An Integrated Resource Allocation Scheme for Multi-Tenant Data-center

Mohan Gurusamy, Tho Ngoc Le, Dinil Mon Divakaran
Department of Electrical and Computer Engineering, National University of Singapore
E-mail: {elegm,elelnt,eledmd}@nus.edu.sg

Abstract—The success of multi-tenant data-centers depends on the ability to provide performance guarantees in terms of the resources provisioned to the tenants. As bandwidth is shared in a best-effort way in today’s data-centers, traffic generated between a set of VMs affect the traffic between another set of VMs sharing the same physical links.

This paper proposes an integrated resource allocation scheme that considers not only server-resources, but also network-resource, to decide the mapping of VMs onto servers. We present a three-phase mechanism that finds the right set of servers for the requested VMs, with an aim of reducing the bandwidth on shared links. This mechanism provisions the required bandwidth for the tenants, besides increasing the number of tenants that can cohabit in a data-center. We demonstrate, using simulations, that the proposed scheme accommodates 10%-23% more requests in comparison to a load-balancing allocator that does not consider bandwidth requirements of VMs.

Index Terms—Data-center, resources, bandwidth

I. INTRODUCTION

Cloud providers today are looking forward to leasing out multiple instances of data-centers, or virtual data-centers (VDCs), to tenants. Realization of this dream of multi-tenancy in data-centers requires guaranteeing of predictable performance to the applications or tasks carried out in the VDCs allocated to tenants. Though a VDC will obtain, from the provider, the requested server-resources (computational and storage) allocated for it as well as isolated from other VDCs, the time to complete the tasks running on the VMs (virtual machines) depends not only on these resources, but also on another important resource — the network bandwidth. The bandwidth available for communication between VMs of a VDC is dependent on the traffic between VMs, possibly, belonging to other VDCs. This network resource, unless allocated as part of the VDC, need not be available at the required time on the path(s) between the communicating VMs, resulting in unpredictable delay of the tasks running on the VMs of a VDC. Recent studies have shown that the variability of network performance in data-centers is significant [4], [12], and hence cannot be ignored.

As the performance meted out to a tenant’s VDC depends critically on the network bandwidth, providers need to shun the best-effort way of bandwidth-sharing adopted in today’s data-centers. Instead, network bandwidth should be accounted and provisioned in such a way that maximizes the number of simultaneous VDCs hosted in a physical data-center, with all VDCs having predictable performance. A VDC request can specify the server-resources as well as bandwidth requirement between VMs of the VDC. The allocation manager then has to consider bandwidth requirements, along with server-resources, for allocating a VDC to a tenant. We highlight the importance of this using a simple example.

Consider the scenario given in Fig. 1, which is a simplified version of the three-tier hierarchical architecture widely used in data-centers. There are four servers, each having capacity to host two VMs. Assume 10 units of capacity on each labelled link (in one direction); furthermore assume the unlabeled (server-to-switch) links are not bottlenecks. The server $S_1$ is currently hosting a VM. While in this state, consider a request for a VDC requiring four VMs and having the following inter-VM bandwidth requirements:

$$B = \begin{pmatrix} 10 & 1 & 0 & 0 \\ 0 & 1 & 10 & 0 \\ 10 & 0 & 0 & 1 \\ 0 & 10 & 0 & 0 \end{pmatrix}$$

where $B_{i,j}$ denotes the bandwidth requirement from VM $V_i$ to VM $V_j$.

Let us estimate the bandwidth on the links in two cases where the allocation manager does not take the inter-VM bandwidth requirements into consideration. In the following, we assume a server can support the internal bandwidth required between the VMs it hosts. Note that, due to the symmetry of the requests, the bandwidth in both directions will be equal.

![Fig. 1. An example scenario of a simplified data-center](image-url)
than the link capacities, and hence there will be no congestion by taking bandwidth into consideration. As the mapping has the residual bandwidths on the links available for the VDC. a VDC depending on its bandwidth requirements as well as requirements of a VDC. By doing so, the allocation manager allocates is, to consider for allocation, the inter-VM bandwidth require-
ments (where balancing is with respect to the number of VMs) and then maps them onto server-groups. Depending on the scenarios, we find that the number of VDCs accepted by IRA can be about 10%-23% greater than LBRA.

We develop and discuss in detail on the three-phase integrated resource allocation scheme in the next section. Section III elaborates on the settings, the metrics used for performance evaluation and the scenarios considered; after which the performance evaluation of our proposed scheme is carried out. Related works are briefed in Section IV before concluding in Section V.

II. THREE-PHASE INTEGRATED RESOURCE ALLOCATION

Many different architectures have been proposed for datacenters recently. Some are designed with the aim to minimize the over-subscription in the data-center networks. For example, the over-subscription in a fat-tree is 1:1 [2], which means, for bandwidth allocation, an allocation manager needs to check the residual bandwidth of only those links that connect the edge-switches and the servers. But fat-tree and its successors face a deployment challenge, as they limit incremental expansion of data-centers [13], besides having high complexity [7].

Our focus is on the three-tier hierarchical architecture (refer Fig 3) commonly used in data-centers today [1], so as to allow multi-path routing as well as over-subscription. The three tiers are the core, the aggregation and the edge. This architecture allows over-subscriptions between different levels as discussed in the next section. Importantly, this means that when servers

1) Using the least number of Pods to fit a VDC:
Fig. 2(a) shows one possible mapping, where the allocation manager fits the entire VDC on a single Pod (here Pod 1) — \{V_1, V_2\} is mapped on S_3, and \{V_3, V_4\} on S_4. Summing the traffic demands between \{V_1, V_2\} and \{V_3, V_4\}, the total bandwidth required by this VDC on links l_{0,2} and l_{0,3} will be 22 units each in one direction. These requirements are higher than the link capacities.

2) Mapping VMs onto servers in order:
Fig. 2(b) shows a mapping where servers from left to right are mapped, one after the other, with the maximum number of VMs possible. With such a mapping, the bandwidth required on links l_{0,0}, l_{0,1}, l_{1,0}, l_{1,1} and l_{0,2} in one direction will be 11, 20, 11, 11 and 11 units respectively, resulting in total bandwidth of 128 units. Observe that, this time the links in the higher level (l_{1,0} and l_{1,1}) will be in use, resulting in longer paths and higher delays for communications.

In both the above cases, the allocations require bandwidth greater than the link capacities, and hence would face network congestion when VMs communicate at full rate, leading to unpredictable delay in task completions. While the first allocation has total bandwidth demand of 88 units from the network, the second one demands 128 units. At the same time, the second allocation also uses more number of links than the first one, including the links in higher level of the hierarchy. Hence the tasks running on the VDC obtained through second allocation will be affected by congestion in the links at level 0 and level 1 (as these links may potentially be used by other VDCs). Besides, they can also affect other VDCs sharing these links.

The solution to this problem of unpredictable congestion is, to consider for allocation, the inter-VM bandwidth requirements of a VDC. By doing so, the allocation manager allocates a VDC depending on its bandwidth requirements as well as the residual bandwidths on the links available for the VDC. For the above example, the mapping in Fig. 2(c) is obtained by taking bandwidth into consideration. As the mapping has \{V_1, V_3\} on S_3 and \{V_2, V_4\} on S_4, the bandwidth on l_{0,2} and l_{0,3} is just 4 units each. These bandwidth allocations are less than the link capacities, and hence there will be no congestion when the VMs communicate.

We argue that an integrated approach that provisions computational, storage and bandwidth resources for a VDC request is the need of the hour. Such an integrated resource management scheme should try to meet two important objectives:

1) Allocate server-resources — computational and storage, as well as network resource — bandwidth;

2) Maximize the number of concurrent VDCs hosted.

The second objective is important for providers of data-centers supporting multi-tenancy, as their revenue is a function of the number of VDCs hosted (in addition to the resource requirements of a VDC). While recent works focus on the first objective, the second objective has not received significant attention.

This paper proposes a three-phase integrated resource allocator (IRA) that allocates both server-resources and network-resource as per the demands of the tenants. Assuming network to be the crucial resource for sharing, our scheme attempts to increase the number of concurrent VDCs hosted by forming VM-groups and mapping them onto server-groups in such a way that the inter-group bandwidth — bandwidth between VM-groups — is reduced, leading to more effective use of network link bandwidth. Through simulations, we demonstrate the effectiveness of the proposed scheme. We find that this scheme performs better than a load-balancing resource-allocator (LBRA) that aims to form balanced VM-groups (where balancing is with respect to the number of VMs) and then maps them onto server-groups. Depending on the scenarios, we find that the number of VDCs accepted by IRA can be about 10%-23% greater than LBRA.

Our focus is on the three-tier hierarchical architecture (refer Fig 3) commonly used in data-centers today [1], so as to allow multi-path routing as well as over-subscription. The three tiers are the core, the aggregation and the edge. This architecture allows over-subscriptions between different levels as discussed in the next section. Importantly, this means that when servers

![Fig. 2. Three possible groupings for the example VDC request](image-url)
try to communicate at their link speed, congestion may occur at the aggregation or the core.

A request for a VDC from a user is of the form \((N, R, B)\), where \(N\) is the number of VMs required, \(R\) is a vector of server-resource\(^1\) units required by the VMs, and \(B\) is the the matrix of bandwidth requirements - that is, \(B_{i,j}\) is the bandwidth required from \(V_i\) to \(V_j\). Without loss of generality, we assume every VM in a request requires constant number of server-resource units. That is, instead of the vector \(R\), in this paper we consider each VM (of a VDC) requires a constant \(R\) units of server-resources (specified as part of input), and the total number of server-resource units required of the VDC is thus \(N \times R\). This simplification helps us to focus more on network-resource provisioning.

We use the term server-group to refer to the servers connected to the same edge-switch. Hence, if an entire VDC can be mapped onto a single server-group, in terms of network resources it means, the VDC will use only the links of a single switch — the edge-switch by which it is connected — for communication among its VMs.

Our proposed allocation mechanism proceeds as follows. There is an integrated resource allocator, IRA in short, for a (physical) data-center. The IRA receives requests for allocations of VDCs. There are three phases that the IRA goes through. Phase One splits a VDC into VM-groups depending on the bandwidth required between the VM-groups. Phase Two tries to find a server-group for each of the VM-groups of a VDC. Finally, the resources are allocated in Phase Three. The IRA strives to fit a VDC into a single-server group, and if not into multiple server-groups of the same Pod, and if not, into server-groups of different Pods. The basic idea is to localize the traffic of a VDC.

All the three phases are executed (one after the other) only if a VDC cannot be mapped onto a single server-group. To determine this, on receiving a request the IRA first checks to see if it can fit a VDC onto a single server-group. This is done by (skipping Phase One and) calling the function \textit{find_mapping} in Phase Two (Algorithm 2) for each server-group in the data-center until a \textit{mapping} is found. If so, Phase Two is continued and on successfully finding a mapping, it is enforced in Phase Three. In case Phase Two fails in finding a mapping, it means the VDC cannot be mapped onto a single server-group, and hence (as mentioned earlier) all the three phases are executed one after the other.

A. Phase One - Grouping

As mentioned earlier, grouping is needed only if a VDC cannot be fit into a single server-group. Recall that, all the servers in a server-group are attached to the same edge-switch. In this phase, the VMs of a VDC-request are classified into VM-groups depending on their bandwidth requirements. As the different VM-groups formed are finally mapped onto different server-groups residing in the same Pod or in different Pods, the important cost parameter here is the inter-group bandwidth requirement. Hence, we take inter-group bandwidth requirement as the cost of forming and mapping VM-groups onto different server-groups. For clarity, from now on, we restrict our grouping mechanism to \textit{two-grouping}, where the number of VM-groups formed is only two. The generalization of two-grouping to \textit{n}-grouping is discussed towards the end of this section.

In the algorithm for two-grouping, we determine the cost of different size-combinations of VM-groups. For example, for a VDC of size 10 (i.e., \(N = 10\)), the possible groupings are \(\{9, 1\}\), \(\{8, 2\}\), \(\{7, 3\}\), \(\{6, 4\}\) and \(\{5, 5\}\); and the heuristic given below attempts to find the minimum cost of each such grouping. This problem is similar to the classical problem of finding the minimum cut of an edge-weighted graph (the weight being the bandwidth requirement); the major differences being:

1) If the number of vertices in a graph is \(N\), a minimum-cut algorithm gives two sets of vertices, one of cardinality \(M(< N)\) and the other of cardinality \(N - M\). Let us call this a two-grouping of size \(\{N - M, M\}\). The min-cut cost of this grouping is the minimum among all the grouping. If we are not able to map this grouping (of two VM-groups) onto the data-center, then it would be meaningful to try to map another grouping which has a higher min-cut cost than \(N, M\), but not higher than any of the other possible groupings. Hence, we are interested in finding the cost of different sizes (such as \(\{N - 1, 1\}\), \(\{N - 2, 2\}\), \ldots, \(\{N/2, N/2\}\)) for a given \(N\).

2) Given that we would like to find the min-cut cost for a given size of the grouping, it is possible to obtain such a grouping with more than one cut.

The complexity of a simple min-cut algorithm based on the maximum adjacency search (and not based on max-flow algorithm) is \(O(N|E| + N^2 \log(|N|))\) [14], where \(E\) is the set of edges in the network graph of the VM.

Let \(V\) denote the set of VMs. Algorithm 1 gives the function for two-grouping, \(\textit{find_group}\).

The output of the above function will be in a list \(\zeta\), where each item is of the form \((\{G_1, G_2\}, \beta, f(\beta))\). For each such

\(^1\)For simplicity, we assume the computational and storage requirements can be expressed in terms of server-resource units.
Algorithm 1 Function: find_group(\(\mathbb{N}, B\))

1: \(\zeta \leftarrow ()\); \(G \leftarrow V\); \(\beta \leftarrow 0\)
2: for \(i\) from 1 to \(N/2\) do
3: Select \(v\) from \(G\) s.t. cost(\(v, G\)) is minimum \(\forall v \in G\)
4: \(\beta = \beta + \text{cost}(v, G)\)
5: \(G = G\setminus \{v\}\)
6: \(\zeta, \text{append}((G, V\setminus G), \beta, f(\beta))\)
7: end for
8: return \(\zeta\)

---

item, \(\beta\) is the cost of the grouping \((G_1, G_2)\) in terms of the bandwidth required for communication between (VMs of) the two VM-groups \(G_1\) and \(G_2\). If the VM-groups formed are mapped onto a single Pod the cost of the mapping is \(\beta\), whereas if the VM-groups are mapped onto server-groups of different Pods, the cost is \(f(\beta)\), where \(f\) is such that \(\forall \beta, f(\beta) \geq \beta\). This cost differentiation will help in reducing the bandwidth-use on links that are higher up the tree, thereby reducing the probability of creating bottlenecks at the core of the data-center.

The algorithm starts with the set \(G\) initialized to \(V\), the set of all VMs. It then removes VMs one by one from \(G\) using the criteria given in line 3. The function cost(\(v, G\)) finds the cost of moving VM \(v\) from the group \(G\) to the group \(V\setminus G\). If \(\alpha(v, G)\) denotes the ingress and egress traffic from VM \(v\) to (all other) VMs of group \(G\) (computed from \(B\)), then,

\[
\text{cost}(v, G) = \alpha(v, G) - \alpha(v, V\setminus G),
\]

where \(v \in V\). Once we obtain the cost for different groupings, we proceed to find the server-groups onto which one of these groupings can be mapped to.

**Complexity:** The running time for line 3 is \(O(|E| + N)\), where \(E\) is the set of edges. The for loop runs \(\frac{N}{2}\) times. Hence the complexity of Algorithm 1 is \(O(N|E| + N^2)\).

**B. Phase Two - Resource Discovery**

For the minimum-cost grouping in the list \(\zeta\), the IRA has to check if ‘suitable’ server-groups can be found. A server-group is suitable for mapping (VMs onto it), if it can satisfy both the server-resource and bandwidth requirements of a VM-group in the grouping. The grouping is successfully selected for mapping only if,

1) there is a suitable server-group for every VM-group in the grouping,

2) if there is sufficient bandwidth between the selected server-groups.

If any of the above conditions cannot be met, the grouping is abandoned for mapping, and the next grouping with minimum cost is selected for resource discovery. We recall that, the mechanism always attempts to map all the VM-groups (of a grouping) onto server-groups belonging to the same Pod, and if not, onto server-groups of different Pods.

Let \(n_p\) denote the number of Pods in a data-center, and \(n_e\) the number of edge-switches within a Pod. Every edge-switch is connected to \(n_s\) servers. The total number of servers in a data-center is, therefore, \(n_p \times n_e \times n_s\). Note, the maximum size of a server-group is \(n_s\). For every server-group connected to an edge-switch \(\{e|e \in 1..n_e\}\) in Pod \(\{p|p \in 1..n_p\}\), a matrix \(S_{p,e}\) of size \(n_s \times 3\) is maintained, wherein the residual server-resource (\(p\)), the available ingress bandwidth and the available egress bandwidth for each server in the server-group are stored.

Let \(R_{\text{max}}\) denote the maximum number of server-resource units that a VM can request for. \(C_{p,e}\) is a vector of size \(R_{\text{max}}\), maintaining server-resource information for the server-group at edge \(e\) of Pod \(p\). \(C_{p,e}\) gives the number of VMs with \(i\) server-resource requirement each, that can be allocated in this server-group \(\{p, e\}\). There is also such a vector corresponding to each Pod, \(C_p\), such that \(C_p = \sum_{e=1}^{n_e} C_{p,e}\). The vector \(C_{p,e}\) is computed from the matrix \(S\) mentioned above. For example, if a server has five units of \(p\), then it can fit five one-unit VMs, two two-unit VMs, one three-unit VMs, one four-unit VM or one five-unit VM.

\[
C_{j,e} = \sum_{i=1}^{n_s} S_{i,j,e} / j.
\]

Hence, in a single lookup, it can be determined if a server-group is a potential candidate for allocating VMs of one of the groups. The list of potential server-groups can be found in \(O(n_p n_e)\) steps. Once such a list is formed, a potential server-group is selected (in order) for mapping one of the VM-groups. The next step is to determine if this server-group is suitable for the VM-group. For that, we proceed as stated in Algorithm 2.

Let \(V\) be a matrix with the same shape as \(S\), denoting the requirements of VMs of the selected VM-group within the minimum-cost grouping. map (initialized in line 1) will have the set of sever-VM mappings at the end of the algorithm. After sorting \(S_{p,e}\) using server-resource unit as the index, for each row \(s\) (which is also the server index) in \(S\), that has at least \(R\) units of residual server-resources, the algorithm computes \(n\), the number of VMs that can fit in. \(n\) is at least one. The function find_bw_match does the next task — find \(x, x \leq n\) VMs from the VM-group such that the aggregate ingress-egress bandwidth requirements of these \(x\) VMs can be satisfied by the server’s link. The set of VMs found is assigned to \(M\).

The number of VMs in the VM-group (represented by \(V\) in the above algorithm) is \(m\). Hence the function above returns success only if \(m\) VMs can be mapped.

**Complexity:** The sorting in line 2 is of the order \(O(n_s \log(n_s))\). The running time of find_bw_match is linear in \(m\), as the function searches for the \(x\) VMs whose aggregate bandwidth is less than or equal to the server’s residual bandwidth, by simply adding VMs one after the other. Since there are \(n_s\) servers in a server-group, the complexity of Algorithm 2 is \(O(n_e \log(n_e))\) + \(n_e m\).

The function find_mapping is called for all VM-groups (two, in our study here) of a grouping. If it succeeds in finding suitable server-groups for all the VM-groups in the grouping,
Algorithm 2 Function: find_mapping($S^{p,e}$, $V$)

1: $map \leftarrow \phi$, $m \leftarrow$ number of rows in $V$
2: sort $S^{p,e}$ in increasing order of $\rho$
3: for each row $s$ in $S^{p,e}$ with $R$ or more $\rho$ units do
4:   $n \leftarrow s_p/R$
5:   $M \leftarrow$ find_bw_match($n$, $s$, $\hat{V})$
6:   if $M \neq \phi$ then
7:       $count \leftarrow count + |M|$
8:       $map \leftarrow map \cup \{s, M\}$
9:   if $count = m$ then
10:      return $\{map, p, e\}$
11:   end if
12: end for
13: end if
14: return FAIL

then the IRA checks to find if there is enough bandwidth in the paths connecting these groups\(^2\). If $X_{p,e}$ is the switch where the server-group corresponding to the mapping $\{map, p, e\}$ is located, Algorithm 3 performs this check.

Algorithm 3 Function: find_paths($\{p_1, e_1\}, \{p_2, e_2\}, \beta$)

1: $P \leftarrow$ dfs($X_{p_1, e_1}$, $X_{p_2, e_2}$)
2: $A \leftarrow$ aggregate residual bandwidth in paths of $P$
3: if $A < \beta$ then
4: return NULL
5: else
6: return $P$
7: end if

A DFS (depth-first-search) algorithm is used to find the paths between the two switches. The residual bandwidth of a path is the minimum of residual bandwidth of all the links in the path. The complexity of Algorithm 3 is $O(|E|)$.

Phase Three is invoked if the function find_paths returns a non-empty set of paths between the server-groups; or else, the last discovered VM-group to server-group mapping is ignored, and find_mapping is invoked for this VM-group with the next server-group in the order. This process is continued either until a successful mapping is discovered, or until the list of groupings is exhausted, in which case, the IRA rejects the VDC request.

The selection of potential server-group candidates proceeds in such a way that, server-groups belonging to different Pods are selected for resource-discovery only if there are no potential server-groups within a Pod that are suitable for VM-group mapping. This way, we try to fit a VDC into a Pod as much as possible.

The mapping obtained from Phase Two along with the set of paths $P$ is passed to Phase Three.

\(^2\)We assume multi-path routing is deployed in the data-center.

C. Phase Three - Allocation

In this final phase, the resources are allocated according to the mapping. The server-resources required for VMs are allocated on the corresponding servers, and residual server-resources are updated in the matrices and vectors. Similarly, bandwidth is allocated on the links connecting the selected servers to the edge-switches, and the residual bandwidth is updated. Bandwidth is allocated on all the paths in $P$, such that the aggregate is $\beta$ units. The residual bandwidth on the links are updated in the corresponding data-structures.

On generalizing two-grouping

Though we restrict to two-grouping in this work, it can be extended to $n$-grouping, where $n$ is the number of VM-groups formed. This can be achieved incrementally, by looping over Phase One and Phase Two; i.e., we first start with two-grouping in Phase One, and if the two VM-groups cannot be mapped in Phase Two, we return to Phase One to split the bigger VM-group into two, thereby forming three groups. The cost of this three-grouping is the sum of the bandwidth required between every two of the three VM-groups. From here, we again proceed to Phase Two to discover a potential mapping. This process is repeated until a pre-determined maximum value of $n$ is reached.

III. PERFORMANCE EVALUATION USING SIMULATION

To study the performance of our scheme, we compare it with a load-balancing resource-allocator (LBRA). As in our scheme, the LBRA first attempts to fit a VDC onto a single server-group. If this is not possible, it splits the VMs of the VDCs into two balanced VM-groups — balanced with respect to the number of VMs, and hence the number of server-resource requirements in our case, and then searches for server-groups to map each of the VM-groups. For example, if there are 10 VMs in a VDC request, the LBRA will try to fit all the 10 VMs onto a single server-group. If this is not successful, it splits the VMs into two VM-groups of size five each, and then tries to fit these two VM-groups onto server-groups. If it fails again, it tries grouping into sizes of six and four, and so on and so forth. We assume no other intelligence in formation of two groups — specifically, the LBRA does not consider the bandwidth requirement while forming groups, hence the cost (which is the inter-group bandwidth) is dependant on the set of VMs that formed into the different VM-groups. The LBRA, like ours, also tries to fit the groups within the same Pod if possible, before trying to fit into server-groups of different Pods.

In the following, we first describe the settings and then define the metrics used for performance evaluation. The scenarios are described in Section III-C before presenting the results of the simulations in Section III-D.

A. Settings for simulations

As mentioned earlier, we consider the multi-rooted three-tier hierarchical architecture commonly used in data-centers [1] (refer Fig 3). There are eight core switches and 16 Pods. A
Pod has two aggregation-switches, and 12 edge-switches. The links between the core- and the aggregation-switches, as well as the aggregation- and the edge-switches have capacities of 10G. Each edge-switch has 48 1G ports to connect 48 servers, and two 10G ports to connect to the two aggregation-switches in the same Pod. The edge to aggregation over-subscription is therefore 2.4:1, while aggregation to core over-subscription is 1.5:1. Hence, under high load, the bandwidth available per server is limited to \( \approx 277 \) Mbps. Regarding server-resources, each server was assigned 10 units of server-resources.

VDC requests are generated randomly and fed to the simulator one after the other, until the maximum number of requests is reached. The size of a VDC \( (N) \), in terms of the number of VMs, as well as the server-resource units \( (R) \) of a VM, are both randomly generated. The range of the server-resource requirement of a VM is \([1−2]\). Note, the VM-graph (the graph formed by the network of VMs of a VDC) is represented by \( B \), which gives the bandwidth requirement on links connecting VM-pairs. With probability \( p \), there is a link between a VM-pair of a VDC. The values for inter-VM bandwidth in \( B \) are randomly generated such that the aggregate bandwidth requirement of a VM is 200 Mbps on an average.

For our performance study here, though we set \( f(\beta) = \beta \), the IRA will map VM-groups of a grouping onto server-groups of different Pods, only if they cannot be mapped onto server-groups of the same Pod.

### B. Metrics

We use two metrics for performance evaluation:

1) **Number of VDCs accepted:** As one of our objectives is to increase the number of VDCs that can be concurrently hosted on a data-center, we consider this as an important metric.

2) **Cost:** We assume this cost to be the inter-group bandwidth requirement depending on the VM-groups formed. Hence if an entire VDC is mapped onto a single server-group, the cost incurred is zero, while every VDC that is split into two VM-groups will incur a cost equal to the inter-group bandwidth demand (computed from \( B \)).

### C. Scenarios

We consider three scenarios that are generated by changing two important parameters of the input VDCs: (i) range of VDC sizes, and (ii) distribution of bandwidth required on the links connecting VMs. With respect to Scenario 1, we change each of these to obtain the other two scenarios.

**Scenario 1:**

1. \( (i) \) The range of VDC sizes, in terms of the number of VMs required, is \([120−360]\). (ii) The distribution of bandwidth required on the links is uniform as in Scenario 1.

**Scenario 3:**

1. \( (i) \) The range of VDC sizes is same as in Scenario 1. (ii) The distribution of bandwidth required is non-uniform. 90% of links between VMs have bandwidth requirement of \( x \) units, and remaining 10% have \( 10x \) units of bandwidth requirement, such that the average aggregate bandwidth required by a VM is 200 Mbps. Hence, \( x \) is dependent on \( N \) as well as \( p \).

### D. Results

In the following, we present and discuss the results for the three scenarios. The results are given for two values of \( p \) (the probability of a link between a VM-pair) : 0.25 and 0.75. The input VM-graph is sparsely connected for \( p = 0.25 \) and densely connected for \( p = 0.75 \). For each setting, 10 instances were simulated; i.e., each point on the graphs is the mean of the values obtained from these 10 instances.

1) **Scenario 1:** Fig. 4 plots the number of VDCs accepted for the two allocation schemes against different VDC-sizes given as input. The two curves for each scheme corresponds to the two values for the parameter \( p \). For inputs of VDC-sizes less than or equal to 125, both perform similarly. This happens as there are plenty of resources available initially. But as the total number of VDCs generated increases, either server-resources or bandwidth on links connecting the servers to the edge-switches, or both, become the bottleneck. This means it is not possible to fit a VDC onto a single server-group, necessitating splitting of a VDC request into two VM-groups. Then the inter-group bandwidth plays an important role in deciding the server-groups for the VM-groups. The number of VDCs accepted by the IRA is more than that accepted by the LBRA once there is a bottleneck (at single server-groups), resulting in acceptance of more number of VDCs. The gap between the two methods decreases once the number of available resources becomes too less to satisfy a request. The improvement attained by IRA over LBRA is about 10% here. Fig 4 also reveals that inputs with densely connected VMs face higher probability of rejection with increasing VDC-size.

The graph corresponding to the cost metric is plotted in Fig. 5. As seen in the figure, IRA allocates resources incurring lesser cost than LBRA. Observe that, even in the cases were both methods accepted all VDCs, the cost is lesser in IRA than in LBRA. The higher cost eventually leads to rejection of VDC requests in the LBRA scheme.

2) **Scenario 2:** In this scenario, the VDC-sizes are larger than in the previous scenario. Hence the number of VDCs that gets mapped onto single server-groups is much lesser than in Scenario 1. Fig 6 plots the number of VDCs accepted by both the schemes. As more number of VDCs are now mapped onto two server-groups, the performance gain of IRA over LBRA is higher than in Scenario 1. Here, IRA accepts about 23% more requests than LBRA in this scenario.

Observe that the number of VDCs rejected here is higher than in the previous scenario. We note that, VDCs of high sizes require more than two VM-groups to be accepted. But
since we restricted the number of VM-groups to two (in both the schemes), this resulted in higher number of rejections. In such cases, increasing the number of VM-groups could reduce the number of VDCs rejected.

The cost (plotted in Fig 7) also reflects this trend, where the number of VDCs accepted does not increase beyond a point due to the restriction in the number of VM-groups formed.

3) Scenario 3: In this scenario, we experiment with non-uniform distribution of bandwidth required on links connecting VMs of a VDC. This scenario can be considered as a representative of the kind of VDC requests that in reality arrive at a data-center. The resulting plots are given in Fig. 8. The number of VDCs accepted here is lesser than in Scenario 1 due to the non-uniform distribution of bandwidth requirement. The relative performance gain of IRA over LBRA is higher – IRA accepts about 18% more VDCs than LBRA, in the best case here.

Fig. 9 plots the cost incurred. Unlike Scenario 1, the cost is lesser for denser network. As we have kept the (average) aggregate bandwidth requirement of a VM constant through-out, for a VM-graph with less density, large bandwidth is distributed among small number of links; whereas, for a VM-graph with high density, large bandwidth is distributed among large number of links. The second case leads to more number of VMs being hosted on the same server, and hence the same server-group, thereby resulting in lesser cost.

IV. Related works

Recent research works have started considering allocation of bandwidth to clients. Seawall focuses on enforcing link-bandwidth allocation to competing VMs based on weights assigned to the VMs [12]. Bandwidth obtained by a VM on a shared link is proportional to its weight, and end-to-end bandwidth is the minimum of the bandwidth of all links in the path. In [10], the authors argue that having such a weight as the payment, and guaranteeing minimal network bandwidth to VMs, are two important objectives that data-center networks should strive to meet, while showing there exists tradeoff between these two objectives.
Similar in line with Seawall, Gatekeeper is designed to support bandwidth guarantees [11]. While Seawall shares a link bandwidth among VMs using the link, Gatekeeper can achieve bandwidth-sharing among tenants. Another work developing a pricing model for data-center network has recommended guaranteeing a minimum bandwidth (based on a tenant’s quota) and sharing the spare link-bandwidth in proportion to tenant’s quotas [3].

SecondNet assumes as part of a request, a matrix specifying bandwidth requirement between every VM pairs [6]. The allocation algorithm in SecondNet first locates a cluster based on the server-resource requirement, and then proceeds to build a bipartite graph of VMs and servers based on both the server-resources and bandwidth requirement. A matching is obtained by translating the problem to min-cost network flow, where weight of a link is the used server bandwidth. Hence this mechanism does not try to reduce the bandwidth used on inter-group links. Besides, it is the clustering heuristics that plays major role in determining the VM groups.

In another work that does bandwidth provisioning for VDCs, the authors consider a single aggregate value for (ingress and egress) bandwidth requirement of each VM in a request [4]. The allocation manager searches for a suitable sub-tree at each level that satisfies the server-resources as well as bandwidth requirements. With the constraint of having a single value for bandwidth (say $b$) for every VM, the bandwidth allocated at the outbound link of every server-group hosting $m$ VMs is $\min(m, N - m) \cdot b$. Due to the assumption of single aggregate bandwidth requirement per VM, the problem of reducing the inter-group bandwidth does not arise here.

There are also works that focus on developing a cloud-network platform, for example CloudNaaS [5]. Such a platform can leverage on software-defined networking techniques, say OpenFlow [8], to provide fine-grained control of network.

Observe that, while the above works address the issues related to bandwidth provisioning, none has focussed on reducing the bandwidth between groups of VMs. As far as we know, [9] is the only work that looked into bandwidth between VM-groups. Therein, traffic-aware VM placement problem was identified as an NP-hard problem; and assuming static, single-path routing, the authors proposed off-line algorithms for clustering of server machines and VMs, and a method to find a mapping between them using the traffic between VMs as a metric (but without considering server-resource requirements of VMs). Our work is different, as we focus on the allocation of VDCs, and define successful mapping of the VM-groups in a data-center based on the inter-group bandwidth as well as the server and bandwidth demands of each VM within a group.

V. CONCLUSIONS

In this work, we investigated the problem of hosting multiple tenants in a data-center that share not only server-resources such as processors and storage systems, but also network bandwidth. In this context, we proposed a three-phase integrated resource allocation scheme that maps a VDC onto a data-center based on server-resources as well as network bandwidth. While doing so, our mechanism aims to reduce the bandwidth used between VM-groups of a VDC, thus making space for more VDCs. Our study (using simulations) showed that the IRA scheme is able to accept about 10% to 23% more VDCs than a load-balancing resource-allocator.

We are currently working on extending two-grouping to $n$-grouping, wherein the cost of mapping $n$ VM-groups (of a VDC request) onto the data-center will be determined. We expect more significant improvement with $n$-grouping.

In simulations, the VDC requests were assumed to arrive one after the other; and we did not consider sojourn times of VDCs. In future, we plan to explore true dynamics in the requests, where a VDC’s sojourn time is given as part of the input, such that it departs at the end of the sojourn time.

Yet another interesting direction for future work is to look into the generation of the traffic matrices from simpler user-specifications of requirements, removing the assumption that
user will be able to predict the exact bandwidth requirement between every VM pairs.

VI. ACKNOWLEDGEMENT

This work was supported by Singapore A*STAR-SERC research grant No: 112-172-0015 (NUS WBS No: R-263-000-665-305).

REFERENCES