Online Traffic-aware Virtual Machine Placement in Data Center Networks

Daniel S. Dias and Luís Henrique M. K. Costa
Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
Rio de Janeiro, Brazil
Email: {dias, luish}@gta.ufrj.br

Abstract—**Data centers rely on virtualization to provide different services over a shared infrastructure. The placement of the different services and tasks in the physical machines is crucial for the performance of the whole system. A misplaced service can overload some network links, lead to congestion, or even connection disruptions. On the other hand, virtual machine migration allows reallocating services and changing the traffic matrix, leading to more efficient use of bandwidth. In this paper, we propose a Virtual Machine Placement (VMP) algorithm to (re)allocate virtual machines in the data center servers, based on the current traffic matrix, CPU, and memory usage. Analyzing the formation of community patterns in terms of traffic using graph theory, we are able to find virtual machines that are correlated because they exchange high amount of data. Those virtual machines are aggregated and allocated to servers as close as possible to each other, reducing traffic congestion. Our simulation results show that VMP was able to improve the traffic distribution. In some specific cases we were able to reduce 80% of the core traffic, concentrating it at the edge of the network.**

I. Introduction

Data centers allows computing power aggregation and the support of ever larger applications. Previous work [1], [2] have shown that virtualization improves the flexibility of data centers, for example to improve energy efficiency. With virtualization it is possible to deploy many isolated services into a single hardware. Using virtualization, we can increase the total usage of CPU cycles, increase the available memory or disk space, and control the placement of services in different machines in the data center network. Cloud computing strongly relies on virtualization to share the physical resources among multiple users.

Data center providers take the expected service demands into account to design their server and network infrastructure. An oversized data center is cost ineffective. The physical interconnection of servers, link capacities, and how much redundancy is needed by the application are some key parameters of data center design. Nevertheless, service demands may change over time, because application demands and usage are dynamic. The traffic matrix of a data center may even change during the periods of the day [3].

In this paper, we propose a virtual machine placement (VMP) algorithm to (re)allocate virtual machines (VMs) among the servers of a data center, based on the traffic matrix in the network. The first part of VMP consists of analyzing the topology, making partitions of servers that have a high capacity of connectivity (usually servers directly connected to the same switch), accounting for the amount of CPU and memory available in those servers. Then, VMP gathers the amount of data exchanged between virtual machines and the usage of CPU and memory of each virtual machine, clustering correlated virtual machines. We correlate virtual machines by analyzing the amount of traffic exchanged and aggregate them into clusters by searching community patterns in the traffic. Then, VMP tries to fit all the clusters in the partitions, ensuring that the partition has enough CPU and memory resources to handle all of the virtual servers designated to it.

The rest of this paper is organized as follows. Section II describes related work. Section III presents the basic idea and operation of the algorithm. Section IV evaluates the algorithm proposal. Finally, we conclude the paper and describe future work in Section V.

II. Related Work

Different aspects of data center networks have been studied in the literature. Kandula *et al.* [4] study one large data center running map-reduce applications, gathering information from the network traffic. They conclude that 86% of the aggregation and core switches presented congestion of more than 10 seconds. Using common methods to infer a data center traffic matrix from link level, SNMP data do not produce satisfactory results. In this scenario, comparing to the package level data, there is a mean error of 60% on the traffic estimation.

Benson *et al.* [5] show that data centers based on three-tier tree topologies concentrate the traffic in top-of-rack switches. The study also reveals that patterns of virtualization and consolidation are not yet noticed. They conclude that operators choose the location where services will be deployed, therefore the positioning is not done randomly. On those data centers, core switches have high utilization, with a high variation in a day, and aggregation and edges switches have less traffic, with low variation. Their work shows that the placement of VMs is important and must be different according to the type of data center but also that the traffic distribution is dynamic.

Benson *et al.* [6] analyze 19 data centers with different topologies and applications, using packet traces and SNMP statistics. The topology is a 2-tier or a 3-tier on all of them but the applications vary among them, making the number of flows and total traffic on the network have a different behavior for each data center. They observe some traffic characteristics, like the ON/OFF pattern of traffic arrival rate, with a log-normal distribution. We use those conclusions to model our traffic generator in Section IV.

Other proposals suggest the optimization of virtual machine placement in the data center. Sandpiper implements automated detection and migration of virtual machines [7]. Sandpiper tries to identify and solve CPU and memory hotspots in a data center, monitoring the usage of each server and the needs of each virtual machine, and migrating virtual machines as needed. Different from our work, Sandpiper does not take the network traffic into account, therefore we can not compare the data center network performance to VMP.

Meng *et al.* [8] discuss virtual machine placement in the data center. They formulate a minimization problem based on fixed costs of required bandwidth of each virtual machine and the cost of communication between the virtual machines, using min-cut algorithms to find divisions. CPU and memory resources are not used in the algorithm since they assume that each virtual machine has the same size and each server supports a fixed number of VMs. They show that the virtual machine placement as an optimization problem is NP-hard. To reduce the complexity of the proposed algorithm, they analyze the data using a scenario with constant traffic or grouping specific VMs, reducing the number of analyzed components.

Wang *et al.* [9] propose a model of VM consolidation with network bandwidth constraint imposed by network devices, where VMs with known bandwidth demands must be consolidated into a number of servers with identical capacity limit. Their work is not concerned with traffic engineering of the core network, it only models the VM consolidation into each server of the network. The model does not use CPU, memory and core utilization in the calculations.

Biran *et al.* [10] describe a minimization problem to determine the location of VMs in a data center based in the network bandwidth, CPU, and memory resources. Their formulation is complex and does not scale to the size of data centers, thus they also create 2 heuristics based on the minimization algorithm. The algorithm has to be executed in a tree topology which can be achieved by transformation, assuming that any topology can be transformed. The calculation is made off-line, and at every change it needs to be executed. They assume that each user has a specific number of VMs that can only talk to each other, therefore all the VMs are already clustered in their approach.

None of the above proposals can identify cluster patterns using only the traffic matrix, needing extra information. In an environment where predefined clusters do not exist, the algorithms can not be executed. As our synthetic data and the real data are not properly clustered, we can not compare VMP with those algorithms. VMP treats both the classification into clusters of VMs and the deployment of them in the host, allocating the VMs to minimize the impact in the network while using CPU and memory constraints to size the clusters.

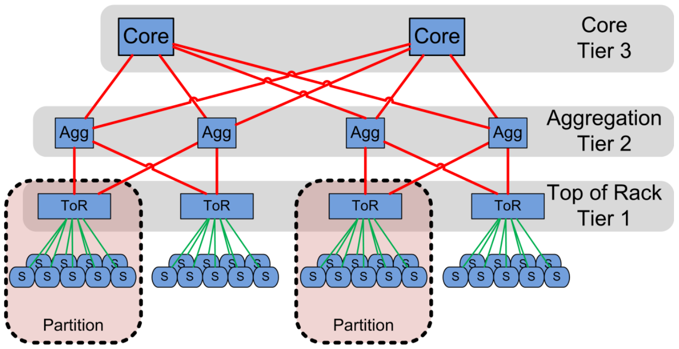


Figure 1. Three-tier topology with partition example.

III. Virtual Machine Placement Algorithm

According to Cisco [11], most data centers use a three-tier architecture. The lowest tier is composed of the physical hosts, connected to a switch in each rack. Those top-of-rack (ToR) switches are connected to intermediary switches, forming the aggregation tier. On the last layer there are the core switches, used to interconnect the aggregation switches. In the transition of each layer, the number of switches decrease from the previous tier while the link capacity increases. Thus, the core of the network is under-provisioned while the bandwidth of top-of-rack switches may be wasted. Figure 1 illustrates a three-tier topology with 2 core switches, 4 aggregation switches, and 8 ToR switches.

VMP considers topological characteristics and the current data center network load, CPU and memory usage of virtual machines to avoid bottlenecks. It is designed to migrate the virtual machines that have correlated services and exchange more traffic to closer placements in the topology producing less traffic in the network core. The basic idea is to consolidate those correlated virtual machines in the same host server or, if not possible, in the same rack, leading to more usage of parts with less utilization in the topology.

In the process of finding the right spot for each virtual machine, CPU and memory usage of virtual machines are used. CPU is measured by the number of cores needed by each VM and the availability of cores in each server. Memory usage is accounted by the gigabytes of memory allocated to each VM and by the total memory available in each server. A virtual machine or a group of them, can only be instantiated in a server with enough free CPU cores and memory. Figure 1 illustrates our concept of partition in a three-tier topology. The dashed line over the servers and the top of rack switch form a partition of the data center with high degree of communication, since all the servers are connected to the same switch, being able to exchange data at full link speeds.

The last point to look at is when VMP operates. VMP is meant to work online and it should be able to respond to traffic pattern changes throughout the day. In our implementation, for 1,000 nodes VMP runs in less than 1 second and for 10,000 nodes in 3 minutes with good results. Those tests do not take into consideration the time needed to gather all the information from the nodes nor the time needed to execute the migrations, because both of them depend on the data center characteristics like the sampling frequency of the traffic matrix, the number of deployed VMs and the time to migrate a VM.

To the best of our knowledge, there is no study specific to the traffic matrix characterization of data center networks. Nevertheless, the work of Benson *et al.* [5] shows that each type of data center presents a specific pattern over the time. In some of them the number of congested links can vary by up to 25% in a couple of hours. The frequency of execution will logically depend on the variation of the traffic matrix. Since a migration should not last less than a second [12], for 1,000 machines our implementation should work correctly. The execution time of VMP is analyzed in Section IV.

A. VMP Operation

The operation of VMP can be divided into four stages: Data acquisition (Section III-A1), responsible for collecting data from each virtual machine and from the data center topology; Server partitioning (Section III-A2), which discovers the topology and finds the best division; Clustering virtual machines (Section III-A3), which associates correlated virtual machines and put them into previously formed clusters; and Data output (Section III-A4), responsible for analyzing and triggering the migrations.

1) Data acquisition: VMP uses three types of data as input: the traffic between all the virtual machines, the CPU and memory consumption of each virtual machine and the topology of the data center, including how many virtual machines a server can hold by analyzing CPU and memory installed on each server. The traffic exchanged between each pair of virtual machines is used to create a traffic matrix where the entries are the sum of all the traffic, the uplink plus the downlink of each virtual machine. The matrix is quadratic, with the value 0 in every entry in the main diagonal, since the loopback interface of virtual machines are not monitored.

The topology of the data center, including CPU and memory installed, should be known. The algorithm needs to know the portion of the infrastructure that forms partitions with a high degree of connectivity plus the CPU and memory resources that each partition can hold. In each kind of topology, in each different corporation there is a distinct way to interconnect servers. It is worth noting that VMP does not need a specific topology to execute, it can run in trees, fat-trees [13], in Bcube [14] and other kinds of topologies.

The most used topologies in data centers share similarities. All of them have a portion, usually called core, that is responsible for connecting big segments of the network. This portion is frequently unable to support all the traffic generated simultaneously for all downlink servers, due to cost or technology limitations. If all the servers interconnected by this core exchange traffic, congestion will occur and the full adapter bandwidth will not be available to the servers, leading to loss and delays.

Therefore, traffic engineering should push traffic to the edges of the network, reducing it in higher layers of the topology. In fat-tree topologies, a pod is a good definition of a division. In a tree topology, the leaves of the tree or the rack are good definitions of a division. In a Bcube topology a division can be formed with all the servers that connect to the same switch. As we discuss next, there are various possibilities to divide a topology and each one offers some advantages.

2) Server partitioning: Considering that most data centers are based in three-tier architectures, we propose a simple algorithm to find those servers with a higher degree of connectivity. Network topologies used in data centers have an oversubscription of core and aggregation links, making them overused while links at the edges, on the access layer, are underused. This oversubscription ratio makes natural partitions on the data center. Traffic from servers at the same portion of the access layer, or same switch, can exchange traffic at full speed while traffic that needs to go up a layer or two to get to the other server may suffer from congestion at some links. We use this natural division to define our partition with high degree of connectivity. We assume that servers that use the same rack and the same top-of-rack switch can connect using full bandwidth of the network adapter, having a high degree of connectivity and contributing to decrease the amount of traffic that transit at higher layers. We call this selection a *partition of servers*.

Those partitions of servers that can host VMs are then arranged and the total amount of CPU, memory and virtual machines is totalized. We measure the number of cores as the metric for the CPU resource and the size in megabytes for memory. Note that CPU and memory are only used to assure that a partition can hold all the VMs assigned by the VMP.

Depending on the choice of the partitions, the algorithm will not be able to determine the specific server for a virtual machine. In that case, we need to divide the problem. If we use the virtual machines that were allocated into a partition and further divide that partition, for example choosing each server as a partition, we can determine in which servers each virtual machine will be deployed.

3) Clustering virtual machines: In this stage VMP analyzes the virtual machine traffic matrix. We model the data center network as a graph where virtual machines are vertices and network links are edges weighted according to the traffic exchanged between the VMs. The algorithm tries to cluster virtual machines that exchange traffic using the concept of graph community. According to Porter [15], a community consists of a group of nodes that are relatively densely connected to each other but sparsely connected to other dense groups in the network.

Community finding algorithms are CPU intensive. Thus, we focus in finding an algorithm that could run in less time for our specific case. Over the algorithms that we have studied, the Girvan-Newman algorithm is the one with the best results [16], [17], [18], in finding communities, but it is very CPU intensive. Nevertheless, this algorithm can be modified to reduce the complexity, as we show next.

According to Porter [15], an edge has high betweenness if it lies on a large number of short paths between vertices.

If one starts at a node and wants to go to some other node in the network, it is clear that some edges will experience a lot more traffic than others. The betweenness of an edge quantifies such traffic by considering strictly shortest paths or densities of random walks between each pair of nodes and taking into account all possible pairs. Due to a particularity of the problem, we can do one simplification. The shortest path calculation and count are not needed. All the machines can connect to any other machine in the data center, therefore we have a complete graph in the beginning, and the shortest path has one edge for every pair of virtual machines. Instead of counting how many times an edge is used as the shortest path between two nodes, we use the edge weight to determine ”how much” one virtual machine is connected to another. This weight represents the traffic between virtual machines, therefore when VMs are not much correlated, this value is low, while for VMs that share services that are communication-intensive, this value is high. Using this value we can locate communities based on the traffic exchange between two VMs.

After finding a community, we need to allocate all the nodes that are part of the community into the servers previously aggregated into partitions. VMP only needs to decide the right partition where the community will be deployed. At this moment, the amount of CPU and memory necessary for each VM are added together to form the CPU and memory requirements of the cluster. Then, VMP matches the cluster resources with the resources available at each partition of servers, with the possibility that a community may not fit into a cluster. For example in the first attempt the algorithm will try to allocate all the nodes into a single partition. Most of the times this can not be done, therefore the process to find more connected communities begins.

Two tests are performed after each round of the community finding algorithm. First it looks for all of the connected subgraphs in the graph. Each subgraph is a community and we call this phase as *Clustering virtual machines*. Next, the algorithm tests if the community can fit into any partition of servers. If all the virtual machines of that cluster can fit into a partition, meaning that the sum of CPU cores and memory size needed by all nodes in that cluster can be assigned to a partition of servers, the allocation is made and those VMs are marked as allocated. At the same time the resources on the partition are updated with new values, removing the usage needed by the virtual machines allocated to it.

The decision of where a cluster should be allocated is a bin packing problem [19]. The bins are the partition of servers and the items to be packed are the cluster of virtual machines. We use three common greedy approaches [19], namely first fit, best fit, and worst fit. We compare them in Section IV.

After all the communities that can be allocated are found, we need to check if all the nodes were designated to a server. If some nodes were not allocated, the algorithm needs to find a smaller community, a community that can fit into some partition of servers. At this moment VMP removes a percentage of edges with low weight, disconnecting some of the subgraphs, making them smaller.

The proposed algorithm works as follows: first VMP identifies all the connected subgraphs that are part of the main graph containing all the virtual machines. Each subgraph or community is then tested to fit into one of the partitions of the servers. Two actions can happen here, if a partition of servers can support it, the nodes of that community are allocated into the servers, removing the allocated the resources on the partition and marking as placed the nodes from the main graph. The second action happens if the community can not fit into any partition of servers, forcing the algorithm to check the next community. Once all the communities to allocate are tested, the round is over. At that time we need to try to expand the number of communities. We do this by removing the edges with lowest weight and going back to check for subgraphs. This is done until there are no unmarked nodes in the main graph. Our modification of the Girvan-Newman algorithm combined with the proposed allocation scheme is represented in the pseudo-code bellow.

Algorithm 1 Placement of virtual machines

```
$G \leftarrow$ Graph of virtual machines;
$P \leftarrow$ Partitions of servers;
while $G \neq \emptyset$ do
    Identify all connected subgraphs $C$ of $G$;
    for all subgraph $C \leq P$ do
        Remove the edges and the nodes of $C$ in $G$;
        Update $P$ with new allocations;
    end for
    Remove $n$ edges from $G$ with the lowest weight;
end while
```

Figures 1 and 2 illustrate VMP operation. The partition is represented by the dashed line over the physical servers. In Figure 2, first we see the representation of virtual machines (vertices) and the traffic matrix (edges) as a graph. The graph is a mesh but, for better viewing, some edges were not represented as each vertex should connect to all other vertices. In the second portion of the figure the clustering process is represented. In this stage the lower value edges are removed from the graph and the connected vertices form a community. In the last stage, the communities are allocated into previously selected partitions of physical servers.

4) Data output: The output of the allocation algorithm is the location of the virtual machines in the data center. Using the previous location and the suggested location of the virtual machines VMP defines the migrations. All the suggested migrations should be analyzed before since some of them can cause undesirable effects. For example, VMP can suggest migration of the same virtual machine at each round, wasting bandwidth for each migration. One possible solution is to mark the communities that had VMs migrated on previous turns, using a counter to avoid frequent migration. Another solution to be incorporated is the priority to migrate virtual machines that transits more traffic in core links. And a last solution to be incorporated is to start the allocation algorithm with some machines bonded to a specific cluster, this will

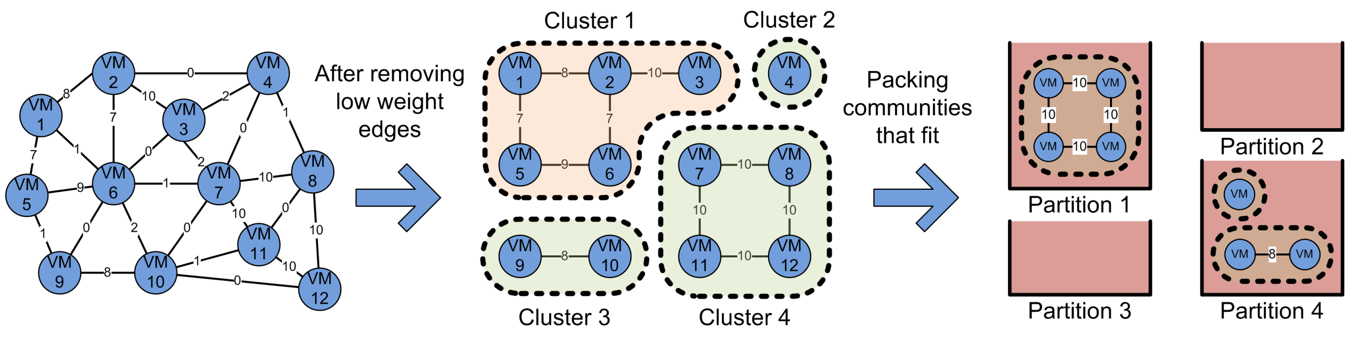


Figure 2. Clustering plus bin packing example.

guarantee that some migrations are going to be avoided.

The trend in data center networks is that some specific clusters are always composed of the same virtual machines or at least have a high quantity of the same virtual machines in a sample period. When VMP is allocating a specific cluster it can not ensure that in subsequent executions a specific cluster will remain on the same partition of servers. In a future work, we will treat this issue, for now, our algorithm is used only to evaluate the impact in the network traffic.

IV. Results

Data center networks can have thousands of machines, therefore packet-level simulator are not well suited because of the long simulation time. Some data-center specific simulators do exist [20], [21], but they do not respond to all of our needs. For example they do not permit model of traffic matrix as input, the implementation of the modified Girvan-Newman algorithm, and gathering info on the traffic from individual nodes. As a consequence, we have written a simulator in C++. The following sanity checks were made. Duplicity on the allocation of virtual machines, duplicity of servers, variable overflow, CPU, memory parameters, and if at the end of execution the sum of variables like CPU, memory, total traffic were consistent with the start of the execution.

We performed simulations using VMP algorithm and an algorithm that randomly allocates virtual machines into partitions. The partitions are the same for both algorithms, the only difference is that VMP finds clusters and allocates them while the random algorithm only allocates VMs. At the bin packing portion of the algorithm each virtual machine/cluster is assigned to a partition. Restrictions of CPU and memory are used, to ensure that the servers can sustain the VMs that will be deployed on them. In both approaches we use the first fit, the best fit and the worst fit in the bin packing problem [19]. All the simulations were executed on a Core i7-3930K at 3.20 GHz with 8 GB of memory.

A. Data input characteristics

The traffic matrix is generated using a log-normal distribution, with density function shown in Equation 1. We choose that specific distribution based on the work of Benson *et al.* [6], which shows that the traffic arrives in a node and leaves a node following a log-normal pattern. As shown in Equation 1, each element of the traffic matrix is generated using a log-normal distribution, with mean (μ) and standard deviation (σ) as variables.

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}, x > 0 \quad (1)$$

In all tests, the mean μ is set to 0 while the standard deviation σ changes between simulations. Changing σ we can modify the pattern of the traffic matrix. Increasing σ we create a more scattered and a high total value of traffic between virtual machines. With σ above 3, we have substantial differences in the traffic matrix, creating an environment where some virtual machines are linked with a high traffic exchange.

We execute each simulation 10 times, calculating the average and a 95% confidence interval, represented by error bars in the graphs. The number of machines in the test is compatible with the literature [4], [5]. Most papers suggest that a data center can have between 500 and 10,000 machines. The virtual machine resources are sorted by a normal distribution, or fixed. Each virtual machine may have 1 or 2 CPUs and between 256 MB and 2048 MB of memory. All the host servers have the same configuration, having 8 cores and having 16 GB of memory. The number of servers varies with the simulation and are explained in the simulations. The bin packing portion of the algorithm is simulated using all the three algorithms: worst fit, first fit, and best fit. For sake of simplicity, we only plot the results of the worst fit algorithm when the differences between them are not significant.

B. Simulations

To evaluate the performance of VMP, we perform a series of simulations varying the number of virtual machines in the topology and the fraction of edges that are removed from the graph at each round. We fixed the standard deviation value to 4 and varied the number of virtual machines from 1,000 to 10,000, increasing the number of servers as we increase the number of virtual machines to get all the servers completely used. The number of partitions is 20 for all the cases and each virtual machine has fixed CPU and memory.

Figure 3 plots the execution time of our implementation of VMP. This time only accounts for the algorithm computation,

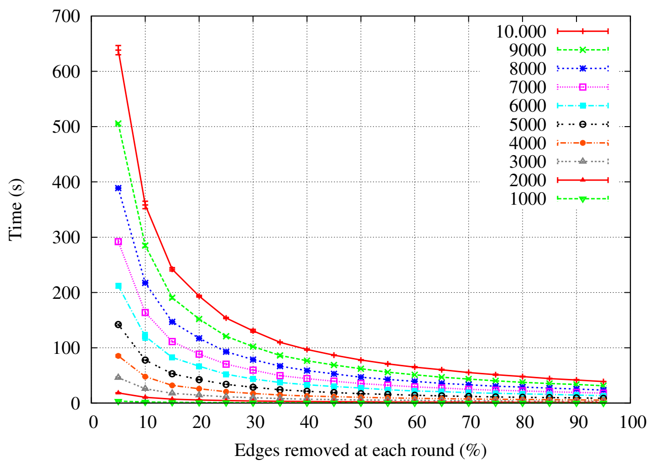


Figure 3. VMP simulation time.

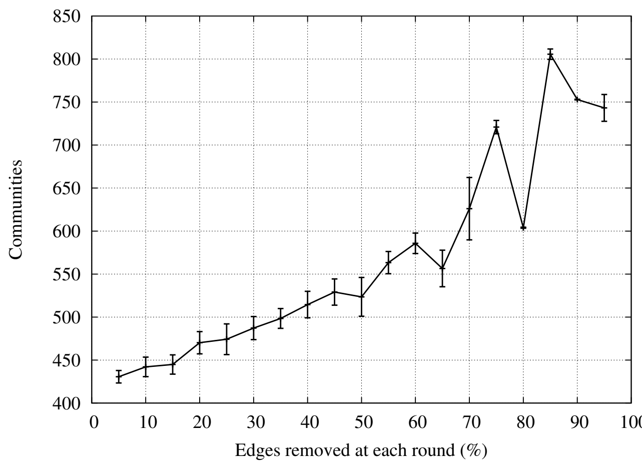


Figure 4. Communities found by percentage of edge removed per round.

not the time to gather the input data. Two aspects are worth noting: the exponential increase of the time with the number of nodes and the reduction of the execution time when we increase the number of edges removed by round, represented by the percentage over total edges, the X axis of Figure 3.

To evaluate the execution time and the performance of our modification of the Girvan-Newman algorithm we analyze the number of communities when we vary the quantity of edges removed by round or when we vary the standard deviation of the traffic matrix. The quantity of edges removed by round affects the number of communities that can be found. If we remove too much edges at each round we can not find communities. If we remove too few edges, the algorithm takes longer to converge. We took the characteristic of the previous simulation with 1,000 VMs and investigate the number of communities formed. In Figure 4 we see that removing 35% or less of edges in each round we locate communities with the average of more than 2 virtual machines in each community. Removing 35% of edges instead of 10%, we can improve the execution time of the algorithm by more than five times in the best case. In the simulations we remove 10% of edges per round. We are not interested in performance since we are in a controlled environment testing the benefits of the algorithm.

The standard deviation of the input traffic matrix changes the ability to form communities. We vary the number of VMs and the standard deviation. We set the quantity of edges removed by round to 10% and the partition size to 20, calculating the number of servers to be fully occupied with VMs. We plot in Figure 5 the number of communities found by varying the standard deviation and the number of nodes. In this figure, less communities is better, since it means that our algorithm can bond many nodes into a single community.

Figure 5 shows that with low standard deviation ($\sigma < 1$) VMP has difficulties finding communities. Above ($\sigma > 1$), the number of communities found stabilizes. This means that VMP can form well defined communities when the traffic between virtual machines has significant differences. When

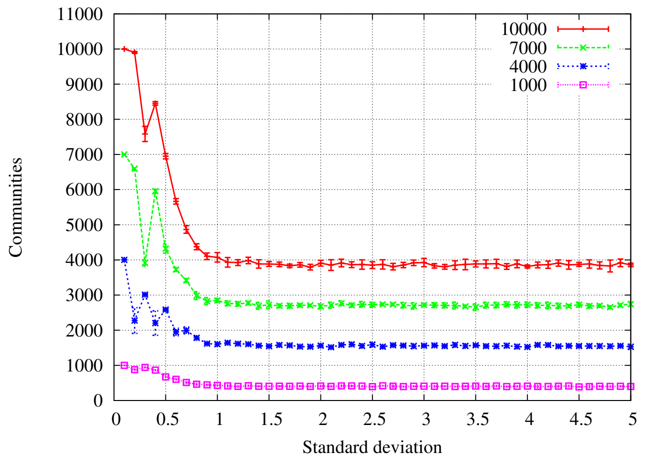


Figure 5. Communities found by varying number of nodes and standard deviation.

there is high traffic concentration, there is less virtual machines exchanging traffic, and thus it is easier to find communities.

To see the improvement that VMP can bring to data centers, we execute a simulation changing the standard deviation of the input matrix. The objective is to simulate the traffic in different categories of data centers, the ones that have a characteristic of few changes in the pattern and low traffic, with low level of standard deviation ($\sigma < 2.5$), and the ones with high degree of changes and a high volume of traffic, having a high standard deviation ($\sigma > 2.5$). We use the same parameters as before, with 1,000 virtual machines.

Figure 6 shows the proportion of traffic that leaves all the clusters over the whole traffic of the data center, or in other words, the amount of traffic that is supposed to transit at the core of a data center. We can see a significant improvement in our proposal over the traffic-agnostic method for positioning virtual machines. When the volume of traffic is high ($\sigma > 2.5$) VMP reduces the traffic that needs to transit at the core by up to 20% of the total traffic in the data center, making it transit

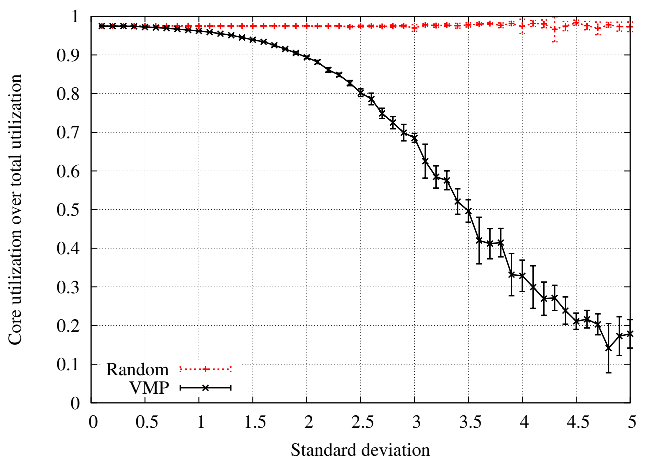


Figure 6. Network core use.

only inside the partitions of the data center.

To evaluate the influence of the size of the data center in the results, we execute two simulations varying the number of servers per partition and the number of VMs. On both situations we set σ to 3 and the quantity of edges removed by round to 10%. In the first analysis we fix the number of partitions to 10, changing the number of nodes and the number of servers per partition to get a data center fully allocated with VMs. In this simulation, shown in Figure 7(a), we see how the algorithm behaves when scaling the size of the data center.

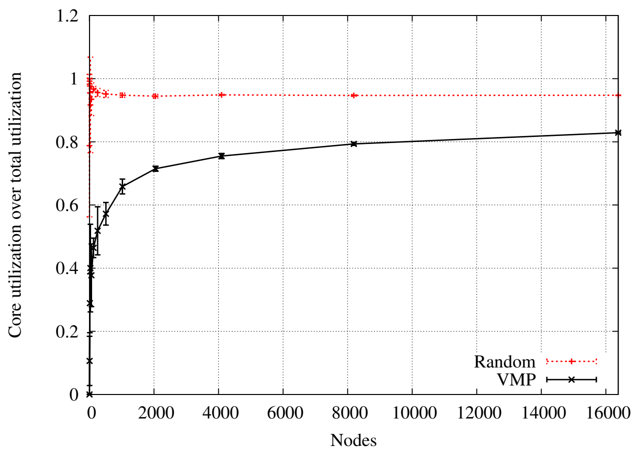
In the second analysis, we fix the number of VMs in the data center to 1,000 and start to increase the size of the data center. Using 5 partitions, we increase the number of servers in each partition of the data center. In this simulation, shown in Figure 7(b), we see how the algorithm behaves when having a demand lower than have all server fully occupied by VMs.

When the number of nodes increases, Figure 7(a), the core utilization tends to stabilize around 82% of core utilization. This means that in the worst case, when there are 16,384 virtual machines, we could expect a 12.5% improvement over the algorithm that does not analyze the placement of virtual machines based on traffic.

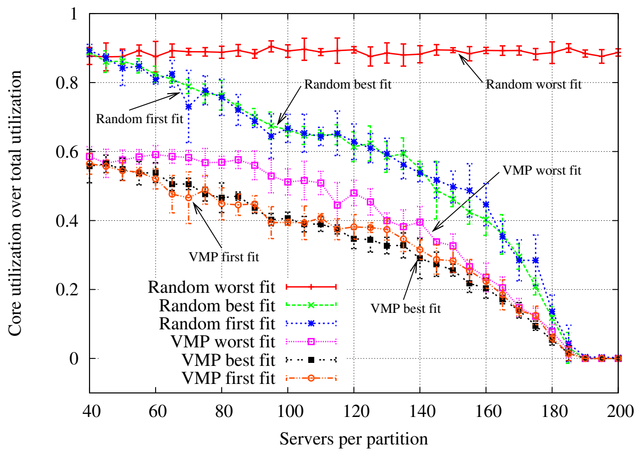
When the number of servers per partition is increased without changing the number of virtual machines, Figure 7(b), it means a less crowded data center. The core utilization goes to near 0% because we have many spare servers. VMP always provides better results than the random placement algorithm.

Finally, we look for the differences in the bin packing algorithms. The simulations were executed using 15 partitions, 1,000 virtual machines and 10% of the edges removed per round. For these simulations, the number of servers per partition was changed from 96% of occupation of the servers to 48% of occupation of the servers.

The distributions can be seen in Figure 8. While at 96% we can see few variations in the deployment of VMs, with 48% occupation we can see an almost flat distribution for the worst fit algorithm and a concentrated distribution for the first and



(a) Increasing the number of virtual servers.



(b) Increasing the number of servers per partition.

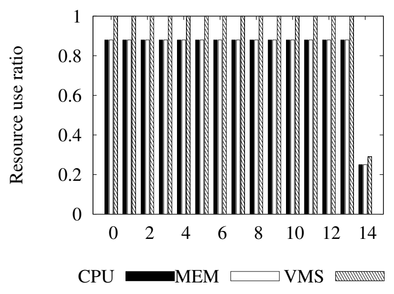
Figure 7. Topology influence in the core traffic.

best fits. The worst fit algorithm is therefore best suited for the problem, since it guarantees spare capacity on the network and on the servers, which can absorb some unpredicted demand in traffic, CPU, or memory. The first and the best fit algorithms tend to concentrate the allocation of VMs into few partitions. Therefore those methods could be used to save energy by turning off an entire unused portion of the data center.

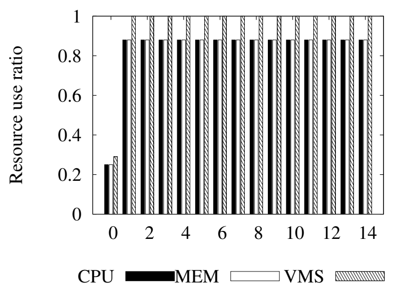
V. Conclusion

In this work we have proposed an algorithm to mitigate the problem of traffic concentration in data center networks. With the deployment of VMs on host servers in a optimized way, we can improve the throughput of the network, moving the traffic from the core switches to the edges switches.

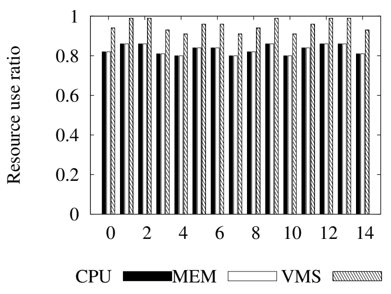
VMP is different from previous work since it is an online algorithm that can be executed in feasible time, responding traffic pattern changes in a fast manner. Using simulations, we show that our proposal can scale to big data centers, with over 16 thousand machines and still be executed in a feasible time, having a improvement of 12.5% over no management on the placement of virtual machines.



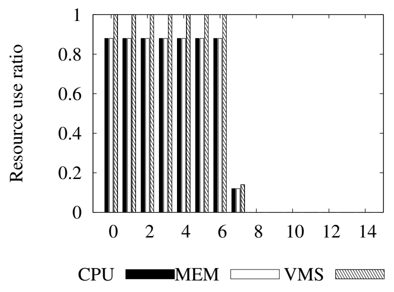
(a) Best fit with 100% of occupation.



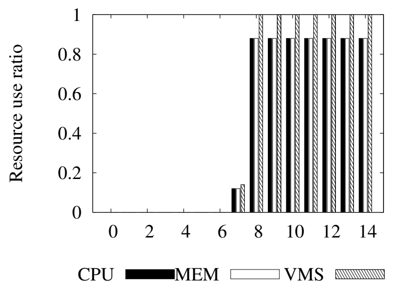
(b) First fit with 100% of occupation.



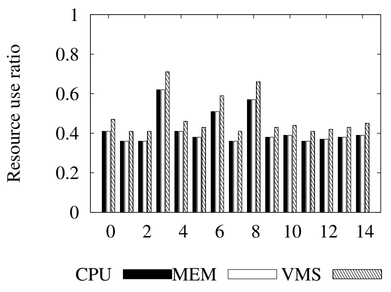
(c) Worst fit with 100% of occupation.



(d) Best fit with 50% of occupation.



(e) First fit with 50% of occupation.



(f) Worst fit with 50% of occupation.

Figure 8. Influence of the bin packing algorithm.

In best case scenarios VMP has the potential to reduce more than 80% in the use of the core network, moving a big portion of the traffic to the edge of the network. This leads to a big improvement on the quality of the network traffic, specially on the bandwidth available for the VMs at the data center.

Several research directions can be identified. First we need to address the migrating analysis, considering how migrations will affect the network or other VMs. Second we need to adjust our algorithm with data from diverse models of data centers, which execute different applications. Our third step is to implement a prototype of VMP as a solution for migrating VMs in a data center environment.

Acknowledgment

This work was supported by FINEP, FUNTTEL, CNPq, CAPES, FUJB, and FAPERJ.

References

[1] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, p. 4008, Jul. 2008.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[3] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 74–85, Aug. 2011.

[4] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09, 2009, pp. 202–208.

[5] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10, 2010, pp. 267–280.

[6] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[7] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923 – 2938, 2009.

[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*, march 2010, pp. 1 –9.

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

[10] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, may 2012, pp. 498 –506.

[11] "Cisco data center infrastructure 2.5," Tech. Rep.

[12] P. S. Pisa, N. C. Fernandes, H. E. T. Carvalho, M. D. D, M. Elias, M. Campista, L. H. M. K. Costa, and C. M. B. Duarte, "Openflow and xen-based virtual network migration," vol. 327, pp. 170–181, 2010.

[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008*. ACM, 2008, pp. 63–74.

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," pp. 63–74, 2009.

[15] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," 2009.

[16] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, p. 066133, Jun 2004.

[17] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[18] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004.

[19] E. Coffman Jr, M. Garey, and D. Johnson, "Approximation algorithms for bin packing: A survey," in *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996, pp. 46–93.

[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, 2011.

[21] T. R. Henderson, M. Lacage, and G. F. Riley, "Network simulations with the ns-3 simulator," 2008.