centers, \mathbb{F} ; the set of FCMs of CPs in the federation, and G ; the graph of the application request. The algorithm returns *AllocationMap* as the output which includes the mapping between the virtual components and DCs of the federation, $\mathcal{D}_{lk} \in \mathbb{D}$.

In the initialization phase (Lines 1–6), *DCList* contains all the DCs in the federation, with only local data centers \mathcal{D}_n to start with. *Best_{DC}* is the DC with the lowest energy cost, but has resources for sharing. *Clist* has clusters of virtual components not assigned to any DC. *Best_{cluster}* is the cluster with the highest resource demand in *Clist* (the largest cluster). The resource demand of a cluster of virtual components is

the total demand across all virtual components of the cluster. *AllocationMap* contains pairs of assignment; (VC_i, \mathcal{D}_{lk}) , where a virtual component VC_i is assigned to a data center \mathcal{D}_{lk} .

Algorithm 1: CAVCA Cost-Aware Virtual Components Allocation

```

Input: List of local data centers  $\mathcal{D}_n$ 
         List of federated cloud managers  $\mathbb{F}$ 
         Application request graph  $G(C, W)$ 

Output: AllocationMap( $VC_i, \mathcal{D}_{lk}$ )

1 Initialization:
2    $DClst \leftarrow \mathcal{D}_n$ 
3    $Best_{DC} \leftarrow NULL$ 
4    $Clist \leftarrow NULL$ 
5    $Best_{cluster} \leftarrow NULL$ 
6    $AllocationMap \leftarrow NULL$ 

7 for each  $\mathcal{D}_{nk} \in \mathcal{D}_n$  do
8   | update Status of  $\mathcal{D}_{nk}$ 
9 end
10 for each  $F_l \in \mathbb{F}$  do
11 | Send ReqInfo to  $F_l$ 
12 | Receive Status from  $F_l$ 
13 | add received data centers to  $DClst$ 
14 end
15 Sort  $DClst$  according to  $E_{lk}$  in increasing order
16  $Best_{DC} \leftarrow DClst.head$ 
17  $Clist \leftarrow \{G\}$ 
18 while ( $Clist \neq NULL$ ) do
19 | Sort  $Clist$  according to capacity in decreasing order
20 |  $Best_{cluster} \leftarrow Clist.head$ 
21 | if ( $Best_{DC}.capacity > 0$ ) then
22 |   repeat
23 |   | Send ReqInfo( $Best_{DC}$ ) to FCM of  $Best_{DC}$ 
24 |   | receive Status from FCM of  $Best_{DC}$ 
25 |   | if  $Best_{DC}.capacity \geq Best_{cluster}.res$  then
26 |   |   | if ( $Comm(Best_{cluster}, \mathcal{D}_{lk}) \leq BW^{Best_{DC}, \mathcal{D}_{lk}} \forall \mathcal{D}_{lk} \in \mathbb{D}$ ) then
27 |   |   |   | Submit  $Best_{cluster}$  to FCM of  $Best_{DC}$ 
28 |   |   |   |  $Clist \leftarrow Clist - \{Best_{cluster}\}$ 
29 |   |   |   | for each  $VC_i \in Best_{cluster}$  do
30 |   |   |   | |  $AllocationMap.add(VC_i, Best_{DC})$ 
31 |   |   |   | end
32 |   |   |   |  $Best_{cluster} \leftarrow Best_{cluster}.next$ 
33 |   |   | else
34 |   |   |   |  $last_{cluster} \leftarrow Best_{cluster}$ 
35 |   |   |   |  $Best_{cluster} \leftarrow Best_{cluster}.next$ 
36 |   |   | until  $Best_{cluster} = NULL$ ;
37 |   |   |  $Clist \leftarrow Clist - \{last_{cluster}\}$ 
38 |   |   |  $Clist \leftarrow Clist \cup \text{Partitioning}(last_{cluster})$ 
39 |   | else
40 |   |   |  $Best_{DC} \leftarrow Best_{DC}.next$ 
41 |   | end
42 | end
43 return AllocationMap

```

When an FCM, say F_n gets an application request (as graph G), it contacts its DCMs to get the status (lines 7–8). It also contacts the other FCMs to collect information about the status of all DCs in the federation (lines 9–12). F_n combines all the received status messages and creates $DClst$. CAVCA sorts the DCs in ascending order with respect to the electricity price (line 13) and selects the $Best_{DC}$, having lowest electricity price, for

placing virtual components (line 14) to minimize the energy cost. Initially, the algorithm attempts to schedule the entire application to one DC to eliminate the bandwidth cost. Hence, it starts with the entire request graph G as one cluster (line 15). If it is not possible to do the same G is partitioned.

The algorithm executes the allocation process by mapping big clusters of correlated virtual components to the current $Best_{DC}$ and partitioning the smallest cluster, when no cluster from $Clist$ can be placed in $Best_{DC}$, by repeating the steps until all virtual components are placed (lines 16–36). First, CAVCA sorts the list of clusters $Clist$ in descending order according to resource demand and takes the largest cluster with the highest resource demand for allocation (lines 17–18). Note that DCs in the federated cloud are used by multiple CPs simultaneously. Thus, F_n checks the resource availability in a data center \mathcal{D}_{lk} from the corresponding FCM, whenever it needs to assign some workload to \mathcal{D}_{lk} , and that is before submitting the allocation request to that FCM (lines 21–22). CAVCA scans the sorted $Clist$ in order and assigns to $Best_{DC}$, the clusters that can be placed there based on the resource availability (lines 23–32). The algorithm checks the availability of CPU, memory, and storage resources in $Best_{DC}$ (line 23). It also checks the availability of the required network bandwidth of $Best_{cluster}$ (line 24). It calculates the sum of the edges between the virtual components of $Best_{cluster}$ and the virtual components that are already placed in a DC (e.g. \mathcal{D}_{lk}), and compares this sum with the available bandwidth between $Best_{DC}$ and \mathcal{D}_{lk} to satisfy the traffic requirements of the virtual components of the cluster $Best_{cluster}$. In case, $Best_{DC}$ has available resources but no cluster in $Clist$ can fit into it, then F_n calls *Partitioning* function (Algorithm 2) to partition the smallest cluster, the last one in $Clist$ (lines 33–34). If the current $Best_{DC}$ does not have free resources, F_n selects the next one based on the electricity price (line 36). This process is repeated till all the clusters are placed to get the allocation map (line 37).

Algorithm 2 lists the steps of *Partitioning* phase, that takes as input c , the cluster which need to be partitioned and a weight range δ for edges to be removed in each iteration, and it returns smaller clusters. The algorithm iteratively removes (lines 5–8) from C all edges with weight less than or equal to a threshold, till C becomes disconnected ($n_c > 1$). If the range δ is very small, the number of iterations in *Partitioning* becomes large. On the other hand, if δ is large, then the algorithm would remove huge number of edges and the number of clusters would be large. We chose δ based on the variation in traffic demand between virtual components. We normalize the traffic matrix A , such that the traffic demand between virtual components is in the range $[0,1]$. We define δ , the range of edges to be removed in each iteration as 0.1. This algorithm minimizes inter-DC communication by containing traffic inside a cluster and minimizing the traffic between clusters.

Algorithm 2: Partitioning

Input: C Cluster of virtual components
 δ range of edge's weight to be removed
Output: $UpdatedC$ the cluster after removing edges

```

1 Initialization:
2    $UpdatedC \leftarrow C$ 
3    $n_c \leftarrow 1$ 
4    $\sigma \leftarrow \delta$ 
5 while ( $n_c == 1$ ) do
6   |  $UpdatedC =$  Remove all edges with ( $weight \leq \sigma$ )
7   |  $\sigma \leftarrow \sigma + \delta$ 
8   |  $n_c =$  Number of clusters in  $UpdatedC$ 
9 end
9 return  $UpdatedC$ 

```

5.1 Complexity Analysis

Let the number of data centers in the federation be M and the number of virtual components in the request be N . The size of traffic matrix A is N^2 . The algorithm sorts the list of data centers once in $mathcal{O}(M \log(M))$ time. Then, it divides the request graph containing N virtual components recursively to create clusters where every time *Partitioning* needs to scan the matrix A . Accordingly, the complexity of the graph partitioning is $mathcal{O}(N^2 \log(N))$. As ($N \gg M$), the overall running time of the algorithm is in $mathcal{O}(N^2 \log(N))$. Regarding required memory, the algorithm needs to store DC_{list} , the list of data centers in the federation, and the request graph which includes the set of virtual components and the traffic matrix. Therefore, the space complexity of the algorithm is in $mathcal{O}(M + N + N^2)$; that is in $mathcal{O}(N^2)$.

5.2 Cost of Running the Algorithm

It is very important to ensure that the cost of running the algorithm does not exceed the amount of saving in the cost of hosting the data-intensive application. The proposed algorithm collects the status information of all data centers of the federation. Then, it partitions the application request graph to clusters of correlated virtual components and maps the created clusters to DCs to minimize the operating cost. Therefore, the cost of running the algorithm can be modeled as two components; (i) the cost of data transfer due to exchange of status information messages, and(ii) the cost of computation to find the best placement.

5.2.1 Cost of Communication

First, the algorithm requires the status of all data centers in a federation with L cloud providers, where each FCM sends the status information of its data centers, which

involves $M + L$ number of messages. During the placement of the clusters, checking the resource availability in each data center requires on an average $\log(N)$ messages. Accordingly, the total number of messages required to run the algorithm is $M + L + \log(N)$.

5.2.2 Cost of Computation

The algorithm sorts the list of data centers which requires $M \log(M)$ operations. The request graph containing N virtual components is divided recursively to create clusters. *Partitioning* requires $N^2 \log(N)$ instructions. Hence, the total number of instructions is $M \log(M) + N^2 \log(N)$.

Therefore, we define the total cost of running the algorithm to be

$$(M + L + \log(N)) \text{Status}_{size} \overline{d_{lk,nh}} + \frac{M \log(M) + N^2 \log(N)}{VM_{freq} VM_{cores}} VM_{price} \quad (13)$$

where Status_{size} is the size of a *Status* message defined in the manuscript which is not more than 20 Bytes. $\overline{d_{lk,nh}}$ is the average data transfer price between data centers. VM_{cores} , VM_{freq} , and VM_{price} are the processing rate of a virtual core, the number of virtual cores, and the price of the virtual machine that is executing the algorithm, respectively.

For instance, assume a federated cloud system with 100 data centers and a very large application request with about 10,000 virtual components, the number of messages will not exceed 1000 messages and the data transferred due to status exchange is less than 1 MB with a cost less than 1\$. Also, the computing time using a VM with 2 cores and a frequency of 2.5 GHZ (i.e., D2d v4 Azure VM) will be in order of minutes (less than five minutes). As the cost of such a VM is not more than 0.120 \$/hour, the cost of computing part of the algorithm will be in terms of cents. We notice that the cost of running the algorithm is negligible if we compare it with the cost of running a data-intensive application which may run for days or months.

6 Numerical Results

In this section, we present the results of evaluating the proposed algorithm (abbreviated as CAVCA) in different scenarios. The proposed algorithm was implemented using MATLAB Production Server R2015a on a server with Intel Xeon processor and 128 GB RAM, running Ubuntu 14.04 64-bit OS.

6.1 Experimental Setup

Based on practical cloud deployments and DC locations, we implemented a federated cloud system with three CPs having eight DCs distributed across three regions as illustrated in Fig. 4. Each DC is labeled with a tuple indicating the CP, the

Federated Cloud

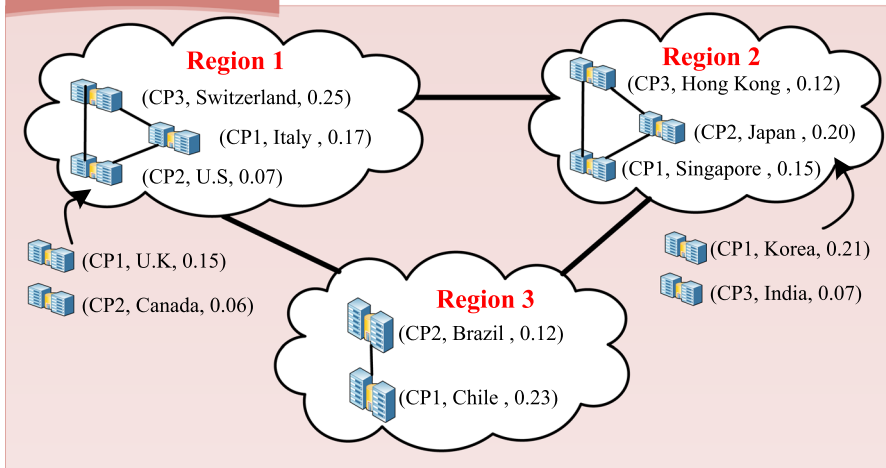


Fig. 4 Simulation setup of a federated cloud system with three cloud providers having data centers distributed in three regions. Each DC is labeled with a tuple indicating the cloud provider, the country of the data center, and electricity price in that country (in \$/KWh). Initially, only data centers inside the clouds are considered, while other data centers (outside the clouds) are added later on to study the impact of number of data centers in the federation

location, and electricity price (in \$/KWh).⁴ The locations were based on those used by Google, Microsoft and Amazon cloud providers.⁵ The number of servers at each location is chosen at random in the range of [150–250]. All the servers have a maximum configuration of 16 cores, 32 GB RAM and 2 TB disk, but a CP can announce varied resource availability for sharing. The base power of a server assumed is 100 W and the peak power of CPU and disk were 70 W and 16 W, respectively [44]. The bandwidth between data centers is in the range [40–100 Gbps] [47]. The data transfer cost is set to 0.01 \$/GB within a region and 0.08 \$/GB across regions [48].

6.1.1 Application Request

Based on real data-intensive application instances generated by Pegasus [49], We evaluate the proposed approach with application requests of various size. The volume of data blocks is taken in the range [0–100] GB. The number of CPU cores and the size of RAM required by compute VMs are chosen from {2, 4, 8} and {4, 8, 16} GB, respectively. The traffic demand between *DBs* and *VMs* is uniformly distributed in the range [0–2] GB/h and the traffic among *VMs* in the range [0–0.5] GB/h.

⁴ https://www.energy.eu/electricity_natural-gas_prices_european_union, <https://www.eia.gov/>.

⁵ <https://www.google.com/about/datacenters/inside/locations>, <https://azure.microsoft.com/en-in/global-infrastructure/>, <https://aws.amazon.com/about-aws/global-infrastructure>.

6.2 Results

To show the advantages of considering the operating cost in the placement of virtual components, we also implemented three allocation approaches used commonly in cloud management platforms, such as Eucalyptus and OpenStack; first fit (FF), best fit (BF), and worst fit (WF) [50, 51]. FF scans data centers in order of resource capacity and allocates a virtual component VC_i to the first DC that can accommodate it, while BF and WF allocate VC_i to the DC that contains smallest and largest amount of resources available, respectively. For fairness, we used First Fit Cost-Aware (FF-CA) approach to allocate VC_i to DCs according to lowest electricity price [20]. In all the experiments, we consider a time duration of 24 h for calculating the cost. Each experiment was repeated 1000 times and the 95% confidence intervals are shown in the plots. All the results were normalized to the maximum value.

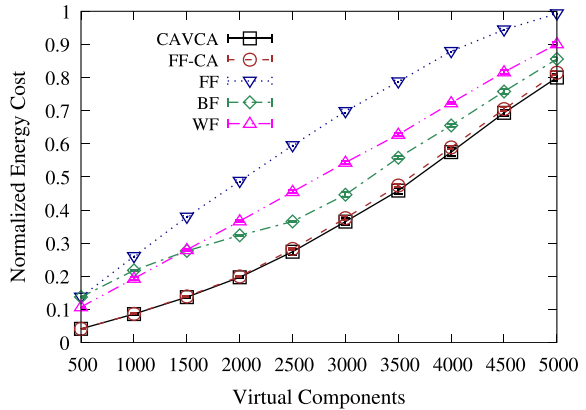
6.2.1 Impact of Application Request Size

To understand the impact of the number of virtual components on the cost, we evaluated CAVCA and other baseline approaches using the setup shown in Fig. 4, while varying the request size from 500 to 5000 virtual components. We assume a uniform PUE value of 1.2 for all DCs,⁶ and $Best_{DC}$ in CAVCA is selected based on the electricity price. Figure 5 shows the impact of the request size on energy cost, communication cost, and the total cost. As expected, the energy cost for all approaches increases with the size of the request. However, the energy cost with CAVCA and FF-CA are minimum, because they map requests to DCs operating with cheaper electricity, while the other approaches are oblivious to the electricity price variation. CAVCA gives the similar energy cost as FF-CA, which is the optimal one in terms of the energy cost. It can also be seen from Fig. 5b that, the communication cost also increases with the application request size for all the approaches. However, the communication cost is lowest with CAVCA since it creates and assigns clusters of correlated virtual components to a single DC. While CA-FF achieves the lowest energy cost, it fails to reduce the communication cost. Since CAVCA results in minimum energy cost and communication cost, the total cost is also the lowest as evident from Fig. 5c.

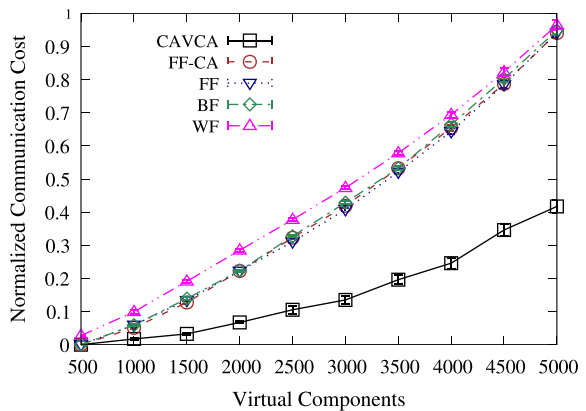
Figure 5a shows that, the reduction in cost with CAVCA decreases with increasing requests (in terms of virtual components). This is mainly because the choice gets limited as the capacity limit is reached. In other words, when an FCM receives a very big application request that requires almost all the resources offered by the federation, there is no option of selecting a cheaper DC. We conclude that CAVCA is more useful when DCs do not operate at their peak utilization, which often is the case.

⁶ <https://www.vertatique.com/no-one-can-agree-typical-pue>.

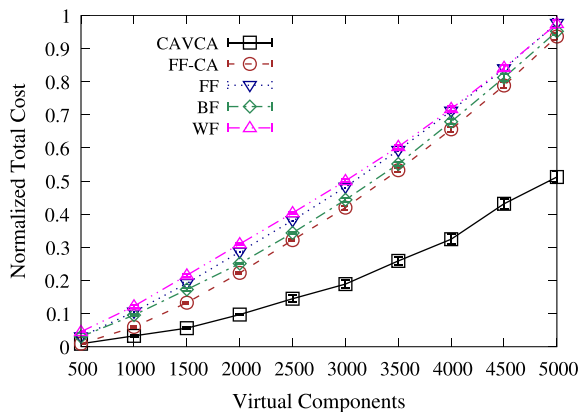
Fig. 5 Impact of application request size on the cost



(a) Energy cost



(b) Communication cost



(c) Total cost

6.2.2 Impact of PUE Variation

To understand the impact of PUE diversity, we repeat the previous experiment by varying the PUE across the DCs. PUE is randomly set between 1.1 and 2,⁷ and we use PUE to select the $Best_{DC}$ for placement along with the electricity price. Similar to the earlier case, the energy cost obtained with CAVCA is 14% lower than that with FF-CA, as seen from Fig. 6a. This is because, CAVCA selects DCs based on the product of electricity price and PUE, thereby reducing the total energy cost, and the energy consumption as well. Similarly, CAVCA also gives the minimum communication cost as seen from Fig. 6b. Consequently, the total cost is also lower as evident from Fig. 6c.

6.2.3 Impact of the Federation

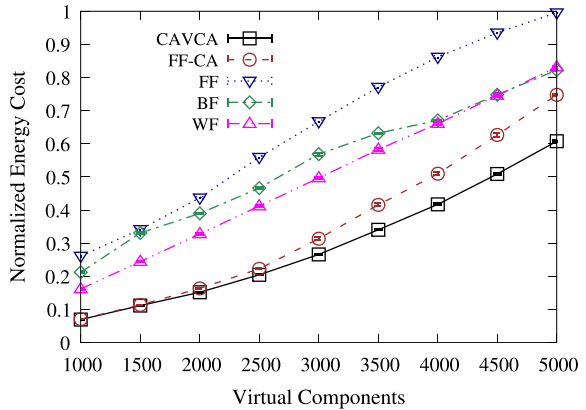
To understand the advantages of federation, we evaluated CAVCA for two scenarios; federated and non-federated cloud, based on the setup given in Fig. 4. We assume that F_1 (FCM of CP_1) receives application requests with size varying from 500 to 2000 (the maximum capacity of CP_1 's data centers). CP_1 can use only its own DCs in non federated scenario, while in federated system it can assign the workload to any DC that shares its resources. Therefore, the energy cost using federation is lower than that with the non-federated system as seen from the results in Fig. 7a. Regarding the communication cost, when a single DC is able to accommodate the entire application request, the communication cost is zero (for instance with request size of 500 virtual components in Fig. 7b. For larger request sizes, F_1 selects a nearby DC within the same region in the federated scenario, while it has to select a DC in different region in the non-federated scenario. This increases the communication cost as evident from Fig. 7d. Accordingly, the total cost obtained by the algorithm using federation is less than that in non federated system as depicted by Fig. 7c. Further, a CP can accept larger requests using federation as already shown in Fig. 5.

6.2.4 Impact of Varying Data Transfer Price

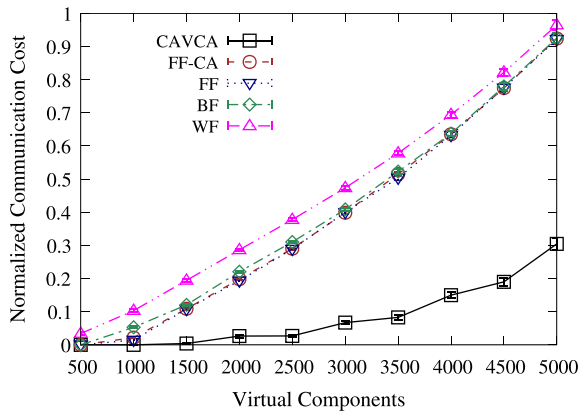
Since many CPs, such as Microsoft [42], use different prices for data transfer across the regions, we evaluated CAVCA and other methods considering this scenario. Since energy cost is not affected by the change in data transfer prices, we only report the communication cost and the total cost. The results in Fig. 8a show almost 65% reduction in the communication cost with CAVCA compared to the other baseline methods, because of clustering used in CAVCA to reduce the traffic between DCs. Consequently, there is a significant reduction in the total cost with CAVCA as evident from Fig. 8b. Similarly, Fig. 9 demonstrates the performance of CAVCA for federated and non-federated scenarios with varying data transfer price, showing that

⁷ <https://www.datacenterdynamics.com/news/apac-data-center-survey-reveals-high-pue-figures-across-the-region/>.

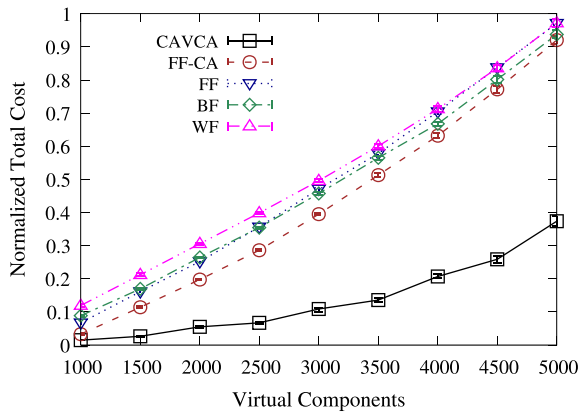
Fig. 6 Impact of application request size on the cost factors for varied PUE



(a) Energy cost

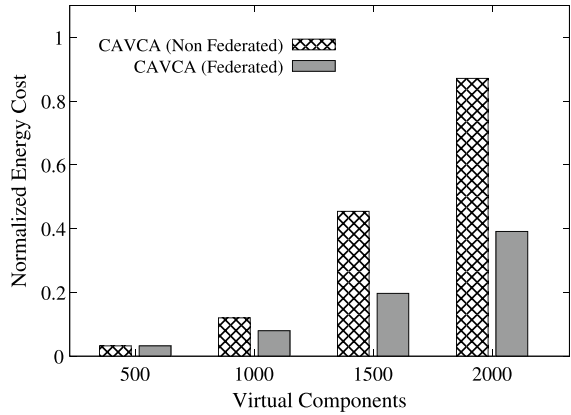


(b) Communication cost

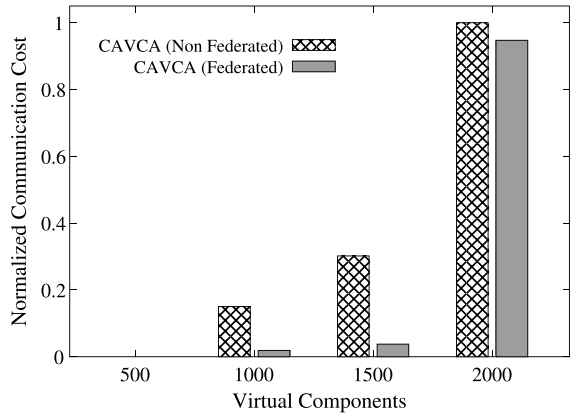


(c) Total cost

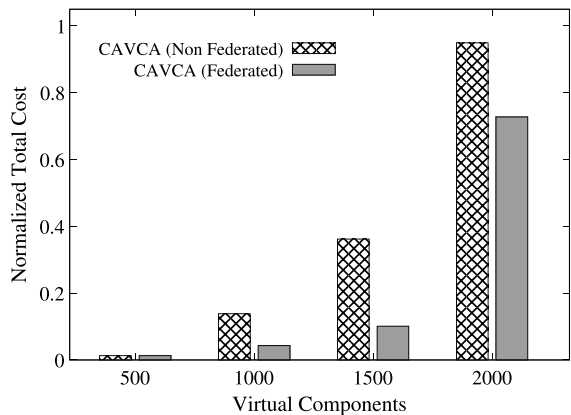
Fig. 7 Comparison of energy cost, communication cost, and total cost in federated and non-federated clouds



(a) Energy cost



(b) Communication cost



(c) Total cost

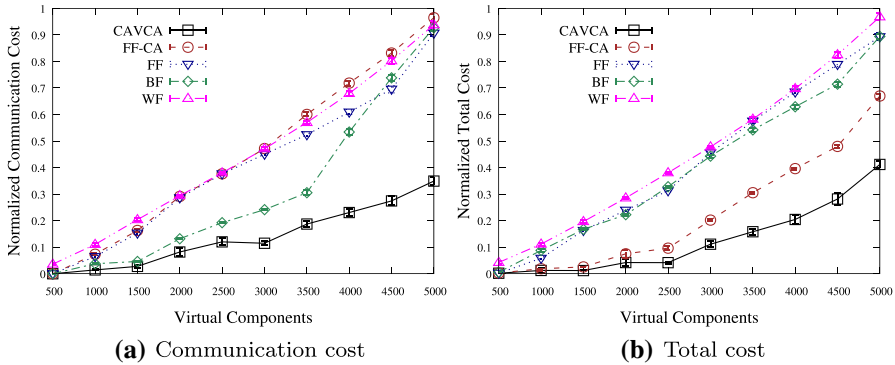


Fig. 8 Impact of request size on the communication and total costs, while the data transfer price varies

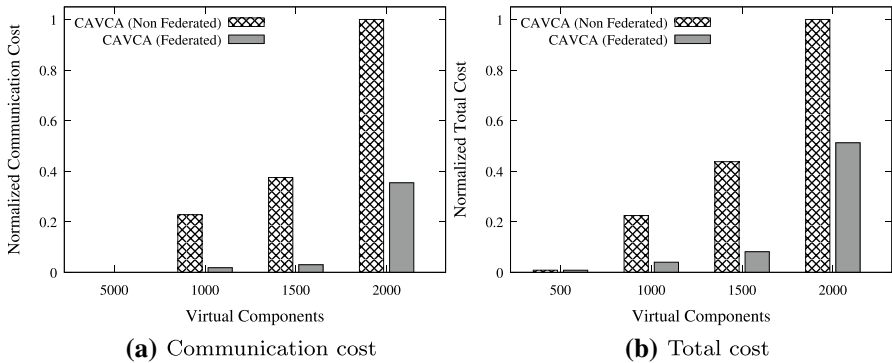


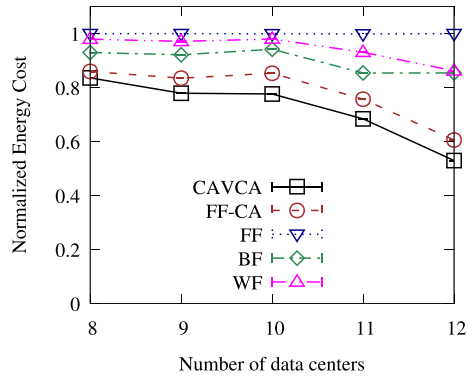
Fig. 9 Comparison of the total cost and communication cost across the federated cloud and non-federated cloud scenarios as the data transfer price changes

federation can lead to profit compared to the non-federated case, because a CP can select DCs from other CPs in the same or closer regions.

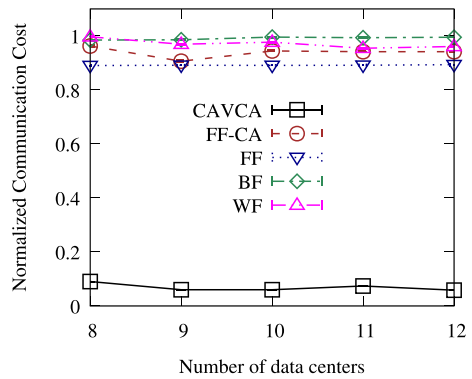
6.2.5 Impact of Adding Data Centers to the Federation

In this scenario, we assume that four DCs located in Korea, U.K., India and Canada are added in sequence to the federation shown in Fig. 4. The size of the application request is fixed at 5000 virtual components. Since the capacity of the first eight DCs can satisfy the resource demands of 5000 virtual components, the energy, communication and total costs of FF approach are constant which can be observed from Fig. 10. The change in energy cost of BF and WF depends on the capacity of the newly joined DC. The energy cost with FF-CA and CAVCA decreases while adding new DCs, since they exploit the variation in electricity price. Hence, they can use the new DC for workload allocation instead of an existing DC having higher electricity price. However, CAVCA outperforms FF-CA because it considers electricity

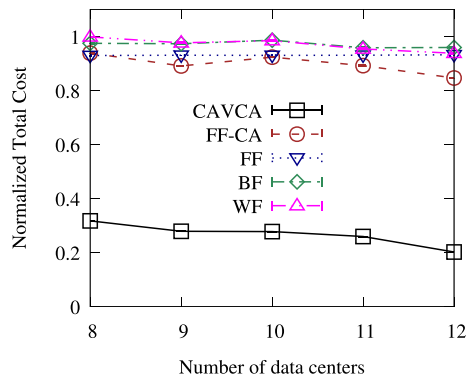
Fig. 10 Impact of number of data centers in the federation on different cost factors



(a) Energy cost



(b) Communication cost



(c) Total cost

price and PUE while selecting $Best_{DC}$ for placement. Figure 10b demonstrates the communication cost when the number of DCs is increased from 8 to 12. Since we allocate clusters of highly correlated virtual components, CAVCA method performs

better than the others, in terms of the total cost. CAVCA reduces the total cost when the electricity price varies significantly across DCs, which is true in practice.

6.2.6 Impact of Data Center Locations on the Total Cost

To study how the locations of DCs affect the operating cost of CPs in the federation, we evaluated CAVCA with a fixed request size of 3000 virtual components for different sets of locations given below:

- Set1={Switzerland, Italy, U.K., HK, Japan, Singapore, Korea, India}
- Set2={Switzerland, Italy, U.S, U.K, Canada, HK, Singapore, India}
- Set3={Switzerland, Italy, U.S, HK, Japan, Singapore, Brazil, Chile}
- Set4={Italy, U.S, U.K, Canada, HK, India, Brazil, Chile}
- Set5={Italy, U.S, U.K, Canada, HK, Singapore, Brazil, Chile}
- Set6={U.S, Canada, HK, Japan, Singapore, Korea, India, Brazil}
- Set7={Switzerland, Italy, HK, Japan, Singapore, Korea, India, Chile}
- Set8={Italy, U.S, U.K, Canada, HK, Singapore, India, Brazil}
- Set9={Switzerland, Italy, U.K, Japan, Singapore, Korea, Brazil, Chile}

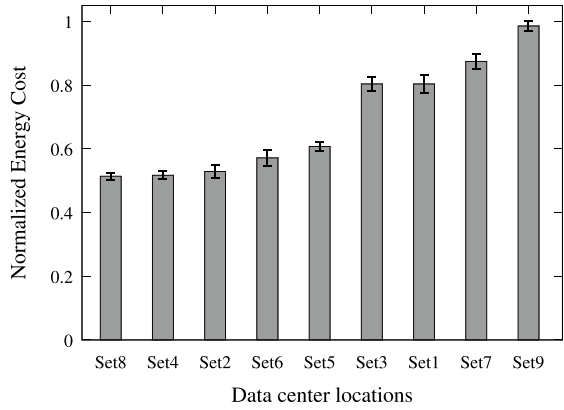
Figure 11 shows the energy cost, communication cost, and the total operating cost obtained with CAVCA executed on different sets of DCs. Set8 gives the lowest energy cost, since it contains eight DCs in countries having the lower electricity prices. However, this choice of DCs has the fifth lowest communication cost, because the DCs are distributed across three different regions, that leads to higher bandwidth cost. Set8, Set4, Set2 and Set6 lead to lower energy cost since all of them include DCs in U.S and Canada, where the electricity is cheapest. However, Set1 leads to lowest communication cost, since the DCs of this set are nearby. Although, Set2 neither has lower energy cost nor communication cost, it gives the minimum total operating cost. This is because, the DCs of Set2 are in regions where the inter-connection bandwidth price is low and also they have average electricity price.

We conclude that establishing DCs in countries where energy is cheap is not the optimum choice, because the bandwidth cost also needs to be considered. On the other hand, it might be preferable to have all the data centers in a single region, but this is not practical. Federation system gives the best balanced solution for cost and global coverage at the same time.

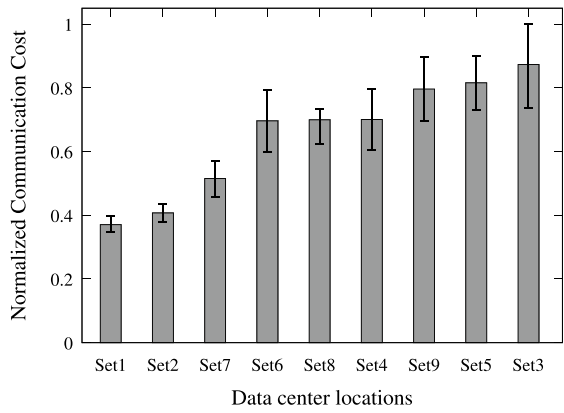
6.2.7 Average Running Time

Average running time is a significant performance metric to evaluate a placement approach and to check its feasibility of being used in real-time. Figure 12 shows the numerical results for the running time of the proposed algorithm CAVCA. It increases non-linearly with increasing application size. As evident from the figure, for small to medium applications, CAVCA takes a small fraction of time. Even for a large application with 10,000 virtual components, CAVCA does not need more than three minutes. The time taken is negligible compared to the running time of data-intensive application which may take days. From the results we conclude that

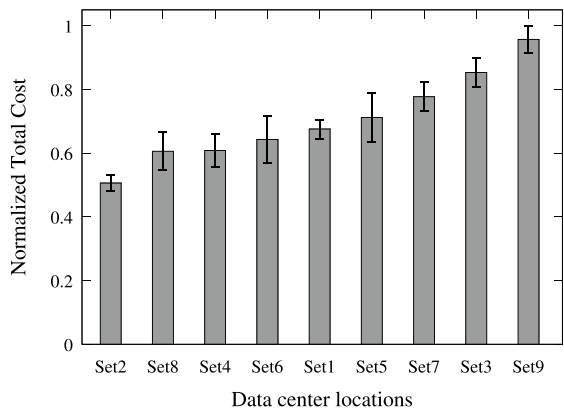
Fig. 11 Impact of data center locations (grouped into sets) on the different cost factors



(a) Energy cost

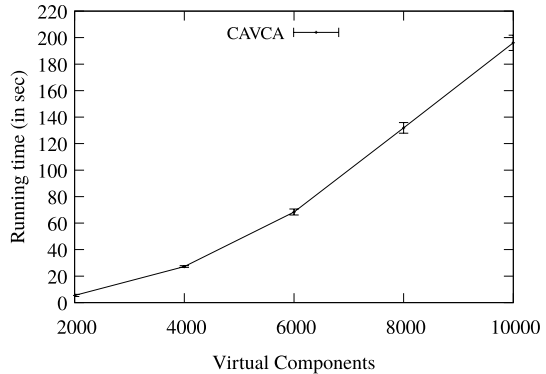


(b) Communication cost



(c) Total cost

Fig. 12 Average running time of the CAVCA algorithm for different number of virtual components, reflecting the application size



CAVCA is cost-efficient in terms of the running time for large applications and hence, it can be used for on-demand data-intensive application placement in federated cloud data centers.

7 Conclusion

The work in this paper shows how to place data-intensive applications in a federated cloud system, using a cost-aware approach. We proposed a hierarchical resource management framework that caters to the requirements of the federated cloud in terms of semi-autonomy and privacy. We formulated the problem of virtual components allocation for data-intensive applications in a federated cloud, as an optimization problem to minimize the costs of the energy consumption and communication. We proposed a heuristic algorithm to allocate a cluster of highly communicating virtual components to a DC, while minimizing the operating cost. We evaluated the proposed algorithm using extensive simulations covering realistic scenarios. A comparison with the most commonly used approaches in cloud management systems showed significant improvement in terms of the energy cost, communication cost and the total cost for all cases. We conclude that CAVCA algorithm achieves a better reduction in the total cost of operation when the cost factors vary significantly across the CPs in a federation and the correlated virtual components are scheduled together. In certain cases, CAVCA reduces the total operating cost by 60%. We conclude that minimizing operating cost in geo-distributed clouds for data-intensive applications warrants the use of federation to balance the cost and global coverage at the same time. We plan to extend the proposed approach to fog/edge computing architectures hosting delay-sensitive applications. We would like to extend the work to other federated cloud architectures such as, interconnected cloud and multi-cloud, where the service provider acts as a broker between the end user and the cloud (infrastructure) providers. We also would exploit the concept of “MultiCloud Tournament” [52] for federated cloud management and selection of the best data center for the VM placement.

Acknowledgements The authors would like to thank Dr. Bala Prakasa Rao Killi for his help in the simulation part.

References

1. Al Jawarneh, I.M., Bellavista, P., Corradi, A., Foschini, L., Montanari, R.: Big spatial data management for the Internet of Things: a survey. *J. Netw. Syst. Manag.* (2020). <https://doi.org/10.1007/s10922-020-09549-6>
2. Hajeer, M., Dasgupta, D.: Handling Big Data using a data-aware HDFS and evolutionary clustering technique. *IEEE Trans. Big Data* **5**(2), 134–147 (2019). <https://doi.org/10.1109/TBDAT.2019.2782785>
3. Calcaterra, C., Carmenini, A., Marotta, A., Bucci, U., Cassioli, D.: MaxHadoop: An efficient scalable emulation tool to test SDN protocols in emulated Hadoop environments. *J. Netw. Syst. Manag.* (2020). <https://doi.org/10.1007/s10922-020-09552-x>
4. Pretto, G.R., Dalmazo, B.L., Marques, J.A., Wu, Z., Wang, X., Korkhov, V., Navaux, P.O.A., Gaspar, L.P.: Boosting HPC applications in the cloud through JIT traffic-aware path provisioning. In: *Computational Science and Its Applications—ICCSA 2019. Lecture Notes in Computer Science*, pp. 702–716. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24305-0_52
5. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of “big data” on cloud computing: review and open research issues. *Inf. Syst.* **47**, 98–115 (2015). <https://doi.org/10.1016/j.is.2014.07.006>
6. Clemm, A., Zhani, M.F., Boutaba, R.: Network management 2030: operations and control of network 2030 services. *J. Netw. Syst. Manag.* (2020). <https://doi.org/10.1007/s10922-020-09517-0>
7. Li, P., Guo, S., Yu, S., Zhuang, W.: Cross-cloud MapReduce for big data. *IEEE Trans. Cloud Comput.* **8**(2), 375–386 (2020). <https://doi.org/10.1109/TCC.2015.2474385>
8. Mashayekhy, L., Nejad, M.M., Grosu, D.: A trust-aware mechanism for cloud federation formation. *IEEE Trans. Cloud Comput.* (2019). <https://doi.org/10.1109/TCC.2019.2911831>
9. Ehsanfar, A., Grogan, P.T.: Auction-based algorithms for routing and task scheduling in federated networks. *J. Netw. Syst. Manag.* **28**(2), 271–297 (2020). <https://doi.org/10.1007/s10922-019-09506-y>
10. Lee, C.A., Bohn, R.B., Michel, M.: The NIST Cloud Federation Reference Architecture. Tech. Rep. NIST SP 500-332. National Institute of Standards and Technology, Gaithersburg (2020). <https://doi.org/10.6028/NIST.SP.500-332>
11. Hadji, M., Zeghlache, D.: Mathematical programming approach for revenue maximization in cloud federations. *IEEE Trans. Cloud Comput.* **5**(1), 99–111 (2017). <https://doi.org/10.1109/TCC.2015.2402674>
12. Middy, A.I., Ray, B., Roy, S.: Auction based resource allocation mechanism in federated cloud environment: TARA. *IEEE Trans. Serv. Comput.* (2019). <https://doi.org/10.1109/TSC.2019.2952772>
13. Deng, H., Huang, L., Xu, H., Liu, X., Wang, P., Fang, X.: Revenue maximization for dynamic expansion of geo-distributed cloud data centers. *IEEE Trans. Cloud Comput.* **8**(3), 899–913 (2020). <https://doi.org/10.1109/TCC.2018.2808351>
14. Ray, B.K., Saha, A., Khatua, S., Roy, S.: Toward maximization of profit and quality of cloud federation: solution to cloud federation formation problem. *J. Supercomput.* **75**(2), 885–929 (2019). <https://doi.org/10.1007/s11227-018-2620-2>
15. Darzanos, G., Koutsopoulos, I., Stamoulis, G.D.: Cloud federations: economics, games and benefits. *IEEE/ACM Trans. Netw.* **27**(5), 2111–2124 (2019). <https://doi.org/10.1109/TNET.2019.2943810>
16. Venkatesh, G., Arunesh, K.: Map reduce for big data processing based on traffic aware partition and aggregation. *Clust. Comput.* (2018). <https://doi.org/10.1007/s10586-018-1799-6>
17. Eugster, P., Jayalath, C., Kogan, K., Stephen, J.: Big data analytics beyond the single datacenter. *Computer* **50**(6), 60–68 (2017). <https://doi.org/10.1109/MC.2017.163>
18. Cisco: Cisco Global Cloud Index, Forecast and Methodology, 2016–2021. White paper. Cisco System (2018). <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html>

19. Andrae, A.: Total consumer power consumption forecast. Nordic Digital Business Summit 10 (2017)
20. Forestiero, A., Mastroianni, C., Meo, M., Papuzzo, G., Sheikhalishahi, M.: Hierarchical approach for efficient workload management in geo-distributed data centers. *IEEE Trans. Green Commun. Netw.* **1**(1), 97–111 (2017)
21. Ehsanfar, A., Grogan, P.T.: Mechanism design for exchanging resources in federated networks. *J. Netw. Syst. Manag.* **28**(1), 108–132 (2020). <https://doi.org/10.1007/s10922-019-09498-9>
22. Najm, M., Tamarapalli, V.: Cost-efficient deployment of big data applications in federated cloud systems. In: 2019 11th International Conference on Communication Systems Networks (COMSNETS), pp 428–431 (2019). <https://doi.org/10.1109/COMSNETS.2019.8711284>
23. Ahmad, B., Maroof, Z., McClean, S., Charles, D., Parr, G.: Economic impact of energy saving techniques in cloud server. *Clust. Comput.* **23**(2), 611–621 (2020). <https://doi.org/10.1007/s10586-019-02946-w>
24. Ismaeel, S., Karim, R., Miri, A.: Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres. *J. Cloud Comput.* **7**(1), 10 (2018). <https://doi.org/10.1186/s13677-018-0111-x>
25. Tripathi, A., Pathak, I., Vidyarthi, D.P.: Modified dragonfly algorithm for optimal virtual machine placement in cloud computing. *J. Netw. Syst. Manag.* (2020). <https://doi.org/10.1007/s10922-020-09538-9>
26. Marcon, D.S., Neves, M.C., Oliveira, R.R., Bays, L.R., Boutaba, R., Gaspary, L.P., Barcellos, M.P.: IoNCloud: Exploring application affinity to improve utilization and predictability in data-centers. In: 2015 IEEE International Conference on Communications (ICC), pp. 5497–5503 (2015). <https://doi.org/10.1109/ICC.2015.7249198>
27. Qie, X., Jin, S., Yue, W.: An energy-efficient strategy for virtual machine allocation over cloud data centers. *J. Netw. Syst. Manag.* **27**(4), 860–882 (2019). <https://doi.org/10.1007/s10922-019-09489-w>
28. Kumar, M., Sharma, S.C., Goel, S., Mishra, S.K., Husain, A.: Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm. *Neural Comput. Appl.* (2020). <https://doi.org/10.1007/s00521-020-04955-y>
29. Khosravi, A., Andrew, L.L.H., Buyya, R.: Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. *IEEE Trans. Sustain. Comput.* **2**(2), 183–196 (2017). <https://doi.org/10.1109/TSUSC.2017.2709980>
30. Najm, M., Tamarapalli, V.: VM migration for profit maximization in federated cloud data centers. In: 2020 International Conference on COMMUNICATION SYSTEMS NETWORKS (COMSNETS), pp. 882–884 (2020). <https://doi.org/10.1109/COMSNETS48256.2020.9027429>
31. Najm, M., Tamarapalli, V.: Inter-data center virtual machine migration in federated cloud. In: Proceedings of the 21st International Conference on Distributed Computing and Networking, p. 1 (2020)
32. Maenhaut, P.J., Volckaert, B., Ongenaes, V., De Turck, F.: Resource management in a containerized cloud: status and challenges. *J. Netw. Syst. Manag.* **28**(2), 197–246 (2020). <https://doi.org/10.1007/s10922-019-09504-0>
33. Kumar, M., Sharma, S.C., Goel, A., Singh, S.P.: A comprehensive survey for scheduling techniques in cloud computing. *J. Netw. Comput. Appl.* **143**, 1–33 (2019). <https://doi.org/10.1016/j.jnca.2019.06.006>
34. Zhang, X., Li, K., Zhang, Y.: Optimising data access latencies of virtual machine placement based on greedy algorithm in datacentre. *Int. J. Comput. Sci. Eng.* **18**(2), 186–194 (2019). <https://doi.org/10.1504/IJCSE.2019.097945>
35. Ferdous, M.H., Murshed, M.M., Calheiros, R.N., Buyya, R.: An algorithm for network and data-aware placement of multi-tier applications in cloud data centers. *CoRR* (2017). [abs/1706.06035](https://arxiv.org/abs/1706.06035)
36. Shabeera, T., Kumar, S.M., Salam, S.M., Krishnan, K.M.: Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm. *Int. J. Eng. Sci. Technol.* **20**(2), 616–628 (2017). <https://doi.org/10.1016/j.jestch.2016.11.006>
37. Gu, L., Zeng, D., Li, P., Guo, S.: Cost minimization for big data processing in geo-distributed data centers. *IEEE Trans. Emerg. Top. Comput.* **2**(3), 314–323 (2014)
38. Xiao, W., Bao, W., Zhu, X., Liu, L.: Cost-aware big data processing across geo-distributed data-centers. *IEEE Trans. Parallel Distrib. Syst.* **28**(11), 3114–3127 (2017)

39. Deng, K., Ren, K., Zhu, M., Song, J.: A data and task co-scheduling algorithm for scientific cloud workflows. *IEEE Trans. Cloud Comput.* **8**(2), 349–362 (2020). <https://doi.org/10.1109/TCC.2015.2511745>
40. Latif, S., Gilani, S.M.M., Ali, L., Iqbal, S., Liaqat, M.: Characterizing the architectures and brokering protocols for enabling clouds interconnection. *Concurr. Comput. Pract. Experience* **32**(21), e5676 (2020)
41. Khorasani, N., Abrishami, S., Feizi, M., Esfahani, M.A., Ramezani, F.: Resource management in the federated cloud environment using Cournot and Bertrand competitions. *Fut. Gener. Comput. Syst.* **113**, 391–406 (2020). <https://doi.org/10.1016/j.future.2020.07.010>
42. (2020) Virtual network pricing. <https://azure.microsoft.com/en-in/pricing/details/virtual-network/>. Accessed 25 July 2020
43. Dayarathna, M., Wen, Y., Fan, R.: Data center energy consumption modeling: a survey. *IEEE Commun. Surv. Tutorials* **18**(1), 732–794 (2016). <https://doi.org/10.1109/COMST.2015.2481183>
44. Singh, N., Rao, S.: Modeling and reducing power consumption in large it systems. In: *Proceedings of IEEE International Systems Conference*, pp. 178–183 (2010). <https://doi.org/10.1109/SYSTEMS.2010.5482354>
45. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. *J. Netw. Syst. Manag.* **23**(3), 567–619 (2015). <https://doi.org/10.1007/s10922-014-9307-7>
46. Masdari, M., Salehi, F., Jalali, M., Bidaki, M.: A survey of PSO-based scheduling algorithms in cloud computing. *J. Netw. Syst. Manag.* **25**(1), 122–158 (2017). <https://doi.org/10.1007/s1092-016-9385-9>
47. Yassine, A., Shirehjini, A.A.N., Shirmohammadi, S.: Bandwidth on-demand for multimedia big data transfer across geo-distributed cloud data centers. *IEEE Trans. Cloud Comput.* **8**(4), 1189–1198 (2020). <https://doi.org/10.1109/TCC.2016.2617369>
48. (2020) Google cloud network pricing. <https://cloud.google.com/compute/pricing#network>. Accessed 25 July 2020
49. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. *Fut. Gener. Comput. Syst.* **29**(3), 682–692 (2013). <https://doi.org/10.1016/j.future.2012.08.015>
50. Baker, T., Aldawsari, B., Asim, M., Tawfik, H., Maamar, Z., Buyya, R.: Cloud-SEnergy: a bin-packing based multi-cloud service broker for energy efficient composition and execution of data-intensive applications. *Sustain. Comput. Inf. Syst.* **19**, 242–252 (2018). <https://doi.org/10.1016/j.suscom.2018.05.011>
51. Wolke, A., Tsend-Ayush, B., Pfeiffer, C., Bichler, M.: More than bin packing: Dynamic resource allocation strategies in cloud data centers. *Inf. Syst.* **52**, 83–95 (2015). <https://doi.org/10.1016/j.is.2015.03.003>
52. Assis, M.R.M., Bittencourt, L.F.: MultiCloud tournament: a cloud federation approach to prevent free-riders by encouraging resource sharing. *J. Netw. Comput. Appl.* **166**, 102694 (2020). <https://doi.org/10.1016/j.jnca.2020.102694>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Moustafa Najm is a PhD research scholar in the Department of Computer Science and Engineering (CSE), Indian Institute of Technology Guwahati. He received M.Tech from National Institute of Technology Warangal in 2016. His research interests include federated cloud computing, IoT, and network management.

Rakesh Tripathi received his M.Tech. degree from the Tezpur University, India. He received Ph.D from Indian Institute of Technology Guwahati in 2018. His research interests include data center networks, distributed systems, and game theory in networks.

Mohammad Shadi Alhakeem is an Assistant Professor of Cyber Security and Digital Forensics at Naif Arab University for Security Sciences. He received Ph.D in operating and communication systems from the Technical University of Berlin in 2010. His research areas are IoT, Cloud Computing and Cyber

Security.

Venkatesh Tamarapalli (Senior Member, IEEE) received the Ph.D. degree from the Indian Institute of Technology Madras in 2009. He is an Associate Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati. He has coauthored a book entitled *An Analytical Approach to Optical Burst Switched Networks* (Springer, 2009). His areas of research include wireless sensor networks, data center networks, and multimedia content delivery. He was a recipient of the Microsoft Ph.D. Fellowship and the Microsoft Young Outstanding Faculty Award. He is also serving as an Associate Editor for the *Journal of Network and Systems Management* (Springer).

Authors and Affiliations

Moustafa Najm¹  · **Rakesh Tripathi¹** · **Mohammad Shadi Alhakeem²** · **Venkatesh Tamarapalli¹**

Rakesh Tripathi
t.rakesh@iitg.ac.in

Mohammad Shadi Alhakeem
malhakeem@nauss.edu.sa

Venkatesh Tamarapalli
t.venkat@iitg.ac.in

¹ Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati, Assam 781039, India

² Naif Arab University for Security Sciences, Riyadh 11452, Kingdom of Saudi Arabia