An Adaptive Allocation Scheme for Load Balancing and SLA Maintenance in Multi-Location Data Center Networks

Mirza Mohd Shahriar Maswood*+, Deep Medhi+
University of Missouri–Kansas City, USA
Email: mmnt7@mail.umkc.edu, dmedhi@umkc.edu

***Abstract*—Allocation of resources in data centers (DCs) needs to be done in a dynamic fashion for cloud enterprise customers who require virtualized reservation-oriented services on demand. Due to the spatial diversity of data centers, the cost of using different DCs also varies. In this paper, we propose an allocation scheme to balance the load among these DCs with different cost to minimize the total provisioning cost in a dynamic environment while ensuring that the service level agreements (SLAs) are met. Compared to a benchmark scheme (where all requests are first sent to the cheapest data center), our scheme can decrease the proportional utilization from 24% (for heavy load) to 30% (for normal load) and achieve a significant balance in the cost incurred by individual DCs. Our scheme can also achieve 7.5% reduction in total provisioning cost under certain service level agreement (SLA) in exchange of low increment in blocking. Finally, we tested our heuristic on 5 DCs to show that our allocation schemes follows the weighted cost proportionally.**

***Index Terms*—Data Center Networks, Resource Optimization and allocation on-demand, Denial of Service, Load Balancing, Virtual Network**

I. Introduction

Enterprise customers use cloud data centers (DC) for a variety of applications. Often, they require virtual network (VN) services from the DC providers in a reservation-oriented mode for both network and compute resources. From the perspective of a DC provider, it wishes to accommodate as many enterprise customers while meeting the service goals. Furthermore, the provider with multiple DC locations must balance between resource utilization and cost, especially given that the geographically distributed DCs have widely varying costs, for example, due to electricity pricing. Such price variations seem to suggest that it is cheaper to allocate customers to the cheaper DC; on the other hand, this could lead to violation of service level agreements (SLAs).

Most work related to allocation of requests in intra-DC networks consider east-west traffic, i.e., the intra-DC traffic between servers. In our work, we focus instead on enterprise customers' requests that result in north-south traffic in DCs requiring both network bandwidth and server resources, with multiple DC locations. Even though, due to the advancement of server virtualization technologies, these days the percentage of east-west traffic is increasing in intra-DC network compared to north-south traffic, still now 20% of the intra-DC traffic is north-south. Furthermore, this percentage is very high for university and private enterprise DCs where 40-90% of the intra-DC traffic is north-south [1]. A recent study by Cisco showed that more than 90% of traffic in campus network is north-south traffic [2]. This motivates us to work with north-south traffic. In particular, we address serving different enterprise customer groups using VNs in the DC networks through dynamic traffic engineering by allocating both network bandwidth and processing resources. That is, we consider the north-south traffic environment where each request consists of a two-tuple demand: one for DC network bandwidth and the other for the processing demand at the end hosts. In particular, we propose an adaptive allocation scheme to balance the load among multi-location DCs with different costs in a dynamic demand environment from enterprise customers in a reservation-oriented mode and measure the performance of this scheme in terms of overall cost, the cost and utilization incurred by individual DCs and blocking. A salient feature of our approach is that we consider the demand request from enterprise customers to consist of both bandwidth and compute resources. Furthermore, we also aim to reduce power consumptions. We factor in the cost variation for DCs, assuming that this cost can vary from one place to another as the cost of energy and bandwidth has a spatial diversity. Our proposed scheme strives balance the load among the available DCs while minimizing the incurred cost due to provisioning the requests. We model in the potential SLA violation through proportional DC utilization. That is, if the proportional utilization of a DC goes beyond a threshold, a penalty cost is incurred. We compare our scheme to a benchmark scheme that always prefers the least cost DC.

The rest of the paper is organized as follows. The related work is discussed in Section II. In Section III, we propose our adaptive allocation scheme (LBSel). In Section IV, we summarize the simulation setup and parameter details. Results are discussed in Section V. Finally, in Section VI, we summarize our concluding remarks and discuss potential future work.

II. Related Work

A number of research works has addressed balancing the load among geographically distributed servers. In these works, researchers proposed various policies to distribute the workload among geographically distributed DCs to achieve

Acknowledgement: *Partially supported by a School of Graduate Studies Research Grant at the University of Missouri–Kansas City, +Partially supported by National Science Foundation Grant # 1526299.

different objectives. [3], [4], [5], [6], [7], [8] focused on minimizing electricity cost.

In [9], [10], the prime objective was to improve energy efficiency. [11], [12], [13] considered load balancing to maximize the usage of renewable energy. Some other research works aimed to achieve different goals such as minimizing bandwidth cost [14], reduction of carbon footprint [15], [16], and achieving cooling efficiency [17]. A scheme is proposed in [3] to reduce the electricity cost but not consumption by managing the majority of the requests to be served by the DC with low electricity price. In [4], a framework is proposed to minimize the total electricity cost considering price variation in several electricity markets while ensuring the quality of service (QoS). In [5], authors proposed a scheme to reduce the power cost in geo-distributed DCs that is especially effective for handling delay tolerant workloads. In [6], they proposed a centralized algorithm to reduce the electricity price by using energy storage through backup batteries. They utilize the energy storage idea in a way that charging the batteries at a low price time and then using the batteries to support electricity need at the time when the price is high. In [7], authors proposed an algorithm to minimize the electricity cost by ensuring QoS for premium customers only and reducing throughput of ordinary customers in the situation when the electricity cost exceeds a desired monthly budget.

In [9], authors proposed a solution to reduce the electricity consumption of operating DCs by utilizing a diversity of global electricity market and heterogeneity of geo-distributed DCs. In [12], they proposed a strategy to maximize the usage of renewable energy and minimize the consumption of cooling energy by optimally placing the requests among all available DCs. In [14], they proposed a model to minimize the bandwidth and energy cost by considering the majority of requests to be served by low price DCs. However, they did not simply imply the naive idea of reducing energy cost by transferring the requests towards low price DC, rather they considered minimizing bandwidth cost as well. A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers was proposed in [18]. A distributed framework for carbon and cost aware geographical job scheduling in a hybrid DC infrastructure is proposed in [19]. [20] proposed a load balancing scheme to distribute the workload in geo-distributed data centers of a cloud while considering the minimization of service delay.

There are a number of ways our work differs from the above works. First, we model the DC explicitly by using a DC topology, which allows us to also generate the actual VN allocation (with paths) for each customer, along with the hosts the VN customers are allocated to. Second, in our approach, the request is made up of two resource tuples, one for the bandwidth resources and the other for host compute resources. Third, Our approach reduces the energy cost by minimizing the energy consumption as well as taking advantage of spatial diversity of price. In an earlier work, a mixed integer linear programming formulation was developed [21]. Since this formulation was not scalable to solve large scale problems, we developed a heuristic to solve large scale problems in [22]. However, neither of these two works considered proportional distribution of load among geo-distributed DCs (with DC network awareness) or diversity of cost of using different DCs. This led us to develop our proposed LBSel to distribute the load among geo-distributed DCs proportionally when the cost of different DCs is different. In another concurrent work [23], we considered latency as a Quality of Service (QoS) requirement for VN customers, which is not considered here since our focus here is on load balancing with network awareness among available DCs.

III. Adaptive Allocation Scheme (LBSel)

In this section, we present our proposed adaptive allocation scheme, LBSel. In this scheme, a DC is selected for satisfying a request in a way that the scheme can balance the load as much as possible to reduce the penalty cost due to SLA violation. Our allocation approach considers new request arrivals at random from customers, for which the resource allocation (both network bandwidth and host resources) is done at each review point $t \in T$, where T is a discrete temporal window consisting of review points. The duration of a new VN request that uses the DC is assumed to be random. Note that since the DC is set up to serve VN customers, at any time instant, there are existing VN tunnels and host resources allocated for prior requests. Thus, any (micro-)workload that needs immediate access to resources, that is, workload that cannot wait until the next review point, is assumed to be served by existing VN channels and host resources assigned to the customers that were set up at earlier review points. Since such immediate workloads are served through existing resources, they are not considered in our case. In other words, the scope of our work is to consider new requests at review points that are major requests requiring allocation of new bandwidths, VN tunnels and new resources.

To illustrate our approach, consider the DC topology shown in Fig. 1, which depicts just one site of the multi-location DCs. The entry point (EP) in a DC is then the north-end and the serving host is the south-end of the north-south traffic. Our approach assumes that there is a central controller that is responsible to operate the proposed heuristic to set up the allocations. For instance, this can be accomplished by using a software-defined network (SDN) based approach.

In our framework, each request consists of a 2-tuple $\langle h, r \rangle$ where h is the bandwidth demand and r is the processing resources required from a serving host. Thus, at a particular

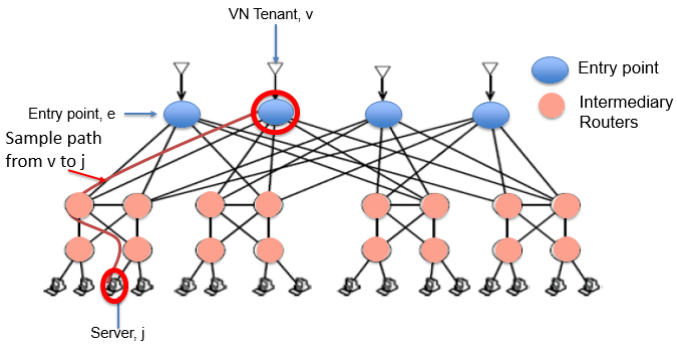


Fig. 1: Data Center Topology [24]

TABLE I: Notations Summary

Constants/Parameters:
D = Set of data centers, $N = \#(D)$
J_d = Set of servers in data center d, $H = \#(J_d)$
I_d = Set of entry points in data center d
V = Set of virtual networks
$K_v(t)$ = Set of requests from virtual network v at review point t
F = Set of frequencies in which a particular server can run
L_d = Set of links in data center d
$P_{ij}^{vkd}(t)$ = Set of paths from entry point i to server j in data center d for request k from VN v at review point t
b_{jf}^{d} = Power consumption in server j of data center d at frequency f
$h^{vk}(t)$ = Bandwidth demand for request k from VN v at t
$r^{vk}(t)$ = CPU processing capacity demand for request k from VN v at review point t
a_{jf}^{d} = Capacity of server j of data center d at frequency f
$c_l^d(t)$ = Available capacity on link l of DC d at review point t
$m_j^d(t)$ = Server in use indicator: 1 if server j of data center d is in use, 0 otherwise
wc_d = Weighted cost of using data center d
rs_d = Request served by data center d in one turn
$\beta^d(t)$= Normalized cost of data center d at review point t
g_d = Proportional utilization of data center d
δ = Threshold of proportional utilization of data center d; a penalty cost q is be added if the proportional utilization of a data center goes beyond this point
q = Penalty cost due to SLA violation
α, μ, γ: Weight parameters related to 3 individual cost that comprises total cost
Variables:
$u^{vkd}(t)$ = Binary decision variable to choose data center d to satisfy request k from virtual network v at review point t
$z_l^{vkd}(t)$ = Bandwidth needed on link l of datacenter d for request k from VN v at review point t
$w_{jf}^{vkd}(t)$ = Binary decision variable to choose the optimum frequency f from the range of available frequencies of server j of data center d to meet the required demand of CPU processing capacity for request k from VN v at review point t

review point t, if a VN customer $k \in K_v(t)$ has a request, the request tuple is further represented by $\langle h^{vk}(t), r^{vk}(t)\rangle$, which is to be served by DC $d \in D$. While the bandwidth demand needs to be satisfied by the capacity of the links within the DC $l \in L_d$ from the entry point $i \in I_d$ to a server $j \in J_d$, the processing resources must be satisfied by the servers' available resources. We assume that there is a given set of paths $P_{ij}^{vkd}(t)$ from entry point i to server j, which could be potentially different at each review point t.

For energy consumption, we consider that every server can run at a given set of CPU frequencies $f \in F$. At each particular frequency, a server works at a particular processing capacity a_{jf}^{d}. A specific amount of power b_{jf}^{d} is required to run the server at that frequency. If we run the server at the highest frequency, it offers the highest processing capacity, but consumes the highest amount of power.

We solve the resource allocation problem at each review point t. For this, we use our heuristic which is presented in Algorithm 1, 2, and 3 in which we attempt to balance the load among geo-distributed DCs to reduce SLA violation and accommodate as many requests as possible while minimizing the resources requirement towards satisfying those requests in order to reduce the overall cost. Note that Algorithm 1 calls the procedures described in Algorithm 2 and 3. All notations used in our heuristic are summarized in Table I. The input and output of this heuristic are given below:

DC related Input: Number of DCs (N), all paths available $p \in P_{ij}^{vkd}$ $i \rightarrow j$, capacity of each link (c_l^d), capacity of each server at different frequencies (a_{jf}^{d}), wc_d, rs_d.

VN related Input: Resource requirement (r^{vk}) and bandwidth

Algorithm 1 LBSel Heuristic

update $a_{jf}^{d}, f \in F, j \in J_d, d \in D$
update $c_l^d, l \in L_d, d \in D$
update $wc_d, rs_d, d \in D$
1. $d = 1$
2. $\hat{K} = \sum_{v \in V} \#(K_v)$
3. **while** $\hat{K} \neq 0$ && $D == \emptyset$ **do**
4. *count_v* $= 0$
5. Change DC: **for all** $i \in I_d$ **do**
6. **for all** $j \in NS(i)$ **do**
7. **if** $m_j^d == 0$ **then**
8. $s_j^d = max\{a_{jf}^{d}\}$
9. **for all** $v \in V$ **do**
10. **for all** $k \in K_v$ **do**
11. **if** $\beta^d == wc_d$ && *count_v* $\leq rs_d$ **then**
// CPU Resource Allocation (Algorithm 2)
12. $s_j^d, w_{jf}^{vkd}, r^{vk} =$ CRA($v, k, d, r^{vk}, s_j^d, F, a_{jf}^{d}, w_{jf}^{vkd}, j$)
13. *use shortest path* $p \in P_{ij}^{vkd}, i \rightarrow j$
// Bandwidth Allocation (Algorithm 3)
14. $z_l^{vkd}, h^{vk}, count\_v, u^{vkd}, \hat{K}, m_j^d =$ BA($p, l, c, h^{vk}, k, v, d, z_l^{vkd}, u^{vkd}, m_j^d, count\_v$)
15. **else**
16. d ++
17. **if** $d = N + 1$ **then**
18. $d = 1$
19. **end if**
20. *go to Change DC*
21. **end if**
22. **end for**
23. **end for**
24. **end if**
25. **end for**
26. **end for**
27. **if** $\sum_{j \in J_d} m_j^d == H$ **then**
28. $D = D \setminus d$
29. **end if**
30. **end while**

count_blocking = 0
for all $v \in V$ **do**
for all $k \in K_v$ **do**
if $r^{vk} == 0$ && $h^{vk} == 0$ **then**
return $u^{vkd}, w_{jf}^{vkd}, z_l^{vkd}$
else
count_blocking ++
end if
end for
end for

Algorithm 2 LBSel CPU Resource Allocation (CRA)

CRA($v, k, d, r, s, F, a, w, \hat{j}$)
/ This procedure allocates CPU resources to satisfy requets arrived at review point t */*
1. **if** $s_j^d \geq r^{vk}$ **then**
2. $a_{j\hat{f}}^{d} = min_{f \in F, a_{jf}^{d} \geq r^{vk}}\{a_{jf}^{d}\}$
3. **if** $s_j^d \geq a_{j\hat{f}}^{d}$ **then**
4. $s_j^d = s_j^d - a_{j\hat{f}}^{d}$
5. $w_{j\hat{f}}^{vkd} = 1$
6. $r^{vk} = 0$
7. **end if**
8. **end if**
9. **return**(s, w, r)

Algorithm 3 LBSel Bandwidth Allocation (BA)

BA($p, l, c, h, k, v, d, z, u, m, count\_v$)
/ This procedure allocates link bandwidth to satisfy requets arrived at review point t */*
1. $\hat{c} = min_{l \in P}\{c_l^d\}$
2. **if** $h^{vk} \leq \hat{c}$ **then**
3. **for all** l *used in* p **do**
4. $z_l^{vkd} = c_l^d - h^{vk}$
5. **end for**
6. $h^{vk} = 0$
7. $count\_v++$
8. $u^{vkd} = 1$
9. $\hat{K} = \hat{K} - 1$
10. $m_j^d = 1$
11. **end if**
12. **return**$(z, h, count\_v, u, \hat{K}, m)$

requirement (h^{vk}) to satisfy the requests at review point t.
Output: Near optimal solution to satisfy a request or report that request as blocked.

Algorithm 1: At first, the heuristic updates the existing capacity of resources based on the given input discussed above. To find the best way of allocating resources, the heuristic picks one DC among all available DCs and continues to use it until it reaches the max. number of requests it can serve in one turn based on the weighted cost. Then, the turn to serve the requests is given to another available DC and the number of requests served by that DC depends on its weighted cost as well. In this way, all available DCs get their turn to serve requests as round robin fashion. Therefore, when all the DCs complete their turn, the DC used at the first attempt, gets the turn to serve VN customers back again. This is how the process continues until all the incoming requests at a particular review point are served or the resources (servers and link capacity) of all available DCs are exhausted. If all the requests at a particular review point cannot be served with existing resources, then the requests which are not satisfied are reported as blocked requests. Thus, our heuristic also enables us to compute the blocking rate under a particular load condition.

When the turn of one of the available DC comes, the heuristic starts with one entry point (EP) of that DC and continues to allocate requests through this until either all neighbor servers (NS) or all required links to establish a path from that entry point to an NS are occupied. By neighbors, we mean that two edge switches are considered as the neighbor edge for each entry point; then, for a particular entry point, the servers which are connected to this neighbor edge of the EP are considered as a NS for this entry point. From all the available neighbor servers, the heuristic picks a server from a neighbor server rack and continues to use the servers from that rack until all servers are occupied. When all the servers from that rack are occupied or do not find enough capacity for any of the required links to establish a path, the heuristic starts with another neighbor server rack. This way the heuristic continues to allocate the available resources to satisfy all the requests arriving at a review point.

Algorithm 2: In our approach, a server's goal is to fit as many requests as it can. To do so, at first, this server starts with the maximum available capacity and continues to fit requests until it reaches the limit of its capacity or all the considered requests are allocated with required compute resources. While doing so, from all the available capacity of that server, the heuristic tries to find the minimum capacity using which resource requirement from one request can be satisfied. After finding the minimum resource requirement, this quantity is reduced from the maximum available capacity. Through this, the heuristic is able to determine the best capacity in which a server should run. Furthermore, the heuristic gives us the information that by running the server at this frequency, how processing capacity that is generated is fractionally allocated among different requests.

Algorithm 3: After being ensured about the resource fulfillment from a server, the heuristic uses the shortest path to route all the requests that can be satisfied by that server from the entry point to the targeted server. Now, for all the requests served by this server, once the shortest path is established, link capacity is modified by reducing the required link capacity from the currently available link capacity (from the given input in the review point) for each link.

We use the following formula to compute the total provisioning cost:

$$cost = \alpha \sum_{d \in D} \sum_{v \in V} \sum_{l \in L_d} z_l^{vd}(t)$$
$$+\mu \sum_{d \in D} \sum_{j \in J} \sum_{v \in V} \sum_{f \in F} b_{jf}^d w_{jf}^{vd}(t) + \gamma \sum_{d \in D} \sum_{v \in V} wc_d u^{vd}(t). \quad (1)$$

From (1), we can see that the total cost consists of three sub cost where the first cost is for using the amount of bandwidth, the second cost is due to the energy usage and the third cost incurs from using that data center which is dependent upon the cost of bandwidth and energy in the location where the DC is situated. wc_d varies based on the spatial diversity of price of bandwidth and energy. For simplicity, we consider that wc_d does not vary based on the total amount of bandwidth and energy usage. It is fixed for a DC in a particular location. Now, considering SLA violation as explained before, (1) is extended as follows:

$$Total\_cost = cost + q \sum_{d \in D} (g_d - \delta)_+, \quad (2)$$

where $z_+ = \max\{z, 0\}$.

IV. Simulation Study Setup and Parameter Values

To conduct our study, we compare our scheme LBSel with a benchmark scheme LCSel where all the requests are directed to the cheapest DC if resources are available without considering load balancing. LCSel also uses a heuristic similar to LBSel except the concept of weighted distribution. We used the DC topology shown in Fig. 1. In the first set of extensive study, we used two DCs ($N = 2$) to select from; later, we also used 5 DCs to test whether our allocation scheme is

TABLE II: DC related parameters

Number of links in each DC	56
Capacity of each link	600
Number of nodes in each DC	820
Number of Entry points	4
Number of Servers	800

TABLE III: CPU Frequencies, Capacities and Power Consumption (watts)

Frequency Option	1	2	3	4	5	6	7	8	9	10
Normalized Capacity	.1	.2	.3	.4	.5	.6	.7	.8	.9	10
Power Consumption	10	20	30	40	50	60	70	80	90	100

TABLE IV: Number of Requests Served by a Data Center in One Turn Based on the Weighted Cost

Weighted Cost(wc_d)	1	1.2	1.4	1.6	1.8	2
Number of Request Served(rs_d)	20	18	16	14	12	10

proportional to the weighted costs of each DC. Here, two DCs are heterogeneous in a sense that the usage cost of them are different means one is cheaper than another. However, the resources available in each DC are considered to be identical in this study; each consisted of $I_d = 4$ entry points and $J_d = 800$ servers and all links inside the DC are set with the same capacity. We set $P_{ij}^{vd}(t) = 4$ paths from an entry point to a server among which only one path will be used for a specific request for the duration of this request. Parameter values used for the DCs are summarized in Table II.

Here, we vary the wc_d for one DC from 1 to 2 in increments of 0.2 for that DC, where we keep the wc_d for other DC as fixed at 1. wc_d is the total weighted cost that is incurred by using DC d. Our assumption of the total cost for getting service from a DC consists of bandwidth cost and the energy cost of the server. By using a higher wc_d for a DC, we simply assume that the combination of bandwidth and energy cost is higher for that DC due to its geo-location as we know that the bandwidth cost and energy cost have spatial diversity. The number of requests to be served in one turn by a DC is determined based on the value of wc_d of that DC (see Section III). However, we must choose this number carefully. After doing some preliminary simulation, we determined a suitable value for this number to maximize the utilization of each DC (load balancing) while keeping the provisioning cost and blocking within an acceptable limit as well. The number of requests served in one turn by a DC (based on wc_d) used for this study is shown in Table IV.

We consider $V = 3$ VN classes that generate requests. A request is represented by the tuple $\langle h, r \rangle$. We vary $\langle h, r \rangle$ to create different VN classes. The demand type we use for these VNs is generated by considering the variation between different VNs while keeping the request same within each VN, i.e., $\langle h^1, r^1 \rangle = \langle 3, 0.3 \rangle$, $\langle h^2, r^2 \rangle = \langle 6, 0.6 \rangle$, $\langle h^3, r^3 \rangle = \langle 9, 0.9 \rangle$ as shown in Table V. We assume that the request arrivals follow a Poisson process since an earlier study found that the batch arrivals to DCs follow Poisson process [25]. and the service duration for the request arrivals is assumed to follow the negative exponential distribution with an average value of 5 time units measured in terms of the number of discrete review points.

Note that with an increase in the arrival load, the system may not have sufficient capacity to accommodate all requests. Thus, our simulation environment also recorded any requests that were not satisfied by the system by tracking the blocked requests to determine the blocking rate. Through our initial experimentation, we attempted to find the arrival rate for which the blocking was approximately 1%. We refer to that arrival rate as a normal loaded network condition, and assigned the normalized load of 1.0. We then continued to increase the arrival rate until we found the arrival rate for which the average blocking was approximately 7% to indicate highly overloaded condition. Also, through our initial experimentation, we chose the weight factors for each term in the cost of eqn.(1) and set them as $\alpha = 0.3, \mu = 0.05, \gamma = 8.1$ since we found these values to provide a proper balance among the three cost components, without any one term being more dominant than the other two terms.

For simulation, we first determined the warm-up time and then collected the data for a steady-state region after the warm-up time. For each arrival rate, we used 10 independent seeds and reported the results on the average value. We also computed the confidence interval and found the 90% confidence interval to be approximately 5% in cost variation for low arrival rates to 2.5% for high arrival rates. To compute the power consumption cost which is a part of the total cost, we use the power consumption and processing capacity of a particular server that runs at a specific frequency, as shown in Table III. We consider fractional power consumption for using a fraction of capacity to satisfy a request from the capacity in which the server was originally running, since at some instances, one server can satisfy multiple requests.

V. Results

We divided our analysis into two parts. From subsection V-A to V-D, we used 2 DCs to show a comparative analysis of our scheme (LBSel) and the benchmark scheme (LCSel) based on how the individual DC cost, individual DC utilization, total provisioning cost and blocking rate vary with the different weighted cost under certain network load. Then in subsection V-E, we used 5 DCs to show that our allocation scheme follows the weighted cost proportionally.

A. Individual DC Cost

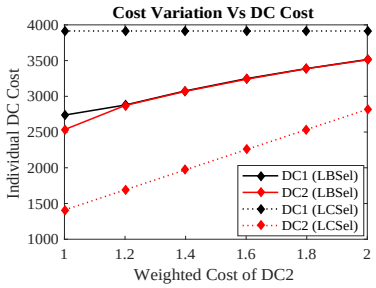
The DC cost is calculated by considering the third part of the cost (1). Under a normal loaded condition, using our scheme we can keep the cost of the two DCs almost similar regardless of increasing the weighted cost. However, as we increase the load of the system, the cost of the expensive DC starts to increase with the increment of the weighted cost of that DC. The reason behind this is that as we increase the load even though we try to fit more requests to the cheaper DC, the resources of it become exhausted and the remaining requests need to use the expensive DC. Therefore, from the point of view of balancing cost between two DCs, our scheme can work at its best under normal load condition because of less requests being directed towards the expensive DC by force. However, as we can see from Fig. 2a to 2d, our scheme can achieve a prominent success in keeping a balance between the cost of two DCs compared to the benchmark scheme.

B. Individual DC Utilization

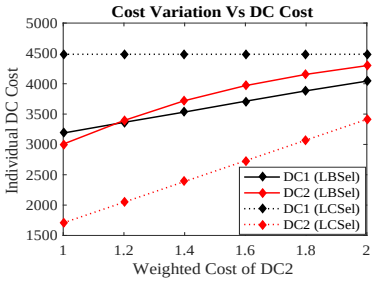
Recall that, the utilization of a DC is computed based on the percentage of requests served by that DC. From Fig.

TABLE V: Values associated with the demand type used for this research required by customers from different VN classes.

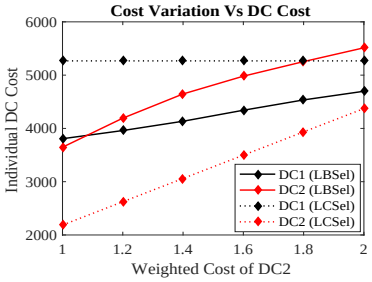
Demand type	Parameters	Values
Different Bandwidth and CPU Processing Capacity demand for different VNs while the demand is fixed within each VN	Bandwidth Demand-VN-1	3
	Bandwidth Demand-VN-2	6
	Bandwidth Demand-VN-3	9
	CPU Processing Capacity Demand-VN-1	0.3
	CPU Processing Capacity Demand-VN-2	0.6
	CPU Processing Capacity Demand-VN-3	0.9



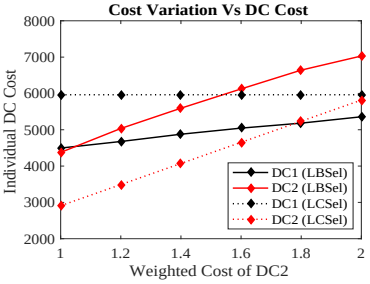
(a) normal load



(b) 1% overload than normal load

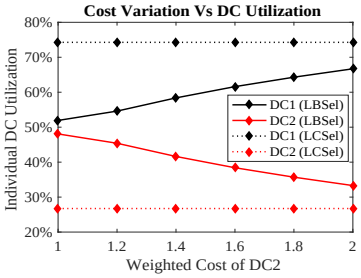


(c) 2% overload than normal load

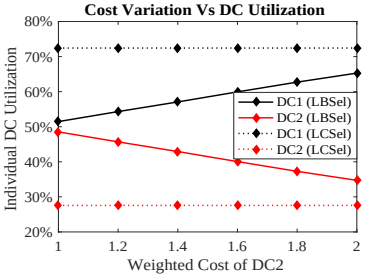


(d) 3% overload than normal load

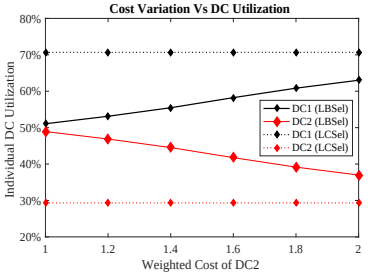
Fig. 2: DC Cost: Scheme-1 vs. Scheme-2 under different load condition



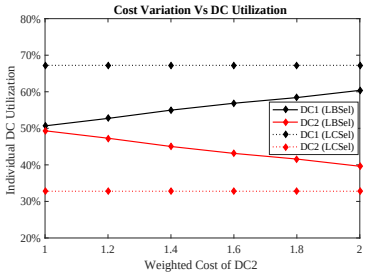
(a) normal load



(b) 1% overload than normal load



(c) 2% overload than normal load



(d) 3% overload than normal load

Fig. 3: DC Utilization: Scheme-1 vs. Scheme-2 under different load condition

3a to 3d depicts the comparison between our scheme and benchmark scheme for balancing DC utilization. From these figures, it is obvious that our scheme can obtain a better load balancing from normal loaded condition to very heavily loaded condition. The gap between the two schemes decreases as the network load increases which is also due to the forced entry into the expensive DC as the resources of the cheaper DC gets occupied.

C. Total Provisioning Cost

The cost is calculated using (2) of Section III. We consider an SLA which requires that the proportional utilization of a DC to be within a threshold(δ) value. If the proportional utilization of a DC goes beyond this threshold, an additional penalty cost is incurred. This penalty cost is incurred as shown in (2). From Fig. 4a to 4d depicts the variation in total cost under different load condition. In each sub-figure, we presented the comparison between our scheme LBSel and the benchmark scheme LCSel. We varied the value of penalty weight, q. We tested for three values of δ: 0.55, 0.575, and 0.60. For $\delta = 0.55$, the representative values of q_1 (normalized to per arrival) that we tested were 0.76, 1.16, 1.52, 1.78 and for q_2, we used 0.32, 0.5, 0.74, 1.16 from normal to heavy loads. For $\delta = 0.575$, q_1 used were 0.76, 1.16, 1.52, 2.4 and for q_2, we used 0.35, 0.59, 0.93, 1.76 from normal to heavy loads. Finally, for $\delta = 0.6$, we used 0.76, 1.16, 1.57, 4.98 for q_1 and 0.42, 0.73, 1.25, 3.64 for q_2. Due to the space limitation, we showed the result for $\delta = 0.575$. What this means is that for normal loads, it is better to have the penalty incurred to be a low factor for the system to benefit from load balancing. On the other hand, in overloaded situations, much more traffic is denied acceptance and the data center loads are also highly utilized in both data centers so that load balancing does not result in much improvement in avoiding SLA violations.

D. Blocking

From Fig. 5a to 5d, we can see that the blocking is always higher in our scheme LBSel than benchmark LCSel scheme. However, there are two interesting factors to observe: first, the blocking of our proposed scheme continues to increase as the weighted cost of expensive DC increases. This is due to the fact that with the increment of the weighted cost, the number of requests served by expensive DC in one turn decreases and more swapping is done between DCs. As a result, such a situation may arise when the last server used in a turn may serve fewer requests than its capacity as the turn of that DC ends. To reduce this, we can increase the number of requests to be served in one turn, but then we will achieve less performance in balancing the load between two DCs. Second, the variation in blocking continues to reduce as the load of the system increases. This is due to the fact that under heavy load, blocking due to load is the prominent factor behind the total system blocking than the blocking incurred due to swapping between DCs (used in LBSel).

It may be noted that blocking sharply increases at a much smaller overload for large-scale systems. This behavior is consistent with a single link loss system model (without routing and server selection) that can be computed with the Erlang-B blocking formula for large offered loads. The nonlinear concave behavior of Erlang-B blocking is well-known as the load and capacity increase impacting blocking, especially when the services have heterogeneous bandwidth requirements; see [26, Chapter 11] for a discussion.

TABLE VI: Distribution of Load among 5 Geo-distributed DCs based on their Weighted Cost

DC#	1	2	3	4	5
wc_d	1	1	1.2	1.2	1.2
Percentage of Request Served	24.18%	22.9%	18.7%	17.61%	16.61%
wc_d	1	1	1.6	1.6	1.6
Percentage of Request Served	26.5%	25.4%	17.25%	16.21%	14.63%
wc_d	1	1	2	2	2
Percentage of Request Served	30.94%	28.97%	13.87%	13.27%	12.95%

E. Proportionality of Allocation

Next, we tested on 5 geo-distributed DCs using our heuristic to see whether our allocation scheme follows the weighted cost proportionally. We found that our developed heuristic could balance the load for 5 geo-distributed DCs based on weighted costs; see Table VI. We understand that a DC provider with multiple DCs having nearly same amount of weighted cost should balance the load among available DCs almost equally. However, if the weighted cost varies by a big margin then, the DC provider should choose the cheaper DC to serve more percentage of requests while ensuring that the cheaper DCs are not over-utilized and the expensive DCs are not underutilized; this is where our heuristic was able to follow the allocations to different DCs based on their weights costs.

F. Key observations

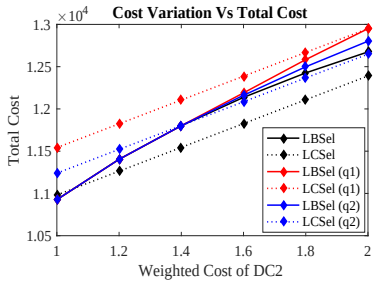
Compared to the benchmark scheme, we achieve the following with our proposed LBSel scheme:

- Total provisioning cost reduces up to 7.5%.
- The proportional utilization of the low cost DC can be reduced up to 30% for normal loads and 24% for heavy loads.
- The SLA violation and its impact depends on the threshold (δ) used and the penalty weighs, factoring in on normal and heavy loads.

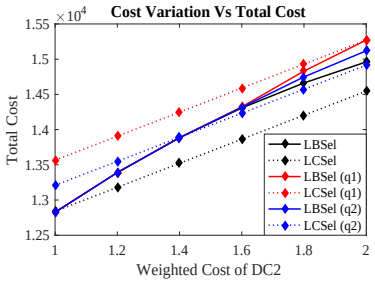
Hence, similar to almost all existing system there is a trade-off in our scheme too. However, since the blocking increment is not that significant even in worst case, hence, using our scheme can help the cloud service providers to achieve a better load distribution among DCs with different cost factor without violating SLA in most cases.

VI. Conclusion and Future Work

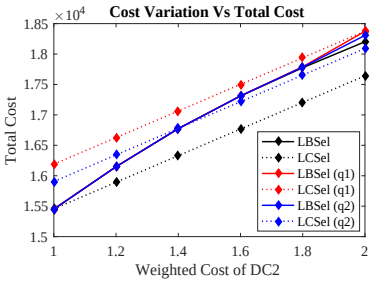
In this work, we proposed a novel adaptive allocation scheme (LBSel) that can be operated by an SDN controller to balance the load among different geo-distributed DCs with different cost due to spatial diversity. LBSel can achieve a significant improvement in load distribution to maintain SLA and keeping a balance between cost of cheaper and expensive DCs in the cost of a lower increment in blocking.



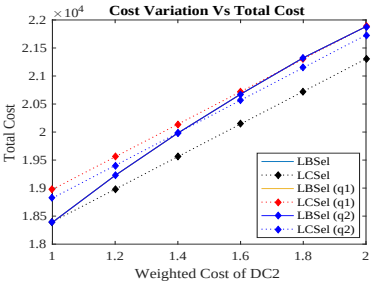
(a) normal load



(b) 1% overload than normal load

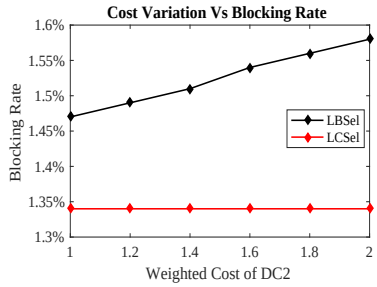


(c) 2% overload than normal load

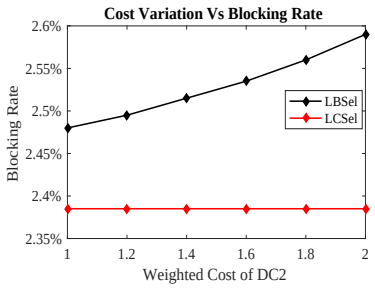


(d) 3% overload than normal load

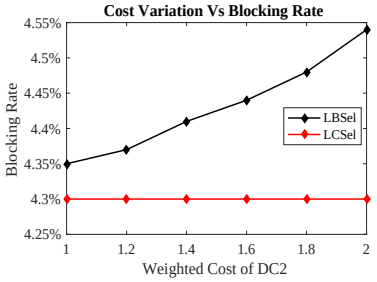
Fig. 4: Total Provisioning Cost: Scheme-1 vs. Scheme-2 under different load condition [$\delta = 0.575$]



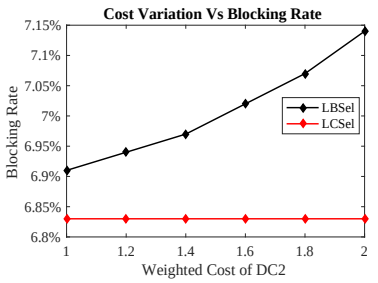
(a) normal load



(b) 1% overload than normal load



(c) 2% overload than normal load



(d) 3% overload than normal load

Fig. 5: Blocking Rate: Scheme-1 vs. Scheme-2 under different load condition

We compared our scheme to a benchmark scheme (LCSel) where all the requests were directed towards the cheapest DC at first (if resources available).

Our approach allows to understand the trade-off study when the SLA violation as a penalty is taken into consideration. Furthermore, the penalty incurred is a parameter in the model that can be adjusted in a sliding scale, as and when needed by the data center service provider.

In our future work, we plan to study the impact of cost increment of bandwidth and energy individually. We also plan to extend our scheme by considering the geographical distance of available DCs from a VN customer in a time of taking the decision of forwarding the request of that customer to a particular DC.

REFERENCES

[1] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

[2] "Trends in data center security: Part 1 traffic trends." [Online]. Available: http://blogs.cisco.com/security/trends-in-data-center-security-part-1-traffic-trends

[3] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 123–134.

[4] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[5] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1431–1439.

[6] Y. Guo, Z. Ding, Y. Fang, and D. Wu, "Cutting down electricity cost in internet data centers by using energy storage," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–5.

[7] Y. Zhang, Y. Wang, and X. Wang, "Capping the electricity cost of cloud-scale data centers with impacts on power markets," in *Proceedings of the 20th international symposium on High performance distributed computing*. ACM, 2011, pp. 271–272.

[8] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *Green Computing Conference (IGCC), 2012 International*. IEEE, 2012, pp. 1–10.

[9] A. N. Sankaranarayanan, S. Sharangi, and A. Fedorova, "Global cost diversity aware dispatch algorithm for heterogeneous data centers," in *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5. ACM, 2011, pp. 289–294.

[10] J. Li, Z. Li, K. Ren, and X. Liu, "Towards optimal electric demand management for internet data centers," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 183–192, 2012.

[11] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D. E. Culler, and R. H. Katz, "Integrating renewable energy using data analytics systems: Challenges and opportunities." *IEEE Data Eng. Bull.*, vol. 34, no. 1, pp. 3–11, 2011.

[12] R. Carroll, S. Balasubramaniam, D. Botvich, and W. Donnelly, "Dynamic optimization solution for green service migration in data centres," in *2011 IEEE International Conference on Communications (ICC)*. IEEE, 2011, pp. 1–6.

[13] Y. Zhang, Y. Wang, and X. Wang, "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2011, pp. 143–164.

[14] N. Buchbinder, N. Jain, and I. Menache, "Online job-migration for reducing the electricity bill in the cloud," in *International Conference on Research in Networking*. Springer, 2011, pp. 172–185.

[15] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *Green Computing Conference, 2010 International*. IEEE, 2010, pp. 3–14.

[16] J. Doyle, D. O'Mahony, and R. Shorten, "Server selection for carbon emission control," in *Proceedings of the 2nd ACM SIGCOMM workshop on Green networking*. ACM, 2011, pp. 1–6.

[17] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen, "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 22.

[18] A. N. Toosi and R. Buyya, "A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2015, pp. 186–194.

[19] A. H. Mahmud and S. Iyengar, "A distributed framework for carbon and cost aware geographical job scheduling in a hybrid data center infrastructure," in *Autonomic Computing (ICAC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 75–84.

[20] J. Srinivas, A. A. M. Qyser, and B. E. Reddy, "Exploiting geo distributed datacenters of a cloud for load balancing," in *Advance Computing Conference (IACC), 2015 IEEE International*. IEEE, 2015, pp. 613–616.

[21] M. M. S. Maswood, C. Develder, E. Madeira, and D. Medhi, "Dynamic virtual network traffic engineering with energy efficiency in multi-location data center networks," in *Proc. of the 28th International Teletraffic Congress*, September 2016, pp. 10–17.

[22] ——, "Energy-efficient dynamic virtual network traffic engineering for north-south traffic in multi-location data center networks," *Computer Networks*, vol. 125, pp. 90–102, 2017.

[23] M. M. S. Maswood, R. Nasim, A. J. Kassler, and D. Medhi, "Cost-efficient resource scheduling under qos constraints for geo-distributed data centers," Under Submission.

[24] M. A. Owens and D. Medhi, "Temporal bandwidth-intensive virtual network allocation optimization in a data center network," in *2013 IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 3493–3497.

[25] H. Khazaei, M. Jelena, and B. Vojislav, "Performance evaluation of cloud data centers with batch task arrivals," *Communication Infrastructures for Cloud Computing*, pp. 199–223, 2013.

[26] D. Medhi and K. Ramasamy, *Network routing: algorithms, protocols, and architectures*. Morgan Kaufmann/Elsevier, 2007.