Journal of Network and Computer Applications 98 (2017) 65–83

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An algorithm for network and data-aware placement of multi-tier applications in cloud data centers

Md Hasanul Ferdaus[a,b,*], Manzur Murshed[c], Rodrigo N. Calheiros[d], Rajkumar Buyya[b]

[a] Faculty of Information Technology, 25 Exhibition Walk, Clayton campus, Monash University, VIC 3800, Australia
[b] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Australia
[c] Faculty of Science and Technology, Federation University Australia, Northways Road, Churchill, VIC 3842, Australia
[d] School of Computing, Engineering and Mathematics, Western Sydney University, Australia

ARTICLE INFO

Keywords:
Virtual machine
Network-aware
Storage
Data center
Placement
Optimization
Cloud application
Cloud computing

ABSTRACT

Today's Cloud applications are dominated by composite applications comprising multiple computing and data components with strong communication correlations among them. Although Cloud providers are deploying large number of computing and storage devices to address the ever increasing demand for computing and storage resources, network resource demands are emerging as one of the key areas of performance bottleneck. This paper addresses network-aware placement of virtual components (computing and data) of multi-tier applications in data centers and formally defines the placement as an optimization problem. The simultaneous placement of Virtual Machines and data blocks aims at reducing the network overhead of the data center network infrastructure. A greedy heuristic is proposed for the on-demand application components placement that localizes network traffic in the data center interconnect. Such optimization helps reducing communication overhead in upper layer network switches that will eventually reduce the overall traffic volume across the data center. This, in turn, will help reducing packet transmission delay, increasing network performance, and minimizing the energy consumption of network components. Experimental results demonstrate performance superiority of the proposed algorithm over other approaches where it outperforms the state-of-the-art network-aware application placement algorithm across all performance metrics by reducing the average network cost up to 67% and network usage at core switches up to 84%, as well as increasing the average number of application deployments up to 18%.

1. Introduction

With the pragmatic realization of computing as a utility, Cloud Computing has recently emerged as a highly successful alternative information technology paradigm through the unique features of on-demand resource provisioning, pay-as-you-go business model, virtually unlimited amount of computing resources, and high reliability (Buyya et al., 2009). In order to meet the rapidly increasing demand for computing, communication, and storage resources, Cloud providers are deploying large-scale data centers comprising thousands of servers across the planet. These data centers are experiencing sharp rise in network traffic and a major portion of this traffic is constituted of the data communication within the data center. Recent report from Cisco Systems Inc. (Cisco, 2015) demonstrates that the Cloud data centers will dominate the global data center traffic flow for the foreseeable future and its importance is highlighted by one of the top-line projections from this forecast that, by 2019, more than four-fifths of the total data center traffic will be Cloud traffic (Fig. 1). One important trait pointed out by the report is that a majority of the global data center traffic is generated due to the data communication within the data centers: in 2014, it was 75.4% and it will be around 73.1% in 2019.

This huge amount of intra-data center traffic is primarily generated by the application components that are correlated to each other, for example, the computing components of a composite application (e.g., MapReduce) writing data to the storage array after it has processed the data. This large growth of data center traffic may pose serious scalability problems for wide adoption of Cloud Computing. Moreover, by the way of continuously rising popularity of social networking sites, e-commerce, and Internet-based gaming applications, large amount of data processing has become an integral part of Cloud applications. Furthermore,

* Corresponding author at: Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, Building 168, The University of Melbourne, Parkville, VIC 3053, Australia.
E-mail addresses: md.ferdaus@monash.edu (M.H. Ferdaus), manzur.murshed@federation.edu (M. Murshed), R.Calheiros@westernsydney.edu.au (R.N. Calheiros), rbuyya@unimelb.edu.au (R. Buyya).

http://dx.doi.org/10.1016/j.jnca.2017.09.009
Received 5 September 2016; Received in revised form 15 April 2017; Accepted 14 September 2017
Available online 19 September 2017

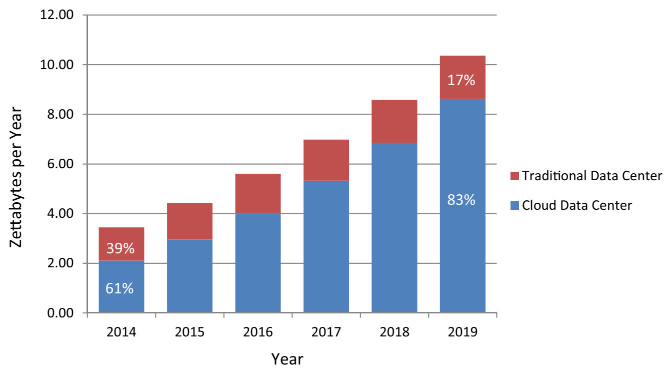


Fig. 1. Worldwide data center traffic growth (data source: Cisco).

scientific processing, multimedia rendering, workflow, and other massive parallel processing and business applications are being migrated to the Clouds due to the unique advantages of high scalability, reliability, and pay-per-use business model. Over and above, recent trend in Big Data computing using Cloud resources (Assuncao et al., 2015) is emerging as a rapidly growing factor contributing to the rise of network traffic in Cloud data centers.

One of the key technological elements that have paved the way for the extreme success of Cloud Computing is virtualization. Modern data centers leverage various virtualization technologies (e.g., machine, network, and storage virtualization) to provide users an abstraction layer that delivers a uniform and seamless computing platform by hiding the underlying hardware heterogeneity, geographic boundaries, and internal management complexities (Zhang et al., 2010). By the use of virtualization, physical server resources are abstracted and shared through partial or full machine simulation by time-sharing, and hardware and software partitioning into multiple execution environments, known as *Virtual Machines* (VMs), each of which runs as a complete and isolated system. It allows dynamic sharing and reconfiguration of physical resources in Cloud infrastructures that make it possible to run multiple applications in separate VMs having different performance metrics. It also facilitates Cloud providers to improve utilization of physical servers through VM multiplexing (Meng et al., 2010a) and multi-tenancy, i.e., simultaneous sharing of physical resources of the same server by multiple Cloud customers. Furthermore, it enables on-demand resource pooling through which computing (e.g., CPU and memory), network, and storage resources are provisioned to customers only when needed (Kusic et al., 2009). By utilizing these flexible features of virtualization for provisioning physical resources, the scalability of data center network can be improved through minimization of network load imposed due to the deployment of customer applications.

On the other side, modern Cloud applications are dominated by multi-component applications such as multi-tier applications, massive parallel processing applications, scientific and business workflows, content delivery networks, and so on. These applications usually have multiple computing and associated data components. The computing components are usually delivered to customers in the form of VMs, such as Amazon EC2 Instances,[1] whereas the data components are delivered as data blocks, such as Amazon EBS.[2] These computing components of such applications have specific service roles and are arranged in layers in the overall structural design of the application. For example, large enterprise applications are often modeled as 3-tier applications: the presentation tier (e.g., web server), the logic tier (e.g., application server), and the data tier (e.g., relational database) (Urgaonkar et al., 2005). The computing components (VMs) of such applications have specific communication requirements among themselves, as well as with the data blocks that are associated to those VMs (Fig. 2). As a consequence, overall performance of such applications highly depends on the communication delays among the computing and data components. From the Cloud providers' perspective, optimization of network utilization of data center resources is tantamount to profit maximization. Moreover, efficient bandwidth allocation and reduction of data packet hopping through network devices (e.g., switches or routers) trim down the overall energy consumption of network infrastructure. On the other hand, Cloud consumers' concern is to receive guaranteed Quality of Service (QoS) of the delivered virtual resources, which can be assured through appropriate provisioning of requested resources.

Given the issues of sharp rise in network traffic in data centers, this paper addresses the scalability concern of data center network through a traffic-aware placement strategy of multi-component, composite application (in particular, VMs and data blocks) in virtualized data center that aims at optimizing the network traffic load incurred due to placement decision. Such placement decisions can be made during the application deployment phase in the data center. VM placement decisions focusing on other goals rather than network efficiency, such as energy consumption reduction (Feller et al., 2011; Beloglazov and Buyya, 2012) and server resource utilization (Gao et al., 2013; Ferdaus et al., 2014), often result in placements where VMs with high mutual traffic are placed in host servers with high mutual network cost. For example, one of our previous works (Ferdaus et al., 2014) on the placement of a cluster of VMs strives to consolidate the VMs into a minimal number of servers in order to reduce server resource wastage. By this process, unused servers can be kept into lower power states (e.g., suspended) so as to improve power efficiency of the data center. Since this approach does not consider inter-VM network communication patterns, such placement decisions can eventually result in locating VMs with high mutual network traffic in long distant servers, such as servers locating across the network edges. Several other VM placement works focusing on non-network objectives can be found in (Wu and Ishikawa, 2015; Farahnakian et al., 2015; Nguyen et al., 2014; Corradi et al., 2014; Alboaneen et al., 2014). With a network-focused analysis, it can be concluded that research works such as the above ones considered single-tier applications and VM clusters without consideration of mutual network communication within the application components or VMs. On the contrary, this paper focuses on placing mutually communicating components of applications (such as VMs and data blocks) in data center components (such as physical servers and storage devices) with lesser network cost so that network overhead imposed due to the application placement is minimized. With this placement goal, the best placement for two communicating VMs would be in the same server where they can communicate through memory copy, rather than using the physical network links. This paper effectively addresses network-focused placement problem of multi-tiered applications with components having mutual network communication rather than single-tiered ones. The significance of the network-focused placement of multi-tiered applications is evident from the experimental results presented later in Section 5, where it is observed that an efficient non-network greedy placement algorithm, namely First Fit Decreasing (FFD), incurs higher network costs compared to the proposed network-aware placement heuristic.

Moreover, advanced hardware devices with combined capabilities are opening new opportunities for efficient resource allocation focusing on application needs. For example, Dell PowerEdge C8000 moduler servers are equipped with CPU, GPU, and storage components that can work as multi-function devices. Combined placement of application components with high mutual traffic (e.g., VMs and their associated data components) in such multi-function servers will effectively reduce the data transfer delay since the data accessed by the VMs reside in the same devices. Similar trends are found in high-end network switches (e.g., Cisco MDS 9200 Multiservice Switches) that come with additional built-in processing and storage capabilities. Reflecting on these tech-

[1] Amazon EC2 - Virtual Server Hosting, 2016. https://aws.amazon.com/ec2/.
[2] Amazon Elastic Block Store (EBS), 2016. https://aws.amazon.com/ebs/.

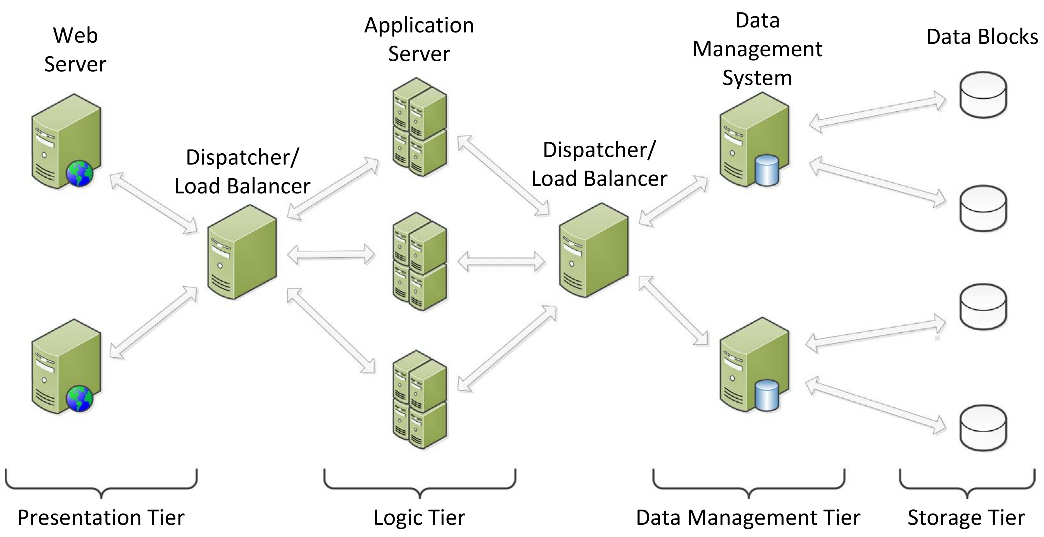


Fig. 2. Multi-tier application architecture.

nological development and multi-purpose devices, this paper has considered a generic approach in modeling computing, network, and storage elements in a data center so that placement algorithms can make efficient decision for application components placement in order to achieve the ultimate goal of network cost reduction.

This research work investigates the allocation, specifically on-demand placement of composite application components (modeled as an *Application Environment*) requested by the customers to be deployed in Cloud data center focusing on network utilization, with consideration of computing, network, and storage resources capacity constraints of the data center. In particular, this paper has the following contributions:

1. The Network-aware Application environment Placement Problem (NAPP) is formally defined as a combinatorial optimization problem with the objective of network cost minimization due to the placement. The proposed data center and application environment models are generic and are not restricted to any specific data center topology and application type or structure, respectively.
2. Given the resource requirements and structure of the application environment to be deployed, and information on the current resource state of the data center, a Network- and Data location-aware Application environment Placement (NDAP) scheme is proposed. NDAP is a greedy heuristic that generates mappings for simultaneous placement of the computing and data components of the application into the computing and storage nodes of the data center, respectively, focusing on minimization of incurred network traffic, while respecting the computing, network, and storage capacity constraints of data center resources. While making placement decisions, NDAP strives to reduce the distance that data packets need to travel in the data center network, which in turn, helps to localize network traffic and reduces communication overhead in the upper layer network switches.
3. Finally, performance evaluation of the proposed approach is conducted through elaborate simulation-based experimentation across multiple performance metrics and several scaling factors. The results suggest that the NDAP algorithm successfully improves network resource utilization through efficient placement of application components and outperforms compared algorithms significantly across all performance metrics.

The proposed NDAP greedy heuristic for placement of application environments, while optimizing the overall network overhead, is addressing an important sub-problem of a much bigger multi-objective placement problem that simultaneously optimizes computing, storage, and communications resources. While many multi-objective works are available in the literature aiming at consolidation of the first two kinds of resources (computing and storage), works addressing all three kinds of resources are few and these works only considered placement of VMs in isolation. Development of NDAP is the first step in addressing the comprehensive optimization problem considering the placement of a group of closely-linked VMs, hereby termed as an application environment.

The remainder of this paper is organized as follows. A brief background on the related works is presented in Section 2. Section 3 formally defines the addressed application placement problem (NAPP) as an optimization problem, along with the associated mathematical models. The proposed network-aware, application placement approach (NDAP) and its associated algorithms are elaborately explicated in Section 4. Section 5 details the experiments performed and shows the results, together with their analysis. Finally, Section 6 concludes the paper with a summary of the contribution and future research directions.

2. Related work

During the past several years, a good amount of research works have been carried out in the area of VM scheduling, placement, and migration strategies in virtualized data centers, and more recently, focusing on Cloud data centers. A major portion of these works focus on servers resource utilization (Nguyen et al., 2014; Gao et al., 2013), energy-efficiency (Farahnakian et al., 2014; Beloglazov, 2013), and application performance (Gupta et al., 2013; Calcavecchia et al., 2012), and so on (Ferdaus and Murshed, 2014) in the context of large infrastructures. Recently, a handful of works are published in the area of VM placement and migration with focus on network resources that are briefly described below. Kakadia et al. (2013) presented a VM grouping mechanism based on network traffic history within a data center at run-time and proposed a fast, greedy VM consolidation algorithm in order to improve hosted applications performance and optimize the network usage by saving internal bandwidth. Through simulation-based evaluation, the authors have shown that the proposed VM consolidation algorithm achieves better performance compared to traditional VM placement approaches, within an order to magnitude faster and requires much less VM migrations. Dias and Costa (2012) addressed the problem of traffic concentration in data center networks by reallocating VMs in physical servers based on current traffic matrix and server resource usage. The authors proposed a scheme for partitioning server based on connectivity capacity and available computing resources, as well as VM clustering mechanism depending on the amount of data exchanged among the VMs. The proposed VM placement algorithm tries to find mappings for matching all the VM clusters in the server partitions, respecting the server resource capacity constraints. Shrivastava et al. (2011) proposed a topology-aware VM migration scheme for managing overloaded VMs considering the complete application context running on the VMs and the server resource capacity constraints. The goal of the proposed VM migration algorithm is to relocate overloaded VMs to physical servers so that the run-time network load within data center is minimized. Similar VM placement and relocation works can be found in (Zhang et al., 2016; Biran et al., 2012; Meng et al., 2010b), demand-based VM provisioning works for multi-component Cloud applications are presented in (Srirama and Ostovar, 2014) and in (Sahu et al., 2014), and policy-aware. All the above mentioned traffic-aware VM placement and consolidation works aim at run-time scenarios for relocating running VMs within the data center through VM migration.

Several other recent VM placement and consolidation works have been proposed focusing on simultaneous optimization of energy and traffic load in data centers. Vu and Hwang (2014) addressed the issues of VM migration from underloaded and overloaded PMs at run-time and presented an offline algorithm for individual VM migration with the focus on traffic- and energy consumption reduction. Energy efficiency is achieved by consolidating VMs in high capacity servers as much as possible and traffic efficiency is achieved by migrating VMs near to communicating peer VMs. Wang et al. (2014) addressed the problem of unbalanced resource utilization and network traffic in data center during run-time, and proposed an energy-efficient and QoS-aware VM placement mechanism that groups the running VMs into partitions to reduce traffic communication across the data center, determines to server for migrating the VMs, and finally, uses the OpenFlow controller to assign paths to balance the traffic load and avoid congestion. Takouna et al. (2013) presented mechanisms for dynamically determining the bandwidth demand and communication patter of HPC and parallel applications in data center and reallocating the communicative VMs through live migration. The objective of the proposed approach is to reduce network link utilization and energy saving through aggregating communicative VMs. The authors have shown substantial improvement in data center traffic volume through simulation-based evaluation. Gao et al. (2016) addressed the problem of energy cost reduction under both server and network resource constraints within data center and proposed a VM placement strategy based on Ant Colony Optimization incorporating network resource factor with server resources. Huang et al. (2013) addressed the server overload problem and presented a three-stage joint optimization framework that minimizes the number of used servers in the data center in order to reduce power consumption, communication costs, and finally, a combined approach that focus on both the above goals through the use of VM migrations. Lloyd et al. (2014) investigated the problem of virtual resource provisioning and placement of service oriented applications through dynamic scaling in Cloud infrastructures and presented a server load-aware VM placement scheme that improves application performance and reduces resource cost. Similar multi-objective VM placement and migration works can also be found in (Zhang et al., 2012; Huang et al., 2012; Wang et al., 2013; Song et al., 2012) that target optimization of energy consumption reduction, server resource utilization, and network usage. Given the fact that VM live migrations are costly operations (Liu et al., 2013), the above mentioned VM relocation strategies overlook the impact of necessary VM migrations and reconfiguration on hosted applications, physical servers and network devices. Further recent works on network-aware VM placement and migration can be found in (Li and Qian, 2015; Alharbi and Walker, 2016; Cui et al., 2017; Zhao et al., 2015; Wang et al., 2016). A detailed taxonomy and survey on various existing network-aware VM management strategies can be found in our previous work (Ferdaus et al., 2015).

Contrary to the above mentioned works, this paper addresses the problem of network efficient, on-demand placement of composite applications consisting of multiple VMs and associated data components, along with inter-component communication pattern, in a data center consisting of both computing servers and storage devices. The addressed problem does not involve VM migrations since the placement decision is taken during the application deployment phase. Recently, Georgiou et al. (2013) have addressed the benefit of user-provided hints on inter-VM communication during the online VM cluster placement and proposed two placement heuristics utilizing the properties of PortLand network topology (Mysore et al., 2009). However, this work does not involve any data component for VM-cluster specification. On the other hand, both of the proposed composite, multi-tier application and data center models of this paper are generic and are not restricted to any particular application or data center topology. Data location-aware VM placement works can be found in (Piao and Yan, 2010) and in (Korupolu et al., 2009), however, these works modeled the applications as single instance of VM, which is an oversimplified view of today's Cloud or Internet applications that are mostly composed of multiple computing and storage entities in multi-tier structure with strong communication correlations among the components. In order to reflect on this, this paper investigates a much wider VM communication model by considering placement of Application Environments, each involving a number of VMs and associated data blocks with sparse communication links between them.

3. Problem statement

While deploying composite applications in Cloud data centers, such as multi-tier or workflow applications, customers request multiple computing VMs in the form of a VM cluster or a Virtual Private Cloud and multiple Data Blocks (DBs). These computing VMs have specific traffic flow requirements among themselves, as well as with the data blocks. The remainder of this section formally defines such composite application environment placement as an optimization problem. Fig. 3 presents a visual representation of the application placement in data center and Table 1 provides the various notations used in the problem definition and proposed solution.

3.1. Formal definition

An *Application Environment* is defined as $AE = \{VMS, DBS\}$, where VMS is the set of requested VMs: $VMS = \{VM_i: 1 \leq i \leq N_v\}$ and DBS is the set of requested DBs: $DBS = \{DB_k: 1 \leq k \leq N_d\}$. Each VM VM_i has specification of its CPU and memory demands represented by VM_i^{cpu} and VM_i^{mem}, respectively, and each DB DB_k has specification of its storage resource demand denoted by DB_k^{str}.

Data communication requirements between any two VMs, and between a VM and a DB are specified as *Virtual Links* (VLs) between $\langle VM, VM\rangle$ pairs and $\langle VM, DB\rangle$ pairs, respectively, during AE specification and deployment. The bandwidth demand or traffic load between VM_i and VM_j is represented by $BW(VM_i, VM_j)$. Similarly, the bandwidth demand between VM_i and DB_k is represented by $BW(VM_i, DB_k)$. These bandwidth requirements are provided as user input along with the VM and DB specifications.

A *Data Center* is defined as $DC = \{CNS, SNS\}$ where CNS is the set of computing nodes (e.g., physical servers or computing components of a multi-function storage device) in DC: $CNS = \{CN_p: 1 \leq p \leq N_c\}$ and SNS is the set of storage nodes: $SNS = \{SN_r: 1 \leq r \leq N_s\}$. For each computing node CN_p, the available CPU and memory resource capacities are represented by CN_p^{cpu} and CN_p^{mem}, respectively. Here available resources mean the remaining usable resources of a CN that may have already hosted other VMs that are consuming the rest of the resources. Similarly, for each storage node SN_r, the available storage resource capacity is represented by SN_r^{str}.

Computing nodes and storage nodes are interconnected through *Physical Links* (PLs) in the data center communication network. PL distance and available bandwidth between two computing nodes CN_p and CN_q are denoted by $DS(CN_p, CN_q)$ and $BA(CN_p, CN_q)$, respectively.

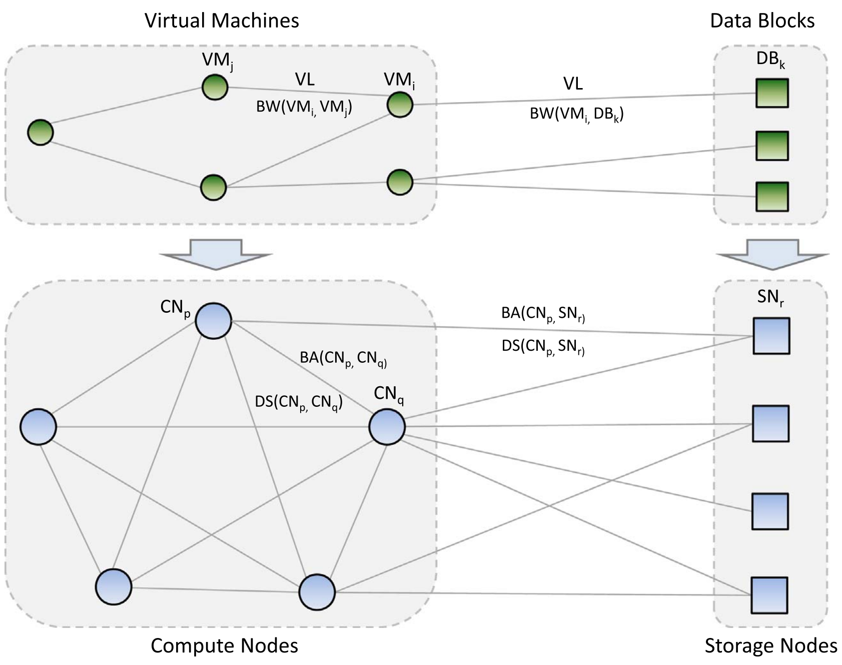


Fig. 3. Application environment placement on data center.

Table 1
Notations and their meanings.

Notation	*Meaning*
VM	Virtual machine
DB	Data block
AN	AE node (either a VM or a DB)
VMS	Set of VMs in an AE
DBS	Set of DBs in an AE
ANS	Set of ANs ($ANS = \{VMS \cup DBS\}$) in an AE
N_v	Total number of VMs in an AE
N_d	Total number of DBs in an AE
VL	Virtual Link
VCL	Virtual Computing Link that connects two VMs
VDL	Virtual Data Link that connects a VM and a DB
$vclList$	Ordered list of VCLs in an AE
$vdlList$	Ordered list of VDLs in an AE
N_{vc}	Total number of VCLs in an AE
N_{vd}	Total number of VDLs in an AE
N_{vn}	Average number of NTPP VLs of a VM or a DB
$BW(VM_i, VM_j)$	Bandwidth demand between VM_i and VM_j
$BW(VM_i, DB_k)$	Bandwidth demand between VM_i and DB_k
CN	Computing Node
SN	Storage Node
$DN(AN)$	DC node where AN is placed
CNS	Set of CNs in a DC
SNS	Set of SNs in a DC
N_c	Total number of CNs in a DC
N_s	Total number of SNs in a DC
$cnList$	Ordered list of CNs in a DC
$snList$	Ordered list of SNs in a DC
PL	Physical network Link
PCL	Physical Computing Link that connects two CNs
PDL	Physical Data Link that connects a CN and a SN
$DS(CN_p, CN_q)$	Network distance between CN_p and CN_q
$DS(CN_p, SN_r)$	Network distance between CN_p and SN_r
$BA(CN_p, CN_q)$	Available bandwidth between CN_p and CN_q
$BA(CN_p, SN_r)$	Available bandwidth between CN_p and SN_r

Similarly, PL distance and available bandwidth between a computing node CN_p and a storage node SN_r are represented by $DS(CN_p, SN_r)$ and $BA(CN_p, SN_r)$, respectively. PL distance can be any practical measure, such as link latency, number of hops or switches, and so on. Also, the proposed model does not restrict the data center to a fixed network topology. Thus, the network distance DS and available bandwidth BA models are generic and different model formulations focusing on any particular network topology or architecture can be readily applied in the optimization framework and proposed solution. In the experiments, the number of hops or switches between any two data center nodes is used as the only input parameter for DS function in order to measure the PL distance. Although singular distances between $\langle CN, CN\rangle$ and $\langle CN, SN\rangle$ pairs are used in the experiments, network link redundancy and multiple communication paths in data center can be incorporated in the proposed model and placement algorithm by appropriately defining distance function (DS) and available bandwidth function (BA), respectively.

Furthermore, $DN(VM_i)$ denotes the computing node where VM_i is currently placed, otherwise if VM_i is not already placed, $DN(VM_i) = null$. Similarly, $DN(DB_k)$ denotes the storage node where DB_k is currently placed.

The network cost of placing VM_i in CN_p and VM_j in CN_q is defined as the following:

$$Cost(VM_i, CN_p, VM_j, CN_q) = BW(VM_i, VM_j) \times DS(CN_p, CN_q). \quad (1)$$

Likewise, the network cost of placing VM_i in CN_p and DB_k in SN_r is defined as the following:

$$Cost(VM_i, CN_p, DB_k, SN_r) = BW(VM_i, DB_k) \times DS(CN_p, SN_r). \quad (2)$$

Given the AE to deploy in the DC, the objective of the NAPP problem is to find placements for VMs and DBs in CNs and SNs, respectively, in such a way that the overall network cost or communication overhead due to the AE deployment is minimized. Thus, the *Objective Function f* is defined as the following:

$$\underset{\substack{\forall i:DN(VM_i)\\ \forall k:DN(VM_k)}}{\text{minimize}} f(AE, DC) = \sum_{i=1}^{N_v}\left(\sum_{j=1}^{N_v} Cost(VM_i, DN(VM_i), VM_j, DN(VM_j)) + \sum_{k=1}^{N_d} Cost(VM_i, DN(VM_i), DB_k, DN(DB_k))\right). \quad (3)$$

The above AE placement is subject to the constraints that the available resource capacities of any CN and SN are not violated:

$$\forall\, p: \sum_{\forall i:DN(VM_i)=CN_p} VM_i^{cpu} \le CN_p^{cpu}. \quad (4)$$

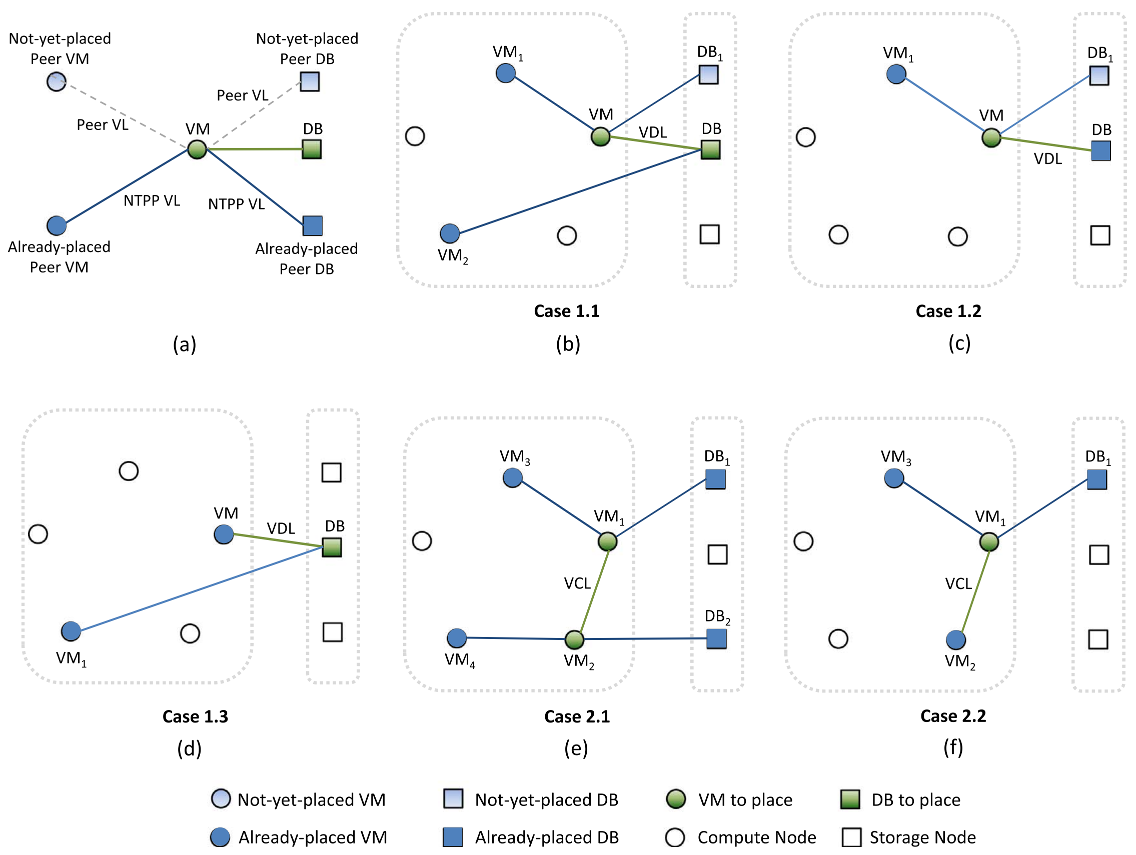


Fig. 4. (a) Peer VL and NTPP VL, and (b-f) Five possible VL placement scenarios.

$$\forall\ p\!: \sum_{\forall\, i:DN(VM_i)=CN_p} VM_i^{mem} \le CN_p^{mem}. \tag{5}$$

$$\forall\ r\!: \sum_{\forall\, k:DN(DB_k)=SN_r} DB_k^{str} \le SN_r^{str}. \tag{6}$$

Furthermore, the sum of the bandwidth demands of the VLs that are placed on each PL must be less or equal to the available bandwidth of the PL:

$$\forall\ p\ \forall\ q\!: BA(CN_p, CN_q) \ge \sum_{\forall\, i:DN(VM_i)=CN_p}\ \sum_{\forall\, j:DN(VM_j)=CN_q} BW(VM_i, VM_j). \tag{7}$$

$$\forall\ p\ \forall\ r\!: BA(CN_p, SN_r) \ge \sum_{\forall\, i:DN(VM_i)=CN_p}\ \sum_{\forall\, k:DN(DB_k)=SN_r} BW(VM_i, DB_k). \tag{8}$$

Given that every VM and DB placement fulfills the above mentioned constraints (Eq. (4)–(8)), the NAPP problem defined by objective function f (Eq. (3)) is explained as: among all possible feasible placements of VMs and DBs in AE, the placement that has minimum cost is the optimal solution. Thus, NAPP falls in the category of combinatorial optimization problem. In particular, it is an extended form of the *Quadratic Assignment Problem* (QAP) (Loiola et al., 2007), which is proven to be computationally $\mathcal{NP}-$ hard (Burkard et al., 1998).

4. Proposed solution

The proposed network-aware VM and DB placement approach (NDAP) tries to place the VLs in such a way that network packets need to travel short distances. For better explanation of the solution approach, the above described models of AE and DC are extended by adding few other notations.

Every AE node is represented by *AN* which can either be a VM or a DB, and the set of all ANs in an AE is represented by *ANS*. Every *VL* can be either a *Virtual computing Link (VCL)*, i.e., *VL* between two VMs or a *Virtual Data Link (VDL)*, i.e., VL between a VM and a DB. The total number of VCL and VDL in an AE is represented by N_{vc} and N_{vd}, respectively. All the VCLs and VDLs are maintained in two ordered lists *vclList* and *vdlList*, respectively. While VM-VM communication (VCL) and VM-DB communication (VDL) may be considered closely related, they differ in terms of actor and size. As only a VM can initiate communications, VCL supports an "active" duplex link while VDL supports a "passive" duplex link. More distinctly, bandwidth demands of VDLs are multiple orders larger than the same of VCLs.

Every DC node is represented by *DN* which can either be a *CN* or a *SN*. All the CNs and SNs in a DC are maintained in two ordered lists *cnList* and *snList*, respectively. Every *PL* can be either a *Physical computing Link (PCL)*, i.e., *PL* between two CNs or a *Physical Data Link (PDL)* i.e., *PL* between a CN and a SN.

The proposed NDAP algorithm is a greedy heuristic that first sorts the *vdlList* and *vclList* in decreasing order of the bandwidth demand of VDLs and VCLs. Then, it tries to places all the VDLs from *vdlList*, along with any associated VCLs to fulfill placement dependency, on the feasible PDLs and PCLs, and their associated VMs and DBs in CNs and SNs, respectively, focusing on the goal of minimizing the incurred network cost due to placement of all the VDLs and associated VCLs. Finally, NDAP tries to place the remaining VCLs from *vclList* on PCLs, along with their associated VMs and DBs in CNs and SNs, respectively, again targeting on reducing the incurred network cost.

As mentioned in Section 3, NAPP is in fact an $\mathcal{NP}-$ hard combinatorial optimization problem similar to QAP and Sahni and Gonzalez (1976) have shown that even finding an approximate solution for QAP within some constant factor from the optimal solution cannot be done in polynomial time unless $\mathcal{P} = \mathcal{NP}$. Considering the fact that greedy heuristics are relatively fast, easy to understand and implement, and very often used as an effective solution approach for $\mathcal{NP}$ − *complete* problems, this paper proposes NDAP greedy heuristic as a solution for the NAPP problem.

A straight forward placement of an individual VL (either VDL or VCL) on a preferred PL is not always possible since one or both of its ANs can have *Peer ANs* connected by *Peer VLs* (Fig. 4(a)). At any point during an AE placement process, a VL can have Peer ANs that are already placed. The peer VLs that have already-placed peer ANs is termed as *need-to-place peer VLs* (NTPP VLs), indicating the condition that placement of any VL also needs to perform simultaneous placement of its NTPP VLs, and the average number of NTPP VLs for any VM or DB is denoted by N_{vn}. The maximum value of N_{vn} can be $N_v + N_d - 1$ which indicates that the corresponding VM or DB has VLs with all the other VMs and DBs in the AE. Since, for any VL placement, the corresponding placement of its NTPP VLs is an integrated part of the NDAP placement strategy, firstly the VL placement feasibility part of the NDAP algorithm is presented in the following subsection. Afterwards, the next four subsections describe other constituent components of the NDAP algorithm. Finally, a detailed description of the final NDAP algorithm is provided along with the pseudocode.

4.1. VL placement feasibility

During the course of AE placement, when NDAP tries to place a VL that has one or both of its ANs not placed yet (i.e., $DN(AN) = null$), then a feasible placement for the VL needs to ensure that (1) the VL itself is placed on a feasible PL, (2) its ANs are placed on feasible DNs, and (3) all the NTPP VLs are placed on feasible PLs.

Depending on the type of VL and the current placement status of its ANs, five different cases may arise that are presented below. The NDAP placement algorithm handles these five cases separately. Fig. 4(b)-(f) provide a visual representation of the five cases where the VL to place is shown as solid green line and its NTPP VLs are shown as solid blue lines.

VDL Placement: When trying to place a VDL, any of the following three cases may arise:

Case 1.1: Both the VM and DB are not placed yet and their peers VM_1, DB_1, and VM_2 are already placed (Fig. 4(b)).

Case 1.2: DB is placed but VM is not placed yet and VM's peers VM_1 and DB_1 are already placed (Fig. 4(c)).

Case 1.3: VM is placed but DB is not placed yet and DB's peer VM_1 is already placed (Fig. 4(d)).

VCL Placement: In case of VCL placement, any of the following two cases may arise:

Case 2.1: Both the VMs (VM_1 and VM_2) are not placed yet and their peers VM_3, DB_1, VM_4, and DB_2 are already placed (Fig. 4(e)).

Case 2.2: Only one of the VMs is already placed and its peers VM_3 and DB_1 are already placed (Fig. 4(f)).

In all the above cases, placement feasibility of the NTPP VDLs and VCLs of the not-yet-placed VMs and DBs must be checked against the corresponding PDLs and PCLs, respectively (Eq. (7) & (8)).

4.2. Feasibility and network cost of VM and peer VLs placement

When NDAP tries to place a VM in a CN, it is feasible when (1) the computing and memory resource demands of the VM can be fulfilled by the remaining computing and memory resource capacities of the CN, and (2) the bandwidth demands of all the NTPP VLs can be satisfied by the available bandwidth capacities of the corresponding underlying PLs (Fig. 5(a)):

$$VMPeerFeas(VM, CN) = \begin{cases} 1, & \text{if Eq0.4\&5holds and, } DN(AN)\neq\text{null and} \\ & BW(VM, AN) \leq BA(CN, DN(AN)) \text{ for } \forall AN; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

When NDAP tries to place two VMs (VM_1 and VM_2) in a single CN, it is feasible when (1) the combined computing and memory resource demands of the two VMs can be fulfilled by the remaining computing and memory resource capacities of the CN, and (2) the bandwidth demands of all the NTPP VLs of both the VMs can be satisfied by the available bandwidth capacities of the corresponding underlying PLs:

$$VMPeerFeas(VM_1, VM_2, CN) = \begin{cases} 1, & \text{if Eq.4\&5holds for}(VM_1 + VM_2) \text{ and,} \\ & \forall AN: DN(AN)\neq\text{null and,} \\ & BW(VM_1, AN) + BW(VM_2, AN) \leq BA(CN, DN(AN)); \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The network cost of a VM placement is measured as the accumulated cost of placing all of its NTPP VLs:

$$VMPeerCost(VM, CN) = \sum_{\forall AN:\, DN(AN)\neq null \wedge BW(VM,AN)>0} Cost(VM, CN, AN, DN(AN)). \tag{11}$$

4.3. Feasibility and network cost of DB and peer VLs placement

When trying to place a DB in a SN, it is feasible when (1) the storage resource demand of the DB can be fulfilled by the remaining storage resource capacity of the SN, and (2) the bandwidth demands of the NTPP VLs can be satisfied by the available bandwidth capacities of corresponding underlying PLs (Fig. 5(a)):

$$DBPeerFeas(DB, SN) = \begin{cases} 1, & \text{if Eq.6holds and,} DN(AN)\neq\text{null and} \\ & BW(AN, DB) \leq BA(DN(AN), SN) \text{ for } \forall AN; \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The network cost of any DB placement is measured as the total cost of placing all of its NTPP VLs:

$$DBPeerCost(DB, SN) = \sum_{\forall AN:\, DN(AN)\neq null \wedge BW(AN,DB)>0} Cost(AN, DN(AN), DB, SN). \tag{13}$$

4.4. VM and peer VLs placement

Algorithm 1 shows the subroutine for placing a VM and its associated NTPP VLs. Firstly, the VM-to-CN placement is accomplished

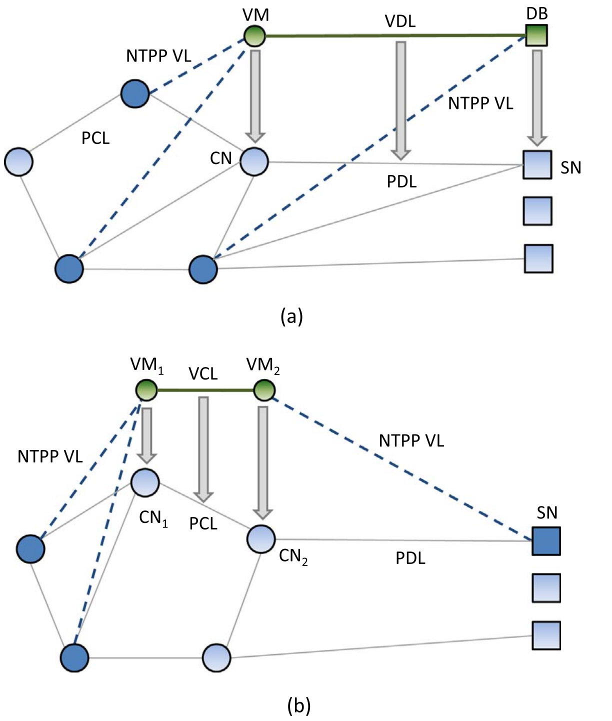


Fig. 5. Placement of (a) VDL and (b) VCL along with NTPP VLs.

by reducing the available CPU and memory resource capacities of the *CN* by the amount of CPU and memory resource requirements of the *VM* and setting the *CN* as the DC node of the *VM* [line 1]. Then, for each already-placed peer *AN* of *VM* (i.e., any *AN* that has non-zero traffic load with *VM* and $DN(AN) \neq null$), it is checked if the selected *CN* is different from the computing node where the peer *AN* is placed, in which case the available bandwidth capacity of the PL that connects the selected *CN* and $DN(AN)$ is reduced by the amount of the bandwidth demand of the corresponding NTPP VL [lines 2–4]. In those cases where the selected *CN* is the computing node where the peer *AN* is placed, the *VM* can communicate with the peer *AN* through memory copy instead of passing packet through physical network links. Afterwards, the NTPP *VL* is removed from the *vclList* or *vdlList*, depending on whether it is a VCL or VDL, respectively, in order to indicate that it is now placed [lines 5–7].

Algorithm 1. PlaceVMandPeerVLs

```
Input: $VM$ to place, $CN$ where $VM$ is being placed, set of all ANs $ANS$, $vclList$, and $vdlList$.
Output: $VM$-to-$CN$ and $VL$-to-$PL$ placements.
1:  $CN^{cpu} \leftarrow CN^{cpu} - VM^{cpu}$; $CN^{mem} \leftarrow CN^{mem} - VM^{mem}$; $DN(VM) \leftarrow CN$;
2:  for each $AN \in ANS$ do
3:    if $BW(VM, AN) > 0 \wedge DN(AN) \neq null$ then
4:      if $DN(AN) \neq CN$ then $BA(CN, DN(AN)) \leftarrow BA(CN, DN(AN)) - BW(VM, AN)$;
        endif
5:      $VL \leftarrow virtualLink(VM, AN)$;
6:      if $VL$ is a $VCL$ then $vclList.remove(VL)$;
7:      else $vdlList.remove(VL)$;
8:      endif
9:    end if
10: end for
```

4.5. DB and peer VLs placement

Algorithm 2 shows the subroutine for placing a *DB* in a *SN* and its associated NTPP VLs. Firstly, the *DB*-to-*SN* placement is performed by reducing the available storage capacity of the *SN* by the amount of the storage requirements of the *DB* and by setting the *SN* as the DC node of *DB* [line 1]. Then, for every already-placed peer *AN* of *DB* (i.e., any *AN* that has non-zero traffic load with *DB* and $DN(AN) \neq null$), the available bandwidth capacity of the PDL that connects the selected *SN* and $DN(AN)$ is reduced by the amount of the NTPP *VL*'s bandwidth requirement and the NTPP *VL* is removed from the *vdlList* to mark that it is now placed [lines 2–6].

Algorithm 2. PlaceDBandPeerVLs

```
Input: $DB$ to place, $SN$ where $DB$ is being placed, set of all ANs $ANS$, and $vdlList$.
Output: $DB$-to-$SN$ and $VL$-to-$PL$ placements.
1:  $SN^{str} \leftarrow SN^{str} - DB^{str}$; $DN(DB) \leftarrow SN$;
2:  for each $AN \in ANS$ do
3:    if $BW(AN, DB) > 0 \wedge DN(AN) \neq null$ then
4:      $BA(DN(AN), SN) \leftarrow BA(DN(AN), SN) - BW(AN, DB)$; $VL \leftarrow virtualLink(AN, DB)$; $vdlList.remove(VL)$;
5:    end if
6:  end for
```

4.6. NDAP algorithm

The pseudocode of the final NDAP algorithm is presented in Algorithm 3. It receives the *DC* and *AE* as input and returns the network cost incurred due to the *AE* placement. NDAP begins by performing necessary initialization and sorting the *vdlList* and *vclList* in decreasing order of their VLs' bandwidth demands [line 1]. Afterwards, it iteratively takes the first VDL from *vdlList* (i.e., VDL with highest bandwidth demand) and tries to place it (along with its VM and DB, and all NTPP VLs) in a PDL among the feasible PDLs so that the total network cost incurred due to the placement is minimum [lines 2–29] (Fig. 5(a)). As explained in Section 4.1, there can be three cases for this placement depending on current placement status of the VDL's VM and DB.

When the VDL matches Case 1.1 (both VM and DB are not placed), then for each feasible CN and SN in DC (Eq. (9) and (12)), it is checked if the bandwidth demand of the VDL can be satisfied by the available bandwidth of the corresponding PDL connecting the CN and SN. If it can be satisfied, then the total cost of placing the VDL and its associated NTPP VLs is measured (Eq. (11) and (13)). The $\langle CN, SN \rangle$ pair that offers the minimum cost is selected for placing the $\langle VM, DB \rangle$ pair and the available bandwidth capacity of the PDL that connects the selected $\langle CN, SN \rangle$ pair is updated to reflect the VDL placement [lines 4–13]. When the VDL matches Case 1.2 (VM is not placed, but DB is placed), the feasible CN that offers minimum cost placement is selected for the VM and the total cost is measured [lines 14–18]. In a similar way, Case 1.3 (VM is placed, but DB is not placed) is handled in lines 19–24 and the best SN is selected for the DB placement.

If NDAP fails to find a feasible CN or SN, it returns -1 to indicate failure in finding a feasible placement for the AE [line 25]. Otherwise, it activates the placements of the VM and DB along with their NTPP VLs by using subroutines *PlaceVMandPeerVLs* (Algorithm 1) and *PlaceDBandPeerVLs* (Algorithm 2), accumulates the measured cost in variable *totCost*, and removes the VDL from *vdlList* [lines 26–28]. In this way, by picking the VDLs from a list that is already sorted based on bandwidth demand and trying to place each VDL, along with its NTPP VLs, in such a way that the incurred network cost is minimum in the current context of the DC resource state, NDAP strives to minimize the total network cost of placing the AE as formulated by the objective function f (Eq. (3)) of the proposed optimization. In particular, in each iteration of the first while loop (lines 2–29), NDAP pick the next highest bandwidth demanding VDL from the *vdlList* and finds the best placement (i.e., minimum cost) for it along with its NTPP VLs. Moreover, the placement of the VDLs are performed before the placement of the VCLs since the average VDL bandwidth demand is expected to be higher than the average VCL bandwidth demand considering the fact that the average traffic volume for $\langle VM, DB \rangle$ pair is supposed to be higher than that for $\langle VM, VM \rangle$ pair.

After NDAP has successfully placed all the VDLs, then it starts placing the remaining VCLs in the *vclList* (i.e., VCLs that were not NTPP VLs during the VDLs placement). For this part of the placement, NDAP applies a similar approach by repeatedly taking the first VCL from *vclList* and trying to place it on a feasible PCL so that the incurred network cost is minimum [lines 30–55] (Fig. 5(b)). This time, there can be two cases depending on the placement status of the two VMs of the VCL (Section 4.1).

Algorithm 3. NDAP Algorithm.

```
Input: $DC$ and $AE$.
Output: Total network cost of $AE$ placement.
1:  $totCost \leftarrow 0$; Sort $vdlList$ and $vclList$ in decreasing order of VL's bandwidth demands;
2:  while $vdlList \neq \emptyset$ do { NDAP tries to place all VDLs in vdlList }
3:    $VDL \leftarrow vdlList[0]$; $minCost \leftarrow \infty$; $VM \leftarrow VDL.VM$; $DB \leftarrow VDL.DB$; $selCN \leftarrow null$; $selSN \leftarrow null$;
4:    if $DN(VM) = null \wedge DN(DB) = null$ then {Case 1.1: Both VM and DB are not placed}
5:      for each $CN \in cnList \wedge VMPeerFeas(VM, CN) = 1$ do
6:        for each $SN \in snList \wedge DBPeerFeas(DB, SN) = 1$ do
7:          if $BW(VM, DB) \leq BA(CN, SN)$ then
8:            $cost \leftarrow BW(VM, DB) \times DS(CN, SN) + VMPeerCost(VM, CN) + DBPeerCost(DB, SN)$;
9:            if $cost < minCost$ then $minCost \leftarrow cost$; $selCN \leftarrow CN$; $selSN \leftarrow SN$; endif
10:         end if
11:       end for
12:     end for
13:   if $minCost \neq \infty$ then $BA(selCN, selSN) \leftarrow BA(selCN, selSN) - BW(VM, DB)$; endif
14: else if $DN(VM) = null \wedge DN(DB) \neq null$ then { Case 1.2: VM is not placed and DB is already placed }
15:   for each $CN \in cnList \wedge VMPeerFeas(VM, CN) = 1$ do
16:     $cost \leftarrow VMPeerCost(VM, CN)$;
17:     if $cost < minCost$ then $minCost \leftarrow cost$; $selCN \leftarrow CN$; endif
18:   end for
19:   else if $DN(VM) \neq null \wedge DN(DB) = null$ then {Case 1.3: VM is already placed and DB is not placed}
20:       for each $SN \in snList \wedge DBPeerFeas(DB, SN) = 1$ do
21:         $cost \leftarrow DBPeerCost(DB, SN)$;
22:         if $cost < minCost$ then $minCost \leftarrow cost$; $selSN \leftarrow SN$; endif
23:       end for
24:   end if
25:   if $minCost = \infty$ then return $-1$; endif {Feasible placement not found}
26:   if $selCN \neq null$ then $PlaceVMandPeerVLs(VM, selCN)$; endif {For Case 1.1 and Case 1.2}
27:   if $selSN \neq null$ then $PlaceDBandPeerVLs(DB, selSN)$; endif {For Case 1.1 and Case 1.3}
28:   $totCost \leftarrow totCost + minCost$; $vdlList.remove(0)$;
29: end while
30: while $vclList \neq \emptyset$ do {NDAP tries to place remaining VCLs in vclList}
31:   $VCL \leftarrow vclList[0]$; $minCost \leftarrow \infty$; $VM_1 \leftarrow VCL.VM_1$; $VM_2$
        $\leftarrow VCL.VM_2$; $selCN_1 \leftarrow null$; $selCN_2 \leftarrow null$;
32:   if $DN(VM_1) = null \wedge DN(VM_2) = null$ then {Case 2.1: Both VMs are not placed}
33:     for each $CN_1 \in cnList \wedge VMPeerFeas(VM_1, CN_1) = 1$ do
34:         for each $CN_2 \in cnList \wedge VMPeerFeas(VM_2, CN_2) = 1$ do
35:           if $CN_1 = CN_2 \wedge VMPeerFeas(VM_1, VM_2, CN) = 0$ then continue; endif
36:           if $BW(VM_1, VM_2) \leq BA(CN_1, CN_2)$ then
37:             $cost \leftarrow BW(VM_1, VM_2) \times DS(CN_1, CN_2)$;
38:             $cost \leftarrow cost + VMPeerCost(VM_1, CN_1) + VMPeerCost(VM_2, CN_2)$;
39:             if $cost < minCost$ then $minCost \leftarrow cost$; $selCN_1 \leftarrow CN_1$; $selCN_2 \leftarrow CN_2$; endif
40:           end if
41:         end for
42:       end for
43:     if $minCost \neq \infty$ then $BA(selCN_1, selCN_2) \leftarrow BA(selCN_1, selCN_2) - BW(VM_1, VM_2)$; endif
44:   else if $DN(VM_1) \neq null \vee DN(VM_2) \neq null$ then {Case 2.2: One of the VMs is not placed}
45:     if $DN(VM_1) \neq null$ then swap values of $VM_1$ and $VM_2$; endif {Now $VM_1$ denotes the not-yet-placed VM}
46:     for each $CN_1 \in cnList \wedge VMPeerFeas(VM_1, CN_1) = 1$ do
47:       $cost \leftarrow VMPeerCost(VM_1, CN_1)$;
48:       if $cost < minCost$ then $minCost \leftarrow cost$; $selCN_1 \leftarrow CN_1$; endif
49:     end for
50:   end if
51:   if $minCost = \infty$ then return $-1$; endif {Feasible placement not found}
52:   $PlaceVMandPeerVLs(VM_1, selCN_1)$; {For Case 2.1 and Case 2.2}
53:   if $selCN_2 \neq null$ then $PlaceVMandPeerVLs(VM_2, selCN_2)$; endif {For Case 2.1}
54:   $totCost \leftarrow totCost + minCost$; $vclList.remove(0)$;
55: end while
56: return $totCost$;
```

When the VCL matches Case 2.1 (both VMs are not placed), then for each feasible CN in DC (Eq. (9)), it is first checked if both the VMs (VM_1 and VM_2) are being tried for placement in the same CN. In such cases, if the combined placement of both the VMs along with their NTPP VLs are not feasible (Eq. (10)), then NDAP continues checking feasibility for different CNs [line 35]. When both VMs placement feasibility passes and the bandwidth demand of the VCL can be satisfied by the available bandwidth of the corresponding PCL connecting the CNs, then the total cost of placing the VCL and its associated NTPP VLs is measured (Eq. (11) and 13) [lines 36–40]. When both the VMs are being tried for the same CN, then they can communicate with each other using memory copy rather going through physical network link and the available bandwidth check in line 36 works correctly since the intra-CN available bandwidth is considered to be unlimited. The $\langle CN_1, CN_2\rangle$ pair that offers the minimum cost is selected for placing the $\langle VM_1, VM_2\rangle$ pair and the available bandwidth capacity of the PCL connecting the selected $\langle CN_1, CN_2\rangle$ pair is updated to reflect the VCL placement [lines 39–43]. When the VCL matches Case 2.2 (one of the VMs is not placed), the feasible CN that offers minimum cost placement is selected for the not-yet-placed VM (VM_1) and the total cost is measured [lines 44–50].

Similar to VDL placement, if NDAP fails to find feasible CNs for any VCL placement, it returns −1 to indicate failure [line 51]. Otherwise, it activates the placements of the VMs along with their NTPP VLs by using subroutine *PlaceVMandPeerVLs* (Algorithm 1), accumulates the measured cost in *totCost*, and removes the VCL from *vclList* [lines 52–55]. For the same reason as for VDL placement, the VCL placement part of the NDAP algorithm fosters the reduction of the objective function *f* value (Eq. (3)).

Finally, NDAP returns the total cost of the *AE* placement, which also indicates a successful placement [line 56].

5. Performance evaluation

This section describes the performance of the proposed NDAP algorithm compared to other algorithms through a set of simulation based experiments. Section 5.1 gives a brief description of the evaluated algorithms, Section 5.2 describes the various aspects of the simulation environment, and finally, the results are discussed in the subsequent sections.

5.1. Algorithms compared

The following algorithms are evaluated and compared in this work:

Network-aware VM Allocation (NVA): This is an extended version of the network-aware VM placement approach proposed by Piao and Yan (2010) where the authors have considered already-placed data blocks. In this version, each $DB \in DBS$ is placed randomly in a $SN \in SNS$. Afterwards, each VM that has one or more VDL is placed according to the VM allocation algorithm presented by the authors, provided that all of its NTPP VLs are placed on feasible PLs. For any remaining $VM \in VMS$, it is placed randomly. All the above placements are subject to the constraints presented in Eq. (4), (5), (6), (7), and (8). In order to increase the probability of feasible placements, DB and VM placements are tried multiple times and the maximum number of tries (N_{mt}) is parameterized by a constant which is set to 100 in the simulation. For the above mentioned implementation, the worst-case time complexity of NVA algorithm is given by:

$$T_{NVA} = O(N_d N_{mt}) + O(N_v N_c N_{vn}) + O(N_v N_{mt}). \tag{14}$$

Given the fact that N_{mt} is a constant and the maximum number of VMs (N_v) and DBs (N_d) in an AE is generally much less than the number of computing nodes (N_c) in DC, the above time complexity reduces to:

$$T_{NVA} = O(N_v N_c N_{vn}). \tag{15}$$

Given that NVA starts with already-placed DBs, and VM placements are done in-place using no auxiliary data structure, NVA algorithm itself does not have any memory overhead.

First Fit Decreasing (FFD): This algorithm begins by sorting the CNs in *cnList* and SNs in *snList* in decreasing order based on their remaining resource capacities. Since, CNs have two different types of resource capacities (CPU and memory), L1-norm mean estimator is used to convert the vector representation of multi-dimensional resource into scalar form. Similarly, all the VMs in *vmList* and DBs in *dbList* are sorted in decreasing order of their resource demands, respectively. Then, FFD places each DB from *dbList* in the first feasible SN of *snList* according to the *First First* (FF) algorithm. Afterwards, it places each VM from *vmList* in the first feasible CN of *cnList* along with any associated NTPP VLs. All the above placements are subject to the constraints presented in Eq. (4), (5), (6), (7), and (8). For this implementation of FFD, the worst-case time complexity of FFD algorithm is given by:

$$T_{FFD} = O(N_c \lg N_c) + O(N_s \lg N_s) + O(N_v \lg N_v) + O(N_d \lg N_d) + O(N_d N_s) + O(N_v N_c). \tag{16}$$

Given the fact that, in a typical setting the number of VMs (N_v) and DBs (N_d) in an AE is much less than the number of CNs (N_c) and SNs (N_s) in DC, respectively, the above term reduces to:

$$T_{FFD} = O(N_c \lg N_c) + O(N_s \lg N_s) + O(N_d N_s) + O(N_v N_c). \tag{17}$$

Given that merge sort (Cormen et al., 2001) is used in FFD to sort *cnList*, *snList*, *vmList*, and *dbList*, and N_c is usually greater than each of N_s, N_v, and N_d in a typical setting, it can be concluded that the memory overhead for the sorting operation is $O(N_c)$. Apart from sorting, the placement decision part of FFD works in-place without using any additional data structure. Therefore, the memory overhead of FFD algorithm is $O(N_c)$.

Network- and Data-aware Application Placement (NDAP): The NDAP algorithm is implemented primarily based on the description presented in Section 4 and follows the execution flow presented in Algorithm 3. The final NDAP algorithm utilizes the feasibility check (Eq. (9), (10), and (12)), network cost computation (Eq. (11) and (13)), and the placement subroutines (Algorithm 1 and 2). All of these NDAP components need to go through a list of NTPP VLs for the corresponding VM or DB, and in the implementation, this list is stored in an array. Thus, the time complexity for each of these NDAP components is $O(N_{vn})$. For the above mentioned implementation, the running time of NDAP algorithm (refering to pseudocode in Algorithm 3) is the sum of the time needed for sorting *vdlList* and *vdlList* (T_2), the time needed for placing all the VDLs in *vdlList* (T_{3-34}), and the time needed for placing all the remaining VCLs in *vclList* (T_{36-65}). The time complexity for placing a single VDL (considering three cases) is given by:

$$\begin{aligned} T_{6-33} &= O(N_c N_s N_{vn}) + O(N_c N_{vn}) + O(N_s N_{vn}) + O(N_{vn}) \\ &= O(N_c N_s N_{vn}). \end{aligned} \tag{18}$$

Therefore, the time complexity for placing all the VDLs is:

$$T_{3-34} = O(N_{vd} N_c N_s N_{vn}). \tag{19}$$

Similarly, the time complexity for placing all the remaining VCLs is:

$$T_{36-65} = O(N_{vc} N_c^2 N_{vn}). \tag{20}$$

Thus, the worst-case time complexity of NDAP algorithm is given by:

$$\begin{aligned} T_{NDAP} &= T_2 + T_{3-34} + T_{36-65} \\ &= O(N_{vd} \lg N_{vd}) + O(N_{vc} \lg N_{vc}) + O(N_{vd} N_c N_s N_{vn}) + O(N_{vc} N_c^2 N_{vn}). \end{aligned} \tag{21}$$

For this implementation of NDAP algorithm, merge sort is used in order to sort *vdlList* and *vclList* [line 2, Algorithms 3]. Given that AEs are typically constituted of a number of VMs and DBs with sparse communication links between them, it is assumed that $N_{vd} = N_{vc} = O(N_v)$ since N_{vd} and N_{vc} are of the same order. Thus, the memory overhead for this sorting operation is $O(N_v)$. Apart from sorting, the placement decision part of NDAP [lines 3–67] works in-place and no additional data structure is needed. Therefore, the memory overhead of NDAP algorithm is $O(N_v)$.

The detailed computational time complexity analyses presented above may be further simplified as follows. While the number of computing node outweighs the number of storage node in a typical DC, these may be assumed of the same order, i.e., $N_s = O(N_c)$. Moreover, the size of a typical DC is at least multiple order higher than that of an AE. Hence, it can also be assumed that $N_v, N_d, N_{vc}, N_{vd}, N_{vn} = o(N_c)$. From Eq. (15), (17), & (21), it can be concluded that the running time of NVA, FFD, and NDAP algorithms are $O(N_c)$, $O(N_c \lg N_c)$, and $O(N_c^2)$, respectively, i.e., these are linear, linearithmic, and quadratic time algorithms, respectively. Regarding the overhead of the above mentioned algorithms, although there are variations in the run-time memory overhead, considering that the input optimization problem (i.e., AE placement in DC) itself has $O(N_c)$ memory overhead, it can be concluded that, overall, all the compared algorithms have equal memory overhead of $O(N_c)$.

For all the above algorithms, if any feasible placement is not found for a VM or DB, the corresponding algorithm terminates with failure status. The algorithms are implemented in Java (JDK and JRE version 1.7.0) and the simulation is conducted on a Dell Workstation (Intel Core i5-2400 3.10 GHz CPU (4 cores), 4 GB of RAM, and 240 GB storage) hosting Windows 7 Professional Edition.

5.2. Simulation setup

5.2.1. Data center setup

In order to address the increasing complexity of large-scale Cloud data centers, network vendors are coming up with network architecture models focusing on the resource usage patterns of Cloud applications. For example, Juniper Networks Inc. in their ”Cloud-ready data center reference architecture” suggests the use of Storage Area Networks (SAN) interconnected to the computing network with converged access switches (Juniper, 2012), similar to the one shown in Fig. 6. The simulated data center is generated following this reference architecture with three-tier computing network topology (core-aggregation-access) (Kliazovich et al., 2013) and SAN-based storage network. Following the approach presented in (Korupolu et al., 2009)), the number of parameters is limited in simulating the data center by using the number of physical computing servers as the only parameter denoted by N. The number of other data center nodes are derived from N as follows: $5N/36$ high-end storage devices with built-in spare computing resources that work as multi-function devices for storage and computing, $4N/36(=N/9)$ regular storage devices without additional computing resources, $N/36$ high-end core switches with built-in spare computing resources that work as multi-function devices for switching and computing, $N/18$ mid-level aggregation switches, and $5N/12(=N/3+N/12)$ access switches. Following the three-tier network topology (Kliazovich et al., 2013), $N/3$ access switches provide connectivity between N computing servers and $N/18$ aggregation switches, whereas the $N/18$ aggregation switches connects $N/3$ access switches and $N/36$ core switches in the computing network. The remaining $N/12$ access switches provide connectivity between $N/4$ storage devices and $N/36$ core switches in the storage network. In such a data center setup, the total number of computing nodes (CNs) $N_c = N + 5N/36 + N/36 = 7N/6$ and the total number of storage nodes (SNs) $N_s = 5N/36 + 4N/36 = N/4$.

Network distance between $\langle CN, CN\rangle$ pairs and between $\langle CN, SN\rangle$ pairs are measured as $DS = h \times DF$, where h is the number of physical hops between two DC nodes (CN or SN) in the simulated data center architecture as defined above, and DF is the *Distance Factor* that implies the physical inter-hop distance. The value of h is computed using the analytical expression for tree topology as presented in (Meng et al., 2010b) and DF is fed as a parameter to the simulation. Network distance of a node with itself is 0 which implies that data communication is done using memory copy without going through the network. A higher value of DF indicates greater relative communication distance between any two data center nodes.

5.2.2. Application environment setup

In order to model composite application environments for the simulation, multi-tier enterprise applications and scientific workflows are considered as representatives of the dominant Cloud applications. According to the analytical model for multi-tier Internet applications presented in (Urgaonkar et al., 2005), three-tier applications are modeled as comprised of 5 VMs ($N_v = 5$) and 3 DBs ($N_d = 3$) interconnected through 4 VCLs ($N_{vc} = 4$) and 5 VDLs ($N_{vd} = 5$) as shown in Fig. 7(a). In order to model scientific applications, Montage workflow is simulated as composed of 7 VMs ($N_v = 7$) and 4 DBs ($N_d = 4$) interconnected through 5 VCLs ($N_{vc} = 5$) and 9 VDLs ($N_{vd} = 9$) following the structure presented in (Juve et al., 2013) (Fig. 7(b)). While deploying an application in data center, user provided hints on estimated resource demands are parameterized during the course of the experimentation. Extending the approaches presented in Meng et al. (2010b) and in Shrivastava et al. (2011), computing resource demands (CPU and memory) for VMs, storage resource demands for DBs, and bandwidth demands for VLs are stochastically generated based on normal distribution with parameter means ($meanCom$, $meanStr$, and $meanVLBW$, respectively) and standard deviation (sd) against normalized total resource capacities of CNs and SNs, and bandwidth capacities of PLs, respectively.

5.2.3. Simulated scenarios

For each of the experiments, all the algorithms start with their own empty data centers. In order to represent the dynamics of the real Cloud data centers, two types of events are simulated: (1) AE deployment and (2) AE termination. With the purpose of assessing the relative performance of the various placement algorithms in states of both higher and lower resource availability of data center nodes (CNs and SNs) and physical links (PCLs and PDLs), the experiments simulated scenarios where the average number of AE deployments doubles the average number of AE terminations. Since during the initial phase of the experiments the data centers are empty, algorithms enjoy more freedom for the placement of AE components. Gradually, the data centers get loaded due to higher number of AE deployments compared to the number of AE terminations. In order to reflect upon the reality of application deployment dynamics in real Clouds where the majority of the Cloud application spectrum is composed of multi-tier enterprise applications, in the simulated scenarios, 80% of the AE deployments are considered to be enterprise applications (three-tier application models) and 20% are considered as scientific applications (Montage workflow models). Overall, the following two scenarios are considered:

Group Scenario: For all the placement algorithms, AE deployments and terminations are continued until any of them fails to place an AE due to the lack of feasible placement. For maintaining fairness among algorithms, the total number of AE deployments and terminations for each of the placement algorithms are equal and the same instances of AEs are deployed or terminated for each simulated event.

Individual Scenario: For each of the algorithms, AE deployment and termination is continued separately until it fails to place an AE due to the lack of a finding feasible placement. Similar to the group scenario, all the algorithms draw AEs from same pools so that all the algorithms work with the same AE for each event.

All the experiments presented in this paper are repeated 1000 times and the average results are reported.

5.2.4. Performance evaluation metrics

In order to assess the network load imposed due the placement decisions, the average network cost of AE deployment is computed

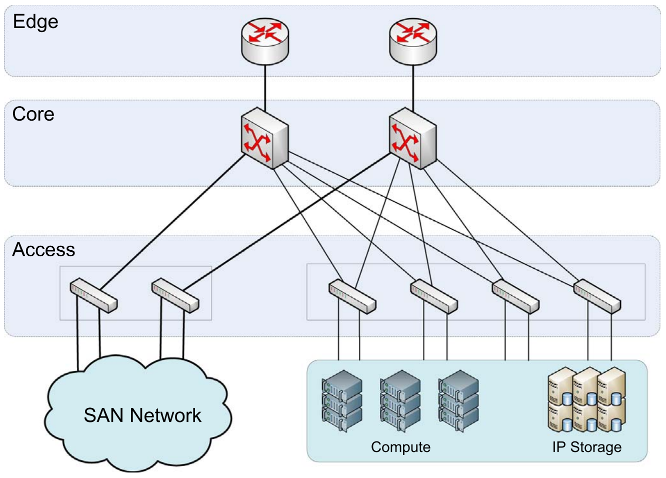


Fig. 6. Cloud-ready data center network architecture.

(using objective function f accordingly to Eq. (3)) for each of the algorithms in the group scenario. Since the cost functions (Eq. (1) and (2)) are defined based on network distance between DC nodes and expected amount of traffic flow, it effectively provides measures of the network packet transfer delays, and imposed packet forwarding load and power consumption for the network devices (e.g., switches and routers) and communication links. With the aim of maintaining a fair comparison among the algorithms, the average cost metric is computed and compared in the group scenario where all the algorithms terminate when any of them fails to place an AE due to the feasible resource constraints (Eq. (4), (5), (6), (7), and (8)) in DC and, as a consequence, each algorithm works with the same instances of AE at each deployment and termination event, and the average cost is computed over the same number of AEs.

In order to measure how effectively each of the algorithms utilizes the network bandwidth during AE placements, the total number of AE deployments in empty DC is measured until the data center saturates in the individual scenario. Through this performance metric, the effective capacity of the DC resources utilized by each of the placement algorithms is captured and compared.

In order to assess how effectively the placement algorithms localize network traffic and, eventually, optimize network performance, the average network utilization of access, aggregation, and core switches are measured in the group scenario. In this part of the evaluation, the group scenario is chosen so that when any of the algorithms fail to place an AE, all the algorithms halt their placements with the purpose of keeping the total network loads imposed on the respective data centers for each of the algorithms remain same. This switch-level network usage assessment is performed through scaling the mean and standard deviation of the VLs' bandwidth demands.

Finally, the average placement decision computation time for AE deployment is measured for the individual scenario. Average placement decision time is an important performance metric to assess the efficacy of NDAP as an on-demand AE placement algorithm and its scalability across various factors.

All the above performance metrics are measured against the following scaling factors: (1) DC size, (2) mean resource demands of VMs, DBs, and VLs, (3) diversification of workloads, and (4) network distance factor DF. The following subsections present the experimental results and analysis for each of the experiments conducted.

5.3. Scaling data center size

In this part of the experiment, the placement quality of the algorithms with increasing size of the DC is evaluated and compared. As mentioned in Section 5.2.1, N is used as the only parameter to denote DC size, and its minimum and maximum values are set to 72 and 4608, respectively, doubling at each subsequent simulation phase. Thus, in the largest DC there are a total of 5376 CNs and 1152 SNs. The other parameters $meanCom$, $meanStr$, $meanVLBW$, sd, and DF are set to 0.3, 0.4, 0.35, 0.5, and 2, respectively.

Fig. 8(a) shows the average cost of AE placement incurred by each of the three algorithms in the group scenario for different values of N. From the chart, it is quite evident that NDAP consistently outperforms the other placement algorithms at a much higher level for the different DC sizes and its average AE placement cost is 56% and 36% less than NVA and FFD, respectively. Being network-aware, NDAP checks the feasible placements with the goal of minimizing the network cost. FFD, on the other hand, tries to place the ANs in DNs with maximum

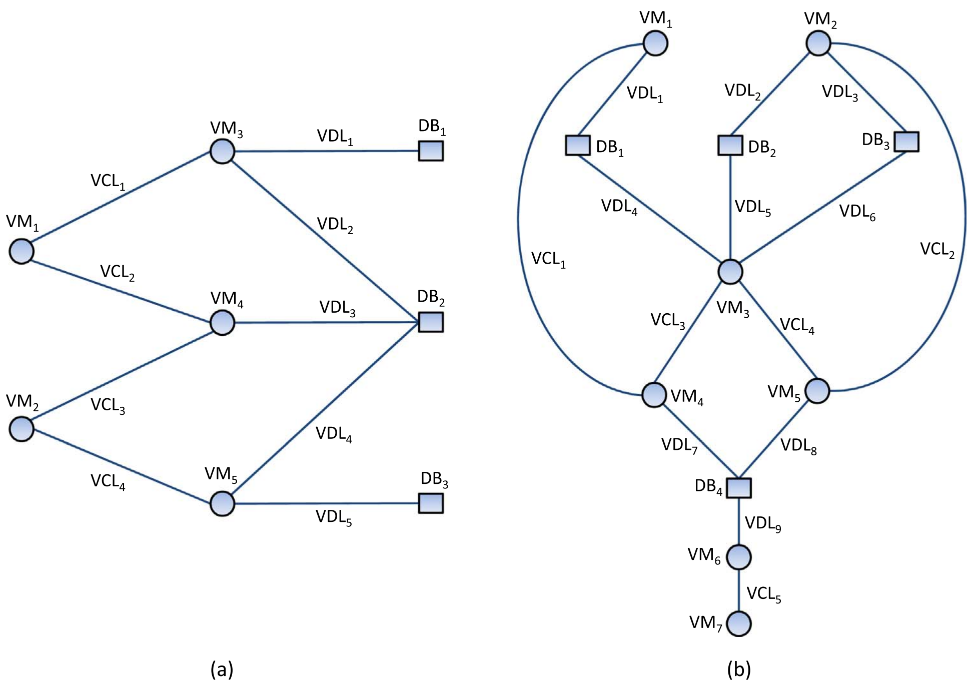


Fig. 7. Application environment models for (a) Multi-tier application and (b) Scientific (Montage) workflow.

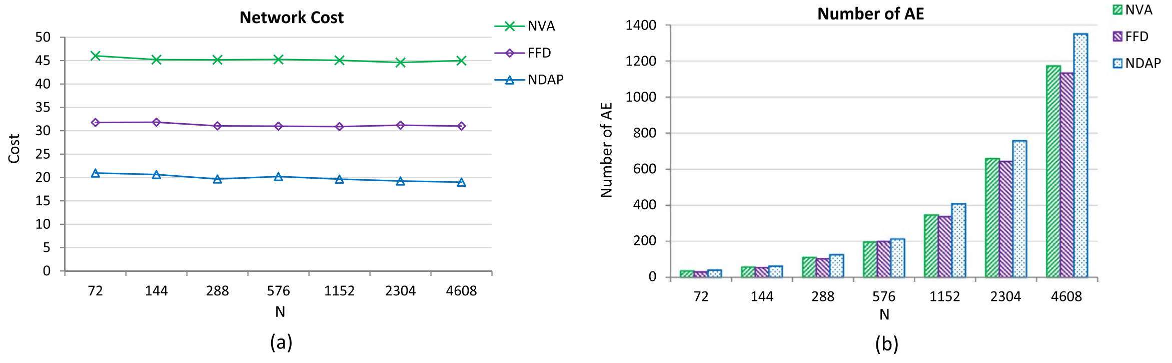


Fig. 8. Performance with increasing N: (a) Network cost and (b) Number of AE deployed in DC.

available resource capacities and, as a result, has possibility of placing VLs on shorter PLs. And, finally, NVA has random components in placement decisions and, thus, incurs higher average cost.

From Fig. 8(b), it can be observed that the average number of successful AE deployments in the individual scenario by the algorithms increases non-linearly with the DC size as more DNs and PLs (i.e., resources) are available for AE deployments. It is also evident that NDAP deploys larger number of AEs in data center compared to other algorithms until the data center is saturated with resource demands. The relative performance of NDAP remains almost steady across different data center sizes— it deploys around 13–17% and 18–21% more AEs compared to NVA and FFD, respectively. This demonstrates the fact that NDAP's effectiveness in utilizing the data center resources is not affected by the scale of the data center.

5.4. Variation of mean resource demands

This experiment assesses the solution qualities of the placement algorithms when the mean resource demands of the AEs increase. Since the AE is composed of different components, the mean resource demands are varied in two different approaches presented in the rest of this subsection. As for the other parameters N, sd, and DF are set to 1152, 0.4, and 2, respectively.

5.4.1. Homogeneous mean resource demands

The same *mean* (i.e., $meanCom = meanStr = meanVLBW = mean$) is used to generate the computing (CPU and memory) resource demands of VMs, storage resource demands of DBs, and bandwidth demands of VLs under normal distribution. The experiment starts with a small *mean* of 0.1 and increases it upto 0.7, adding up with 0.1 at each subsequent phase.

The average cost for AE placement is shown in Fig. 9(a) for the group scenario. It is obvious form the chart that NDAP achieves much better performance compared to other placement algorithms— on average it incurs 55% and 35% less cost compared to NVA and FFD, respectively. With the increase of mean resource demands, the incurred cost for each algorithm increases almost at a constant rate. The reason behind this performance pattern is that when the mean resource demands of the AE components (VMs, DBs, and VLs) increase with respect to the available resource capacities of the DC components (CNs, SNs, and PLs), the domain of feasible placements is reduced which causes the rise in the average network cost.

Fig. 9(b) shows the average number of AEs deployed in empty DC with increasing *mean* for the individual scenario. It can be seen from the chart that the number of AEs deployed by the algorithms constantly reduces as higher mean values are used to generate the resource demands. This is due to the fact that when resource demands are increased compared to the available resource capacities, the DC nodes and PLs can accommodate fewer number of AE nodes and VLs. One interesting observation from this figure is that FFD was able to deploy fewer number of AEs compared to NVA when the *mean* was small. This can be attributed to the multiple random tries during ANs placement by NVA which helps it to find feasible placements, although with higher average cost. Overall, NDAP has been able to place larger number of AEs compared to other algorithms across all mean values: 10–18% and 12–26% more AEs than NVA and FFD, respectively.

5.4.2. Heterogeneous mean resource demands

In order to assess the performance variations across different mean levels of resource demands of AE components, two different mean levels L (low) and H (high) are set in this part of the experiment for mean VM computing resource demands ($meanCom$ for both CPU and memory), mean DB storage resource demands ($meanStr$), and mean VL bandwidth demands ($meanVLBW$). L and H levels are set to 0.2 and 0.7 for this simulation. Given the two levels for the three types of resource demands, there are eight possible combinations.

Fig. 10(a) shows the average network costs of the three algorithms for the eight different mean levels (x axis of the chart). The three different positions of the labels are set as follows: the left-most, the middle, and the right-most positions are for $meanCom$, $meanStr$, and

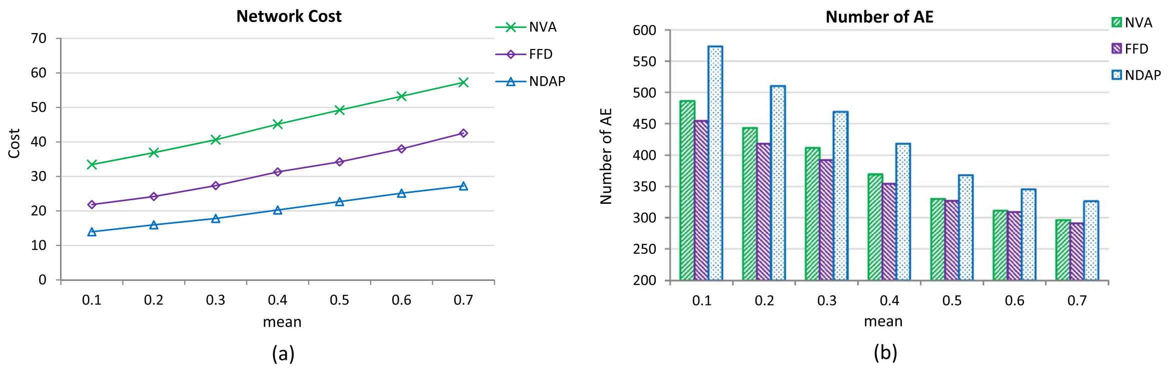


Fig. 9. Performance with increasing mean (homogeneous): (a) Network cost and (b) Number of AE deployed in DC.

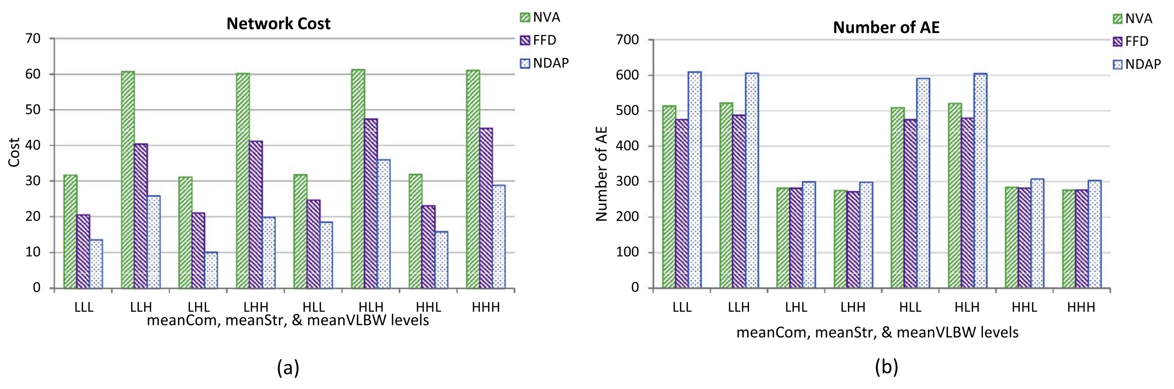


Fig. 10. Performance with mixed levels of means (heterogeneous): (a) Network cost and (b) Number of AE deployed in DC.

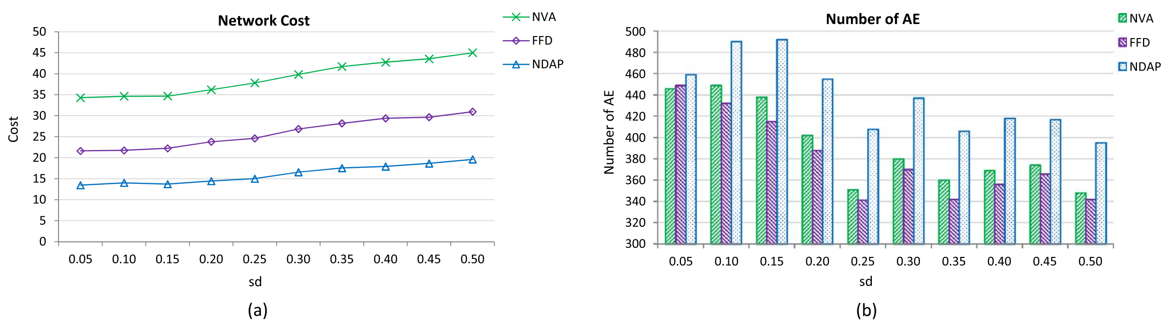


Fig. 11. Performance with increasing standard deviation of resource demands: (a) Network cost and (b) Number of AE deployed in DC.

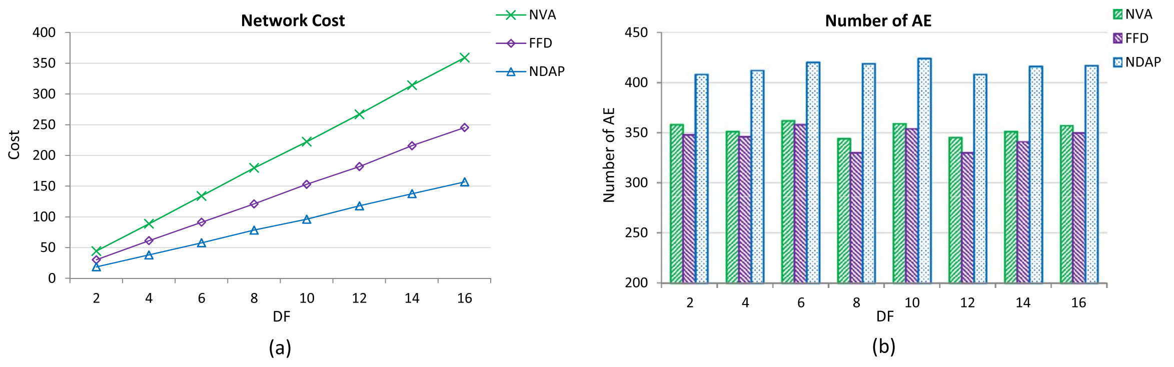


Fig. 12. Performance with increasing distance factor DF: (a) Network cost and (b) Number of AE deployed in DC.

meanVLBW, respectively. As the chart shows, NDAP performs much better in terms of incurred cost compared than the other algorithms for each of the mean combinations. Its relative performance is highest for combinations *LHL* and *LHH* incurring on average 67% and 52% less costs compared to NVA and FFD; whereas its performance is lowest for combinations *HLL* and *HLH* incurring on average 42% and 25% less costs compared to NVA and FFD, respectively. The reason behind this pattern is the algorithmic flow of NDAP as it starts with the VDLs placement and finishes with the remaining VCLs placement. As a consequence, for relatively higher mean of DB storage demands, NDAP relatively performs better.

A similar performance trait can be seem in Fig. 10(b) that shows that NDAP places more AEs in DC compared to other algorithms. An overall pattern demonstrated by the figure is that when the *meanStr* is high (*H*), the number of AEs deployed is reduced for all algorithms compared to the cases when *meanStr* is low (*L*). This is because the simulated storage resources are fewer compared to the computing and network resources of DC with respect to the storage, computing, and bandwidth demands of AEs, respectively. Since NDAP starts AE deployment with efficient placement of DBs and VDLs, on average it deploys 17% and 26% more AEs compared to NVA and FFD, respectively, when $meanStr = H$; whereas this improvement is 9% for both NVA and FFD when $meanStr = L$.

5.5. Diversification of workloads

This part of the experiment simulates the degree of workload diversification of the deployed AEs through varying the standard deviation of the random (normal) number generator used to generate the resource demands of the components of AEs. For this purpose, the initial value for *sd* parameter is set to 0.05 and increased gradually by adding 0.05 at each simulation phase until a maximum of 0.5 is reached. The other parameters *N*, *meanCom*, *meanStr*, *meanVLBW*, and *DF* are set to 1152, 0.3, 0.4, 0.35, and 2, respectively.

As shown in Fig. 11(a), the average network cost for NDAP is much lower compared to the other algorithms when the same number of AEs are deployed (as the simulation terminates when any of the algorithms fail to deploy an AE in the group scenario) as, on average, it incurs 61% and 38% less cost compared to NVA and FFD, respectively. Moreover, for each algorithm, the cost increases with the increase of workload variations. This is due to the fact that for higher variation in resource demands, the algorithms experience reduced scope in the data center for AE components placement as the feasibility domain is shrunk. As a consequence, feasible placements incur increasingly higher network cost with the increase of *sd* parameter.

In the individual scenario, NDAP outperforms other algorithms in terms of the number of AEs deployed across various workload variations (Fig. 11(b)) by successfully placing on average 12% and 15% more AEs compared to NVA and FFD, respectively. Due to the random placement component, overall NVA performs better compared to FFD which is deterministic by nature. Another general pattern noticeable from the chart is that, all the algorithms deploy more AEs for lower values of *sd*. This is due the fact that for higher value of *sd*, resource demands of the AE components demonstrate higher variations and, as a consequence, resources of data center components get more fragmented during the AE placements and, thus, the utilization of those resources get reduced.

5.6. Scaling network distances

This experiment varies the relative network distance between any two data center nodes by scaling the *DF* parameter defined in Section 5.2.1. As the definition implies, the inter-node network distance increases with *DF* and such situation can arise due to higher delays in network switches or due to geographical distances. Initially, the *DF* value is set to 2 and increased upto 16. Other parameters *N*, *meanCom*, *meanStr*, *meanVLBW*, and *sd* are set to 1152, 0.3, 0.4, 0.35, and 0.5, respectively.

Since network distance directly contributes to the cost function, it is evident from Fig. 12(a) that the placement cost rises with the increase of the *DF* parameter in a linear fashion for the group scenario. Nevertheless, the gradients for the different placement algorithms are not the same and the rise in cost for NDAP is much lower than other algorithms.

Fig. 12(b) shows the average number of AEs deployed in data center for each *DF* values for the individual scenario. Since network distance does not contribute to any of the resource capacities or demands (e.g., CPU or bandwidth), the number of AE deployment remains mostly unchanged with the scaling of *DF*. Nonetheless, through efficient placement, NDAP outpace other algorithms and successfully deploys 18% and 21% more AEs than NVA and FFD, respectively.

5.7. Network utilization

This part of the experiment is conducted for the purpose of comparing the network utilization of the placement algorithms at the access, aggregation, and core switch levels of the data center network. This is done by stressing the network in two different scaling factors separately: the mean and the standard deviation of VLs bandwidth demand *meanVLBW* and *sdVLBW*, respectively. In order to ensure that the computing and storage resource demands (of VMs and DBs, respectively) do not stress the computing and storage resource capacities (of the CNs and SNs, respectively), the *meanCom* and *meanStr* parameters are kept at a fixed small value of 0.05, and the standard deviation *sdComStr* for both computing and storage resource demands is set to 0.1. As for the other parameters, *N* and *DF* are set to 1152 and 2, respectively.

5.7.1. Scaling mean bandwidth demand

Here the data center network is stressed using the group scenario where all the algorithms terminate if any of the algorithms fail to place an AE. Application of the group scenario for this experiment ensures that the total network loads imposed for each of the placement algorithms are same when any of the algorithms fail. Initially, the mean VL bandwidth demand *meanVLBW* is set to 0.1 and raised upto 0.7, each time increasing it by 0.1. Standard deviation of VL bandwidth demand *sdVLBW* is kept fixed at 0.3.

Fig. 13 shows the average network utilization of the access, aggregation, and core switches for different *meanVLBW* values. It is evident from the charts that, for all the switch levels, NDAP incurs minimum average network utilization and compared to NVA and FFD, NDAP placements on average result in 24% and 16% less network usage for access layer, 49% and 30% less network usage for aggregation layer, and 83% and 75% less network usage for core layer. This represents the fact that NDAP localizes network traffic more efficiently compared to other algorithms and achieves incrementally higher network efficiency at access, aggregation, and core switch levels. Furthermore, as the figure demonstrates, the results reflect a similar trend of performance as the results of average network cost for

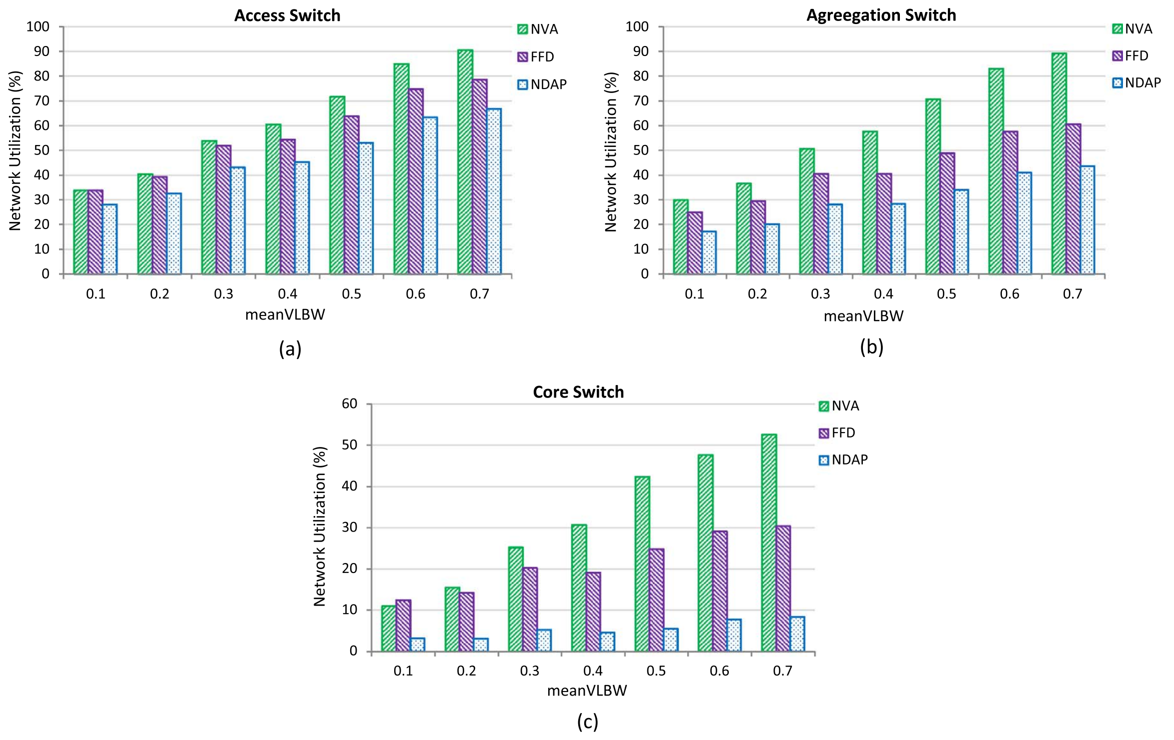


Fig. 13. Average network utilization with increasing mean VL bandwidth demand: (a) Access switch, (b) Aggregation switch, and (c) Core switch.

placement algorithms presented in the previous subsections. This is reasonable since network cost is proportional to the distance and bandwidth of the VLs and while placing a VL, higher network distance indicates the use of higher layer switches. Thus, these results validate the proposed network cost model (Eq. (1) & (2)) in the sense that indeed it captures the network load perceived by the network switches. Also, it can be observed that the utilization for each switch increases with increasing *meanVLBW*. This is due to the fact that *meanVLBW* contributes to the average amount of data transferred through the switches since *meanVLBW* is used as the mean to generate the VLs bandwidth demands.

5.7.2. Diversification of bandwidth demand

This experiment is similar to the above one, however, here the standard deviation of VLs bandwidth demands (*sdVLBW*) is scaled rather than the mean— initially, *sdVLBW* is set to 0.05 and gradually increased upto 0.5, each time raising it by 0.05. The mean VLs bandwidth demand *meanVLBW* is set to 0.4.

The results of this experiment is shown in Fig. 14. The charts clearly demonstrate the supervisor performance of NDAP that causes minimum network usage across all switch levels and compared to NVA and FFD, it has on average 26% and 16% less network usage for access layer, 50% and 30% less network usage for aggregation layer, and 84% and 75% less network usage for core layer. Furthermore, the figure shows that the network utilizations for each algorithm at each layer across different *sdVLBW* values do not fluctuate much. This is due to the fact that although the variation of VLs' bandwidth demand increases with increasing *sdVLBW*, the overall network load levels do not change much and, as a result, the average network loads perceived by the network switches at different layers differ in a small range.

5.8. NDAP decision time

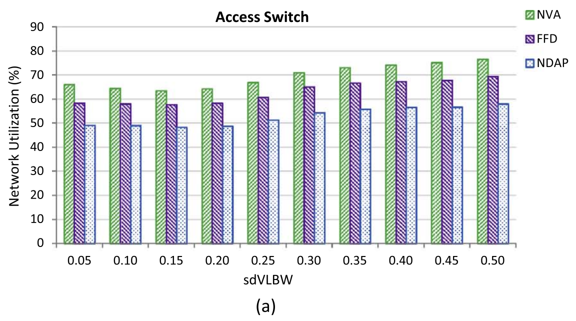
In this part of the experiment, the time taken by NDAP for making AE placement decision is measured in order to assess the feasibility of using NDAP for real-time, on-demand placement scenarios. Fig. 15 shows the average time needed by NDAP for computing AE placements in the individual scenario by scaling all the above mentioned scaling factors. For each of the scaling factors, other parameters are set similar to the corresponding preceding sections (Section 5.3–5.6).

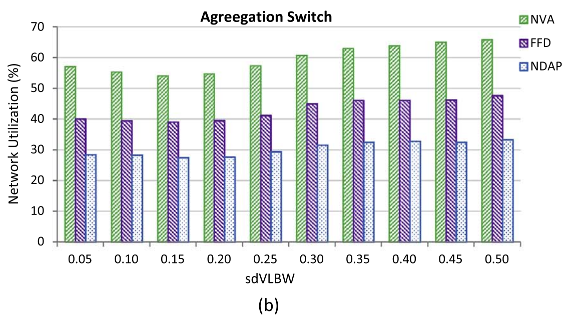
From Fig. 15(a), it can be seen that NDAP's run time increases non-linearly with increasing data center size (other parameters are set as in Section 5.3). It is evident from the figure that for small to medium data centers, NDAP's decision making time is in the range of a small fraction of a second, whereas for the largest data center simulated with $N = 4608$ (i.e., several thousand servers), NDAP needs only about 0.7 s. Furthermore, it can be observed from Fig. 15(b)-(e) that NDAP's run time remains largely unaffected by other scaling factors and the run time is within the range of 0.03–0.06 s for $N = 1152$. From the above results and discussion, it can be concluded that NDAP is suitable for on-demand AE placement scenarios, even for large data centers.

6. Conclusions and future work

With the growing complexity of modern Internet applications, as well as size of data, network resource demands in large data centers, such as Clouds, are becoming increasingly complex. Rising bandwidth requirements among application components are causing rising pressure on the underlying communication infrastructure and, thus, make it a key area of performance bottleneck. This paper addressed the issue of network-focused, multi-component application placement in large data centers and formally defined it as an optimization problem. After presenting the constitutional components of the proposed network- and data-aware application placement approach, it presented NDAP, a greedy heuristic that performs simultaneous deployment of VMs and data components respecting computing, network, and storage resource requirements and capacity constraints with the goal of minimizing the network cost incurred due to the placement decision. Moreover, a detailed analysis on the computational complexity for each of the compared algorithms in terms of run-time and memory overhead is presented. It is revealed from the analysis that the proposed NDAP algorithm has quadratic time complexity, which is slightly higher than those of the compared algorithms (linear and linearithmic), and the memory overheads are same for each of the algorithms. Furthermore, performance of the proposed NDAP algorithm is compared with related works through extensive simulation-based experiments and the results demonstrate superior performance of NDAP over competitor approaches across multiple performance metrics.

The proposed NDAP algorithm is developed as a generic application placement heuristic, which is very application-unaware, so that it can





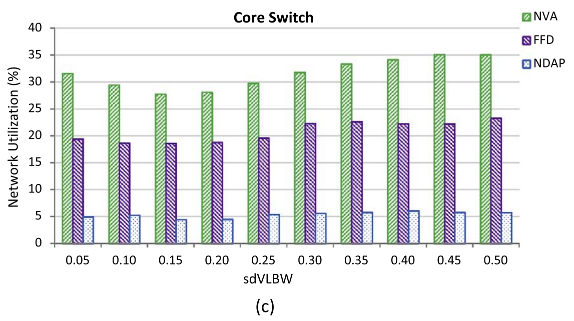


Fig. 14. Average network utilization with increasing standard deviation of VL bandwidth demand: (a) Access switch, (b) Aggregation switch, and (c) Core switch.

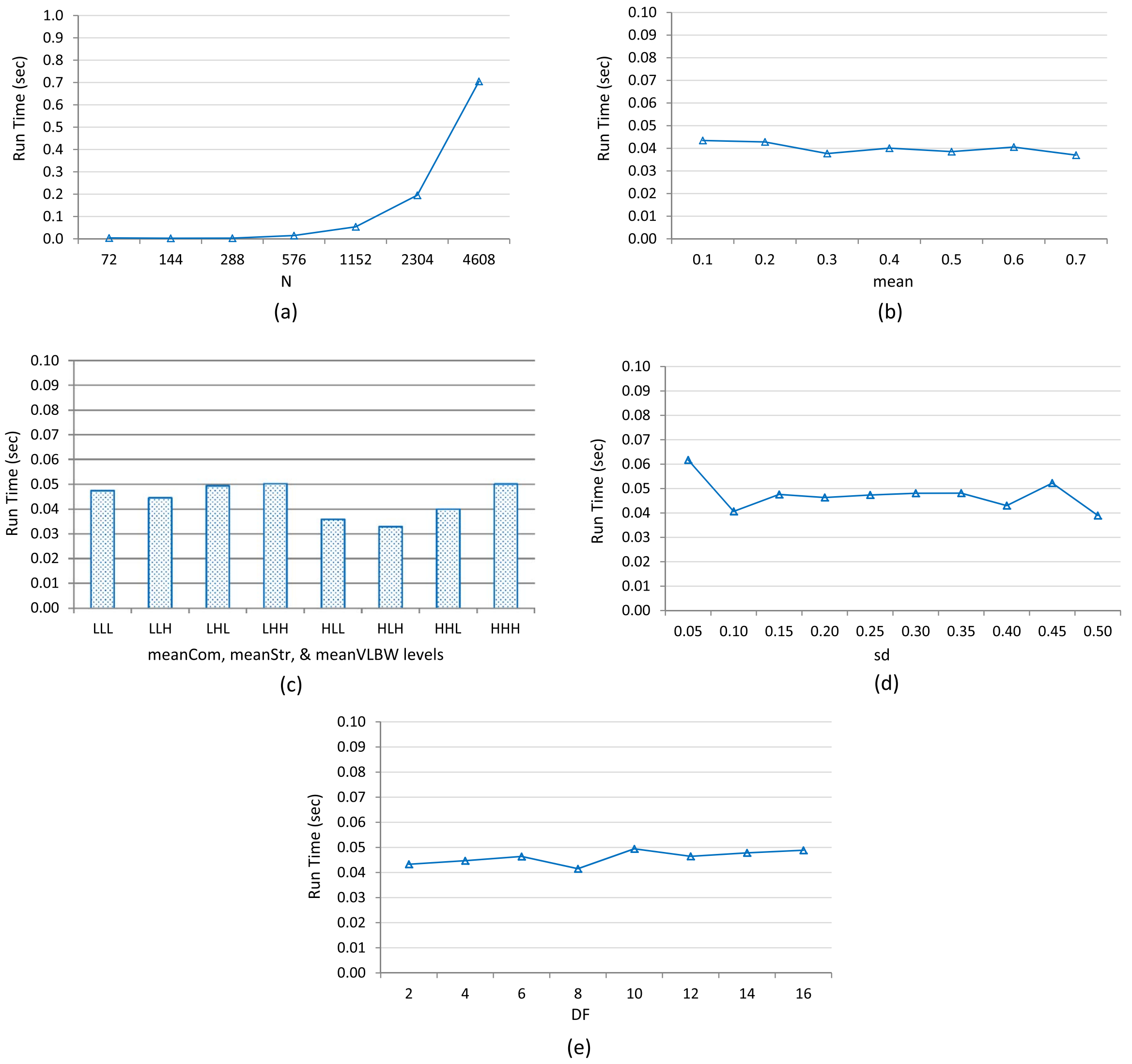


Fig. 15. NDAP's placement decision time while scaling (a) Data center size (*N*), (b) Homogeneous mean (*mean*), (c) Heterogeneous mean (*meanCom*, *meanStr*, *meanVLBW*), (d) Diversification of workload (*sd*), and (e) Distance factor (*DF*).

be used for a wide range of multi-tier or composite applications. The effectiveness of NDAP is validated in minimizing overall communications overhead for two different representative application environments: multi-tier and scientific (Montage) workflows. A further optimization of application placement, considering application-aware communication models and specific structure for application components, can be a potential future work. Moreover, the proposed NDAP strategy is flexible enough to be extended in future to accommodate such special cases.

Widespread use of virtualization technologies, high speed communication, increased size of data and data centers, and, above all, the broad spectrum of modern applications are opening new research challenges in network resource optimization. Appropriate combination and coordination of the online and offline VM placement and migration techniques with the goal of efficient network bandwidth management is one of the key areas for future research.

Furthermore, periodic and threshold-based reconfiguration of application virtual components using VM migration and reallocation with focus on network traffic localization can be an effective strategy for data center resource optimization. Such online or real-time optimization techniques must employ appropriate resource demand forecasting for both computing (e.g., CPU utilization) and network resources (e.g., bandwidth usage). In addition, VM migration and reconfiguration overhead can have significant impact on the performance of the hosted applications, as well as on the underlying communication substrate, consequently, questioning the scalability and viability of the offline strategies that often consider simplistic measures for VM migration overheads. Incorporation of reconfiguration overhead estimation with offline optimization techniques focusing on multiple objectives will produce more pragmatic VM migration schemes trading off between resource usage optimization and incurred overhead.

References

Alboaneen, D.A., Pranggono, B., Tianfield, H., 2014. Energy-aware virtual machine consolidation for cloud data centers. In: Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing. IEEE Computer Society, pp. 1010–1015.

Alharbi, Y., Walker, S., 2016. Data intensive, computing and network aware (dcn) cloud vms scheduling algorithm. In: Future Technologies Conference (FTC). IEEE, pp. 1257–1264.

Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A., Buyya, R., 2015. Big Data computing and clouds: trends and future directions. J. Parallel Distrib. Comput. 79, 3–15.

Beloglazov, A., 2013. Energy-effcient management of virtual machines in data centers for cloud computing. Ph.D. thesis. Department of Computing and Information Systems, The University of Melbourne.

Beloglazov, A., Buyya, R., 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of Virtual

Machines in Cloud data centers. Concurr. Comput.: Pract. Exp. 24 (13), 1397–1420.

Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E., 2012. A stable network-aware VM placement for Cloud systems. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012). IEEE Computer Society, pp. 498–506.

Burkard, R.E., Cela, E., Pardalos, P.M., Pitsoulis, L.S., 1998. The quadratic assignment problem. In: Handbook of Combinatorial Optimization. Springer, pp. 1713–1809.

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., 2009. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25 (6), 599–616.

Calcavecchia, N.M., Biran, O., Hadad, E., Moatti, Y., 2012. VM placement strategies for Cloud scenarios. In: Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD 2012). IEEE, pp. 852–859.

Cisco, 2015. Cisco Global Cloud Index: Forecast and Methodology, 2014–2019. [online] ⟨http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf⟩, (Accessed 30 August 2016).

Cormen, T., Leiserson, C., Rivest, R., Stein, C., 2001. Introduction to Algorithms. MIT Press.

Corradi, A., Fanelli, M., Foschini, L., 2014. Vm consolidation: a real case based on openstack cloud. Future Gener. Comput. Syst. 32, 118–127.

Cui, L., Tso, F.P., Pezaros, D.P., Jia, W., Zhao, W., 2017. Plan: joint policy-and network-aware vm management for cloud data centers. IEEE Trans. Parallel Distrib. Syst. 28, 1163–1175.

Dias, D.S., Costa, L.H.M., 2012. Online traffic-aware Virtual Machine placement in data center networks. In: Proceedings of the 2012 Global Information Infrastructure and Networking Symposium (GIIS). IEEE, pp. 1–8.

Farahnakian, F., Ashraf, A., Liljeberg, P., Pahikkala, T., Plosila, J., Porres, I., Tenhunen, H., 2014. Energy-aware dynamic VM consolidation in Cloud data centers using Ant Colony S ystem. In: Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014). IEEE, pp. 104–111.

Farahnakian, F., Ashraf, A., Pahikkala, T., Liljeberg, P., Plosila, J., Porres, I., Tenhunen, H., 2015. Using ant colony system to consolidate vms for green cloud computing. IEEE Trans. Serv. Comput. 8 (2), 187–198.

Feller, E., Rilling, L., Morin, C., 2011. Energy-aware Ant Colony based workload placement in Clouds. In: Proceedings of the 12th IEEE/ACM International Conference on Grid Computing (GRID 2011). IEEE Computer Society, pp. 26–33.

Ferdaus, M.H., Murshed, M., 2014. Cloud Computing: Challenges, Limitations and R & D Solutions. Computer Communications and Networks. Ch. Energy-Aware Virtual Machine Consolidation in IaaS Cloud Computing. Springer International Publishing, 179–208.

Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R., 2014. Virtual Machine consolidation in Cloud data centers using ACO metaheuristic. In: Proceedings of the 20th International European Conference on Parallel and Distributed Computing (Euro-Par 2014). Springer International Publishing, pp. 306–317.

Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R., 2015. Emerging Research in Cloud Distributed Computing Systems. Ch. Network-aware Virtual Machine Placement and Migration in Cloud Data Centers. IGI Global, 42–91 ⟨http://www.igi-global.com/chapter/network-aware-virtual-machine-placement-and-migration-in-cloud-data-centers/130268⟩.

Gao, C., Wang, H., Zhai, L., Gao, Y., Yi, S., 2016. An energy-aware ant colony algorithm for network-aware virtual machine placement in cloud computing. In: Parallel and Distributed Systems (ICPADS), 2016 IEEE Proceedings of the 22nd International Conference on. IEEE, pp. 669–676.

Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L., 2013. A multi-objective Ant Colony System Algorithm For Virtual Machine Placement In Cloud computing. J. Comput. Syst. Sci. 79 (8), 1230–1242.

Georgiou, S., Tsakalozos, K., Delis, A., 2013. Exploiting network-topology awareness for VM placement in IaaS Clouds. In: Proceedings of the 3rd International Conference on Cloud and Green Computing (CGC 2013). IEEE, pp. 151–158.

Gupta, A., Kale, L.V., Milojicic, D., Faraboschi, P., Balle, S.M., 2013. HPC-aware VM placement in infrastructure Clouds. In: Proceedings of the 2013 IEEE International Conference on Cloud Engineering (IC2E). IEEE, pp. 11–20.

Huang, D., Gao, Y., Song, F., Yang, D., Zhang, H., 2013. Multi-objective Virtual Machine migration in virtualized data center environments. In: Proceedings of the 2013 IEEE International Conference on Communications (ICC). IEEE, pp. 3699–3704.

Huang, D., Yang, D., Zhang, H., Wu, L., 2012. Energy- aware Virtual Machine placement in data centers. In: Proceedings of the 2012 IEEE Conference on Global Communications (GLOBECOM). IEEE, pp. 3243– 3249.

Juniper, June 2012. Cloud-ready Data Center Reference Architecture. [online] ⟨http://www.juniper.net/us/en/local/pdf/brochures/1600040-en.pdf⟩, (Accessed 30 August 2016).

Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K., 2013. Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. 29 (3), 682–692.

Kakadia, D., Kopri, N., Varma, V., 2013. Network-aware Virtual Machine consolidation for large data centers. In: 2013 Proceedings of the Third International Workshop on Network-Aware Data Management (NDM 2013). ACM, pp. 6:1-6:8.

Kliazovich, D., Bouvry, P., Khan, S.U., 2013. DENS: data center energy-efficient network-aware scheduling. Clust. Comput. 16 (1), 65–75.

Korupolu, M., Singh, A., Bamba, B., 2009. Coupled placement in modern data centers. In: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS). IEEE, pp. 1–12.

Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N., Jiang, G., 2009. Power and performance management of virtualized computing environments via lookahead control. Clust. Comput. 12 (1), 1–15.

Li, X., Qian, C., 2015. Traffic and failure aware vm placement for multi-tenant cloud computing. In: Quality of Service (IWQoS), 2015 IEEE Proceedings of the 23rd International Symposium on. IEEE, pp. 41–50.

Liu, H., Jin, H., Xu, C.-Z., Liao, X., 2013. Performance and energy modeling for live migration of Virtual Machines. Clust. Comput. 16 (2), 249–264.

Lloyd, W., Pallickara, S., David, O., Arabi, M., Rojas, K., 2014. Dynamic scaling for service oriented applications: Implications of Virtual Machine placement on IaaSClouds. In: Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E). IEEE, pp. 271– 276.

Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T., 2007. A survey for the quadratic assignment problem. Eur. J. Oper. Res. 176 (2), 657–690.

Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., Pendarakis, D., 2010a. Efficient resource provisioning in compute clouds via VM multiplexing. In: Proceedings of the 7th International Conference on Autonomic Computing (ICAC 2010). ACM, pp. 11–20.

Meng, X., Pappas, V., Zhang, L., 2010b. Improving the scalability of data center networks with traffic-aware Virtual Machine placement. In: Proceedings of the 29th IEEE Conference on Computer Communication (INFOCOM 2010). IEEE, pp. 1–9.

Mysore, R.N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A., 2009. Portland: A scalable fault-tolerant layer 2 data center network fabric. In: Proceedings of the 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)., 9. pp. 39–50.

Nguyen, T.H., Di Francesco, M., Yla-Jaaski, A., 2014. A multi-resource selection scheme for Virtual Machine consolidation in Cloud data centers. In: Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014). IEEE, pp. 234–239.

Piao, J.T., Yan, J., 2010. A network-aware Virtual M achine placement and migration approach in Cloud computing. In: Proceedings of the 9th International Conference on Grid and Cooperative Computing (GCC 2010). IEEE, pp. 87–92.

Sahni, S., Gonzalez, T., 1976. P-complete approximation problems. J. ACM (JACM) 23 (3), 555–565.

Sahu, S., Gupta, H., Singh, S., Ghosh, S.K., 2014. A demand based resource provisioner for cloud infrastructure. In: Advanced Computing, Networking and Informatics-Volume 2. Springer, pp. 425–432.

Shrivastava, V., Zerfos, P., Lee, K.-w., Jamjoom, H., Liu, Y.-H., Banerjee, S., 2011. Application-aware Virtual Machine migration in data centers. In: Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM 2011).IEEE, pp. 66–70.

Song, F., Huang, D., Zhou, H., You, I., 2012. Application-aware Virtual Machine placement in data centers. In: Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012). IEEE, pp. 191–196.

Srirama, S.N., Ostovar, A., 2014. Optimal resource provisioning for scaling enterprise applications on the Cloud. In: Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2014). IEEE, pp. 262–271.

Takouna, I., Rojas-Cessa, R., Sachs, K., Meinel, C., 2013. Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers. In: Proceedings of the 6th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2013). IEEE, pp. 251–255.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., Tantawi, A., 2005. An analytical model for multi-tier internet services and its applications. In: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems., 33, ACM, pp. 291–302.

Vu, H.T., Hwang, S., 2014. A traffic and power-aware algorithm for Virtual Machine placement in Cloud data center. Int. J. Grid Distrib. Comput. 7 (1), 350–355.

Wang, R., Wickboldt, J.A., Esteves, R.P., Shi, L., Jennings, B., Granville, L.Z., 2016. Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters. IEEE Trans. Netw. Serv. Manag. 13 (2), 267–280.

Wang, S., Gu, H., Wu, G., 2013. A new approach to multi- objective Virtual Machine placement in virtualized data center. In: Proceedings of the 8th IEEE International Conference on Networking, Architecture and Storage (NAS 2013). IEEE, pp. 331–335.

Wang, S.-H., Huang, P. P.-W., Wen, C.H.-P., Wang, L.-C., 2014. EQVMP: Energy-efficient and qos-aware Virtual Machine placement for software defined datacenter networks. In: Proceedings of the 2014 International Conference on Information Networking (ICOIN). IEEE, pp. 220– 225.

Wu, Q., Ishikawa, F., 2015. Heterogeneous virtual machine consolidation using an improved grouping genetic algorithm. In: Proceedings of the 17th IEEE International Conference on High Performance and Communications (HPCC 2015). IEEE, pp. 397–404.

Zhang, B., Qian, Z., Huang, W., Li, X., Lu, S., 2012. Minimizing communication traffic in data centers with power-aware VM placement. In: Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012). IEEE, pp. 280–285.

Zhang, Q., Cheng, L., Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. 1 (1), 7–18.

Zhang, W., Han, S., He, H., Chen, H., 2016. Network-aware virtual machine migration in an overcommitted cloud. Future Generation Computer Systems.

Zhao, Y., Huang, Y., Chen, K., Yu, M., Wang, S., Li, D., 2015. Joint vm placement and topology optimization for traffic scalability in dynamic datacenter networks. Comput. Netw. 80, 109–123.

Md Hasanul Ferdaus is currently working as a Research Assistant in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne (Australia). He has obtained his PhD degree in Computer Science from the Faculty of Information Technology, Monash University (Australia) in 2016, MS degree in Information and Communication Technology from Politecnico di Torino (Italy) and Karlsruhe Institute of Technology, (Germany) in 2009 under European Union (Erasmus Mundus) scholarship program, and his BSc (Hons) degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (Bangladesh) in 2004. He served as a Teaching Associate in the Faculty of IT, Monash University (Australia) from 2012 to 2016, as a Lecturer in the department of Computer Science and Engineering, American International University-Bangladesh from 2010 to 2011, as a R&D Assistant in the FZI Research Center for Information Technology (Germany) in 2008 and in the Telematics Institute, KIT (Germany) in 2009, and as a System Analyst in Robi Axiata Limited (Bangladesh) from 2005 to 2006. His research interest is focused on Cloud Computing, Distributed and Parallel Computing, Software Engineering, and Middleware Systems.

Manzur Murshed received the BScEngg (Hons) degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 1994 and the PhD degree in computer science from the Australian National University (ANU), Canberra, Australia, in 1999. He also completed his Postgraduate Certificate in Graduate Teaching from ANU in 1997. He is currently an Emeritus Professor Robert HT Smith Professor and Personal Chair at the Faculty of Science and Technology, Federation University Australia. Prior to this appointment, he served the School of Information Technology, Federation University Australia as the Head of School, from January 2014 to July 2014, the Gippsland School of Information Technology, Monash University as the Head of School 2007 to 2013. He was one of the founding directors of the Centre for Multimedia Computing, Communications, and Applications Research (MCCAR). His major research interests are in the fields of video technology, information theory, wireless communications, distributed computing, and security & privacy. He has so far published 181 refereed research papers with 2,625 citations as per Google Scholar and received more than $1M nationally competitive research funding, including three Australian Research Council Discovery Projects grants in 2006, 2010, and 2013 on video coding and communications, and a large industry grant in 2011 on secured video conferencing. He has successfully supervised 19 and currently supervising 6 PhD students. He is an Editor of International Journal of Digital Multimedia Broadcasting and has had served as an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology in 2012 and as a Guest Editor of special issues of Journal of Multimedia in 2009-2012. He received the Vice-Chancellor's Knowledge Transfer Award (commendation) from the University of Melbourne in 2007, the inaugural Early Career Research Excellence award from the Faculty of Information Technology, Monash University in 2006, and a University Gold Medal from BUET in 1994. He is a Senior Member of IEEE.

Rodrigo N. Calheiros is a Lecturer in the School of Computing, Engineering and Mathematics, Western Sydney University, Australia. He works in the field of Cloud computing and related areas since 2008, and since them he carried out R&D supporting research in the area. His research interests also include Big Data, Internet of Things, Fog Computing, and their application.

Dr. Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012-2016. He has authored over 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He also edited several books including "Cloud Computing: Principles and Paradigms" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index=112, g-index=235, 63,300+ citations). Microsoft Academic Search Index ranked Dr. Buyya as #1 author in the world (2005-2016) for both field rating and citations evaluations in the area of Distributed and Parallel Computing. "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing. Recently, Dr. Buyya is recognized as a "2016 Web of Science Highly Cited Researcher" by Thomson Reuters and a Fellow of IEEE for his outstanding contributions to Cloud computing. Software technologies for Grid and Cloud computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Dr. Buyya has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of Dr. Buyya are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Frost & Sullivan New Product Innovation Award". Recently, Dr. Buyya received "Bharath Nirman Award" and "Mahatma Gandhi Award" along with Gold Medals for his outstanding and extraordinary achievements in Information Technology field and services rendered to promote greater friendship and India-International cooperation. He served as the founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience, which was established over 45 years ago. For further information on Dr.Buyya, please visit his cyberhome: http://www.buyya.com