Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 780 – 794

doi:10.3906/elk-1802-166

Efficient virtual data center request embedding based on row-epitaxial and batched greedy algorithms

Sivaranjani BALAKRISHNAN[1,*], **Surendran DORAISWAMY**[2]

[1]Department of Computer Science and Engineering, Government College of Engineering, Bodinayakkanur, Theni Dist, Tamil Nadu, India

[2]Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, Tamil Nadu, India

Received: 20.02.2018 • **Accepted/Published Online:** 14.11.2018 • **Final Version:** 22.03.2019

Abstract: Data centers are becoming the main backbone of and centralized repository for all cloud-accessible services in on-demand cloud computing environments. In particular, virtual data centers (VDCs) facilitate the virtualization of all data center resources such as computing, memory, storage, and networking equipment as a single unit. It is necessary to use the data center efficiently to improve its profitability. The essential factor that significantly influences efficiency is the average number of VDC requests serviced by the infrastructure provider, and the optimal allocation of requests improves the acceptance rate. In existing VDC request embedding algorithms, data center performance factors such as resource utilization rate and energy consumption are not taken into consideration. This motivated us to design a strategy for improving the resource utilization rate without increasing the energy consumption. We propose novel VDC embedding methods based on row-epitaxial and batched greedy algorithms inspired by bioinformatics. These algorithms embed new requests into the VDC while reembedding previously allocated requests. Reembedding is done to consolidate the available resources in the VDC resource pool. The experimental testbed results show that our algorithms boost the data center objectives of high resource utilization (by improving the request acceptance rate), low energy consumption, and short VDC request scheduling delay, leading to an appreciable return on investment.

Key words: Data center, virtual data center, VDC request embedding, row-epitaxial algorithm, batched greedy algorithm, return on investment

1. Introduction

Corporations, government departments, and private parties are seeking computing, storage, memory, and network services from cloud providers. Government and private service sectors are now extensively dependent on cloud resources for their advantages, which are on-demand and automated resources, fair billing, and high availability. Infrastructure providers (InPs) lease their data center resources to service providers who provide the cloud resources.

As the demand increases for data center resources, InPs could scale their resources to satisfy the customers' requirement without measuring the existing resource utilization. In contrast, this situation is also an opportunity to evaluate the utilization rate of the data center's components to increase its efficiency.

Virtual data centers (VDCs) are emerging as a solution for effective resource utilization. A VDC provides virtualization of resources such as computation, memory, storage, and networking. Amazon provides VDC

*Correspondence: bsranjani@gmail.com

resources via the Amazon Virtual Private Cloud [1]. The benefits of a VDC are full automation, highly available resources, flexible infrastructure allocation, manageable architecture through online portals from anywhere, support for geographically distributed data centers, and easy and agile virtual machine (VM) migration. It also enables users to build private and public clouds. A VDC offers a dedicated data center, improved business agility, decreased workload for information and technology (IT) personnel, optimization of the process, reduction in operational costs, high availability, and support for continuous integration of the physical data center.

Infrastructure as a service (IaaS) is offered by InPs as a standard cloud service model. This model allocates the physical components to cloud users based on their requests. Cloud resource allocation is performed on the premises of the InP. Allocation of physical resource strategies differs for physical data centers and VDCs. As far as the end customer is concerned, the request is posted in a queue and waits for the data center components to be allocated.

The software-defined data center has evolved as an approach to the virtualization of networking, storage, and security. Its aim is to offer IT resources and applications [2]. A software-defined data center provides the ability to rapidly match workloads with hardware components from a pool of infrastructure components. In addition, it employs optimal embedding techniques for the VDC, which is the logical framework of the physical data center and can be fabricated according to the needs of customized user requests.

The main challenge to improving the performance is to efficiently map VDC requests to the physical data center components. Multiple strategies have been proposed for achieving the optimal allocation of VDC requests to data center components [3–8]. Because these studies mainly concentrate on servicing requests, in our study, performance factors are taken into consideration to improve data center operation.

Moreover, according a Natural Resources Defence Council report [9], data centers were the major consumers of electricity in the United States in the year 2013, consuming an estimated 91 billion kWh. The report projected an annual US consumption of 140 billion kWh by the year 2020. This energy is equivalent to the yield of 50 power stations and produces nearly 100 million metric tons of carbon footprint per year, which leads to severe environmental consequences.

The continuous process of allocating of requests incoming to a data center is a nondeterministic NP-hard problem because the VDC governing parameters change over time and scheduling cannot be solved in polynomial time. In this paper, we propose two dynamic programming-based VDC embedding methods to solve the allocation of incoming requests to a VDC. The main motive is to explore a new direction for VDC embedding by improving the resource utilization rate and optimal energy utilization. Row-epitaxial and batched greedy algorithms are used in this study. These algorithms are used in bioinformatics for placement and reembedding [10, 11].

The paper is organized as follows: Section 2 surveys related work based on VDC embedding techniques. In Section 3, we present our algorithms for VDC request mapping based on row-epitaxial and batched greedy algorithms. In Section 4, we evaluate the performance of our algorithms within a testbed environment. Finally, we conclude our study and discuss future work in Section 5.

2. Related work

Many algorithms have been developed to address the issues of resource underutilization. The VDC Planner [3] and Venice [4] methods find the best fitting algorithm for VDC request embedding into a physical data center using fat-tree and star topologies, respectively. However, the utilization of existing resources is not taken into consideration. Several studies have been devoted to finding the application mapping algorithm for physical

resources in a multidimensional resource scenario [12]. Zuo et al. proposed a reliable algorithm for VDC embedding by reducing the bandwidth consumption by up to 30% [5]. To reduce the bandwidth consumption, the migration cost is increased. In this scenario, balancing the load among the physical machines is a drawback.

Li et al. developed a strategy to reduce the virtual switching of data center traffic during VM migration to increase data center performance [13]. Yang et al. proposed a computationally and energy-efficient VDC embedding algorithm to reduce the energy consumption of growing data centers and reduce the embedding cost by selecting collocated physical servers. However, this approach suffers from CPU overhead because similar jobs, such as compute-intensive jobs, are submitted to the same physical machine [6].

The VDC bandwidth consumption can also be reduced by splitting the embedding problem into three subproblems, namely VM placement, VDC clustering, and virtual edge allocation [7]. The ViDCAlloc algorithm increases the request acceptance rate and turnover of the service provider and is much faster than existing algorithms such as LoCo [14] and SecondNet [15]. The queuing time of requests is increased in these approaches because allocation into the physical data center is slower.

Roohitavaf et al. [16] proposed an availability metric in the VDCs. This ensures that the VDC services are available for the incoming requests and avoids service denial. This storage area network model does not take any of the data center configurations such as topologies, VMs, virtual links, or switches into account in the experiment. The authors of [17] proposed a cloud management framework to save energy in VDCs by dividing the data centers according to their type of VM. This minimizes the carbon emissions by mapping the VDCs to the corresponding green data centers.

Amokrane et al. proposed the Greenhead algorithm, which improves the acceptance rate of the InP within a green computing environment. Their work emphasizes renewable energy usage to reduce the carbon footprint. The authors used a VDC partitioning strategy based on the bandwidth requirements among VMs before allocating the resources [8]. This algorithm converges the resource allocation slowly.

Kliazovich et al. used techniques such as dynamic voltage and frequency scaling, dynamic voltage scaling, and dynamic network shutdown for obtaining high energy-saving results in data centers [18]. These techniques were applied to two- and three-tier high-speed data centers, obtaining significant reduction in energy consumption. This approach lacks an efficient mapping of resources to physical counterparts.

Zou et al. proposed a multipath algorithm for embedding requests into the VDC. This algorithm chooses resources by routing the requests to all data center resources. However, it suffers from unreasonably high bandwidth use within the data center [19].

Google first reported its power usage effectiveness (PUE) for all of its six data centers in 2008 as 1.22, which is very close to optimal (the ideal PUE is 1.0). Google is further striving toward energy efficiency by including all of the overheads incurred in the data centers. It achieved a PUE of 1.12 in the year 2015. Microsoft also reported a PUE of 1.22 in 2008. Data centers find it difficult to achieve a good PUE, which proves their energy efficiency for green computing goals. Data centers globally have an average PUE of more than 2.0, which could be reduced to attain better data center performance.

Kansal and Chana proposed an algorithm for resource utilization in cloud computing based on an artificial bee colony. It primarily concentrates on allocating resources to the available data center components. This algorithm lacks a metric for the performance of the data center in terms of the optimal allocation of resources and request wait time [20].

The major shortcomings of existing VDC request embedding methods are a lack of data center operation

efficiency in terms of acceptance rate, resource utilization, resource allocation concurrency, and energy use. Improving these factors will substantially improve the return on investment (ROI) of the InPs while also fulfilling service-level agreements.

3. Proposed methodology

The main aim of the algorithm developed in this study is to enhance the fair usage of resources. A 2012 Gartner report stated that this utilization rate is below 12%. The main reason for this underutilization is unmanaged data center resources [21].

The main aim of the service provider or the InP is to earn revenue from the existing setup without further investment. At the same time, they must guarantee services according to quality of service parameters defined through an SLA. The best possible solution is for the VDC to balance the existing setup usage and consumer satisfaction as per the SLA. Virtualizing the entire data center is performed by virtualizing distinct components such as memory, CPUs, network, and storage.

An essential part of a VDC is the method of allocating resources to users [22, 23]. It is mandatory to concentrate on the request allocation scheme so that the ROI will be improved. If an optimal scheduling algorithm is developed, then the InP can attain better resource utilization and hence earn more revenue without further investment.

3.1. Problem description

To attain good ROI through high utilization rates, we propose practical solutions for improving the VDC request embedding into a physical data center. It yields good ROI for the InP by increasing the number of requests serviced, optimizing energy use, and reducing the wait time of requests. Our algorithm is capable of reembedding old requests while allocating resources for new requests to improve performance.

The ROI calculation includes the main management costs, such as the cost of energy utilized for IT equipment, cooling equipment, software license renewals, Internet, and labor.

$$Maximize\ ROI = \frac{Net\ profit\ from\ data\ center}{Investment\ cost\ of\ the\ data\ center} \times 100 \tag{1}$$

The main objective of our algorithm is to increase the net profit from the data center. The net profit is the difference between the revenue and investment cost of the data center. Initially, the revenue is very small compared to the investment cost. However, once the data center reaches stability, it starts yielding a good ROI, which is calculated over particular intervals such as a month or year.

The total cost of ownership consists of the cost of the space, hardware (computation and networking), cooling equipment, and the operation of the data center.

$$DC\_Cost_{total} = Cost_{space} + Cost_{hardware\ and\ equipment} + Cost_{depreciation} + Cost_{power} + Cost_{IT\,Personnel,license} \tag{2}$$

The first two factors (space, hardware, and cooling equipment costs) are capital investment costs that are incurred once.

Operational cost plays a major role in determining the revenue of the data center. It can be reduced through careful operation of the data center. The operational cost includes the cost of power usage, IT personnel, depreciation of hardware, and software licensing renewal.

The net profit is calculated from the prices for the services provided by the data center. The pricing depends on the computation, memory, storage, network, and bandwidth provisions agreed through the SLA as well as the revenue generated from servicing the customers. The number of customer requests serviced is directly proportional to the net profit of the data center. If we employ an efficient and parallel algorithm to allocate and service VDC requests, then the request acceptance rate will directly increase.

3.2. System architecture

A classic data center comprises computing servers, top-of-rack (TOR) switches, aggregation switches, and core switches. The speed of the links between networking components increases when it generalizes to the top. VMs are created in the computing servers and used to form the VDC. Until the VMs are created (on demand), the servers are switched off using a dynamic network shutdown algorithm that shuts down the networking components that are not in use. The rack servers are provided with two access switches to provide fault tolerance to the physical machines. The access switches are cheaper than Layer 3 switches, which also provide the access support in the case of switch failures. The organization of a classic data center is illustrated in Figure 1 based on fat-tree topology [24].

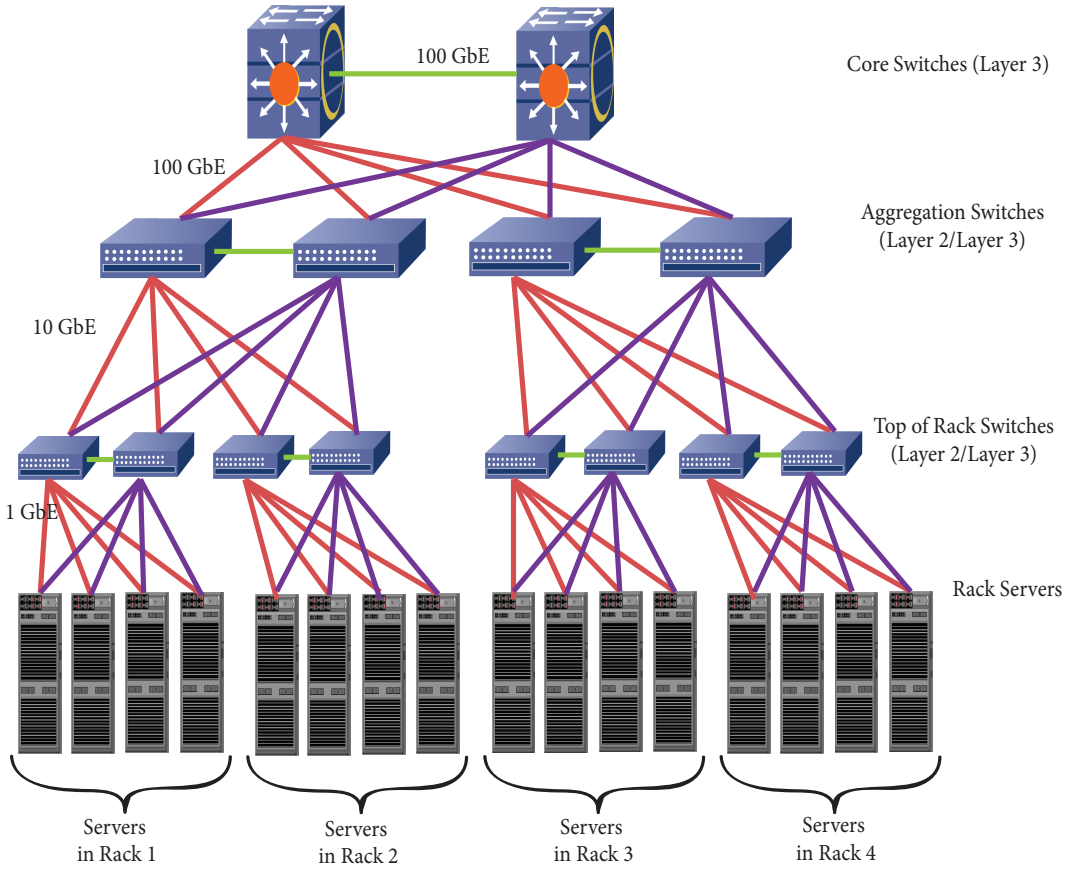

Figure 1. Classic fat-tree three-tier data center architecture.

Each VDC is created to have the required VMs from the available physical servers. All VMs in a VDC are connected to the VDC through virtual switches. A virtual switch is mapped or configured using one or two physical switches in various layers to access VMs in different servers.

An incoming request can be formulated as a specification that includes the number of VMs(n), their platform (VM_ρ), their software requirements (VM_δ), interconnecting switch speeds ($vSwitch_\tau$), approximate duration of the VDC (VDC_α), and IP address range ($IP_\beta - \gamma$) as follows:

$$Request_{VDC} = n * VM_{(\rho,\delta)}, vSwitch_\tau, VDC_\alpha, IP_\beta - \gamma \tag{3}$$

The virtual links are mapped to physical links for communication once the data center agrees to host the VDC. The created VDCs should be independent and isolated from existing VDCs because they include VMs from many physical servers. Before accepting a VDC request, it is necessary to check resource availability in the data center resource pool. This feasibility check eliminates the need for requests to wait in a queue unnecessarily. The main aim of optimal VDC embedding is to reduce the embedding cost. A VM is selected using the proposed algorithm and assigned to the data center. Next, an unmapped VM is selected from the VDC request and placed in the data center with a link to the existing embedded VMs.

3.3. Proposed algorithms

Requests need to be allocated their assigned VDC without any interruption until their completion. Our proposed algorithms find the optimal VDC assignment for the incoming request that reduces the allocation time. They also reduce the wait time of the requests in the queue and yield a high acceptance rate. The algorithm has some prerequisites.

If all possible embeddings are considered, then the problem is even more complicated. Hence, the problem has been solved in two phases. First, the initial embeddings of the requests are determined and an assignment of these embeddings within the VDC is found. This is usually referred to as the placement at the beginning. In the second phase, a post-placement optimization rearranges the requests in the data center so that conflicts with the neighboring VDCs are reduced.

Better results could be achieved if the placement of new VDC requests and reembedding of previously placed VDC requests are considered together, not separately. Because of the high number of embeddings of each data center in a heterogeneous environment, it is very hard to devise algorithms that make optimum use of this additional task and attain reasonable running times in real time. In this paper, we propose a strategy called the row-epitaxial algorithm to place and reembed the VDC requests at the same time.

3.4. Row-epitaxial algorithm

The epitaxial algorithm places a random request in the center of the servers and proceeds to occupy the servers adjacent to the already-filled servers. This technique reduces the energy wasted by running servers without a VM allocation. Priority is given to the physical servers with the largest number of filled neighboring servers. At each consolidation phase, it finds all empty servers and locates an unassigned request with the minimum distance to the filled neighbors. This algorithm employs a greedy technique to select the next server to be assigned.

Algorithm 1 both allocates the new requests and concurrently uses Algorithm 2 to tune the performance of the data center by consolidating the servers and VMs.

A data center is a large-scale environment with a huge number of computing resources. In the row-epitaxial algorithm, the VDC requests are placed in a predefined order because placing the VDC requests dynamically causes practical complications such as long wait times and unused resources.

There are three main distinguishing features of the row-epitaxial algorithm.

1. The algorithm reshuffles an existing preoptimized VDC placement rather than starting with an initial placement.

2. The physical servers are occupied by new VDCs in a fixed order, namely rack by rack from left to right, so that each new VDC is adjacent to two already-embedded neighbor VDCs.

3. The VDC for embedding in the next physical server is chosen from a limited number of unembedded VDCs that lie in close proximity.

Features 1 and 3 significantly increase the quality of the row-epitaxial algorithm. Feature 2 limits the runtime of the row-epitaxial algorithm by $O(n^2)$.

Algorithm 1 Row-epitaxial embedding algorithm for VDC request R.

Data: VDC request as set of VMs (VM_0, VM_1, VM_2, VM_3), edges (E_0, E_1, E_2, E_3), virtual switch (vS_1)- R
Result: VMs VM_0, VM_1, VM_2, VM_3 to be used as end nodes along with the connecting edges to virtual switch

Randomly select VM VM_0 and E_0 in R
Choose $VM_1, E_1 \in$ R minimizing distance d(VM_1, VM_0)
Choose $VM_2, E_2 \in$ R minimizing d(VM_2, VM_0) + d(VM_2, VM_1)
Choose $VM_3, E_3 \in$ R minimizing d(VM_3, VM_0)+d(VM_3, VM_1)+d(VM_3, VM_2)
Return $VM_0, VM_1, VM_2, VM_3, E_0, E_1, E_2, E_3, vS_1$ with instance id and ip addresses

Experimental results show that the row-epitaxial algorithm is the best large-scale placement algorithm, achieving up to a 9% reduction over other placement algorithms in the number of physical servers used [3, 4, 8, 23]. The main issue with the traditional place and reembed approach is that the allocation of VDCs in the physical server is decided on the basis of embeddings that are likely to change during the reembedding phase.

Algorithm 2 Reembedding algorithm for VDCs.

Data: VDC R and the neighboring VDC R_n; Data center consisting of columns col_{left} to col_{right} and rows row_{top} to row_{bottom}
Result: VMs in R are placed in row-epitaxial fashion

Let Q = R $\cup R_n$
For i = row_{top} to row_{bottom}
For j = col_{left} to col_{right}
Let $p_{i,j}$ = q
Q = Q/ q

3.5. Batched greedy algorithm

The main idea of the batched greedy algorithm is to design an algorithm that is similar in process to the row-epitaxial algorithm but that better estimates the gains resulting from combining the placement and reembedding phases.

To optimize VM embedding, we implemented a greedy algorithm (Algorithm 3). This algorithm finds a physical machine for a VM that offers the largest cost gain with respect to the optimal reembedding of the fixed embeddings of its neighbor. The algorithm performs the reembedding, updates the gain, and repeats

the reembedding process again. A faster batched version of the greedy algorithm moderately sacrifices its greedy character in favor of runtime by occasional gain updates. In other words, during a first batched phase, we reembed the VMs in a greedy manner according to the cost gain from the rearrangement. However, the algorithm does not update the gain while there are still VMs with positive unchanged gains.

This algorithm optimally reembeds the VM that yields the largest decrease in embedding cost until no further decreases are possible. To improve the runtime, the greedy selections of the new VMs are made in phases, in a batched manner. In each phase, the cost gains for all VMs are computed and then a set of nonadjacent VMs is selected for assignment by traversing the allocated VMs in a decreasing order of gain.

The batched greedy algorithm fills the physical servers rack by rack from left to right in a greedy fashion. For each server s, it looks at requests and selects the request that can be placed at s with minimum gain. All possible embeddings of request ρ are considered instead of only its initial embedding. This can be accomplished by temporarily placing request ρ in server s and computing its optimal embedding cost with respect to the already-filled neighbor servers of s.

The batched greedy algorithm is derived from the Steiner tree (Figure 2) and spanning tree methods. A Steiner tree is a tree spanning the terminals with additional points for a set of terminals, called Steiner points. If all terminals are leaves, then it is full Steiner tree. An optimal Steiner tree is a set of three points (2 VMs and a switch). The minimum network interconnecting all the VMs in the VDC is called a minimum Steiner tree. The Steiner minimum trees may include nonallocated vertices. Those vertices are called Steiner points (StPts). A k-restricted Steiner tree is a full component of T that has maximum k terminals [11]. Algorithm 4 generates a greedy triple Steiner tree.

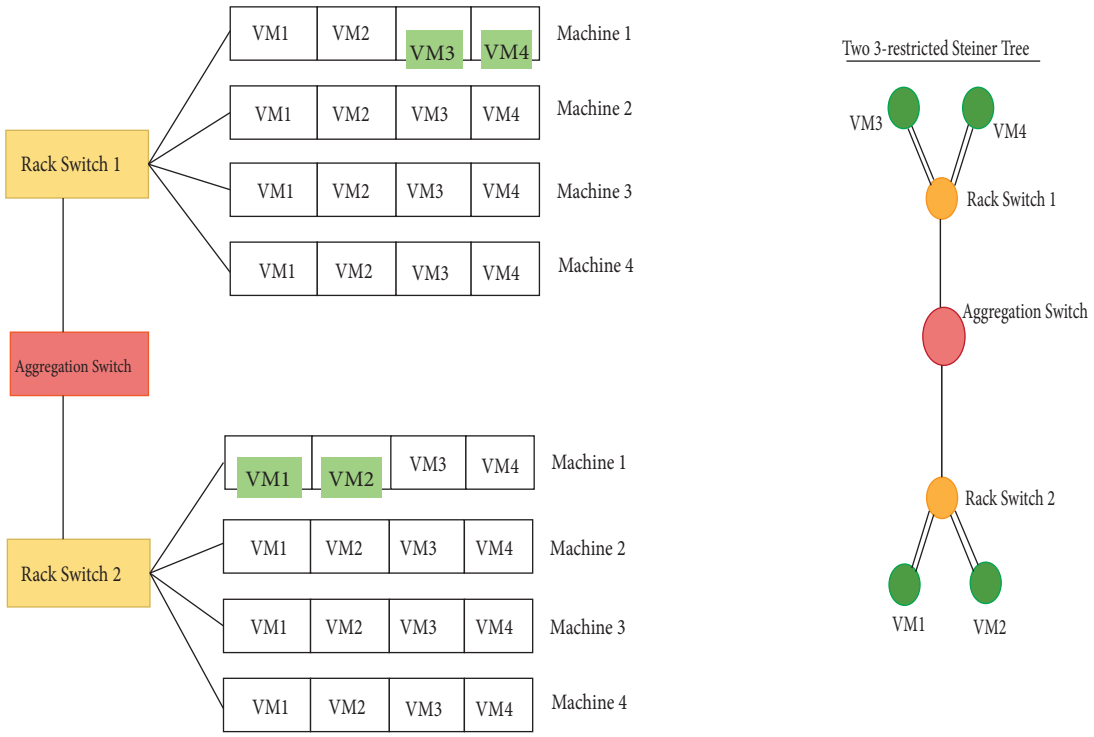

Figure 2. Steiner tree for VDC distribution.

BGA finds a 3-restricted Steiner tree by choosing restricted components that reduce the cost of the minimum spanning tree (MST) using the greedy method. Let G_S be the VDC graph on a given set S of vertices, and let MST(S) be a MST of G_S. To obtain the MST of the given VDC graph, the most expensive link should be removed at each phase.

Let E_1 and E_2 be the two virtual links that must be removed and let $\mathrm{R}(\tau) = E_1, E_2$. The cost gain of τ is gain(τ) = cost(R(τ))-cost(τ).

Algorithm 3 Batched greedy algorithm for VDC embedding.

Data: Set of VMs in a VDC request
Result: Tree T with VMs, links, and switches, spanning tree S with all VMs and minimum links

1: Compute MST(S) for the selected VMs and virtual links
2: Compute a set VDC triples of size $O(nlogn)$ containing all triples of partially completed VDCs
3: StPt $\rightarrow \varnothing$
4: **while** *VDC Triples* $\neq \varnothing$ **do**
  For each $\tau \in$ VDC Triples
   compute R(τ),A(τ) and gain(τ), discarding triples with negative gain
  Sort VDC triples in descending order of gain
  Unmark all virtual links of MST(S) of the given VDC graph
  For each $\tau \in$ VDC Triples do two virtual links in A(τ), i.e. MST(S)$\leftarrow$ MST(S) - R(τ)+A(τ)
  StPt $\leftarrow$ StPt + center(τ)
end
5:**if** *StPt= null* **then**
  return the VDC Graph MST of S
else
  S$\leftarrow$ S + StPt
  Compute the final MST of S and remove all Steiner points with degree less than 2
  Go to Step 2
end

The batched greedy algorithm is slower to evaluate the embedding cost for each VDC request ρ for a server s than the row-epitaxial algorithm. The space and time complexity of this algorithm is $O(n)$ and $O(C * nlog_2 n)$, respectively. C is the total number of grouped reembedding cycles and n is the number of VMs. However, in practice, the total number of cycles C is very small and almost a constant. To reduce the time for this algorithm, the number of requests waiting in the queue must be reduced.

According to the batched greedy algorithm, if the resources are available for the incoming request, the request will be allocated immediately without any scheduling delay. If resources are partially available, the request has to wait until the consolidation algorithm pulls the idle resources of other VDCs into the resource pool. The scalability of the VDC is highly improved because it pulls unused resources from idle VDCs and pushes them to VDCs that need them. Based on our experiments, the energy consumed by the data center before and after running the algorithm is almost constant even though the number of requests serviced is increased.

Algorithm 4 Greedy triple generation algorithm.

Data: Set S of VM terminals
Result: 3-Restricted Steiner tree T, spanning tree S

1: T $\leftarrow$ MST(G_S)
2: F$\leftarrow$ null
3: Repeat Forever Find a VM triple τ with maximum cost gain
    If gain(τ) $\leq$ 0, then go to Step 4
    F$\leftarrow$ F $\cup$ τ //Include τ
    T $\leftarrow$ T-Rτ + A(τ)
4: Output MST(F $\cup$ VDC Graph MST(GS))

4. Performance evaluation

4.1. Simulation setting

Physical data center configuration: We evaluated the performance of our algorithm through test bed experiments. In our test bed environment, we used 128 identically configured physical servers, 16 TOR switches (each connected with eight physical servers), four aggregation switches, and two core switches on the Internet. Each physical machine consisted of four CPU cores, 8 GB of memory, 240 GB of disk space, and a Gigabit Ethernet adapter. Each VM was connected to two TOR switches to provide fault tolerance in the case of TOR failures.

VDC specifications: The data center for testing the algorithm was placed at one location. Each VDC was created with a capacity of 3–10 VMs before allocating any requests. Each VM size could vary from 1 to 4 CPU cores, 2 to 4 GB RAM, and 20 to 50 GB of disk space. The data transfer rate between any two VMs on the same VDC could vary between 80 and 100 Mbps within the data center.

Data center topologies: Three-level fat-tree topology [24] was used in our experiment. The physical machines were connected through the TOR switches located in the same rack. These rack switches were in turn connected through Layer 2 switches. Aggregation switches were used to connect the Layer 2 switches within the physical machine premises. These aggregation switches were connected to core switches through fiber optic cables for faster transmission and to eliminate copper wires within the data center.

Arrival rate of VDC requests: The arrival rate of VDC requests was random. Most of the time, as we evaluated, they do not follow any order or rate. The main pattern we noticed while testing our algorithm is that during the daytime there are more real-time and network-intensive job requests such as web services submitted for VDCs. During the night, CPU-intensive jobs such as batch processing are submitted for VDCs.

The average lifetime of a VDC requests is 5 h, which is a combination of short burst jobs and long batch-processing jobs. The maximum allocation wait time of VDC requests is 1 h for previous algorithms such as VDC Planner [3]. However, this wait time is too long for time-sensitive jobs such as web services and real-time jobs. If the request cannot be allocated, it needs to be rejected as soon as possible. This enables it to be allocated to another InP without any job processing delay.

In our simulations, we used a Python 2.7.13 interpreter and the Python schedule 0.5.0 library to implement the proposed algorithms. We ran the proposed algorithms on a PC with an Intel Core i7 CPU, 16 GB memory, and a Windows 64-bit professional operating system. Postplacement algorithms were run on an 8-core processor 2.1 GHz Intel Xeon Server with 16 GB RAM. We conducted an extensive survey of VDC performance parameters to determine VDC productivity enhancement [22, 23]. The following metrics were measured mainly to determine the reduction of operational cost by improving the number of requests serviced:

1. Resource utilization rate of each VDC
2. Acceptance rate
3. Power use before and after applying proposed algorithm
4. Average wait time of each VDC request

The values of these metrics for the proposed algorithms were compared with those of existing algorithms: VDC Planner [3], Venice [4], Greenhead [8], and the VDC algorithm for federated data centers [23]. Our algorithms eliminate traditional overprovisioning practices and enable rapid reallocation based on actual use patterns of existing VDC allocations.

4.2. Simulation results and analysis

Table 1 shows the various simulation setups and the corresponding performance metrics measured for the batched greedy and row-epitaxial algorithm.

Table 1. Performance comparison of the batched greedy and row-epitaxial algorithms.

Physical server capacity	Number of VMs created (standard)	Maximum incoming VDC requests	Random allocation (utilization rate %)	Batched greedy algorithm				Row epitaxial algorithm			
				Acceptance rate	Utilization rate %	PUE reduction %	Avg. waiting time (min)	Acceptance ratio	Utilization rate %	PUE reduction %	Avg. waiting time (min)
8	16	4	56.6	100	76.53	13.8	2	100	66.16	15.94	2
16	32	10	55.96	90	74.23	15.4	5	90	64.58	16.19	5
32	64	21	55.2	85.7	74.01	16.05	6	80.9	63.92	16.72	6
64	128	35	54.8	80	70.67	16.53	6	71.4	63.45	16.69	8
128	256	59	50.36	74.6	65.45	18.2	10	77.9	61.78	17.94	9

Acceptance rate: The rate of accepted requests is calculated as

$$Acceptance\ rate = \frac{Accepted\ VDC\ requests}{Total\ VDC\ requests\ arrived\ in\ the\ DC} \tag{4}$$

It is assumed that the accepted VDC requests are successfully embedded into the VDC and completed. As per our simulation experiments, 80% of the incoming requests can be mapped to the VDC. Our algorithms perform appropriately even with a high incoming request rate and maintain a similar rate throughout the testing process.

PUE: The PUE of the data centers is calculated as

$$\mathrm{P}UE = \frac{Total\ facility\ power}{IT\ equipment\ power} \tag{5}$$

Lower PUE values indicate good efficiency in terms of data center power usage. Our rescheduling algorithms reduce IT equipment power consumption and in turn reduce total power consumption of the data center. Our algorithms guarantee efficient power usage in the data center computing resources. Data center infrastructure efficiency (DCiE) can be calculated as

$$\mathrm{D}CiE = \frac{IT\ equipment\ power}{Total\ facility\ power} \tag{6}$$

DCiE is the reciprocal of PUE. Our algorithms increase the DCiE significantly by employing concurrent server consolidation at particular time periods, which improves total data center efficiency in terms of energy consumption. As the number of VDCs increases over time, our algorithms continue to perform well in terms of energy efficiency and resource utilization.

Average waiting time: Our rescheduling-based VDC algorithms show the traces of reduced wait time of incoming VDC requests. The average wait time W_avg of a VDC request is calculated as

$$W_{avg} = \frac{\sum_{i=1}^{n} W_{VDC}^{i}}{n} \tag{7}$$

There are a total of n accepted VDC requests and each VDC request waits in the allocation queue for W_{VDC}^{i} time. The wait time of the incoming requests varies from 2 to 10 min depending on the various data center metrics such as inbound and outbound traffic, customer subscription plans, SLA parameters, and resource availability. The average wait time calculated in our test bed environment is 45 min, which is 25% less than that of VDC Planner [3].

Table 1 compares the performance of the batched greedy and row-epitaxial algorithms. This comparison clearly shows that the two algorithms yield good results based on the simulation parameters. Before running the algorithms, requests were randomly allocated and the performance was studied. The allocation algorithms perform substantially better with respect to acceptance rate, utilization rate, energy consumption, and request wait time.

The major strength of both proposed algorithms is their ability to address problems systematically for a wide variety of data center settings. The results show that we should look beyond the traditional metrics of solution quality and total computational times. However, we show in Table 1 the results for the best embedding of the most challenging problem (maximum 59 VDC requests), which demonstrates a substantial improvement over previous algorithms.

4.3. Performance comparison of the batched greedy and row-epitaxial algorithms with baseline algorithms

Table 2 compares the results of our proposed algorithms with four previously proposed allocation algorithms: VDC Planner [3], Venice [4], Greenhead [8], and Sun et al.'s method [23]. The performance of the existing baseline algorithms is compared with the proposed batched greedy and row-epitaxial algorithms under identical simulation environments. Baseline algorithms show a significant improvement in acceptance rate and resource utilization rate of the data center in our data center simulation environment. At the same time, the reduction in PUE is negligible after the consolidation procedure for the baseline algorithms is run. For the proposed simulation environment, waiting time for the VDC requests remains the same for the baseline algorithms.

Table 2. Baseline algorithms compared with batched greedy and row-epitaxial algorithms in our simulation experiments.

	Parameters/ algorithms	VDC Planner [3]	Venice [4]	Greenhead [8]	Sun et al. [23]	Batched greedy algorithm	Row-epitaxial algorithm
Simulation setting	Topology used	Fat-tree	Fat-tree	NSFNet	Fat-tree	Fat-tree	Fat-tree
	Physical servers	128	128	14 nodes	128	128	128
	No. of switches	16	16	-	16	16	16
	Pattern of request arrival	10 requests/h	10 requests/h	8 requests/h	10 requests/h	10 requests/h	10 requests/h
Performance	Resource utilization rate	59%	62%	Network utilization reduced	44%	72%	64%
	Acceptance rate	33%	49%	40%	22%	86%	84%
	Average VDC lifetime	3 h (exponential distribution)	3 h (exponential distribution)	6 h	4 h	4 h	4 h
	PUE reduction before and after server consolidation	8%	11%	Not measured	11%	25%	28%
	Average request waiting time	18 min	46 min	Not measured	1 h	10 min (45% less than the VDC Planner)	9 min (50% less than the VDC Planner)
Revenue	Revenue gain (compared to SecondNet)	15%	23%	48%	14%	53%	58%

In the testbed environment, the batched greedy and row-epitaxial algorithms performed better than the baseline algorithms. Our algorithms show improvement in utilization rate, acceptance rate, PUE reduction, and waiting time of the VDC request. However, in order to utilize the proposed scheme effectively, the data center should be coupled with the consolidation algorithm to organize its resources. Our server consolidation algorithms run before and after the VDC allocation, but in the baseline algorithms, it runs on a particular time interval. This approach wastes the computing resources and affects the efficiency of the data center. The row-epitaxial algorithm attains a significant trade-off between the optimal solution and runtime (over 19% improvement with 8 times speed-up over the batched greedy algorithm with fat-tree topology). In comparison with the baseline algorithms, the revenue of the infrastructure provider is increased to 53% and 58% for the batched greedy and row-epitaxial algorithms, respectively. Furthermore we notice that, by reducing the number of active machines through the consolidation procedure, the revenue is improved. Combining consolidation with our embedding technique, we found that new algorithms can achieve up to 28% increase in the average net income compared to the baseline algorithms.

5. Conclusion and future work

In this paper, we studied the VDC request allocation problem in the physical data centers of cloud resources. The need for data center optimization and revenue generation in cloud environments led to a number of recent research proposals to address allocating resources to virtual data centers. A main challenge that needs to be

addressed is the dynamic VDC embedding problem, which aims at efficiently allocating computing and network resources to multiple VDCs. Moreover, at the same time it is important to maximize the total revenue and minimize the total energy consumption in the data center. However, existing solutions have not measured the problem in a dynamic and performance based environment. We formulated two VDC embedding algorithms based on the placement and reembedding requests. These algorithms are necessary to improve the performance of the data centers. This approach was motivated by the probe placement method of protein structures in bioinformatics. The proposed algorithms continuously consolidate physical server resources to improve acceptance rate, data center resource utilization, and revenue for the InP. This paper focused on allocation strategies that emphasize the efficient utilization of the resources to obtain a good ROI. The batched greedy algorithm and row-epitaxial algorithm improve the requests' acceptance ratio and InP revenue and reduce the average waiting time of the VDC requests. The proposed two algorithms demonstrated convincing results on simulation data compared with previous VDC allocation methods. These algorithms are also simulated in our testbed environment to study the performance. This approach can be further extended to run using real-time data. Moreover, providing security to VDCs is a future research direction. VDC security is essential because VDC resources are shared among multiple tenants. The logical isolation of storage and network elements is required to protect the environment of the stored information. It will be necessary to identify the possible security threats and their remedial measures to provide reliable VDC services to end users.

References

[1] Yang Y, Chang X, Liu J, Li L. Towards robust green virtual cloud data center provisioning. IEEE T Cloud Comput 2017; 5: 168-181.

[2] Wen X, Han Y, Yuan H, Zhou X, Xu Z. An efficient resource embedding algorithm in software defined virtualized data center. In: IEEE Global Communications Conference (GLOBECOM); 6–10 December 2015; San Diego, CA, USA.

[3] Zhani MF, Zhang Q, Simona G, Boutaba R. VDC Planner: dynamic migration-aware Virtual Data Center embedding for clouds. In: IFIP/IEEE International Symposium on Integrated Network Management (IM 2013); 27–31 May 2013; Ghent, Belgium. pp. 18-25.

[4] Zhang Q, Zhani MF, Jabri M, Boutaba R. Venice: Reliable virtual data center embedding in clouds. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications; 27 April–2 May 2014; Toronto, Canada. pp.289-297.

[5] Zuo C, Yu H, Anand V. Reliability-aware virtual data center embedding. In: 6th International Workshop on Reliable Networks Design and Modeling (RNDM); 17–19 November 2014; Barcelona, Spain. pp. 151-157.

[6] Yang Y, Chang X, Liu J, Li L. Towards robust green virtual cloud data center provisioning. IEEE T Cloud Comput 2017; 5: 168-181.

[7] Shi L, Katramatos D, Yu D. Virtual data center allocation with dynamic clustering in clouds. In: IEEE 33rd International Performance Computing and Communications Conference (IPCCC); 5–7 December 2014; Austin, TX, USA. pp. 1-10.

[8] Amokrane A, Zhani MF, Langar R, Boutaba R, Pujolle G. Greenhead: virtual data center embedding across distributed infrastructures. IEEE T Cloud Comput 2013; 1: 36-49.

[9] Dayarathna M, Wen Y, Fan R. Data center energy consumption modeling: a survey. IEEE Commun Surv Tutorials 2016; 18: 732-794.

[10] Kahng AB, Măndoiu II, Pevzner PA, Reda S, Zelikovsky AZ. Scalable heuristics for design of DNA probe arrays. J Comput Biol 2004; 11: 429-447.

[11] Kahng AB, Mandoiu II, Zelikovsky AZ. Highly scalable algorithms for rectilinear and octilinear Steiner trees. In: Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference; 21–24 January 2003; Kitakyushu, Japan. pp. 827-833

[12] Sun X, Su S, Xu P, Jiang L. Optimizing multi-dimensional resource utilization in virtual data center. In: 4th IEEE International Conference on Broadband Network and Multimedia Technology; 28–30 October 2011; Shenzhen, China. pp. 395-400.

[13] Li M, Bi J, Li Z. Virtual-switching-aware VM consolidation in virtualized data centers. In: IEEE 6th International Conference on Cloud Computing Technology and Science; 15–18 December 2014; Singapore. pp. 817-822.

[14] Fuerst C, Schmid S, Feldmann A. Virtual network embedding with collocation: benefits and limitations of pre-clustering. In: IEEE 2nd International Conference on Cloud Networking (CloudNet); 11–13 November 2013; San Francisco, CA, USA. pp. 91-98.

[15] Guo C, Lu G, Wang H, Yang S, Kong C, Sun P, Wu W, Zhang Y. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In: Proceedings of the 6th International Conference ACM CoNEXT; 30 November–3 December 2010; Philadelphia, PA, USA.

[16] Roohitavaf M, Entezari-Maleki R, Movaghar A. Availability modeling and evaluation of cloud virtual data centers. In: International Conference on Parallel and Distributed Systems; 15–18 December 2013; Seoul, South Korea. pp. 675-680.

[17] Xu L, Li C, Li L, Liu Y, Yang Z, Liu Y. A virtual data center deployment model based on the green cloud computing. In: IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS); 4–6 June 2014; Taiyuan, China. pp. 235-240.

[18] Kliazovich D, Bouvry P, Khan SU. GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. J Supercomput 2012; 62: 1263-1283.

[19] Zou J, Yan F, Lee TT, Hu W. An add/drop algorithm for virtual data center embedding. In: Asia Communications and Photonics Conference; 11–14 November 2014; Shanghai, China.

[20] Kansal NJ, Chana I. Artificial bee colony based energy-aware resource utilization technique for cloud computing. Concurr Comput Pract Exper 2015; 27: 1207-1225.

[21] Correa ES, Fletscher LA, Botero JF. Virtual data center embedding: a survey. IEEE Lat Am Trans 2015; 13: 1661-1670.

[22] Sivaranjani B, Vijayakumar P. A technical survey on various VDC request embedding techniques in virtual data center. In: National Conference on Parallel Computing Technologies (PARCOMPTECH); 19–20 February 2015; Bangalore, India.

[23] Sun G, Liao D, Bu S, Yu H, Sun Z, Chang V. The efficient framework and algorithm for provisioning evolving VDC in federated data centers. Future Gener Comp Sy 2017; 73: 79-89.

[24] Leiserson CE. Fat-trees: Universal networks for hardware-efficient supercomputing. IEEE T Comput 1985; C34: 892-901.