VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers

Vijay Mann, Avinash Kumar, Partha Dutta, Shivkumar Kalyanaraman

VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers

Vijay Mann[1], Avinash Kumar[2], Partha Dutta[1], and Shivkumar Kalyanaraman[1]

[1] IBM Research, India
{vijamann, parthdut, shivkumar-k}@in.ibm.com
[2] Indian Institute of Technology, Delhi
{avinash.ee}@student.iitd.ac.in

Abstract. Networking costs play an important role in the overall costs of a modern data center. Network power, for example, has been estimated at 10-20% of the overall data center power consumption. Traditional power saving techniques in data centers focus on server power reduction through Virtual Machine (VM) migration and server consolidation, without taking into account the network topology and the current network traffic. On the other hand, recent techniques to save network power have not yet utilized the various degrees of freedom that current and future data centers will soon provide. These include VM migration capabilities across the entire data center network, on demand routing through programmable control planes, and high bisection bandwidth networks. This paper presents VMFlow: a framework for placement and migration of VMs that takes into account both the network topology as well as network traffic demands, to meet the objective of network power reduction while satisfying as many network demands as possible. We present network power aware VM placement and demand routing as an optimization problem. We show that the problem is NP-complete, and present a fast heuristic for the same. Next, we present the design of a simulator that implements this heuristic and simulates its executions over a data center network with a CLOS topology. Our simulation results using real data center traces demonstrate that, by applying an intelligent VM placement heuristic, VMFlow can achieve 15-20% additional savings in network power while satisfying 5-6 times more network demands as compared to recently proposed techniques for saving network power.

Keywords: networking aspects in cloud services, energy and power management, green networking, VM placement
Corresponding author: Vijay Mann, IBM Research, 4, Block C, Institutional Area, Vasant Kunj, New Delhi - 110070, INDIA

1 Introduction

In current data centers, networking costs contribute significantly to the overall expenditure. These include capital expenditure (CAPEX) costs such as cost of networking equipment which has been estimated at 15% of total costs in a data center [6]. Other networking costs include operational expenditure (OPEX)

such as power consumption by networking equipment. It has been estimated that network power comprise of 10-20% of the overall data center power consumption [9]. For example, network power was 3 billion kWh in the US alone in 2006.

Traditionally, data center networks have comprised of single rooted tree network topologies which are ill-suited for large data centers, as the core of the network becomes highly oversubscribed leading to contention [7]. To overcome some of these performance problems of traditional data center networks, such as poor bisection bandwidth and poor performance isolation, recent research in data center networks has proposed new network architectures such as VL2 [7], Portland [16], and BCube [8]. Data centers are also increasingly adopting virtualization and comprise of physical machines with multiple Virtual Machines (VMs) provisioned on each physical machine. These VMs can be migrated at downtime or at runtime (live). Furthermore, emergence of programmable control plane in switches through standardized interfaces such as Openflow [3], has enabled on demand changes in routing paths.

With the above recent advances, opportunities have emerged to save both network CAPEX and OPEX. Network CAPEX is being reduced through several measures such as increased utilization of networking devices, and a move towards cheaper and faster data plane implemented in merchant silicon, while all the intelligence lies in a separate sophisticated control plane [3]. Components of OPEX, such as network power costs, are being reduced through techniques that switch off unutilized network devices as the data center architectures move to higher levels of redundancy in order to achieve higher bisection bandwidth. However, most of the recent techniques to save network power usage do not seem to utilize all the features that current and future data center will soon provide. For instance, recent techniques to save network power in [9] exploit the time-varying property of network traffic and increased redundancy levels in modern network architectures, but do not consider VM migration. On the other hand, current VM placement and migration techniques such as [19] mainly target server power reduction through server consolidation but do not take into account network topology and current network traffic. A recent work by Meng et. al [15] explored network traffic aware VM placement for various network architectures. However, their work did not focus on network power reduction.

There are multiple reasons why saving network power is increasingly becoming more important in modern data centers [13]. Larger data centers and new networking technologies will require more network switches, as well as switches with higher capacities, which in turn will consume more power. In addition, rapid advances in server consolidation are improving server power savings, and hence, network power is becoming a significant fraction of the total data center power. Also, server consolidation results in more idle servers and networking devices, providing more opportunity for turning off these devices. Finally, turning off switches results in less heat, thereby reducing the cooling costs.

In this context, this paper presents VMFlow: a framework for placement and migration of VMs that takes into account both the network topology as well as network traffic demands, to meet the objective of network power reduction while satisfying as many network demands as possible. We make the following contributions:

1. We formulate the VM placement and routing of traffic demands for reducing network power as an optimization problem.
2. We show that a decision version of the problem is NP-complete, and present a fast heuristic for the same.
3. We present the design of a simulator that implements this heuristic and simulates its runs over a data center network with a CLOS topology.
4. We validate our heuristic and compare it to other known techniques through extensive simulation. Our simulation uses real data center traces and the results demonstrate that, by applying an intelligent VM placement heuristic, VMFlow can achieve 15-20% additional savings in network power while satisfying 5-6 times more network demands as compared to recently proposed techniques for saving network power.
5. We extend our technique to handle server consolidation.

The rest of this paper is organized as follows. We give an overview of related research in Section 2. Section 3 presents the technical problem formulation. Section 4 describes our greedy heuristic. We describe the design and implementation of our simulation framework and demonstrate the efficacy of our approach through simulation experiments in Section 5. Section 6 extends our technique to handle server consolidation. Finally, we conclude the paper in Section 7.

2 Background and Related Work

Data center network architectures. Conventional data centers are typically organized in a multi-tiered network hierarchy [1]. Servers are organized in racks, where each rack has a Top-of-Rack (ToR) switch. ToRs connect to one (or two, for redundancy) aggregation switches. Aggregation switches connect to a few core switches in the top-tier. Such a hierarchical design faces multiple problems in scaling-up with number of servers, e..g, the network links in higher tiers become progressively over-subscribed with increasing number of servers. Recently, some new data center network architectures have been proposed to address these issues [7, 8, 16].

VL2 [7] is one such recently proposed data center network architecture (Figure 1), where the aggregation and the core switches are connected using a CLOS topology, i.e., the links between the two tiers form a complete bipartite graph. Network traffic flows are load balanced across the aggregation and core tier using Valiant Load Balancing (VLB), where the aggregation switch randomly chooses a core switch to route a given flow. VL2 architecture provides higher bisection bandwidth and more redundancy in the data center network.

Traffic-aware VM placement. VM placement has been extensively studied for resource provisioning of servers in a data center, including reducing server power usage [19]. Some recent papers have studied VM placement for optimizing network traffic in data centers [12, 15]. These approaches differ from VMFlow in two important ways: (a) the techniques do not optimize for network power, and (b) their solutions do not specify the routing paths for the network demands.

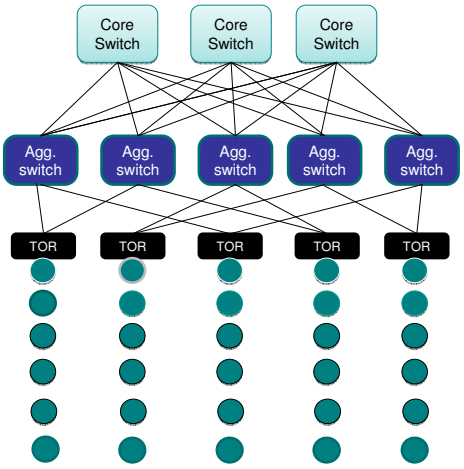


Fig. 1. CLOS network topology

Network power optimization. Most of the recent research on data center energy efficiency has focused on reducing the two major components of data center power usage: servers and cooling [6, 9]. Recently, some papers have focussed on reducing the power consumed by networking elements (which we call, network power) in a data center [9, 13, 17]. In the ElasticTree approach presented in [9], a network power manager dynamically turns on or off the network elements (switches and links), and routes the flows over the active network elements, based on the current network traffic demands. In the primary technique presented in ElasticTree, called greedy bin-packing, every demand is considered sequentially, and the demand is routed using the leftmost path that has sufficient capacity to route the demand. Here, a path is considered to be on the left of another path if, at each switch layer of a structured data center topology, the switch on the first path is either to the left or is identical to the switch on the second path. ElasticTree also proposes an topology-aware approach that improves upon the computation time as compared to greedy bin packing − however, it does not improve the network power in the solution. The VMFlow framework that we present in this paper fundamentally differs from ElasticTree [9] and from the work in [13, 17] because we exploit the flexibility of VM placements that is available in the modern data centers. Also, for a given demand, we jointly perform the VM placement and flow routing by greedily selecting the VM placements as well as the routing paths that require minimum additional network power.

3 Network-Aware VM Placement Problem

In this section we present the Network-Aware Virtual Machine Placement (NAVP) Problem. Our problem formulation extends the ElasticTree formulation in [9].

Input. The data center network is composed of network switches and links that connect the hosts (physical servers). The data center network topology is modeled using a weighted directed graph $G = (V, E)$, where V are the set of vertices and the $E \subseteq V \times V$ is the set of directed edges. A link $e = (u, v)$ has capacity (maximum supported rate) of $C(e)$. There are three types of vertices in V: the switches, the hosts, and a special vertex v_E. Vertex v_E models the external clients that communicate with the data center. The edges represent the communication links between the switches, and between the switches and the

hosts. (We use edges and links interchangeably in this paper.) Let there be n hosts $H = \{h_1, \ldots, h_n\}$, and q switches $W = \{w_1, \ldots, w_q\}$.

We consider a set of m Virtual Machines (VMs) $M = \{vm_1, \ldots, vm_m\}$, where $m \leq n$, and at most one VM can be placed on a host (we consider the case of multiple VMs per server in Section 6). The network traffic source or destination is one of the m VMs or an external client. We model the traffic to and from any external clients as traffic to and from v_E, respectively. Let M' be $M \cup \{v_E\}$. We are given a set of K demands (rates) among the nodes in M', where the j^{th} demand requires a rate of r_j from source $s_j \in M'$ to destination $d_j \in M'$, and is denoted by (s_j, d_j, r_j).

When a switch w_i or a link e is powered on, let $P(w_i)$ and $P(e)$ denote the power required to operate them, respectively. A *VM placement* is a (one-to-one) mapping $\Pi : M \rightarrow H$ that specifies that the mapping of VM vm_i to host $h_{\Pi(vm_i)}$. In addition, we assume that in all VM placements, v_E is mapped to itself.

Constraints. We model the VM placement problem as a variant of the multi-commodity flow problem [4]. A *flow assignment* specifies the amount of traffic flow on every edge for every source-destination demand. We say that a flow assignment on G satisfies a set of demands if the following three standard constraints holds for the flow assignment: edge capacity, flow conservation and demand satisfaction. (Please see [14] for the detailed constraints.)

In addition, we consider another set of constraints that result from the power requirements: (1) a link can be assigned a non-zero flow only if it is powered on, and (2) a link can be powered on only if the switches that are the link's end nodes are powered on. Thus, the *total (network) power* required by a flow assignment is the sum of the power required for powering on all links with non-zero flow assignment, and the power required for powering on all switches that are end nodes of some link that has a non-zero flow assignment.

Due to adverse effect of packet reordering on TCP throughput, it is undesirable to split a traffic flow of a source-destination demand [11]. Therefore, the NAVP problem requires that a demand must be satisfied using a network flow that uses only one path in the network graph (*unsplittable flow constraint*).

The optimization problem. Given the above-mentioned K demands among the nodes in M', we say that a given VM placement Π is *feasible* if there is a flow assignment on G that satisfies the following K demands among the hosts: the j^{th} demands requires a rate of r_j from source host $\Pi(s_j)$ to destination host $\Pi(d_j)$. Then the Network-Aware VM Placement (NAVP) problem is stated as follows: among all possible feasible VM placements, find a placement and an associated flow assignment that has the minimum total power.

Problem Complexity: The following decision version of the NAVP problem is NP-complete: given a constant B, does there exist a feasible VM placement and an associated flow assignment that has total power less than or equal to B. We show the NP-completeness by reduction from the bin packing problem [18].

Theorem 1. *The decision version of the Network-Aware VM Placement (NAVP) problem is NP-complete.*

Please see [14] for the proof.

4 Heuristic Design

We now present a greedy heuristic for the NAVP problem. The algorithm considers the demands one by one, and for each demand, the algorithm greedily chooses a VM placement and a flow assignment (on a path) that needs minimum increase in the total power of the network. We now describe the algorithm in more details. (The pseudocode is presented in Figure 2.)

Primary variables. The algorithm maintains four primary variables: W_{on} and E_{on} which are the set of switches and edges that have been already powered on, respectively, the set H_{free} of hosts that have not yet been occupied by a VM, and a function RC that gives the residual or free capacity of the edges. For each VM v, we assume that initially the placement $\Pi(v)$ is set to $\perp$.

In each iteration of the main loop (in function VMFlow), the algorithm selects a demand in the descending order of the demand rates, and tries to find a feasible VM placement. If a feasible VM placement is found, then the primary variables are updated accordingly; otherwise, the demand is skipped.

Residual graph. To find a feasible VM placement for a demand (s_j, d_j, r_j), the algorithm construct a residual graph G_{res}. The residual graph contains all switches, and all hosts where VMs s_j and d_j can be possibly placed (sets $\Phi(s_j)$ and $\Phi(d_j)$, respectively). G_{res} also contains every edge among these nodes that have at least r_j residual capacity. Therefore, in G_{res}, any path between two hosts has enough capacity to route the demand.

The algorithm next focuses on finding VM placements for s_j and d_j such that there is path between the VMs that requires minimum additional power. To that end, the nodes and edges in the residual graph are assigned weights equal to the amount of additional power required to power them on. Thus, the weight of a switch v is set to 0 if it already powered on, and set to $P(v)$, otherwise. Edges are assigned weights in a similar manner. However, the weights of the hosts are set to ∞ so that they cannot be used as an intermediate node in a routing path. Next, the weight of a path in the residual graph ($pathWt$) is defined as the sum of the weights of all intermediate edges and nodes on the path (and excludes the weights of the end nodes of the path).

VM placement. In the residual graph G_{res}, the algorithm considers all possible pairs of hosts for placing s_j and d_j (where the first element in the pair is from $\Phi(s_j)$ and the second element is from $\Phi(d_j)$). Among all such pairs of hosts, the algorithm selects a pair of hosts that has a minimum weight path. (A minimum weight path between the hosts is found using a variant of Dijkstra's shortest path algorithm whose description is omitted here due to lack of space.) The source and the destination VMs for the demand (s_j and d_j) are placed at the selected

hosts, and the flow assignment is done along a minimum weight path. Note that, sometimes G_{res} may be disconnected and each hosts may lie in a distinct component of G_{res}. (For instance, this scenario can occur when the residual edge capacities are such that no path can carry a flow of rate r_j.) In this case, there is no path between any pair of hosts, and the algorithm is unable to find a feasible VM placement for this demand. Depending on the data center's service level objectives, the algorithm can either abort the placement algorithm, or continue by considering subsequent demands.

Input: described in input part of Section 3
function Initialization
  $W_{on} \leftarrow \emptyset$; $E_{on} \leftarrow \emptyset$ {*set of switches and edges currently powered on*}
  $H_{free} \leftarrow H$ {*set of hosts currently not occupied by a VM*}
  $\forall e \in E$, $RC(e) \leftarrow C(e)$ {*current residual capacity of edge e*}
function VMFlow
  select a demand (s_j, d_j, r_j) from the set of demands in the descending order of their rates
    if $\Pi(s_j) = \perp$ **then** $\Phi(s_j) \leftarrow H_{free}$ **else** $\Phi(s_j) \leftarrow \{\Pi(s_j)\}$
    if $\Pi(d_j) = \perp$ **then** $\Phi(d_j) \leftarrow H_{free}$ **else** $\Phi(d_j) \leftarrow \{\Pi(d_j)\}$
    $V_{res} \leftarrow W \cup \Phi(s_j) \cup \Phi(d_j)$ {*residual nodes*}
    $E_{res} \leftarrow \{(u,v) \in E : (RC(e) \geq r_j) \wedge (u, v \in V_{res})\}$ {*residual edges*}
    $G_{res} \leftarrow (V_{res}, E_{res})$ {*residual graph*}
    $\forall v \in W$, **if** $v \in W_{on}$ **then** $WT_{res}(v) \leftarrow 0$ **else** $WT_{res}(v) \leftarrow P(v)$ {*switch wt. in G_{res}*}
    $\forall v \in \Phi(s_j) \cup \Phi(d_j)$, $WT_{res}(v) \leftarrow \infty$ {*host wt. in G_{res}*}
    $\forall e \in E_{res}$, **if** $e \in E_{on}$ **then** $WT_{res}(e) \leftarrow 0$ **else** $WT_{res}(e) \leftarrow P(e)$ {*edge wt. in G_{res}*}
    $minWt \leftarrow Min\{pathWt(G_{res}, u, v) : (u \in \Phi(s_j)) \wedge (v \in \Phi(d_j)) \wedge (u \neq v)\}$ {*see Sec. 4*}
    if $(minWt < \infty)$ **then** {*found a routing for this demand*}
      $(\Pi(s_j), \Pi(d_j)) \leftarrow$ any (u,v) s.t. $(u \in \Phi(s_j)) \wedge (v \in \Phi(d_j)) \wedge (u \neq v) \wedge (pathWt(G_{res}, u, v) = minWt)$
      assign the minimum weight path P from $\Pi(s_j)$ to $\Pi(d_j)$ to the flow for (s_j, d_j, r_j)
      for all switches v on path P, $W_{on} \leftarrow W_{on} \cup \{v\}$
      for all edges e on path P, $E_{on} \leftarrow E_{on} \cup \{e\}$; $RC(e) \leftarrow RC(e) - r_j$
      $H_{free} \leftarrow H_{free} \setminus \{\Pi(s_j), \Pi(d_j)\}$
    else
      skip demand (s_j, d_j, r_j) {*cannot find a VM placement for this demand*}

Fig. 2. A greedy algorithm for NAVP

It is easy to see that each iteration selects a placement and flow routing that require minimum increase in network power. Thus, although the heuristic is sub-optimal, each iteration selects a locally optimal placement and routing.

Lemma 1. *In each iteration, for a given traffic demand, the heuristic selects source and destination VM placements and a flow routing that need minimum increase in network power.*

A practical simplification. We now present a simple observation to reduce the computation time of our heuristic. We observe that in most conventional and modern Data Center network architectures, multiple host are placed under a Top-of-the-Rack (ToR) switch. Thus, for a given demand, if a source (or destination) VM is placed on a host, then the parent ToR of the host needs to be turned on (if the ToR is not already on) to route the demand. Therefore, for VM

placement, we first try to place both the source and destination VMs under the same ToR. If such a placement is possible, then only the common parent ToR needs to be turned on to route the demand. If no such ToR is available, then we compute a minimum weight path between all possible pairs of ToRs (where path weights include the end ToR weights), instead of computing minimum weight paths between all possible pairs of hosts. This simplification significantly reduces the computation time because the number of ToRs is much lower than the number of hosts.

5 Experimental Evaluation

We developed a simulator to evaluate the effectiveness of VMFlow algorithm. It simulates a network topology with VMs, switches and links as a discrete event simulation. Each server (also called host) is assumed to host at most one VM (the case of multiple VMs per host is discussed in Section 6). VMs run applications that generate network traffic to other VMs, and VMs can migrate from one node to the other. Switches have predefined power curves − most of the power is static power (i.e. power used to turn on the switch). At each time step, network traffic generated by VMs (denoted by an entry in the input traffic matrix) is read and the corresponding VMs are mapped to hosts as per the algorithm described in Section 4. The corresponding route between the hosts is also calculated based on the consumed network power, the network topology and available link capacities. At each time step of the simulation, we compute the total power consumed by the switches and the fraction of the total number of network demands that can be satisfied. A new input traffic matrix (described below), that represents the network traffic at that time stamp, is used at each time step of the simulation.

Network Topology: The simulator creates a network based on the given topology. Our network topology consisted of 3 layers of switches: the top-of-the-rack (ToR) switches which connect to a layer of aggregate switches which in turn connect to the core switches. Each ToR has 20 servers connected to it. At most one virtual machine (VM) can be mapped to a server (the case of multiple VMs per server is discussed in Section 6). We assume a total of 1000 servers (and VMs). There are 50 ToR switches, and each server connects to a ToR over a 1 Gbps link. Each ToR connects to two aggregate switches over 1 or 10 Gbps links. We assume a CLOS topology between the aggregate and core switches similar to VL2 [7] with 1 or 10 Gbps links as shown in Figure 1. All switches are assumed to have a static power of 100 watts. This is because current networking devices are not energy proportional and even completely idle switches (0% utilization) consume 70-80% of their fully loaded power (100% utilization) [9].

Our simulator has a topology generator component that generates the required links between servers and ToRs, ToRs and aggregate switches, and aggregate switches and core switches. It takes in as input the number of servers and the static power values for each kind of switch. It then generates the number of ToR switches (assuming 20 servers under 1 ToR) and the number of aggregate and core switches using the formulae given in [7] for a CLOS topology.

For *(d*d)/4* ToRs, the number of aggregate switches are d and the number of core switches are *d/2*. For 1000 servers and 50 ToRs, this results in around 15 aggregate switches and around 8 core switches.

Input Data: To drive the simulator, we needed a traffic matrix (i.e. which VM sends how much data to which VM). Such fine grained data is typically hard to obtain from real data centers [5, 10] because of the required server level instrumentation. We obtained data from a real data center with 17,000 VMs with aggregate incoming and outgoing network traffic from each VM. The data had snapshots of aggregate network traffic over 15 minute intervals spread over a duration of 5 days. To understand the variance in the data, we compared the discrete time differential (Δ) with the standard deviation (σ) of the data. Data that satisfied the following condition was considered constant: $-\sigma/2 \leq \Delta \leq +\sigma/2$. A histogram of percentage of data that was constant for all VMs is given in Figure 3.

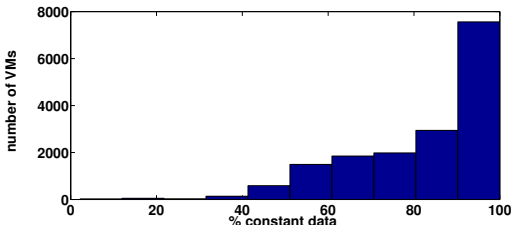


Fig. 3. Histogram of number of VMs with their percentage constant data

It can be observed that most of the VMs have a very large percentage of data that is constant and does not show much variance. A very large variance can be bad for VMFlow, as it will mean too frequent VM migrations, whereas a low variance means that migration frequency can be kept low.

Our first task was to calculate a traffic matrix out of this aggregate per VM data. Given the huge data size, we chose data for a single day. Various methods have been proposed in literature for calculating traffic matrices. We used the simple gravity model [20] to calculate the traffic matrices for all 17,000 VMs at each time stamp on the given day. Simple gravity model uses the following equation to calculate the network traffic from one VM to another: $D_{ij} = (D_i^{out} * D_j^{in})/(\Sigma_k D_k^{in})$. Here D_i^{out} is the total outgoing traffic at VM i, and D_j^{in} is the total incoming traffic at VM j. Although, existing literature [10] points out that traffic matrices generated by simple gravity model tend to be too dense and those generated by sparsity maximization algorithms tend to be too sparse than real data center traffic matrices, simple gravity model is still widely used in literature to generate traffic matrices for data centers [15].

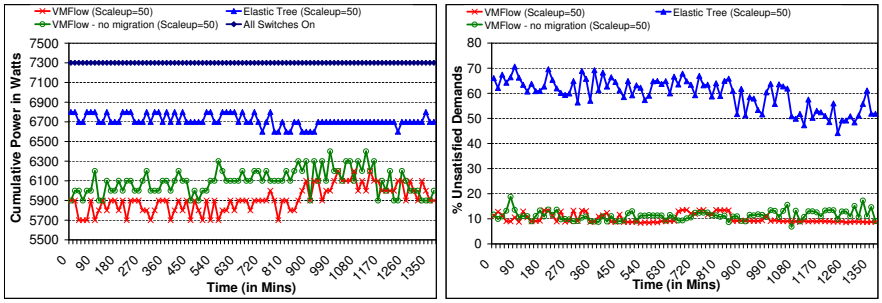
After generating the traffic matrices for each time stamp for the entire data center (17,000 VMs), we used the traffic matrices for the first 1000 VMs in order to reduce the simulation time required. Since gravity model tends to distribute all the traffic data observed over all the VMs in some proportion, it resulted in

a large number of very low network demands. In order to make these demands significant, we used a scale-up factor of 50 for all the data (i.e. each entry in all the input traffic matrices was multiplied by 50).

Simulation Results: The simulator compares VMFlow approach with the ElasticTree's greedy bin-packing approach proposed by Heller et. al [9] to save network power. Recall that the ElasticTree's bin-packing approach chooses the leftmost path in a given layer that satisfies the network demand out of all the possible paths in a deterministic left-to-right order. These paths are then used to calculate the total network power at that time instance. For placing VMs for the ElasticTree approach, we followed a strategy that placed the VMs on nodes that had the same id as their VM id.

We assume that all the network power comprise of power for turning on the switches, and the power required for turning on each network link (i.e., for ports on the end switches of the link) is zero. We calculated the total power consumed by the network and the proportion of network demands that were unsatisfied at a given time stamp for both VMFlow and ElasticTree approaches. In the first set of experiments, we simulated an oversubscribed network where the ToR-aggregate switch links and aggregate-core switch links had 1 Gbps capacity. This resulted in a 10:1 over subscription ratio in the ToR-aggregate switch link layer. The results for the oversubscribed network are shown in Figure 4(a) and Figure 4(b), respectively.

VMFlow outperforms the ElasticTree approach by a factor of around 15% and the baseline (i.e., all switches on) by around 20% in terms of network power at any given time instance. More importantly, one can see the effectiveness of VMFlow in the percentage of network demands that remain unsatisfied. VMFlow saves all the network power while satisfying 5-6 times more network demands as compared to ElasticTree approach.

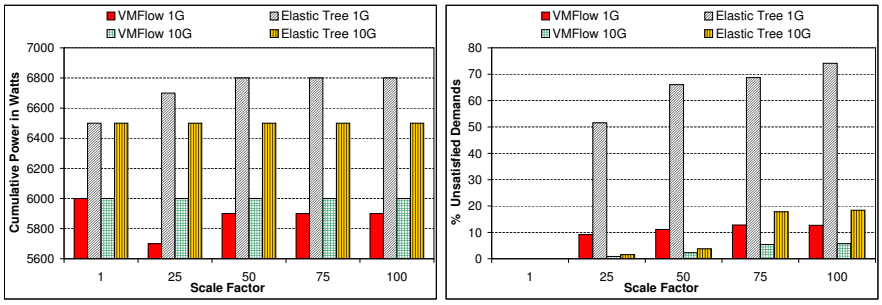


(a) Network power (b) Unsatisfied network demands

Fig. 4. Comparison at different time steps in the simulation

Since the input data had very little variation over time, we conducted an experiment to compare VMFlow with and without any migration after the first placement was done using the VMFlow algorithm. In order to simulate no migration, VMFlow simulator calculated an optimal placement of VMs based on the first input traffic matrix (representing the traffic for the first time stamp)

and only calculated the route at each subsequent time step. Figure 4(a) and Figure 4(b) show the performance of VMFlow approach without any migration. One can note that even with low variance input data, VMFlow with migration outperforms VMFlow without migration slightly by a margin of 5% in terms of power. Both the approaches perform roughly the same with respect to percentage of unsatisfied network demands. This indicates that migration frequency has to be properly tuned keeping in mind the variance in network traffic. Each VM migration has a cost associated with it that depends on the size of VM, its current load, the nature of its workload and the Service Level Agreement (SLA) associated with it. We are currently working on a placement framework that will take into account this cost and the potential benefit a migration can achieve in terms of network power and network demand satisfaction.



(a) Network power (b) Unsatisfied network demands

Fig. 5. Comparison at various scale factors in the simulation

We also compared the effect of using various scale-up factors on the input network traffic data. In this set of experiments we used a single input traffic matrix (representing the traffic at the first time stamp) and simulated both an oversubscribed network (ToR-aggregate and aggregate-core switch links have 1 Gbps capacity) and a network with no over subscription(ToR-aggregate and aggregate-core switch links have 10 Gbps capacity). The results are shown in Figure 5(a) and Figure 5(b). VMFlow outperforms the ElasticTree approach consistently, both in terms of network power reduction as well as percentage unsatisfied demands in the oversubscribed network (1G). However, in the no over subscription case (10G) the performance difference between VMFlow and ElasticTree is relatively lower. This is along expected lines, since VMFlow is expected to outperform mainly in cases where the top network layers are oversubscribed.

6 Handling Server Consolidation

In the earlier sections, we have assumed that at most one VM can be mapped to a host; i.e., the VM placement mapping Π is one-to-one. It is, however, easy to modify our algorithm to handle the case when multiple VM are mapped to a host, i.e., in case of server consolidation. In modern data centers, server consolidation is often employed to save both on the capital expenditure (e.g., by consolidating several VMs onto a small number of powerful hosts), and operational expenditure (e..g, any unused host can be turned off to save power) [19].

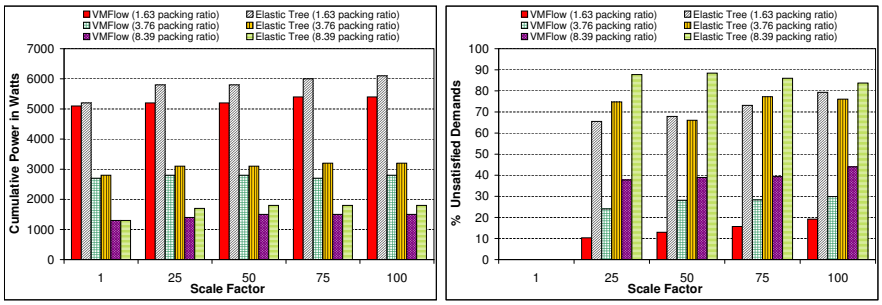
Although the basic idea of server consolidation is simple, deciding which VMs are co-located on a host is a complex exercise that depends on various factors such as fault and performance isolations, and application SLA. Thus, a group of VMs co-located during server consolidation may not be migrated to different hosts during network power optimization. Nevertheless, a group of VMs mapped to a single host can be migrated together to a different host, provided the new host has enough resources for that group of VMs. We now describe, how our greedy algorithm can be easily extended to handle server consolidation.

We assume that we have an initial server consolidation phase which maps zero, one or more VMs to each host. A set of VMs that is mapped to the same host during server consolidation is called a VMset. We make the following two assumptions: (1) while placing VMs on the host, the only resource constraint we need to satisfy is on the compute resource, and (2) a group of VMs consolidated on a host are not separated (i.e., migrated to different hosts) during the network power optimization phase. Now, our greedy algorithm requires two simple modifications to handle server consolidation. First, instead of mapping a VM to a host, the algorithm maps a VMset to a host, and the traffic demand between two VMsets is the sum of the demands of individual VMs comprising the two VMsets. Second, in each iteration for placing the source and destination VMset of a traffic demand, the set of possible hosts ($\Phi(s_j)$ and $\Phi(d_j)$) is defined as the set of free hosts that have equal or more compute resources than the respective VMsets. We omit details of these simple modifications from this paper.

We evaluated the performance of VMFlow in presence of server consolidation using the same simulation framework described in Section 5. Typically server consolidation is done based on the CPU and memory demands of the VMs and the individual host CPU and memory capacities. Since we did not have access to the CPU data for the 1000 VMs that were used in the earlier experiments, we generated synthetic CPU demands that were normally distributed with a given mean and standard deviation. We used the methodology given in [12], to generate mean and standard deviation of the CPU demands. We assumed the servers to be IBM HS22v blades [2], that go into a Bladecenter H chassis (each chassis can host 14 blades). We simulated an oversubscribed network where the server-TOR links, the ToR-aggregate switch links and aggregate-core switch links had 1 Gbps capacity. We assumed 3 different configurations of HS22v blades and created a VM packing scheme based on CPU demands using first fit decreasing (FFD) bin-packing algorithm. Following 3 configurations of HS22v blades (with increasing CPU capacity) were used in the simulation:

1. 1 Intel Xeon E5503 with 2 cores at 2.0 GHz - assuming a mean of 1.6 GHz and std. dev of 0.6, 1000 VMs with normally distributed CPU demands were packed into 612 servers (a packing ratio of 1.63 VMs per server). This required a network with 44 TORs, 14 Aggregate and 7 Core switches.
2. 1 Intel Xeon X5570 with 4 cores at 2.93 GHz - assuming a mean of 2.34 Ghz and std. dev of 0.93, 1000 VMs with normally distributed CPU demands were packed into 265 servers (a packing ratio of 3.76 VMs per server). This required a network with 19 TORs, 9 Aggregate and 5 Core switches.
3. 2 Intel Xeon X5570 with 4 cores at 2.93 GHz - assuming a mean of 2.34 Ghz and std. dev of 0.93, 1000 VMs with normally distributed CPU demands

were packed into 119 servers (a packing ratio of 8.39 VMs per server). This required a network with 9 TORs, 6 Aggregate and 3 Core switches.



(a) Network power (b) Unsatisfied network demands

Fig. 6. Comparison at various scale factors and configurations in the server consolidation scenario simulation

The input network traffic matrix was modified such that input and output network demands for VMs that were packed together to create a VMset, were added. VMsets resulting from the VM packing step were then used by the simulator to compare VMFlow and Elastic tree at various network traffic scale factors. Results for network power and percentage unsatisfied demands are shown in Figures 6(a) and 6(b),respectively. VMFlow continues to outperform Elastic tree even in presence of server consolidation. While the network power savings are consistently in the range of 13-20% at higher scale factors, the gains for percentage unsatisfied demands decrease (5x at 1.63 packing ratio, 3x at 3.76 packing ratio, and 2x at 8.39 packing ratio) as more VMs are packed into a single server and network traffic matrix becomes denser.

7 Future Work

This paper presented VMFlow, a framework for reducing power used by network elements in data centers. VMFlow uses the flexibility provided by VM placement and programmable flow-based routing, that are available in modern data centers, to reduce network power while satisfying a large fraction of the network traffic demands. We formulated the Network-Aware VM Placement as an optimization problem, proved that it is NP-Complete, and proposed a greedy heuristic for the problem. Our technique showed reasonable improvement (15-20%) in network power and significant improvement (5-6 times) in the fraction of satisfied demands as compared to previous network power optimization techniques.

Each VM migration in a data center has a cost associated with it. Similarly, the benefits that can be achieved by a VM migration in terms of network power reduction and meeting more network demands, depends on the variance in network traffic over time. In future, we plan to work on a placement framework that takes into account this cost and the potential benefit a migration can achieve in terms of network power and network demand satisfaction. We also plan to apply our technique to other network topologies and evaluate its benefits.

References

1. Cisco: Data center: Load balancing data center services, 2004.
2. IBM BladeCenter HS22V, http://www-03.ibm.com/systems/xbc/cog/bc_hs22v_7871/bc_hs22v_7871aag.ht%ml
3. The OpenFlow Switch, http://www.openflowswitch.org
4. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows - Theory, Algorithms and Applications. Prentice-Hall (1993)
5. Benson, T., Akella, A.A.A., Zhang, M.: Understanding Data Center Traffic Characteristics. In: ACM SIGCOMM WREN Workshop (2009)
6. Greenberg, A., Hamilton, J., Maltz, D., Patel, P.: The Cost of a Cloud: Research Problems in Data Center Networks. In: ACM SIGCOMM CCR (January 2009)
7. Greenberg, A., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D., Patel, P., Sengupta, S.: VL2: A Scalable and Flexible Data Center Network. In: ACM SIGCOMM (August 2009)
8. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In: ACM SIGCOMM (August 2009)
9. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: ElasticTree: Saving Energy in Data Center Networks. In: USENIX NSDI (April 2010)
10. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The Nature of Data Center Traffic: Measurements and Analysis. In: ACM SIGCOMM Internet Measurement Conference (IMC) (2009)
11. Kandula, S., Katabi, D., Sinha, S., Berger, A.: Dynamic load balancing without packet reordering. Computer Communication Review 37(2) (2007)
12. Korupolu, M.R., Singh, A., Bamba, B.: Coupled placement in modern data centers. In: IEEE IPDPS (2009)
13. Mahadevan, P., Banerjee, S., Sharma, P.: Energy proportionality of an enterprise network. In: 1st ACM SIGCOMM Workshop on Green Networking (2010)
14. Mann, V., Kumar, A., Dutta, P., Kalyanaraman, S.: VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers. Tech. rep. (2010)
15. Meng, X., Pappas, V., Zhang, L.: Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In: IEEE INFOCOM (2010)
16. Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In: ACM SIGCOMM (August 2009)
17. Shang, Y., Li, D., Xu, M.: Energy-aware routing in data center network. In: 1st ACM SIGCOMM Workshop on Green Networking (2010)
18. Vazirani, V.: Approximation Algorithms. Addison-Wesley (2001)
19. Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. In: ACM/IFIP/USENIX Middleware (2008)
20. Zhang, Y., Roughan, M., Duffield, N., Greeberg, A.: Fast accurate computation of large-scale IP traffic matrices from link loads. In: ACM SIGMETRICS (2003)