HAL
open science

VDC Planner: Dynamic Migration-Aware Virtual Data Center Embedding for Clouds

Mohamed Faten Zhani, Qi Zhang, Gwendal Simon, Raouf Boutaba

To cite this version:

Mohamed Faten Zhani, Qi Zhang, Gwendal Simon, Raouf Boutaba. VDC Planner: Dynamic Migration-Aware Virtual Data Center Embedding for Clouds. IM 2013 : proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, May 2013, Ghent, Belgium. hal-00846058

HAL Id: hal-00846058

https://hal.science/hal-00846058v1

Submitted on 18 Jul 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

VDC Planner: Dynamic Migration-Aware Virtual Data Center Embedding for Clouds

Mohamed Faten Zhani*, Qi Zhang*, Gwendal Simon† and Raouf Boutaba*
*David R. Cheriton School of Computer Science, University of Waterloo, Canada.
{mfzhani, q8zhang, rboutaba}@uwaterloo.ca
† Telecom Bretagne, Institut Mines Telecom, France.
gwendal.simon@telecom-bretagne.eu

***Abstract*—Cloud computing promises to provide computing resources to a large number of service applications in an on-demand manner. Traditionally, cloud providers such as Amazon only provide guaranteed allocation for compute and storage resources, and fails to support the bandwidth requirements and performance isolation among these applications. To address this limitation, recently a number of proposals advocate providing both guaranteed server and network resources in the form of Virtual Data Centers (VDCs). This raises the problem of optimally allocating both servers resources and data center networks to multiple VDCs in order to optimize total revenue, while minimizing the total energy consumption in the data center. However, despite recent studies on this problem, none of the existing solutions have considered the possibility of using VM migration to dynamically adjust the resource allocation, in order to meet the fluctuating resource demand of VDCs.**

In this paper, we propose VDC Planner, a migration-aware dynamic virtual data center embedding framework that aims at achieving high revenue while minimizing the total energy cost over-time. Our framework supports various usage scenarios, including VDC embedding, VDC scaling as well as dynamic VDC consolidation. Through experiments using realistic workload traces, we show our proposed approach achieves both higher revenue and lower average scheduling delay compared to existing solutions in the literature.

I. Introduction

Cloud computing is a model that promises to allocate resources to large-scale service applications in an on-demand fashion. In a cloud computing environment, the traditional role of service providers is divided into two: The *Infrastructure Providers* (InPs) own the physical resources in data centers, and lease them using a pay-as-you-go pricing model, while the *Service Providers* (SP) rent the resources offered by InPs and provide services to *end users* over the Internet. Traditionally, InPs offer resources in terms of virtual machines (VMs), and ignore network requirement imposed by the services running in these VMs. This has led to a number of issues regarding network performance, security and manageability [2].

To address these limitations, a large number of research proposals have recently advocated to offer *Virtual Data Center* (VDCs) instead of VMs. A VDC consists of virtual machines (VMs) connected through virtual switches, routers and links with guaranteed bandwidth. Compared to traditional VM-based offerings, selling resources in terms of VDCs allows SPs to to achieve better performance isolation and Quality of Service (QoS) for their applications. Moreover, InP can make informed decisions for traffic engineering given VDC-specific traffic requirements, which eases the burden of InP for network management.

However, despite its benefits, designing an efficient resource management scheme for VDCs is a challenging problem. One of the key challenges is the *VDC embedding problem*, which asks to map VDC components (e.g., virtual machines, virtual switches and links) onto physical nodes and links. From an InP's perspective, the goal is to adopt efficient allocation schemes to maximize their net income (i.e., the difference between revenue and operational costs) while satisfying the resource requirements (CPU, memory, disk and bandwidth) of every VDC. This can be divided into several inter-dependent objectives: (1) maximizing the total revenue obtained from the embedded VDC requests, (2) minimizing request scheduling (i. e., queueing) delay, which refers to the time a request spent in the waiting queue before it is scheduled, and (3) minimizing the total energy consumed by the data center. The scheduling delay is an important performance metric not only because it concerns the responsiveness of the cloud data center to demand fluctuations, but also because it affects the performance of cloud applications (e. g., running time of MapReduce jobs) [16]. It is noteworthy that VDC embedding is an $\mathcal{NP}$-hard problem as it generalizes the bin-packing problem.

To make the matter worse, InPs can offer more flexibility to SPs by allowing them to scale up and down their VDCs according to their needs. For instance, an SP can ask for an increase of the VDC capacity in terms of VMs and virtual links to accommodate rapid increase in service demand. An SP can also reduce the size of its VDC during idle periods to save resource rental cost. Although flexibility is a key advantage of cloud computing, previous works related to VDC embedding do not focus on the management of such re-scaling operations. Yet, scaling up embedded VDCs is not trivial. For example, a SP may wish to increase the bandwidth allocation for a given embedded VM but the physical machine that hosts this VM may not have sufficient free bandwidth to support this operation. On the other hand, scaling down embedded VDCs is an opportunity to reduce operational costs. In particular, scaling down VDCs reinforces interest of VM consolidation algorithms [11], [10], [17], [14], [15], which aim at maximizing the utilization of active machines while allowing idle machines to be turned off.

To accommodate flexibility in VDC embedding, a simple yet common solution is to re-embed the VDC from scratch (e.g., [3]). This solution can however result in disrupting services offered by the VMs. A more promising solution is to migrate some embedded VMs from a physical machine to another. However, migrating a VM has associated costs in terms of service disruption and bandwidth usage. In particular, migrating a VM can cause the VM run at reduced speed, thereby violating the Service Level Agreement (SLA). Such a violation is translated into a penalty that the InP ha to pay. Hence, the InP must weigh the benefit and the cost of a migration and make the right decision that minimizes his overall costs. To the best of our knowledge, previous works related to VM migration do neither address VDC embedding, nor take into account all the parameters in terms of management, migration and impact on performance.

To address the aforementioned challenges, we introduce *VDC Planner*, a framework that supports *migration-aware virtual data center embedding*. Migration is a key feature in VDC Planner: it is used to both improve VDC embedding capability and process scaling up and down requests. VDC Planner differs from previous work on VDC embedding mainly in the fact that it is migration-aware, and attempts to use migration to improve solution quality while minimizing total migration costs. In our work, we provide a general formulation, which rigorously defines the problem of embedding, scaling up and down VDC requests while considering the migration cost. To the best of our knowledge, our formulation is the first one to consider migration cost and the existence of multiple resources.

The rest of this paper is organized as follows. In Section 2 we survey recent research effort related to migration-aware VDC embedding. We provide a formal formulation of the migration-aware VDC problem in Section 3. Section 4 provides an overview of VDC Planner and describes various usage scenarios and our proposed algorithm for each scenario in details. Lastly, we demonstrate the effectiveness of VDC Planner in Section 5, and conclude the paper in Section 6.

II. Related Work

Realizing that data center networks today do not provide performance isolation between collocated service applications, there is an emerging trend towards virtualizing data center networks to provide guaranteed network bandwidth to each service application. In this context, a key research challenge is to find scalable yet efficient resource allocation schemes that simultaneously allocate both VMs and network resources. Recently a number of proposals have been put forth to address this challenge. In particular, SecondNet [8] is a data center network virtualization architecture that defines virtual data center (VDC) as an abstraction for resource allocation in data center environments. It further provides a greedy heuristic for VDC embedding problem. Similarly, Oktopus [1] proposed two abstractions (virtual cluster and virtual oversubscribed cluster) that can be allocated efficiently in the tree-like data center network topologies. However, both SecondNet and Oktopus do not consider the cost of VM migration in their resource allocation algorithms. Furthermore, none of them have considered energy consumption in their embedding decisions.

The VDC embedding problem also shares many similarities with traditional virtual network (VN) embedding problem. For instance, Chowdhury et. al. proposed algorithms that provide coordinated embedding of both virtual nodes and links [5]. Butt et. al. [6] studied the problem of topology-aware VN embedding and re-optimization that leverages migration techniques. However, VN embedding models differs from VDC embedding in that they only consider CPU and network resources, whereas in VDC embedding other resources such as memory and disk also need to be considered. Finally, minimizing energy consumption has not been addressed in existing VN embedding models.

There is also a large body of work on migration-aware server consolidation in data centers in the past decade. For instance, Entropy [10] is a resource management framework that relies on VM migration to achieve dynamically server consolidation while meeting requirements of all VMs in terms of processing and memory. It models the optimal VM placement as a variant of the vector bin-packing problem, and solves it by means of Constraint Satisfaction Programming (CSP). pMapper [12] is a dynamic server consolidation framework that takes into account VM migration cost. It relies on greedy heuristics to solve the optimal VM placement problem. However, both Entropy and pMapper have not considered network requirement and locality when making consolidation decisions. More recently, Shrivastava et. al. proposed AppAware [11], a network-aware VM migration scheme that minimizes the network distance between communication-dependent VMs while minimizing total migration cost. However, energy consumption is not considered in their framework. Wang et. al. [13] studied the problem of VM consolidation with stochastic bandwidth demands and proposed an online approximation algorithm for the problem. However, VM migration cost is not considered in their model.

III. Models for Migration-aware VDC Embedding

In this section, we present a mathematical formulation of the embedding problem that considers migration. We first introduce the general long-term model from the perspective of an InP. Then, we present the model for the one-shot migration-aware VDC embedding, which is applied upon the receival of any VDC request (either initial embedding or a scale-up request).

A. General Long-term Embedding Formulation

In a nutshell, migration-aware VDC embedding leverages migration techniques to achieve effective and efficient placement of VDCs over time. In this section, we introduce a formal model for migration-aware VDC Embedding. In our model, time is divided into slots of equal duration[1]. Let $\bar{G} = (\bar{N}, \bar{L})$ represents the data center network, where $\bar{N}$

[1]We can adjust the length of time slots to simulate VDC embedding in continuous time.

consists of physical nodes and switches and $\bar{L}$ represents physically links. Define $y_{\bar{n}}(t) \in \{0,1\}$ as a variable that indicates whether physical node $\bar{n} \in \bar{N}$ is active, and $p_{\bar{n}} \in \mathbb{R}^+$ as the cost for using physical machine $\bar{n}$ during each time slot. For instance, $p_{\bar{n}}$ can be the energy cost. Thus, the total cost during time slot t can be computed as

$$\mathcal{C}(t) = \sum_{\bar{n} \in \bar{N}} y_{\bar{n}}(t) p_{\bar{n}} \tag{1}$$

Let $G^i = (N^i, L^i)$ represent the VDC request i, where N^i is set of virtual nodes and L^i represents the set of virtual links. Let I_t denote the set of VDC requests available at time t. More specifically, define D_t as the set of VDC requests arrived at time t, and L_t as the set of VDC request that have left the system at time t (e. g. due to request completion or withdrawal) respectively. We can compute I_t using the following equation:

$$I_{t+1} = I_t \cup D_t \backslash L_t \tag{2}$$

Define $A_t \subseteq I_t$ as the set of running VDCs. Let $m_n(t) \in \{0,1\}$ be an integer variable that denotes whether migration occurs at the end of period t, and $g_n(t)$ denote the cost for migrating $n \in N^i$ at time t, the total migration cost at time t can be computed as

$$\mathcal{M}(t) \quad = \quad \sum_{i \in A_t} \sum_{n \in N^i} m_n(t) g_n(t) \tag{3}$$

On the other hand, for each VDC $i \in I_t \backslash A_t$ that is waiting to be scheduled, we assume there a penalty $p^i(t)$ as function of the duration of time interval t.

$$\mathcal{P}(t) \quad = \quad \sum_{i \in I_t \backslash A_t} p^i(t) \tag{4}$$

We also assume for each VDC i there is a revenue $\mathcal{R}_i(t)$ earned by the InP at time t. According to the current practice, $R_i(t)$ is computed as a weighted sum of the total resources (CPU, memory, disk, bandwidth) used by VDC i at time t. Therefore, the objective of the InP is to maximize the difference between the revenue and the costs, which includes migrations and energy cost, as well well as penalties due to scheduling delays.

$$\max \left(\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \left(\sum_{i \in A_t} \mathcal{R}_i(t) - \mathcal{C}(t) - \mathcal{M}(t) - \mathcal{P}(t) \right) \right) \tag{5}$$

However, this problem is intractable to solve because it requires solving a multi-dimensional bin-packing problem dynamically over time. Even the static version of the problem generalizes the $\mathcal{NP}$-hard multi-dimensional bin-packing problem. Due to its high complexity, it is not possible to solve the problem directly in a timely manner given the large number of physical machines and VDCs in typical production data centers. Therefore, a more scalable yet cost-effective solution is needed.

B. One-shot Migration-aware Embedding Formulation

Since the optimal dynamic VDC embedding problem is difficult to solve, it is necessary to break down the problem based on usage scenarios. In this section, we present a formal model for one-shot migration-aware VDC embedding, whose objective is to deal with either an initial embedding request or a scaling up request. Since we focus on one-shot embedding, we can omit the notion of time in this model.

Specifically, given a data center network $\bar{G} = (\bar{N}, \bar{L})$, let R denote the different types of resources offered by each node (e.g. memory and CPU for servers). Assume each node $\bar{n} \in \bar{N}$ has a capacity $c^r_{\bar{n}}$ for each resource type $r \in R$, and each link $\bar{l} \in \bar{L}$ has a bandwidth capacity $b_{\bar{l}}$. Furthermore, every physical link $\bar{l}$ has a source node and a destination node. We define

$$\bar{s}_{\bar{n}\bar{l}} = \begin{cases} 1 & \text{if } \bar{n} \text{ is the source of } \bar{l} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and

$$\bar{d}_{\bar{n}\bar{l}} = \begin{cases} 1 & \text{if } \bar{n} \text{ is the destination of } \bar{l} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

as boolean variables that indicate whether $\bar{n}$ is the source and destination node of $\bar{l} \in \bar{L}$, respectively. Similarly, we assume there is a set of VDC requests I, each request $i \in I$ asks for embedding a virtual network $G^i = (N^i, L^i)$. We also assume each node $n \in N^i$ has a capacity c^{ir}_n for resource type $r \in R$, and each link $l \in L^i$ has a bandwidth capacity b_l.

Let $x^i_{n\bar{n}} \in \{0,1\}$ be a boolean variable that indicates whether virtual node n of VDC i is embedded in substrate node $\bar{n}$, and $f^i_{l\bar{l}} \in \mathbb{R}^+$ be a variable that measures the bandwidth of edge $\bar{l}$ allocated for virtual link $l \in L^i$. To ensure no violation of the capacities of physical resources, the following constraints must be satisfied:

$$\sum_{i \in I} \sum_{n \in N^i} x^i_{n\bar{n}} c^{ir}_n \leq c^r_{\bar{n}} \qquad \forall \bar{n} \in \bar{N}, r \in R \tag{8}$$

$$\sum_{i \in I} \sum_{l \in L^i} f^i_{l\bar{l}} \leq b_{\bar{l}} \qquad \forall \bar{l} \in \bar{L} \tag{9}$$

We also require link embedding to satisfy the flow constraint between every source and destination node pairs in each VDC topology, formally:

$$-\sum_{\bar{l} \in \bar{L}} \bar{d}_{\bar{n}\bar{l}} f^i_{l\bar{l}} + \sum_{\bar{l} \in \bar{L}} \bar{s}_{\bar{n}\bar{l}} f^i_{l\bar{l}} \quad = \quad \sum_{n \in N^i} x^i_{n\bar{n}} s^i_{nl} b_l - \sum_{n \in N^i} x^i_{n\bar{n}} d^i_{nl} b_l \qquad \forall i \in I, l \in L^i, \bar{n} \in \bar{N} \tag{10}$$

Here $\sum_{n \in N^i} x^i_{n\bar{n}} s^i_{nl}$ is equal to 1 if n is the source of the link l of VDC i and n is embedded in the physical node $\bar{n}$. Equation 10 essentially states that the total outgoing flow of a physical node $\bar{n}$ for a virtual link L^i should be zero unless $\bar{n}$ hosts either source or destination node of virtual link i. Next, we need to consider node placement constraints. We define

$$\tilde{x}^i_{n\bar{n}} = \begin{cases} 1 & \text{if node } n \text{ of VDC } i \text{can be embedded in } \bar{n} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

that indicates whether virtual node n can be embedded in physical node $\bar{n}$. For example, VMs can only be embedded in physical machines rather than switches. Thus, if a virtual node n from VDC i is a virtual server, we have $\tilde{x}^i_{n\bar{n}} = 0 \, \forall \bar{n} \in \bar{N}_s$ and $\tilde{x}^i_{n\bar{n}} = 1 \, \forall \bar{n} \in \bar{N}_m$ where $\bar{N}_s$ and $\bar{N}_m$ are the sets of physical switches and physical machines respectively (i.e., $\bar{N} = \bar{N}_s \cup \bar{N}_m$). Note that the placement constraint can also describe whether a switch can be embedded exclusively in physical switches or in physical servers or in both types of equipments. The following equation captures the placement constraint:

$$x^i_{n\bar{n}} \leq \tilde{x}^i_{n\bar{n}} \qquad \forall i \in I, n \in n, \bar{n} \in \bar{N} \tag{12}$$

To ensure embedding of every virtual node n, we must have:

$$\sum_{\bar{n} \in \bar{N}} x^i_{n\bar{n}} = 1 \qquad \forall i \in I, n \in N^i \tag{13}$$

In our model, we also define $y_{\bar{n}}$ as a boolean variable that indicates whether physical node $\bar{n}$ is active. A node is active if a virtual node of the a VDC runs on the physical node. This implies the following constraints must hold:

$$y_{\bar{n}} \geq x^i_{n\bar{n}} \qquad \forall i \in I, n \in N^i, \bar{n} \in \bar{N} \tag{14}$$

$$y_{\bar{n}} \geq \frac{1}{b_l} f^i_{l\bar{l}} \bar{s}_{\bar{n}\bar{l}} \qquad \forall i \in I, \bar{n} \in \bar{N}, l \in L^i, \bar{l} \in \bar{L} \tag{15}$$

$$y_{\bar{n}} \geq \frac{1}{b_l} f^i_{l\bar{l}} \bar{d}_{\bar{n}\bar{l}} \qquad \forall i \in I, \bar{n} \in \bar{N}, l \in L^i \tag{16}$$

Finally, we also to consider the migration cost. In our formulation, we treat migration cost as a *one-time embedding cost*. The one-time embedding cost $g^i_{n\bar{n}}$ of a virtual node n of VDC i, which is currently embedded in node $\bar{m} \in \bar{N}$ in substrate node $\bar{n} \in N$ is given by:

$$g^i_{n\bar{n}} = \begin{cases} mig(n, \bar{m}, \bar{n}) & \text{if } \bar{n} \neq \bar{m} \\ 0 & \text{if } \bar{n} = \bar{m} \\ 0 & \text{if } n \text{ is currently not embedded} \end{cases}$$

where $mig(n, \bar{m}, \bar{n})$ denotes the cost of migrating virtual node n from substrate node $\bar{m}$ to substrate node $\bar{n}$. Thus, when a virtual node n is already embedded but needs to be migrated from $\bar{m}$ to $\bar{n}$, the one-time embedding cost is equal to the migration cost. This cost is equal to zero when n is already embedded in the physical node $\bar{n}$ (i.e., $\bar{m} = \bar{n}$). Its value is also zero when the node n is embedded for the first time.

Let $p_{\bar{n}}$ denote the cost in dollars of leaving the node $\bar{n}$ active, the goal of the migration-aware embedding can be stated by finding an embedding that achieves

$$\min \sum_{k \in K} \Big(\sum_{\bar{n} \in \bar{N}} y_{\bar{n}} p_{\bar{n}} + \sum_{i \in I} \sum_{n \in N^i} \sum_{\bar{n} \in \bar{N}} \gamma_n x^i_{n\bar{n}} g^i_{n\bar{n}} \Big), \tag{17}$$

subject to equations (7) - (15). Here, γ_n is a weight factor that captures the tradeoff between the migration cost and operational cost. Even though the migration-aware embedding problem is easier than the original online embedding problem, it is still difficult to solve as it still generalize a multi-dimensional bin packing problem.

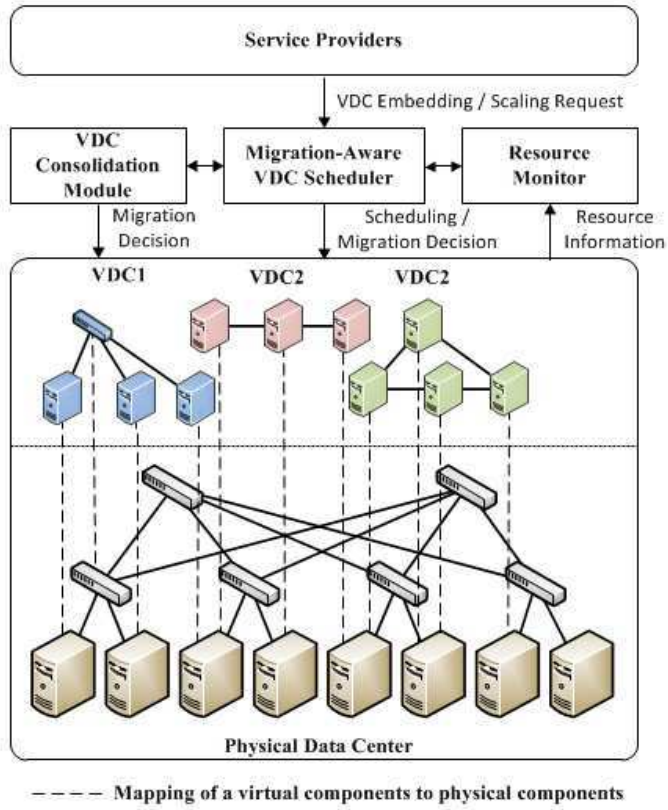


Figure 1: VDC Planner Architecture

IV. VDC Planner

In order to reduce the complexity of the online VDC embedding problem, we have designed VDC Planner, a framework that provides cost-effective VDC embedding in production data centers. Instead of solving the online problem directly, VDC Planner divides the overall problem into several usage scenarios, such that each scenario can be solved effectively and efficiently. We describe hereafter the overall architecture as well as our heuristic algorithms for each scenario.

A. Architecture

The architecture of VDC Planner is shown in Figure 1. It consists of the following components:

- *VDC Scheduler*: Upon receiving a VDC request from a SP, the VDC Scheduler is responsible for scheduling the VDC on the available physical machines. If there is no feasible embedding in data center, the request is kept in a scheduling queue until the SP decides to withdraw it. Different from existing VDC embedding algorithms, our VDC scheduler leverages migration to improve the revenue gain from embedding VDC requests.
- *Resource Monitor*: The Resource Monitor is in charge of monitoring the physical and virtual data centers. It also notifies the VDC scheduler if a failure of any physical or virtual node occurs in the physical data center.
- *VDC Consolidation Module*: The VDC Consolidation Module consolidates the VDCs over time in order to

reduce resource fragmentation (i.e., residual capacities in physical machines and network components that are not capable of scheduling any VDC components). VDC consolidation improves the overall resource utilization of data center and maximizes the number of machines that can be turned off.

Our strategy for reducing the complexity of migration-aware VDC embedding is to divide the overall problem into several "scenarios", such that each scenario can be easily addressed. Figure 2 illustrates the scenarios we consider for VDC Planner. They can be described as follows:

- *Initial VDC Embedding*: A tenant submits a new VDC request and the scheduler has to map it onto the physical data center. When the data center is heavily loaded, it may be impossible to directly embed the VDC due to lack of space. In this case, VM migration can adjust previous resource allocations in order to accommodate the new request.
- *VDC Scaling*: A tenant requests the topology of VDC to be dynamically scaled up and down. For example, if the tenant runs a web application in the data center and it experiences a demand spike, the tenant can submit a request to increase the resource allocation of his VDC. Migration can also be used to increase the chance of satisfying the embedding request, while minimizing the total bandwidth usage for satisfying the requests.
- *Dynamic VDC Consolidation*: As VDCs continuously enter and leave the system, the VDC embedding can become obsolete and suboptimal. We believe it is beneficial to re-optimize the embedding of VDCs at run-time in order to achieve better server and network consolidation. Overtime, this allows more physical servers and network components (e.g., switches and ports) to be turned off to save energy cost [9].

We have developed two heuristic algorithms to support the above scenarios. The first heuristic is designed for migration-aware VDC embedding. It leverages migration to handle VDC embedding as well as scaling up requests. The second heuristic is designed for dynamic VDC consolidation. It also utilizes migration to improve utilization and save energy. We describe each heuristic separately in the following subsections.

B. Migration-Awaref VDC Embedding Heuristic

We describe now our heuristic for migration-aware VDC embedding. Given a VDC embedding request (either an initial embedding or scaling up request), the goal is to find a feasible embedding of the request that incurs minimal migration cost. Our heuristic is depicted by Algorithm 1. Intuitively, upon receiving a VDC request i, the algorithm first sorts the physical machines based on whether they are active or inactive. It then sort virtual nodes in the request based on their size. Specifically, for each $n \in N^i$, we define its size $size_n^i$ as

$$size_n^i = \sum_{r \in R} w^r c_n^{ir}, \tag{18}$$

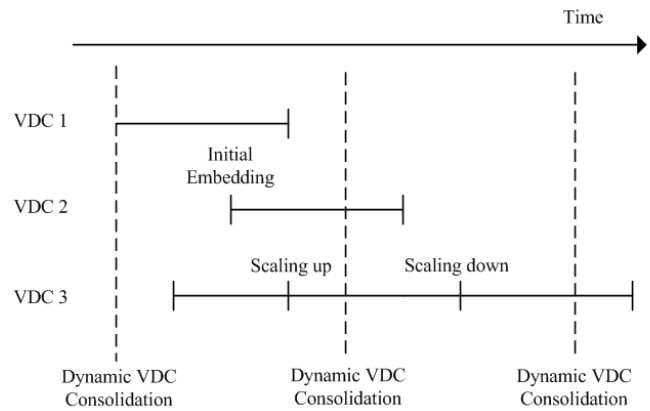


Figure 2: Scenarios for dynamic VDC embedding

where w^r is a weight factor for resource type r. The intuition is that $size_n^i$ measures the difficulty of embedding node n. Accordingly, w^r is selected based on the scarcity of resource type $r \in R$.

After sorting all virtual nodes in N^i according to $size_n^i$, our algorithm then tries to embed each node in the sorted order, based on whether it is connected to any embedded nodes. For each selected node $n \in N^i$ and each physical node $\bar{n} \in \bar{N}$, the algorithm computes the embedding cost $cost^i(n, \bar{n})$ as:

$$\begin{aligned} cost^i(n, \bar{n}) &= \gamma_n(mig(n, \bar{m}, \bar{n}) + MigOther(n, \bar{n})) \\ &\quad + \sum_{n' \in N^i : (n', n) \in L^i} d(\bar{n'}, \bar{n}) \cdot b_{(n', n)} \end{aligned} \tag{19}$$

where the last term represents the communication distance $d(\bar{n'}, \bar{n})$ weighted by the bandwidth requirement $b_{(n',n)}$ between $\bar{n}$ and the other node $n' \in N^i$ that is embedded on physical node $\bar{n'}$. If a particular n' is not embedded, $d(\bar{n'}, \bar{n})$ is set to zero. The intuition here is to minimize the communication distance between virtual nodes in order to reduce bandwidth consumption. In the long run, it also allows more physical network devices to be turned off.

Finally, $MigOther(n, \bar{n})$ is the cost of migrating away the VMs not belonging to G^i on $\bar{n}$ in order to accommodate n on $\bar{n}$. This is similar to the migration plans defined in Entropy. Formally, we denote by $loc(\bar{n})$ the set of virtual nodes hosted on physical node $\bar{n}$. Let $mig(\tilde{n}, \bar{n})$ denote the minimum cost for migrating away $\tilde{n} \in loc(\bar{n})$ to another node that has capacity to host $\tilde{n}$ with minimum distance. Computing $MigOther(n, \bar{n})$ becomes a problem of migrating away a set of node $\tilde{N}$ located on $\bar{n}$ such that there is enough capacity to accommodate n on $\bar{n}$, while minimizing the total migration cost:

$$\begin{aligned} \min_{x_{\tilde{n}} \in \{0,1\}} & \sum_{\tilde{n} \in loc(\bar{n})} x_{\tilde{n}} mig(\tilde{n}, \bar{n}) \\ \text{s. t.} & \sum_{\tilde{n} \in loc(\bar{n})} x_{\tilde{n}} c_{\tilde{n}}^{jr} \geq c_n^{ir} \qquad \forall r \in R \end{aligned}$$

This problem generalizes a minimum knapsack problem [4], which is $\mathcal{NP}$-hard. We adopt a simple greedy algorithm to

Algorithm 1 Algorithm for embedding VDC request i

1: Sort $\bar{N}$ based on their states (active or inactive)
2: $S \leftarrow N^i$
3: **repeat**
4: Let $C \subseteq S$ be the nodes that are connected to already embedded nodes
5: **if** $C == \emptyset$ **then**
6: Sort S according $size_n^i$ defined by equation (18).
7: $n^* \leftarrow$ first node in S
8: **else**
9: Sort C according $size_n^i$ defined by equation (18).
10: $n^* \leftarrow$ first node in C
11: **end if**
12: **for** $\bar{n} \in \bar{N}$ in sorted order **do**
13: Compute embedding cost $cost^i(n^*, \bar{n})$ according to equation (19). If not feasible, set $cost^i(n^*, \bar{n}) = \infty$.
14: **end for**
15: **if** $cost^i(n^*, \bar{n}) = \infty \forall \bar{n} \in \bar{N}$ **then**
16: **return** VDC i is not embeddable
17: **else**
18: Embed n^* on the node $\bar{n} \in \bar{N}$ with the lowest $cost^i(n, \bar{n})$.
19: $S \leftarrow S \backslash n^*$
20: **end if**
21: **until** $S == \{\emptyset\}$

solve the problem. In particular, for a virtual node $\tilde{n} \in loc(\bar{n})$ that belongs to VDC j, we compute a cost-to-size ratio $r_{\tilde{n}}$:

$$r_{\tilde{n}} = \frac{mig(\tilde{n}, \bar{n})}{\sum_{r \in R} w^r c_{\tilde{n}}^{jr}} \tag{20}$$

Then, we sort $loc(\bar{n})$ based on the values of $r_{\tilde{n}}$, and greedily migrate away $r_{\tilde{n}}$ in the sorted order until there is sufficient capacity to accommodate n on $\bar{n}$. The total migration cost of this solution produces $MigOther(n, \bar{n})$. If there is no feasible solution, we set $MigOther(n, \bar{n}) = \infty$. Lastly, for a selected node n^*, once the embedding cost $cost^i(n, \bar{n})$ is computed for every $\bar{n} \in \bar{N}$, we embed n^* on the node with the minimum value $cost^i(^*, \bar{n})$. The algorithm repeats until all nodes in N^i are embedded, or $cost^i(n^*, \bar{n}) = \infty$, which indicates VDC i is not embeddable.

As for the running time of the algorithm, line 4 takes $O(n)$ time to complete as it essentially partitions the physical machines into active and inactive machines. Line 6 and 9 take $O(|N^i|)$ time to execute assuming the number of resource types is constant. Line 13 requires running the greedy algorithm for the minimum knapsack problem. Assume each physical node can host at most n_{max} virtual nodes, the running time of the greedy minimum knapsack problem is $O(|\bar{N}| n_{max})$. The remaining lines each takes $O(1)$ time to run. Thus the total running time of the algorithm is $O(|N^i||\bar{N}| n_{max})$.

C. Dynamic VDC Consolidation Algorithm

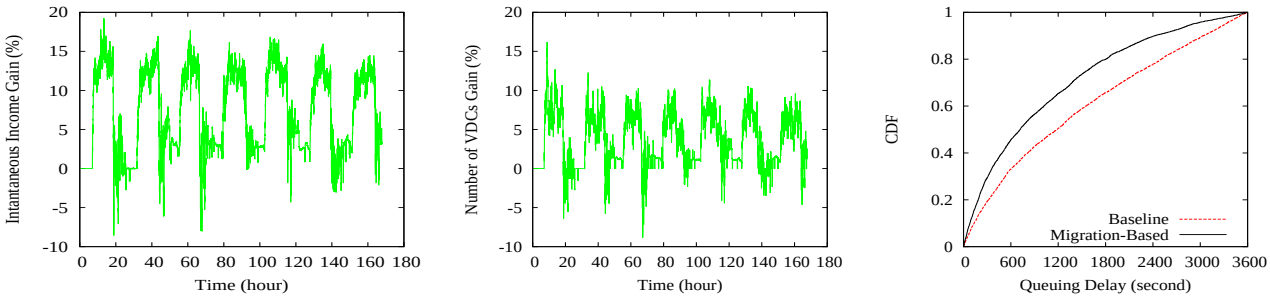
The previous heuristic leverages migration to maximize the number of number of VDC requests. However, as VDC requests can scale down and leave the system over time, a large number of physical nodes may become under-utilized. In production data centers, this typically happens at night time, where the number of VDC requests becomes low. In this case, we would like to dynamically consolidate VDCs such that a large number of physical machines can be turned off. We point out that VDC Planner merely tries to minimize the number of active machines used by VDCs. Deciding the number of machines to be turned on a particular time is a different problem that has been studied extensively (e.g., [16]). Thus existing techniques can be readily applied to control the number of active machines. Our migration-aware dynamic VDC consolidation algorithm is represented by Algorithm 2. Specifically, the algorithm first sort the physical nodes in increasing order of their utilizations. For each $\bar{n} \in \bar{N}$, we define the utilization $U_{\bar{n}}$ as the weighted sum of the utilization of each type of resources:

$$U_{\bar{n}} = \sum_{r \in R} \sum_{i \in I} \sum_{n \in N^i : n \in loc(\bar{n})} \frac{w^r c_n^{ir}}{c_{\bar{n}}^r}, \tag{21}$$

The intuition here is to select the nodes with lowest utilization as candidate for consolidation. Once physical nodes are sorted, for each physical node we sort virtual nodes $n \in loc(n)$ according their size $size_n^i$. Let i denote the VDC n belongs to. We then run Algorithm 1 on VDC i with physical nodes excluding $\bar{n}$. This will find an embedding where $\bar{n}$ is not used, (i.e., n has been migrated to different physical node). Once all the virtual node has been migrated, we compute the cost of the solution according to equation (17) and compare it to the energy saving, which is represented by $p_{\bar{n}}$. If the total saving is greater than the total cost of the solution, migration is performed and $\bar{n}$ becomes inactive. Otherwise, the algorithm proceeds to the next physical node $\bar{n}$ in the list until the

Algorithm 2 Dynamic VDC Consolidation Algorithm

Let $\bar{S}$ represent the set of active machines
repeat
Sort $\bar{S}$ in increasing order of $U_{\bar{n}}$ according to equation (21).
$\bar{n} \leftarrow$ next node in $\bar{S}$
$S \leftarrow loc(\bar{n})$
Sort S according to $size_n^i$ defined in equation (18).
for $n \in S$ **do**
$n \leftarrow$ next node in S. Let i denote the VDC to which n belongs
Run Algorithm 1 on VDC i over $\bar{S} \backslash \{\bar{n}\}$.
end for
$cost(\bar{n}) \leftarrow$ the total cost according to equation (17)
if $cost(\bar{n}) \leq p_{\bar{n}}$ **then**
Migrate all virtual nodes according to Algorithm 1
Set $\bar{n}$ to inactive
end if
$\bar{S} \leftarrow \bar{S} \backslash \{\bar{n}\}$
until $U_{\bar{n}} \geq C_{th}$



(a) Instantaneous income gains (b) Gain in acceptance ratio (c) Gain in terms of queuing delay

Figure 3: Migration-aware embedding vs. baseline algorithm

cluster is sufficiently consolidated (i.e., all the machines in the cluster have reached a threshold C_{th}. Using the algorithm, the VDC consolidation component is able to make dynamic consolidation decisions that considers migration cost.

Finally we analyze the running time of the algorithm. Line 3 takes $O(|\bar{N}|)$ time to complete assuming the number of resource types is constant. Line 6 takes $O(n_{max}\log(n_{max}))$ time to complete. Line 9 runs algorithm 1, which takes $O(|N^i||\bar{N}|n_{max})$ time to complete. Line 13 takes $O(n_{max})$ time to finish. The remaining lines each takes $O(1)$ time to complete. Thus the total running time of the algorithm is $O(|\bar{N}|^2\log|\bar{N}| + |\bar{N}|n_{max}^2 N_{max})$ assuming the maximum of virtual nodes per VDC is N_{max}.

Lastly, we need to answer the question that when should VDC consolidation be performed at run time. A naïve solution is to perform VDC consolidation periodically. However, we have found that periodic consolidation may not be beneficial when request arrival rate is high. In this case, even if we can reduce the number of active machines for a particular time instance, the high arrival rate of new VDC requests will force more machines to be active, rendering the consolidations effort ineffective. Motivated by this observation, we perform VDC consolidation only when arrival rate is low over a period of time (i.e., below a threshold λ_{th} requests per second over T minutes). Even though more sophisticated techniques such as predicting the future arrival rate allows for more accurate consolidation decisions, we have found this simple policy achieves a good balance between migration cost and energy cost at run time in our experiments.

V. Experiments

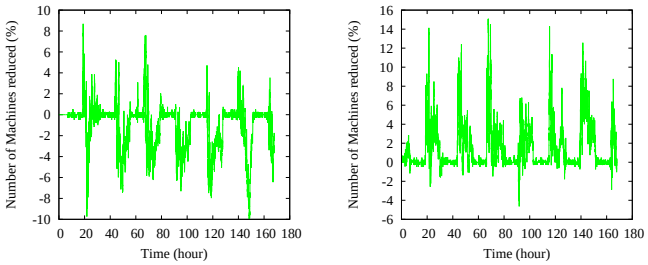
We have implemented VDC Planner and evaluated its performance through simulation studies. Specifically, we have simulated a data center with 400 physical machines organized in 4 racks. The topology used in our experiment is the clos topology described in VL2 [7], which provides full bisection bandwidth in the data center network. To implement this topology, we have also added 4 top-of-rack switches, 4 aggregation switches as well as 4 core switches in our topology.

In our experiments, VDC requests arrive following a Poisson distribution with an average rate of 0.010 requests per second during night time and 0.020 requests per second during day time. This reflects the time-of-the-day effect where resource demand is higher during day time. For convenience, we set $\gamma_n = 1$, and $\lambda_{th} = 0.015$. In practice, the value of λ_{th} can be obtained through experience. The number of VMs per VDC is generated randomly between 1 and 20. In our simulations, each physical machine has 4 CPU cores, 8GB of memory, 100GB of disk space, and contains a 1Gbps network adapter. The size of each VM for CPU, memory and disk are generated randomly between $0-4$ cores, $0-2$GB of RAM and $0-10$GB of disk space, respectively. The bandwidth requirement between any two VMs that belong to the same VDC is generated randomly between 0 and 10Mbps. Furthermore, the lifetime of VDCs follows an exponential distribution with an average of 3 hours. In our implementation, a VDC can wait in the queue for a maximum duration of 1 hour after which it is automatically withdrawn.

In our first experiment, we evaluated the revenue gain achieved when using the migration-aware embedding algorithm compared to a baseline algorithm similar to SecondNet that does not consider VM migration and energy-aware VDC consolidation. Let R_m and R_n denote the infrastructure provider's income over a period of time using the migration-aware algorithm and the baseline algorithm, respectively. The revenue gain is defined as

$$G_{m/n} = 100 \times \frac{R_m}{R_n} - 100. \tag{22}$$

The same formula is used to compute the gain in terms of request *acceptance ratio* (i. e. successfully embedded VDC requests divided by the total number of received VDC requests), and the number of inactive machines. Figure 3a and Figure 3b show the instantaneous revenue gain and the increase in acceptance ratio, respectively. Every point in each figure represents the gain over a 1 minute interval. It can be seen that from midnight till the morning, the migration-aware algorithm is providing the same revenue as the baseline approach. However, during the day time when resource demand is high, the migration-aware embedding algorithm can achieve an increase up to 17% in revenue gain over the baseline approach. This is expected since during idle periods (e. g., night time), it is easy to embed VDC requests given the



(a) Migration-aware algorithm (b) Migration-aware embedding + consolidation

Figure 4: Change in number of machines

ample free capacities in the data center. However, when the cluster is busy, it becomes difficult to embed VDC requests directly. In this case, migration-aware approach is able to leverage migration to find more embeddable VDCs. This result is also confirmed by Figure 3b which shows an improvement of 10% in terms of VDC requests acceptance ratio during busy periods. We also compare the queuing delay experienced by the VDC requests. Figure 3c compares the cumulative distribution function (CDF) of scheduling delays achieved by each of the studied algorithms. It is clear that the migration aware approach significantly reduces the average scheduling delay. In particular, our algorithm can reduce the scheduling delay by up to 25%.

On the other hand, even though the migration-aware embedding algorithm is able to improve InP's revenue, at the same time it also uses more physical machines to host embedded VDCs than the baseline algorithm, as shown in Figure 4a. Therefore, we also implemented the dynamic VDC consolidation algorithm as described in Section IV-C to reduce the number of physical machines used. Compared to Figure 4a, Figure 4a shows that percentage reduction in number of machines can reach 14% when the consolidation algorithm is applied along with the migration-aware embedding. The benefit is apparent especially at night (8pm to midnight) when VDCs are leaving the system. We also notice that, by reducing the number of active machines, consolidation also improves the income as defined in Eq. 5. Combining consolidation with the migration-aware embedding technique, we found on average VDC Planner achieves a 15% increase in net income compared to baseline algorithms that resembles the existing algorithms proposed in the literature (e.g., SecondNet).

VI. Conclusions

Driven by the need to support network performance isolation and QoS guarantee in public clouds, recently a number of research proposals have proposed to allocate resources to SPs in the form of virtual data centers that include both guaranteed server resources and network resources. A key challenge that needs to be addressed in the context is dynamic VDC embedding problem, which aims at optimally allocating both servers resources and data center networks to multiple VDCs to optimize total revenue, while minimizing the total energy consumption in the data center. However, despite recent studies on this problem, none of the existing solutions have considered the problem in a dynamic and online setting, where migrations can be utilized to flexibly and dynamically adjust the allocation of physical resources.

In this paper, we have described VDC Planner, a migration-aware dynamic virtual data center embedding framework that aims at achieving high revenue while minimizing the total energy cost over-time. Our framework supports various scenarios, including VDC embedding, VDC scaling as well as dynamic VDC consolidation. Through simulation experiments, we show our proposed approach is able to achieve higher net income as well as lower scheduling delay compared to existing solutions in the literature.

References

[1] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. *in Proceedings of ACM SIGCOMM*, August 2011.

[2] MF. Bari, R. Boutaba, R. Esteves, L. Granvilley, M. Podlesny, M. Rabbani, Q. Zhang, and M. F. Zhani. Data center network virtualization: A survey. *IEEE Communications Surveys and Tutorials*, Aug. 2012.

[3] T. Benson, A. Akella, A. Shaikh, and S. Sahu. Cloudnaas: a cloud networking platform for enterprise applications. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 8. ACM, 2011.

[4] T. Carnes and D. Shmoys. Primal-dual schema for capacitated covering problems. *Int. Prog. and Combinatorial Optimization*, 2008.

[5] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. Netw.*, 20(1):206–219, 2012.

[6] Nabeel Farooq Butt, Mosharaf Chowdhury, and Raouf Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. In *IFIP International Conference on Networking*, 2010.

[7] 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. In *Proceedings ACM SIGCOMM*, August 2009.

[8] Chuanxiong Guo, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International Conference Co-NEXT*, 2010.

[9] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *Proceedings USENIX NSDI*, April 2010.

[10] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pages 41–50, 2009.

[11] V. Shrivastava, P. Zerfos, Kang won Lee, H. Jamjoom, Yew-Huey Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *Proceedings of IEEE INFOCOM*, pages 66 –70, april 2011.

[12] A. Verma, P. Ahuja, and A. Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264. Springer-Verlag New York, Inc., 2008.

[13] M. Wang, X. Meng, and L. Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *INFOCOM*, 2011.

[14] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53:2923–2938, December 2009.

[15] Jing Xu and José Fortes. A multi-objective approach to virtual machine management in datacenters. In *Proceedings of the ACM international conference on Autonomic computing (ICAC)*, pages 225–234, 2011.

[16] Qi Zhang, Mohamed Faten Zhani, Quanyan Zhu, Shuo Zhang, Raouf Boutaba, and Joseph L. Hellerstein. Dynamic energy-aware capacity provisioning for cloud computing environments. In *IEEE/ACM International Conf. on Autonomic Computing (ICAC)*, 2012.

[17] Xiangliang Zhang, Zon yin Shae, Shuai Zheng, and Hani Jamjoom. Virtual machine migration in an over-committed cloud. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2012.