Heuristics for Consolidation of Ensembles of Virtual Machines

Geetha Sowjanya Akula, Anupama Potluri
School of Computer and Information Sciences
University of Hyderabad, Hyderabad, India
Email: geethasowjanya.akula@gmail.com, apcs@uohyd.ernet.in

***Abstract*—Virtualization has lead to great energy efficiency in data center networks through consolidation of computing and other resources. Despite this, there is still room for innovation when considering the off-peak hours and the consolidation required to power down some of the physical machines. Migration of Virtual Machines (VMs) without consideration of network traffic patterns between these VM can lead to inefficiency. There have been algorithms in literature that have considered traffic or other resource consumption to migrate VMs singly or as an ensemble. But, very few algorithms consider migration of ensembles with consolidation based on the traffic pattern of the migrating VMs. In this paper, we present two heuristics for migration with consolidation of VMs based on their communication graph and other resource requirements such as CPU, memory etc.. The heuristics are based on Prim's Maximum Spanning Tree and a Modified Bread-First Search (MBFS) of the communication graph of VMs identified for migration. The algorithms work by identifying the connected components of the communication graph and placing the VMs of a component in physical machines that are in proximity to each other if it cannot be entirely migrated to a single machine. The MBFS and modified Prim's algorithms are used to partition a component that does not fit entirely into a single machine, such that the partitions can fit into a single physical machine.**

I. Introduction

Data centers consist of thousands or tens of thousands of Virtual Machines (VMs) running on hundreds or thousands of physical machines arranged in hundreds of racks. Energy efficiency is a primary concern in data centers. During off-peak hours, there may be many physical machines running much below the average load. Consolidation of VMs into fewer racks and machines will lead to savings in power consumption and cooling costs. Communication between VMs also contributes to energy costs of the data centers, in addition to latency and inefficient utilization of bandwidth of the switches. When VMs that communicate with each other are co-located in the same physical machine, the data never reaches the physical network card and is handled entirely through software switches such as Open vSwitch (OVS) [1]. This can be utilized to conserve energy by switching off unnecessary ports in the top-of-the-rack (TOR) switches or core switches etc. during off-peak hours.

Recently, algorithms have been proposed for VM placement that are traffic-aware and energy-efficient [3], [8], [2]. There have been algorithms proposed for consolidation of VMs through migration to reduce the number of physical machines that are powered on [9], [11], [4], [5]. [10] proposes an algorithm that is a traffic-aware migration algorithm but does not perform consolidation. Live Migration of Ensembles (LIME) [7] is a migration algorithm that takes into account the traffic of a VM scheduled for migration and moves an ensemble of communicating VMs and the network elements being used by these VMs rather than the single VM alone. However, there is no discussion on consolidation during migration of the ensemble of VMs.

In this paper, we propose heuristics for migration and consolidation of an ensemble of communicating VMs. Our contribution may be stated as follows:

1) We propose a new heuristic for consolidating heterogeneous VMs taking into consideration their communication graph. Most consolidation or migration algorithms do not consider either heterogeneous VMs or their traffic matrix or both.
2) We propose modifications to Breadth First Search and Prim's Maximum Spanning Tree algorithms to partition a set of communicating VMs that do not fit into a single machine such that the communication between VMs hosted on different physical machines is minimized.

The rest of this paper is organized as follows: we review the traffic-aware VM placement and ensemble migration algorithms in Section II. We present our algorithm for migration of an ensemble with consolidation in Section III. We conclude with Section IV.

II. Related Work

LIME [7] is an ensemble migration technique. It performs live joint migration of VMs and the network. Instead of migrating an individual VM it migrates VMs, network and management system because applications running on multiple VMs of a single tenant are tightly coupled with the underlying network state and have high communication with each other.

When frequently communicating VMs are placed on Physical Machines (PMs) that are far apart, it introduces a large amount of network traffic. This results in network congestion affecting the availability of the network for other applications running in the data center. AAGA [3], TVMPP [8], K-means cluster algorithm [12] are some of the solutions in the literature that take this parameter into consideration during VM placement. These algorithms can be extended through simple modifications to handle consolidation. However, the TVMPP algorithm is having a complexity of $O(N^4)$ for each recursive call. AAGA uses the k-cut algorithm and proposes to use

the TVMPP algorithm in future. However, K-means algorithm considers each VM separately and not as an ensemble.

Sercon [9] is a VM consolidation technique that tries to minimize the number of migrations. The key idea is to migrate VMs from a least loaded server onto a heavy loaded server only if a node can be released entirely. However, it does not consolidate taking the communication between VMs into consideration.

III. Heuristics for Migration with Consolidation of Ensembles of VMs

There have been many intelligent VM placement algorithms to ensure that network resource consumption is taken into account [8], [6] and recently for ensemble migration based on communication pattern between the VMs [7]. However, there has been very little work that considers network communication during consolidation of VMs. Many papers also do not consider the case of heterogeneous VMs – i.e., VMs that have varying requirements of memory and CPU power except in a few [9]. In this paper, we propose two heuristics that consolidate heterogeneous VMs of lightly loaded servers of multi-tenant data centers taking into consideration the traffic pattern between them.

We make some assumptions in our problem definition. These are the following:

1) VM placement is done such that the VMs of the same tenant are co-located to the greatest extent possible. This is reasonable for placement algorithms that consider the traffic pattern between VMs.
2) During off-peak hours, a single tenant's VMs may be scattered across many lightly loaded PMs and can be consolidated together.
3) Inter-tenant VM communication is likely to be non-existent or small.
4) We do not consider traffic going out of the data center to other popular resources such as the Google search engine etc.

TABLE I. Notation Used in the Algorithms

Symbol	Description
P	Set of PMs $\{P_1, P_2, \cdots, P_m\}$
P_i	i^{th} PM
VM_{P_i}	Set of VMs on PM P_i
VM_{mig}	Set of VMs to be migrated
M_{sched}	Mapping of VM_{mig} to target PMs
T	Traffic Matrix between VM_{mig}
C_G	Communication graph of VM_{mig}
$\mathbb{C}$	Set of Connected Components of C_G
C_i	i^{th} Connected Component
V_{C_i}	Vertices of C_i
E_{C_i}	Edges of C_i
P_R	Set of target PMs that can accommodate VM_{mig}
P_X	Set of target PMs in whose proximity other PMs need to be found
E_A	Edges incident on a VM set A
$load_u$	Resource usage of VM_u
$load_A$	Resource usage of VM set A
rc_{P_i}	Idle resource of PM_i
D	Distance Matrix of set P

Our algorithm starts off by identifying the physical machines that are loaded beyond a given threshold and those with load less than a second threshold. The first set of servers are not candidates for VMs to migrate into whereas the second set are candidates for shutdown. Next, we identify the set of target PMs that have sufficient residual capacity, P_R, and sort them from least loaded first to the most heavily loaded last. We identify the set of VMs, VM_{mig}, whose load can be accommodated by these physical machines. We construct the communication graph, $C_G = (V_G, E_G)$, of these VMs. The edges of this graph are weighted based on the amount of traffic between the VMs. If there is no communication between a pair of VMs, there is no edge between them. It is likely that this graph C_G is a disconnected graph based on assumption 3. We determine the connected components of C_G, $\mathbb{C}$, and sort them in decreasing order of their size. The consolidation algorithm attempts to migrate the largest connected component first. If we migrate the least connected components first, they will fit into the physical machines identified and the probability that a large connected component fits into a single server is reduced. We also therefore identify the physical machine with the least load as the target for migration. If a connected component can fit into a single physical machine,i.e., function *FitComponent* returns TRUE in Algorithm 1, the migration schedule, M_{sched}, is updated to include the mapping between the VMs of this component and the PM, PM_{R_1} to which they are migrating. The residual capacity of the physical machine, PM_{R_1}, is decremented. If the residual capacity becomes zero, it is removed from the list of target physical machines and the algorithm proceeds to the next connected component.

If a connected component cannot be fit into a single physical machine, the component is further partitioned into smaller components. We perform this partition using one of two methods – a modified BFS algorithm and a modified maximum Prim's algorithm – until the capacity of the target physical machine, $rc_{P_{R_1}}$, is reached. We repeat the algorithm until all connected components are migrated. The start or root node for Prim's and BFS algorithms r, is the node with the largest communication in the selected component. If there is a tie, we break the tie with the node that has the highest degree. This entire procedure is given in Algorithm 1.

Algorithm 1 VM_Consolidation

Require: $P, T, Algo$

```
1: M_sched ⟵ φ
2: VM_mig ⟵ φ
3: P = sort(P)
4: {P_R, t} = partition(P)
5: for i = t → |P| do
6:     VM_mig = VM_mig ∪ VM_{P_i}
7: end for
8: C_G = ConstructCommGraph(VM_mig, T)
9: ℂ = SortedConnectedComponents(C_G)
10: for i = 1 to |ℂ| do
11:     if ¬FitComponent(C_i, P_R) then
12:         r = MaxRankNode(C_i)
13:         lwt = min_{j=1}^{|E_{C_i}|} w_j
14:         ℂ' = Partition(C_i, r, P_R, M_sched, Algo, lwt)
15:         sort(P_R)
16:         ℂ = ℂ ∪ ℂ'
17:         sort(ℂ)
18:     end if
19: end for
20: return M_sched
```

In the modified BFS algorithm, we perform the breadth-first search except that the edges with the least weight of the connected component are not walked. We stop the walk when the capacity of the target physical machine is reached or no more edges with a weight higher than the least weight are incident on the nodes walked. If there are no edges with sufficient weight, it leads to a partition of the connected component. By ignoring the least weight edges, we do not perform a strict BFS. Instead, we proceed to the next level keeping all the nodes that have high communication between them together as long as they fit on a single physical machine. Only the VMs which represent the nodes walked to are migrated to the target physical machine, P_{R_1}, the least loaded of the target PM set, P_R. If the entire component is not migrated to a PM, the part that is migrated is removed from the original component, C_i. A communication graph for the remaining component is constructed, $C_G^{'}$, and the connected components of this graph, $\mathbb{C}^{'}$, are determined. The algorithm attempts to find a PM that can accommodate the largest component of $C_G^{'}$ entirely such that the new PM has the least distance to PM_{R_1}. All these PMs which hold partitions of the same component are added to the set P_X and the new PM found has to be close to at least one of these PMs. By placing the partitions of a single component close together, we reduce unnecessary traffic across multiple switches and especially core switches that connect multiple racks. Any of the partitions that cannot be fit into a single PM are added to the original set of connected components so that later in the iteration, the component is further partitioned. This algorithm is shown in Algorithm 2. The only difference between the MBFS algorithms and the modified maximum Prim's algorithm is that in Prim's technique we construct the maximum spanning tree until the capacity of the target physical machine, rc_{PR_1} is reached. Whenever there are edges with the same weight to be walked, we resolve the tie by considering the sum of weights of the edges from the nodes that are potential candidates to nodes already in the tree. We choose the node with the higher weight since this implies that this node has higher communication with the nodes already present in the spanning tree and is a better candidate to be co-located with the other nodes.

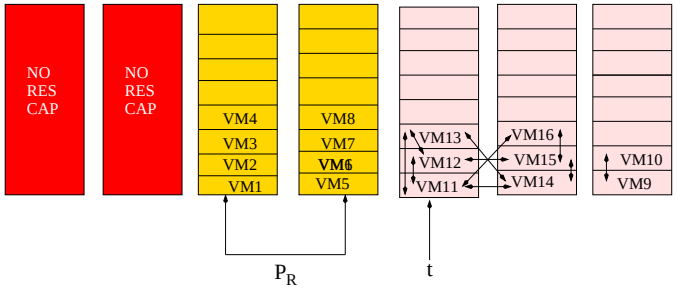

Fig. 1. Set of Physical Machines, Virtual Machines and their current state in our example along with the communication between VMs shown by the arrows

We illustrate the above algorithm with two communication graphs. For this illustration, we assume that all VMs have equal load. We find that the Prim-Partition algorithm works better than MBFS in one whereas in the other, MBFS works similar to Prim-Partition. Our example has seven physical machines (PMs) as shown in Fig. 1, each of which can accommodate 8 VMs at full capacity. The first two PMs in our example

Algorithm 2 Partition

Require: $C_i, r, P_R, M_{sched}, Algo, lwt$

$A \leftarrow \phi$
$P_X \leftarrow \phi$
if $(Algo = MBFS)$ **then**
    Use BFS to walk the component C_i until rc_{PR_1} is not exceeded while ignoring the edges with weight $\leq lwt$ and collect them into set A
else
    Use MaxPrim to construct the spanning tree A such that rc_{PR_1} is not exceeded
end if
$M_{sched} = M_{sched} \cup \{P_{R_1}, A\}$
$rc_{P_{R_1}} = rc_{P_{R_1}} - load_A$
if $(rc_{P_{R_1}} = 0)$ **then**
    $P_R = P_R \setminus P_{R_1}$
end if
if $V_{C_i} = \phi$ **then**
    return NULL
end if
$P_X = P_X \cup P_{R_1}$
$C_G{}^{'} = \{V_{C_i} \setminus A, E_{C_i} \setminus E_A\}$
$\mathbb{C}^{'}$ = ConnectedComponents($C_G{}^{'}$)
$\mathbb{C}^{'} = sort(\mathbb{C}^{'})$
for $i = 1$ *to* $|\mathbb{C}^{'}|$ **do**
    if Proximity($C_i^{'}, P_R, P_X, D$) **then**
        $\mathbb{C}^{'} = \mathbb{C}^{'} \setminus C_i^{'}$
    end if
end for
return $\mathbb{C}^{'}$

are fully loaded and cannot be used for migration. The PMs in yellow are the target PMs, P_R, and they have the residual capacity to accommodate more VMs. The PMs in pale pink can be powered down after migrating all of their VMs as the residual capacity of the PMs in P_R can accommodate all these VMs. The index t indicates the most heavily loaded of the PMs that are marked for shutdown.

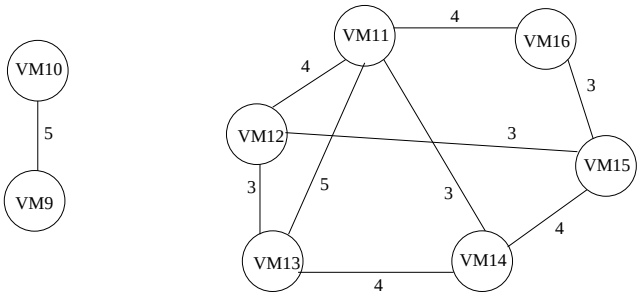

Fig. 2. Example Communication Graph where Prim-Partition and Modified BFS return similar results

In the example graph shown in Fig. 2, the larger connected component consisting of VMs 11-16 is the first one considered for migration since we sort the connected components based on their cardinality. VM11 is selected as the root from which the maximum spanning tree is constructed because it is the node with the highest sum of weight of edges and also has the highest degree. The spanning tree construction stops when four VMs are traversed because the residual capacity of the

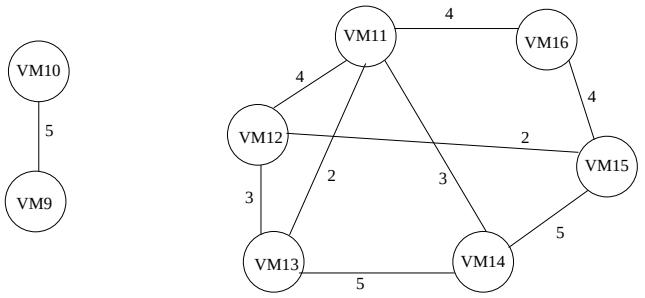

Fig. 3. Example Communication Graph where Prim-Partition works better than Modified BFS

target server is reached. These are VM11, VM12, VM13 and VM14. Here, though the maximum Prim's algorithm can select the subset {VM11, VM12, VM13, VM16}, our algorithm will select VM14 instead of VM16 because VM13 and VM11 are both in communication with VM12 and VM14. The sum of the weights of the edges between these is higher than with VM16. Thus, our modified Maximum Prim's algorithm will ensure that the highly communicating VMs are fit into a physical machine wherever possible. These are migrated to the PM which is least loaded. Since in our example, both target PMs are equally loaded, the lower indexed PM is chosen. The residual capacity of this PM is exhausted by this migration. The remaining VMs of this component, V15, V16, can be accommodated in the only other available PM in P_R. In the next iteration of the Algorithm $VM\_Consolidation$, the component {VM9, VM10} will be migrated to the same PM. When MBFS is run on this graph, it migrates VMs 11, 12, 13 and 16 to the first PM. The remaining VMs 14 and 15 and the next component comprising of VMs 9 and 10 are all migrated to the only other PM. The inter-PM communication cost, i.e., the sum of edges that cross the PMs, is 11 in the case of Prim-Partition and 13 in the case of MBFS.

In the example graph shown in Fig. 3, for the Prim's algorithm the VMs 11, 12, 15 and 16 are migrated to the first PM and the VMs 13 and 14 of this component and the next component are migrated to the second PM. In MBFS, the schedule is VMs 11, 12, 14 and 16 are migrated to the first PM and 13, 15 of this component and the next component are migrated to the second PM. The inter-PM cost of communication in this example is 13 for Prim-Partition and 21 for MBFS.

IV. Conclusion and Future Work

In this paper, we have proposed an algorithm to consolidate an ensemble of VMs identified for migration. We construct the communication graph of the VMs that are to be migrated and identify the connected components of this graph. We attempt to migrate each component as a whole to a single physical machine based on the load of the VMs of the component and the residual capacity of the target physical machine. Where this is not possible, we use either the modified breadth-first search algorithm or a modified Prim's maximum spanning tree algorithm to partition the component. The partitions are migrated to physical machines in proximity to each other based on their distance matrix. Wherever the partition cannot be entirely migrated, it is further partitioned.

In future, we plan to implement these algorithms and compare them against the TVMPP and K-means consolidation algorithms. We plan to integrate this into LIME architecture to also migrate the related network components as needed.

References

[1] "Open vswitch: A open virtual switch," http://openvswitch.org/.

[2] M. Al Shayeji and M. Samrajesh, "An energy-aware virtual machine migration algorithm," in *Advances in Computing and Communications (ICACC), 2012 International Conference on*, 2012, pp. 242–246.

[3] J. Chen, K. Chiew, D. Ye, L. Zhu, and W. Chen, "AAGA: Affinity-aware grouping for allocation of virtual machines," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, 2013, pp. 235–242.

[4] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 594–601.

[5] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 26–33.

[6] D. Huang, D. Yang, H. Zhang, and L. Wu, "Energy-aware virtual machine placement in data centers," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, 2012, pp. 3243–3249.

[7] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, "Live migration of an entire network (and its hosts)," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 109–114.

[8] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[9] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," vol. 28, no. 3, 2011, pp. 212–231.

[10] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," in *Parallel Processing (ICPP), 2010 39th International Conference on*, 2010, pp. 228–237.

[11] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 71–75.

[12] B. Zhang, Z. Qian, W. Huang, X. Li, and S. Lu, "Minimizing communication traffic in data centers with power-aware vm placement." in *IMIS*, I. You, L. Barolli, A. Gentile, H.-D. J. Jeong, M. R. Ogiela, and F. Xhafa, Eds. IEEE, 2012, pp. 280–285.