# Towards Cost-aware Resource Management in Federated Cloud Data Centers




Moustafa Najm

# Towards Cost-aware Resource Management in Federated Cloud Data Centers

*Thesis submitted in partial fulfilment*
*of the requirements for the degree of*

**Doctor of Philosophy**

in

**Computer Science and Engineering**

by

**Moustafa Najm**

Under the supervision of

**Dr. Venkatesh Tamarapalli**

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

**March 2022**

***This thesis is dedicated to my parents, wife and son***

*- whose blessings, inspriation, unconditional love and support made my path of sucess.*

# Declaration

I hereby certify that

a. The work contained in this thesis is original and has been done by myself and the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.

d. No part of this thesis can be considered plagiarism to the best of my knowledge and understanding and I take complete responsibility if any complaint arises.

Date : ____/_____/_____ **Moustafa Najm**

Place: IIT Guwahati, India

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
गुवाहाटी - 781039
**Indian Institute of Technology Guwahati**
**Guwahati** - 781039

**Dr. Venkatesh Tamarapalli**
**Associate Professor**

: +91 361 258 2366
: +91 361 269 2787
: t.venkat@iitg.ac.in

Department of Computer Science and Engineering

# Thesis Certificate

This is to certify that the thesis entitled **"Towards Cost-aware Resource Management in Federated Cloud Data Centers"** being submitted by **Moustafa Najm** to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bonafide research work carried out by him under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date : ____/____/_____

Place: IIT Guwahati, India

_____

**Dr. Venkatesh Tamarapalli**
**(Thesis Supervisor)**

# Acknowledgments

***"No one who achieves success does so without acknowledging the help of others. The wise and confident acknowledge this help with gratitude."***

First and foremost, praises and thanks to the God, the Almighty, for granting me health, knowledge, and countless blessings throughout my research work to complete this thesis.

I acknowledge and express deep sense of sincere gratitude to my supervisor Dr. T. Venkatesh for his patience, motivation, enthusiasm, and continuous support during my study and research. His guidance helped me at all the time of research and writing of this thesis. It was great experience working under his supervision. I am extremely grateful for what he has offered me. I would also like to thank him for his friendship and empathy. Besides my advisor, I would like to thank my doctoral committee members: Prof. Diganta Goswami, Prof. S. K. Bose, Prof. S. V. Rao, and Dr. Ramesh Kumar Sonkar, for their valuable comments and suggestions during my Ph.D.

I would like to show my greatest appreciation to Dr. Moumita Patra and Dr. Mohammad Shadi Alhakeem for their valuable inputs to this thesis, and generously spent precious time in giving the guidance, suggestion, and comments. I would like also to thank my friends and colleagues Dr. Rakesh Tripathi and Dr. Bala Prakasa Rao Killi, for their support and help at the beginning of my Ph.D. journey. I extend my sincere thanks to all the faculty and the staff of the Department of Computer Science & Engineering, IIT Guwahati, for their continued support; special thanks to those who taught me during my course work: Prof. Pradip Kr. Das, Dr. John Jose, Dr. Arnab Sarkar, and Dr. Aryabartta Sahu.

I thank all my lab mates, colleagues, and friends, Hema, Ujjwal, Shakeel, Swagat, Subrata, Abd Alghaniy, Nilotpal, Alakesh, Debabrata, Bhawan, only to name few, for a cherished time spent together in the lab, and in social settings. I am very thankful to my two best friends Basit Andrabi and Bitopan Kalita, who were always there for me in my difficult times. My thanks are extended to my best friends outside the academic life, Muaz, Kawkab, Fayez, Aysha, Ahmad Othman, Ahmad Alammar, Anas, for their supports in different ways. Further, yet importantly, sense of respect and love goes to my two lifelines (parents) father Mr. Mohamad, mother Mrs. Nahedah for their prayers, caring sacrifices for no reward except seeing me in present stage. Without their strong support as well as regular encouragement in every step I could not have reached where I am today. Similarly, I would also like to extend my deepest gratitude to my parents-in-law, Jamal and Najwa, for always remembering me in their prayers for the ultimate success. I am extending my heartfelt thanks to my siblings, Nehal, Maher, Abdulkader, Ahmad, Iman, Safaa, Anas, Malek, to my brothers and sisters in law, Mohamad, Nour, Ahmad, Fatima, and to their

families, for their love, constant inspiration, and spiritual support throughout writing this thesis and my my life in general.

Last but not least, I would like to express my wholehearted thanks to my beloved wife for having faith in me and for her unlimited love, understanding, moral support, prayers, encouragement, and motivation throughout my life and my research work. Without her patience and great support, I would not have made it to complete this thesis.

Finally, my thanks to all the people who have supported me to complete the research work directly or indirectly.

**IIT Guwahati, March 2022** **Moustafa Najm**

# Abstract

Cloud providers face the challenge of expanding the existing data centers or build new ones to meet the ever-increasing user demand. Federated cloud data centers offer a promising solution for handling user requests in a scalable and cost-efficient manner. Federation allows a collection of cloud providers to share their resources in a collaborative environment to provide better services with lower cost. The ultimate objective of each cloud provider in the federated cloud is to maximize its own profit by minimizing its operating cost, while meeting the Quality of Service (QoS) requirements of the end-users. Resource management is an important aspect which has a direct impact on the cost and the performance of the cloud data centers. Resource management in cloud data centers is typically handled in a centralized manner, which is not scalable and compatible with the federation requirements. In addition, heterogeneous applications are hosted in federated cloud with diverse requirements, which makes it impractical to use the same resource management method. Furthermore, migration technology plays a vital role in dynamic resource management in a federated cloud system, where a cloud provider can rent resources from other cloud providers and migrate its virtual machines (VMs) for load balancing, QoS, and minimizing the operating cost. However, migration brings in an additional overhead in terms of both cost and time, and it might degrade the performance.

In this thesis, we design a resource management framework for federated cloud data centers along with some efficient techniques for dynamic resource allocation to minimize the operating cost of the cloud provider, along with considering the requirements of various applications. First, we propose a hierarchical framework for resource management in a federated cloud, catering to the requirements of each provider in the federation. This framework is used in handling all the other problems in our work. We address the problem of cost-aware resource allocation for hosting data-intensive applications in a federated cloud. We formulate it as an optimization problem to minimize the total operating cost, including energy and communication costs. We propose an efficient algorithm to allocate the virtual components of a data-intensive application, in two phases; *partitioning* and *mapping*. While *partitioning* creates clusters of correlated VMs, *mapping* exploits spatial variation in energy cost in the federation, and allocates the clusters of VMs 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 up to 60% compared to baseline methods.

Next, we address the problem of cost and delay aware dynamic resource management for hosting modern vehicular applications in a federated cloud. We introduce federated vehicular cloud computing (FVCC) architecture as an extension for the traditional vehicular cloud

computing platform. We present a dynamic resource pricing model for sharing resources between the cloud providers in the federation. We propose an online algorithm to solve the problem. The proposed algorithm is seen to meet the delay requirements with a significant cost reduction to the provider up to 50% compared to baseline methods. Also, it can serve more than 90% of the requests with lower cost and fewer VM migrations.

Finally, we address the problem of cost-aware VM migration in federated clouds to maximize the profit of the cloud provider. We present a model for pricing a virtual machine in a federated cloud. We analyse the impact of VM migration across the data centers in a federation, by modeling migration cost, migration time, and downtime. We formulate an optimization problem to minimize the total cost of operation (TCO) of the cloud provider and the migration time. We model the TCO to include the cost of hosted VMs and their migration cost. We propose efficient heuristics to select the source and destination data centers for migration, as well as the VM to be migrated. Based on these heuristics, we design a polynomial-time VM migration algorithm to minimize the TCO and the migration time. The proposed algorithm considers the characteristics of the VMs, the bandwidth and data transfer price across data centers, to give the highest benefit in migration. The proposed algorithm achieves a significant reduction in the TCO while reducing the migration time. All the proposed algorithms are evaluated based on the data from practical cloud deployments and under different scenarios. Results show that exploiting federation brings great benefits for the cloud provider and the end-users alike.

# Table of Contents

# List of Figures

**Page**

# List of Tables

**Page**

# List of Algorithms

# List of Abbreviations

| Abbreviation | Expansion |
|---|---|
| ACO | Ant Colony Optimization |
| AMPL | A Mathematical Programming Language |
| BF | Best Fit |
| CA | Cost-Aware |
| CAVCA | Cost-Aware Virtual Components Allocation |
| CB | Cloud Broker |
| CDRAM | Cost and Delay Aware Resource Allocation Method |
| CIP | Cloud Infrastructure Provider |
| CP | Cloud Provider |
| CSMA | Carrier Sense Multiple Access |
| CSP | Cloud Service Provider |
| DA | Delay-Aware |
| DB | Data Block |
| DC | Data Center |
| DCM | Data Center Manager |
| DS | Delay-Sensitive |
| DSRA | Delay-Sensitive Resource Allocation |
| DT | Delay-Tolerant |
| DTRA | Delay-Tolerant Resource Allocation |
| FCM | Federated Cloud Manager |
| FF | First Fit |
| FF-CA | First Fit Cost-Aware |
| FRA | Federated Resource Allocation |
| FVCC | Federated Vehicular Cloud Computing |

| | |
|---|---|
| IaaS | Infrastructure as a Service |
| IT | Information Technology |
| LAN | Local Area Network |
| MDP | Markov Decision Process |
| MEC | Mobile Edge Computing |
| MILP | Mixed Integer Linear Programming |
| MIs | Million Instructions |
| MIPS | Million Instruction Per Second |
| MMT | Minimum Migration Time |
| NSP | Network Service Provider |
| PaaS | Platform as a Service |
| PAM | Price-Aware Migration |
| PDR | Page Dirty Rate |
| PUE | Power Usage Effectiveness |
| QoS | Quality of Service |
| RSU | Road Side Unit |
| RTT | Round Trip Time |
| SaaS | Software as a Service |
| SDN | Software Defined Network |
| SLA | Service Level Agreement |
| SMDP | Semi-Markov Decision Process |
| SNMP | Simple Network Management Protocol |
| TCO | Total Cost of Operation |
| V2V | Vehicle-to-Vehicle |
| VC | Virtual Component |
| VCC | Vehicular Cloud Computing |
| VM | Virtual Machine |
| WAN | Wide Area Network |
| WF | Worst Fit |

CHAPTER

# 1

# Introduction

Recently, cloud computing has become one of the main utilities along with electricity, water, gas, and oil [1]. It has become a common platform for hosting wide range of applications, and most enterprises have shifted their business to the cloud to save initial setup and maintenance costs. Cisco predicts that, by 2025, 92 percent of the data center workload will be handled by cloud data centers while only 8 percent will be processed by traditional data centers [2]. The size of the global cloud market was valued at $325,689 million in 2019, and it is expected to reach $1,620,597 million by 2030 increasing about five times [3].

The rapidly growing demand for cloud services impels cloud providers to scale up their resources by extending their data centers or establishing new data centers in different regions to meet the ever-increasing demand for cloud resources. Federated cloud data centers offer a promising platform for handling the increasing resource demand in a scalable manner while avoiding the cost associated with having a large infrastructure to support peak workload [4, 5]. Federated cloud systems integrate cloud infrastructure from multiple collaborating cloud providers into a single computing platform [6]. Federation allows the cloud provider to lease resources or rent out its free resources from/to other cloud providers in the federation to scale up its capacity or to maximize the utilization of its resources [7, 8], as shown in Figure. 1.1. Federated cloud provides a pool of data centers which brings substantial benefits for the cloud providers and the customers as well. In addition, the geographic diversity of the data centers in the federation presents many advantages [9].

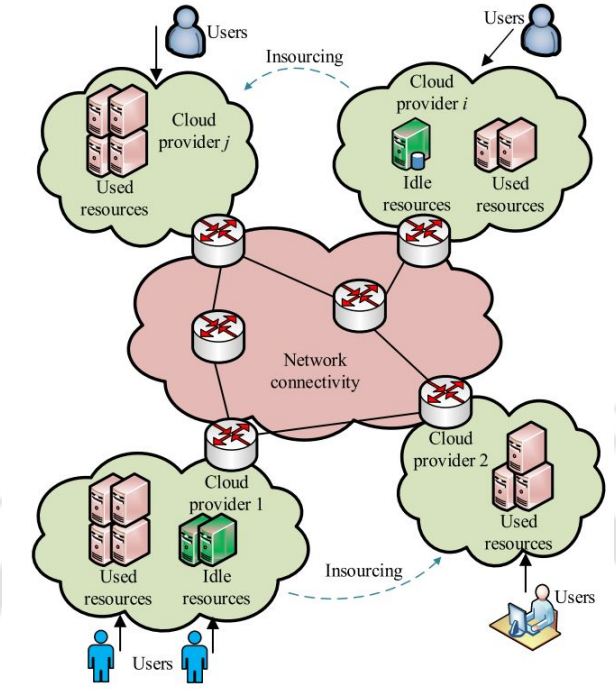



Figure 1.1: Resource sharing in federated cloud [8]

**Advantages of Federated Cloud Data Center**

Some advantages of the federated cloud compared to the traditional cloud system (non-federated) are mentioned below.

- Federated clouds offer nearly infinite resources for users as the cloud provider can rent resources from other cloud providers whenever it needs to expand its resources [9].
- With federation, even a small cloud provider can offer a truly global service without building new infrastructure by leasing resources (outsourcing) from various cloud providers across different geographical regions [10, 11].
- Federation helps the cloud provider maximize its revenue by renting out (insourcing) its idle resources to other cloud providers [12, 13].
- Using federation, a cloud provider can deliver better Quality of Service (QoS). For instance, it can reduce latency to the clients as their requests can be forwarded to a closer data center or a more capable provider [14].
- The federated platform provides high availability and fault tolerance. It can safeguard

against an unplanned outage of an entire data center by replicating the service across different cloud providers [15].

- Migration of applications between data centers in the federation is possible to handle user mobility, and to minimize the user delay, the energy consumption, or the cost of the cloud provider [16, 17, 18].

- Federation allows the cloud provider to meet legal issues and regulations such as "local data law" which inhibits data crossing territory boundaries [19].

- Flexibility and avoiding vendor lock-in are the other benefits of a federation where the customer is not dependent on a single provider. For example, the customer can place its data in Amazon storage and deploy the applications in Google cloud [9, 20].

Cloud providers offer physical resources, such as computing power, memory, storage, and network bandwidth, to end-users in a shareable manner where resources are virtualized and exposed to the user as Virtual Machines (VMs). Resource management takes care about the allocation and mapping VMs to physical machines (or servers) according to the objective of the cloud provider (e.g, load balancing, energy saving, delay minimization, or cost optimization). In federated cloud data centers, a cloud provider receives requests from end-users and from other cloud providers (insourcing), as depicted in Figure. 1.1. While every cloud provider represents an autonomous domain, the cloud provider still can allocate (or migrate) its VMs to any data center in the federation while maintaining privacy. The problem is how can a cloud provider divide the requests of its users among local and external resources to achieve its objective [12]. The ultimate goal of each cloud provider in the federation is to maximize its profit by minimizing it total cost of operation (TCO) while meeting the QoS requirements of user applications. TCO is mainly defined by the operating cost of the resources, which depends on the energy consumption cost. However, in a federated cloud, bandwidth cost introduces another factor that may impact the TCO due to massive data transfer across the data centers.

**Energy consumption cost:** Recent years have witnessed a significant increase in the size and the number of data centers, resulting in a massive increase in the power consumption of cloud data centers. A study revealed that the energy consumed by data centers worldwide in

2016 was estimated to be 268 TWh, much higher than the total energy consumption of the U.K, and it is predicted to reach up to 752 TWh in 2030, about 2.13% of the global power consumption [21]. Accordingly, energy consumption incurs the cloud provider substantial amount in electricity bills. Google pays annually $67 Million electricity bill for 1,120 GWh of power to operate its data centers, and Microsoft spends $ 36 Million for power consumption of 600 GWh [22]. Also, it has been reported that data centers in the U.S. alone consumed 91 TWh in 2020, and that is expected to rise by 60% up to 140 TWh by 2025, costing the cloud providers about $30 billion in electricity bills [22]. It is worth knowing that only 45% of the total energy consumed is used to power the information technology (I.T.) equipment in the data center, while the remaining energy is used for cooling and lighting [22]. To address this issue, cloud providers also try to build their data centers in cold places to reduce energy wastage and enhance power usage effectiveness (PUE). Federation helps to minimize the cost of energy consumption by routing the requests to a data center operating in region with cheapter electricity price (not necessarily from same provider).

**Communication cost:** Within the data center, servers are connected by a Local Area Network (LAN) with a bandwidth that may reach up to 1 Tbps. In contrast, data centers across different regions are connected through WAN (Wide Area Network) connections with limited and expensive bandwidth. The bandwidth cost between data centers varies based on the distance. Various enterprise applications hosted in the cloud generate a massive amount of data to the tune of hundreds of terabytes per day [23], and this data may be transferred between data centers. The data transfer across the data centers increases the TCO of the cloud provider significantly. A recent report predicted the annual traffic between data centers to be 20.6 ZB by 2025, resulting in $4.45 billion as bandwidth cost [2]. Federation helps the cloud provider in selecting large data centers with enough resources or nearby data centers to allocate the VMs of an enterprise application and reduce the communication cost.

Cloud systems hosts various applications with diverse features and requirements. The cloud provider cannot use the same resource management approach for all applications as the characteristics of the application should be considered [24]. In this thesis, we consider two types of applications, data-intensive applications and vehicular applications. Next, we discuss their importance before addressing the challenges of resource management.

**Data-intensive applications:** The world is witnessing an enormous growth in data generation, and it is expected that the data generated in 2025 will be fifty times the data generated in 2015 [25, 26]. This data explosion has been triggered due to the high demand for big data applications such as social media analysis and recommendation systems. With cloud computing becoming the clear future of computing, data-intensive applications are increasingly being deployed in the cloud to address their resource requirements [27, 28]. It has been reported that 71% of the cloud applications are data-intensive applications, bringing in an income for cloud providers to the tune of $126 billion [29], and this income is increasing as the cloud computing market is expected to touch $1,600 billion by 2030 [3]. Data-intensive applications are typically scheduled on physical servers using two different virtual components: data blocks (DBs) as storage nodes and 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 [23]. Due to the features such as the huge volume, and the dispersiveness of big data coupled with the scarcity of WAN bandwidth, it is inefficient and/or infeasible to process the data within a single data center [30]. Federated cloud data centers offer a promising platform for hosting such applications in a scalable and cost-efficient manner [31].

**Modern vehicular applications:** Recently, cloud has been used to support various vehicular services [1], ranging from convenience and safety to commercial and entertainment applications, such as real-time navigation, remote vehicle diagnosis, online games, and mobile social networking [32, 33]. These applications require massive resources, and cannot be run within the limited resources of the vehicles. Moreover, vehicular services are typically delay-aware, where the request should be served within a deadline, otherwise it will be dropped causing a violation of the Service Level Agreement (SLA). Vehicular Cloud Computing (VCC) platform has evolved to fulfill the massive resource demands of modern vehicular services by hosting them in the cloud [34]. However, the number of users of vehicular services is increasing rapidly. It is expected that by 2025, about 98% of vehicles will be connected to Internet, and the number of connected vehicles will reach upto two billion by 2035 [35]. This tremendous increase in the number of connected vehicles, and their high mobility pose a serious challenge to cloud providers to fulfill the requirements of all the

[1] We use vehicular services and vehicular applications interchangeably throughout the thesis

requests [36]. Federated clouds allow the cloud provider to serve an increasing number of mobile users in a scalable manner while minimizing its cost and meeting the SLAs.

**VM live migration:** Live migration is an essential technique used for dynamic resource management in cloud data centers [37, 38]. It allows the cloud provider to migrate its VMs between servers within a single data center or across different data centers to re-optimize the current resource allocation, subject to various objectives, such as load balancing, energy consumption minimization, and QoS improvement [39, 40, 41, 42, 43]. The main performance metrics of live migration are migration time and downtime, which need to be lowered. Unlike live migration within the data center, inter-data center VM migration is carried out over WAN links and requires transferring the VM's memory image and the disk image as well [17]. This is costly and might degrade the performance. For instance, migration of a VM hosting a Web service and configured with 1.7 GB memory and 10 GB disk, over an 85 Mbps WAN link between two data centers located in Texas and Illinois takes more than 40 minutes. The response time may reach up to 52 ms during the migration compared to 10 ms in a typical scenario [20]. Inter-data center migration may not be suitable for some applications, such as data-intensive applications, as it results in high bandwidth cost increasing the TCO of the cloud provider, and requires a long migration time [44]. Examining the impact of migration is important, and the decision to include migration in resource management should be taken carefully, considering the nature of the application. In a federated cloud, the cloud provider can migrate its VMs across various data centers to handle user mobility, reduce its cost or the user delay, and maximize its profit [45, 46, 47].

**Resource Management in Federated Cloud Data Centers:** Resource management in the federated cloud system is challenging for the cloud provider due to its direct impact on the cost of the cloud provider and the performance of the cloud system [38]. Though resource management has been widely explored within a single data center, federated cloud data centers pose new challenges. The challenges are the semi-autonomy and privacy of each member in the federation, the possibility of sharing resources with other cloud providers, the spatial cost diversity, inter-data center communication, and the VM migration between data centers. This thesis addresses the problem of resource management in federated data centers considering some of these challenges.

## 1.1 Motivation for the Research Work

We noted from the literature survey that a major part of cloud providers' income comes from infrastructure services where physical resources are exposed as a service (i.e., VMs). However, the cost of this service represents a significant fraction of the cloud provider's expenses. The ultimate goal of the cloud provider is to maximize its profit by minimizing the cost while meeting the requirements of the user applications. The cloud provider can utilize the variation of resource prices in a federated cloud system and outsource its VMs to other cloud providers to minimize its own cost. Most of the work done in the literature focuses on resource management within a single data center, while few works consider the federated cloud. In addition, cloud providers typically offer several types of VM instances with different resource capacities and diverse prices, which may vary across data centers. Most of the works in the literature assumed VMs to be homogeneous, which is not practical. The work in this thesis considered heterogeneous VMs and data centers. An efficient resource management strategy should also consider the requirements of the hosted applications to minimize the cost of the cloud provider and to meet the SLA requirements of the end-user.

The primary research objective in this thesis is to minimize the TCO of the cloud provider while considering the characteristics and the constraints of various hosted applications. We design a hierarchical framework for resource management in a federated cloud, based on which we design algorithms to handle the resource allocation for different applications. We study an efficient migration technique for migrating VMs across data centers of the federation as a part of the thesis work. In what follows, we discuss the motivation behind our proposals in detail.

**Resource management framework for federated cloud data center:** From the literature, we observe that resource management was usually handled in a centralized manner, mainly in a single data center or on a limited scale. This approach requires the cloud provider to store updated information about the resources of all servers in a central database, yielding a non-scalable solution. In addition, it leads to a single point of failure where the whole federation fails if the central manager fails. This approach may be appropriate for a single-owner cloud system where the cloud provider has full access to the resources of all data centers and servers in the cloud system. However, the semi-autonomous

nature of federated cloud environment and the privacy of each member in the federation warrant that a cloud provider cannot directly access the resources information in the other cloud providers. Another solution is the distributed framework with no central controller, and all entities (servers or data center managers) communicate with each other to manage the resources and decide the allocation. This method is also not scalable, and it requires extensive communication resulting in high network overhead. Therefore, the hierarchical approach is an intermediate solution that is most suitable for federated cloud systems. Resource management is performed on multiple levels in this approach, i.e., data center and cloud provider levels.

**Optimization of energy consumption cost:** Energy consumption is estimated to account for 30% to 50% of the TCO of the cloud provider, predominating the initial cost [48]. Therefore, energy cost optimization is an essential factor in minimizing the TCO. An efficient resource management strategy should mitigate energy consumption and its associated cost. Electricity price varies remarkably across different countries, ranging between $60 and $250 per MWh [49]. Energy cost can be minimized in a federated cloud by leveraging the spatial variation of electricity prices across various data centers. However, data centers are different in terms of energy consumption efficiency, measured by PUE. From the literature, we have observed that the PUE of data centers ranges between 1.1 to 2 [50]. Thus, considering electricity price as the only parameter for optimizing energy cost is not enough, e.g., the data center may be located in an area where the electricity is cheap, but the ambient temperature is high, leading to more energy consumption by the cooling system. Consequently, while selecting the best data center for resource allocation, in terms of energy cost, we have to consider electricity price and PUE together to reduce the cost of energy and power consumption as well.

**Optimization of inter-data center communication cost:** Another important factor that affects the TCO of the cloud provider is the communication cost. In federated cloud, the data transfer price varies based on the domain. For instance, Google cloud charges (per GB) 0$ within the data center, 0.01-0.08$ between two data centers within the same geographical region, and 0.08-0.23$ between data centers across different regions [51]. Most of the works in the literature consider independent VMs for allocation, without considering the traffic

demand between them. This may allocate VMs with high communication in different data centers across various regions, that increases inter-data center communication cost. Therefore, the cloud provider has to consider the features and the traffic pattern between the VMs of the application and select a set of data centers such that the communication cost is minimized.

**Placement of Data-intensive Applications in Federated Cloud:** Data-intensive applications often require huge amount of resources and involve bulk data transfer between data and computing virtual components, posing a major challenge for small cloud providers [52]. Federated cloud data centers present a promising platform for hosting these applications. The placement of communicating virtual components on a single data center is a potential solution to avoid latency and bandwidth cost. However, a single data center may not be able to accommodate all the virtual components of an application with large resource requirements [30, 53]. Therefore, these components may be distributed across multiple data centers in a federation, wherein inter-data center communication cost may lead to a drastic increase in the TCO, and the bandwidth cost may surpass other cost components [52]. Reducing WAN traffic of data-intensive applications is critical to minimize the bandwidth cost. Accordingly, highly communicated virtual components should be placed within the same data center to reduce inter-data center communication. On the other hand, the data-intensive applications lead to higher energy cost, which contributes to a significant part of the TCO. Therefore, minimizing the cost of energy consumption while hosting data-intensive applications is another critical issue for the cloud provider. We have observed from the literature that energy consumption is usually modeled as a function of CPU utilization only. However, sometimes the energy consumed by the memory or the storage resources might be significant; for example, in data-intensive applications, storage consumes about 40% of the total energy of the data center [50]. Thus, power consumption of various resources should be considered according to the pattern of the application resource demand.

While the literature focuses only on the bandwidth cost or the energy consumption cost, we argue that both should be considered to optimize the cost and performance of data-intensive applications. The cloud provider should allocate clusters of correlated virtual components (to minimize inter-data center communication cost) while leveraging

the variation in electricity price and PUE across data centers of the federation (to reduce energy consumption and cost).

**Dynamic Resource Allocation of Vehicular applications:** Exploiting federated cloud in VCC brings many benefits to users as well as to the vehicular service providers. For instance, there could be a large number of users in a specific region (e.g. city center) that can not be served by a single data center. Another case is where there is a sudden increase in the density of vehicles during peak time. Using federated clouds, the vehicular service provider can avoid infrastructure expansion for peak workload by shifting the overload to a neighboring cloud provider temporarily [16]. The geographic diversity of the cloud providers in a federation presents other benefits for vehicular services, such as cost efficiency and low latency [19]. Federation also helps to handle mobility, which typically is done by migrating the VMs to a closer data center to minimize the delay [18].

Vehicular applications are delay-aware with various delay thresholds. They can be classified based on their delay requirements into two categories; delay-sensitive applications, such as delivery of safety messages, and delay-tolerant applications, such as supporting social network messages [54]. Resource management is a crucial task for the cloud providers, where the requirements of various applications have to be met while minimizing the operating cost of the cloud provider [55]. In VCC, user mobility adds another dimension to the problem of resource management, making it more challenging [56]. The service provider needs to continuously monitor the network delay and trigger VM migration when required to meet the delay requirements while considering the cost as well [54]. There are a few works in the literature that consider optimizing the cost in VM migration, but do not consider the delay requirements, inherent to vehicular applications. In the literature, most of the works either minimize the delay of the request or the cost of the cloud provider. We argue that a trade-off should be considered for the benefit of both the cloud provider and the end-user. In addition, VM migration should be utilized for dynamic resource allocation to tackle user mobility.

**Inter-Data Center VM Migration in Federated Cloud Data Center:** As discussed earlier, the ultimate objective of the cloud provider in a federated cloud system is to maximize its own profit by minimizing its TCO [57]. A cloud provider can reduce its TCO by utilizing

the spatial variation in the cost of resources across the data centers of the federation and migrating its VMs to a data center that offers lower price [58]. However, inter-data center VM migration in the federated cloud is a costly affair as it may produce a massive amount of data that need to be sent over WAN links with restricted bandwidth and high data transfer prices [17]. Accordingly, migration brings an additional cost and increased delay, resulting in performance degradation. Therefore, analysis of migration overhead in terms of time and cost is essential during dynamic resource allocation in federated clouds. Although the problem of VM live migration has been widely explored in the literature, most of the works have been done within a single data center where the migration occurs over LAN and the migration overhead can be neglected. Only few works addressed the live migration between data centers, mostly without considering migration overhead.

Generally, VM migration studies in the literature considered homogeneous VMs [59]. In practice, the cloud providers host heterogeneous VMs with different types of demands and requirements to handle various applications (such as compute-intensive, memory-intensive, and storage-intensive applications) [60, 61]. The type of VM influences the migration cost, time, and benefit [2]. For example, a compute-intensive VM can be migrated easier than a memory-intensive and storage-intensive VM, with a higher migration benefit and lower migration time. Migration may not be preferable in some cases, like migrating a database server [48]. Further, migrating a VM with a longer residual lifetime gives higher migration profit. Furthermore, some VMs are not eligible for migration due to security or special user requirements (e.g., local data low) [62]. In addition, as we use pre-copy live migration, the rate of modifying the data of the VM (i.e., page dirty rate (PDR)) should be considered. Higher PDR may increase the traffic leading to longer migration time and a higher migration cost. In some cases, the migration cost of a VM may surpass the migration profit resulting in a negative migration benefit. Therefore, the VM and workload characteristics should be considered while selecting the VM to be migrated [59].

In summary, while minimizing the TCO of the cloud provider, various factors need to be considered based on the application. For data-intensive applications, resource management strategy needs to consider inter-data center communication cost with the energy cost. On

[2] We define migration profit as the gain due to the difference in the VM cost between the source and the destination data center, while the migration benefit is the profit after subtracting the migration cost.

the other hand, the delay requirements need to be considered besides minimizing the cost. Further, while performing inter-data center VM migration to maximize the profit, the cost of migration has to be taken into account, considering the migration overhead to maintain a good performance.

## 1.2 Contributions of the Thesis

The main objective of the thesis is to work towards a cost-aware resource management framework for a federated cloud, that effectively manages resource sharing across the cloud providers in a federation with a dynamic resource allocation to minimize the TCO of the cloud provider. While doing this our resource management strategies also consider the requirements of various applications. Considering the aforementioned motivational factors, we summarize the set of problems for cost-aware dynamic resource allocation in a federated cloud. We highlight the salient contributions of the thesis, spannning across different aspects of resource management.

### 1.2.1 Design a Hierarchical Management Framework for Federated Clouds

**Research problem: How to design a scalable and efficient resource management framework that meets the requirements of the federated cloud system with a minimum network overhead?**

The first objective of the thesis is to design a framework for resource sharing and workload allocation that suits the requirements of a federated cloud, such as semi-autonomy, privacy, and scalability. The proposed framework is designed in a hierarchical manner using two different layers for each cloud provider in the federation: (i) a Data Center Manager (DCM) for each data center in the federation, which is part of a local management layer and (ii) a Federated Cloud Manager (FCM) for each cloud provider, which is a part of a federation management layer. The FCM is responsible for communication with its own DCMs, its end-users, and FCMs of other cloud providers. When an FCM needs to distribute workload, it contacts other FCMs to collect the status of their data centers.

The proposed framework is scalable and it can be grown organically as more cloud providers 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. It does not suffer from single point of failure; if an FCM fails, the cloud provider can simply create a new FCM without any state transfer.

### 1.2.2 A Cost-aware Placement of Data-intensive Applications on Federated Cloud

**Research problem: How can a cloud provider minimize its TCO while allocating the resources for deploying a data-intensive application in a federated cloud system, taking into account the cost of energy consumption and network bandwidth?**

In this work, a cloud provider tries to host data-intensive applications at minimum cost. Continuous communication and a massive data transfer across the virtual components (VCs) of a data-intensive application, scheduled across the data centers adds to the communication cost. The communication cost must be minimized by allocating correlated VCs to the same data center. The energy consumption cost of the VCs should be minimized by leveraging electricity price and PUE variation across different data centers in the federation. Therefore, the problem at hand is, how to allocate groups of collaborating VCs to a set of data centers in a federation to minimize the TCO, including the costs of energy consumption and network bandwidth.

The main contributions of this work are:

1. We formulate the problem of placing collaborating VCs of a data-intensive application as an optimization problem to minimize the TCO, including costs of energy consumption and inter-data center communication, with a set of constraints that represent resource availability.

2. Using the hierarchical management framework, we propose an algorithm for resource allocation that divides the VCs into a set of clusters based on the communication requirements. The data-intensive application is represented as a weighted directed graph where the nodes are the virtual components, and the edges express their traffic demand. The proposed algorithm receives an application request and runs iteratively in two phases; *Partitioning* and *Mapping*. The *Partitioning* phase partitions the request graph into clusters based on the traffic density between the nodes by removing the lowest weighted edges, minimizing inter-data center communication and bandwidth

cost. The *Mapping* phase allocates the created clusters of virtual components to a data center with a minimum energy cost (considering electricity price and PUE) to minimize the operating cost. We analyze the time and space complexity of the algorithm with respect to the number of VCs and the number of data centers, and find them to be polynomial. We find the cost of running the algorithm to be negligible compared to the cost saving achieved by the algorithm. The proposed algorithm is seen to outperform other baseline approaches in terms of energy cost, communication cost and the total cost.

### 1.2.3 Cost-and-Delay Aware Dynamic Resource Allocation in Federated Vehicular Clouds

**Research problem: How can a cloud provider hosting vehicular services allocate resources dynamically in a federated cloud system to minimize the cost while assuring the QoS requirements of different vehicular services, considering the user mobility?**

In this problem, we aim to minimize the cost of the cloud provider while considering the delay constraints of various applications. Typically, modern vehicular applications are delay-aware. Based on their delay requirements, they can be classified into two categories; delay-sensitive (DS) and delay-tolerant (DT) applications, which are characterized by different delay thresholds and corresponding cost thresholds [54]. The cloud provider has to minimize the delay for DS applications and the delay should not exceed a threshold. In contrast, DT applications only require the delay not to exceed a reasonably large threshold. For both applications, the associated cost thresholds should be considered during resource allocation. Further, user mobility in VCC makes the problem more challenging, where VM migration might be employed to meet the delay and the cost requirements continuously.

The main contributions of this work are:

1. We introduce Federated Vehicular Cloud Computing (FVCC) architecture which applies the federated cloud in VCC to handle the resource-intensive vehicular applications. FVCC leverages the advantages of the federation for hosting vehicular applications, such as minimizing the cost and the delay.

2. We propose a dynamic pricing model for resource sharing among the cloud providers.

This pricing model defines a VM instance's renting price to other cloud providers in the federation, which ranges dynamically between the operating cost and the end-user price based on resource utilization.

3. We propose a cost-and-delay aware dynamic resource management algorithm in FVCC that classifies the user request based on application category (DS or DT) and performs online resource allocation considering the application's requirements and the associated cost threshold. It also migrates the VMs when required (based on mobility and delay requirements). The proposed algorithm is shown to meet the delay requirements with a significant cost reduction to the provider up to 50% compared to baseline methods. It serves more than 90% of requests with lower cost and fewer VM migrations.

### 1.2.4 Towards Cost-aware VM Migration to Maximize the Profit in Federated Clouds

**Research problem: How can a cloud provider utilize VM live migration across data centers in a federated cloud effectively to maximize its profit while minimizing the migration cost and time?**

In this problem, we explore inter-data center live migration in a federated cloud system to maximize the profit of the cloud provider by minimizing its TCO through leveraging the variation in resource price across data centers of the federation. However, inter-data center migration adds extra cost due to data transfer of the migrated VMs. We attempt to minimize the TCO and the migration time at the same time. We also study the various parameters that affect the migration performance and cost, including the parameters related to VM (i.e., type, lifetime, PDR, and eligibility for migration) and the parameters related to data centers (bandwidth and data transfer price). We present a comprehensive model for estimating migration performance and cost considering all parameters. The main contributions of this work are:

1. We model the VM cost in a federated cloud system; based on the energy cost (if the VM is hosted locally) and using the dynamic pricing model proposed in the previous problem (if the VM is outsourced to another cloud provider). We also model the migration cost, migration time, and downtime of migration.

2. We formulate the problem of inter-data center live VM migration in a federated cloud as an optimization problem, with the objective of minimizing the TCO of the cloud provider and the total migration time as well. This problem formulation is a variant of the dynamic bin packing problem, that is in NP-hard.

3. We propose three heuristics to select the source and the destination data centers of the migration, and the VM to be migrated. To include the migration bandwidth and the data transfer price in the decision, we define *migration benefit* metric, as the profit gained by migrating the VM to a potential destination after subtracting the migration cost.

4. Based on the heuristics, we propose an efficient VM migration algorithm in a federated cloud that migrates only the eligible VMs, whose migration maximizes the profit of migration. The proposed algorithm consists of two algorithms: one to select the source data center and the VM to be migrated, while the other one to select the destination data center that gives the highest migration benefit. We show that the time and space complexity of the algorithms are polynomial. The proposed algorithm is shown to outperform other baseline approaches, with up to 48% reduction in the TCO along with minimum migration time, downtime, and number of migrations.

## 1.3 Organization of the Thesis

The rest of the thesis is organized as follows:

**Chapter 2** discusses in brief, the general architecture of a federated cloud, followed by an overview of the vehicular cloud computing architecture. Then, it briefly defines the problem of resource management in cloud data centers and various migration techniques. Next, a comprehensive taxonomy of resource management problem is presented. Finally we present a state-of art literature survey on energy-aware and network-aware resource allocation, resource management frameworks, the placement of data-intensive applications, resource management in vehicular cloud and VM live migration.

**Chapter 3** begins by presenting the proposed hierarchical resource management framework for federated cloud and then presents the optimization formulation of the placement of data intensive applications, with an objective of minimizing the TCO of the

cloud provider. In this chapter we also discuss the models for the cost of energy consumption and inter-data center communication. We present a two-phase algorithm for the placement of data-intensive applications in a federated cloud system and then discuss the results from evaluation of the proposed algorithm for different scenarios.

**Chapter 4** presents the work done for delay and cost aware deployment of vehicular cloud applications in a federated cloud. We first describe the system model, including the proposed FVCC architecture, and the dynamic resource pricing model for resource sharing. Next, an algorithm for solving the problem is proposed and then the simulation results are presented to show the benefits of the proposed algorithm.

**Chapter 5** discusses the problem of live VM migration across data centers in a federated cloud system to maximize the profit of the cloud provider. It provides a model for VM cost in a federated cloud and introduces a model for estimating migration performance. It presents the problem formulation to minimize the TCO of the cloud provider, considering migration cost, and the migration time as well. We present an inter-data center migration algorithm that uses three heuristics for selecting the source and the destination data centers for migration, and the VM to be migrated. Finally, we discuss the experimental results to demonstrate the efficiency of the proposed algorithm in terms of TCO and migration performance.

Finally, **Chapter 6** concludes the thesis with a summary of the research contributions and suggests some directions for future work.

CHAPTER

# 2

# Background and Literature Survey

In this chapter, we present some background material relevant to the problems addressed in the thesis and also discuss the literature surveyed on these topics. First, we present the general architecture of federated cloud data centers considered in the work and also discuss vehicular cloud computing briefly. Next, we discuss two important concepts, virtualization, and migration. We give an overview of various migration techniques and explain the pre-copy live migration technique used in our work. Finally, we present a comprehensive taxonomy of resource management problems and then discuss the state-of-art literature related to the problems addressed in the thesis.

**Organization of the Chapter:** The rest of the chapter is organized as follows. Section 2.1 discusses the architecture of federated cloud data center considered in the thesis. In Section 2.2, we discuss the salient features of vehicular cloud computing, relevant to the work in Chapter 4. The basic concepts of virtualization and migration are discussed in Section 2.3. Section 2.4 details the taxonomy of the resource management problems in cloud networks. We discuss the literature relevant to the problems addressed in the thesis and the limitations therein in Section 2.5. The chapter is summarized in Section 2.6.

## 2.1 Federated Cloud Data Center Architecture

We are living in the era of cloud computing, which is an evolving paradigm that allows rapid and on-demand access to a pool of shared configurable resources that can be provisioned easily,

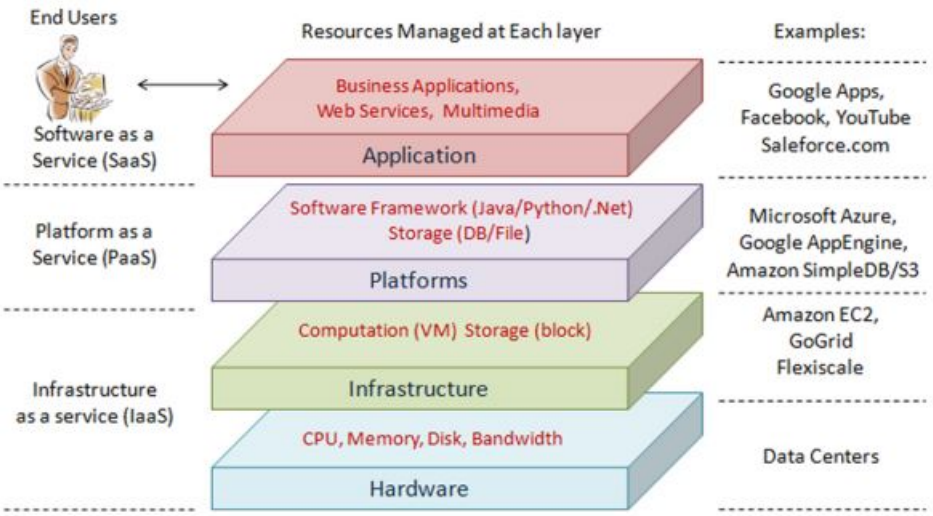



Figure 2.1: Cloud service model

typically over Internet [62]. This encouraged most of the big companies and organizations to shift their businesses to the cloud [3]. Cloud computing services can be delivered in three different models based on pay-as-you-use, as depicted in Figure. 2.1. They are (i) Infrastructure as a Service (IaaS) which offers physical resources like processing power, memory, storage, and network bandwidth to customers, (ii) Platform as a Service (PaaS), where a cloud service provider delivers the computing environment which is required for building cloud applications, and (iii) Software as a Service (SaaS) which provides applications hosted on cloud and the end-user access the application through internet [63]. In this thesis, we focus on IaaS where customers can share physical resources that are hosted in data centers.

The massive demand for cloud resources has led to the emergence of the geo-distributed cloud data center, which is a collection of many small data centers located in different geographical areas. A data center may contain just a few servers in a single private cloud or may have several thousands of interconnected servers across a public geo-distributed cloud. It has been reported that the number of hyper-scale data centers worldwide increased by 226% during the last five years, going up from 259 in 2015 to 597 data centers at the end of 2020 [64]. For example, Amazon operates 81 availability zones; each has one or more data centers across 25 regions in 4 continents [65].

However, the problem is that the big cloud users cannot get all the required resources from a single cloud provider, no matter how large it is. Most of the main cloud providers also have a limited geographic presence, and they usually establish their data centers in places profitable for their businesses [30]. If we look at the majority of applications that are deployed in the cloud, such as social media, e-commerce, big data analysis, and online games, they all require humongous resources and global coverage. Furthermore, some customers may have specific requirements. For example, a customer may have already stored data in Amazon S3 but wants to deploy a computing service using Google compute engine without moving the data.

The federated cloud data center has emerged as a promising solution for hosting such applications in a scalable and cost-efficient manner. Federated cloud systems aggregate the resources provided by the data centers of collaborating cloud providers into a single platform [7, 6]. The architecture of a federated cloud system is illustrated in Figure. 2.2. Actors or stakeholders usually participating in the federated cloud system are [8]:

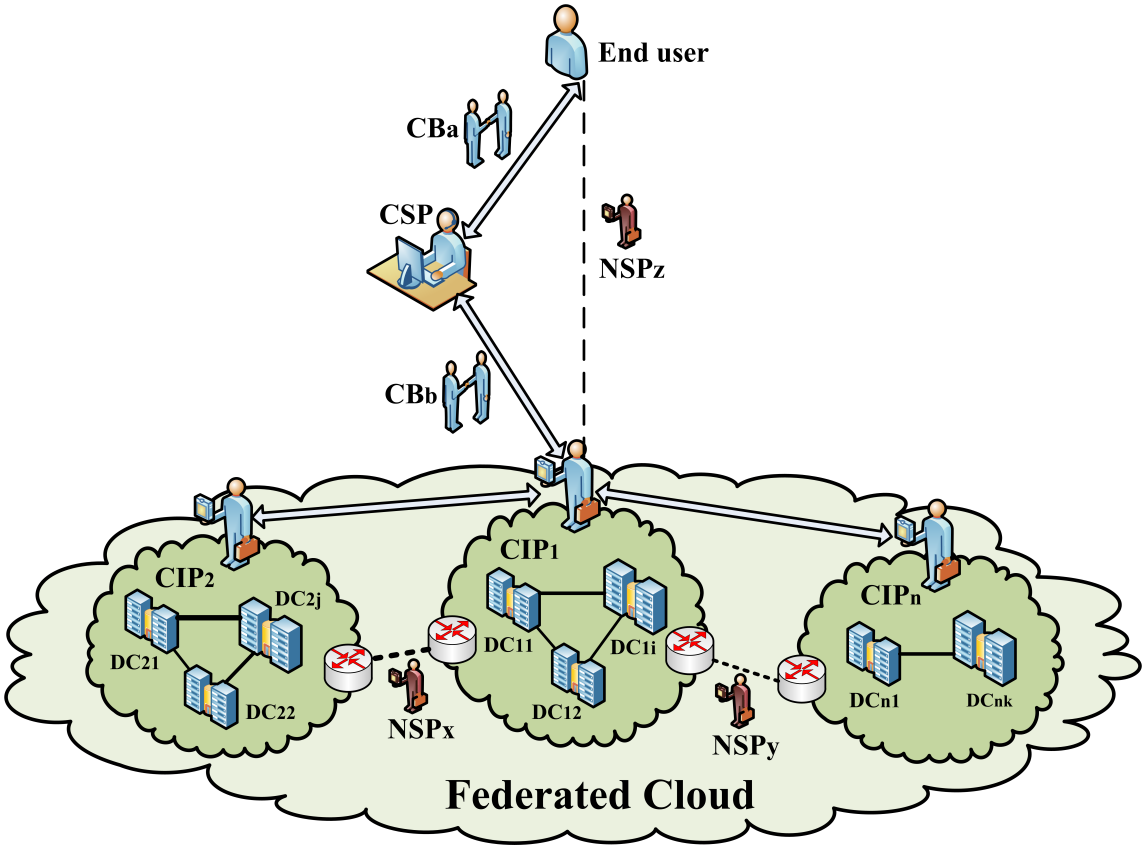



Figure 2.2: Federated cloud system architecture

- *Cloud Infrastructure Provider (CIP):* It owns and manages data centers which include physical resources. Every CIP represents an anonymous subsystem within the federation with one or more data centers.

- *Cloud Service Provider (CSP):* It offers various cloud services to customers using resources provided by the CIPs participating in the federation. It may use resources from different CIPs.

- *End User*: This is the cloud consumer that requests services from the CSP; it may be a mobile user in some cases, such as a vehicle moving on a highway.

- *Cloud Broker (CB):* It acts as a proxy between the end users and the CSPs and between the CSP and CIP to find the best CSP (or CIP) according to pre-specified criteria.

- *Network Service Provider (NSP):* It provides network access or bandwidth and provides connectivity between end users and data centers of the cloud, as well as across the data centers of various CIPs.

Figure. 2.2 presents the general architecture of the federated cloud system. The CSP may not have any data center where it offers services by leasing resources from CIPs. A CIP may offer infrastructure services (and other services) directly to end users acting as a CSP, such as Google and Amazon cloud providers. In this case, end users may request the service from the CIP directly without any broker. In traditional geo-distributed data centers, CSP connects to one CIP that has multiple data centers in different geographical locations where the CIP can use the resources of its data centers only. On the other hand, CSP can connect to several CIPs simultaneously in a multi-cloud scenario. However, in a federated cloud, a CSP connects to only one CIP, and this CIP may communicate with other providers and rent resources from them. This is transparent to the CSP, which interacts only with one CIP. Federation minimizes the resource management effort for the CSP and also minimizes the communication between the CSP and CIPs as well. The problem of resource management in a federated cloud is restricted within the boundary of the federation. Each CIP represents an individual entity within the federation where the privacy of the CIPs should be maintained, and each CIP can decide its own strategy regarding resource sharing.

In traditional geo-distributed data centers, the CIP usually has its own inter-data center WAN network. But, in a federated system, a CIP needs to engage with an NSP to connect to another CIP. There may be various NSPs that offer different types of network connectivity services at different prices. Hence, the data centers of a federation are connected through links with diverse bandwidths and pricing. The price of network bandwidth (typically measured in $/GB) also varies based on the geographical distance [51]. Since we focus on resource management in a federated cloud system, we assume CIP to also act as a CSP, which we term as *cloud provider (CP)* in the thesis. The problem here is that the CP has to choose the best resource allocation for hosting applications according to their requirements and the cost of various resources across the federation, considering the possibility of resource sharing and workload outsourcing.

## 2.2 Vehicular Cloud Computing

Recently, vehicular networks have attracted the attention of industry and academia as a vital part of the smart city idea [66]. Vehicular networks are still in the process of development, with the Internet acting as an essential service platform (an indispensable part of an intelligent transportation system). Various vehicular services have been proposed to enhance the safety conditions and to improve the driving experience. For instance, it was reported that deploying safety message systems at highway intersections could reduce vehicle accidents in Western Europe by 25% [67]. Another application is real-time traffic analysis, where real-time traffic data could be collected from a large number of vehicles and transmitted to cloud data centers for processing, and in return, some useful information could be broadcast to traffic officers or drivers to help them in route planning.

Earlier vehicular ad-hoc networks (VANETs) consisted of vehicles with onboard units that can communicate and share information to enhance the driving experience [68]. Despite the modern technologies used in vehicles, the rapid evolution of vehicular services presents a significant challenge for traditional VANETs. Vehicles usually have limited computing and storage resources due to the requirements of low cost and small-sized hardware, which result in low data processing speeds [67]. On the other hand, modern vehicular services require complex computation and demand massive storage. To this end, VANETs are taking a new direction, with increased usage of cloud computing to support resource-intensive

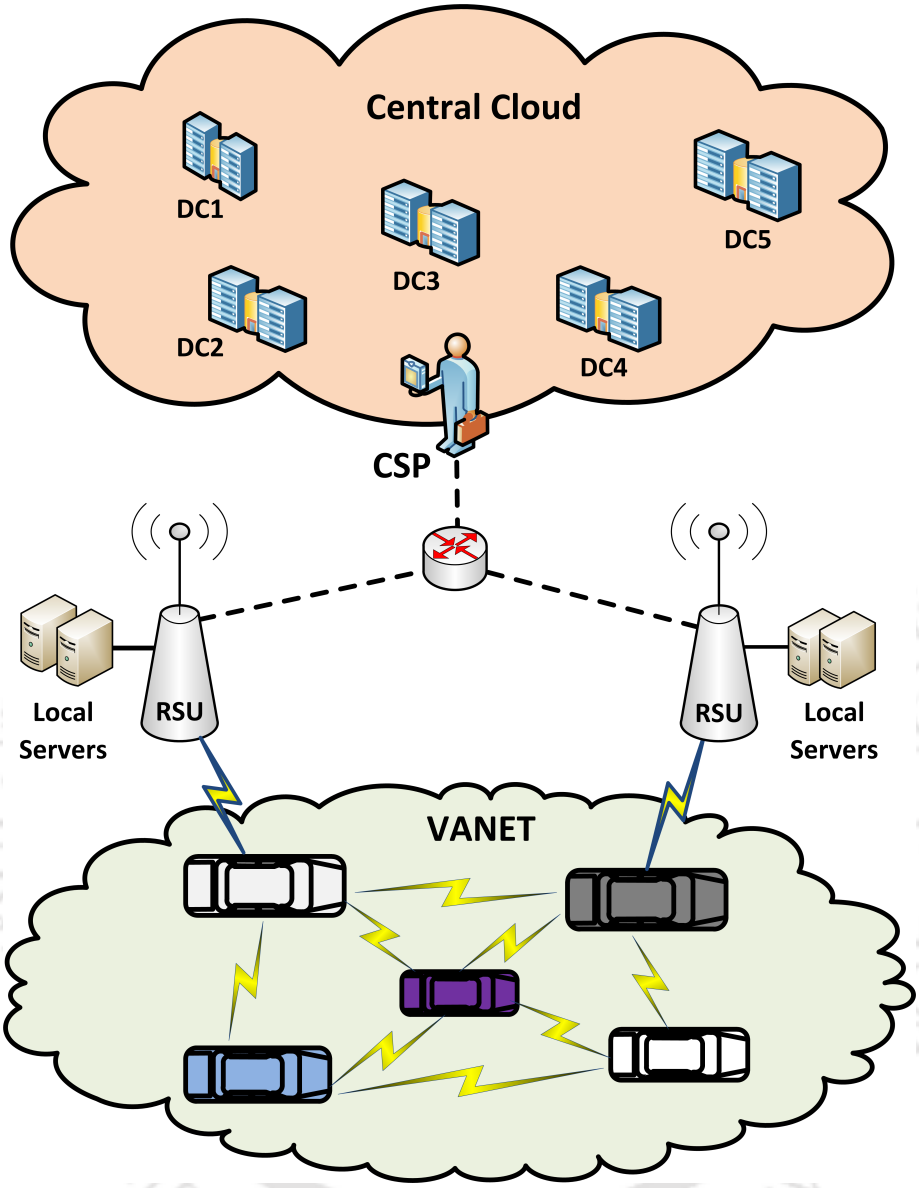



Figure 2.3: Vehicular cloud Computing architecture

applications that otherwise would not work with the limited resources on board [54]. As shown in Figure. 2.3, Vehicular Cloud Computing (VCC) is the technology in which VANETs with the gateway Road Side Units (RSUs) and cloud computing are integrated together to host emerging vehicular applications. With the help of VCC, several services such as real-time traffic analysis, remote vehicle diagnosis, real-time navigation, and mobile social networking can be supported [69, 34, 67]. In VANET, vehicles can communicate with each other directly to offer some services such as inter-vehicle video and audio communications [67]. RSUs

may have limited local resources, and they can provide some services such as location-based commercial advertisements [67]. RSUs also act as gateways for vehicles to request vehicular services from the central cloud (or to send data for processing in the central cloud). CSP represents a front-end proxy in the cloud wherein one or several data centers are available at the back-end to serve the user requests. While the front-end is separated logically from the back-end, the CSP may be deployed physically in one of the data centers. Vehicles communicate with each other and with RSUs through wireless channels, while RSUs are connected to the central cloud through the Internet. To demand a service from the cloud, a vehicle sends a request to the nearest RSU, which forwards it to the CSP. In turn, the CSP directs the request to an appropriate data center that has enough resources and meets the delay requirements of the requested service. Moreover, due to vehicle mobility, the resource management in VCC is different from the traditional cloud, as the CSP needs to monitor the delay and might have to migrate the VM allocated to the vehicle to meet the SLA. After processing the request, the data center sends the response back to the CSP, which forwards it to RSU, and finally, the RSU transmits the response to the vehicle. The CSP may also consider various objectives besides the delay while distributing the load across data centers, like load balancing or minimizing the operating cost.

## 2.3 Virtualization and Migration

Cloud providers typically deliver diverse services at all levels. Figure. 2.1 shows the resources managed at each level in the IaaS model. Under this model, end users request VMs in the cloud that are then allocated to physical resources of the data centers. Virtualization and migration are key enablers for flexible and cost-efficient resource management in cloud data centers [16, 70]. In this section, we present an overview of virtualization and the different migration techniques used.

### 2.3.1 Virtualization Technology

Virtualization is one of the main technologies in cloud data centers that expedites the provisioning of virtual resources to the end users [41]. It facilitates dynamic sharing of physical resources, allowing various applications to run separately in different performance-isolated platforms called virtual machines on a single physical machine as shown in Figure. 2.4 [16].

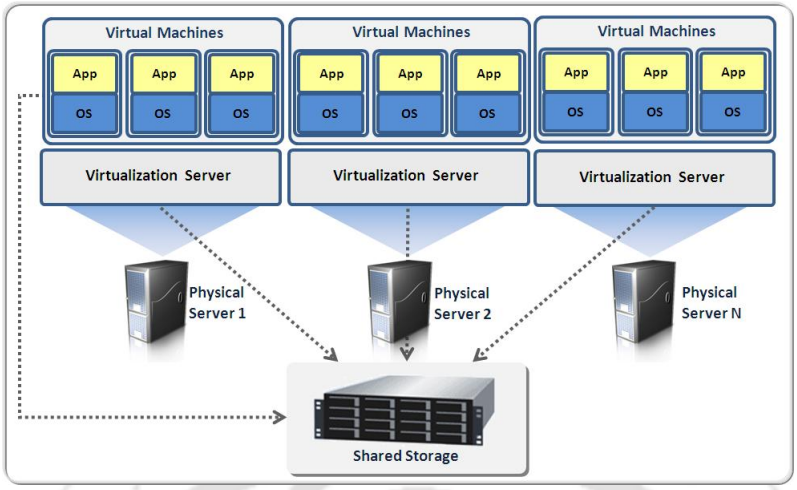
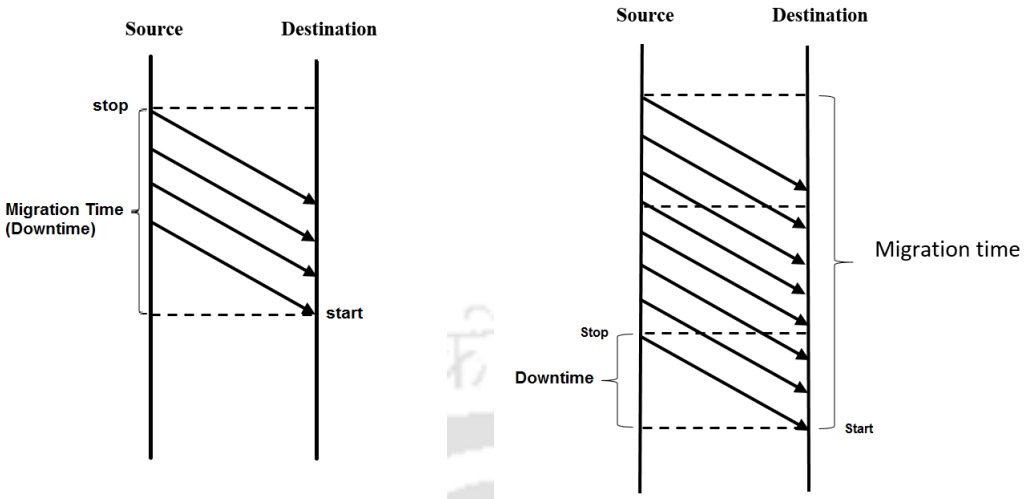



Figure 2.4: Resource virtualization

A VM is typically a composition of multiple configurable resources (CPU, memory, storage, and bandwidth). The resource requirements of a VM may fluctuate over time, and the CP calculates the cost accordingly based on pay-as-you-use. In a public cloud setting, it is common that the CP offers standardized types of VMs. Some CPs allow the user to select among predefined types of VMs only [61, 71], while others allow costumers to specify their own VM configuration giving them more flexibility [60]. In addition, CPs may have different pricing strategies such as price for a long term VM, on-demand pricing according to usage, and discount price for spot instances [60]. In this thesis we consider on-demand pricing for all the cost models.

### 2.3.2 VM Migration Techniques

Migration allows VMs to move from one physical machine to another one with a little downtime of the service hosted [59]. It transfers VM image, which includes CPU current state and memory, across the network [20]. It may also need to transfer the storage of the VM if the migration is happening across different data centers. Basically, there are two migration technologies; *cold* (non-live) migration and *live* migration, as illustrated in Figure. 2.5 [20]. In cold migration, the VM is stopped on the source, and then the VM configuration file is transferred to the destination. Thereafter, the VM starts again on the destination. However, this is not common in cloud data centers. In live VM migration, the VM is migrated without shutdown [62]. However, because the migrated VM might




(a) Cold Migration (b) Live Migration

Figure 2.5: Performance of VM migration technologies

have executed write instructions during the migration, the *dirty pages* or modified pages should be copied to the destination. There are various implementations for live migration technology such as pre-copy, post-copy, and hybrid-copy [20]. We will explain the technique often used in cloud data centers, namely the pre-copy live migration. It includes two stages, the pre-copy stage and the stop-and-copy stage [20]. In the pre-copy stage, the migration process starts by transferring the basic VM image while the VM is still running on the source. Then, dirty data is transferred in an iterative manner up to a defined threshold (termed *termination condition*) which can be set based on one of the conditions: (i) the number of iterations reaches a predefined value, (ii) the size of remaining dirty data is less than a specified value, or (iii) the ratio between the size of the copied data to the size of the VM basic image is larger than a predefined value. It is worth noting that there is no specific termination condition that can suit all VMs because they run different workloads with diverse data modification patterns. Therefore, an efficient migration strategy usually defines the termination condition as a combination of several conditions to suit most of the VMs. The second phase is the stop-and-copy; it starts when the termination condition is reached. In this phase, the VM is suspended, and then the remaining dirty data are transferred. After this, the VM is resumed on the destination, while the VM at the source is discarded. This migration technique is implemented in most of the modern virtualization platforms, such as VMWare ESX (VMotion), Oracle VirtualBox, and Microsoft Hyper-V [20].

As seen from Figure. 2.5, cold migration is much faster, easier, takes smaller migration time with low network traffic, but the downtime is equal to the migration time. On the other hand, live migration needs longer migration time and incurs larger network traffic, but has smaller downtime. Due to high availability requirements, live migration is commonly used in cloud data centers.

Figure. 2.6 shows the basic steps of pre-copy live migration between data centers which includes six stages [20, 72].

1. Stage 1: *Pre-Migration*. After selecting the VM to be migrated, the destination data center is selected for migrating the VM, according to the objective of the cloud provider (e.g., minimize the cost, minimize the delay, etc.).

2. Stage 2: *Resource Reservation*. A VM with the same resource configuration as the migrated VM is reserved in the destination data center.

3. Stage 3: *Iterative Pre-copy*. The data of the migrated VM are copied iteratively to the destination data center. The entire VM image (including memory and storage) is transferred in the first iteration, and then the dirty data are transmitted in successive iterations.

4. Stage 4: *Stop-and-copy*. When the iterative pre-copy loop meets one of the defined termination conditions, the migrated VM is suspended in the source data center, and the remaining dirty data are completely transferred to the destination data center. At this point, both the source and the destination data centers have the same copy on the migrated VM. In case of migration failure due to some reason, the migrated VM can be resumed in the source data center.

5. Stage 5: *Commitment*. The destination data center informs the source data center that it has received the migrated VM successfully, and then the source data center can release the resources reserved for the migrated VM.

6. Stage 6: *Activation*. The migrated VM is resumed in the destination data center.

The performance of live migration is typically measured by the following metrics:

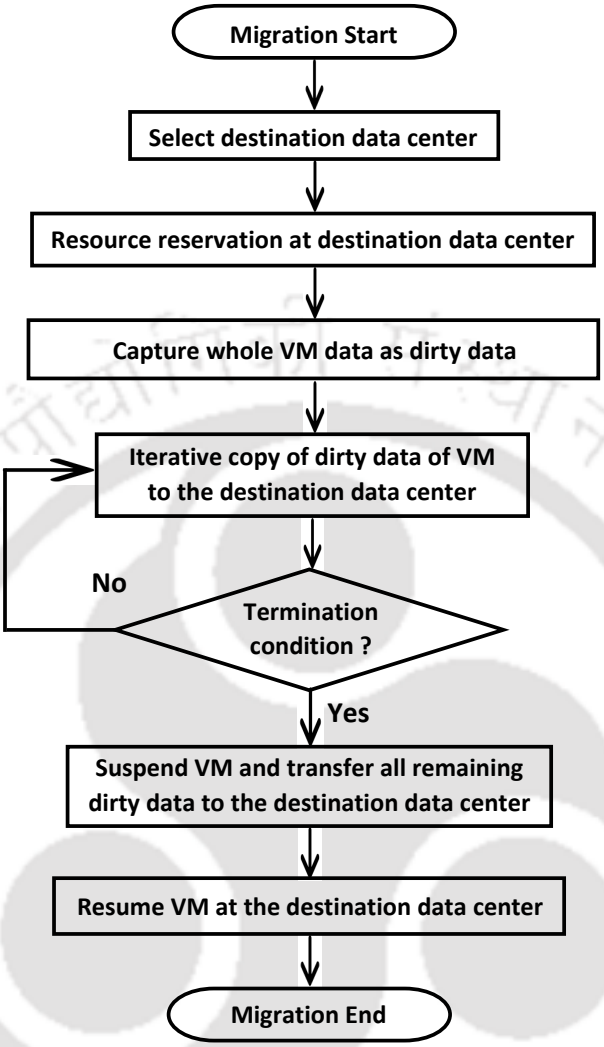



Figure 2.6: Basic steps of inter-data center pre-copy migration technique

- Total migration time: It is the duration between the time when the migration process starts and the time when the source VM can be discarded. So, it is the total time to transfer the VM from the source to the destination.

- Downtime: It is the duration for which the VM is non-responsive, i.e., *stop-and-copy* stage in our case.

- Total network traffic: It is the total data transmitted during the migration process. It is critical as it is related to the workload feature (page dirty rate) and the bandwidth between the source and the destination data centers.

## 2.4 Taxonomy of Resource Management Problem

Resource management problem in cloud is all about the allocation of VMs according to the objectives of the cloud provider and subsequent migration to optimize the resource allocation [70]. The literature on the resource management problem is based on the considered system model, including the hosted application, the system configuration, the objectives of the problem, and the resources considered into account. In addition, some problems address only static resource allocation while others consider changing the allocation through migration as well. In this section, we present a comprehensive taxonomy of the resource management problem in the cloud, as shown in Figure 2.7.

### 2.4.1 Cloud Applications

Cloud data centers have become the preferred hosting platform for a wide range of applications, including video streaming, big data analysis, social networks, high performance computing (HPC), and scientific applications. These applications have different features. Based on the resource demand, applications can be classified into four classes [44]:

- Data-intensive applications: These applications handle massive amounts of data with enormous data transfer, e.g., big data analysis [28]. In this case, the storage constraints might be a bottleneck rather than computing. Besides that, data locality should be maintained in such a way data nodes and associated computing nodes should be allocated close to each other [53].

- Communication-intensive applications: In these applications, the network bandwidth may become a bottleneck, e.g., video sharing where each request generates a new video streaming [73]. The resource management strategy should reduce the latency and minimize the network traffic.

- Computation-intensive applications: Scientific and HPC applications for solving advanced mathematical problems require high computing resources, but they generate very less network traffic. Usually, the resource management strategy for these applications focuses on consolidating VMs in a minimum number of servers to reduce energy consumption [40]. However, load balancing should be considered to avoid performance degradation.

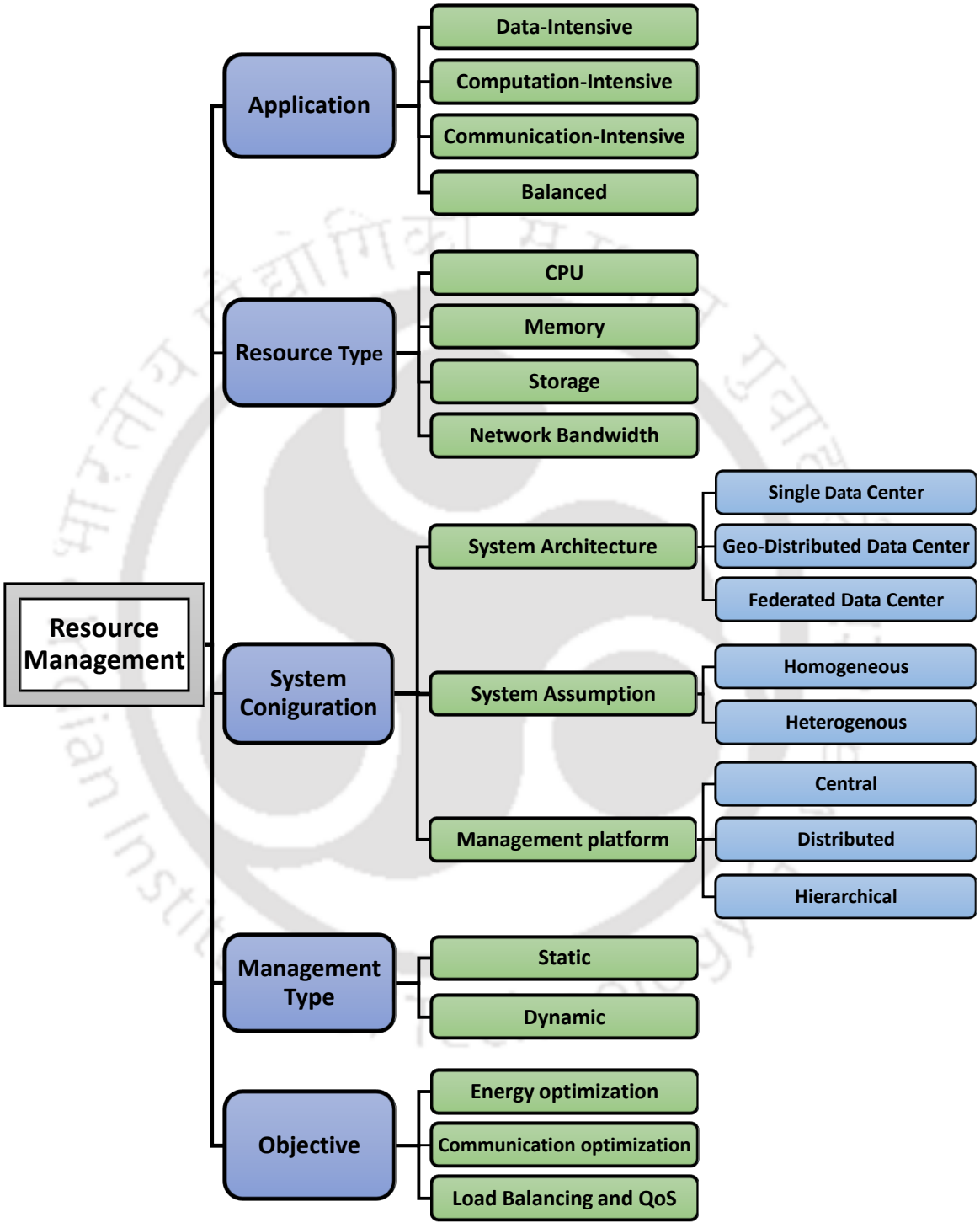



Figure 2.7: Taxonomy of resource management

- Balanced applications: This type of application requires a moderate amount of storage, computing, and bandwidth, such as GIS (Geographic Information System), which needs to read a large amount of data and do heavy processing. Here, resource management should balance the load of various resources [24].

Since different applications have diverse resource demands and QoS requirements, a practical resource management strategy must cater to the application's requirements.

### 2.4.2 Resource Types

Cloud providers have physical resources, and end users request them in units of VMs. Resource management handles the allocation of VMs to the physical resources. VMs may have various resource demands, where a VM is usually characterized by the number of CPU cores, memory, and storage resources. The users may also configure the required bandwidth. Cloud providers offer various types of VMs with different combinations of resources to meet the requirements of applications, such as compute-optimized, memory-optimized, and storage-optimized VMs [61]. In a data center, computation and storage resources may be combined in one server or may be separated, e.g., in Amazon Elastic MapReduce (Amazon EMR), computation is performed by Amazon EC2, while data is stored in Amazon S3.

Resource management is generally a bin packing problem. Physical machines are assumed to be the bins, and the VMs are considered the items that need to be packed into the bins while considering resource availability. The problem is usually formulated as an optimization problem with a specific objective (or multiple objectives) subject to a set of constraints, including resource availability constraints and other constraints (e.g., latency is less than a threshold) [74]. Most of the works in literature formulated resource management as a single dimension problem, taking into account only the CPU resources. However, according to the hosted application, other resources might be a bottleneck, as discussed above. Therefore, the problem of resource management in a cloud data center is a multi-dimension optimization problem that needs to consider multiple types of resources based on the application [75]. Different problem formulations consider various resources. Most of the papers formulate the problem as a single dimension problem, taking into account only the CPU resource. In contrast, others formulate it as a multi-dimensional problem with diverse resources.

In a federated cloud data center, the cloud provider can rent resources from other cloud providers and offer them to its users. Hence, the resources available to a cloud provider in a federated system can be of two types:

- *Local resources:* It includes the resources owned by the cloud provider. These are *whitebox*, i.e., the cloud provider has complete details about their status (utilization, power consumption, etc.), and it is responsible for managing them.

- *External resources:* It includes the resources that are rented from other cloud providers. These are *blackbox*, i.e., the cloud provider only has an interface to request and manage VMs without knowing the underlying infrastructure of external cloud providers.

Therefore, the problem of resource management in a federated cloud is how to divide the workload among local and external resources to maximize the profit and minimize the cost while considering the QoS of various applications [38].

### 2.4.3 System Configuration

In the literature, the resource management problem is also handled based on the system configuration considered. For each setup, a different variant of the problem is addressed. Based on the system configuration, we will discuss some of the categories of resource management problems in the literature.

The problem of resource management in a single data center has been widely explored in the literature. In this context, the typical objective is optimizing resource utilization to minimize energy by minimizing the number of working servers [40, 41, 75, 62]. The servers in a data center are connected over a high-speed LAN, wherein the network bandwidth is not considered to be a limited resource. On the other hand, in a geo-distributed data center, the cloud provider can take advantage of spatial and temporal diversity in electricity prices across different locations. Here, the common objective is to choose a subset of data centers with the lowest electricity price to reduce the cost [45, 10]. In a federated data center, due to the possibility of resource sharing between cloud providers, the problem gets more challenging. The problem here is how to divide requests among local and external resources to maximize the profit while considering various factors (e.g., availability of shared resources, cost diversity, latency, inter-data center bandwidth and the cost of data transfer) [14, 16, 24].

Usually, the problem of VM allocation is modeled as a homogeneous problem where all servers have the same specification, and all VMs have the same resource demand [76]. In practice, the servers may have different types of resources (e.g., in Amazon EMR, computing and storage are separate). Also, the data centers are gradually updated and expanded with recent equipment. Federated clouds have large heterogeneity in terms of the type and number of resources [16].

In a single data center, the resources can be usually managed in a centralized manner. However, it does not scale up for geo-distributed and federated data centers. A common approach is to use a distributed platform which requires all-to-all communication between various resource containers (servers or data centers) [77]. This might work with traditional geo-distributed data centers but not with federated clouds where the semi-autonomy must be preserved, and the resource sharing and access should be decided by the cloud provider. The hierarchical approach is yet another intermediate solution where the resources are managed at various levels.

### 2.4.4 Types of Resource Management

Resources in a cloud data center are allocated either statically or dynamically. Dynamic resource management generally includes initial VM placement and re-optimization [38]. The initial resource allocation is performed when a new request is received, while the re-optimization is usually carried out using VM migration due to change in some environment's conditions such as the server load, the VM demand, electricity price, or external resources price (in a federated cloud) [45, 63]. Dynamic resource management is more suitable for some applications. For instance, migration is very important to handle user mobility in vehicular applications [66]. However, migration creates additional overhead, and it is not a preferred choice in some cases like data-intensive applications where the static resource allocation is more suitable [48]. Moreover, inter-data center migration is a very critical process. Therefore, considering the impact of migration is essential, and its use in resource management should consider the system architecture and the applications.

### 2.4.5 Resource Management Objectives

In the literature, resource management works have considered various objectives ranging from energy optimization to communication optimization with respect to QoS [74]. We briefly describe the common objectives which have been addressed the most in the literature.

**Energy optimization:** Due to its high cost and environmental impacts, the optimization of energy consumption has attracted the attention of the research and industry. Minimizing energy consumption is the most popular objective in a single data center. It is usually done using some techniques such as DVFS (dynamic voltage and frequency scaling) which scales the CPU frequency up and down to improve energy consumption and the performance [78, 79], and VM consolidation which consolidates VMs on a minimum number of servers and switches off the idle ones or turns them to sleep mode [40, 41, 75, 80]. In large-scale cloud systems, data centers are different in terms of energy efficiency which is measured by power usage effectiveness (PUE). This metric is defined as the ratio of the total power entering the data center to the amount of power used by IT equipment.

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power} \tag{2.1}$$

From the literature, we have observed that the PUE of data centers ranges between 1.1 to 2. When PUE is equal to 1, it means the data center is perfect in consuming energy, and there is no energy wastage, while PUE of 2 means that half of the consumed energy is wasted in cooling and lighting. In geo-distributed and federated cloud data centers, energy consumption can be reduced by selecting the data center with the lowest PUE [48]. Further, electricity price variation presents a major advantage of distributed data centers, and it helps cloud providers significantly to reduce their operational costs.

Energy consumption is modeled using various models, such as linear, non-linear, and table-based models [81]. The linear model is the most common one, which models energy consumption of a server as a function of its CPU utilization as

$$ServerPower = P_{idle} + U_c\,(P_{max} - P_{idle}) \tag{2.2}$$

where $U_c$ is the current utilization of the CPU, and $P_{idle}$, and $P_{max}$ are the power consumed

by the server when it is idle and at its maximum utilization, respectively. However, memory or storage might sometimes become a bottleneck in power consumption; for example, in data-intensive applications, storage consumes 40% of the total energy consumption. Moreover, cooling systems also contribute about 50% of the total energy in the data center [50]. Therefore, the hosted application should also be considered into account for modeling the energy consumption.

**Communication optimization:** It is another common objective for resource management. Some works addressed this objective in a single data center by allocating VMs to the same server or ones within the same rack or connected to the same switch to minimize network traffic [82, 83]. However, servers within a single data center are connected through a high bandwidth LAN [84], and the communication cost can be ignored. Moving to distributed data centers connected through WAN links with limited bandwidth, VMs of an enterprise application could be placed in multiple data centers. Thus, the optimization of inter-data center communication becomes a challenge. Moreover, some applications such as big data analytics need to optimize communication. Further, in a federated cloud data center, the problem becomes more challenging as there are different network providers that offer diverse bandwidth and data transfer prices.

**Load balancing and QoS:** QoS is very critical for some applications, such as safety messages in VCC, and violation of QoS may cause additional cost to the cloud provider [85]. For instance, Amazon lost 1% due to 100 ms rise in response time, and Google incurred a 20% reduction in revenue when the response time increased by 500 ms. QoS requirements are typically expressed through Service Level Agreement (SLA), an agreement between the cloud provider and the end user regarding expected service quality (e.g., response time, latency, availability, etc.). SLA may be hard or soft, where the hard SLA must be satisfied, while the soft SLA should be satisfied as much as possible. In case of an SLA violation, the cloud provider has to pay the penalty to the user. Hence, the QoS requirements should be considered carefully during the problem formulation while optimizing other objectives (e.g, minimizing the cost). The hard SLA must be modeled as a constraint, while soft SLAs should be included in the objective function. Also, resource management strategy must balance the load and avoid overload status to minimize the number of SLA violations.

Further, for some applications with mobile users (e.g., vehicular applications), dynamic resource management should be utilized to handle the user mobility and maintain the QoS [18]. In geo-distributed data centers, QoS is a challenging problem, due to diversity in network latency and limited choices of data centers. Federated clouds represent a promising platform for meeting QoS where the cloud provider has more options. It can allocate (or migrate) a VM to a data center of another cloud provider near to the user to minimize the delay and response time.

Indeed, the aforementioned objectives are not fully independent and all of them contribute to the cost. The ultimate objective of the CP is maximizing the profit by reducing the cost while meeting QoS. Accordingly, the CP would typically need to solve multi-objective problem to optimize the various objectives.

## 2.5 Related Work

In this section, we will present a survey of the literature in the area of resource management, which are closely related to the problems addressed in our thesis.

### 2.5.1 VM Allocation and Resource Management in Cloud Data Centers

Minimizing the TCO is a primary objective for the cloud provider. The TCO in a federated cloud has mainly two components: the cost of energy consumption and the cost of network communication. We review some important papers that have addressed these cost components as an objective. To the best of our knowledge, there are very few papers that considered both energy consumption and network bandwidth costs together while optimizing the TCO of the cloud provider.

**Energy-aware Resource Allocation:** In modern data centers, energy efficiency has become an important research topic; the reduction of operational costs, generated heat and environmental impact are some of the reasons for this [74]. Next, we discuss the problem of energy-aware resource allocation within a single data center and in large-scale data centers. *Energy-aware resource allocation in a single data center:* The problem of VM allocation is mostly explored for the data center at a single site with minimizing the energy consumption and maximizing the resource utilization as an objective [86, 75]. In fact, data centers

are usually designed for peak traffic. However, the average data center workload stays around 30% most of the time, causing unnecessary energy wastage. VM consolidation is widely used to allocate VMs on a minimum number of physical servers and to switch off idle servers [87, 88, 89, 90]. In [91], VMs that have a complementary temporal traffic requirements are packed in a group. Then, VMs of the same application or group are placed in nearby servers to maximize resource utilization and minimize the number of active servers. However, only homogeneous VMs are considered for the study. In [80], 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. In [92] and [93], VM migration was proposed to relocate the live VMs to reduce the number of working servers. HEROS [83] examined the trade-off between energy saving and load balancing in heterogeneous data centers. It allocates VMs to the server, which gives the maximum score value; the score is the product of the server selection function and the communication potential function. The server selection function chooses the most energy-efficient server, considering heterogeneity, while the communication potential function is a decreasing function with increasing link load. HEROS has been evaluated using GreenCloud simulator, and results demonstrated significant energy savings up to 46%. Authors in [84] considered data centers with heterogeneous bandwidth focusing on the fact that VM migration could be faster by using higher bandwidth, and that leads to earlier shutdown of the under-loaded servers and saving energy as a result. All these works aim to reduce the energy consumption within a single data center, where there is no diversity in the electricity price, and the network bandwidth is not a major issue.

*Energy-aware resource allocation in a geo-distributed data centers:* In context of large-scale data centers, the problem becomes interesting because of spatial variation in the electricity price across data centers. Kumar *et al.*, in [94], proposed a meta-heuristic approach for VM scheduling in a large-scale data centers 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 [95], considered distributed data centers 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 [47], an algorithm was

proposed to reduce the total energy cost in a federated cloud platform by migrating VMs to a data center where the electricity price is lower while considering migration cost. However, this work did not study large data migration. In [96], the authors have explored the problem of minimization energy cost for geo-distributed interactive services subject to latency. They proposed McTail; an efficient algorithm for determining and meeting latency constraint while exploiting electricity price variation between data centers to minimize the cost. The latency has been calculated including the network delay between the source of the traffic and the destination data center, and the processing latency within the data center. Evaluation of McTail has been done using traces from Google and Facebook, and results demonstrated 7% reduction in energy cost while latency requirement satisfaction is more than 95%. Authors in [58] presented an algorithm to reduce total energy cost in a federated data center platform by migrating VMs to a data center with enough resources and where electricity price is cheaper. They assumed that the hosting fee is determined based on electricity price, and all data centers have same PUE. However, PUE may differ due to the difference in temperature and cooling systems. Moreover, considering only electricity price is not enough, and PUE has to be taken into account as well to guarantee energy efficiency. In [48], the authors proposed two algorithms for initial VM allocation and for VM migration in distributed cloud data centers to minimize the energy cost. They also reported the trade-off between load balancing and energy cost. However, they did not consider inter-data center communication, and the cost of VM migration. Note that inter-data center VM migration is a costly operation [97], especially for some applications, such as data-intensive applications where a large amount of data needs to be migrated [48, 44].

From the literature, we can observe that leveraging the variation in electricity price across data centers of the federation can achieve a significant saving in the cost of the CP. Moreover, PUE of the data center plays an important role in the overall energy consumption as well.

**Network-aware Resource Allocation:** In fact, a VM is usually not an independent component but a part of an application wherein VMs need to communicate with each other to perform a particular task, e.g., multi-tier applications and big data analytics. Because of this, it is useful to allocate correlated VMs together in the same server or data center since this will reduce the network traffic and improve the performance [74]. In [82], the

authors proposed VMP, an online algorithm for VM allocation in a single data center with a three-tier topology. The objective is reducing the traffic in higher network layers by consolidating VMs with massive communication on the same server or, if it is not possible, on closer servers within the same rack. VMP creates partitions of highly connected servers and forms clusters of VMs based on traffic. Then, the clusters are allocated to servers using a bin-packing technique considering CPU and memory resources. Simulation results have shown that the VMP can reduce the traffic in the core network layer up to 80%. However, this work has been done for a specific topology, and the algorithm used for community finding (Girvan-Newman) is CPU intensive.

In geo-distributed data centers, the challenge is how to select a relatively small number of high-capacity data centers, such that the inter-data center distance is small. In NACER (Network-Aware Cost-Efficient Resource allocation) [98], the authors have addressed this challenge and tried to minimize inter-data center communication cost by reducing the number of selected data centers and the total distance between data centers as well. They proposed two algorithms based on A* algorithm; NACER1 only considers network distance between data centers, and NACER2, which also considers inter-VM communication. Although the algorithm can find a globally optimal solution, VMs are assumed to be homogeneous. The authors did not consider the response time, which is critical in many applications. Moreover, they did not consider the cost of computing resources and price variation across data centers which highly affects the operating cost. In [99], the relationship between the response time and the number of selected data centers has been examined. The authors have modeled the response time as the sum of the latency and the data center's processing time. Accordingly, they formulated requests allocation as an optimization problem with an objective to minimize the response time. Similarly, the problem of minimizing job completion time for a big data application has been studied in [100]. The authors formulated it as a nonlinear optimization problem and proposed an approximation algorithm. Simulation results demonstrated the efficiency of the proposed algorithm showing that it gives near-optimal solution within a very short time. Because the prices of resources and the network bandwidth vary across data centers, the streaming workflow allocation has been addressed in [73]. The problem has been formulated as MILP (Mixed Integer Linear Programming) problem with the objective of cost minimization and solved optimally using GLPK solver. To reduce the complexity

and the search space, two heuristics have been proposed; one for grouping tasks according to density of input and output, and another for grouping data centers based on the region.

In summary, we observe that minimizing the communication between data centers is a major objective for cloud providers in resource management. It reduces the bandwidth usage over WAN links and its associated cost. It also has a high impact on the performance and the delay of application requests, especially for enterprise applications distributed across multiple data centers in the federation. Earlier works mainly focus on the energy consumption cost of independent VMs without considering the traffic demand between them and the cost of inter-data center communication, which would largely affect the operating cost of a cloud provider in a federated platform [101]. On the other hand, some works have addressed the problem of minimizing the network traffic and placing correlated VMs together in the same server or the same data center, while they ignore the cost of energy consumption. While most previous works consider either energy or communication costs, we consider both of them in this thesis.

### 2.5.2 The Design of Resource Management Framework

Cloud systems are mostly managed using centralized approaches [102]. In [95], it was assumed that the cloud provider consists of several data centers 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 resources, which is not compatible with a federated system where the autonomy of different cloud providers for internal management makes it difficult to maintain a central database. Similarly, in [16], the authors introduced a central third party that is responsible for resource management and communication with the cloud providers in the federation. Again, this model suffers from a single point of failure where the entire federation fails if this central manager fails. In [48], a distributed approach for workload management across multiple data centers in different geographical areas was proposed. Every data center has a manager, and all the managers exchange information for dynamic VM allocation. This approach requires periodic all-to-all communication between data centers, increasing the communication overhead and the WAN traffic. It is not suitable for a federated cloud where collaborating data centers may be large in number and dispersed geographically. In summary, the existing management platforms are not applicable for a

federated cloud environment where the number of data centers is large, and the privacy of cloud providers should be maintained. Therefore, we design a hierarchical resource management framework that is suitable for federated cloud systems.

### 2.5.3 Placement of Data-intensive Applications

As discussed earlier, data-intensive applications consist of two types of virtual components; data blocks and compute VMs with dense communication among them. In what follows, we summarize some works on the placement of data-intensive applications in a cloud data center.

**Placement of data-intensive applications in a single data center:** The placement of virtual components for data-intensive applications should consider the traffic demand between the application components. To deal with this issue, Zhang *et al.* proposed a VM placement algorithm for data-intensive applications in a single data center to minimize the data access latency [103]. 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 data center, considering the communication pattern between the virtual components [104]. 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. The algorithm starts by allocating every data block with associated compute VMs, and then it places the remaining VMs. Simulation has been done using three-tier data center architecture, and the results showed a significant reduction in network traffic at core switches. However, simulated applications are composite of a small number of components, less than ten. The algorithm allocates couples of virtual components, and it has quadratic time complexity. Thus, it is not suitable for applications with a large number of virtual components. Further, it minimizes the network traffic within a data center without considering resource utilization and energy consumption cost. These works considered single data centers where there is no spatial variation in electricity prices and bandwidth costs.

**Placement of data-intensive applications in large-scale data centers:** There are a few works that addressed the problem of deploying data-intensive applications in a geo-distributed cloud. The work in [105] addressed the placement of data blocks and compute VMs in distributed data centers. They proposed a meta-heuristic approach based on Ant Colony Optimization (ACO) to minimize network distance by selecting a subset of physically closer servers 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 data centers, considering data and task placement along with data transfer [76]. The problem has been formulated as MILP problem with the objective of minimizing the number of active servers and the 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 data centers have an equal number of servers and allocated the VMs without considering cost variation across the different data centers. Xiao *et al.*, studied the distributed deployment of MapReduce workload across geo-distributed data centers to minimize costs and latency [106]. 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 valid only for MapReduce applications. In [53], the authors proposed an algorithm that schedules data and associated tasks in the same data center to reduce the latency and the data transfer across the data centers while achieving load balancing. However, they focused only on reducing inter-data center network latency to improve the performance of data-intensive applications, while they did not consider the cost-efficiency of resource allocation. The authors of [107] introduced a framework for processing big data applications using federated clouds, where a pricing strategy offered the users options based on the quantity and the time of renting resources. The resource allocation was modeled as a Stackelberg game to maximize the revenue of the service provider.

Most of the literature focused on either minimizing the energy consumption or the network traffic of data-intensive applications. 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 cloud provider.

### 2.5.4 Resource Allocation for Modern Vehicular Applications

The problem of resource management in vehicular cloud computing (VCC) has recently gained a lot of attention from researchers [54, 108]. However, the federated cloud has not been explored earlier in VCC. In addition, vehicular cloud applications are delay-aware. In what follows, we discuss some works related to resource allocation in VCC (without federation) and delay-aware resource allocation in cloud data centers.

**Resource allocation in VCC:** The work in [109] modeled the resource allocation problem in VCC using semi-Markov decision processes (SMDP) to optimize the long-term energy consumption cost. However, this work did not consider latency and vehicle mobility. The authors of [110] formulated a multi-objective optimization problem to reduce the response time of the request and increase the throughput of the system in VCC. They proposed a hybrid meta-heuristic resource allocation approach based on genetic algorithm and particle swarm optimization. However, this work neither considered the operating cost nor handled VM migration required to maintain the desired QoS level. In [111], a VM migration policy had been proposed to minimize the cost due to energy consumption in VCC system while considering user mobility. Though the energy consumed by various components of the system was considered, the network latency was ignored. The authors of [78] integrated Mobile Edge Computing (MEC) in VCC and formulated the resource allocation problem with the objective of minimizing the energy consumption cost of the server and the vehicle. They proposed an algorithm to solve the problem using Lyapunov optimization. The algorithm compares the cost of processing a task locally in the vehicle with that of offloading to the cloud server to jointly optimize the offloading decision and adapt the vehicle's CPU frequency. However, they did not consider the delay requirements and the mobility of the users. Similarly, the *MoBQoS* framework discussed in [112] used software defined network (SDN) and edge computing in VCC to handle mobility and QoS. The resource management problem was designed to trade-off bandwidth allocation and network latency without considering the operating cost and VM migration. The authors of [33] used fog computing in traditional VCC, where the resource allocation algorithm maximizes energy efficiency while meeting hard QoS constraints of real-time applications. It configures the processing rate of VMs and performs VM consolidation to minimize energy consumption. It

applies admission control for incoming service requests to meet QoS requirements. Though the algorithm works for mobile users, it does not consider VM migration that minimizes the energy consumption. Moreover, this work is meant only for delay-sensitive applications with hard QoS constraints.

**Delay-aware resource allocation in cloud data center:** Some works have addressed the problem of delay-aware resource management in traditional cloud platforms. In [113], the authors discussed the problem of resource allocation for delay-sensitive applications in a cloud system that contains multiple cloud providers and multiple users. The problem has been modeled as a hierarchical non-cooperative game where each user chooses the data center with the minimum delay at the lowest price. In [114], an algorithm has been proposed to distribute the application workload across a multi-tier cloud system based on their resource demand and delay sensitivity. Similarly, the authors of [115] addressed the problem of delay-aware resource allocation in edge computing. They classified requests into two categories (delay-sensitive and delay-tolerant) based on the accepted delay threshold. Some of the applications are hosted in edge servers, and others are outsourced to the cloud according to the type of the application. The work in [116] proposed various scheduling strategies in hierarchical cloud architecture to reduce the latency but did not consider the cost. In [117], the problem of scheduling delay-sensitive tasks in vehicular fog computing has been formulated to minimize the delay that includes transmission delay and computation delay. Authors in [85], integrates edge computing with VCC to reduce the latency for mobile devices. In [118], the authors addressed the problem of processing multiple tasks on a hybrid cloud platform which composed of various heterogeneous cloud providers having limited storage and computing resources with different performance and prices. The problem has been formulated as a non-linear optimization problem with the objective of minimizing the cost subject to a deadline constraint. It has been modeled using AMPL (A Mathematical Programming Language) and solved by CBC solver. Evaluation has been done for data-intensive and computation-intensive tasks separately to study the relationship between cost and deadline. However, all tasks are assumed to be independent and homogeneous. None of these works have considered user mobility which is an essential feature in VCC. *Follow Me Cloud* platform proposed in [18] handles mobile users through service migration. An algorithm based on Markov Decision Process (MDP) moves the service to the optimal data

center to minimize the delay while considering migration cost. This work only aims to reduce the user's response time without optimizing the operating cost for the cloud provider.

None of the earlier works explored federation in VCC. We argue and demonstrate that taking advantage of diversity in price across the data centers in a federation helps the vehicular service provider to serve more requests while meeting their QoS requirements and increasing its profit [16]. VM migration can also be utilized in a federated cloud to handle user mobility and meet the assured delay bounds. Most of the works consider minimizing either the delay of the request or the cost of the cloud provider. We consider a trade-off between the delay and the cost to satisfy the objectives of the CP and the end user.

### 2.5.5 VM Live Migration

Many works have addressed the problem of VM migration within a data center for minimizing energy consumption through consolidating VMs on a minimum number of servers and switching off idle servers [40, 41, 75]. The authors of [77] presented a collaborative agent-based platform for efficient heterogeneous workload distribution taking advantage of VM live migration. Agents collaborate in a distributed manner using two protocols to balance VM consolidation while minimizing energy consumption. They have implemented various heuristics for selecting (i) the VMs to be migrated, (ii) the destination server for migration, (iii) when to trigger the migration, and (iv) when to turn on/off a server. In addition, a heuristic for balancing resource usage has been proposed considering CPU and memory. Although this paper presented a novel platform for dynamic resource management, it is designed for a single data center. In addition, the work does not include VM initial allocation and ignores the migration cost. In [40], live migration has been utilized for VM consolidation with the objective of minimizing energy consumption, SLA violation, and the number of migrations, simultaneously.

Some works have addressed VM migration between data centers. In [119], a VM migration technique across data centers with fat-tree topology has been proposed with the objective of energy saving. This technique consists of two algorithms; selection and negotiation. The selection algorithm divides VMs into groups; a group contains VMs residing on sequential servers (connected to the same switch). The negotiation algorithm finds the migration impact (energy reduction due to migration) and the migration time

for each VM group and every data center. The proposed algorithm migrates the groups that give maximum impact and its residual lifetime more than migration time. However, this technique is applicable only for a specific topology. Further, the negotiation algorithm considers only migration time while it ignores the cost of migration.

The work in [120] proposed a model for measuring VM migration energy consumption based on network transfer. the model is limited for CPU and memory-intensive applications. A more accurate model has been proposed to estimate VM migration overhead in [92]. It comprises migration's impact in terms of energy consumption, network cost, and SLA violation. Considering this model, the problem of dynamic VM consolidation has been addressed with the objective of minimizing energy consumption, resource wastage and migration overhead simultaneously. An algorithm has been proposed to tackle this problem based on ACO meta-heuristic. Further, the authors presented a hierarchical framework for VM consolidation management to reduce network traffic. Simulation has been done using CloudSim, and the results demonstrated significant improvement across all objectives. However, this work has been done for a single data center. In [121], the authors presented an analytical model for inter-data center network performance under VM group migration in a federated cloud. They applied Markov chains to model the probability of migration rejection when the bandwidth is not enough. The model includes migration time and downtime. Two cases have been examined; sequential and parallel migration. Results showed that the parallel migration reduces downtime, but the sequential one has very less impact on the network, and by increasing the bandwidth, both will perform better. Although this model is very good for network capacity planning, it considers only memory migration, while storage synchronization was assumed to be done before migration. The work in [122] introduced a model for migration cost within a single data center based on memory size and PDR of the VM and the sensitivity of the hosted application to migration. Based on this model, a VM migration and consolidation algorithm has been proposed to minimize energy consumption. However, both [121] and [122] did not consider the cost of the migration. To minimize the impact of live migration in a distributed data center, an SDN-based QoS model has been proposed [123]. The model implements the DiffServ method, considering four traffic classes and giving the highest priority to inter-data center traffic to speed up the live migration across data centers.

Although the VM live migration within a data center has been extensively studied, inter-data center VM migration in a federated cloud needs more investigation due to the diversity in network bandwidth and data transfer prices. An accurate model to characterize the performance of migration is required. Moreover, most of the works consider migrating homogeneous VMs, which is not realistic.

## 2.6 Summary

In this chapter, we discussed the architecture of a federated cloud data center and the various components associated with it. We also provided a brief overview of VCC. We discussed the concepts of virtualization and the various migration techniques used. We discussed resource management in cloud data centers and presented a comprehensive taxonomy of resource management problems. Finally, we discussed the state-of-art literature review of the problems addressed in the thesis. In the next chapter, we describe the proposed hierarchical resource management framework and discuss the problem of placing the data-intensive applications in a federated cloud.

CHAPTER

# 3

# A Cost-aware Management Framework for Placement of Data-intensive Applications on Federated Cloud

As discussed in Chapter 1, the data-intensive applications are typically deployed in cloud data centers using two types of VCs−DBs as storage nodes and VMs as compute nodes. However, modern big data analysis and business intelligence applications come with high resource demands, along with a requirement for massive data transfer between storage and compute nodes. Due to the large volume of data to be analyzed, dealing with data-intensive applications is a major challenge for the cloud provider as the available resources of a single data center might not be sufficient to host the VCs of an application. Federated cloud systems offer a promising solution for handling such applications in a scalable manner, while avoiding the cost associated with having large infrastructure [30]. The objective of each cloud provider in a federation is to maximize its own profit and to minimize its TCO. The large resources of a data-intensive application consume high amount of energy resulting in large electricity bill. Further, distributing the VCs of an application across different data centers may lead to huge inter-data center network traffic and that increases the cost. Typically, the workload assignment in cloud data centers is handled in a centralized manner, which is not scalable and compatible with the federation requirements [124].

**Motivation:** This work is motivated by the following observations. First, existing cloud management approaches do not consider the semi-autonomous requirements of the federated system. Second, data-intensive applications require a large amount of resources that could

span across multiple data centers, and in this case exploiting federation is a good choice for small cloud providers. Third, cloud providers are mainly interested in maximizing their profit by minimizing their TCO. Energy consumption cost contributes a significant fraction of TCO of the cloud provider. It could be reduced by taking advantage of the electricity price and PUE diversity across the data centers of the federation. In addition, data-intensive applications may trigger large data transfer over WAN, wherein the bandwidth is limited and expensive. Thus, inter-data center communication cost is another critical cost component in data-intensive applications where it could surpass the cost of the energy consumption. That requires maintaining data locality to minimize inter-data center traffic by carefully allocating highly-communicating virtual components together. In the literature, resource allocation usually considers the placement of homogeneous VMs in isolation, which is not suitable when heterogeneous VMs collaborate in an application. Further, most of the works either optimize the energy consumption or the network bandwidth while allocating the resources. However, the cloud provider should minimize the energy consumption and bandwidth costs together while allocating the VCs of a data-intensive application to reduce its TCO.

To deal with these issues, we design a hierarchical framework for resource management and workload allocation considering the features of the federated cloud. Using this framework, we solve the problem of cost-aware deployment of data-intensive applications in federated clouds. We propose an optimization model with the objective of minimizing the TCO including energy consumption cost and bandwidth cost. We 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-data center communication and bandwidth cost, and it leverages the electricity price and PUE variations across data centers of the federation to minimize the cost of energy consumption. The proposed algorithm is evaluated with various scenarios to demonstrate the benefits of federation.

**Organization of the Chapter:** The remainder of this chapter is organized as follows. We first present the hierarchical framework for resource management and workload distribution for federated clouds in Section 3.1. The system model and the problem formulation are presented in Section 3.2. We describe the proposed algorithm to solve the problem, and discuss its complexity and running cost in Section 3.3. Simulation results are reported in Section 3.4 followed by the summary of the chapter in Section 3.5.

## 3.1 Hierarchical Management Framework for Federated Clouds

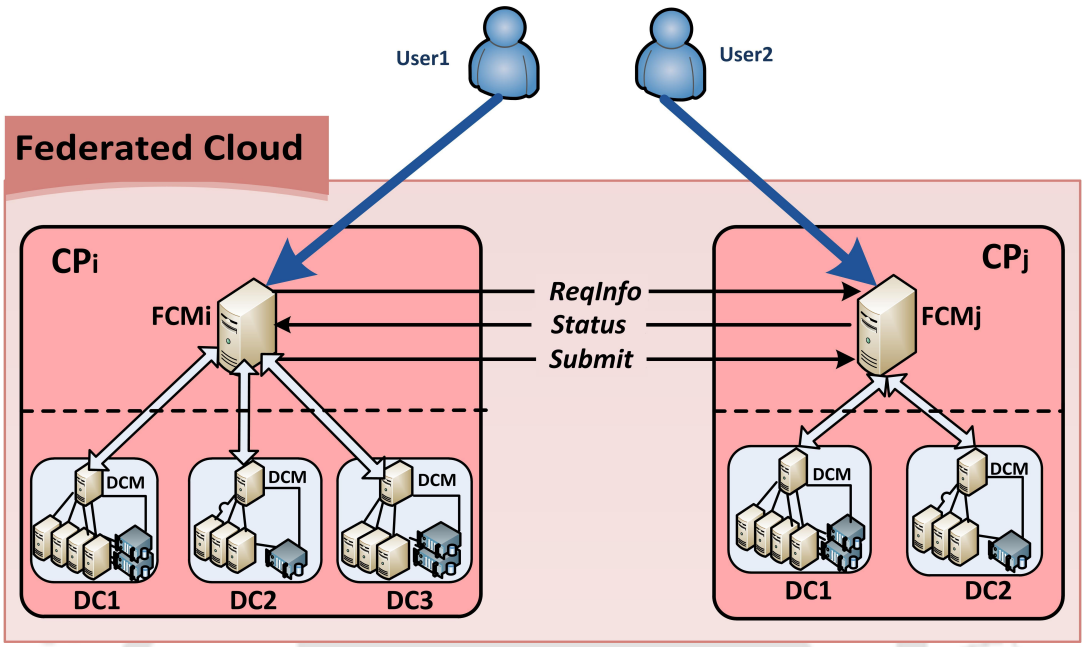



Figure 3.1: Proposed hierarchical management framework for federated cloud system

In this section, we propose a framework for resource management and workload distribution 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 [102]. 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 data centers (DCs) [6]. As illustrated in Figure. 3.1, 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 Figure. 3.1. $FCM_i$ 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.

1. *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 is done by sending *ReqInfo* message with a specific DC identifier.

2. *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 $\{Status_{j1}, Status_{j2}, \ldots, Status_{jn}\}$ of the data centers $\{DC_{j1}, DC_{j2}, \ldots, DC_{jn}\}$ of the $CP_j$.

3. *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:

$$Status_{pk} = \{\mathcal{D}_{pk}, \mathcal{D}_{pk}^{cpu}, \mathcal{D}_{pk}^{mem}, \mathcal{D}_{pk}^{str}, E_{pk}, PUE_{pk}\} \tag{3.1}$$

where $\mathcal{D}_{pk}$ is the $k^{th}$data center of the $p^{th}$ cloud provider, $\mathcal{D}_{pk}^{cpu}$ is the number of CPU cores available in $\mathcal{D}_{pk}$. $\mathcal{D}_{pk}^{mem}$ and $\mathcal{D}_{pk}^{str}$ are the amount of memory and storage available in $\mathcal{D}_{pk}$, respectively. $E_{pk}$ and $PUE_{pk}$ are the electricity price and the PUE at $\mathcal{D}_{pk}$, respectively.

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 has many advantages for the CPs in the federated cloud. It 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.

## 3.2 System Model and Problem Formulation

In this section, we discuss the proposed optimization model for cost-aware placement of data-intensive applications in federated cloud DCs. Table 3.1 summarizes the notation used. First, we state the system model and the assumptions made, before presenting the optimization model.

### 3.2.1 Assumptions

- We consider the initial placement of virtual components assuming that the entire resource requirement of an application is less than the total 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 [53, 125].

- 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 [76].

- FCMs exchange information that includes the DC identifier, the available resources in the DC, the current electricity price, and the PUE (see Eq. 3.1).

Table 3.1: Notation used in the system 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 |
| $D_p$ | Set of the data centers of the $p^{th}$ cloud provider |
| $\mathcal{D}_{pk}$ | $k^{th}$ data center of cloud provider $p$ |
| $\mathcal{D}_{pk}^{cpu}$ | CPU capacity of data center $\mathcal{D}_{pk}$ |
| $\mathcal{D}_{pk}^{mem}$ | Memory capacity of data center $\mathcal{D}_{pk}$ |
| $\mathcal{D}_{pk}^{str}$ | Storage capacity of data center $\mathcal{D}_{pk}$ |
| $E_{pk}$ | Electricity price at data center $\mathcal{D}_{pk}$ (in \$/KWh) |
| $PUE_{pk}$ | Power usage effectiveness at data center $\mathcal{D}_{pk}$ |
| $s_{pk}$ | The number of servers at data center $\mathcal{D}_{pk}$ |
| $P_{pk}$ | Power consumption of data center $\mathcal{D}_{pk}$ (KWh) |
| $DP_{pk,ql}$ | Data transfer price between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ (\$/GB) |
| $BW_{pk,ql}$ | The bandwidth of the link between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ (Gbps) |
| $\mathbb{C}$ | Set of virtual components |
| $\mathbb{V}$ | Set of virtual machines |
| $\mathbb{B}$ | Set of data blocks |
| $VM_i^{cpu}$ | CPU demand of $i^{th}$ virtual machine |
| $VM_i^{mem}$ | Memory demand of $i^{th}$ virtual machine |
| $DB_j^{str}$ | Storage demand of $j^{th}$ data block |
| $x_i^{pk}$ | Virtual component $i$ is assigned to data center $\mathcal{D}_{pk}$ |
| $A_{ji}$ | 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 $v$ |
| $z_v$ | binary variable that indicates if server $v$ is used or not |
| $t$ | The running time of the application (in hours) |

### 3.2.2 System Model

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

**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; $\mathbb{V}$, the set of compute virtual machines and $\mathbb{B}$, the set of data blocks. Each virtual machine $VM_i \in \mathbb{V}$ demands a number of CPU cores, $VM_i^{cpu}$ and memory, $VM_i^{mem}$ (per $GB$), while each data block $DB_j \in \mathbb{B}$ comes only with a storage demand, $DB_j^{str}$ (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. 3.2 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 $DBs$ [53]. 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 $A_{|\mathbb{C}|\times|\mathbb{V}|}$, where $A_{ji}$ is the data volume between $DB_j$ (or $VM_j$) and $VM_i$. We define also a binary variable $x_i^{pk}$ to indicate whether a virtual component $VC_i \in \mathbb{C}$ is placed in the data center $\mathcal{D}_{pk}$ ($x_i^{pk} = 1$) or not ($x_i^{pk} = 0$).

**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

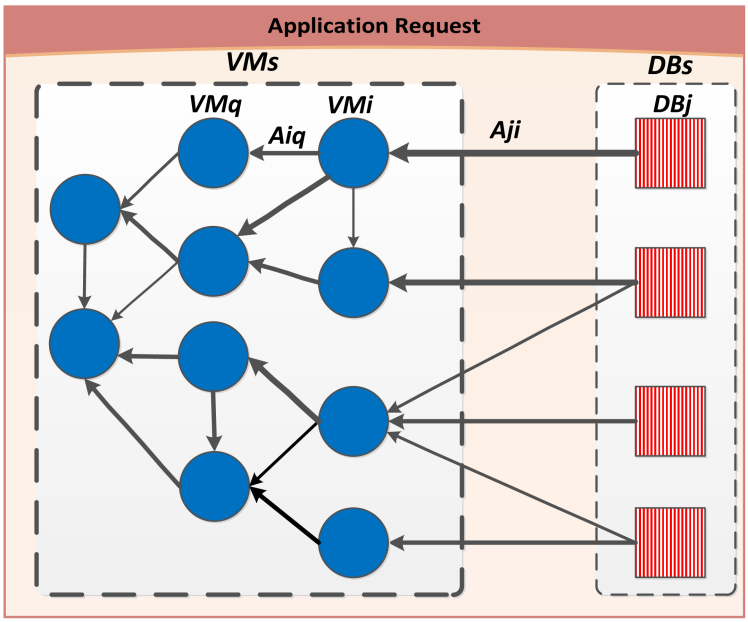



Figure 3.2: Graph representation of a data-intensive application request

energy of a data center for data-intensive applications [50]. Accordingly, in this chapter, we consider CPU and storage to consume a significant portion of power [50]. The total power consumed by a server $v$ is expressed as

$$SP_v = BP_v + \sum_r FP_v^r * \frac{U_v^r}{C_v^r} \tag{3.2}$$

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 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 $\mathcal{D}_{pk}$ 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 [50]. We define the total power consumed (KWh) at a data center $\mathcal{D}_{pk}$ housing $s_{pk}$ servers as

$$P_{pk} = \sum_{v=1}^{s_{pk}} SP_v * z_v * PUE_{pk} \tag{3.3}$$

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

### 3.2.3 Optimization Model

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

**Energy cost:** The cost incurred by power at a data center $\mathcal{D}_{pk}$ can be obtained by multiplying power usage $P_{pk}$ of $\mathcal{D}_{pk}$ (Eq.3.3) with the electricity price $E_{pk}$ (\$/KWh). The energy cost incurred by the federation per hour ($\$/h$) is the summation of the energy consumption cost of all data centers of the federation, and it is defined as

$$EC = \sum_{\mathcal{D}_{pk} \in \mathbb{D}} P_{pk}\, E_{pk} \tag{3.4}$$

**Communication cost:** Comparing to inter-DC bandwidth cost, the communication cost is negligible for virtual components within a DC [53]. We define the communication cost due to data-intensive applications deployed within the federation $CC$ as

$$CC = \sum_{\mathcal{D}_{pk} \in \mathbb{D}} \sum_{\substack{\mathcal{D}_{ql} \in \mathbb{D} \\ \mathcal{D}_{pk} \neq \mathcal{D}_{ql}}} \sum_{i \in \mathbb{C}} \sum_{j \in \mathbb{V}} x_i^{pk} x_j^{ql} A_{ij}\, DP_{pk,ql} \tag{3.5}$$

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

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

$$\textbf{Minimize } (EC + CC)\, t \tag{3.6}$$

**subject to**

$$\sum_{\mathcal{D}_{pk} \in \mathbb{D}} x_i^{pk} = 1, \ \forall i \in \mathbb{C} \tag{3.7}$$

$$\sum_{j \in \mathbb{V}} x_j^{pk}\, VM_j^{cpu} \leq \mathcal{D}_{pk}^{cpu}, \ \forall \mathcal{D}_{pk} \in \mathbb{D} \tag{3.8}$$

$$\sum_{j \in \mathbb{V}} x_j^{pk}\, VM_j^{mem} \leq \mathcal{D}_{pk}^{mem}, \ \forall \mathcal{D}_{pk} \in \mathbb{D} \tag{3.9}$$

$$\sum_{b \in \mathbb{B}} x_b^{pk} \, DB_b^{str} \leq \mathcal{D}_{pk}^{str}, \;\; \forall \mathcal{D}_{pk} \in \mathbb{D} \tag{3.10}$$

$$\sum_{i \in \mathbb{C}} \sum_{j \in \mathbb{V}} x_i^{pk} x_j^{ql} A_{ij} \leq BW_{pk,ql} \;\; \forall \mathcal{D}_{pk}, \mathcal{D}_{ql} \in \mathbb{D} \tag{3.11}$$

$$x_i^{pk} \in \{0,1\}, \;\; \forall i \in \mathbb{C}, \;\; \forall j \in \mathbb{V}, \;\; \forall b \in \mathbb{B}, \;\; \forall \mathcal{D}_{pk}, \mathcal{D}_{ql} \in \mathbb{D} \tag{3.12}$$

Eq. 3.6 minimizes the cost of both the energy consumption and the communication, for time $t$ ($hours$); since we consider *pay-as-you-use* pricing model. The constraint in Eq. 3.7 makes sure that every virtual component is assigned to one and only one DC. Constraints in Eq. 3.8 and Eq. 3.9 describe that the CPU and the memory used by all $VMs$ allocated to a data center $\mathcal{D}_{pk}$ should not exceed its CPU and memory capacities, respectively. In the same way, Eq. 3.10 shows that the storage used by all $DBs$ allocated to a data center $\mathcal{D}_{pk}$ should not exceed its storage capacity. Eq. 3.11 represents the limited bandwidth constraint. It describes that the cumulative traffic requirement between the set of virtual components placed at a data center $\mathcal{D}_{pk}$ and the set of virtual components placed at another data center $\mathcal{D}_{ql}$ cannot exceed the bandwidth of the link between these two data centers. Eq. 3.12 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 [126, 127]. Therefore the problem considered in this chapter is NP-hard [52]. In the next section, we propose a heuristic to find a near-optimal solution.

## 3.3 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 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

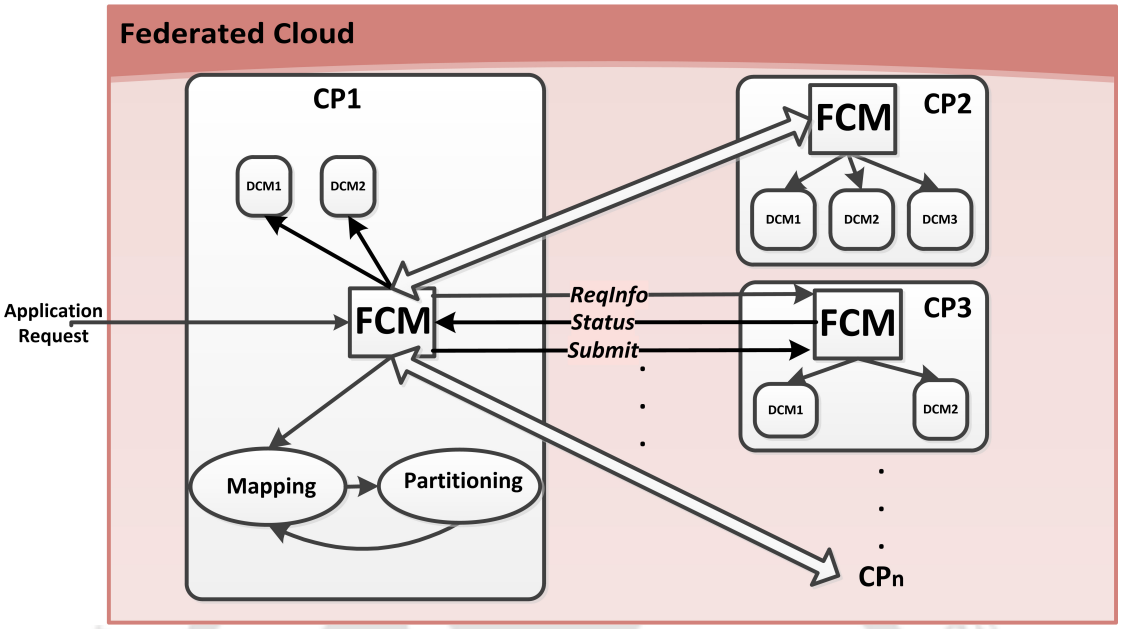



Figure 3.3: Data flow diagram of the proposed algorithm

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 Figure. 3.3. *Partitioning* phase partitions the request graph, shown in Figure. 3.2, 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 **C**ost-**A**ware **V**irtual **C**omponents **A**llocation (**CAVCA**). Without the loss of generality, let us consider that the cloud provider $p \in \mathbb{P}$ is providing the service of data-intensive applications deployment to users, and its FCM $F_p$ runs the algorithm to allocate the required resources for an application request. The algorithm takes as an input $\mathcal{D}_p$; 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, $\mathcal{D}_{ql} \in \mathbb{D}$.

In the initialization phase (Lines 1-6), $DClist$ contains all the DCs in the federation, with only local data centers $\mathcal{D}_p$ to start with. $Best_{DC}$ is the DC with the lowest energy cost,

**ALGORITHM 1: CAVCA** Cost-Aware Virtual Components Allocation

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

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

Initialization:
  $DClist \leftarrow \mathcal{D}_p$
  $Best_{DC} \leftarrow NULL$
  $Clist \leftarrow NULL$
  $Best_{cluster} \leftarrow NULL$
  $AllocationMap \leftarrow NULL$

**for** ***each*** $\mathcal{D}_{pk} \in \mathcal{D}_p$ **do**
  update ***Status*** of $\mathcal{D}_{nk}$
**end**
**for** ***each*** $F_q \in \mathbb{F}$ **do**
  Send ***ReqInfo*** to $F_q$
  Receive ***Status*** from $F_q$
  add received data centers to $DClist$
**end**
Sort $Dlist$ according to $E_{pk}$ in increasing order
$Best_{DC} \leftarrow DClist.head$
$Clist \leftarrow \{G\}$
**while** $(Clist \neq NULL)$ **do**
  Sort $Clist$ according to capacity in decreasing order
  $Best_{cluster} \leftarrow Clist.head$
  **if** $(Best_{DC}.capacity > 0)$ **then**
    **repeat**
      Send ***ReqInfo***$(Best_{DC})$ to $FCM$ of $Best_{DC}$
      receive ***Status*** from $FCM$ of $Best_{DC}$
      **if** $(Best_{DC}.capacity \geqslant Best_{cluster}.res)$ **then**
        **if** $(Comm(Best_{cluster}, \mathcal{D}_{ql}) \leq BW_{Best_{DC}, \mathcal{D}_{ql}})\ \forall \mathcal{D}_{ql} \in \mathbb{D}$ **then**
          ***Submit*** $Best_{cluster}$ to $FCM$ of $Best_{DC}$
          $Clist \leftarrow Clist - \{Best_{cluster}\}$
          **for** ***each*** $VC_i \in Best_{cluster}$ **do**
            $AllocationMap$.add$(VC_i, Best_{DC})$
          **end**
          $Best_{cluster} \leftarrow Best_{cluster}.next$
      **else**
        $last_{cluster} \leftarrow Best_{cluster}$
        $Best_{cluster} \leftarrow Best_{cluster}.next$
    **until** $Best_{cluster} = NULL$;
    $Clist \leftarrow Clist - \{last_{cluster}\}$
    $Clist \leftarrow Clist \bigcup$ **Partitioning**$(last_{cluster})$
  **else**
    $Best_{DC} \leftarrow Best_{DC}.next$
**end**
return $AllocationMap$

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; ($VC_i$, $\mathcal{D}_{ql}$), where a virtual component $VC_i$ is assigned to a data center $\mathcal{D}_{ql}$.

When an FCM, say $F_p$ 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). $F_p$ combines all the received status messages and creates *DClist*. 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, $F_p$ checks the resource availability in a data center $\mathcal{D}_{ql}$ from the corresponding FCM $F_q$, whenever it needs to assign some workload to $\mathcal{D}_{ql}$, 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. $\mathcal{D}_{ql}$), and compares this sum with the available bandwidth between $Best_{DC}$ and $\mathcal{D}_{ql}$ 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,

**ALGORITHM 2: Partitioning**

**Input:** $C$ Cluster of virtual components
$\delta$ range of edge's weight to be removed

**Output:** $UpdatedC$ the cluster after removing edges

```
1 Initialization:
2     UpdatedC ← C
3     n_c ← 1
4     σ ← δ
5 while (n_c == 1 ) do
6 |   UpdatedC= Remove all edges with (weight ⩽ σ)
7 |   σ ← σ + δ
8 |   n_c = Number of clusters in UpdatedC
  end
9 return UpdatedC
```

then $F_p$ 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, $F_p$ 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 $\delta$ 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 $\delta$ is very small, the number of iterations in *Partitioning* becomes large. On the other hand, if $\delta$ is large, then the algorithm would remove huge number of edges and the number of clusters would be large. We chose $\delta$ 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 $\delta$, 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.

**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 $(N >> M)$, 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 $\mathcal{O}(N^2)$.

**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.

*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 $M + L + \log(N)$.

*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

$$(M + L + \log(N))\ Status_{size}\ \overline{DP_{pk,ql}} + \frac{M \log(M) + N^2 \log(N)}{VM_{freq} VM_{cores}}\ VM_{price} \tag{3.13}$$

where $Status_{size}$ is the size of a *Status* message defined in the manuscript which is not more than 20 Bytes. $\overline{DP_{pk,ql}}$ 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 10000 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.

## 3.4 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.

### 3.4.1 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 Figure. 3.4. Each DC is labeled with a tuple indicating the CP, the location, and electricity price (in $/KWh) [49]. The locations were based on those used by Google, Microsoft and Amazon cloud providers [1]. 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 [50]. The bandwidth between data centers is in the range [40-100Gbps] [128]. The data transfer cost is set to $0.01\$/GB$ within a region and $0.08\$/GB$ across regions [51].

[1] https://www.google.com/about/datacenters/inside/locations
https://azure.microsoft.com/en-in/global-infrastructure/
https://aws.amazon.com/about-aws/global-infrastructure

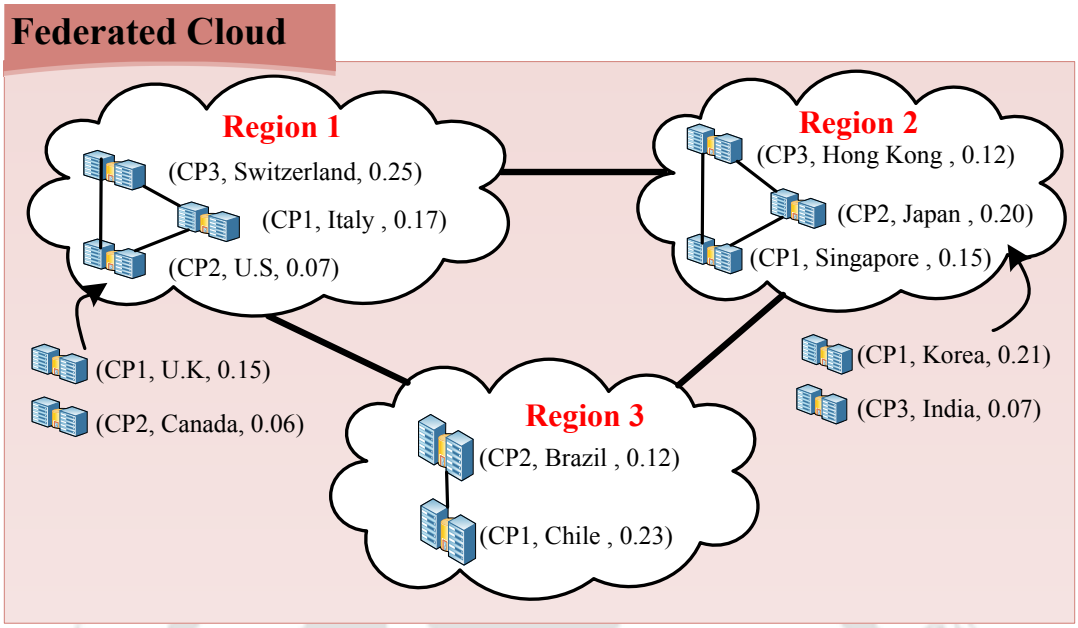



Figure 3.4: Simulation setup of a federated cloud system

**Application Request:** Based on real data-intensive application instances generated by Pegasus [129], 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 $[0-0.5]$ $GB/h$.

### 3.4.2 Results

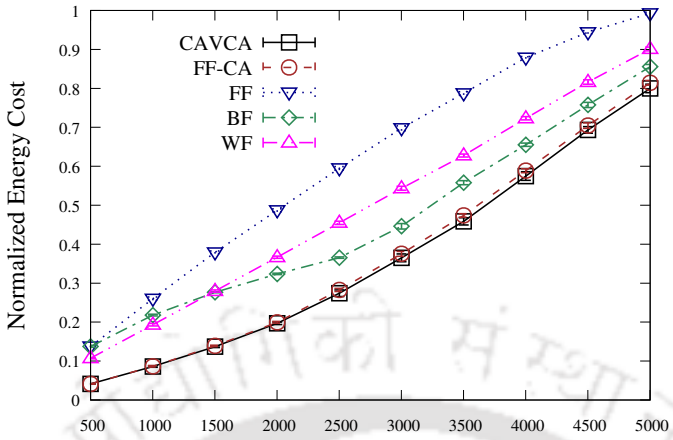
To show the advantages of considering the operating cost in the placement of virtual components, we also implemented three allocation approaches used commonly in cloud management platforms, such as Eucalyptus and OpenStack; first fit (FF), best fit (BF), and worst fit (WF) [130, 131]. FF scans data centers in order of resource capacity and allocates a VC to the first DC that can accommodate it, while BF and WF allocate a VC to the DC with minimum and maximum resources available, respectively. For fairness, we used First Fit Cost-Aware (FF-CA) approach to allocate a VC to DCs according to lowest electricity price [48]. In all the experiments, we take a time slot of 24 hours to calculate the cost. Each experiment was run 1000 times and the 95% confidence intervals are shown in the plots. All the results were normalized to the maximum value.

**Impact of application request size:** We evaluated CAVCA and other baseline approaches using the setup shown in Figure. 3.4 while varying the request size from 500 to 5000 virtual components. We assume a uniform PUE value of 1.2 for all DCs [2], and $Best_{DC}$ in CAVCA is selected based on the electricity price. Figure. 3.5 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 as depicted by Figure. 3.5a. 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 a similar energy cost as FF-CA, which is the optimal one in terms of the energy cost. It can also be seen from Figure. 3.5b 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 Figure. 3.5c. Figure. 3.5a 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.

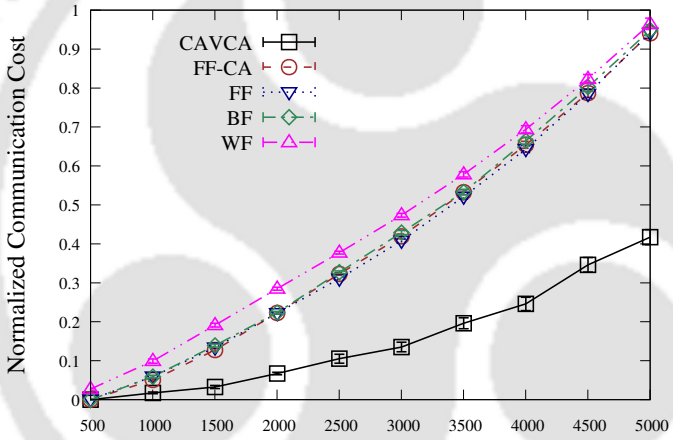
**Impact of PUE variation:** To understand the impact of PUE diversity, we repeat the previous experiment, varying the PUE across the DCs. PUE is randomly set between 1.1 to 2 [3],and we use PUE to select the $Best_{DC}$ for placement along with the electricity price. Similar to the earlier case, the energy cost obtained with CAVCA is 14% lower than that with FF-CA, as seen from Figure. 3.6a. That 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 Figure. 3.6b. Consequently, the total cost is also lower, as evident from Figure. 3.6c.

[2]https://www.vertatique.com/no-one-can-agree-typical-pue

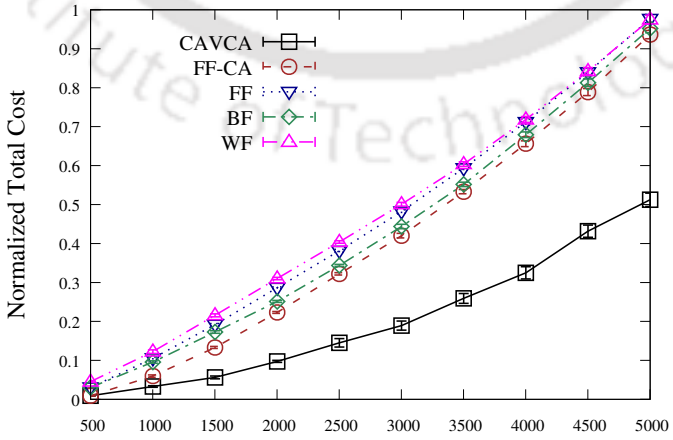
[3] https://www.datacenterdynamics.com/news/apac-data-center-survey-reveals-high-pue-figures-across-the-region/




(a) Energy cost

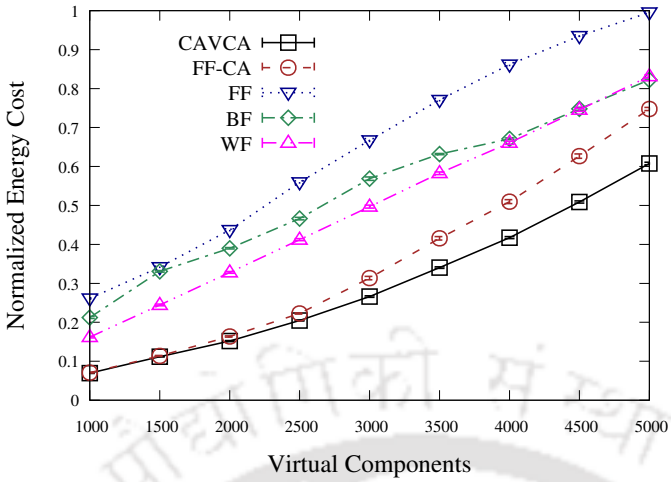



(b) Communication cost

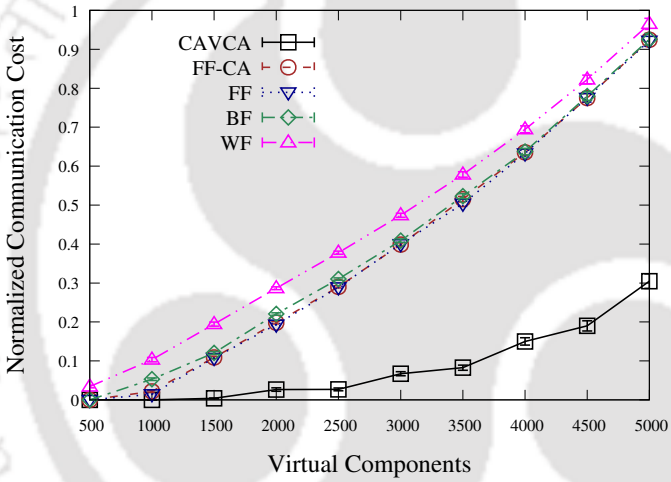



(c) Total cost

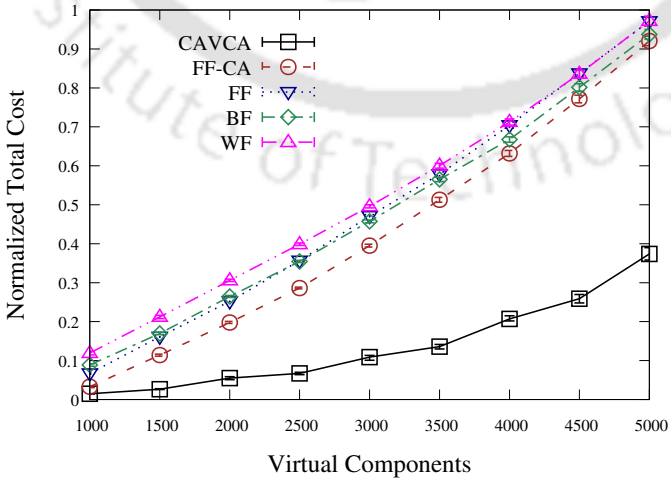
Figure 3.5: Impact of application request size on the cost




(a) Energy cost




(b) Communication cost



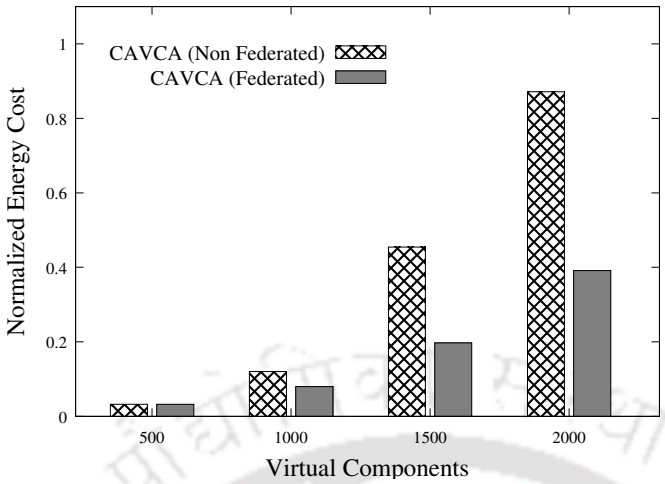

(c) Total cost

Figure 3.6: Impact of application request size on the cost factors for varied PUE

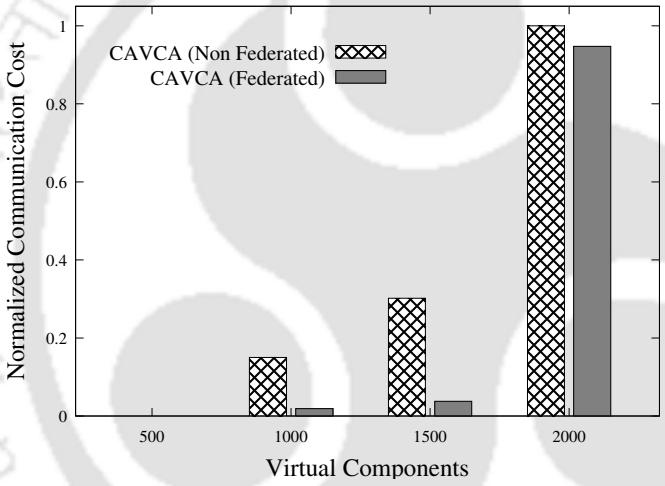
**Impact of the federation:** To understand the advantages of federation, we evaluated CAVCA for two scenarios; federated and non-federated cloud, based on the setup given in Figure. 3.4. We assume that $F_1$ (FCM of $CP_1$) receives application requests with size varying from 500 to 2000 (the maximum 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 Figure. 3.7a. 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 Figure. 3.7b). 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 Figure. 3.7b. Accordingly, the total cost obtained by the algorithm using federation is less than that in non federated system as depicted by Figure. 3.7c. Further, a CP can accept larger requests using federation as already shown in Figure. 3.5.

**Impact of varying data transfer price:** Since many CPs, such as Microsoft [125], 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 Figure. 3.8a 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 Figure. 3.8b. Similarly, Figure. 3.9 demonstrates the performance of CAVCA for federated and non-federated scenarios with varying data transfer price, showing that 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.

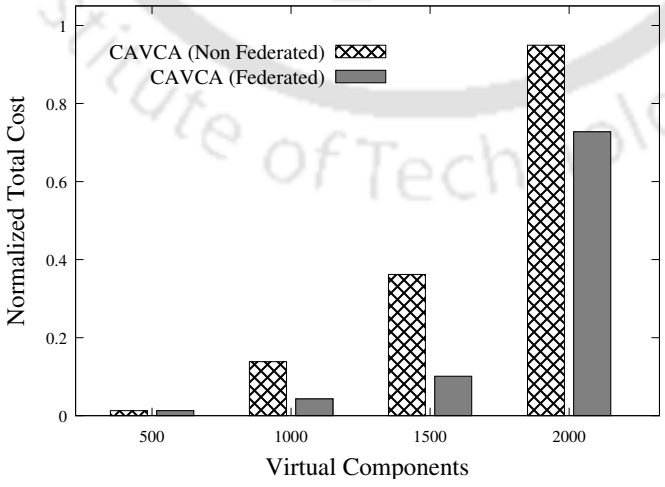
**Impact of adding data centers to the federation:** In this scenario, we assume that four DCs located in Korea, U.K, India and Canada are added in sequence to the federation shown in Figure. 3.4. The size of the application request is fixed at 5000 virtual components. Since the capacity of the first eight DCs can satisfy the resource demands of 5000 virtual




(a) Energy cost

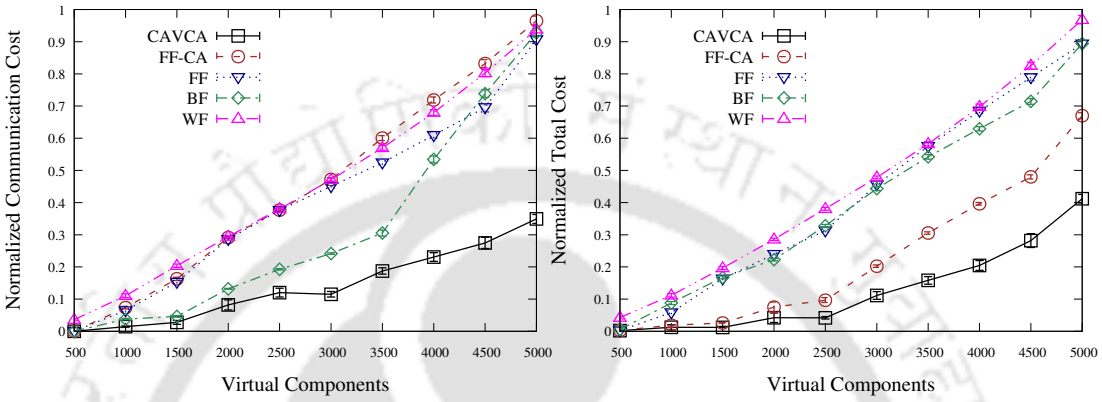



(b) Communication cost



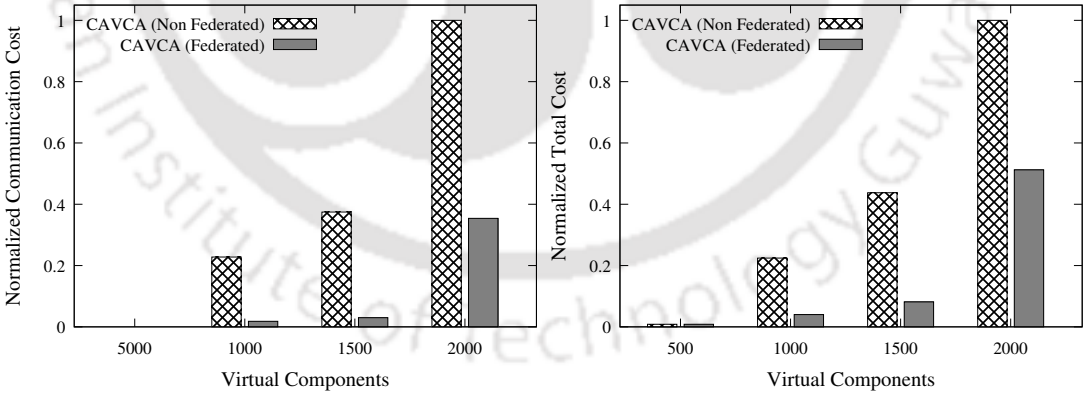

(c) Total cost

Figure 3.7: Comparison of the cost factors in federated and non-federated clouds




(a) Communication cost (b) Total cost

Figure 3.8: Impact of request size on the cost with various data transfer prices




(a) Communication cost (b) Total cost

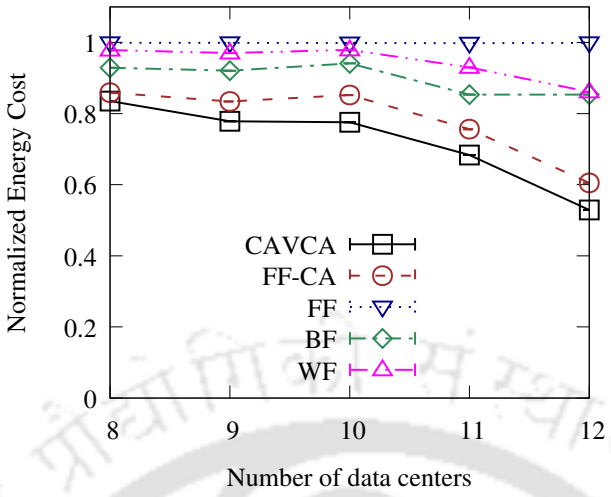
Figure 3.9: Impact of the federation on the cost with various data transfer prices

components, the energy, communication and total costs of FF approach are constant which can be observed from Figure. 3.10. The change in energy cost of BF and WF depends on the capacity of the newly joined DC as depicted by Figure. 3.10a. 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 price and PUE while selecting $Best_{DC}$ for placement. Figure. 3.10b 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 as shown in Figure. 3.10c. CAVCA reduces the total cost when the electricity price varies significantly across DCs, which is true in practice.

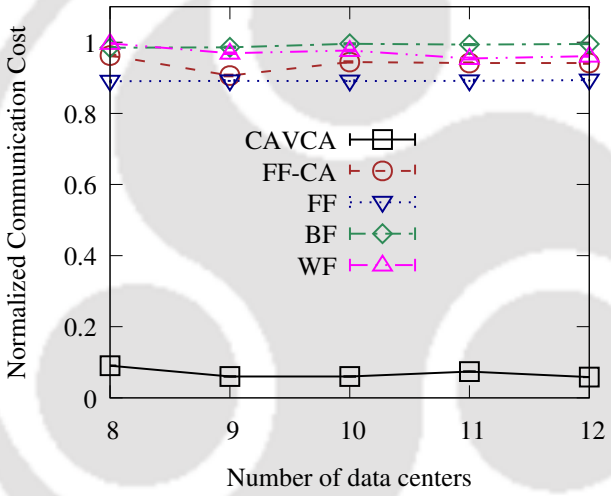
**Impact of data center locations on the total cost:** To study how the locations of DCs affect the cost of CPs in the federation in terms energy consumption cost and communication cost, we evaluated CAVCA with a fixed request size of 3000 virtual components for different sets of locations given below.

- 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}

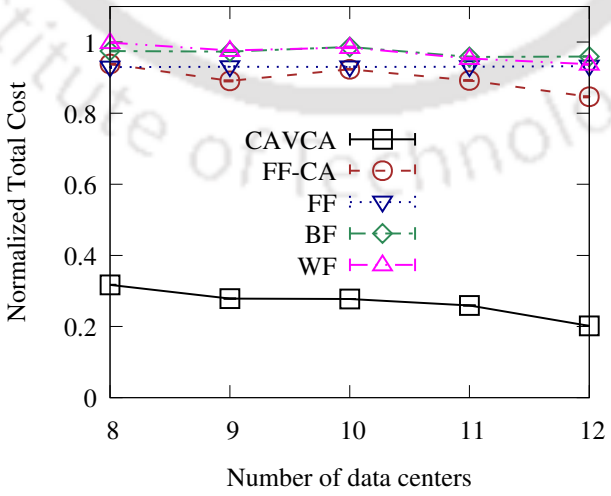
Figure. 3.11 shows the energy cost, communication cost, and the total operating cost obtained with CAVCA executed on different sets of DCs. Set8 gives the lowest energy cost,




(a) Energy cost

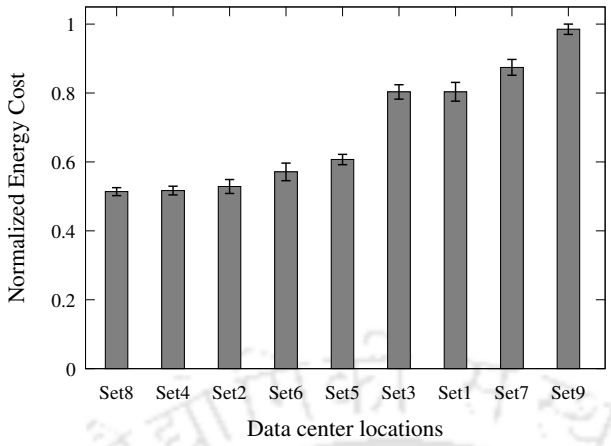



(b) Communication cost

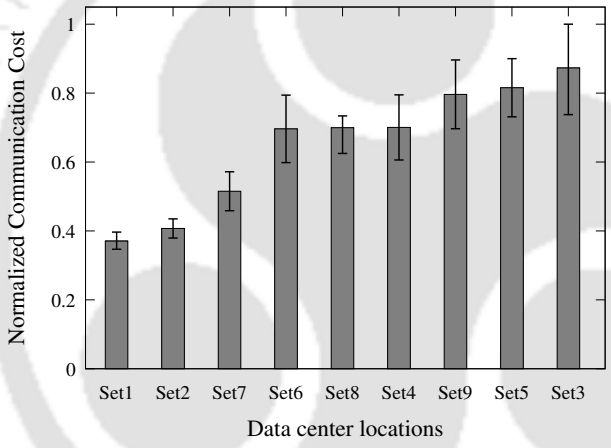



(c) Total cost

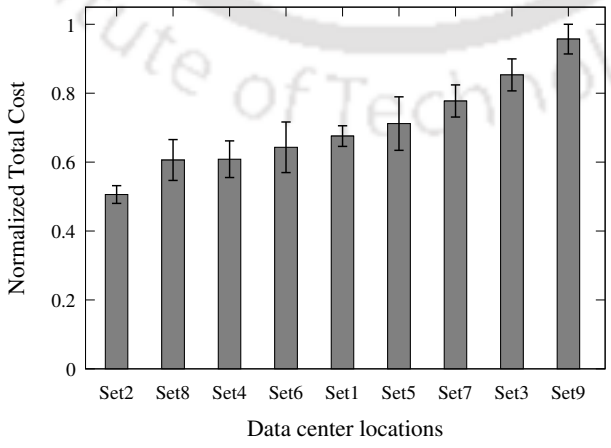
Figure 3.10: Impact of number of data centers in the federation on different cost factors




(a) Energy cost




(b) Communication cost




(c) Total cost

Figure 3.11: Impact of data center locations (grouped into sets) on the different cost factors

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 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.

**Average Running Time:** Average running time is a significant performance metric to evaluate a placement approach and check its feasibility of being used in real-time. Figure. 3.12 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, CAVCA takes a small fraction of time for small to medium applications. Even for a large application with 10000 virtual components, CAVCA does not need more than three minutes.

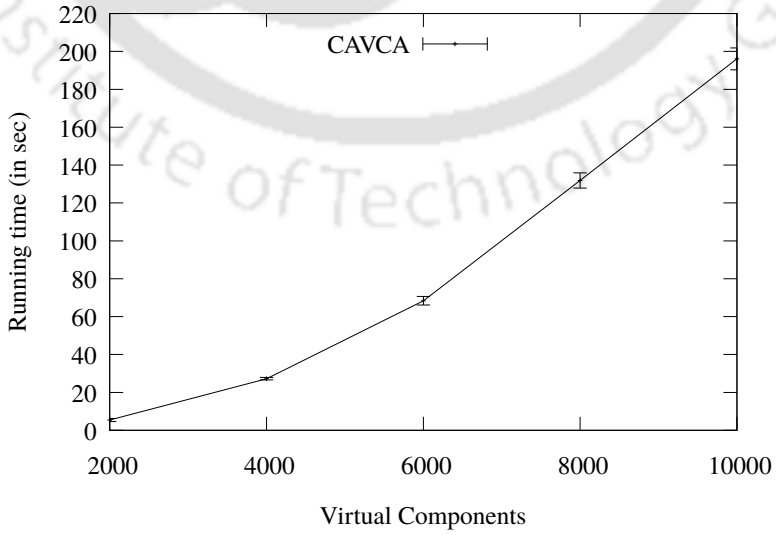



Figure 3.12: Average running time of the CAVCA algorithm for various application sizes

The time taken is negligible compared to the running time of data-intensive application which may take days. From the results we conclude that 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.

## 3.5 Summary

The work in this chapter studied 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 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 considering realistic data. 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 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. In next chapter, we address the problem of resource management for hosting vehicular applications in a federated cloud system where the delay requirements of an application request should be considered to meet the desired QoS of the end-user, besides minimizing the cost of the cloud provider.

CHAPTER

# 4

# Cost-and-Delay Aware Dynamic Resource Allocation in Federated Vehicular Clouds

In this chapter, we address the problem of resource allocation for vehicular applications in cloud data centers where the delay requirements of the application are considered while minimizing the cost. As discussed in Chapter 1, several enterprise vehicular services are being used to enhance the driving experience, like safety messages delivery and infotainment applications. These services require huge resources. To this end, VCC has emerged to host these services in cloud. The main objective of the CP is to maximize its profit by minimizing its operating cost while meeting the QoS requirements of the hosted applications. The user mobility in VCC makes the resource management a challenging problem for the CP. Despite the increasing attention on this topic, most efforts have focused on user-centric solutions, and unfortunately not on the difficulties encountered by CPs in improving their business. Exploiting federation in VCC brings many benefits for the CP and the end users as well. It allows the CP to serve more users within the desired delay and cost requirements, leading to higher revenue [10]. It also helps to deal with vehicle mobility and improve the QoS by monitoring the performance and migrating the VM of the user to a DC that provides lower delay to maintain the SLA.

**Motivation:** Modern vehicular applications are resource-hungry and delay-aware. With increasing user density and due to their high mobility, we argue that Federated Vehicular Cloud Computing (FVCC) is the way forward to handle such applications, while the provider tries to maximize its profit. We observed from the literature that most of the works either

optimize the cost or the delay during resource allocation in VCC architectures (not in a federated scenario). We argue that both should be jointly optimized to meet the objectives of the CP and the users. Additionally, an ideal dynamic allocation strategy should classify the vehicular applications based on their delay requirements and migrate the VMs to handle the mobility in VCC. Note that inter-DC VM migration is a complex operation. Therefore, migration should be performed only when it is required and number of migrations should be minimized. To the best of our knowledge, no existing work in the literature exploited federated cloud for vehicular applications to demonstrate its benefit to both the CP and the users.

In this chapter, the problem at hand is how a CP hosting vehicular services can dynamically allocate resources in a federated cloud environment to minimize its operating cost while assuring the delay requirements of users moving across the network. We introduce FVCC platform which integrates federated cloud concept in VCC. We propose a dynamic pricing model for resource sharing among the CPs in the federation. Towards solving the problem, we propose a cost-and-delay aware algorithm that allocates the resources for vehicular applications in federated cloud dynamically considering the delay requirements of the application and the cost constraints of the CP. This algorithm leverages the diversity in cost and delay across DCs of the federation to meet the SLAs of the users and to minimize the cost of the CP. It migrates VMs only when it is required, based on mobility and delay.
**Organization of the Chapter:** The remainder of this chapter is structured as follows. We discuss the proposed FVCC architecture and the dynamic pricing model in Section 4.1. The proposed algorithm is described in Section 4.2. Simulation results are presented in Section 4.3 followed by a summary of the chapter in Section 4.4.

## 4.1 System Model

In this section we describe the proposed FVCC architecture, the model followed for application requests and the dynamic pricing model.

### 4.1.1 Federated Vehicular Cloud Computing (FVCC)

In this work, we modify the VCC system to leverage the advantages of federated cloud. The proposed FVCC architecture uses the federated cloud to handle the resource-intensive

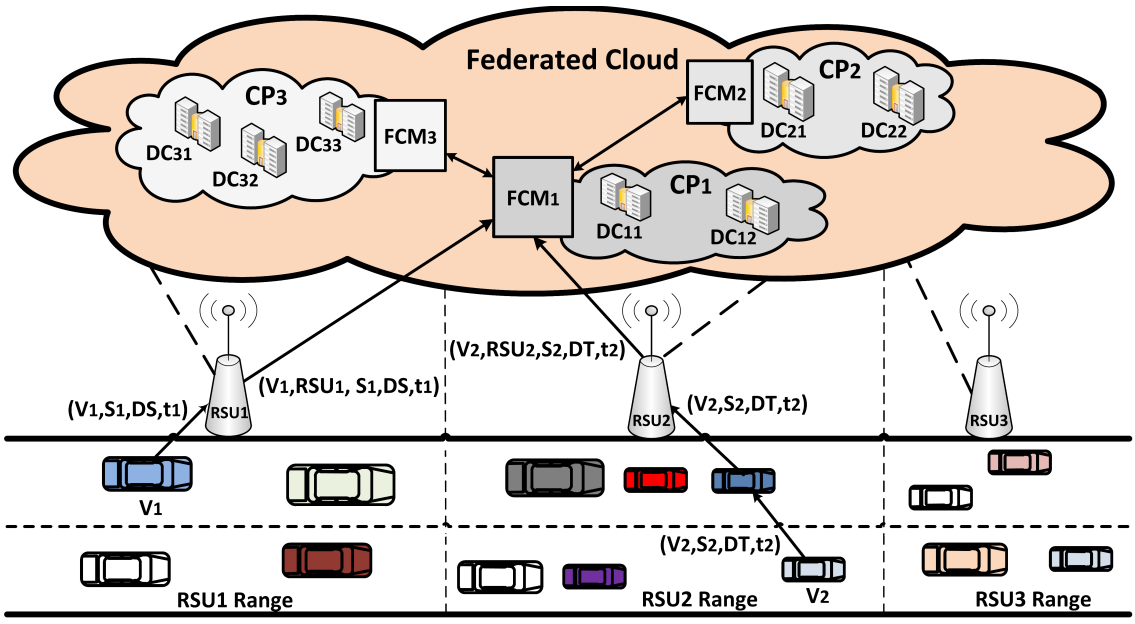



Figure 4.1: FVCC architecture

applications running in the vehicular networks. The proposed architecture has three major modules, that are discussed below and also illustrated in Figure. 4.1.

- *Mobile vehicles*: A set of vehicles that represent the users in the FVCC architecture, move along the road in both directions with arbitrary speeds. The density of vehicles might vary depending on the time of the day and the area (highway or city). The vehicles run different applications that are served from the federated cloud. We assume that the distance between the vehicles as well as the variation in the speed follow an exponential distribution [132].

- *Road Side Units (RSUs)*: The RSU serves as an interface between mobile vehicles and the cloud. The vehicles and RSUs communicate over a wireless link (typically IEEE 802.11p) [133], while the RSUs are connected to the cloud via the backbone Internet. RSUs are spread uniformly in the area with each RSU having a fixed coverage range (about 1 Km) [134]. The distance between the RSUs is decided by the infrastructure planner, with a sparser deployment increasing the response time for the users and a denser deployment increasing the infrastructure cost to the provider [135]. RSUs might be able to host small applications or buffer limited data, but since we assume resource intensive multimedia applications, they are hosted in the cloud [32].

- *Federated cloud*: We consider a federated cloud system where multiple CPs collaborate to share their resources based on a dynamic resource pricing model discussed subsequently [44]. Each CP has one or more DCs, and we assume a CP is also a vehicular service provider apart from serving as an infrastructure provider to the other CPs (e.g. $CP_1$ in Figure. 4.1). The common objective of all CPs in the federation is to maximize their profit [47]. Hence, each CP offers the free resources to other CPs and likewise it may rent resources from other CPs to expand its capacity, meet the SLA, or optimize the cost [136]. We use the hierarchical resource management framework presented in Section 3.1, where each DC has a DCM and every CP has a FCM. The DCM manages the resource allocation and VM scheduling in its DC, while the FCM handles the requests from the users of its CP and brokers with other FCMs regarding resource availability and price. FCM also migrates the VMs across the DCs to meet the QoS requirements and to optimize the cost.

### 4.1.2 Vehicular Application Requests

We classify the vehicular applications into two categories based on the QoS requirements as follows:

- Delay-Sensitive (DS) applications, such as safety applications that have strict delay constraints [114]. For these applications, the CP has to minimize the delay, without it exceeding a threshold, say $dTh_{DS}$. At the same time, the cost incurred in service should also be limited to a threshold, say $cTh_{DS}$.

- Delay-Tolerant (DT) applications, such as mobile social networking and infotainment applications that are less sensitive in terms of delay [54]. Here, we require the delay not to exceed a reasonably larger threshold, say $dTh_{DT}$, while the cost being lower than a threshold of $cTh_{DT}$.

Since DS applications require smaller response times compared to the DT ones (*i.e.*, $dTh_{DS} < dTh_{DT}$), they are served by a VM with better configuration (and costlier) than that used for the DT applications. To choose the optimal service provider for each service, the proposed system considers a trade-off between the delay and the cost of a VM. To maximize the profit, a service provider would strive to reduce the delay of DS applications

while adhering to the cost threshold and to reduce the cost of DT applications while meeting the delay threshold. Thereby, the profit can be maximized, while meeting the delay requirements of various services.

To get a service from the cloud, a vehicle sends a request to the registered service provider through the nearest RSU [137]; requests follow the Poisson distribution [138]. We define a service request using the tuple, $\{V_{id}, Srv_{id}, Srv_{type}, TS\}$, where $V_{id}$ is the vehicle identifier, $Srv_{id}$ is the service identifier, $Srv_{type}$ is the service type (DS or DT), and $TS$ is the time of initiating a request. $Srv_{id}$ is used to identify the service instead of the IP address to make VM migration transparent to the user and achieve service continuity [18]. $Srv_{type}$ is assigned automatically by the device based on the class of the application. On receiving a request, the RSU forwards it to the FCM of the service provider in the cloud after adding $RSU_{id}$, the forwarding RSU's identifier, as indicated in Figure. 4.1. If an RSU is in the transmission range of the vehicle, the vehicle sends the request directly to the RSU (e.g. $V_1$ in Figure. 4.1), else, the vehicle uses multi-hop vehicle-to-vehicle (V2V) communication (e.g. $V_2$ in Figure. 4.1) [139, 140].

Figure. 4.2 shows the sequence of steps in serving a request. First, a vehicle (application) sends the request to the nearest RSU, which is forwarded to the FCM of the service provider in the cloud (say $FCM_1$), where it is added to a scheduler queue. For each request, $FCM_1$ runs the proposed resource allocation algorithm (discussed in Section 4.2) to forward the request to a selected DC. After processing the request, the DC sends the response directly to the sender vehicle through the forwarding RSU. Here, we assume urban mobility scenario (average speeds below 50km/hr), so that the response to a request is routed through the same RSU. However, if a vehicle moves to another RSU before receiving the response, we could use Mobile IP architecture to tackle the handover. Multiple delays incurred in various phases of the request handling [46].

### 4.1.3 Resource Pricing Model

Though many CPs follows fixed pricing model for VMs with fixed configuration, recent studies show that this is not efficient in federated cloud and many CPs nowadays use dynamic pricing based on supply and demand [44]. This allows CPs to rent-out resources based on the price and the demand to maximize their profit and the utilization of their

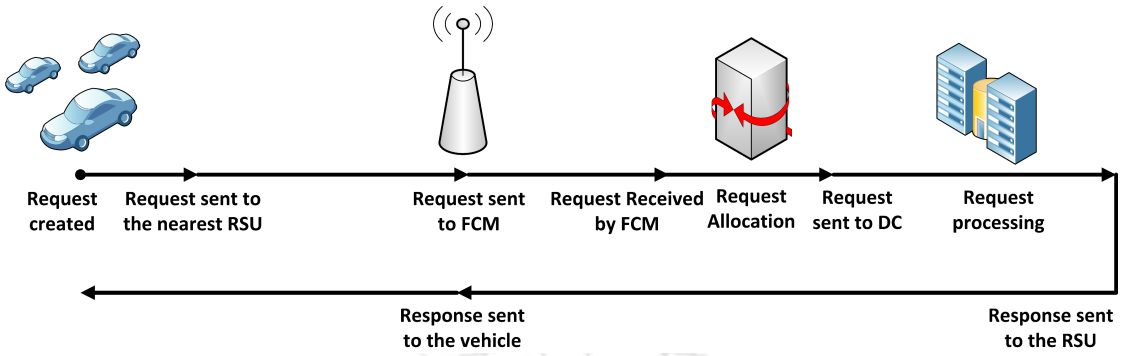



Figure 4.2: Process flow diagram demonstrating the service of requests

resources, and to extend their capacity [16]. We define $fedPrice$ as the outsourcing price offered by a CP $p$ to other CPs in the federation. CPs consider multiple factors, such as the operating cost, price for end-user, the demand, and the resource availability, while setting $fedPrice$. We consider predefined VM instances offered by CPs where $g$ denotes the type of VM instance with specific configurations of CPU and memory (e.g. small, medium, etc). $\mathcal{D}_{pk}$ is defined as the $k^{th}$ DC of the CP $p$. We also define $opCost_g^{pk}$ and $userPrice_g^{pk}$ as the operating cost of running a VM instance of type $g$ at $\mathcal{D}_{pk}$, and the price of this VM instance for the end-user, respectively. Accordingly, the profit for $p$ due to hosting a VM instance of type $g$ at $\mathcal{D}_{pk}$, $profit_g^{pk}$ is defined as the difference between the price paid by the end user and the operating cost of this instance, *i.e.*,

$$profit_g^{pk} = userPrice_g^{pk} - opCost_g^{pk} \tag{4.1}$$

To foster collaboration, a CP $p$ should charge the other CPs less than that charged to the end-user, while considering the increase in demand. Thus, $fedPrice$ offered to other CPs varies between the operating cost and the user price according to resource availability. For example, when the resource availability is high, $fedPrice$ should be low (closer to the operating cost), to encourage other CPs to deploy their instances on $p$'s infrastructure. As the resource availability reduces, $fedPrice$ should increase towards the price offered to the end-user. Let $resAvail_g^{pk}$ and $totalAvail_g^{pk}$ be the available and the total number of shareable VM instances of type $g$ in $\mathcal{D}_{pk}$, respectively. We define the utilization of VM

instances of type $g$ offered for sharing at $\mathcal{D}_{pk}$ as

$$U_g^{pk} = \frac{totalAvail_g^{pk} - resAvail_g^{pk}}{totalAvail_g^{pk}} \tag{4.2}$$

The margin between the operating cost and $fedPrice$ depends on the resource utilization and it is defined as a percentage of the profit returned from the end user. Therefore, $fedPrice_g^{pk}$ – the price of a VM instance of type $g$ at $\mathcal{D}_{pk}$ offered by CP $p$ to other CPs, is given by

$$fedPrice_g^{pk} = opCost_g^{pk} + U_g^{pk}\, profit_g^{pk}, \tag{4.3}$$

which is essentially the basic operating cost plus a dynamic profit based on the resource utilization at the destination DC. Table 4.1 summarizes the notations used in the system model.

Table 4.1: Notation used in the system model

| *Symbol* | *Description* |
|---|---|
| $D_p$ | Set of the data centers of the CP $p$ |
| $\mathcal{D}_{pk}$ | $k^{th}$ data center of CP $p$ |
| $opCost_g^{pk}$ | Operating cost of a VM of type $g$ at $\mathcal{D}_{pk}$ (\$/h) |
| $userPrice_g^{pk}$ | End-user price of a VM of type $g$ at $\mathcal{D}_{pk}$ (\$/h) |
| $profit_g^{pk}$ | Revenue of end-user VM of type $g$ at $\mathcal{D}_{pk}$ (\$/h) |
| $fedPrice_g^{pk}$ | federated price of a VM of type $g$ at $\mathcal{D}_{pk}$ (\$/h) |
| $U_g^{pk}$ | Utilization of VM instances of type $g$ at $\mathcal{D}_{pk}$ |
| $totalAvail_g^{pk}$ | number of type $g$ VMs offered for sharing at $\mathcal{D}_{pk}$ |
| $resAvail_g^{pk}$ | number of type $g$ VMs available for sharing at $\mathcal{D}_{pk}$ |
| $dTh_g$ | Delay threshold for VMs of type $g$ (ms) |
| $cTh_g$ | Cost threshold for VMs of type $g$ (\$/h) |

## 4.2 Proposed Algorithm

In this section, we propose an algorithm to solve the problem and also discuss its complexity. It is an online algorithm for initial resource allocation and also VM migration, that meets the delay requirements and reduces the operating cost. The algorithm is designed to maintain the autonomy and privacy requirements of the CPs.

As shown by Figure. 4.3, the proposed algorithm consists of four different algorithms that are discussed in detail. First, **CDRAM** (Cost and Delay-aware Resource Allocation

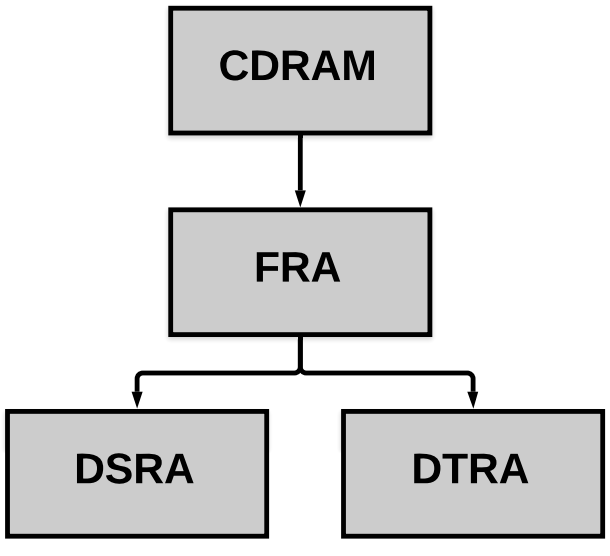



Figure 4.3: Components of the proposed approach

Method) is the main algorithm that creates a VM for each user requesting a service and also checks if migration is required for already placed VMs. It also invokes **FRA** (Federated Resource Allocation) algorithm that collects resource information from DCs of the federation and invokes one among the algorithms; **DSRA** (Delay Sensitive Resource Allocation) or **DTRA** (Delay Tolerant Resource Allocation) to dynamically allocate resources either for the DS applications or DT applications, respectively. These algorithms optimizes the cost and delay constraints jointly.

Algorithm 3 gives **CDRAM** that takes as input $\mathcal{D}_p$, the set of $p$'s DCs ($p$ is the service provider), and $ReqQ$, the queue where $F_p$ (FCM of $p$) stores the service requests. It returns $RsMap$ that maps VMs assigned to the users to DCs of the federation. $RsMap$ is a hash table with a record for each user and an associated service defined as the tuple $(V_{id}, Srv_{id}, RSU_{id}, VM_i, DC_i)$, where $V_{id}$ is the identifier that defines the user, $Srv_{id}$ is the service identifier, $RSU_{id}$ is the identifier of the RSU that forwards requests of $V_{id}$ (it is updated when the user moves to another RSU), $VM_i$ is the VM assigned to $V_{id}$ for the service $Srv_{id}$, and $DC_i$ is the identifier of the DC where $VM_i$ is currently placed. Since a vehicle might be associated with multiple VMs (for various applications), $RsMap$ is indexed by the key $(V_{id}, Srv_{id})$. Initially, $DC_{list}$, the set of DCs in the federation contains only DCs of $p$ and the table $RsMap$ is empty (lines 1-3). $F_p$ serves the requests from $ReqQ$ on FCFS order (line 4). $F_p$ searches through $RsMap$ to know if there is a VM already assigned to

**ALGORITHM 3: CDRAM** Cost and **D**elay Aware **R**esource **A**llocation **M**ethod

**Input:** $D_p$: List of local DCs of CP $p$
$ReqQ$: Requests received from vehicles

**Output:** $RsMap(V_{id}, Srv_{id}, RSU_{id}, VM_i, DC_i)$

```
1  Initialization:
2      DC_list ← D_p;
3      RsMap ← NULL;
4  req = ReqQ.head;
5  while (req ≠ NULL) do
6      if ( !(RsMap.contains(req.V_id, req.Srv_id))) then
7          Create(VM_i, req.Srv_type);
8          VM_i.currentDC ← NULL;
9          VM_i.type ← req.Srv_type;
10         VM_i.currentDelay ← ∞;
11         VM_i.currentPrice ← ∞;
12         VM_i.cost ← 0;
13         newDC= FRA(VM_i, DC_list);
14         Forward (req, (VM_i, newDC));
15         RsMap.insert(req.V_id, req.Srv_id,
                   req.RSU_id, VM_i, newDC);
16     else
17         res ← RsMap.get(req.V_id, req.Srv_id);
18         if (req.RSU_id == res.RSU_id) then
19             Forward (req, (res.VM_i, res.DC_i));
20         else
21             Forward (req, (res.VM_i, res.DC_i));
22             newDC= FRA(res.VM_i, DC_list);
               // Check migration
23             RsMap.update(req.V_id, req.Srv_id,
                       req.RSU_id, VM_i, newDC);
24     update VM_i.cost;
25     req ← req.next;
26 return RsMap;
```

this user for the requested service, otherwise it means $V_{id}$ has requested to join the service (line 6). $F_p$ creates and configures a new VM ($VM_i$) based on the application requirement (lines 7-12). The parameters $currentDelay$ and $currentPrice$ of $VM_i$ are initially set to $\infty$ to be compared later with delay and $fedPrice$ offered by various DCs, respectively (lines 10-11). Then the algorithm calls **FRA** to allocate $VM_i$ in the federated cloud (line 13) after which, $F_p$ forwards the request to $VM_i$ and a record is added to $RsMap$ (line 14-15). Otherwise, if $RsMap$ has a record associated with $V_{id}$ and $Srv_{id}$ (line 16), the algorithm extracts it (line 17) and compares with the received request (line 18). If this request is forwarded through the same RSU as earlier requests from the user (the user is still with the

**ALGORITHM 4: FRA Federated Resource Allocation**

**Input:** $VM_i$: The VM to be allocated or migrated
$\mathbb{F}$: List of federated cloud managers
$DC_{list}$: List of DCs in the federation
$dTh_{DS}$: Delay threshold of DS services
$cTh_{DS}$: Cost threshold of DS services
$dTh_{DT}$: Delay threshold of DT services
$cTh_{DT}$: Cost threshold of DT services

**Output:** $newDC$ : The newly assigned DC to $VM_i$

1 **for** ***each*** $\mathcal{D}_{pk} \in D_p$ **do**
2 Get($resAvail_g^{pk}, fedPrice_g^{pk}$);
3 **for** ***each*** $F_q \in \mathbb{F}$ **do**
4 Request($resAvail_g^{ql}, fedPrice_g^{ql}$) of all $\mathcal{D}_{ql} \in D_q$;
5 Receive($\mathcal{D}_{ql}, resAvail_g^{ql}, fedPrice_g^{ql}$) from $F_q$;
6 Update $DC_{list}$;
7 **if** ($VM_i.type == DS$) **then**
8 $newDC$= **DSRA**($VM_i, DC_{list}, dTh_{DS}, cTh_{DS}$);
9 **if** ($VM_i.type == DT$) **then**
10 $newDC$= **DTRA**($VM_i, DC_{list}, dTh_{DT}, cTh_{DT}$);
11 **return** $newDC$;

same RSU), $F_p$ simply forwards the request to the associated VM using the information in $RsMap$ (lines 18-19).If the request comes from a new RSU, $F_p$ forwards the request to $VM_i$ and checks if migration is required to maintain delay and cost requirements by invoking **FRA** (lines 20-22). The information of $VM_i$ is updated in $RsMap$ (the forwarding RSU and the DC after migration). Typically, in vehicular cloud computing, a VM is composed of a base and a layout. We only migrate the VM layout during VM migration to reduce the size of migrated data. Furthermore, we could use live migration to minimize the service interruption time while migrating the VM. After serving the request the cost of $VM_i$ is updated before the next request (lines 24-25). The cost is calculated using the *pay-as-you-use* policy.

We present **FRA** in Algorithm 4. The input is $VM_i$, the VM to be placed or migrated, $\mathbb{F}$, the set of FCMs of the CPs in the federation, $DC_{list}$, the set of DCs in the federation, and $dTh_{DS}$, $cTh_{DS}$, $dTh_{DS}$, $dTh_{DT}$; the thresholds of delay and cost for DS and DT services. The delay thresholds are configured based on Service Level Agreement (SLA), while the cost thresholds are set by the service provider. It returns $newDC$, the identifier of the DC where $VM_i$ is allocated. **FRA** collects information about the number of available instances

and $fedPrice$ of various VM types across all DCs. $F_p$ gets this for own DCs directly from DCMs or from the FCMs for other CPs (lines 1-5). Then, $F_p$ updates $DC_{list}$ (line 6). The algorithm classifies the type of $VM_i$ (DS or DT) to call the corresponding function for allocation, where **DSRA** is for DS type VMs, and **DTRA** is for DT type VMs (line 7-10). **DSRA** and **DTRA** perform initial VM placement and VM migration as well.

We consider the delay to be a hard constraint, wherein a request is dropped if its delay exceeds the required threshold, while the cost is a soft constraint so that the provider can accept $fedPrice$ more than the defined cost threshold to serve more requests within the deadline [132]. However, $fedPrice$ can not be more than the user price, according to Eq. 4.3.

Algorithm 5 presents **DSRA** that allocates VMs for DS applications. Here, the service provider minimizes the delay also considering the cost threshold $cTh_{DS}$. **DSRA** takes as an input $VM_i$, the VM to be placed, $DC_{list}$ the list of DCs in the federation, and $dTh_{DS}$ and $cTh_{DS}$ the thresholds. It returns identifier of the DC where $VM_i$ has been allocated. After initializing $SelectDC$, a local variable to scan $DC_{list}$, $BestDC$ the best DC for placement and $Candidates_{list}$ for DCs that satisfy $dTh_{DS}$ (but $fedPrice$ is above $cTh_{DS}$) (lines 1-4), the algorithm sorts $DC_{list}$ according to delay and selects the DC that gives minimum delay (lines 5-6). If the delay of the selected DC is greater than $dTh_{DS}$, then the request is dropped (lines 7-8). If the selected DC is the one where $VM_i$ is currently placed and $fedPrice$ in this DC is less than $cTh_{DS}$, then $VM_i$ is already placed in the best DC and there is no need of migration. Hence, the algorithm returns the current allocation (lines 9-10). Otherwise, either $VM_i$ is a new VM to be placed or there may be a better allocation for it. To minimize the delay while considering the cost threshold, the algorithm selects $BestDC$, the first DC with resources to satisfy both $dTh_{DS}$ and $cTh_{DS}$ (lines 12-19). When $dTh_{DS}$ is satisfied but the $fedPrice > cTh_{DS}$, the DC is added to $Candidates_{list}$ (line 18). If no DC satisfies $dTh_{DS}$ within the fixed cost threshold, the algorithm sorts $Candidates_{list}$ based on $fedPrice$ and takes first DC as $BestDC$ (lines 20-25). This is to serve the request within the deadline, but with minimum extra cost above $cTh_{DS}$.

After deciding the best DC for $VM_i$, the algorithm checks if the delay of $BestDC$ is less than the current delay for $VM_i$ (for a new VM, $currentDelay = \infty$) and then it allocates (or migrates) $VM_i$ to $BestDC$ (lines 26-30). If the delay of $BestDC$ is same as the current

**ALGORITHM 5: DSRA** **D**elay-**S**ensitive **R**esource **A**llocation

**Input:** $VM_i$: The VM to be allocated or migrated
$DC_{list}$: List of DCs in the federation
$dTh_{DS}$: Delay threshold of DS applications
$cTh_{DS}$: Cost threshold of DS applications

**Output:** $BestDC$: The DC where $VM_i$ is allocated

Initialization:
    $SelectDC \leftarrow NULL$;
    $BestDC \leftarrow NULL$;
    $Candidates_{list} \leftarrow NULL$;
Sort $DC_{list}$ in increasing order based on Delay;
$SelectDC = DC_{list}.head$;
**if** $(Delay_{i,SelectDC} > dTh_{DS})$ **then**
    **return** $NULL$;
    `// Drop the request`
**if** ( $(SelectDC == VM_i.currentDC)$ & $(fedPrice_{DS}^{SelectDC} \leqslant cTh_{DS})$ ) **then**
    **return** $VM_i.currentDC$;
**else**
    **while** $(resAvail_{DS}^{SelectDC} == 0)$ **do**
        $SelectDC \leftarrow SelectDC.next$;
    **while** $(Delay_{i,SelectDC} \leqslant dTh_{DS})$ **do**
        **if** $(fedPrice_{DS}^{SelectDC} \leqslant cTh_{DS})$ **then**
            $BestDC \leftarrow SelectDC$;
        **else**
            $Candidates_{list}.insert(SelectDC)$;
            $SelectDC \leftarrow SelectDC.next$;
    **if** $(BestDC == NULL)$ **then**
        **if** $(Candidates_{list} \neq NULL)$ **then**
            Sort $Candidates_{list}$ based on $fedPrice_{DS}^{lk}$;
            $BestDC \leftarrow Candidates_{list}.head$;
        **else**
            **return** $NULL$;
**if** $(Delay_{i,BestDC} < VM_i.currentDelay)$ **then**
    **if** $(VM_i.currentDC == NULL)$ **then**
        $Allocate(VM_i, BestDC)$;
    **else**
        $Migrate(VM_i, VM_i.currentDC, BestDC)$;
**else if** ( $(Delay_{i,BestDC} == VM_i.currentDelay)$ &
$(fedPrice_{DS}^{BestDC} < fedPrice_{DS}^{VM_i.currentDC})$ ) **then**
    $Migrate(VM_i, VM_i.currentDC, BestDC)$;
$VM_i.currentDC = BestDC$;
$VM_i.currentDelay \leftarrow Delay_{i,BestDC}$;
$VM_i.currentPrice \leftarrow fedPrice_{DS}^{BestDC}$;
Update$(resAvail_{DS}^{pk}, fedPrice_{DS}^{pk})$;
**return** $BestDC$;

**ALGORITHM 6: DTRA** **D**elay-**T**olerant **R**esource **A**llocation

**Input:** $VM_i$: The VM to be allocated or migrated
$DC_{list}$: List of DCs in the federation
$dTh_{DT}$: Delay threshold of DS applications
$cTh_{DT}$: Cost threshold of DS applications

**Output:** $BestDC$: The DC where $VM_i$ is allocated

```
1  Initialization:
2      SelectDC ← NULL;
3      BestDC ← NULL;
4  Sort DC_list in increasing order based on fedPrice_DT^lk;
5  SelectDC = DC_list.head;
6  if ( (SelectDC == VM_i.currentDC) & (Delay_{i,SelectDC} ⩽ dTh_DT) ) then
7  |   return VM_i.currentDC;
8  while (resAvail_DT^SelectDC == 0) do
9  |   SelectDC ← SelectDC.next;
10 while ( (SelectDC ≠ NULL) & (BestDC == NULL) ) do
11 |   if (Delay_{i,SelectDC} ⩽ dTh_DT) then
12 |   |   BestDC = SelectDC;
   |   else
13 |   |   SelectDC ← SelectDC.next;
14 if (BestDC == NULL) then
15 |   return NULL;
16 else if (fedPrice_DT^BestDC < VM_i.currentPrice) then
17 |   if (VM_i.currentDC == NULL) then
18 |   |   Allocate(VM_i, BestDC);
19 |   else
20 |   |   Migrate(VM_i, VM_i.currentDC, BestDC);
21 |   VM_i.currentDC = BestDC;
22 |   VM_i.currentPrice ← fedPrice_DT^BestDC;
23 |   VM_i.currentDelay ← Delay_{i,BestDC};
24 |   Update(resAvail_DT^pk, fedPrice_DT^pk);
25 |   return BestDC;
```

delay of $VM_i$, but $BestDC$ offers lower $fedPrice$, then $VM_i$ is migrated to $BestDC$ to reduce the cost (lines 31-32). Finally the information of $VM_i$ is updated, $F_p$ is updated with the resource availability and the $fedPrice$ (based on Eq. 4.3) (lines 33-36).

Similar to **DSRA**, **DTRA** presented in Algorithm 6, allocates VMs to DT application requests. It sorts $DC_{list}$ based on $fedPrice$ to minimize the cost while meeting the delay requirements (line 4). In case $VM_i$ is already placed in the DC with minimum $fedprice$ and meets $dTh_{DT}$, then the algorithm simply returns the current location of $VM_i$ (lines 5-7). When $VM_i$ is a new VM or it needs to be migrated, the algorithm selects $BestDC$ to be the DC with shareable VMs to satisfy $dTh_{DT}$ with lowest possible $fedPrice$ (lines 8-13).

If no DC satisfies the delay threshold, the request is dropped (lines 14-15). Otherwise, the algorithm checks whether the $fedPrice$ of the selected $BestDC$ is less than the current price of $VM_i$. If yes, it allocates (or migrates) $VM_i$ to $BestDC$ and updates $VM_i$'s information, along with the resource availability and $fedPrice$ (lines 16-24).

**Time and Space Complexity:** Let $M$ be the number of DCs in the federation and $N$ be the total number of VMs, bounded by the number of users (vehicles) multiplied by the number of applications ($N >> M$). In the proposed algorithm, all operations on $RsMap$ including search, get, insert, and update can be done in $\mathcal{O}(1)$. **DSRA** and **DTRA** use sorting of $DC_{list}$ in $\mathcal{O}(M \log(M))$, and scanning of $DC_{list}$ in $\mathcal{O}(M)$. Therefore, the overall time complexity of the algorithm is in $\mathcal{O}(M \log(M))$. The algorithm needs to store the list of data centers $DC_{list}$ and the table $RsMap$ with an entry for each VM. So the space complexity is in $\mathcal{O}(M + N)$.

## 4.3 Simulation Results

This section discusses the results from implementing the proposed algorithm (termed CDRAM) for the FVCC architecture using a Java-based discrete event simulator running on a server with an Intel Xeon processor and 128GB RAM. We consider a road of length 200 km with multiple lanes. Vehicles travel in both directions with various speeds, and RSUs are deployed uniformly to cover the entire area [134]. We consider two types of VM instances; {2 vCPUs, 8 GB RAM} and {4 vCPUs, 16 GB RAM} to serve DT and DS services, respectively. We assume that all requests of an application have the same QoS and CPU requirements which are defined by the desired delay threshold and the task size in terms of millions of instructions (MIs), respectively [115, 114]. We considered a federated cloud with 12 DCs belonging to three CPs, as shown in Figure. 4.4 where each DC is represented by a tuple that includes the owner CP, the DC location, and the user price in $/h [60, 61, 71]. Each DC has 15000-20000 vCPUs and 60000-80000 GB memory. All vCPUs have the same processing speed [114]. We assume on-demand price based on dynamic pricing model defined in Eq. (4.3). The mean Round Trip Time (RTT) in the cloud is estimated to vary between 10-433 ms [1]. Simulation parameters are listed in Table 4.2.

[1] https://apimetrics.io/apimetrics-free-api-tools-resources/cross-cloud-api-latency-analysis/

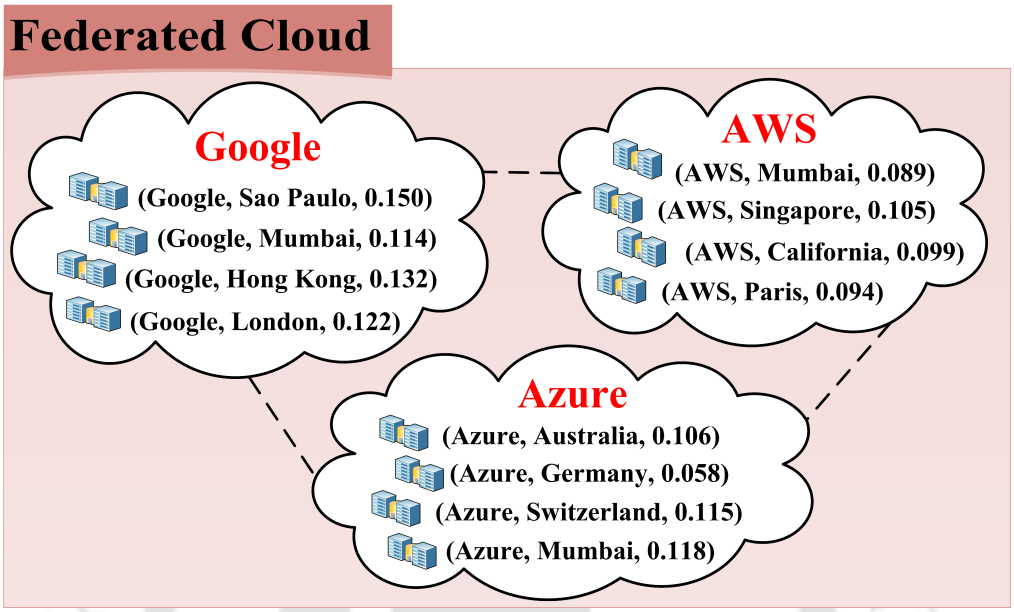



Figure 4.4: Federated cloud scenario for simulation

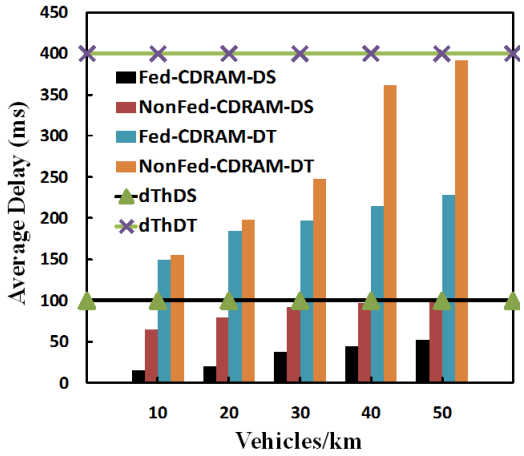
Table 4.2: Simulation parameters

| Parameter | Value |
|---|---|
| Vehicles speed | 30-100 $km/h$ |
| Transmission range | 300 $m$ |
| RSU coverage range | 1 $km$ |
| Packet length | 100 $Byte$ |
| Data rate | 6 $Mbps$ |
| Delay threshold of DS services $dTh_{DS}$ | 100 $ms$ |
| Cost threshold of DS services $cTh_{DS}$ | 0.180 \$/$h$ |
| Average DS task size | 1500 $MIs$ |
| Delay threshold of DT services $dTh_{DT}$ | 400 $ms$ |
| Cost threshold of DT services $cTh_{DT}$ | 0.90 \$/$h$ |
| Average DT task size | 750 $MIs$ |
| Packet generation rate | 10 $Packets/s$ |
| Processing speed of vCPU | 20000 $MIPS$ |
| Mean RTT in cloud | 10-433 $ms$ |

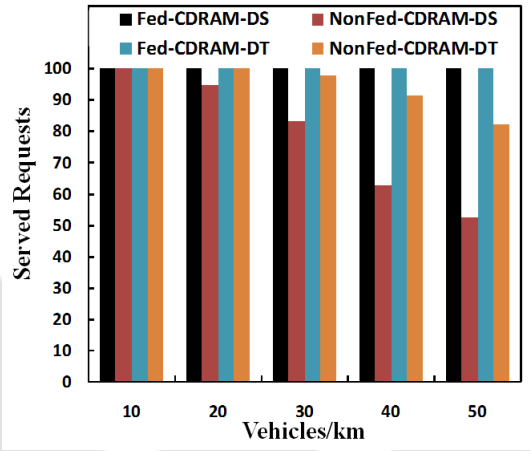
**Impact of federation:** To analyse the benefits of federation in VCC, we evaluated CDRAM for DS and DT applications in federated and non-federated scenarios. We assume $CP_1$ (Google) to be the vehicular services provider in Figure. 4.4 and vary the number of vehicles per km (vehicles/km) from 10 to 50. As shown in Figure. 4.5a, in non-federated scenario the delay increases rapidly as number of vehicles increases. In federated environment, CDRAM gives lower delay and it satisfies QoS requirements of both DS and DT services.

From Figure. 4.5b we can observe that using federation CDRAM is able to serve requests in all the cases, while about 50% of DS requests and 20% of DT requests are dropped when vehicles/km reaches 50 in the non-federated scenario. Furthermore, the average cost per VM exceeds the corresponding cost threshold for DS and DT services without federation (see Figure. 4.5c.) In federated cloud, the cost for both DS and DT services is reduced significantly and all the requests are served within the threshold, because the CP has more choice in finding a DC that meets the delay requirements with lower cost.

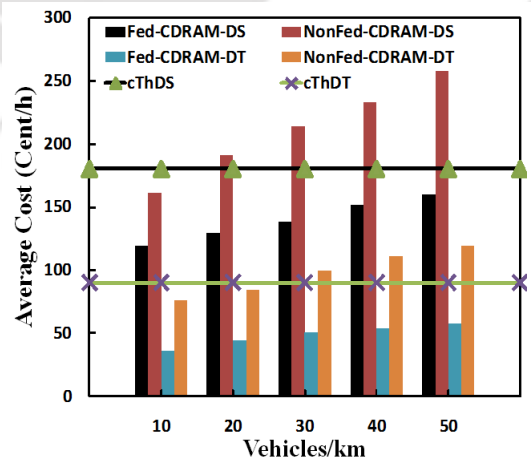
**Impact of vehicle density:** Next, we varied vehicles/km to reflect different traffic density scenarios and compare CDRAM with two baseline approaches **D**elay-**A**ware (**DA**), an algorithm that assigns the requests to a DC serving with least delay and **C**ost-**A**ware (**CA**), an algorithm that assigns the request to a DC operating with least cost [115]. We show the results for DS and DT applications separately to show the impact of application awareness. Figure. 4.6a shows that CDRAM-DS achieves the best delay for DS applications as the algorithm always allocates and migrates VMs to the DC with minimum delay. Though DA allocates VMs to DC with minimum delay, it shows a higher average delay because DA allocates all VMs in the same way. It is also observed that CDRAM-DT gives higher delay than CDRAM-DS and DA, because it tries to reduce the cost of the service while meeting $dTh_{DT}$. Nevertheless, CDRAM-DT satisfies $dTh_{DT}$ for all DT requests even with large number of requests. CA gives the highest delay as it cares only about minimizing the cost rather than the delay. Figure. 4.6b shows that CDRAM can serve all the DT requests and drops less than 10% of DS requests in case of high traffic (vehicles/km more than 80). This could be avoided by placing more RSUs to reduce the channel access contention [132]. CA drops about 40% of total requests even with low traffic (vehicles/km is 10) and it accepts only 40% of the requests when vehicles/km is 100, because CA tries to minimize only the cost. DA accepts all the requests for low traffic but fails with medium and high traffic. Figure. 4.6c demonstrates that the average cost per VM increases with the number of vehicles, due to the increased resource utilization leading to a higher $fedprice$. We can see that the cost with CDRAM-DT is the minimum since it allocates and migrates VMs to the cheaper DC that satisfies $dTh_{DT}$. Since $dTh_{DT}$ is relatively high, CDRAM-DT can serve all requests within the threshold $cTh_{DT}$. CDRAM-DS gives higher cost than CA and CDRAM-DT, since it trades off QoS with cost. With high density, CDRAM-DS may accept




(a) Average delay

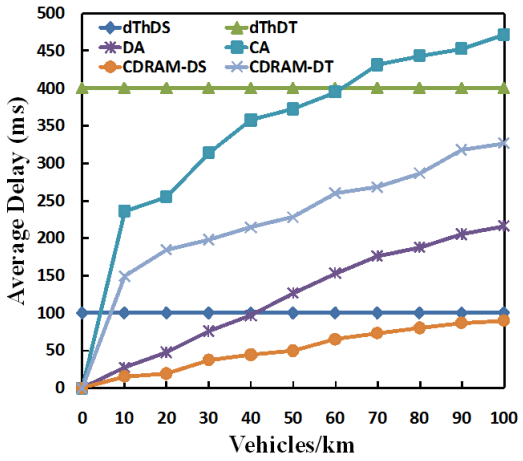



(b) Served requests

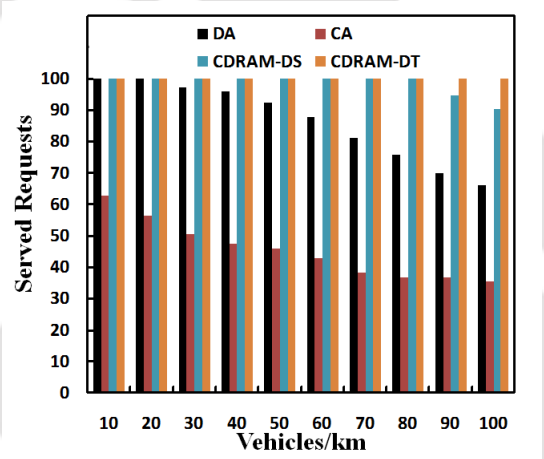



(c) Average cost

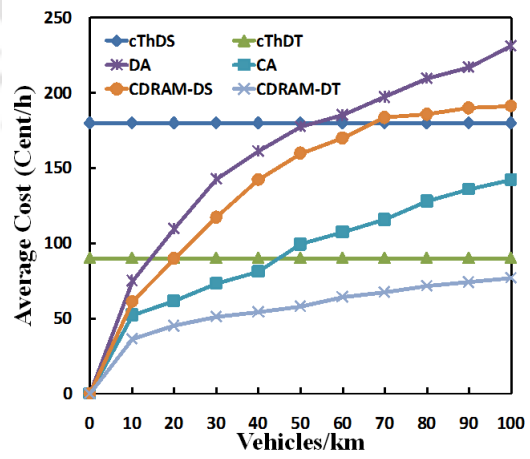
Figure 4.5: Impact of federation on the delay, the served requests, and the cost




(a) Average delay




(b) Served requests




(c) Average cost

Figure 4.6: Impact of vehicle density on the delay, the served requests, and the cost

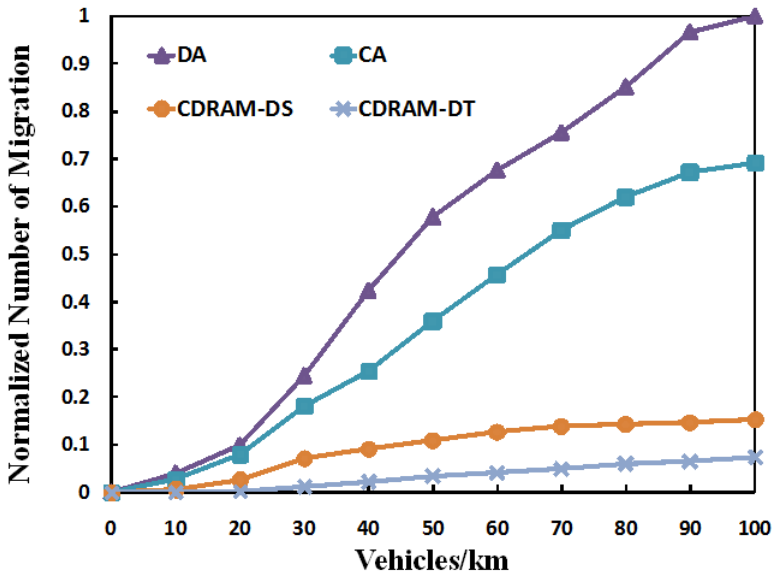
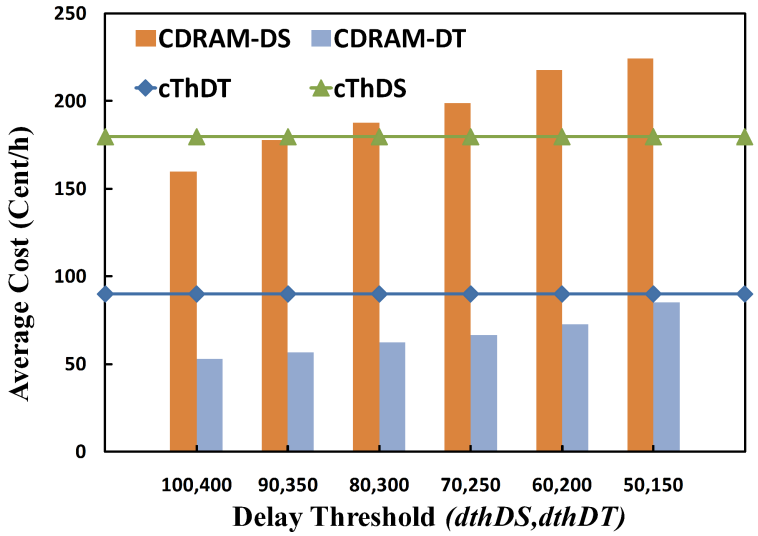



Figure 4.7: Number of migrations

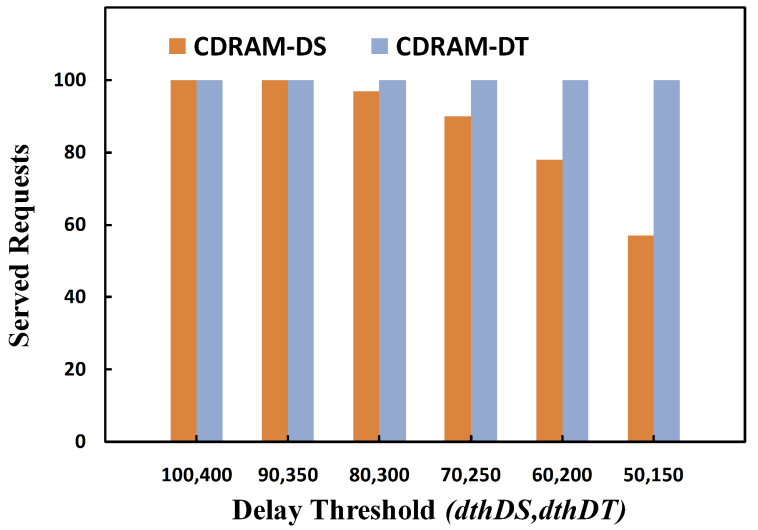
a price higher than $cTh_{DS}$ to satisfy the delay requirement and serve maximum number of DS requests. However, CDRAM-DS still performs better than DA, as it tries to reduce the extra cost above $cTh_{DS}$. CDRAM performs better since it classifies the requests based on the delay and balances the load across the DCs.

**Number of migrations:** Figure. 4.7 shows that CDRAM reduces the VM migrations up to 72% and 90% for DS and DT services, respectively. It is because CA and DA checks the migration decision for every request of a user (causing unnecessary migrations), while CDRAM checks if migration is required only when the vehicle moves to another RSU. CDRAM considers both delay and cost thresholds for migration. CDRAM-DS migrates only if it finds a cheaper DC or that gives lesser delay, while CDRAM-DT only migrates VM to a cheaper DC that satisfies $dTh_{DT}$. Hence, CDRAM-DT leads to fewer migrations than CDRAM-DS.

**Impact of Delay Threshold:** To study the impact of the delay threshold on the cost and the requests served, we fixed the vehicle density at 50 vehicles/km, and varied $dTh_{DS}$ between 100 and 50 $ms$, and $dTh_{DT}$ between 400 and 150 $ms$. Figure. 4.8a shows that there is a trade-off between the delay and the cost. As the delay threshold decreases, the cost increases for both DS and DT services. We can observe that the service provider still




(a) Average cost




(b) Served requests

Figure 4.8: Impact of delay threshold on the cost and the served requests

could match the cost threshold while serving DT requests, while it has to bear an extra cost for the DS services as $dTh_{DS}$ reduces beyond 80 $ms$. Figure. 4.8b depicts that all DT requests are served as $dTh_{DT}$ is relatively large even after decreasing it, and the service provider has more DC choices. Some of the DS requests have to be dropped when $dTh_{DT}$ decreases below 80, as there are limited DCs to choose from that can serve all DS requests within the delay threshold with the DCs available.

## 4.4 Summary

In this chapter, we introduced the concept of federation in vehicular cloud computing and termed the architecture FVCC. We formulated the problem of dynamic resource allocation in FVCC to satisfy the delay requirements of user requests and at the same time, minimize the operating cost of the service provider. We presented a practical dynamic pricing model for sharing resources across CPs and a cost-efficient and delay-aware algorithm for hosting vehicular services in a federated cloud. The proposed algorithm is in polynomial time and works online to migrate VMs to cater to QoS requirements. Experimental evaluation shows that FVCC balances the requirements of both the users and the service provider. The proposed algorithm outperforms baseline approaches in terms of delay and cost and helps to serve more requests. In this chapter, inter-DC VM migration has been utilized in FVCC to minimize the cost and the delay. We tried to reduce VM migration overhead by minimizing the number of migrations and migrating a VM only when required. However, understanding the exact impact of the VM migration across the data centers in a federated cloud is very important. In the next chapter, we will discuss inter-DC migration performance. We will also consider the impact of various parameters that affect migration performance.

CHAPTER

# 5

# Towards Cost-aware VM Migration to Maximize the Profit in Federated Clouds

The previous chapters have focused on resource management for hosting various applications in federated cloud. In Chapter 3, we discussed the static resource allocation for data-intensive applications without including migration. In Chapter 4, we studied the dynamic resource allocation for vehicular applications where migration is utilized to tackle user mobility and meet the delay requirements of user applications and the cost constraints of the CP. However, VM migration across geographically distributed data centers of a federation is challenging and it comes with lots of extra overhead. In this chapter, we address the problem of cost-aware inter-DC VM migration where we model the performance of VM migration across DCs. We propose an inter-DC VM migration algorithm for maximizing the profit of the CP while considering the migration overhead in terms of cost and time.

As already discussed in Chapter 1, live VM migration is a key technique used for dynamic resource management in cloud data centers. Inter-DC VM migration has emerged as a challenging problem to take advantage of the diversity among DCs and to optimize the resource allocation for objectives, such as load balancing, improving the QoS, and minimizing the cost [20]. However, VM migration between DCs is a challenging task for the CP as it requires not only the transfer of memory image but also the data stored by a VM, that could result in a large amount of data to be migrated over WAN links. For some applications, such as data-intensive applications, inter-DC migration incurs huge bandwidth cost and causes longer downtime. Therefore, analysing the performance of VM migration in a federated setup is very essential.

**Motivation** Though the problem of live VM migration has been widely studied in the literature, most of them focused on only a single DC, wherein the migration happens over LAN with minimum migration overhead. Only a few works addressed the live VM migration between the DCs over WAN links, mostly without considering the migration cost. Inter-DC VM migration in a federation poses new challenges due to the variations in resource prices, network bandwidth, and data transfer prices. As mentioned earlier, the main concern of the CP in a federation is to reduce its own TCO. Live VM migration can be exploited in a federated cloud to minimize the cost of the CP by migrating VMs from DCs with higher resource price to a DC with lower prices. However, it brings in extra cost which might surpass the profit gained due to the migration. It also might lead to longer migration time which may degrade the performance. Most of the works that addressed the problem of live VM migration also assumed the VMs to be homogeneous. However, the characteristics of the VM and the features of the workload highly affect the migration overhead. Therefore, studying the exact impact of these parameters on the migration in terms of cost and time are essential. Furthermore, the destination DC should not be selected based on only the resource prices, but also by considering the cost of data transfer between the DCs. An efficient migration algorithm needs to consider migration overhead while minimizing the cost of the CP. It should intelligently select the VM to be migrated and the destination DC for migration to maximize the migration profit with minimum migration overhead.

In this chapter, we address the problem of inter-DC VM migration in federated cloud to minimize the TCO of the CP. The problem at hand is how to use inter-DC live migration effectively to maximize the profit of the CP, while minimizing the migration cost, the number of migrations, the migration time, and the downtime. We model VM cost in federated cloud considering two cases; the VMs hosted locally and outsourced VMs. We also present a model to estimate the migration performance including total migrated data, migration cost, and migration time and downtime. Based on the proposed cost models, we formulate the problem of inter-DC VM migration as an optimization problem with an objective to minimize the TCO and the migration time. We introduce three heuristics for selecting the source and destination DCs for migration and the VM to be migrated considering several parameters. Based on these heuristics, We propose a polynomial-time VM migration algorithm that migrates only eligible VMs whose migration maximizes the benefit of migration.

**Organization of the Chapter:** The rest of this chapter is structured as follows. Section 5.1 discusses the system model, including the models for VM cost and migration performance, and then discusses the problem formulation. The proposed heuristics and the proposed algorithm to solve the problem are described in section 5.2. Section 5.3 discusses the simulation results, and then the chapter is summarized in Section 5.4.

## 5.1 System Model and Problem Formulation

In this section, we describe the system model, present a pricing model for VM costing, a model for migration cost, migration time and migration downtime in a federated cloud. Then, we formulate the problem of cost-aware inter-DC VM migration in federated cloud as an optimization model.

### 5.1.1 System Model

Figure. 5.1 shows the architecture of a federated cloud with multiple CPs, where each CP in turn has several DCs distributed across different geographical regions. The hierarchical management framework presented in Section 3.1 is used for resource management where each DC has a DCM, and each CP has a FCM. The DCM handles VM scheduling within its DC while the FCM manages the resource allocation at the federation level through coordination between its DCMs and FCMs of other CPs. The FCM is also responsible for resource re-allocation and VM migration across the DCs based on the objectives of the CP and to minimize its operating cost.

Let $\mathbb{P}$ and $\mathbb{D}$ be the set of CPs and the set of all the data centers in a federated cloud, respectively. Let $D_p$ be the set of DCs in a CP, $p \in \mathbb{P}$ and $\mathcal{D}_{pk} \in D_p$ be the $k^{th}$ data center of the $p^{th}$ CP. Each DC in a federation $\mathcal{D}_{pk}$ is characterized by its resource capacity; number of virtual CPUs (vCPUs) $\mathcal{D}_{pk}^{cpu}$, memory $\mathcal{D}_{pk}^{mem}$, and storage capacity $\mathcal{D}_{pk}^{str}$. Let the bandwidth of the WAN link between two data centers of a federation, say $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ be denoted by $BW_{pk,ql}$ and the corresponding data transfer price be $DP_{pk,ql}$. CPs offers several types of VMs to meet the requirements of different applications, such as, standard VM with balanced resources, compute-optimized, memory-optimized, and storage-optimized VMs [60, 61]. Let $\mathbb{V}$ be the set of all VMs allocated across the DCs of a federation. Each virtual machine $VM_i \in \mathbb{V}$ has a resource configuration, denoted by $VM_i^{cpu}$ vCPU cores,

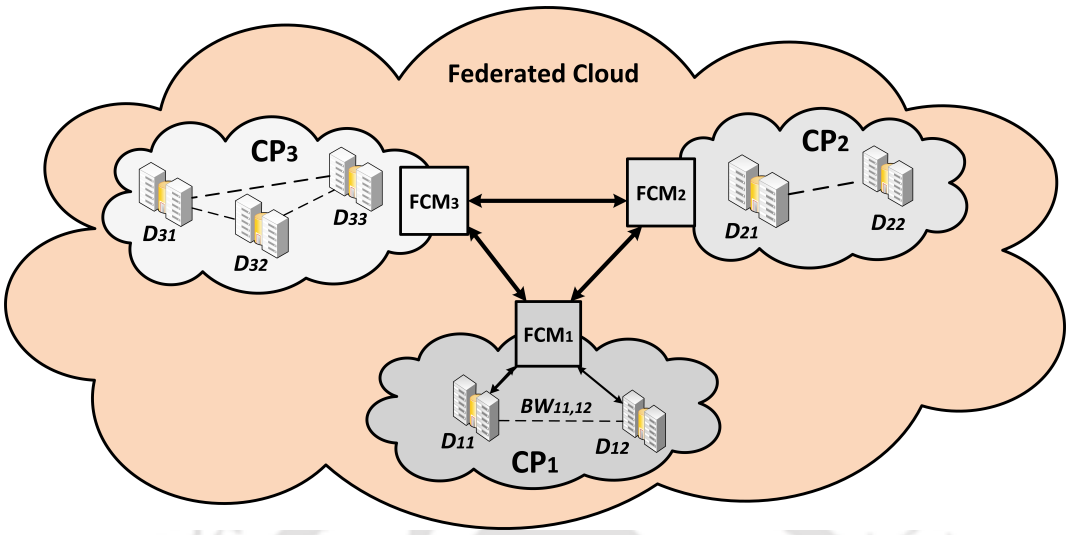



Figure 5.1: Federated cloud system model

$VM_i^{mem}$ memory, and $VM_i^{str}$ storage. Let $VM_i^t$ be the lifetime of a $VM_i$. We also use three binary variables; $y_i$, such that $y_i = 1$ if the $VM_i$ is eligible for migration, $x_i^{p,ql}$ and $m_i^{pk,ql}$ to represent the placement and the migration of $VM_i$, respectively, defined below:

$$x_i^{p,ql} = \begin{cases} 1, & \text{if } VM_i \text{ is placed in } \mathcal{D}_{ql} \text{ by CP } p,\ p, q \in \mathbb{P} \\ 0, & \text{Otherwise} \end{cases}$$

$$m_i^{pk,ql} = \begin{cases} 1, & \text{if } VM_i \text{ is migrated from } \mathcal{D}_{pk} \text{ to } \mathcal{D}_{ql} \\ 0, & \text{Otherwise} \end{cases}$$

### 5.1.2 VM Cost Model

We model the operating cost of a VM based on its power consumption. Since CPU is the main component contributing to the power consumption [41, 141], we express the power consumption of a VM $VM_i$ running on a server $v$ as

$$VM_i^{power} = \frac{SP_v}{S_v^{cpu}}\ VM_i^{cpu} \tag{5.1}$$

where $SP_v$ is the power consumption of the server $v$ (in KWh) [57], $S_v^{cpu}$ is the number of vCPU cores in $v$, and $VM_i^{cpu}$ is the number of vCPU cores of $VM_i$. To consider the power consumed by other facilities, such as the cooling system, PUE is normally included

in calculation of the operating cost of a VM. Since the electricity price and PUE could vary across the DCs of a federation, the cost of a VM also might depend on the location. We express the operating cost of $VM_i$, when it is running in a data center $\mathcal{D}_{pk}$ (in \$/h) as

$$opCost_i^{pk} = VM_i^{power} \ PUE_{pk} \ E_{pk} \tag{5.2}$$

where $PUE_{pk}$ is the PUE of $\mathcal{D}_{pk}$, and $E_{pk}$ is the electricity price in $\mathcal{D}_{pk}$ (\$/KWh). In a federated cloud system, a CP can maximize its profit by renting-out its free resources to other CPs based on the dynamic pricing model which has been introduced in previous chapter (see Section 4.1.3). The price offered to other CPs, $fedprice$ is less than that offered to the end users, and it ranges between the operating cost and the user price based on the resource availability. $fedprice_i^{pk}$, the price of renting $VM_i$ hosted in $\mathcal{D}_{pk}$ to other CPs (in \$/h) is expressed as

$$fedPrice_i^{pk} = opCost_i^{pk} + \ U^{pk}(userPrice_i^{pk} - opCost_i^{pk}) \tag{5.3}$$

where $U^{pk} \in [0-1]$ is the resource utilization at $\mathcal{D}_{pk}$ and $userPrice_i^{pk}$ is the price offered (to the users) for hosting $VM_i$ at $\mathcal{D}_{pk}$ (\$/h).

We combine Eq. 5.2 and Eq. 5.3 to present a comprehensive price model for hosting a VM, $VM_i$ in a DC, $\mathcal{D}_{pk}$ as

$$\begin{aligned} price_i^{pk} = opCost_i^{pk} + x_i^{q,pk} U^{pk}(userPrice_i^{pk} - opCost_i^{pk}), \\ \forall i \in \mathbb{V}, \mathcal{D}_{pk} \in D_p, \ \ p,q \in \mathbb{P}, p \neq q \end{aligned} \tag{5.4}$$

This equation indicates that, when a VM is placed in an own DC, then the operating cost is considered. Otherwise, if the VM is rented to another CP, then $fedPrice$ is considered.

### 5.1.3 VM Migration Performance Model

In this work, we assume the most common technique used in live migration, $pre-copy$ migration, discussed in Chapter 2 (see Section 2.3). It includes mainly two stages as illustrated in Figure. 5.2 [20]. In the *pre-copy* stage, the basic image of the VM, including memory and storage data, is transmitted from the source to the destination DC. Then, the dirty data is transferred iteratively till reaching the termination condition. At this point,

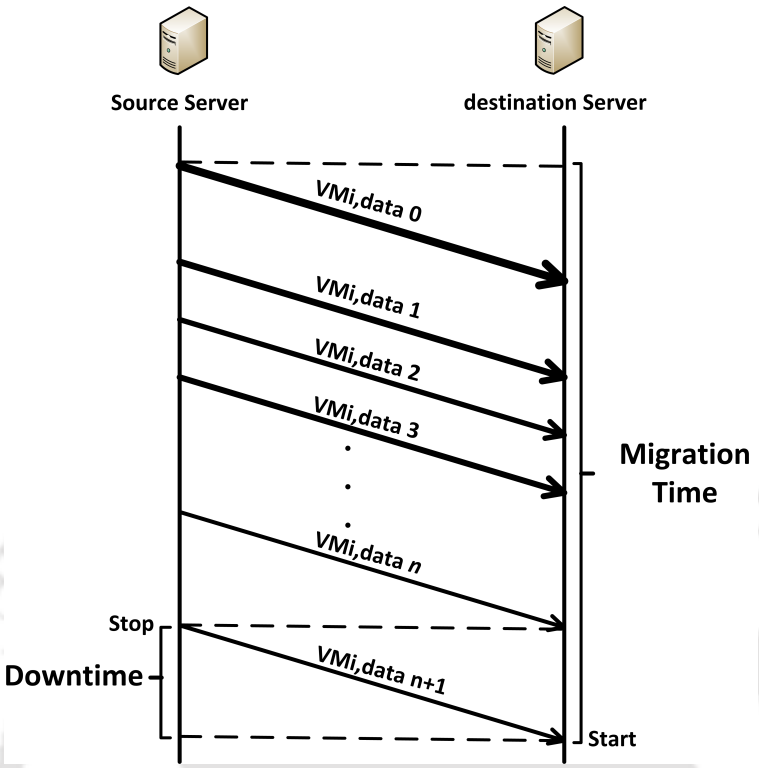



Figure 5.2: Precopy live migration

the second stage, *stop-and-copy*, starts. It suspends the migrated VM in the source DC and transfers the remaining data to the destination DC. Then the new VM is resumed in the destination DC, and the VM in the source DC is released.

Based on the stop condition defined in the migration algorithm of Xen platform [20], we set the termination condition using three criteria: (i) number of iterations in pre-copy stage reached 29, (ii) less than 500 MB was modified during the last pre-copy iteration, and (iii) more than three times the size of the VM has been transferred to the destination DC.

The performance of VM migration is typically measured by the total migrated data, the migration time and the downtime. Next, we model the total migrated data, and based on that we model the migration cost, the migration time, and the downtime of a VM. *Total migrated data:* The size of data transferred for $i^{th}$ VM, in $j^{th}$ iteration, denoted by $VM_{i,data(j)}$ depends on the link bandwidth, the duration of the previous iteration, and the PDR of the VM. The PDR varies across VMs based on the application being executed on the VM. We can express $VM_{i,data(j)}$ by

$$VM_{i,data(j)} = \frac{VM_{i,data(j-1)}}{BW_{Src,Des}} \; VM_i^{PDR}, \;\; \forall\, j \in [1, n+1] \tag{5.5}$$

where $VM_i^{PDR}$ is the PDR of $VM_i$ ($GB/sec$), and $BW_{Src,Des}$ is the bandwidth between the source and the destination ($GBps$). Note that the PDR should be less than the bandwidth for effective migration, otherwise the modified data in each iteration will be more than the transmitted data. We express the total size of the transmitted data (in $GB$) during migration of $VM_i$ by

$$VM_i^{Mdata} = VM_{i,data(0)} + \sum_{j=1}^{n+1} VM_{i,data(j)} \tag{5.6}$$

where $VM_{i,data(0)}$ is the size of basic image of $VM_i$ (including memory and storage data), $n$ is the number of iterations for transferring dirty data till the termination condition is met, and $n+1$ is the last iteration after suspending the VM.

*Migration Cost:* We define the cost of migration to be the cost of data transfer resulting from the migration process. Hence, we express the cost of inter-DC migration of $VM_i$ from $\mathcal{D}_{pk}$ to $\mathcal{D}_{ql}$ (in \$) as

$$MC_i^{pk,ql} = VM_i^{Mdata}\ m_i^{pk,ql}\ DP_{pk,ql} \tag{5.7}$$

where $VM_i^{Mdata}$ is the total migrated data of $VM_i$ (in $GB$), $DP_{pk,ql}$ is the price of data transfer (in \$/$GB$).

*Migration Time:* It is the duration between the time when the migration process starts and the time when the source VM can be discarded [20]. The time required to migrate $VM_i$ from $\mathcal{D}_{pk}$ to $\mathcal{D}_{ql}$ (in s) is expressed as

$$MT_i^{pk,ql} = \frac{VM_i^{Mdata}}{BW_{pk,ql}}\ m_i^{pk,ql} \tag{5.8}$$

where $BW_{pk,ql}$ is the available link bandwidth between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ (in $GBps$).

*Downtime:* It is the time for which the VM is suspended [20]. The downtime of $VM_i$ while migrating it between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ (in $Sec$) can be expressed as

$$DT_i^{pk,ql} = \frac{VM_{i,data(n+1)}}{BW_{pk,ql}}\ m_i^{pk,ql} \tag{5.9}$$

where $VM_{i,data(n+1)}$ is the remaining dirty data after last round of iterative data transfer (in $GB$).

### 5.1.4 Problem Formulation

We formulate the problem of cost-aware VM migration to maximize the profit of the cloud provider in a federated cloud. To maximize its own profit, a cloud provider, $p$ can allocate and migrate its VMs to any data center $\mathcal{D}_{ql} \in \mathbb{D}$, that might be a local one ($p = q$) or an outsourced one ($p \neq q$). The TCO of $p^{th}$ cloud provider, $TCO_p$ includes the operating cost of all its VMs and the cost of their migration as well. In this work, we propose an optimization model for the addressed problem with an objective of minimizing both the $TCO_p$ and the migration time, subject to a set of constraints. Table 5.1 summarizes the notation used in the system model and the problem formulation.

We define $TVC_p$, the total VM cost of the $p^{th}$ CP, as the operational expenses of all VMs served by $p$. Due to migration, a VM may be placed in different DCs at different times and thus $TVC_p$ is expressed as

$$TVC_p = \sum_{i \in \mathbb{V}} \sum_{\mathcal{D}_{ql} \in \mathbb{D}} x_i^{p,ql} \; price_i^{ql} \; t_i^{ql} \tag{5.10}$$

where $price_i^{ql}$ is the price of hosting $VM_i$ at $\mathcal{D}_{ql}$, and $t_i^{ql}$ is the duration of running $VM_i$ at $\mathcal{D}_{ql}$.

We calculate the total migration cost of $p^{th}$ CP, denoted by $TMC_p$ as the sum of the costs of all VM migrations done by $p^{th}$ CP from any of its DCs ($\mathcal{D}_{pk} \in D_p$) to any other DC in the federation. Therefore,

$$TMC_p = \sum_{\mathcal{D}_{pk} \in D_p} \sum_{\substack{\mathcal{D}_{ql} \in D \\ \mathcal{D}_{pk} \neq \mathcal{D}_{ql}}} \sum_{i \in \mathbb{V}} m_i^{pk,ql} \; MC_i^{pk,ql} \tag{5.11}$$

Consequently, the TCO of the $p^{th}$ CP is written as

$$TCO_p = \; TVC_p \; + \; TMC_p \tag{5.12}$$

Similarly, we define the total migration time of the CP $p$, denoted by $TMT_p$, as

$$TMT_p = \sum_{\mathcal{D}_{pk} \in D_p} \sum_{\substack{\mathcal{D}_{ql} \in D \\ \mathcal{D}_{pk} \neq \mathcal{D}_{ql}}} \sum_{i \in \mathbb{V}} m_i^{pk,ql} \; MT_i^{pk,ql} \tag{5.13}$$

Table 5.1: Notation used in the system model

| ***Symbol*** | ***Description*** |
|---|---|
| $\mathbb{P}$ | Set of all cloud providers in the federation |
| $\mathbb{D}$ | Set of all data centers in the federation |
| $D_p$ | Set of the data centers of a cloud provider $p$ |
| $\mathcal{D}_{pk}$ | $k^{th}$ data center of cloud provider $p$ |
| $\mathcal{D}_{pk}^{cpu}$ | CPU capacity of data center $\mathcal{D}_{pk}$ |
| $\mathcal{D}_{pk}^{mem}$ | Memory capacity of data center $\mathcal{D}_{pk}$ |
| $\mathcal{D}_{pk}^{str}$ | Storage capacity of data center $\mathcal{D}_{pk}$ |
| $PUE_{pk}$ | PUE of data center $\mathcal{D}_{pk}$ |
| $E_{pk}$ | Electricity price in $\mathcal{D}_{pk}$ |
| $BW_{pk,ql}$ | Link bandwidth between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ |
| $DP_{pk,ql}$ | Data transfer price between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ |
| $\mathbb{V}$ | Set of all VMs in the federation |
| $VM_i^{cpu}$ | CPU demand of $VM_i$ |
| $VM_i^{mem}$ | Memory demand of $VM_i$ |
| $VM_i^{str}$ | Storage demand of $VM_i$ |
| $VM_i^{PDR}$ | Page dirty rate of $VM_i$ |
| $VM_i^{t}$ | Life time of $VM_i$ |
| $VM_i^{power}$ | Power consumption of $VM_i$ |
| $opCost_i^{pk}$ | Operating cost of $VM_i$ at $\mathcal{D}_{pk}$ |
| $userPrice_i^{pk}$ | User price of $VM_i$ at $\mathcal{D}_{pk}$ |
| $price_i^{pk}$ | Cloud provider price of $VM_i$ at $\mathcal{D}_{pk}$ |
| $y_i$ | $VM_i$ is eligible for migration |
| $x_i^{p,ql}$ | $VM_i$ is placed in $\mathcal{D}_{ql}$ by CP $p$ |
| $m_i^{pk,ql}$ | $VM_i$ is migrated from $\mathcal{D}_{pk}$ to $\mathcal{D}_{ql}$ |
| $MC_i^{pk,ql}$ | Migration cost of $VM_i$ between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ |
| $MT_i^{pk,ql}$ | Migration time of $VM_i$ between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ |
| $DT_i^{pk,ql}$ | Downtime of $VM_i$ between $\mathcal{D}_{pk}$ and $\mathcal{D}_{ql}$ |
| $TVC_p$ | Total operating cost of all VMs of CP $p$ |
| $TMC_p$ | Total migration cost of all VMs of CP $p$ |
| $TCO_p$ | Total cost of operation of CP $p$ |
| $TMT_p$ | Total migration time of CP $p$ |
| $\alpha$, $\beta$ | Weight factors |

Finally, we formulate the problem as

$$\textbf{min} \quad \alpha \; TCO_p \; + \; \beta \; TMT_p \tag{5.14}$$

**subject to**

$$y_i \, m_i^{pk,ql} \, (VM_i^t \, price_i^{pk} \; - \; VM_i^t \, price_i^{ql}) > MC_i^{pk.ql} \tag{5.15}$$

$$\sum_{i \in \mathbb{V}} x_i^{p,ql} \; VM_i^{cpu} \; + \sum_{i \in \mathbb{V}} m_i^{pk,ql} \, {VM_i}^{cpu} \leq \mathcal{D}_{ql}^{cpu} \tag{5.16}$$

$$\sum_{i \in \mathbb{V}} x_i^{p,ql} \; VM_i^{mem} \; + \sum_{i \in \mathbb{V}} m_i^{pk,ql} \, {VM_i}^{mem} \leq \mathcal{D}_{ql}^{mem} \tag{5.17}$$

$$\sum_{i \in \mathbb{V}} x_i^{p,ql} \; VM_i^{str} \; + \sum_{i \in \mathbb{V}} m_i^{pk,ql} \, {VM_i}^{str} \leq \mathcal{D}_{ql}^{str} \tag{5.18}$$

$$\sum_{\mathcal{D}_{ql} \in \mathbb{D}} x_i^{p,ql} = 1 \tag{5.19}$$

$$x_i^{p,ql} \in \{0, 1\}, \;\; \forall i \in \mathbb{V}, \;\; \forall p, q \in \mathbb{P}, \;\; \forall \mathcal{D}_{pk}, \mathcal{D}_{ql} \in \mathbb{D} \tag{5.20}$$

Eq. 5.14 defines the objective as a joint minimization of the TCO and the migration time for the $p^{th}$ CP, where $\alpha$ and $\beta$ are the weight parameters. Constraint (5.15) indicates that the migration of any VM should be profitable, i.e. the cost saving due to migration of $VM_i$ from $\mathcal{D}_{pk}$ to $\mathcal{D}_{ql}$ should be greater than the cost of its migration between these two DCs. We include $y_i$ to ensure migrating only VMs that are eligible for migration. If a VM is not eligible for migration ($y_i = 0$) then the left side of constraint (5.15) (the migration profit) would be zero, definitely less than the migration cost. Thus, this constraint is not satisfied and $VM_i$ is not migrated. For any DC, $\mathcal{D}_{ql} \in \mathbb{D}$, the constraint (5.16) states that the accumulated compute resources of all VMs that are placed and those that are migrated to a DC should not exceed the compute capacity of this DC. Similarly, constraints (5.17) and (5.18) define the constraints of resource availability with respect to the memory and storage resources. Eq. 5.19 ensures that, at any time, every VM of the $p^{th}$ CP must be placed in one and only one DC of the federation. Eq. 5.20 defines the domain of the variables.

The above formulation is a variant of dynamic bin packing problem which is NP-hard, where VMs are the items and DCs are the bins, and VMs can be migrated across the DCs [131]. Hence, the formulated problem is NP-hard. We propose a heuristic algorithm to solve the problem in polynomial time.

## 5.2 Proposed Algorithm

A VM migration algorithm is very crucial in meeting the objectives of the CP, as it is used to select the source and the destination of the migration and also the VM to be migrated. The proposed algorithm tries to minimize the TCO of $p^{th}$ CP by migrating some VMs considering the variation in the cost across the DCs, while considering the migration cost and time. We select the source DC, $SrcDC \in D_p$ as the one that has the highest operating cost (based on electricity prices and PUE), using a heuristic as follows

$$h_{SrcDC} = E_{pk}\, PUE_{pk} \tag{5.21}$$

Next, we select the VM from the source DC, whose migration returns the highest benefit with least migration time. Since the operating cost of a VM is directly related to its energy consumption, which in turn depends on the CPU usage, migrating a VM with higher CPU leads to larger cost saving. However, considering only CPU configuration of a VM is not enough, because VMs with higher data size need more migration time. Hence, we also consider the VM data size in the decision making process. In live migration, the migration time is not only affected by the data size but also by the PDR. Accordingly, the best VM for migration is the one with larger number of vCPU cores, lower data size, and lower PDR. Therefore, we propose the a heuristic to select the source VM as follows:

$$h_{SrcVM} = \frac{VM_i^{cpu}}{VM_{i,data(0)}\ VM_i^{PDR}} \tag{5.22}$$

To choose a destination DC for the VM to be migrated, we utilize the price variation across DCs to maximize the profit. However, the cost of migration resulting from transferring the data of the migrated VM may exceed the profit gained from the migration. Therefore, we do not only consider the VM prices at the destination, but also the data transfer cost that

**ALGORITHM 7: CAVMA** Cost **A**ware **V**M **M**igration **A**lgorithm

**Input:** $D_p$: The list of DCs of CP $p$
$\mathbb{V}$: The list of virtual machines

**Output:** $ResMap(VM_i, \mathcal{D}_{ql})$

```
1  Initialization:
2      SrcDC ← NULL;
3      SrcVM ← NULL;
4      DesDC ← NULL;
5  Sort D_p in desc order based on h_SrcDC;
6  SrcDC ← D_p.head;
7  while (SrcDC ≠ NULL) do
8      Sort V_SrcDC in desc order based on h_SrcVM;
9      SrcVM ← V_SrcDC.head;
10     while (SrcVM ≠ NULL) do
11         if (y_SrcVM ≠ 0) then
12             DesDC=SelectDesDC(SrcVM, SrcDC);
13             if (DesDC ≠ NULL) then
14                 Migrate (SrcVM, SrcDC, DesDC);
15                 Update ResMap(SrcVM, DesDC);
16         SrcVM ← SrcVM.next;
17     SrcDC ← SrcDC.next;
18 return ResMap;
```

impacts the migration cost. We define *migration benefit*, $MB_i^{pk,ql}$ to indicate the profit that can be obtained by migrating $VM_i$ from $\mathcal{D}_{pk}$ to $\mathcal{D}_{ql}$ after subtracting the cost of migration as

$$MB_i^{\mathcal{D}_{pk},\mathcal{D}_{ql}} = VM_i^t \left(price_i^{pk} - price_i^{ql}\right) - MC_i^{pk.ql} \tag{5.23}$$

where the life time of $VM_i$ is taken as a factor since the migration profit increases for VMs with longer life time. We use Eq.5.23 to select the potential destination DC ($\mathcal{D}_{ql}$).

We propose Cost-Aware VM Migration Algorithm (**CAVMA**) to select the source DC, the source VM, and the destination DC for migration based on the aforementioned heuristics. Algorithm 7 shows the steps in the algorithm. It takes as input $D_p$, the list of DCs of CP $p$ and $\mathbb{V}$, the list of VMs, to return $ResMap$, the updated VM placement after migration ($ResMap$ maps VMs to DCs). The steps of the algorithm are as follows:

1. First, the algorithm initializes the variables $SrcDc$, $SrcVM$, and $DesDc$ for migration (lines 1-4). It then sorts $D_p$ in decreasing order based on $h_{SrcDC}$ (computed using Eq.5.21). Then, it selects the first DC in sorted $D_p$ as $SrcDC$, that gives the highest

operating cost (lines 5-6).

2. It sorts $V_{SrcDC}$, the list of VMs placed in $SrcDC$ in decreasing order based on $h_{SrcVM}$ (using Eq.5.22) and then takes the first VM as $SrcVM$ (lines 8-9).

3. If $SrcVM$ is eligible for migration ($y_i = 1$), then select the destination DC for migration $DesDC$, otherwise the algorithm moves to the next VM (line 11).

4. The algorithm selects $DesDC$ by calling **SelectDesDC** algorithm (explained later) (line 12).

5. If a suitable $DesDC$ is found that makes the migration of $SrcVM$ beneficial with a relatively short migration time, the algorithm migrates $SrcVM$ from $SrcDC$ to $DesDC$, and then updates the location of $SrcVM$ in $ResMap$ (lines 13-15).

6. It repeats Steps (3) to (5) for all VMs in $SrcDC$ by checking sorted $V_{SrcDC}$ sequentially, and it migrates only VMs that are eligible for migration and whose migration is beneficial (lines 10-16).

7. The algorithm repeats Steps (2) to (6) for all DCs of CP $p$ in order ($\forall\, SrcDC \in D_p$) (lines 7-17).

8. Finally, the execution of the migration algorithm returns $ResMap$, the updated location of VMs of CP $p$ (line 18).

Next, we discuss the algorithm, **SelectDesDC** to select the best destination DC for migrating a VM, that returns the highest profit with a relatively short migration time. Algorithm 8 shows the steps of the proposed algorithm. It takes as input, $SrcVM$ and $SrcDC$, the source VM and the source DC for migration, respectively, and $\mathbb{F}-$the list of FCMs of the CPs in the federation. It returns $DesDC$, the selected Destination DC in the federation ($DesDC \in \mathbb{D}$) to migrate $SrcVM$. We describe the steps of the algorithm below:

1. The algorithm defines $DClist$ as the list of DCs in the federation which includes initially only local DCs $D_p$, and it initializes $DesDC$ to $NULL$ (lines 1-3).

2. $F_p$ contacts FCMs and requests information of their DCs (resource availability and price), based on which it updates $DClist$ (lines 4-7).

**ALGORITHM 8: SelectDesDC**

**Input:** $SrcVM$: The source VM to be migrated
$SrcDC$: The source DC of migration
$\mathbb{F}$: List of FCMs

**Output:** $DesDC$: The selected destination DC

```
1  Initialization:
2      DClist ← D_p;
3      DesDC ← NULL;
4  for each F_q ∈ 𝔽 do
5  |   Request(resAvail^ql, fedPrice_i^ql) of all 𝒟_ql ∈ D_q;
6  |   Receive(𝒟_ql, resAvail^ql, fedPrice_i^ql) from F_q;
7  |   Update DClist;
8  for each 𝒟_ql ∈ DClist do
9  |   Calculate MB_SrcVM^{SrcDC,𝒟_ql};
10 Sort DClist in Desc order based on MB_SrcVM^{SrcDC,𝒟_ql};
11 DesDC ← DClist.head;
12 if (DesDC == SrcDC) then
13 |   return NULL;
14 while (MB_SrcVM^{SrcDC,DesDC} > 0) do
15 |   if ( (resAvail^DesDC ⩾ SrcVM^resDemand) ∧ (VM_SrcVM^PDR < BW_SrcDC,DesDC)) then
16 |   |   return DesDC;
17 |   else
18 |   |   DesDC ← DesDC.next;
19 return NULL;
```

3. The algorithm calculates the benefit of migrating $SrcVM$ to each DC in $DClist$ using Eq.5.23. Then, it sorts $DClist$ in decreasing order of migration benefit and selects the first DC that gives the highest migration benefit as potential $DesDC$ (lines 8-11).

4. If $DesDC$ is same as $SrcDC$, it means that the current DC of $SrcVM$ is the best location for it and there is no benefit in migrating $SrcVM$ to any other DC. The algorithm returns $NULL$ (line 12-13). In this case, there might be profit due to migrating $SrcVM$ to another DC, but the migration cost is more than the profit; that leads to a negative migration benefit increasing the TCO instead of decreasing it.

5. Otherwise, the algorithm scans the sorted $DClist$ and selects $DesDC$ as the first DC with (i) positive migration benefit, (ii) it has enough resources for $SrcVM$, and (iii) the available bandwidth between $SrcDC$ and this $DesDC$ is larger than the PDR of $SrcVM$, which is to minimize the migration time and the downtime (lines 14-18).

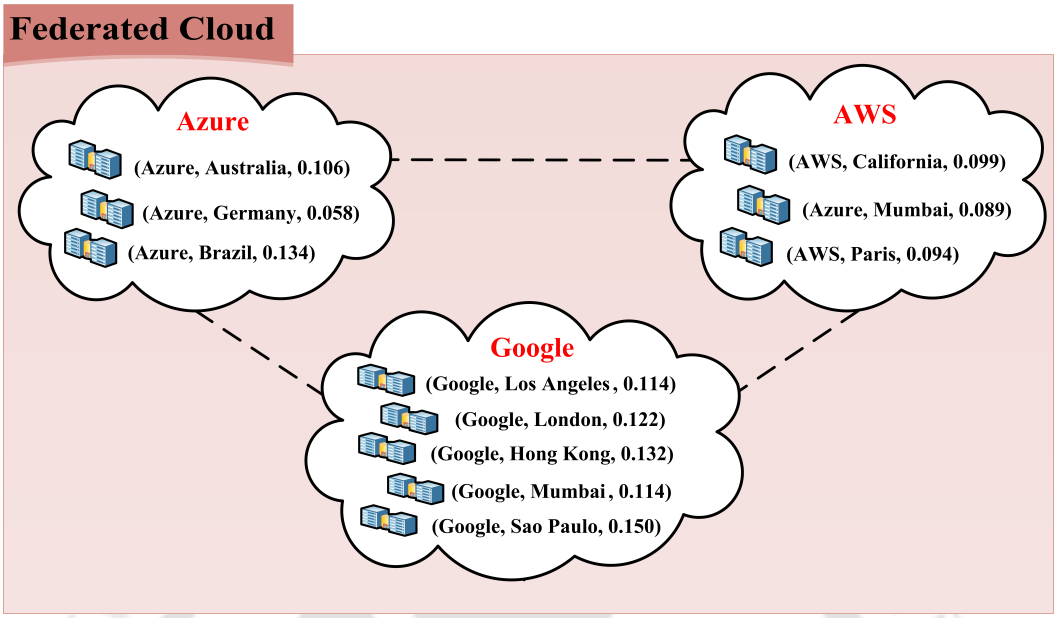



Figure 5.3: Federated cloud system for simulation

6. The algorithm returns $NULL$ if there is no DC that meets the previous conditions (line 19).

In summary, the proposed algorithm minimizes the cost of the CP $p$ by migrating only the VMs, whose migration gives a profit by selecting the VMs to give the highest migration benefit. While deciding the destination DC, it considers the network bandwidth and data transfer price to minimize the migration time, downtime, and the migration cost.

**Time and Space Complexity:** Let $L$ be the number of DCs of the CP $p$ ($|D_p|$), $M$ be the of number DCs in the federation ($|\mathbb{D}|$), and $N$ be the number of VMs ($|\mathbb{V}|$) where ($L < M << N$). The algorithm first sorts $D_p$ in $\mathcal{O}(L \log(L))$, then it sorts $\mathbb{V}$ in $\mathcal{O}(N \log(N))$. Next, for every $VM_i \in \mathbb{V}$, **SelectDesDC** sorts $\mathbb{D}$, that takes $\mathcal{O}(NM \log(M))$. Therefore, the time complexity of the algorithm is $\mathcal{O}(N(\log N + M \log M))$. The space complexity is $\mathcal{O}(L + M + N)$; that is $\mathcal{O}(N)$.

## 5.3 Simulation Results

In this section, we present the results from an evaluation of the proposed algorithm (**CAVMA**) using CloudSim toolkit, the most common simulator for cloud environment [81].

Table 5.2: Simulation parameters

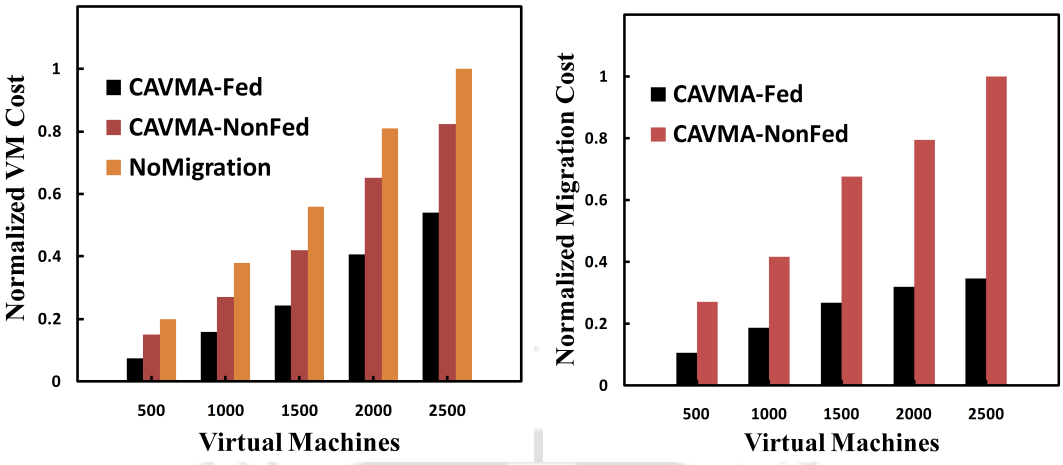
| Parameter | Description | Value |
|---|---|---|
| $\mathcal{D}_{pk}^{cpu}$ | Data center CPU capacity | 20000-100000 vCPU |
| $\mathcal{D}_{pk}^{mem}$ | Data center memory capacity | 50000 - 500000 GB |
| $\mathcal{D}_{pk}^{str}$ | Data center storage capacity | 1000-10000 TB |
| $BW_{pk,ql}$ | Link bandwidth between DCs | 10-100 Gbps |
| $DP_{pk,ql}$ | Bandwidth price between DCs | 0.01-0.08 $/GB |
| $VM_i^{cpu}$ | CPU demand of $VM_i$ | 2-96 vCPU |
| $VM_i^{mem}$ | Memory demand of $VM_i$ | 2-624 GB |
| $VM_i^{str}$ | Storage demand of $VM_i$ | 10-10000 GB |
| $VM_i^{PDR}$ | Page dirty rate of $VM_i$ | 0-30 Gbps |
| $VM_i^{t}$ | Life time of $VM_i$ (in hours) | 1 hour- 6 months |

Figure. 5.3 shows the federated cloud system considered in the simulation. It includes eleven DCs distributed across four different geographical regions (North America, South America, Europe, and Asia), owned by three CPs. Each DC is characterized by its owner CP, its location, and its resource price. DCs have different resource capacities, and they are connected through WAN links with diverse bandwidths and data transfer prices. According to VM types offered by major CPs, we consider several types of VMs with various resource demands and prices [1] [60, 61, 71]. All the parameters used in the simulation are listed in Table 5.2 [51, 128, 20].

**Impact of federation:** First, we conducted some experiments to understand the benefits of VM migration in a federation. We evaluated the proposed algorithm using the system setup shown in Figure. 5.3, with two different scenarios; federated (**CAVMA-Fed**) and non-federated (**CAVMA-NonFed**). We compared the results with those for the initial state without migration (**NoMigration**). We considered $CP_1$ (Google) to execute the migration algorithm, where the VMs (ranging between 500 and 2500) were initially distributed randomly across its DCs.

Figure. 5.4 shows the results demonstrating the benefit of federation in terms of the cost to the CP and on the performance with respect to the migration parameters. In a non-federated case, $CP_1$ can migrate its VMs across only its DCs, while in a federated case $CP_1$ has more choices to select a destination DC for migration. This naturally lowers the TCO of

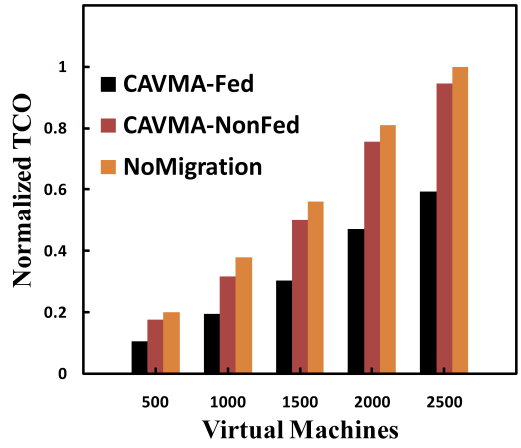
[1] VM price is proportional to the resource demand and we show in Figure.5.3 the price of a standard VM with 2vCPU, 8GB RAM, and 10GB Storage.



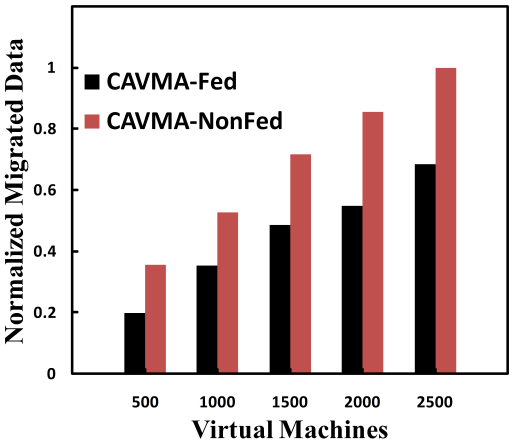

(a) Total VM Cost



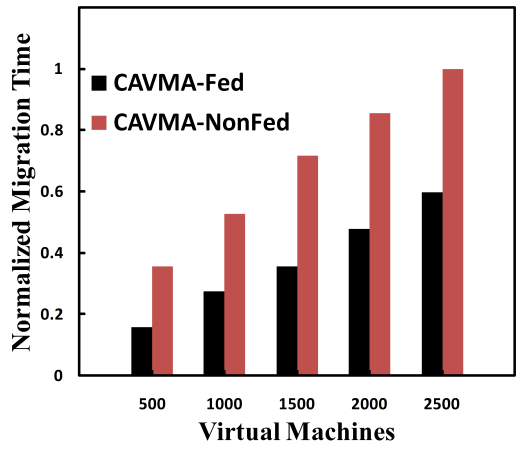

(b) Total Migration Cost



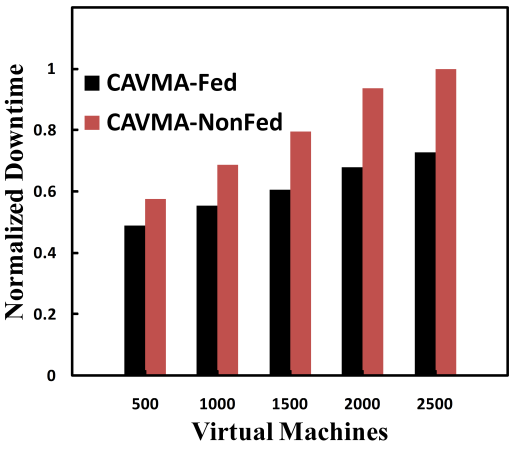

(c) TCO




(d) Total Migrated Data




(e) Total Migration Time

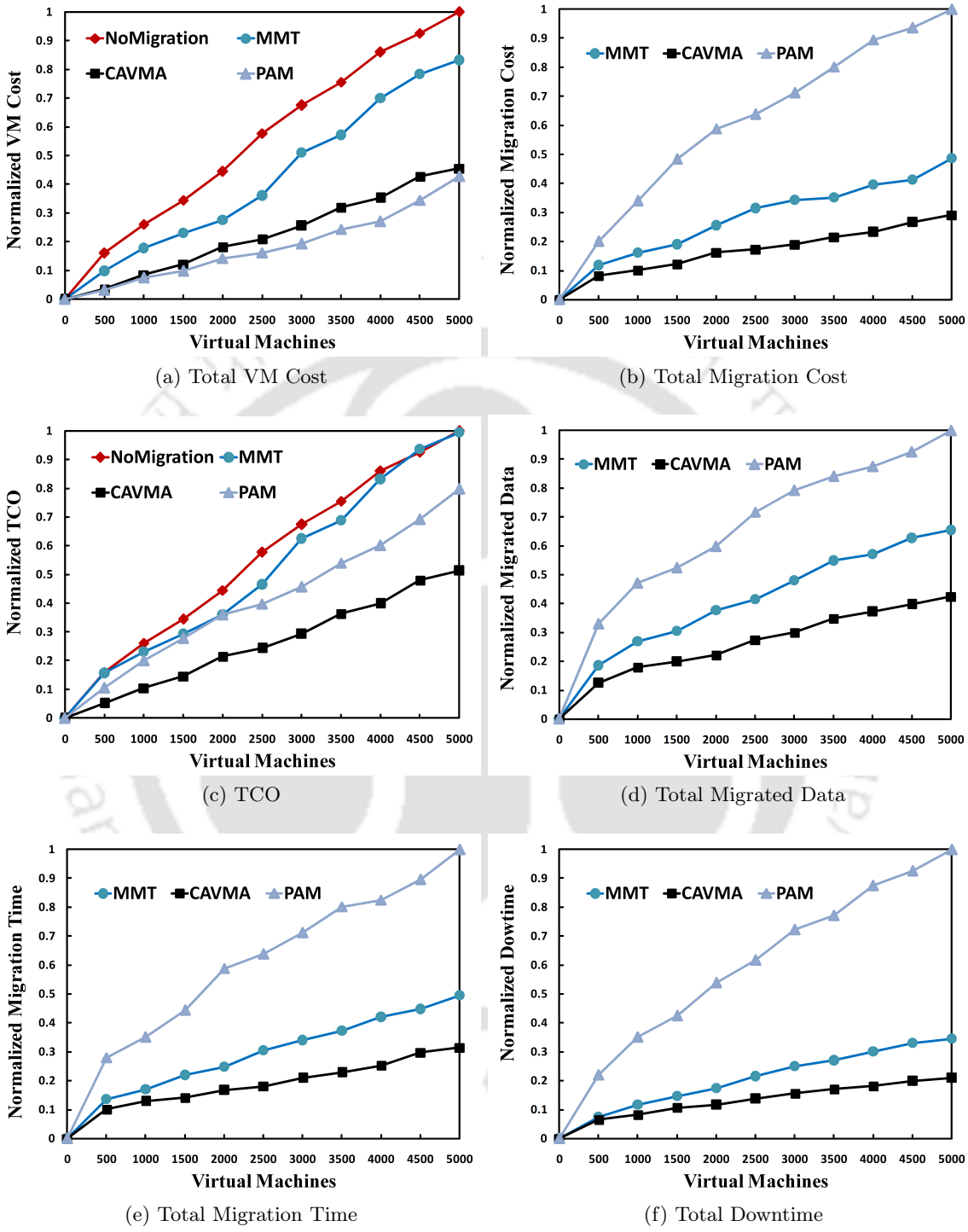



(f) Total Downtime

Figure 5.4: Impact of federation on the cost and the performance of migration-related metrics

the VMs hosted by $CP_1$ (using Eq. 5.10). The reduction in the VM cost reaches up to 45% in federated case compared to less than 20% in a non-federated case, as inferred from Figure. 5.4a. In a federated case, $CP_1$ can migrates its VMs to a DC within a region with low data transfer price to minimize the migration cost (using Eq. 5.11). On the other hand, in a non-federated case, $CP_1$ may be forced to migrate its VMs across its own regions, albeit with a high data transfer price. That leads to higher migration cost as observed in Figure. 5.4b. Consequently, we observe from Figure. 5.4c that CAVMA reduces the TCO (calculated by Eq. 5.12 as the sum of VM cost and migration cost) significantly. $CP_1$ can achieve a profit of about 40% in federated system compared to less than 10% in non-federated scenario. Again, as seen from Figure. 5.4d, CAVMA also transfers less data in a federated environment as it could select a destination DC with larger available bandwidth for migration. This decreases the data transferred in each iteration of the pre-copy stage of migration and the data transferred in the last iteration, i.e., stop-and-copy stage (see Eq. 5.5). This directly affects the migration performance and we observe that CAVMA-Fed gives lower migration time and downtime as seen from Figure. 5.4e and Figure. 5.4f, respectively.

**Comparison with baseline approaches:** Next, we compare the performance of the the proposed algorithm with two baseline approaches; **MMT** (**M**inimum **M**igration **T**ime) [142] and **PAM** (**P**rice **A**ware **M**igration) [45], apart from the initial case without migration (**NoMigration**). MMT sorts VMs in increasing order of the VM size and then it migrates them sequentially to a destination DC, connected with a larger available bandwidth to minimize the migration time. PAM sorts the VMs in decreasing order based on the CPU usage and then it migrates the VMs to a DC with the least pricing. All the algorithms for comparison have also been implemented in CloudSim using the same federation setup in Figure. 5.3. In this experiment, we varied the number of VMs between 500 to 5000.
Figure. 5.5, presents the results demonstrating the efficncy of the proposed algorithm compared to baseline algorithms. Figure. 5.5a shows the total cost of VMs, from where we observe that the proposed algorithm (CAVMA) reduces the total VM cost significantly, giving a cost very close to that with PAM, which aims to minimize the operating cost of VMs. CAVMA considers not only the least price while selecting the destination DC, but also considers the link bandwidth and the data transfer price between the DCs, thereby reducing the migration time and cost. MMT recorded higher VM cost as it does not consider the




(a) Total VM Cost

(b) Total Migration Cost

(c) TCO

(d) Total Migrated Data

(e) Total Migration Time

(f) Total Downtime

Figure 5.5: Comparison of proposed migration algorithm and baseline approaches in terms of cost and migration performance

price at all while selecting the destination DC. Figure. 5.5b shows a comparison of the total cost of migration. Although PAM achieves minimum VM cost, it gives the highest migration cost as it only considers minimizing the operating cost of VMs, while ignoring the migration cost and time. CAVMA gives the minimum migration cost, because MMT considers only the link bandwidth while selecting the destination DC, whereas CAVMA considers both the bandwidth cost and the data transfer price. It also includes migration cost while calculating the migration benefit (see Eq. 5.23). Consequently, we see from Figure. 5.5c that CAVMA gives the minimum TCO (up to 50% of the initial cost without migration) while MMT gives a TCO very near to the initial cost, because it tries to reduce only the migration time and ignores the cost.

We also compared the performance of the migration in terms of the other metrics. Figure. 5.5d shows that CAVMA leads to minimum data transfer, because it selects the VM with smaller size and lower PDR for migration. MMT only consider the VM size, but ignores PDR which causes a significant increase in the data transferred. However, PAM neither consider the VM size nor the PDR while selecting the migrated VM. This could lead to migration of VMs with very large memory, storage requirement, and high PDR resulting in more migration data. Accordingly, CAVMA achieves minimum total migration time and minimum total downtime as can be seen from Figure. 5.5e and Figure. 5.5f, respectively. We also note that the down time obtained by MMT is very close to that with CAVMA, because both the algorithms consider link bandwidth availability for migration. This helps to transfer most of the data from a migrated VM during pre-copy stage and lesser data transfer in stop-and-copy stage.

**Number of migrations:** We also compare the number of VMs migrated with the proposed algorithm and with the other baseline methods. That is another critical metric in migration that needs to be minimized. Figure. 5.6 shows that CAVMA reduces VM migrations up to 67% as it considers multiple factors while taking the decision to migrate a VM. It selects for migration VMs with high CPU capacity and small data size to migrate fewer VMs whose migration returns the highest profit. It migrates VMs with high CPU capacity and low It also triggers migration only if the migration is profitable considering migration cost. While, MMT performs the highest number of migrations as it selects VMs based on only the size, causing the migration of a large number of smaller-sized VMs. Similarly, PAM considers

only the CPU demand of a VM. Contrarily, CAVMA considers the trade-off between higher CPU and lower data size, apart from the PDR. Both MMT and PAM ignore the migration cost while taking the migration decision.

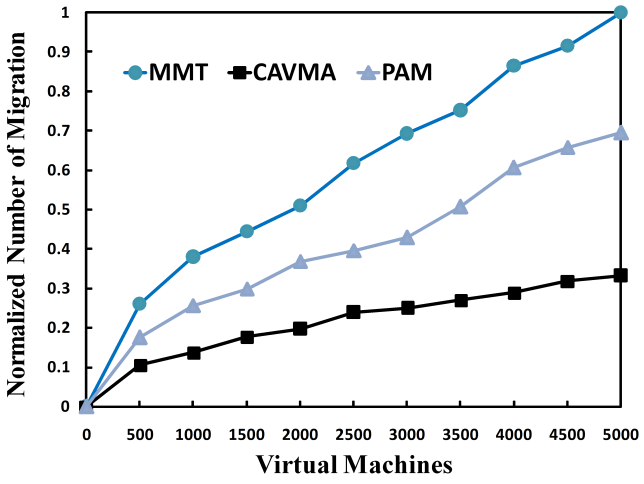
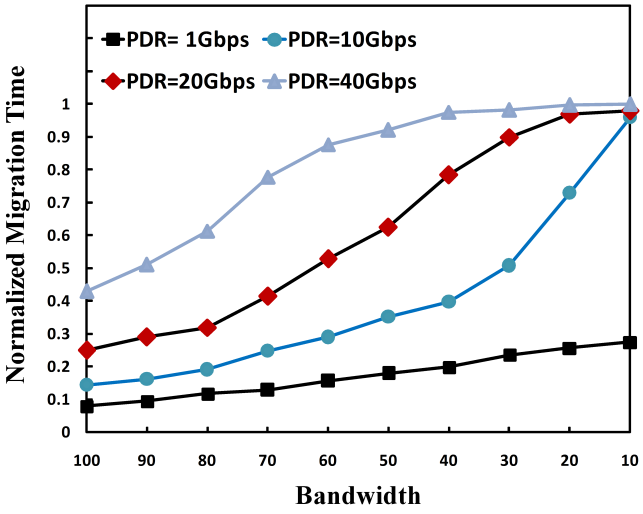
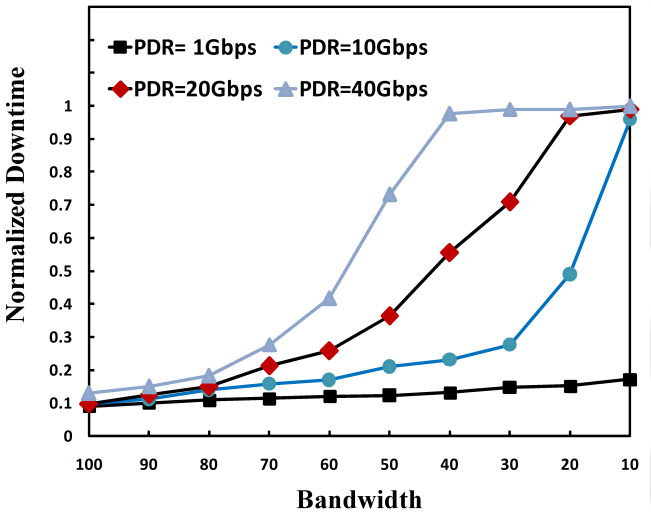



Figure 5.6: Comparison of the algorithms in terms of number of migrations

**Impact of bandwidth and PDR on migration performance:** Finally, we studied the impact of the available link bandwidth for migration and the PDR of the migrated VM on the performance. Note that migration time and downtime are inversely proportional to the bandwidth (see Eq. 5.8 and Eq. 5.9) and that PDR directly affects the size of data transferred in each iteration of the pre-copy stage (see Eq. 5.5). Larger PDR increases the data transfer in each iteration, increases the number of iterations, and increases downtime (causes more data to be sent in the stop-and-copy stage). Figure. 5.7 shows the impact of varying the bandwidth on the migration time and downtime for different PDR values. We run the migration algorithm with PDR values set to {1,10,20, and 40} Gbps (all the VMs had the same PDR). We also varied the migration bandwidth between 10 Gbps and 100 Gbps for each PDR value. While the PDR is below the bandwidth, the algorithm can transfer all the dirty data in a timely fashion, resulting in low migration time and downtime. When the PDR is closer to the bandwidth, the performance degrades, and the migration time and the downtime increase significantly. Figure. 5.7a shows that the migration time increases with decreasing bandwidth, as expected. It also shows higher migration time for VMs with larger PDR, as more data is transferred in each iteration of the pre-copy stage. The migration algorithm also needs more iterations, trying to reduce the migrated data in




(a) Migration Time




(b) Downtime

Figure 5.7: Impact of bandwidth and PDR on migration performance

the last iteration (i.e., stop-and-copy stage) to reduce downtime. When the bandwidth is near the PDR value, the migration time increases to the highest level, as the migration stops the pre-copy stage and moves to the stop-and-copy stage when the data transferred is three times the VM size (set as the termination condition).

Figure. 5.7b shows the downtime which remains almost constant when the PDR is small compared to the bandwidth. This is because the migration algorithm is able to transfer most of dirty data during the iterations of pre-copy stage (due to sufficient bandwidth), leading to very short stop-and-copy stage. When the bandwidth decreases closer to the

PDR value, the algorithm is forced to move to stop-and-copy stage with a large amount of dirty data yet to be transferred and the downtime increases accordingly. Therefore, to achieve better migration performance, the CP needs to ensure that enough bandwidth is available for migration and also needs to consider the PDR of migrated VMs.

## 5.4 Summary

In this chapter, we studied inter-DC live migration in a federated cloud with an objective of minimizing the TCO of the CP. We presented a simple model for VM cost in federated cloud considering various parameters that affect the migration, such as network bandwidth, data transfer price, VM size, and the PDR. We proposed a model to estimate the migration overhead in terms of cost and time, based on which we formulated the problem of cost-aware inter-DC VM migration as an optimization problem. We used an objective of minimizing the TCO of the CP, including the cost of migration and the migration time. To solve the NP-hard problem, we introduced three heuristics to decide the source and the destination DCs for migration, and the VM to migrate. We proposed an algorithm for VM migration across DCs of the federation to improve the migration benefit. The proposed algorithm has been implemented in CloudSim and evaluated with different scenarios. Results demonstrate the benefits of live migration in a federated cloud and show that the proposed algorithm outperforms other baseline approaches in terms of TCO, migration cost and time, and migration downtime. It also minimizes the number of migrations. We also studied the impact of the PDR and the bandwidth on the migration time and downtime.

CHAPTER

# 6

# Conclusions and Future Directions

The federated cloud paradigm is likely to be the future of the cloud, since it allows multiple cloud providers to collaborate and share their resources into a single platform. Despite the advantages offered by a federated cloud, one of the major concerns of a provider in the federation is how to minimize their own TCO while managing the resource allocation across the data centers. This work has made a few major research contributions to the area of cost-aware resource management in a federated cloud. Our objective was to help the cloud provider in minimizing its operating cost by considering the features of various applications while allocating the resources.

First, we designed a hierarchical framework for resource management and workload allocation in a federated cloud. It is useful in managing the resources at data center level and cloud provider level. It also meets the requirements of a federation with minimal communication overhead. Using the framework, we addressed cost-aware resource management for data-intensive applications, that require enormous resources and involve bulk data transfer across the DCs. The problem was formulated as an optimization model, with the main objective to minimize the total cost of operation (TCO) of the cloud provider, including the energy consumption cost and inter-data center communication cost. We proposed a two-phase algorithm for solving the problem. The first phase partitions the application requests into clusters of highly correlated virtual components to reduce the communication cost. The second phase allocates these clusters to data centers of the federation, leveraging the variation in electricity price and PUE to minimize the cost of energy consumption.

The proposed algorithm runs in polynomial-time with a negligible cost compared to the saving it could achieve while placing a data-intensive application. Using real-world data, we evaluated the proposed algorithm (CAVCA) across different scenarios. CAVCA was seen to perform better than other baseline approaches in terms of energy, communication, and total costs. It could reduce the total operating cost by 60% compared to other baseline approaches. Even for an application with 10000 virtual components, CAVCA did not need more than three minutes of running time.

Next, we addressed the resource management problem for hosting vehicular applications in federated cloud. We presented the FVCC architecture, which introduced the concept of the federation in VCC. We introduced a dynamic pricing model for resource sharing among various cloud providers in a federated cloud. We formulated an optimization problem, wherein the main objective was to minimize the service provider's cost while meeting the delay requirements of various applications. We proposed an algorithm to solve the problem also utilizing VM migration across data centers to handle user mobility and to reduce the delay of user requests. The proposed algorithm (CDRAM) was evaluated with federated and non-federated scenarios; to show the benefits of the federated cloud in satisfying the requirements of the service provider and end users, alike. We compared the proposed algorithm with other baseline approaches in terms of delay, cost, number of served requests, and number of migrations. The proposed algorithm could serve more than 90% of requests with a lower cost and fewer VM migrations.

Finally, we addressed the problem of cost-aware inter-data center VM migration to maximize the profit of the cloud provider by minimizing its operating cost while considering the cost of migration. We modeled inter-data center VM migration, including migration cost, migration time and downtime. We presented an optimization model with an objective to minimize the TCO of the cloud provider (considering VM cost and migration cost) and the migration time. We analysed the various parameters that affect the migration performance and formulated three heuristics for selecting the source data center, the destination data center, and the source VM for migration. Accordingly, we proposed an algorithm for inter-data center VM migration (CAVMA) to maximize the profit of migration. Simulation results demonstrated the benefits of utilizing VM migration in a federated cloud. The proposed algorithm is shown to reduce the TCO by 40% in the federated system compared

to less than 10% in the non-federated scenario, also with lower migration time and cost. The proposed algorithm was shown to outperform other baseline methods in terms of TCO, the size of migrated data, migration time, migration cost, migration downtime, and the number of migrations. We observed that CAVMA reduced the VM migrations by up to 67%.

**Conclusions:** From the work in this thesis, we conclude that exploiting federation for hosting cloud applications brings many benefits for the cloud provider and end users. For instance, leveraging the resource price variation among data centers of a federated system reduces the cost of the cloud provider significantly. We demonstrated that it is not effective for the cloud provider to handle all types of applications in the same way. For data-intensive applications, we argued that the cost of energy consumption and communication have to be minimized together to minimize the TCO. We also observed that CAVCA is more useful when the DCs do not operate at their peak utilization, which often is the case. We concluded that CAVCA algorithm achieved better cost reduction when the cost factors vary significantly across the CPs in a federation and the correlated virtual components are scheduled together. We conclude that the federation system gives the balanced solution for cost and global coverage at the same time.

With respect to hosting vehicular applications, we found that a federation gives more choices in finding a data center that meets the delay requirements at a lower cost. We showed that FVCC is a promising platform for hosting vehicular applications. The proposed algorithm CDRAM demonstrated the advantages of jointly optimizing the cost and the delay for vehicular applications. We also showed that CDRAM handles vehicle mobility and improves the QoS by monitoring the performance and migrating the VM to a data center that services with a lower delay. Our results showed that CDRAM considered the trade-off between the cost and the delay to support vehicular applications effectively in a federated cloud.

Our work utilized live migration in a federated cloud for maximizing the profit of the cloud provider. We observed that the migration cost affects the TCO significantly and that the inter-data center migration degrades the performance, unless the migration time is lowered. We conclude that examining the impact of migration in terms of cost and time is a critical issue for the cloud provider. We found that inter-data center migration is not a good

choice for data-intensive applications. We concluded that considering the metrics related to migration improves the efficiency of the migration algorithm significantly. We came to the conclusion that the selection of a VM for migration must consider VM type, residual time, PDR, and eligibility for migration. From the simulation results, we concluded that to achieve better migration performance, the cloud provider must consider the bandwidth for migration and the PDR of migrated VMs.

**Future Directions:** There are several possible directions in which the work in this thesis can be extended. We list a few directions in what follows:

- In the problem of cost-aware placement of data-intensive applications, we focused only on static allocation of virtual components without considering migration, as online data migration requires long time. However, data is typically replicated across two (or more) data centers for high availability and fault tolerance [143, 144]. This problem can be extended to include live migration of only compute VMs with pre-planned data replication. Usually the price of electricity varies over time. In forward pricing approach, the cloud provider knows the price in advance. Accordingly, to utilize the spatio-temporal variation in electricity price, the cloud provider can replicate the data blocks of a cluster during (or after) the initial placement to a data center where electricity price will be cheaper after some time to migrate the cluster of VMs.

- It is interesting to exploit the concept of "MultiCloud Tournament" [145] for resource management in federated cloud using reputation-driven approach where cloud providers are ranked based on their performance. This concept can be used for selection of the best data center for the resource allocation, by assigning a score for data centers according to the rank of the cloud provider (the performance) and the resource price. Other parameters such as the amount of available resources, processing delay, and the location or the network latency, could be included in the score computation.

- The emergence of fog/edge computing reduces the network latency and the response time. However, still large applications need to be offloaded to the cloud. FVCC architecture can be extended further to include fog/edge computing to take advantages of various recent computing paradigms. The resource management problem needs to be handled at multiple levels with each level having different ranges of network latency and

resource prices. The resource management algorithm could select the appropriate level and the most efficient resources within the level based on the application requirements and the cost.

- In the problem of cost-and-delay aware resource allocation for vehicular applications, we assumed the VMs to be independent. Many vehicular applications require VMs of multiple vehicles to communicate and exchange data. When these applications are hosted in the cloud, minimizing the communication delay becomes a major objective for the cloud provider and the user. Accordingly, the problem addressed in this thesis can be extended to cost-and-communication aware resource management for vehicular services. The problem would be non-trivial since we have to consider the delay between VMs and the delay to the end user, which may conflict while minimizing the communication cost.
- We could design a penalty-based model for performance degradation due to migration and reconsider the problem by adding this penalty to the TCO. Also, it could be an interesting challenge to consider the communication between VMs during migration, as migrating a VM out of a group of correlated VMs may increase inter-DC traffic resulting in large delay and communication cost. We could also address the problem of interactive service migration in federated cloud to minimize the TCO while considering the access delay to the end-user.
- Security and privacy of the federated cloud system and securing the communication in the proposed resource management framework are other important issues. Blockchain technology can be utilized to protect the communication from attackers and to implement a trust relationship among cloud providers of the federation. This increases the reliability of a federated system.

# Bibliography


[1] T. Wenhong and Z. Yong, *Optimized Cloud Resource Management and Scheduling*. Boston: Morgan Kaufmann, 2015.

[2] Cisco, *Cisco Global Cloud Index, Forecast and Methodology, 2020-2025, white paper*. Cisco System, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[3] Cloud Services Market Size, Share & Growth | Forecast - 2030. Accessed: 2021-07-11. [Online]. Available: https://www.alliedmarketresearch.com/cloud-services-market

[4] M. J. F. Rosa, C. G. Ralha, M. Holanda, and A. P. F. Araujo, "Computational Resource and Cost Prediction Service for Scientific Workflows in Federated Clouds," *Future Generation Computer Systems*, vol. 125, pp. 844–858, Dec. 2021.

[5] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A Trust-Aware Mechanism for Cloud Federation Formation," *IEEE Transactions on Cloud Computing*, 2019. [Online]. Available: https://doi.org/10.1109/TCC.2019.2911831

[6] C. A. Lee, R. B. Bohn, and M. Michel, "The NIST Cloud Federation Reference Architecture," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 500-332, Feb. 2020.

[7] A. Ehsanfar and P. T. Grogan, "Auction-Based Algorithms for Routing and Task Scheduling in Federated Networks," *Journal of Network and Systems Management*, vol. 28, no. 2, pp. 271–297, Apr. 2020.

[8] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource Management in Cloud Networking Using Economic Analysis and Pricing Models: A Survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 954–1001, Secondquarter 2017.

[9] M. Liaqat, V. Chang, A. Gani, S. H. A. Hamid, M. Toseef, U. Shoaib, and R. L. Ali, "Federated cloud resource management: Review and discussion," *Journal of Network and Computer Applications*, vol. 77, no. Supplement C, pp. 87 – 105, 2017.

[10] H. Deng, L. Huang, H. Xu, X. Liu, P. Wang, and X. Fang, "Revenue Maximization for Dynamic Expansion of Geo-Distributed Cloud Data Centers," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 899–913, Jul. 2020.

[11] B. K. Ray, A. Saha, S. Khatua, and S. Roy, "Toward maximization of profit and quality of cloud federation: Solution to cloud federation formation problem," *The Journal of Supercomputing*, vol. 75, no. 2, pp. 885–929, Feb. 2019.

[12] M. Hadji and D. Zeghlache, "Mathematical Programming Approach for Revenue Maximization in Cloud Federations," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 99–111, Jan 2017.

[13] A. I. Middya, B. Ray, and S. Roy, "Auction based Resource Allocation Mechanism in Federated Cloud Environment: TARA," *IEEE Transactions on Services Computing*, 2019. [Online]. Available: https://doi.org/10.1109/TSC.2019.2952772

[14] J. Xu and B. Palanisamy, "Optimized Contract-Based Model for Resource Allocation in Federated Geo-Distributed Clouds," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 530–543, Mar. 2021.

[15] O. Ayoub, A. De Sousa, S. Mendieta, F. Musumeci, and M. Tornatore, "Online Virtual Machine Evacuation for Disaster Resilience in Inter-Data Center Networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1990–2001, Jun. 2021.

[16] N. Khorasani, S. Abrishami, M. Feizi, M. A. Esfahani, and F. Ramezani, "Resource Management in the Federated Cloud Environment Using Cournot and Bertrand Competitions," *Future Generation Computer Systems*, vol. 113, pp. 391–406, Dec. 2020.

[17] N. Tziritas, S. U. Khan, T. Loukopoulos, S. Lalis, C.-Z. Xu, K. Li, and A. Y. Zomaya, "Online Inter-Datacenter Service Migrations," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1054–1068, Oct. 2020.

[18] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-Me Cloud: When Cloud Services Follow Mobile Users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2019.

[19] G. Darzanos, I. Koutsopoulos, and G. D. Stamoulis, "Cloud Federations: Economics, Games and Benefits," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2111–2124, Oct. 2019.

[20] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 1206–1243, Secondquarter 2018.

[21] M. Koot and F. Wijnhoven, "Usage Impact on Data Center Electricity Needs: A System Dynamic Forecasting Model," *Applied Energy*, vol. 291, p. 116798, Jun. 2021.

[22] S. Albahli, M. Shiraz, and N. Ayub, "Electricity Price Forecasting for Cloud Computing Using an Enhanced Machine Learning Model," *IEEE Access*, vol. 8, pp. 200 971–200 981, 2020.

[23] G. Venkatesh and K. Arunesh, "Map Reduce for big data processing based on traffic aware partition and aggregation," *Cluster Computing*, vol. 22, no. 5, pp. 12 909–12 915, Sep. 2019.

[24] F. Ebadifard and S. M. Babamir, "Federated Geo-Distributed Clouds: Optimizing Resource Allocation Based on Request Type Using Autonomous and Multi-objective Resource Sharing Model," *Big Data Research*, vol. 24, p. 100188, May 2021.

[25] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, and R. Montanari, "Big spatial data management for the internet of things: A survey." *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 990 – 1035, 2020.

[26] M. Hajeer and D. Dasgupta, "Handling Big Data Using a Data-Aware HDFS and Evolutionary Clustering Technique," *IEEE Transactions on Big Data*, vol. 5, no. 2, pp. 134–147, Jun. 2019.

[27] A. Mseddi, M. A. Salahuddin, M. F. Zhani, H. Elbiaze, and R. H. Glitho, "Efficient Replica Migration Scheme for Distributed Cloud Storage Systems," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 155–167, Jan. 2021.

[28] H. Ma, W. Tang, H. Zhu, and H. Zhang, "Resource Utilization-Aware Collaborative Optimization of IaaS Cloud Service Composition for Data-Intensive Applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 1322–1333, Feb. 2021.

[29] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "Big Data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98 – 115, 2015.

[30] P. Eugster, C. Jayalath, K. Kogan, and J. Stephen, "Big Data Analytics beyond the Single Datacenter," *Computer*, vol. 50, no. 6, pp. 60–68, 2017.

[31] A. Rezgui, N. Davis, Z. Malik, B. Medjahed, and H. S. Soliman, "CloudFinder: A System for Processing Big Data Workloads on Volunteered Federated Clouds," *IEEE Transactions on Big Data*, vol. 6, no. 2, pp. 347–358, Jun. 2020.

[32] A. Boukerche and R. E. De Grande, "Vehicular Cloud Computing: Architectures, Applications, and Mobility," *Computer Networks*, vol. 135, pp. 171–189, Apr. 2018.

[33] M. Shojafar, N. Cordeschi, and E. Baccarelli, "Energy-Efficient Adaptive Resource Management for Real-Time Vehicular Cloud Services," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 196–209, Jan. 2019.

[34] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai, "Allocation of Computation-Intensive Graph Jobs Over Vehicular Clouds in IoV," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 311–324, Jan. 2020.

[35] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, "Internet of Vehicles: Architecture, Protocols, and Security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, Oct. 2018.

[36] W. M. Danquah and D. T. Altilar, "Vehicular Cloud Resource Management, Issues and Challenges: A Survey," *IEEE Access*, vol. 8, pp. 180 587–180 607, 2020.

[37] V. Kherbache, É. Madelaine, and F. Hermenier, "Scheduling Live Migration of Virtual Machines," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 282–296, Jan. 2020.

[38] Y. B. Udupi, X. Huang, A. D. Gupta, M. S. Tarre, and R. Jain, "Cloud Resource Placement Optimization and Migration Execution in Federated Clouds," US Patent US10 659 387B2, May, 2020.

[39] H. Shen and L. Chen, "A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 17–31, Jan. 2020.

[40] M. A. Khan, "An Efficient Energy-Aware Approach for Dynamic VM Consolidation on Cloud Platforms," *Cluster Computing*, Jun. 2021. [Online]. Available: https://doi.org/10.1007/s10586-021-03341-0

[41] M. Tarahomi, M. Izadi, and M. Ghobaei-Arani, "An efficient power-aware VM allocation mechanism in cloud data centers: A micro genetic-based approach," *Cluster Computing*, vol. 24, no. 2, pp. 919–934, Jun. 2021.

[42] C. Liu, K. Li, and K. Li, "A Game Approach to Multi-Servers Load Balancing with Load-Dependent Server Availability Consideration," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 1–13, Jan. 2021.

[43] L. Yang, D. Yang, J. Cao, Y. Sahni, and X. Xu, "QoS Guaranteed Resource Allocation for Live Virtual Machine Migration in Edge Clouds," *IEEE Access*, vol. 8, pp. 78 441–78 451, 2020.

[44] P.-J. Maenhaut, B. Volckaert, V. Ongenae, and F. De Turck, "Resource Management in a Containerized Cloud: Status and Challenges," *Journal of Network and Systems Management*, vol. 28, no. 2, pp. 197–246, Apr. 2020.

[45] H. Ali, M. Zakarya, I. U. Rahman, A. A. Khan, and R. Buyya, "FollowMe@LS: Electricity Price and Source Aware Resource Management in Geographically Distributed Heterogeneous Datacenters," *Journal of Systems and Software*, vol. 175, p. 110907, May 2021.

[46] M. Najm, M. Patra, and V. Tamarapalli, "Cost-and-Delay Aware Dynamic Resource Allocation in Federated Vehicular Clouds," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6159–6171, Jun. 2021.

[47] M. Najm and V. Tamarapalli, "VM Migration for Profit Maximization in Federated Cloud Data Centers," in *Proceedings of 2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, Jan. 2020, pp. 882–884.

[48] A. Forestiero, C. Mastroianni, M. Meo, G. Papuzzo, and M. Sheikhalishahi, "Hierarchical Approach for Efficient Workload Management in Geo-Distributed Data Centers," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 97–111, March 2017.

[49] "Electricity pricing," https://en.wikipedia.org/wiki/Electricity_pricing, accessed: 4 April 2021.

[50] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[51] Google Cloud. Google Cloud Networking Pricing. Accessed: 2021-07-11. [Online]. Available: https://cloud.google.com/vpc/network-pricing

[52] P. Li, S. Guo, S. Yu, and W. Zhuang, "Cross-Cloud MapReduce for Big Data," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 375–386, Apr. 2020.

[53] K. Deng, K. Ren, M. Zhu, and J. Song, "A Data and Task Co-Scheduling Algorithm for Scientific Cloud Workflows," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 349–362, Apr. 2020.

[54] S. Olariu, "A Survey of Vehicular Cloud Research: Trends, Applications and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2648–2663, Jun. 2020.

[55] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 38–67, Firstquarter 2020.

[56] H. Peng, L. Liang, X. Shen, and G. Y. Li, "Vehicular Communications: A Network Layer Perspective," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1064–1078, Feb. 2019.

[57] M. Najm, R. Tripathi, M. S. Alhakeem, and V. Tamarapalli, "A Cost-Aware Management Framework for Placement of Data-Intensive Applications on Federated Cloud," *Journal of Network and Systems Management*, vol. 29, no. 3, p. 25, Mar. 2021.

[58] M. Giacobbe, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "An approach to reduce energy costs through virtual machine migrations in cloud federation," in *Proceedings of 2015 IEEE Symposium on Computers and Communication (ISCC)*, July 2015, pp. 782–787.

[59] T. Le, "A Survey of Live Virtual Machine Migration Techniques," *Computer Science Review*, vol. 38, p. 100304, Nov. 2020.

[60] Google Cloud VM instances pricing. Accessed: 2021-07-11. [Online]. Available: https://cloud.google.com/compute/vm-instance-pricing

[61] Microsoft Azure Virtual Machines Pricing. Accessed: 2021-07-11. [Online]. Available: https://azure.microsoft.com/en-in/pricing/details/virtual-machines/linux/

[62] D. G. Lago, R. A. C. da Silva, E. R. M. Madeira, N. L. S. da Fonseca, and D. Medhi, "SinergyCloud: A Simulator for Evaluation of Energy Consumption in Data Centers and Hybrid Clouds," *Simulation Modelling Practice and Theory*, vol. 110, p. 102329, Jul. 2021.

[63] S. Singh, I. Chana, and R. Buyya, "STAR: SLA-aware Autonomic Management of Cloud Resources," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1040–1053, Oct. 2020.

[64] Hyperscale Data Centers Worldwide 2015-2020. Accessed: 2021-07-11. [Online]. Available: https://www.statista.com/statistics/633826/worldwide-hyperscale-data-center-numbers/

[65] AWS Global Infrastructure. Accessed : 2 April 2021. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure

[66] H.-W. Deng, M. Rahman, M. Chowdhury, M. S. Salek, and M. Shue, "Commercial Cloud Computing for Connected Vehicle Applications in Transportation Cyberphysical Systems: A Case Study," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 6–19, 2021.

[67] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Network*, vol. 27, no. 5, pp. 48–55, Sep. 2013.

[68] V. K. Yadav, S. Verma, and S. Venkatesan, "Efficient and Secure Location-Based Services Scheme in VANET," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 567–13 578, Nov. 2020.

[69] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[70] N. Jain and I. Menache, "Resource Management for Cloud Computing Platforms," US Patent US10 644 966B2, May, 2020.

[71] Amazon EC2 On-Demand Pricing. Accessed: 2021-07-11. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

[72] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil, "A critical survey of live virtual machine migration techniques," *Journal of Cloud Computing*, vol. 6, no. 1, p. 23, Nov. 2017.

[73] W. Chen, I. Paik, and Z. Li, "Cost-Aware Streaming Workflow Allocation on Geo-Distributed Data Centers," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 256–271, Feb 2017.

[74] Z. A. Mann, "Allocation of Virtual Machines in Cloud Data Centers: A Survey of Problem Models and Optimization Algorithms," *ACM Computing Surveys*, vol. 48, no. 1, pp. 11:1–11:34, Aug. 2015.

[75] S. Azizi, M. Zandsalimi, and D. Li, "An Energy-Efficient Algorithm for Virtual Machine Placement Optimization in Cloud Data Centers," *Cluster Computing*, vol. 23, no. 4, pp. 3421–3434, Dec. 2020.

[76] L. Gu, D. Zeng, P. Li, and S. Guo, "Cost Minimization for Big Data Processing in Geo-Distributed Data Centers," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 314–323, Sept 2014.

[77] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Collaborative Agents for Distributed Load Management in Cloud Data Centers Using Live Migration of Virtual Machines," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 916–929, Nov 2015.

[78] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation Offloading and Resource Allocation in Vehicular Networks Based on Dual-Side Cost Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019.

[79] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, "A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: Taxonomy and challenges," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 3, pp. 267–286, Mar. 2020.

[80] X. Qie, S. Jin, and W. Yue, "An Energy-Efficient Strategy for Virtual Machine Allocation over Cloud Data Centers," *Journal of Network and Systems Management*, vol. 27, no. 4, pp. 860–882, Oct. 2019.

[81] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software-Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[82] D. S. Dias and L. H. M. K. Costa, "Online traffic-aware virtual machine placement in data center networks," in *Proceedings of 2012 Global Information Infrastructure and Networking Symposium (GIIS)*, Dec 2012, pp. 1–8.

[83] M. Guzek, D. Kliazovich, and P. Bouvry, "HEROS: Energy-Efficient Load Balancing for Heterogeneous Data Centers," in *Proceedings of 2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 742–749.

[84] D. G. Lago, E. R. M. Madeira, and D. Medhi, "Energy-Aware Virtual Machine Scheduling on Data Centers with Heterogeneous Bandwidths," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 1, pp. 83–98, Jan 2018.

[85] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative Cloud and Edge Computing for Latency Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[86] B. Ahmad, Z. Maroof, S. McClean, D. Charles, and G. Parr, "Economic Impact of Energy Saving Techniques in Cloud Server," *Cluster Computing*, vol. 23, no. 2, pp. 611–621, Jun. 2020.

[87] S. Ismaeel, R. Karim, and A. Miri, "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres," *Journal of Cloud Computing*, vol. 7, no. 1, p. 10, Jun. 2018.

[88] A. Tripathi, I. Pathak, and D. P. Vidyarthi, "Modified Dragonfly Algorithm for Optimal Virtual Machine Placement in Cloud Computing," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 1316–1342, Oct. 2020.

[89] S. Ilager, K. Ramamohanarao, and R. Buyya, "ETAS: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 17, p. e5221, 2019.

[90] "Capacity Planner," https://www.vmware.com/products/capacity-planner.html, accessed : 21 July 2021.

[91] D. S. Marcon, M. C. Neves, R. R. Oliveira, L. R. Bays, R. Boutaba, L. P. Gaspary, and M. P. Barcellos, "IoNCloud: Exploring Application Affinity to Improve Utilization and Predictability in Datacenters," in *Proceedings of 2015 IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 5497–5503.

[92] M. H. Ferdaus, M. M. Murshed, R. N. Calheiros, and R. Buyya, "Multi-objective, Decentralized Dynamic Virtual Machine Consolidation using ACO Metaheuristic in Computing Clouds," *Computing Research Repository CoRR*, vol. abs/1706.06646, 2017.

[93] W. Wang, Y. Jiang, and W. Wu, "Multiagent-Based Resource Allocation for Energy Minimization in Cloud Computing Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 205–220, Feb 2017.

[94] M. Kumar, S. C. Sharma, S. Goel, S. K. Mishra, and A. Husain, "Autonomic Cloud Resource Provisioning and Scheduling Using Meta-Heuristic Algorithm," *Neural Computing and Applications*, vol. 32, no. 24, pp. 18 285–18 303, Dec. 2020.

[95] A. Khosravi, L. L. H. Andrew, and R. Buyya, "Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183–196, April 2017.

[96] M. A. Islam, A. Gandhi, and S. Ren, "Minimizing electricity cost for geo-distributed interactive services with tail latency constraint," in *Proceedings of 2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, Nov 2016, pp. 1–8.

[97] M. Najm and V. Tamarapalli, "Inter-Data Center Virtual Machine Migration in Federated Cloud," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, ser. ICDCN 2020. New York, NY, USA: Association for Computing Machinery, Jan. 2020, p. 1.

[98] E. Ahvar, S. Ahvar, N. Crespi, J. Garcia-Alfaro, and Z. A. Mann, "NACER: A Network-Aware Cost-Efficient Resource Allocation Method for Processing-Intensive Tasks in Distributed Clouds," in *Proceedings of 2015 IEEE 14th International Symposium on Network Computing and Applications*, Sept 2015, pp. 90–97.

[99] M. Keller and H. Karl, "Response Time-optimized Distributed Cloud Resource Allocation," in *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*, ser. DCC '14. New York, NY, USA: ACM, 2014, pp. 47–52.

[100] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang, "Traffic-Aware Geo-Distributed Big Data Analytics with Predictable Job Completion Time," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1785–1796, June 2017.

[101] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A Comprehensive Survey for Scheduling Techniques in Cloud Computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, Oct. 2019.

[102] S. Latif, S. M. M. Gilani, L. Ali, S. Iqbal, and M. Liaqat, "Characterizing the architectures and brokering protocols for enabling clouds interconnection," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 21, p. e5676, 2020.

[103] X. Zhang, K. Li, and Y. Zhang, "Optimising Data Access Latencies of Virtual Machine Placement Based on Greedy Algorithm in Datacentre," *International Journal of Computational Science and Engineering*, vol. 18, no. 2, pp. 186–194, Jan. 2019.

[104] M. H. Ferdaus, M. M. Murshed, R. N. Calheiros, and R. Buyya, "An Algorithm for Network and Data-aware Placement of Multi-Tier Applications in Cloud Data Centers," *Computing Research Repository (CoRR)*, vol. abs/1706.06035, 2017.

[105] T. Shabeera, S. M. Kumar, S. M. Salam, and K. M. Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 616 – 628, 2017.

[106] W. Xiao, W. Bao, X. Zhu, and L. Liu, "Cost-Aware Big Data Processing Across Geo-Distributed Datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3114–3127, Nov. 2017.

[107] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and Repurchasing for Big Data Processing in Multi-Clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 266–277, Apr. 2016.

[108] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular Edge Computing and Networking: A Survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, Jun. 2021.

[109] C.-C. Lin, D.-J. Deng, and C.-C. Yao, "Resource Allocation in Vehicular Cloud Computing Systems With Heterogeneous Vehicles and Roadside Units," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3692–3700, Oct. 2018.

[110] S. Midya, A. Roy, K. Majumder, and S. Phadikar, "Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach," *Journal of Network and Computer Applications*, vol. 103, pp. 58–84, Feb. 2018.

[111] R. I. Meneguette and A. Boukerche, "An Efficient Green-Aware Architecture for Virtual Machine Migration in Sustainable Vehicular Clouds," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 1, pp. 25–36, Jan. 2020.

[112] S. Garg, K. Kaur, S. H. Ahmed, A. Bradai, G. Kaddoum, and M. Atiquzzaman, "MobQoS: Mobility-Aware and QoS-Driven SDN Framework for Autonomous Vehicles," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 12–20, Aug. 2019.

[113] H. Zhang, Y. Xiao, S. Bu, F. R. Yu, D. Niyato, and Z. Han, "Distributed Resource Allocation for Data Center Networks: A Hierarchical Game Approach," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 778–789, Jul. 2020.

[114] Z. Ren, T. Lu, X. Wang, W. Guo, G. Liu, and S. Chang, "Resource scheduling for delay-sensitive application in three-layer fog-to-cloud architecture," *Peer-to-Peer Networking and Applications*, vol. 13, no. 5, pp. 1474–1485, Sep. 2020.

[115] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, 2019. [Online]. Available: https://doi.org/10.1109/TCC.2019.2903240

[116] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[117] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-Sensitive Task Offloading in the 802.11p-Based Vehicular Fog Computing Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773–785, Jan. 2020.

[118] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1786 – 1794, 2013.

[119] R. A. C. da Silva and N. L. S. da Fonseca, "Energy-Aware Migration of Groups of Virtual Machines in Distributed Data Centers," in *Proceedings of 2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[120] V. De Maio, R. Prodan, S. Benedict, and G. Kecskemeti, "Modelling energy consumption of network transfers and virtual machine migration," *Future Generation Computer Systems*, vol. 56, pp. 388–406, 2016.

[121] W. Cerroni, "Multiple virtual machine live migration in federated cloud systems," in *Proceedings of 2014 IEEE INFOCOM Workshop on Cross-Cloud Systems*. IEEE, 2014, pp. 25–30.

[122] Q. Wu, F. Ishikawa, Q. Zhu, and Y. Xia, "Energy and Migration Cost-Aware Dynamic Virtual Machine Consolidation in Heterogeneous Cloud Datacenters," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 550–563, Jul. 2019.

[123] M. Abu-Tair, M. I. Biswas, P. Morrow, S. McClean, B. Scotney, and G. Parr, "Quality of Service Scheme for Intra/Inter-Data Center Communications," in *Proceedings of 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, March 2017, pp. 850–856.

[124] A. Ehsanfar and P. T. Grogan, "Mechanism Design for Exchanging Resources in Federated Networks," *Journal of Network and Systems Management*, vol. 28, no. 1, pp. 108–132, Jan. 2020.

[125] "Virtual network pricing," https://azure.microsoft.com/en-in/pricing/details/virtual-network/, 2020, accessed : 25 July 2021.

[126] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, Jul. 2015.

[127] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A Survey of PSO-Based Scheduling Algorithms in Cloud Computing," *Journal of Network and Systems Management*, vol. 25, no. 1, pp. 122–158, Jan. 2017.

[128] A. Yassine, A. A. N. Shirehjini, and S. Shirmohammadi, "Bandwidth On-Demand for Multimedia Big Data Transfer Across Geo-Distributed Cloud Data Centers," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1189–1198, Oct. 2020.

[129] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and Profiling Scientific Workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, Mar. 2013.

[130] T. Baker, B. Aldawsari, M. Asim, H. Tawfik, Z. Maamar, and R. Buyya, "Cloud-SEnergy: A bin-packing based multi-cloud service broker for energy efficient composition and execution of data-intensive applications," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 242 – 252, 2018.

[131] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler, "More than bin packing: Dynamic resource allocation strategies in cloud data centers," *Information Systems*, vol. 52, pp. 83 – 95, 2015.

[132] M. Patra and C. S. R. Murthy, "Performance evaluation of joint placement and sleep scheduling of grid-connected solar powered road side units in vehicular networks," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 1197–1209, 2018.

[133] J. Wang, J. Liu, and N. Kato, "Networking and Communications in Autonomous Driving: A Survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1243–1274, Secondquarter 2019.

[134] Y. Wang, J. Zheng, and N. Mitton, "Delivery Delay Analysis for Roadside Unit Deployment in Vehicular Ad Hoc Networks With Intermittent Connectivity," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8591–8602, 2016.

[135] J. Heo, B. Kang, J. M. Yang, J. Paek, and S. Bahk, "Performance-Cost Tradeoff of Using Mobile Roadside Units for V2X Communication," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9049–9059, Sep. 2019.

[136] M. Najm and V. Tamarapalli, "Cost-Efficient Deployment of Big Data Applications in Federated Cloud Systems," in *Proceedings of 2019 11th International Conference on Communication Systems Networks (COMSNETS)*, Jan. 2019, pp. 428–431.

[137] Y. Wu and J. Zheng, "Modeling and Analysis of the Uplink Local Delay in MEC-Based VANETs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3538–3549, Apr. 2020.

[138] H. Liang *et al.*, "A Novel Adaptive Resource Allocation Model Based on SMDP and Reinforcement Learning Algorithm in Vehicular Cloud System," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 018–10 029, Oct. 2019.

[139] F. Lyu *et al.*, "MoMAC: Mobility-Aware and Collision-Avoidance MAC for Safety Applications in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 590–10 602, Nov. 2018.

[140] F. Yang, J. Han, X. Ding, Z. Wei, and X. Bi, "Spectral Efficiency Optimization and Interference Management for Multi-Hop D2D Communications in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6422–6436, Jun. 2020.

[141] W. Wu, W. Lin, and Z. Peng, "An Intelligent Power Consumption Model for Virtual Machines under CPU-Intensive Workload in Cloud Environment," *Soft Computing*, vol. 21, no. 19, pp. 5755–5764, Oct. 2017.

[142] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and Performance Analysis of Various VM Placement Strategies in CloudSim," *Journal of Cloud Computing*, vol. 4, no. 1, p. 20, Nov. 2015.

[143] M. Saadoon, S. H. Ab. Hamid, H. Sofian, H. H. M. Altarturi, Z. H. Azizul, and N. Nasuha, "Fault Tolerance in Big Data Storage and Processing Systems: A Review on Challenges and Solutions," *Ain Shams Engineering Journal*, Jul. 2021. [Online]. Available: https://doi.org/10.1016/j.asej.2021.06.024

[144] R. Tripathi, S. Vignesh, V. Tamarapalli, and D. Medhi, "Cost Efficient Design of Fault Tolerant Geo-Distributed Data Centers," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 289–301, June 2017.

[145] M. R. M. Assis and L. F. Bittencourt, "MultiCloud Tournament: A Cloud Federation Approach to Prevent Free-Riders by Encouraging Resource Sharing," *Journal of Network and Computer Applications*, vol. 166, p. 102694, Sep. 2020.

# List of Publications

## List of Publications from Thesis

### Refereed Journals

1. **Moustafa Najm**, Rakesh Tripathi, Mohammad Shadi Alhakeem and Venkatesh Tamarapalli, "A Cost-aware Management Framework for Placement of Data-intensive Applications on Federated Cloud", *Journal of Network and Systems Management (JNSM)*, Springer, Volume 29, Issue 3, pp. 33:1-33:25, March 2021.
DOI: `https://doi.org/10.1007/s10922-021-09594-9`

2. **Moustafa Najm**, Moumita Patra and Venkatesh Tamarapalli, "Cost-and-Delay Aware Dynamic Resource Allocation in Federated Vehicular Clouds", *IEEE Transactions on Vehicular Technology (TVT)*, Volume 70, Issue 6, pp. 6159:6171, June 2021.
DOI: `https://doi.org/10.1109/TVT.2021.3079912`

3. **Moustafa Najm** and Venkatesh Tamarapalli, "Towards Cost-aware VM Migration to Maximize the Profit in Federated Clouds", *Communicated to Elsevier Future Generation Computer Systems (FGCS)*

### Refereed International Conferences/Worskhops

4. **Moustafa Najm** and Venkatesh Tamarapalli, "Cost Efficient Deployment of Big Data Applications in Federated Cloud Systems", $11^{th}$ *International Conference on COMmunication Systems and NETworkS (COMSNETS '19)*, Bangalore, India, January 2019, pp.428-431.

5. **Moustafa Najm** and Venkatesh Tamarapalli, "A Cost-aware Algorithm for Placement of Enterprise Applications in Federated Cloud Data Center", $20^{th}$ *International Conference on Distributed Computing and Networking (ICDCN '19)*, Bangalore, India, January 2019, pp.510.

6. **Moustafa Najm** and Venkatesh Tamarapalli, "VM Migration for Profit Maximization in Federated Cloud Data Centers", $12^{th}$ *International Conference on COMmunication Systems and NETworkS (COMSNETS '20)*, Bangalore, India, January 2020, pp.802-804. **[Best PhD Forum Presentation]**

7. **Moustafa Najm** and Venkatesh Tamarapalli, "Inter-Data center Virtual Machine Migration in Federated Cloud", $21^{th}$ *International Conference on Distributed Computing and Networking (ICDCN '20)*, Kolkata, India, January 2020, article no. 62, pp.1.

8. **Moustafa Najm**, Moumita Patra and Venkatesh Tamarapalli, "An Adaptive and Dynamic Allocation of Delay-sensitive Vehicular Services in Federated Cloud", $13^{th}$ *International Conference on COMmunication Systems and NETworkS (COMSNETS '21)*, Bangalore, India, January 2021, pp.97-100.

# Doctoral Committee

**Chairperson:**
Prof. Diganta Goswami
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

**Research Advisor:**
Dr. Venkatesh Tamarapalli
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

**Members:**
Prof. S. V. Rao
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Prof. Sanjay Kumar Bose
Professor
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati

Dr. Ramesh Kumar Sonkar
Assistant Professor
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati

# Vitae

**Moustafa Mohamad Najm** joined the Ph.D. programme in the Department of Computer Science and Engineering at Indian Institute of Technology (IIT) Guwahati, India in July 2016. In IIT Guwahati, he was affiliated with the User-Centric Computing and Networking (UCNET) Lab of the Department of Computer Science and Engineering. Prior to joining his Ph.D., he worked as an assistant professor at the Faculty of Informatics Engineering, Damascus University, Syria. He received his Master of Technology (M.Tech) in Computer Science and Engineering from National Institute of Technology (NIT) Warangal, Telangana, India, in 2014. Prior to that, he received his Bachelor of Informatics Engineering from Faculty of Informatics Engineering, Damascus University, Syria in 2009. He received the Best PhD Forum Award at The 12th International Conference on Communication Systems and Networks (COMSNETS 2020) in Bengaluru, India. His Ph.D. work focuses on Federated cloud data center, resource management in cloud data center, Internet of Things (IoT), and Vehicular Cloud Computing (VCC).

## Contact Information

**E-mail:** moustafa.najm@iitg.ac.in
mmnajm@gmail.com