This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2694838, IEEE Transactions on Services Computing

Availability-aware Virtual Cluster Allocation in Bandwidth-Constrained Datacenters

Jialei Liu, Shangguang Wang, *Senior, Member, IEEE*, *Ao Zhou, Fangchun Yang, and Rajkumar Buyya, Fellow, IEEE*

Abstract—As greater numbers of data-intensive applications are required to process big data in bandwidth-constrained datacenters with heterogeneous physical machines (PMs) and virtual machines (VMs), network core traffic is experiencing rapid growth. The VMs of a virtual cluster (VC) must be allocated as compactly as possible to avoid bandwidth-related bottlenecks. Since each PM/switch has a certain failure probability, a VC may not be executed when it meets with any PM/switch fault. Although the VMs of a VC can be spread out across different fault domains to minimize the risk of violating the availability requirement of the VC, this increases the network core traffic. Therefore, avoiding the decrease in availability caused by the heterogeneous PM/switch failure probabilities and bandwidth-related bottlenecks has been a constant challenge. In this paper, we first introduce a joint optimization function to measure the overall risk cost and overall bandwidth usage in the network core to allocate the same set of data-intensive applications. We then introduce an approach to maximize the value of the joint optimization function. Finally, we performed a side-by-side comparison with prior algorithms, and the experimental results show that our approach outperforms the other existing algorithms.

Index Terms—Bandwidth-constrained datacenter, virtual cluster, availability, bandwidth-related bottleneck, risk, fault domain

1 Introduction

With the growing popularity of cloud computing, datacenters have become common platforms for supporting data-intensive applications using modern distributed computing frameworks, e.g., Spark, MapReduce, and MPI. In such frameworks, a data-intensive application is often processed by a virtual cluster (VC) which is composed of virtual switch, virtual links, and virtual machines (VMs) connected through virtual switch and virtual links with guaranteed bandwidth (as shown in Fig. 2), and its intermediate results are transferred iteratively through multiple stages [1]. Further, the flows not only between VMs but also into the Internet are created by the traffic generated by the application [2], e.g., existing study [3] shows that traffic between VMs in a typical Internet datacenter accounts for about 80% of its total traffic. Therefore, a significant portion of the running time of a data-intensive application is necessary for communication [4], for example, job traces from Facebook reveal that network transfers on average account for 33% of the running time of jobs [1], which can have a significant impact on job performance.

Due to the increase in data-intensive applications required to process big data in cloud datacenters, cloud datacenter traffic is experiencing rapid growth. As Cisco predicted, there will be nearly a tripling of global datacenter IP traffic from 2015 to 2020 with a combined annual growth rate of 27%, that is, from 4.7 ZB/year in 2015 to 15.3 ZB/year in 2020 [5]. Therefore, numerous data-intensive applications consume mass bandwidth resources in the network core of cloud datacenters. In this case, the bandwidth resources in the network core are very easy to become the bandwidth resource bottleneck of the cloud datacenter [6]. Further, traffic interference results in unpredictable running times, which can result in a degradation in performance experienced by end-users due to service unavailability as well as losses to the business, both in terms of immediate revenue and long-term reputation.

It is well known that modern-day cloud datacenters mount hundreds of thousands of physical machines (PMs) interconnected via a mass of switches, which communicate and coordinate tasks to deliver highly available cloud computing services. Service providers typically have specific requirements for their VCs, with certain amounts of resource guarantees (e.g., VMs and bandwidth) [7]. However, failures in cloud datacenter elements (e.g., switches and PMs) have severe impacts on the availability of cloud services; in particular, Top-of-Rack (ToR) switches account for the majority (approximately more than 60%) of the downtime in datacenters [6]. According to a study by the Ponemon institute in 2016 [8], the median total cost associated with unplanned outages is ＄648,174 per unplanned incident for those users who expect to meet their availability requirements which is service uptime divided by the sum of service uptime and service downtime [9], by demanding a more stringent Service Level Agreement.

- *J. Liu is with the State Key Laboratory of Networking and Switching technology, Beijing University of Posts and Telecommunications, Beijing, China and the Department of Computer Science and Information Engineering, Anyang Institute of Technology, Anyang, China. E-mail:jlliu22@163.com.*
- *S.Wang, A.Zhou and F. Yang are with the State Key Laboratory of Networking and Switching technology, Beijing University of Posts and Telecommunications Beijing, China. E-mail: {sgwang;aozhou;fcyang@bupt.edu.cn}*
- *R. Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Australia. Email: rbuyya@unimelb.edu.au.*

An intuitive way to improve the availability of VC allocation is to spread out the VMs of the VC to as many fault domains (i.e., racks) as possible. In this way, the impact of any single failure on the VC is minimized, but this comes at the price of bandwidth usage in the network core. In contrast, an alternative way to minimize bandwidth usage in the network core would be to accommodate these VMs close to each other so that the flows have shorter paths. The shorter the path, the lower the number of switches and links visited by these flows, which can decrease bandwidth usage in the network core. VC allocation benefits from the colocation; however, as a result, the entire VC is unavailable when a failure occurs at the exact PM or ToR switch.

Existing studies have introduced many fault-tolerant approaches to improving the availability of VCs, one strategy is to design new topologies with network redundancy, which provides rich path multiplicity to deliver large bisectional bandwidth, mitigating bandwidth resource bottlenecks [10]. However, these improvements are mainly due to the reduction of the median impact of failures via only 40% of network redundancy at the price of increasing capital expenditures, wiring complexity and energy consumption [6]. Many incumbent datacenter networks (DCNs) are under-provisioned with bandwidth, i.e., oversubscribed. In bandwidth-constrained datacenters, incoming VC requests may be rejected while free VMs are still available. Therefore, only the strategy above is not enough. Another strategy introduced in this paper analyzes the VC allocation problem from a novel angle. It not only considers the risk cost of violating the availability requirement of a data-intensive application due to heterogeneous failure probabilities of the ToR switch/PM, but it also exploits the sum of usage on the core links as an overall measure of the bandwidth usage in the network core; this measure is denoted by *BW*.

To solve these challenges, in this paper, we propose an Availability-aware VC Allocation (AVCA) approach with biogeography-based optimization (BBO) [11] that simultaneously minimizes the bandwidth usage in the network core and risk cost whereby each PM/ToR switch has a certain failure probability. To find a trade-off between the above two objectives, we separately measure the bandwidth usage in the network core and risk cost, and make a joint optimization of these two goals.

To summarize, the **key contributions** of our work are:

- We propose two novel models: One is to formulate the risk cost of violating the availability requirement of a VC while both a ToR switch and a PM have heterogeneous failure probabilities and fail concurrently; the other formulates the bandwidth usage in the network core of a VC.
- Based on the above two models, we first establish a joint optimization model to measure a risk cost and bandwidth usage in the network core of the VC allocation solution. Then, we introduce an AVCA with the BBO algorithm to maximize the joint optimization value, i.e., simultaneously minimize the overall risk cost and bandwidth usage in the network core.
- We build a system model to evaluate the performance of our approach. The experimental results show that our approach can achieve flexible balances between the overall risk cost and bandwidth usage in the network core.

Organization. Section 2 introduces the research background and related work. Section 3 introduces a system model, VC abstraction model, and motivation. Section 4 describes the technical details of our approach. Section 5 provides a performance evaluation, including an introduction to the experiment parameter configuration and comparison results. Section 6 presents the limitations of our approach and Section 7 concludes with research recommendations.

2 Background and Related Work

VC allocation, which is similar to virtual network embedding [2], [12], [13], [14], has attracted a great deal of attention in recent years. The growing demand for always-on data-intensive computing services has driven VC allocation solutions to be efficient in terms of virtual resource utilization (e.g., bandwidth) and availability of allocated VCs [2]. Thus, cloud datacenter failure characteristics have been analyzed in several recent studies, and the main finding is that these datacenters often contain heterogeneous equipment (e.g., PMs, switches) [15] with skewed distributions of failure rates, impact and repair time [3], [6], [16], [17]. In addition, availability-aware VC allocation approaches have also been introduced. Here, we briefly summarize the achievements in this field.

Heterogeneous failure probabilities of physical components. Viswanath et al. [16] first attempted to study PM failures and hardware repairs for large datacenters. They presented a detailed analysis of failure characteristics of 100,000 PMs across multiple Microsoft datacenters over a duration of 14 months. Their analysis yields the following results; 70% of all server failures are due to hard disks, 6% are due to the RAID controller, 5% are due to memory and the rest (18%) are due to other factors. Their reports also show that the number of PM failures is closely connected with the number of hard disks hosted in the PM. Furthermore a PM that has experienced a failure is highly likely to experience another failure in the near future. The above analysis results lead to a skewed distribution of PM failure probabilities. On the other hand, Gill et al. [6] presented the first large-scale analysis of failures in a DCN. Based on their analysis of multiple data sources commonly collected by network operators, several key findings are presented indicating that the failure rates are unevenly distributed; that is, the failure probabilities of different forms of network equipment can vary significantly based on type (PMs, ToR switches, aggregation switches, routers) and model. For example, Load Balancers have a more than 20% failure probability, whereas the ToR switches often have very low failure probability (i.e., less than 5%).

Heterogeneous impact and repair times of failures. Gill et al. [6] introduced the idea that although certain network failures can take up to seconds to fix, PM failures can be fixed within hours [6]. Wu et al. [17] proposed that although most network failures can be mitigated promptly using simple actions, certain failures can still cause significant network downtime. For instance, Greenberg et al. [3] collected failure logs for over a year from eight production datacenters. Their analysis shows that most failures are small in size (e.g., 50% of network device failures involve less than 4 devices and 95% of network device failures involve less than 20 devices) and that large correlated failures are rare. However, downtimes can be significant; i.e., 95% of failures are resolved in 10 min, 98% in less than 1 hr, 99.6% in less than 1 day, but 0.09% last more than 10 days. Similarly, Gill et al. [6] analyzed the correlations among link failures and found that more than 50% of link failures are single link failures, and that more than 90% of link failures involve less than 5 links.

Available VC Allocation. Since providing cloud service availability is significantly important in cloud datacenters, there is a trend toward designing the availability-aware VC allocation algorithm for VC requests. For example, Yeow et al. [18] proposed a technique for estimating the number of backup VMs required to achieve the desired availability objectives. However, they only assumed that PMs have identical failure probabilities. Xu et al. [19] introduced a resource allocation solution for provisioning virtual datacenters with backup VMs and links. However, their solution does not consider the availability of PMs. Bodik et al. [20] presented a detailed analysis of a large-scale Web application and its communication patterns. Based on this, they proposed and evaluated a novel optimization framework for improving service survivability while mitigating the bandwidth bottleneck in the core of the DCN. Their solution improves the fault tolerance by spreading out VMs across multiple fault domains while minimizing total bandwidth consumption. However, they did not consider the heterogeneous failure probabilities of the underlying physical equipment and the heterogeneous configuration of PMs and VMs. In addition, they only considered that a PM hosts a VM. Zhang et al. [12] presented a reliable VDC embedding framework that considers the availability aspect of embedding in terms of dependencies among virtual components and heterogeneous hardware failure rates. Yang et al. [21] not only considered concurrent PM and ToR switch failures but also the minimization of an overall cost that is based on energy consumption and the risk cost of violating the availability requirements. However, they did not consider bandwidth usage in the network core and the heterogeneous configuration of PMs and VMs and only considered that the failure probabilities of the PM or ToR switch were homogeneous. To avoid bandwidth related bottlenecks in the network core, C. da Silva et al. [22] introduced a topology-aware VM placement algorithm to use small regions of the DCN in order to consolidate the network flows produced by the communicating VMs. Meanwhile, Ho et al. [2] introduced an admission control mechanism to detect and rectify bandwidth-wasting VM placement via VM reshuffling as well as to accommodate newly arriving VC requests.

Unlike previous studies, our research not only considers concurrent PM/ToR switches with heterogeneous failure probabilities but also exploits heterogeneous PMs and VMs. In addition, we propose an approach using an original BBO algorithm to maximize the joint optimization value (i.e., simultaneously minimizing the overall risk cost and bandwidth usage in the network core) while processing a set of data-intensive applications.

3 Preliminaries and System Model

In this section, we first describe our system model. Then, we introduce a VC abstraction, which is allocated to a fat-tree DCN. Finally, we introduce our research motivation.

3.1 System Model

We build a system model based on a network topology, for example, fat-tree DCNs interconnected by *k*-port commodity Ethernet switches [10] (as shown in Fig. 1). The fat-tree DCN can be recursively constructed by building blocks (i.e., basic fat-tree networks) that consist of 2 tiers of switches. Since the results can be recursively applied, any properties of these building blocks are still held, explaining the scalability of fat-tree networks.

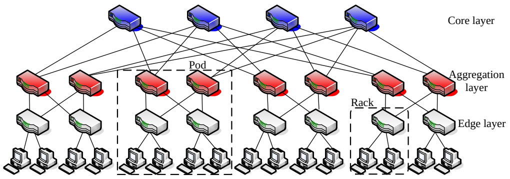


Fig. 1. Fat-tree DCN (the switches in the bottom (white), middle (red), and top (blue) tiers are the edge, aggregation, and core switches, respectively).

Fig. 1 shows a basic fat-tree DCN, which consists of three tiers of switch modules: edge switches, aggregation switches, and core switches. There are *k* pods, each containing two tiers of *k*/2 switches. Each *k*-port switch in an edge tier is an edge switch, which is directly connected to *k*/2 PMs. All PMs physically connected to the same edge switch are in the same rack (i.e., subnet). Since ToR switches account for the majority (roughly over 60%) of downtime in datacenters and it is unlikely to have largely correlated failures, each rack is referred to as an individual fault domain. Each of the remaining *k*/2 port is linked to a *k*/2 aggregation switch in the aggregation tier of the hierarchy. A pod consists of PMs, which share the same aggregation switches. The core tier contains $(k/2)^2$ *k*-port core switches, in which each core switch has one port linked to each *k* pod. Since the pod *i* connects *i-th* port of each core switch, there are the core switches on *(k/2)* strides connecting consecutive ports in the aggregation tier of each pod switch. That is, there are $k^3/4$ PMs in fat tree with *k*-port switches.

3.2 VC Abstraction

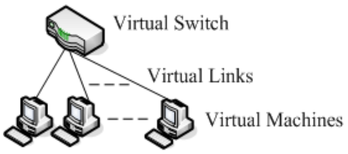


Fig. 2. Virtual cluster abstraction

The VC abstraction (as shown in Fig. 2) is the variant of the hose model, which was originally designed for VPNs [23]. In a hose model abstraction, all VM are connected to a central virtual switch by a dedicated link that has a minimum bandwidth guarantee. The authors in [7] propose Octopus, which includes one type of abstraction, i.e., VC, to expose tenants' virtual network requirements to cloud providers. The VC abstraction is designed for the all-to-all traffic pattern and assumes that a single non-oversubscribed virtual switch connects all VMs, such as MapReduce-like data-intensive applications.

When a data-intensive application is accepted, the application is processed by a VC, which is assigned by the datacenter and consists of a set of virtual links, a virtual switch, and multiple VMs connected by these virtual links. Please note that each virtual link owns the same fraction of link capacity, each virtual switch is mapped to multiple physical switches, each VM occupies *w* fraction of a PM. However, from the point of view of the user, the above details are invisible. That is, the network and PMs exploited have a very simple structure, in which each switch owns non-blocking capability and is connected by multiple private links connecting the PM.

3.3 Motivation

The virtual network embedding is known to be *NP-hard* [24], this is due to that it involves the complex mapping of multiple network component parts (e.g., VMs , virtual switch, and virtual links). Although the VC allocation of this paper does not consider the mapping of the virtual links and virtual switch, its complexity can be not reduced [7]. This is because that the mapping of VMs (i.e., VM placement) is often formulated as a variant of the vector bin-packing problem, which is a classic *NP-hard* optimization problem [25]. For example, a 3-tier web application is processed by three VMs of a VC, which are allocated to three PMs (i.e., a database PM, a web PM, and an application PM), if the web PM fails, the entire web application becomes unavailable regardless of whether the application and database PMs are available. Further, the above situation leads to the unavailability of future VC requests. In this paper, in view of the impact of the PM/ToR switch with heterogeneous failure probabilities on the VC allocation scheme, we research the VC allocation problem from a novel angle. That is, when we solve the VC allocation problem, whereby each PM/ToR switch has heterogeneous failure probability, we need to jointly minimize the overall risk cost and bandwidth usage in the network core. More specifically, we identify the joint minimum of the overall risk cost and bandwidth usage in the network core to accommodate these VCs. The only constraint is that the resources that an incoming VC requires, such as PMs, should be no more than the free resources in the datacenter. Finally, we propose an approach with the original BBO algorithm to solve the joint optimization problem.

4 Proposed AVCA Approach

In this section, we first introduce the details of near-optimal availability-aware VC allocation based on a general application model in Section 5.1 including the risk metric model, BW metric model, and near-optimal availability-aware VC allocation model. Then, we introduce the ecosystem model of the BBO algorithm. Finally, we present implementation scheme of our approach.

4.1 Near-Optimal Availability-aware VC Allocation

4.1.1 Risk Metric Model

Theorem 1. Since the PMs and ToR switches often have heterogeneous failure probabilities in cloud datacenters, we consider a simple situation (i.e., a PM failure and a ToR switch failure) to make the discussion clearer. The PM failure often results in all VMs hosted on the failed PM to fail, while the ToR switch failure leads to all VMs being accommodated in the corresponding rack and all PMs being inaccessible. Therefore, when the VC allocation algorithm is exploited to allocate these VCs, the selected priority of rack is higher than that of PM. That is, the allocation algorithm gives preference to the minimum ToR switch failure probability to accommodate them. We have the following results.

$$risk_i = \frac{\sqrt{\frac{1}{M-1}\sum_m(\sum_m P_{i,m}\delta_{i,m}^{B1} + \sum_m P_{i,m}\delta_{i,m}^{B2} - \delta_{i,m}^{B1} - \delta_{i,m}^{B2})^2}}{S_i - \sum_m P_{i,m}\delta_{i,m}^{B1} - \sum_m P_{i,m}\delta_{i,m}^{B2}} \tag{1}$$

s.t.

$$\delta_{i,m}^{B1} = \sum_n \sum_j \left(X_{i,j}^{m,n} \frac{\prod_{i=1}^{MN}(1-P_i)}{\prod_{i=1}^{MN}(1-P_i) + MNP_k \prod_{i=1,i\neq k}^{MN}(1-P_i)} \right) \tag{2}$$

$$\delta_{i,m}^{B2} = \left(\sum_n \sum_j X_{i,j}^{m,n} + \frac{M-1}{M} \sum_m \sum_n \sum_j X_{i,j}^{m,n} P_k \right) \times \frac{MNP_k \prod_{i=1,i\neq k}^{MN}(1-P_i)}{\prod_{i=1}^{MN}(1-P_i) + MNP_k \prod_{i=1,i\neq k}^{MN}(1-P_i)} \tag{3}$$

where *m, n, i,* and *j* denote the number of ToR switches, the PMs in a rack, VCs, and the VMs in a VC, respectively; *M* and *N* denote the total number of racks and PMs in a rack, respectively; S_i denotes the total number of VMs required by VC *i*; $X_{i,j}^{m,n}$ denotes the allocation of a VM in a VC; if the VM *j* of VC *i* is assigned to PM *n* belonging to rack *m*, then $X_{i,j}^{m,n} = 1$; otherwise, $X_{i,j}^{m,n} = 0$; $P_{i,m}$ denotes the failure rate of the ToR switch *m* associated with VC *i*; P_i and P_k denote the failure rate of different PMs; event *B1* denotes a ToR switch *m* failure along with no PM failure; event *B2* denotes a PM failure concurrently with a ToR switch *m* failure; $\delta_{i,m}^{B1}$ and $\delta_{i,m}^{B2}$ denote the number of unavailable VMs in a VC *i* due to *B1* and *B2* when ToR switch *m* fails, respectively.

Proof. Refer to the proof in Appendix A.

4.1.2 BW Metric Model

Considering overlapping and hierarchical fault domains, we segment the set of PMs (i.e., pmlist) to multiple racks to decrease the complexity of the optimization problem. Since all PMs within a given rack belong precisely to the same fault domains and are indistinguishable in accordance with faults, an assignment of a VC can be described by a set of variables *{$t_{m,i,k}$}*; the variable $t_{m,i,k}$=1 if the VM *k* of VC *i* is allocated to the rack *m;* otherwise, $t_{m,i,k}$=0. To formally define BW_i, the indicator function for which rack pairs are used as its inputs, is represented by $I(\cdot,\cdot)$. For each such pair (m_1,m_2), if traffic from m_1 to m_2 (and vice-versa) traverses through a core link, $I(m_1,m_2)=1$; otherwise, $I(m_1,m_2)=0$. bw_{k_1,k_2} denotes the required bandwidth from VM k_1 to VM k_2. Thus, the overall bandwidth usage in the network core of VC *i* (i.e., BW_i) can be computed as follows.

$$BW_i = \sum\nolimits_{m_1,m_2} \sum\nolimits_{k_1,k_2} I(m_1,m_2) t_{m_1,i,k1} t_{m_2,i,k_2} bw_{k_1,k_2} \tag{4}$$

4.1.3 Optimal Availability-aware VC Allocation Model

In order to identify a near-optimal availability-aware VC allocation solution, our goal is to simultaneously minimize equations (1) and (4). More specifically, we exploit a joint optimization function, abbreviated as JOF, to measure the joint optimization value of the VC allocation solution as follows.

$$JOF = \frac{1}{\min\left((1-\theta)\dfrac{\sum_i BW_i - BW_0}{BW - BW_0} + \theta \dfrac{\sum_i risk_i - risk_0}{risk - risk_0} \right)} \tag{5}$$

s.t.

$$\textstyle\sum_{i=1}^{V} D_i^{mem} x_{ij} < C_j^{mem} \tag{6}$$

$$\textstyle\sum_{i=1}^{V} D_i^{bw} x_{ij} < C_j^{bw} \tag{7}$$

$$\textstyle\sum_{i=1}^{V} D_i^{cpu} x_{ij} < C_j^{cpu} \tag{8}$$

$$\textstyle\sum_{j=1}^{P} x_{ij} = 1,\ x_{ij} = 0\, or\, 1 \tag{9}$$

where θ is a tunable positive weight $0<\theta<1$; *P* is the number of PMs in the cloud datacenter; *V* is the number of VMs in the cloud datacenter; equations (6) to (8) show that the sum of the resource requirements of the VMs must be less than the PM's idle resource capacity; equation (9) shows that a VM can only be placed on a PM, such that x_{ij}=1 if *i-th* VM runs on the *j-th* PM; otherwise, x_{ij}=0; BW_0 denotes the minimum overall bandwidth usage in the network core, its value can be approximately acquired by the IVCA algorithm [26] after the VMs of each VC are clustered together. *BW* denotes the maximum overall bandwidth usage in the network core, its value can be approximately acquired by Algorithm 1. When a VM is allocated to a PM, VCMBW first traverses all the PMs (i.e., pmlist) to identify all other VMs in list of VMs (i.e., vmlist) and in the same VC as the VM. And then all pods accommodating these VMs are removed from a list of pod (i.e., podlist). Finally, a pod is randomly selected from the podlist to accommodate the VM.

Algorithm 1: VC Allocation of the Maximum *BW* (VCMBW)

```
Input: pmlist, vmlist, podlist    Output: VC allocation solution
the VMs of each VC in vmlist are clustered together
for VMs in vmlist do
    for PMs in pmlist do
        for vm1 of vmlist in the PM do    // vm1 is a VM
            if vm1 and VM are in the same VC then
                remove the pod including the vm1 from podlist
    for pods in podlist do
        if the pod can accommodate the VM then
            select a PM to accommodate the VM
return VC allocation solution
```

risk denotes the maximum overall risk cost of all VC allocation solutions, its value can be approximately acquired by Algorithm 2.

Algorithm 2:VC Allocation of the Maximum Risk (VCMaR)

```
Input: vmlist, racklist, pmlist  Output: VC allocation solution
Initialize the failure probability of each rack and PM
the VMs of each VC in vmlist are clustered together
for VMs in vmlist do
    for racks in racklist do
        if the rack has the maximum failure probability then
            for PMs in the rack do
                if the PM has the maximum failure probability then
                    if the PM can accommodate the VM then
                        allocate the VM to the PM
return VC allocation solution
```

As shown in the Algorithm 2, firstly, the VMs of each VC are clustered together. Secondly, when a VM is allocated to a PM, VCMaR first traverses all racks (i.e., racklist) of the cloud datacenter to identify a rack, which has the maximum failure probability. Then, it searches the rack for a PM, which has the maximum failure probability and can accommodate the VM.

risk₀ denotes the minimum overall risk cost of all VC allocation solutions, its value can be approximately acquired by Algorithm 3.

Algorithm 3:VC Allocation of the Minimum Risk (VCMiR)

```
Input: vmlist, racklist, pmlist  Output: VC allocation solution
Initialize the failure probability of each ToR switch and PM
the VMs of each VC in vmlist are clustered together
for VMs in vmlist do
    for racks excluding other VMs in the same VC with VM do
        if the rack has the minimum failure probability then
            for PMs accommodating the VM in the rack do
                if the PM has the minimum failure probability then
                    allocate the VM to the PM
return VC allocation solution
```

As shown in the Algorithm 3, firstly, the VMs of each VC are clustered together. Secondly, when a VM is allocated to a PM, VCMiR first traverses all racks of the cloud datacenter to identify a rack, which has the minimum failure probability and does not accommodate all other VM in the same VC with the VM. Then, it searches the rack for a PM, which has the minimum failure probability and can accommodate the VM.

4.2 Availability-aware VC Allocation Optimization

It is well known that BBO [11], which has had many extensions since its publication in 2008 (e.g., BBO/Complex [27]), applies biogeography [28] to solve a variety of optimization problems. It has certain features in common with other biology-based algorithms (e.g., genetic algorithms [29] and particle swarm optimization (PSO) [30]) and performs well compared to these algorithms [11].

Therefore, we exploit the standard BBO algorithm to solve the availability-aware VC allocation discrete joint optimization problem. In next section, we first introduce the BBO algorithm including the ecosystem model and the definition of parameters and operators. We then propose implementation scheme for AVCA.

4.2.1 The BBO Algorithm

In the BBO algorithm, an archipelago of islands (i.e., habitat) denotes the population of candidate solutions, in which each candidate solution is an island. The goodness (or fitness) of a solution with respect to an objective function is measured by its Habitat Suitability Index (HSI). A good (or poor) solution is an island with a high (or low) HSI. The decision variables are Suitability Index Variables (SIVs) (e.g., temperature and rainfall). A solution is represented by a vector of SIVs. Migration and mutation are two key operators of the BBO algorithm. A distinguishing feature of the BBO algorithm from other population-based optimization methods is migration, which is introduced to probabilistically share SIVs between solutions, thus increasing the quality of low HSI solutions. The mutation is used to probabilistically replace some SIVs in a solution by randomly generating new SIVs. The initial population of candidate solutions evolves iteratively from generation to generation until a termination criterion is met. In each repetition, a migration followed by a mutation is performed. Further, the above stochastic operators model the validity of a potential solution and can improve the latter incrementally.

To exploit the BBO algorithm, we map the availability-aware VC allocation problem to an ecosystem (i.e., population), which is comprised of multiple islands (i.e., individuals). These islands have the same optimization objective (i.e., equation (5)) and constraints (i.e., equations (6) to (9)); that is, each island optimizes itself by sharing information with other islands to optimize the ecosystem.

$$E_{PL}=\begin{vmatrix} e_{1,1} & \mathrm{e}_{1,2} & \cdots & \mathrm{e}_{1,L} \\ e_{2,1} & \mathrm{e}_{2,2} & \cdots & \mathrm{e}_{2,L} \\ \cdots & \cdots & \cdots & \cdots \\ e_{P,1} & \mathrm{e}_{P,2} & \cdots & \mathrm{e}_{P,L} \end{vmatrix} \tag{10}$$

For reader convenience, the ecosystem is represented by a matrix E_{PL} (as shown in equation (10)). The total number of islands in the ecosystem (i.e., the size of the population) is denoted by the row number *P*. The total number of VMs allocated is denoted by the column number *L*. The PM number assigned to *j-th* VM in *i-th* individual is denoted by the matrix element $e_{i,j}$. Therefore, the *i-th* individual can be denoted by the candidate solution $\{e_{i,1}, e_{i,2},\ldots, e_{i,j}, \ldots, e_{i,L}\}$. For the sake of clarity, island, habitat, and individual are equivalent and used interchangeably in following section.

Let $E=\{E_1, E_2, \ldots, E_P\}$ represent an ecosystem including *P* islands. $E_i=\{e_{i,1}, e_{i,2},\ldots, e_{i,L}, O, C_1, C_2, C_3, C_4\}$ denotes the *i-th* island, which contains a vector of *L* SIVs, one objective *O*, and four constraints C_1, C_2, C_3, and C_4. *O* represents one objective (i.e., equation (5)). The four constraints C_1, C_2, C_3, and C_4 correspond to equations (6) to (9). Each *SIV* represents the index of the PM, which hosts a VM. The *HSI* (i.e., fitness) of the island E_i is calculated by equation (5). Considering the above ecosystem and the specific characteristics of the VC allocation joint optimization problem, the parameters and operators of the BBO algorithm are defined as follows.

According to the related literature [11], the immigration rate λ and emigration rate μ of a habitat is a function of species *s* (i.e., the number of PMs used) (as shown in equations (11) to (12)). As the number of species *s* gradually increases, the immigration rate λ_s and emigration rate μ_s gradually decreases and increases, respectively. When λ_s is equal to μ_s, the number of species *s* in the habitat reaches equilibrium state S_0, which migrates as the environment changes. Assume that the maximum immigration rate is equal to the maximum emigration rate, and λ_s and μ_s increase linearly with them.

$$\mu_s=\frac{I\cdot s}{S_{\max}} \tag{11}$$

$$\lambda_s=I\left(1-\frac{s}{S_{\max}}\right) \tag{12}$$

where S_{max} and *I* denote the maximum number of species in a habitat (i.e., the minimum number of PMs and VMs allocated) and the immigration rate, respectively; and the probability that the habitat contains exactly *s* species can be denoted by P_s, which changes from time *t* to time $(t+\Delta t)$ as follows [11].

$$P_s(t+\Delta t)=P_s(t)(1-\lambda_s\Delta t-\mu_s\Delta t)+P_{s-1}\lambda_{s-1}\Delta t+P_{s+1}\mu_{s+1}\Delta t \tag{13}$$

When Δt is small enough, the probability of more than one migration can be ignored. Therefore, taking the limit of (13) as $\Delta t\to 0$, the steady state value for probability P_s is formulated as follows [31].

$$P_s=\begin{cases} \dfrac{1}{1+\sum_{s=1}^{S_{max}}\dfrac{\lambda_0\lambda_1\ldots\lambda_{s-1}}{\mu_1\mu_2\ldots\mu_s}} & \mathrm{s}=0 \\ \dfrac{\lambda_1\lambda_2\ldots\lambda_{S-1}}{\mu_1\mu_2\ldots\mu_s\left(1+\sum_{s=1}^{S_{max}}\dfrac{\lambda_0\lambda_1\ldots\lambda_{s-1}}{\mu_1\mu_2\ldots\mu_s}\right)} & 1\le s\le S_{max} \end{cases} \tag{14}$$

where emigration rate μ_s cannot be assigned to zero to ensure the existence of the above probabilities.

Definition 1 (Migration Operator): $E\to E_i$ is a probabilistic operator that adjusts habitat E_i based on the ecosystem *E*. The probability that E_i is modified is proportional to its immigration rate λ_i, and the probability that the source of the modification comes from E_j is proportional to the emigration rate μ_j.

Definition 2 (Mutation Operator): $E_i\to E_i$ is a probabilistic operator that randomly modifies habitat SIVs based on a *priori* probability of existence of the habitat.

The mutation probability m_s of the habitat is inversely proportional to the number of species *s*, which can be formulated as follows.

$$m_s=m_{\max}\left(1-\frac{P_s}{P_{\max}}\right) \tag{15}$$

where $P_{\max}$ and $m_{\max}$ respectively represent the maximum value of the probability P_s and the mutation probability which is set at 0.1 [31].

Definition 3 (Removal Operator): $E_i \rightarrow E_i$ is an operator that identify overloaded PMs of the habitat and replace them with normal PMs.

In original BBO algorithm [11], the mutation operator simply replaces the original SIV with a randomly generated SIV, and the migration operator replaces the immigrated SIV with the emigrated SIV. The two operators are easy to produce similar solution and lead to poor diversity of population. Therefore, the original BBO algorithm designs the removal operator to eliminate these similar solutions and improve the diversity of population. When they are applied to AVCA, the two operators will generate the overloaded PMs; that is, the resource requirement of all VMs placed in one PM is far more than the maximum capacity of the PM. Thus, to remove these overloaded PMs and improve the diversity of the ecosystem, the removal operator is proposed to identify the overloaded PMs and replace them with normal PMs.

Since the above stochastic operators make the whole algorithm non-deterministic, we exploit two strategies to enhance the performance of the AVCA: (1) the exploitation of elitism to ensure that the best habitat is not lost from one generation to the next. It is common to save the best habitats at the beginning of each generation into a set and then replace the worst habitats with the set at the end of the generation. The size of the set (i.e., *NE*) is a tuning parameter, but it typically includes the best two habitats [32]. (2) The migration rates are introduced to decide how much information to share between habitats; the selected SIVs are replaced in a way that the modified habitat is always feasible and better than the original habitat. Since AVCA exploits the mutation and removal operators to enhance the diversity of population, the two strategies can improve its performance and avoid local extrema.

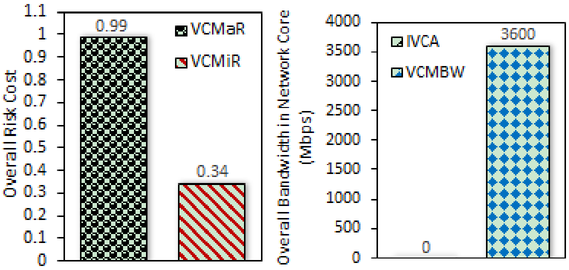
4.2.2 Implementation Scheme of AVCA

In this section, we propose implementation scheme of AVCA with the BBO algorithm to solve the availability-aware VC allocation joint optimization problem. The pseudocode of the AVCA is presented in Algorithm 4.

Algorithm 4: Availability-aware VC Allocation (AVCA)

```
Input: pmlist, vmlist, racklist, VCs     Output: VC allocation solution
Initialize the BBO parameters $Smax$, $I$, $m_{max}$, $G$, $P$, and $NE$
initialize $BW$ using Algorithm 1
initialize $BW_0$ using IVCA algorithm
initialize $risk$ using Algorithm 2
initialize $risk_0$ using Algorithm 3
Initialize and sort a random set of habitats by $HSI$
  for count of generation is not equal to $G$ do
      Save the $NE$ elites.
      Use $\lambda$ and $\mu$ to modify each non-elite habitat using Definition 1
      Mutate each non-elite habitat using Definition 2.
      Remove the overloaded PM using Definition 3.
      Sort all habitats by $HSI$ recomputed.
      Replace the $NE$ habitats at the end with the elites.
      Reorder all habitats by $HSI$.
  end for
return VC allocation solution
```

The algorithm first initializes the size of the population *P*, the number of generations *G*, the maximum species S_{max}, the maximum immigration rates *I*, the maximum mutation rate m_{max}, the number of elites *NE*, the maximum risk cost *risk*, the minimum risk cost $risk_0$, the maximum bandwidth usage *BW*, and the minimum bandwidth usage BW_0. Second, it initializes and sorts a random set of habitats, and each habitat corresponds to a potential solution of the given problem. Third, it probabilistically uses the mutation and migration operator to mutate and modify each non-elite habitat using definitions 1 and 2 and removes the overloaded PM in each habitat using Definition 3. Finally, it re-computes each *HSI* to sort all habitats in the ecosystem, replaces the habitats at the end by *NE* elites, reorder all habitats replaced by *HSI*, and then proceeds to the third step for next iteration. This loop can be terminated after a predefined number of generations *G*.



(a) The value of *risk* and $risk_0$ (b) The value of *BW* and BW_0

Fig. 3. The values of BW_0, *risk*, $risk_0$, and *BW* are obtained by the algorithms of Section 4.1.3

5 Performance Evaluation

In this section, we exploit the experiments to evaluate the efficiency and effectiveness of AVCA. We also publish source code on the Web[1].

5.1 Experiment Setup

We implemented our algorithm in WebCloudSim system [26],[33], which is based on CloudSim [34]. This system including a 16-port fat-tree DCN with 64 core switches and 16 pods is constructed to conduct all of the experiments. There are 8 aggregation switches and 8 edge switches in each pod. Therefore, there are 128 aggregation switches and 128 edge switches in the cloud datacenter, in which each edge switch can connect to 8 PMs, and each PM can host one or more VMs. In order to reflect the effect of VM allocation, we simulate a data center comprising 1024 heterogeneous PMs and 120 VMs. Each PM is modeled randomly to have a dual-core CPU with performance equivalent to 3720 or 5320 MIPS, 4GB of RAM, 1GB/s network bandwidth and 1TB of storage [35]. The CPU and memory capacity of each VM is chosen randomly from four groups: 500 MIPS and 0.6 GB, 1000 MIPS and 1.7 GB, 2000 MIPS and 3.75 GB, or 2500 MIPS and 0.85 GB [35]. The disk capacity of each VM is 1GB. The bandwidth requirement of each VM is set randomly between 100 and 500 Mbps. The failure probabilities of the ToR switch (P_r) and PM (P_s) are respectively set randomly between 0.05 ~ 0.15

[1] http://sguangwang.com/Source code/AVCA-sourcecode.zip

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2694838, IEEE Transactions on Services Computing

and 0.02 ~ 0.12 [6]. Appropriate parameter values of Algorithm 4 are determined on the basis of the related literatures and preliminary experiments, the size of the population was set at 20, the maximum mutation rate was set at 0.1 [31], the number of generations was set at 30, and the number of elites was set at 2 [32]. Based on the algorithms of the Section 4.1.3, we can approximately acquire that the values of *risk* and $risk_0$ are respectively 0.34 and 0.99, and the values of *BW* and BW_0 are respectively 0 Mbps and 3600 Mbps (as shown in Fig. 3) while allocating 5 VC requests.

In order to research different allocation approaches, we exploit the data-intensive application that requires multiple VMs of different sizes to execute in our system model. A study on the number of VMs involved in a data-intensive application shows that more than 80% of applications use fewer than 10 VMs [20]. There are three types of data-intensive applications, including workflow data-intensive applications [36], multi-tiered data-intensive applications [37], and batch data-intensive applications (i.e., MapReduce) [38]. Based on the characteristics of these data-intensive applications, a set of general applications are exploited to make the discussion clearer in our experiments. Each application is comprised of 3 tasks (e.g., t1, t2, and t3), in which each task consists of some computation and communication stages and is processed by a VM. Please note that only if t1 and t2 both transfer data to t3, then, t3 can enter the execution stage [26]. Different from the work in [12], our work is mainly focus on the VC allocation algorithm at any given time. That is, when a certain number of the VC requests are received at any given time, our VC allocation algorithm is triggered to identify a near-optimal VC allocation scheme based on two optimization objectives. Therefore, we only need to give our VC allocation algorithm a certain number of VC requests at some point and does not have to consider the arrival pattern of VC requests.

To assess the performance of our approach (AVCA), in later sections, we compare our approach with three other algorithms: Random First-fit (RFF), PSO [39], and multi-objective grouping genetic algorithm (MGGA) [40]. It is well known that RFF is a classical greedy approximation algorithm. When a VM is allocated, there may be multiple PM candidates that satisfy the constraints. RFF randomly selects the PM to host the VM. Please note that RFF is mainly used as a reference value of other algorithms.

5.2 Experimental Results and Evaluation

In this section, we first compare the performance of AVCA with the three related approaches in terms of JOF, overall bandwidth usage in the network core, and overall risk cost while executing a set of data-intensive applications. We then analyze the impact of experimental parameters including the tunable positive parameter θ, the number of data-intensive applications, and the failure ranges of the ToR switch and PM.

5.2.1 Comparison of Joint Objective Function Result

The first set of experiments aims at analyzing the performance of our approach by comparing the three other approaches in terms of average JOF, average overall bandwidth usage in the network core and average overall risk cost, which processes a set of data-intensive applications. In this experiment, the number of data-intensive applications, VMs, and PMs was set at 5, 120, and 1024, respectively; the tunable positive parameter θ was set at 0.5; the failure ranges of the ToR switch and PM were chosen randomly between 0.05 ~ 0.15 and 0.02 ~ 0.12, respectively.

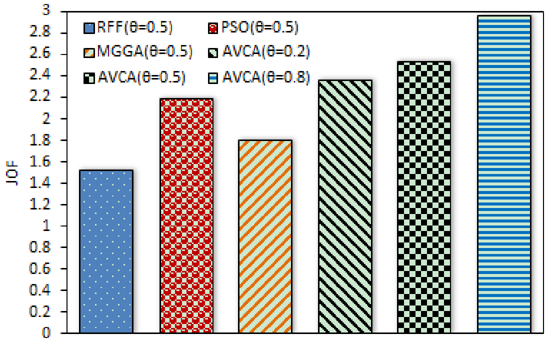


Fig. 4. Comparison of JOF.

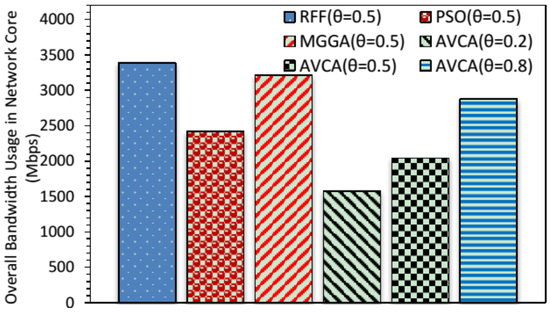


Fig. 5. Comparison of overall bandwidth usage in the network core.

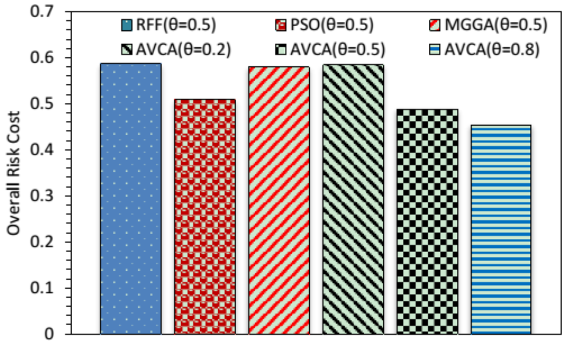


Fig. 6. Comparison of overall risk cost.

As shown in Fig. 4 to 6, the JOF of AVCA is higher than three other approaches (i.e., RFF, PSO, and MGGA). This is due to the fact that RFF is a greedy approximation algorithm, which randomly selects a PM to host a VM. Although PSO and MGGA are heuristic algorithms, they are more likely to clump together in similar groups, while BBO is a new stochastic evolutionary algorithm developed for global optimization, and its solutions do not necessarily have a built-in tendency to cluster. Therefore, the average growth rate in JOF using AVCA for the three other approaches are 73%, 20%, and 46%, respectively. Meanwhile, since the tunable weight factor θ of the RFF, PSO, and MGGA algorithms is set at 0.5, the three approaches consume relatively more overall bandwidth usage in the net-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2694838, IEEE Transactions on Services Computing

work core and overall risk cost. Unlike the above three approaches, the tunable weight factor θ of AVCA is set at 0.2, 0.5, and 0.8. The JOF and the overall bandwidth usage in the network core using the AVCA increase by 7.4% and 29.5%, respectively, and the overall risk cost using the AVCA decreases by 16.5%, when the tunable weight factor θ adjusts from 0.2 to 0.5. Likewise, the JOF and the overall bandwidth usage in the network core increase by 16.4% and 40.8%, respectively, and the overall risk cost decreases by 7%, when the tunable weight factor θ adjusts from 0.5 to 0.8. Thus, the value of the tunable weight factor θ determines the optimization emphasis; that is, when its values are set at 0.2, 0.8, and 0.5, the optimization emphasis is the overall bandwidth usage in the network core, the overall risk cost, or the above two aspects, respectively. Further, the overall bandwidth usage in the network core is increasing and the overall risk cost is decreasing with the increase in the tunable weight factor θ .

Based on the above experimental result analysis in terms of average JOF, average overall bandwidth usage in the network core, and average overall risk cost, our approach (AVCA) outperforms three other approaches. Next, we further analyze the impact of the different failure ranges of the ToR switch and PM on the JOF (as shown in Fig. 7). Meanwhile, we also analyze the impact of the different number of data-intensive applications (i.e., VCs) on the JOF (as shown in Fig. 8).

5.2.2 Sensitivity to the Failure Ranges of ToR switch and PM

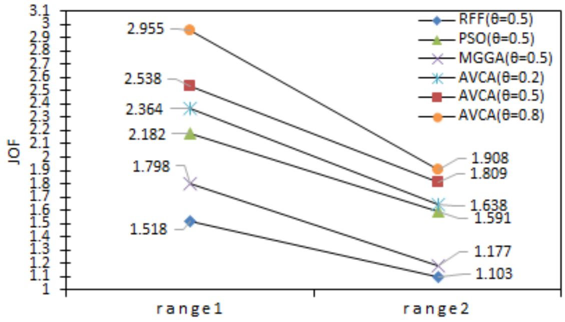
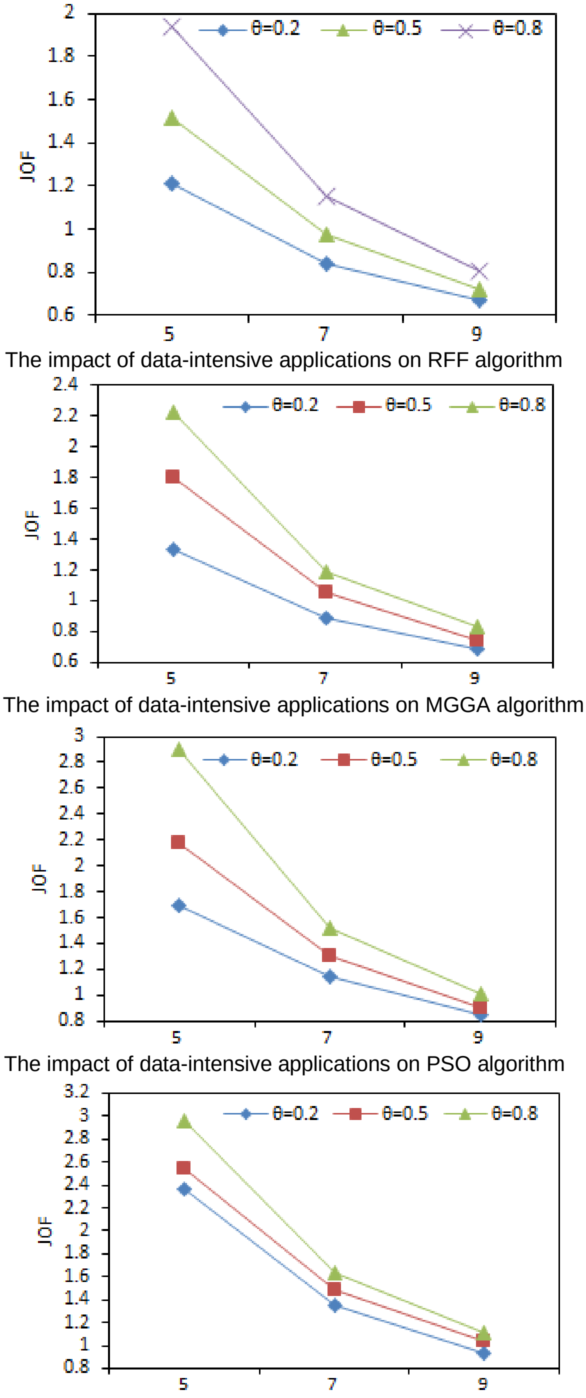


Fig. 7. The sensitivity to different failure ranges of the ToR switch and PM. The failure ranges of the ToR switch and PM represent that each ToR switch and PM can be specified from the two failure probability ranges; The JOF of all approaches tends to decrease when the failure ranges of the ToR switch and PM are chosen from *range1* (i.e., 0.05 ~ 0.15 and 0.02 ~ 0.12) to *range2* (i.e., 0.05 ~ 0.25 and 0.02 ~ 0.22).

Fig. 7 shows the impact of the failure ranges of the ToR switch and PM on all approaches. To clearly show its impact, the number of data-intensive applications, VMs and PMs was set at 5, 120, and 1024, respectively, and the tunable positive parameter θ was set at 0.5. We varied the value of the failure ranges of the ToR switch and PM from *range1* to *range2* in this experiment. The figure shows that the average JOF of each approach tends to decrease as a whole, as the failure ranges of the ToR switch and PM are broadened from *range1* to *range2*. That is, the number of high failure probabilities for the ToR switch and PM in *range2* are greater than those in *range1*. Therefore, when a set of VCs is allocated to the cloud datacenter, where the failure ranges of the ToR switch and PM are in *range2*, the availability of the VC allocation scheme does not make it easy to obtain the guarantee. Although the JOF of each approach decreases from *range1* to *range2*, our approach still outperforms the three other approaches.

5.2.3 Sensitivity to the Number of Data-intensive Applications



(a) The impact of data-intensive applications on RFF algorithm

(b) The impact of data-intensive applications on MGGA algorithm

(c) The impact of data-intensive applications on PSO algorithm

(d) The impact of data-intensive applications on AVCA algorithm

Fig. 8. The sensitivity to different number of data-intensive applications and different values of the tunable positive parameter θ . The number of data-intensive applications represents how many data-intensive applications can be processed by a set of VCs. The tunable positive parameter θ is set at 0.2, 0.5, and 0.8. The JOF decreased with the increase in the number of data-intensive applications for each value of the tunable positive parameter θ .

Fig. 8 shows the impact of the different number of data-intensive applications on all approaches. To clearly show its impact, the number of VMs and PMs was set at 120 and 1024, respectively, the tunable positive parameter θ was set at 0.2, 0.5, and 0.8; and the failure ranges of the ToR switch and PM are chosen randomly from 0.05 ~ 0.15 and 0.02 ~ 0.12, respectively. We varied the number of data-intensive applications from 5 to 9 with a step value of 2 in this experiment. These figures show that the average JOF decreased with the increase in the number of data-intensive applications; the JOF using the AVCA algorithm is the highest for each value of θ because with more applications, the minimum bandwidth usage in the network core and the risk cost needed are both larger; hence, the JOF calculated using equation (5) is smaller.

Moreover, the tunable positive parameter θ is set at different values to further research the impact of data-intensive applications on all approaches. As shown in Fig. 8, when θ varies from 0.2 to 0.8, the differences in the JOF which exploits the AVCA algorithm, are more pronounced than other algorithms. In particular, when θ is set at 0.8, although the JOF using the AVCA algorithm decreases by about 62.2% when the number of data-intensive applications increases from 5 to 9, its JOF is still the highest of all the algorithms. This is due to the fact that the change ranges for JOF searched by all algorithms are smaller with the increase in applications under a certain number of VMs and PMs. Meanwhile, these figures further confirm and extend the analysis of Section 5.2.1; that is, the JOF of each approach increases with the increase from 0.2 to 0.8.

6 Limitations of Our Approach

Besides the main objectives of reducing the bandwidth usage in the network core and the risk cost by considering the concurrent PM and ToR switch with heterogeneous failure probabilities, some further practical issues may need to be considered during the deployment of our approach. Next, we will describe some of them and discuss how our approach can be extended to support them.

DCN topologies: As aforementioned, we have introduced the fat-tree DCN with the VC abstractions to our experimental platform. Although the fat-tree is widely used in DCN, other topologies (e.g., torus, 3D mesh) are also widely used as interconnection network. To be feasible for these topologies, AVCA for more general DCN topologies is among our future directions.

VC migration: To achieve the goals of energy saving, failure recovery, load balancing, and system maintenance, live migration of VMs has become a key ingredient behind the management activities of cloud computing system [41]. However, since most of the live migration techniques of VM mainly focused on the migration of a single VM, this means that these techniques are insufficient when the whole VC or multiple VCs need to be migrated [42]. Therefore, we leave the research of VC migration strategies to improve the migration performance of VCs for our future work.

Management software failures: It is a widely held belief that software reliability is important branch of reliability theory [9]. Although this paper only considers the hardware failures, in our future work, we will research availability-aware VC allocation optimization problem by combining the hardware and software failures.

In addition to the above scenarios, there are interesting extensions for future work. These include the multiple PM and ToR switch failures, real applications, and other performance metrics (e.g., CPU and memory) in addition to network.

7 Conclusions and Future Work

With the growing popularity of cloud computing, datacenters have become common platforms for supporting data-intensive applications. Enhancing the availability requirements of data-intensive applications has become a high-profile problem for cloud providers. In this paper, we proposed and mathematically defined two measures: 1) one measure characterizes the bandwidth usage in the network core; 2) the other measure formulates the risk cost by considering the concurrent PM and ToR switch with heterogeneous failure probabilities. We also introduced and mathematically established a joint optimization function to simultaneously minimize the above measures. Finally, we proposed an approach with the BBO algorithm that is demonstrated based on extensive experiments to be quite effective.

In our experimentations, when we solve our joint optimization problem, the constraint of the core link capacity in the fat-tree DCN and VC migration are not considered. In the future, we will add the constraint and VC migration to our optimization problem.

Acknowledgments

The work presented in this paper is supported by the NSFC (61472047). Shangguang Wang is the corresponding author.

References

[1] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra, " *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'11)*, pp. 98– 109, 2011.

[2] J. Ho,P. Hsiu, and M.Chen, "Improving Serviceability for Virtual Clusters in Bandwidth-Constrained Datacenters," *Proc. 8th IEEE International Conference on Cloud Computing (CLOUD'15)*, pp. 710-717, 2015.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'09)*, pp. 51-62, 2009.

[4] L. Zhang, X. Yin, Z. Li, and C. Wu, "Hierarchical virtual machine placement in modular data centers," *Proc. IEEE 8th International Conference on Cloud Computing (CLOUD'15)*, pp. 171-178, 2015.

[5] Cisco Global Cloud Index, "Forecast and Methodology, 2015–

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2017.2694838, IEEE Transactions on Services Computing

2020," accessed march 2017 from http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf.

[6] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'11)*, pp. 350-361, 2011.

[7] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'11)*, pp. 242-253, 2011.

[8] Ponemon institute. cost of data center outages. [Online]. http://www.emersonnetworkpower.com/en-US/Resources/ Market/Data-Center/Latest-Thinking/Ponemon/Documents/ 2016-Cost-of-Data-Center-Outages-FINAL-2.pdf.

[9] E. Bauer, and R. Adams, "Reliability and availability of cloud computing," John Wiley & Sons, 2012.

[10] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'08)*, pp. 63-74, 2008.

[11] D. Simon, "Biogeography-based optimization," *IEEE Transaction on Evolutionary Computation*, vol.12, no.6, pp.702-713, 2008.

[12] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," *Proc. IEEE International Conference on Computer Communication (INFOCOM'14)*, pp. 289-297, 2014.

[13] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888-1906, 2013.

[14] J. Duan, Z. Guo, and Y.Yang, "Cost efficient and performance guaranteed virtual network embedding in multicast fat-tree DCNs," *Proc. IEEE Conference on Computer Communication (INFOCOM'15)*, pp.136-144, 2015.

[15] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Harmony: Dynamic heterogeneity-aware resource provisioning in the cloud," *Proc. IEEE 33rd International Conference on Distributed Computing Systems (ICDCS'13)*, pp. 510-519, 2013.

[16] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," *Proc. 1st ACM symposium on Cloud computing (SoCC'10)*, pp. 193-204, 2010.

[17] X. Wu, D. Turner, C.-C. Chen, D. A. Maltz, X. Yang, L. Yuan, and M. Zhang, "NetPilot: automating datacenter network failure mitigation," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'12)*, pp. 419-430, 2012.

[18] W.-L. Yeow, C. Westphal, and U. C. Kozat, "Designing and embedding reliable virtual infrastructures," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'11)*, pp. 57-64, 2011.

[19] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," *Proc. IEEE 5th International Conference on Cloud Computing (CLOUD'11)*, pp. 196-203, 2012.

[20] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'12)*, pp. 431-442, 2012.

[21] Z. Yang, L. Liu, C. Qiao, S. Das, R. Ramesh, and A. Y. Du, "Availability-aware energy-efficient virtual machine placement," *Proc. IEEE International Conference on Communications (ICC'15)*, pp. 5853-5858, 2015.

[22] R. A. da Silva and N. L. Da Fonseca, "Algorithm for the placement of groups of virtual machines in data centers," *Proc. IEEE International Conference on Communications (ICC'15)*, pp. 6080-6085, 2015.

[23] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, "A flexible model for resource management in virtual private networks," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'99)*, pp. 95-108, 1999.

[24] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'08)*,, pp. 17-29, 2008.

[25] J. Békési, G. Galambos, and H. Kellerer, "A 5/4 linear time bin packing algorithm," Journal of Computer System Sciences, vol.60, no.1, pp.145–160, 2000.

[26] J. Liu, S. Wang, A. Zhou, S. Kumar, F. Yang, and R. Buyya, "Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability. "*IEEE Transactions on Cloud Computing*, vol.PP, 99, pp.1-1, 2016.

[27] D. Du and D. Simon, "Complex system optimization using biogeography-based optimization," *Mathematical Problems in Engineering*, vol.2013, Article ID 456232, 17 pages, 2013.

[28] R. MacArthur and E. Wilson, "The theory of biogeography," *Princeton University Press, New Jersey*, pp. 19-67, 1967.

[29] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," *Proc. IEEE International Conference on Robotics and Automation*, pp. 1186-1192, 1992.

[30] J. Kenndy and R. Eberhart, "Particle swarm optimization," Proc. *IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

[31] H. Ma, S. Ni, and M. Sun, "Equilibrium species counts and migration model tradeoffs for biogeography-based optimization, *Proc. IEEE 48th International Conference on Decision and Control and 28th Chinese Control Conference (CDC/CCC'09)*, pp. 3306-3310, 2009.

[32] D. Simon, M. Ergezer, and D. Du, "Population distributions in biogeography-based optimization algorithms with elitism," Proc. *IEEE International Conference on Systems, Man and Cybernetics (SMC'09)*, pp. 991-996, 2009.

[33] A. Zhou, S. Wang, Z. Zheng, C. Hsu, M. Lyu, and F. Yang, "On cloud service reliability enhancement with optimal resource usage," *IEEE Transactions on Cloud Computing*, vol. 4, no, 4, pp.452-466, 2014.

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

[35] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and*

Experience, vol. 24, no. 13, pp. 1397-1420, 2012.

[36] S. K. Garg and R. Buyya, "An environment for modeling and simulation of message - passing parallel applications for cloud computing," *Software: Practice and Experience*, vol. 43, no. 11, pp. 1359-1375, 2013.

[37] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma, "Application-driven bandwidth guarantees in datacenters," *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM'14)*, pp. 467-478, 2014.

[38] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reduce-merge: simplified relational data processing on large clusters," *Proc. ACM International Conference on Special Interest Group on Management Of Data (SIGMOD'07)*, pp. 1029-1040, 2007.

[39] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," *Proc. IEEE International Conference on Parallel and Distributed Systems (ICPADS'13)*, pp. 102-109, 2013.

[40] J. Xu, J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," *Proc. IEEE/ACM International Conference on Green Computing and Communications (GreenCom'10)*, pp. 179-188, 2010.

[41] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," *Proc. IEEE International Conference on Computer Communication (INFOCOM'14)*, pp. 574-582, 2014.

[42] K. Ye, X. Jiang, R. Ma, and F. Yan, "Vc-migration: Live migration of virtual clusters in the cloud," *Proc. ACM/IEEE 13th International Conference on Grid Computing (GRID'12)*, pp. 209-218, 2012.

Jialei Liu is a Ph.D. candidate at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He received his M.E. in computer science and technology from Henan Polytechnic University in 2008. His research interests include cloud computing and service reliability.

Shangguang Wang received his PhD degree at Beijing University of Posts and Telecommunications in 2011. He is an associate professor at the State Key Laboratory of Networking and Switching Technology (BUPT). He has published more than 100 papers, and played a key role at many international conferences, such as general chair and PC chair. His research interests include service computing, cloud computing, and mobile edge computing. He is a senior member of the IEEE, and the Editor-in-Chief of the International Journal of Web Science.

Ao Zhou is an assistant professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She received her Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2015. Her research interests include cloud computing and service reliability

Fangchun Yang received his Ph.D.in communications and electronic systems from the Beijing University of Posts and Telecommunication in 1990. He is currently professor at the Beijing University of Posts and Telecommunication, China. He has published six books and more than 80 papers. His current research interests include network intelligence, service computing, communications software, soft-switching technology, and network security. He is a fellow of the IET.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He also serves as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the university, commercializing its innovations in Grid and Cloud Computing. Dr. Buyya has authored/co-authored over 450 publications. He is one of the most highly cited authors in computer science and software engineering worldwide. Microsoft Academic Search Index ranked Dr. Buyya as one of the Top 5 Authors during the last 10 years (2001-2012) and #1 in the world during the last 5 years (2007-2012) in the area of Distributed and Parallel Computing. For further information on Dr. Buyya, please visit: http://www.buyya.com.