Communication-Awareness for Energy-Efficiency in Datacenters

Seyed Morteza Nabavinejad, Maziar Goudarzi
Energy Aware Systems Lab
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
mnabavi@ce.sharif.ir, goudarzi@sharif.ir

Abstract- With the Proliferation of Cloud Computing concept, the datacenters, as the basic infrastructure for Cloud Computing, have gained an ever growing attention during the last decade. Energy consumption in datacenters is one of the several features of them that have been the target of various researches. Two major consumers of energy in datacenters are the cooling system and IT equipment. Computing resources e.g. servers and communicating ones e.g. switches constitute a great portion of IT equipment. Among these two major players, the servers have been considered more than networking equipment. Making servers energy proportional or server consolidation are two essential approaches regarding reduction of servers' energy consumption. However, some researches indicate that 10% to 20% of energy consumption of IT equipment goes to network equipment and hence they must also be considered en route to better energy consumption in datacenters. The focus of this chapter is energy consumption of network equipment in datacenters and conducted researches in this area. First, a quick summary about network energy consumption in datacenters is presented. After that, related state of the art approaches and techniques are categorized, reviewed and discussed. Finally, the chapter is concluded with presentation of recent original work of authors and its details.

Keywords: Virtualized datacenter, VM consolidation, inter-VM communication, Energy Efficiency.

Table of abbreviations used in this chapter

Abbreviation	Original Term
IT	Information Technology

PUE	Power Usage Effectiveness
VM	Virtual Machine
PM	Physical Machine
NIC	Network-Interface Card
MPI	Message Passing Interface
VDC	Virtual DataCenter
IRA	Integrated Resource Allocator
TEA	Traffic aware Embedding Algorithm
FFLM	First-Fit virtual Link Mapping
VC	Virtual Cluster
CS	Communication Skeleton
TOP-VCM	Topology-aware partial VC mapping
VMM	Virtual Machine Manager
QoS	Quality of Service
CIVSched	communication-aware inter-VM scheduling
BCN	Bidimensional Compound Network
HCN	Hierarchical irregular Compound Network
NPE	Network Power Effectiveness
SaaS	Storage as a Service
SDN	software defined network
NS2	Network Simulator 2
SABVMP	Simulated Annealing Based VM Placement
CAVMP	Communication-Aware VM Placement
ILP	Integer Linear Programming
SCAVP	Structural Constraint-Aware Virtual Machine Placement

1. Introduction

Energy consumption is a critical concern for operators of datacenters and there are several factors that affect this energy consumption. Various parts such as IT equipment, cooling and power distribution consume power in datacenters. One measure of power efficiency in datacenters is the Power Usage Effectiveness (PUE) metric which is the ratio of total power consumption over IT equipment power consumption [1]. Many works have traditionally reduced the PUE by mostly considering the cooling power consumption [2-5].These efforts have led to very notable PUEs such as 1.12 for Google [1] or 1.07 for Facebook[6] by reducing total power consumption in the PUE definition. More recent works focus on improving power consumption of the IT equipment. Between severs and switches which are the major IT equipment that consume power, many works have been done to reduce the energy consumption of servers so as to make them energy proportional [7-10]. A study [11] on various kinds of web servers shows that the average utilization of servers varies between 11% and 50%. Virtualization has improved utilization of today datacenters by allowing consolidating several virtual machines (VM) on a single physical machine (PM) server. Since most today servers are not energy proportional [12], this consolidation helps operate the servers in their more energy-efficient operating regions. Virtualized datacenters enable a datacenter to service more requests per unit time and energy compared to non-virtualized ones which results in a greener datacenter with smaller carbon footprint. The amount of energy reduction by VM consolidation highly depends on the VM placement algorithm employed in the datacenter. Many different VM consolidation techniques have been presented [13-19] and each of them considers different parameters to reduce the number of ON PMs and to place VMs on them.

Aforementioned references indicate the importance of servers in datacenters, however, measurement reports reveal that about 10% to 20% of power consumption of IT equipment goes to network equipment [20]. A case study on Google datacenters also indicates that when the server utilization is 100%, the network infrastructure consumes around 20% of total IT equipment power, but when the server utilization drops down to 15%, the network share increases to 50% and even higher when energy-proportional servers are employed [21] (see Fig. 1). These observations demonstrate the significance of network power consumption and its need for more attention.

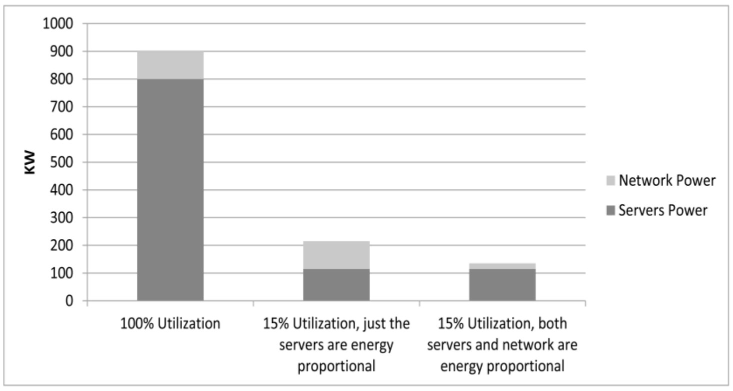


Fig. 1. Impact of energy proportionality of network on total power consumption[21]

This chapter surveys various aspects of network power consumption in datacenters and reviews state of the art approaches and mechanisms. In section 2, power consumption of components that constitute the datacenter networks are presented and discussed. Section 3 is dedicated to categorization of numerous techniques that are presented for energy reduction of network in datacenters. The remaining sections of chapter are dedicated to the recent approach of authors toward reducing energy consumption of datacenters with respect to communication.

2. Power Consuming Components in Networks

There are two parts in the networking infrastructure that consume power: switches and the Network-Interface Cards (NIC) of servers. According to [22], NICs consume only about 5% of total power in servers; thus, the network switches are the main target when addressing power consumption in networks. In Fig. 2 the share of each component in total power consumption of IT equipment is demonstrated. As can be seen, the share of switches is way more than NICs in servers.

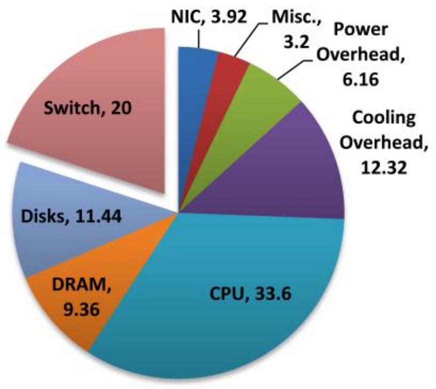


Fig. 2 Distribution of power consumption in IT equipment[20, 22]

For a better understanding of the power consumption mechanism in network switches note that each switch consists of a chassis containing a mainboard with one or more linecards connected to it, and each linecard has several ports. The power consumption of ports varies with their bit rate. Depending on the equipment, bit-rate of ports can be individually set and they can even be turned off if not connected or not needed. Similarly, each linecard can be separately turned on or off. In general, we can say that Energy consumption of a switch depends on the following factors: (a) type of switch, (b) number of ports, (c) port transmission rates, and (d) employed cabling solutions [18]. Thus total power consumption of a network switch can be formulated as equation 1[23]:

$$P_{switch} = P_{chassis} + n_{linecard} * P_{linecard} + \sum_{i=0}^{R} n_{ports} * P_r \qquad (1)$$

Where$P_{chassis}$ is the amount of power the switch consumes when turned ON, regardless of other parameters. The $n_{linecard}$ parameter indicates the number of linecards in chassis and $P_{linecard}$ is the power consumption of an active linecard. Finally, n_{ports} is the number of ports in switch and P_r corresponds to power consumed by an active port (transmitter) running at the rate ***r***.

3. Power reduction Techniques

As Eq. 1 shows, power consumption of network switches can be reduced in several ways. For classification purposes, we divide these approaches into three categories: use better equipment, better use of equipment, and reduce use of equipment. The first category deals with ways to reduce the constant values in Eq. 1 so that the network equipment is more energy-proportional; i.e., their power consumption is effectively reduced to near zero when not in use. The second category, *better use of equipment*, deals with static and dynamic techniques that help turn-off more linecards to save power. These two categories do not affect total communication volume transmitted over the network, but try to do it in a more efficient way by actually reducing *power* consumption of the equipment. The third category, which comprises more sophisticated and more recent techniques, represents various approaches that influence total volume of data to be transmitted by the network so that total operating time of equipment is reduced. In other words, these techniques reduce total *energy* consumption of the network equipment by reducing the *time* factor. Obviously, one needs to simultaneously apply these three categories of techniques to obtain the most effective power reduction outcomes, but for classification purposes, we present each of them separately in the following subsection.

3.1 Use Better Equipment

Eq. 1 practically covers most popular network switches in widespread use today. The constant values, such as the power consumption of the mainboard or the base power of each linecard, plays an important role in total power consumption; even when there is no activity in the network, these constant values are consumed to operate the switch. Reducing these constants is one important way to make switches energy-proportional. Moreover, the channels in many current switches are always ON regardless of transmission rate (even when there is no packet to transmit). More advanced switches, such as Dynamic InfiniBand switches, can be adapted to transmission rate and save power [24-26]. Use of other technologies, such as optical cables and networks, is another way for energy-proportionality [27-29]. For example, above Infiniband switches along with optical links make it possible for links to operate with fewer lanes and at a lower data rate to reduce the power consumption [21].

3.2 Better Use of Equipment

In this category, the components of a switch such as ports or linecards are considered and the main concern is to reduce the power consumption of them in order to reduce the energy consumption of switches, and consequently, reducing the network energy consumption. For example, in [30] the behavior of ports in switches is studied and a number of techniques is proposed as below: the simplest technique suggests disabling unused ports since some of them are never used or used just for short periods of time. This technique is inexpensive to implement statically, but if the communication requirements change dynamically, turning on-off should also be dynamic which is harder to implement. Port rate adaptation is another technique they suggest that can be useful for ports with low and almost constant utilization, but not for ones with fluctuating utilization. The third technique proposes aggregating the entire set of active ports on as few linecards as possible. It makes it possible to shut down the unused linecards to save the energy. Finally, the linecards can also similarly be consolidated across fewer switches. It provides the opportunity to reduce the number of active switches and save energy. These works [31-35] have used one or more of aforementioned techniques. However, moving ports from sparse switches to crowded ones needs significantly rewiring the network.

3.3 Reducing the Usage of Equipment

These techniques are more focused on concepts like communication time, communication volume or communication scheduling in datacenter network. Each of the aforementioned concepts can affect the energy consumption of datacenter network. For example, if one can decrease the amount of data that transfers among the network, the usage of switches will decrease and consequently they consume less energy. In the following subsections we will categorize these kinds of approaches and give more examples and explanation about each subcategory. In Fig. 3 you can see an overview of subcategories

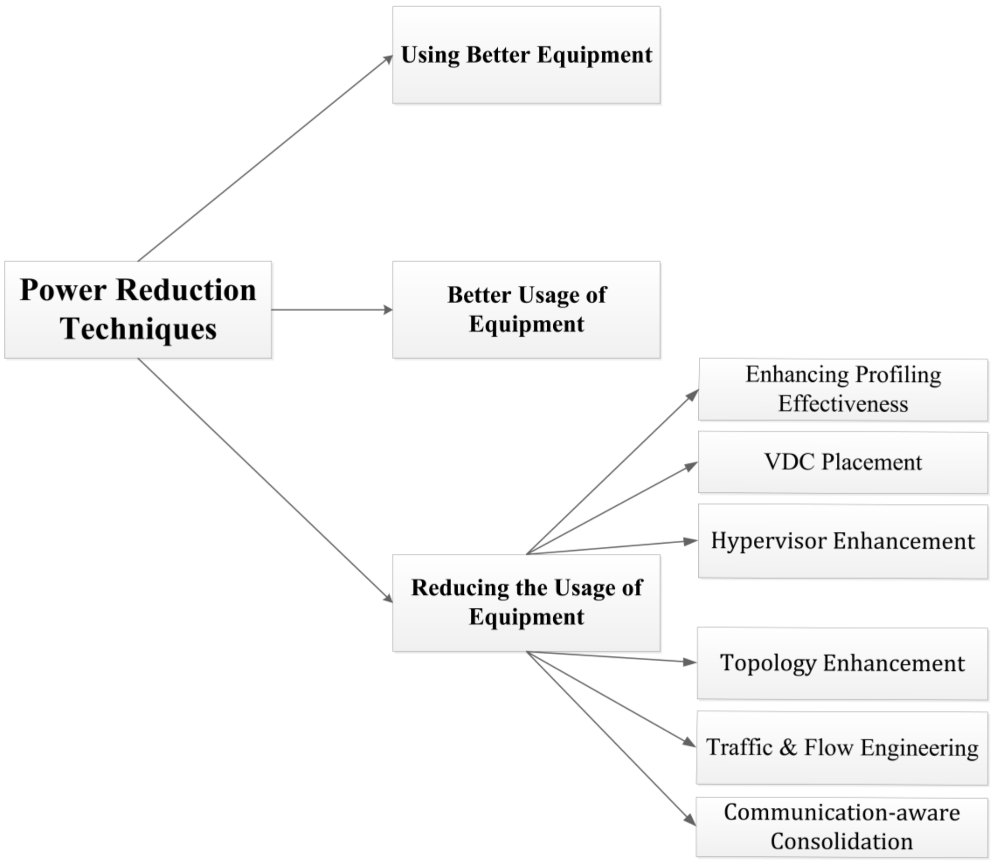

Power Reduction Techniques
Using Better Equipment
Better Usage of Equipment
Reducing the Usage of Equipment
Enhancing Profiling Effectiveness
VDC Placement
Hypervisor Enhancement
Topology Enhancement
Traffic & Flow Engineering
Communication-aware Consolidation

Fig. 3 Overview of Power Reduction Techniques

3.3.1 Enhancing Profiling Effectiveness

These kinds of works are the first ones in the area of communication-aware task placement algorithms and are the base for many algorithms that are proposed for datacenters and clouds. Ref. [36] is one of the primitive works that considers the communication when trying to balance the load in a cluster that runs parallel workloads. COM-aware is designed to handle a wide variety of applications by mean of an application behavioral model that includes load of CPU, disk and network generated by application.

For constructing this model, a profiling step is needed. The profiling can be either offline or online. This model then will be used to calculate the load impact of each application on cluster. The model considers several phases for a process of application and for each phases, the total execution time of that phase is broken down to communication time, disk I/O time and CPU time. The below equation shows this break down by this assumption that there are N phases:

$$for\ each\ i\ in\ N:\ \ T^i = T^i_{COM} + T^i_{Disk} + T^i_{CPU}$$

After that, the Communication, Disk and CPU requirements are estimated as below:

$$R_{COM} = \sum_{i=1}^{N} T^i_{COM}\ , \qquad R_{Disk} = \sum_{i=1}^{N} T^i_{Disk}\ , \qquad R_{CPU} = \sum_{i=1}^{N} T^i_{CPU}$$

The profiling step that mentioned earlier is responsible for obtaining the T^i_X times. This information then will be used to do communication-aware load balancing.

For evaluating the proposed schema, a 32 node cluster is simulated and a variety of workloads are used to evaluate its performance. The performance metrics that are used for evaluating are:
Turn-around time: it is a common metric for evaluating the job performance and measures as the elapsed time between job submission and job completion.
Slowdown: it can be calculated as T'_p/T_p where the T'_p is the turn-around time of job in a none dedicated system and T_p is turn-around time in the same system but this time the resources are dedicated to the job and there is no resource sharing.

Communication pattern which consists of three key attributes (volume, spatial and temporal) is a useful resource for perceiving the communication behavior of parallel applications and is extracted from communication trace. Before [37], the parallel applications must be run on whole cluster in order to produce communication trace but this process has two main weaknesses: it uses lots of resources and takes a long time. This work proposes a new method for producing communication trace by less resources and time, but by sacrificing the temporal attribute of communication pattern. It uses the MPI program and analyzes it to find the critical parts and then uses them to generate reliable communication trace. This work is also one of the first works that are done for communication awareness and later are applied to clouds and datacenters.

3.3.2 VDC Placement

For service providers who deploy their services on the cloud, sole VMs without considering their communication overhead cannot guaranty the desired performance. Instead they expect an entity that considers computing and communication elements simultaneously. For satisfying such demands, the Virtual DataCenter (VDC) concept emerged. A VDC is a set of VMs, switches and routers that are connected through virtual links and each virtual link is characterized by its bandwidth and delay. The VDCs can address the performance issue since they are kind of isolated regarding network and hence can better guaranty network resources availability.

Mapping the VDC resources onto physical infrastructure resources efficiently is a problem that several works are proposed to address it. Ref [38] proposes an approach for placing VDCs that aims to achieve two goals: allocating the computing and networking resources to each VDC and maximizing the number of placed VDCs and guarantying their performance. Since the available bandwidth for each VDC can affect the performance of running tasks on it, it is important to consider the network bandwidth in VDC placement. This work tries to reduce the inter-VDC bandwidth in order to increase the performance. It first proves that the problem of clustering the VMs of a VDC in order to reduce the inter-cluster bandwidth is NP hard. After that, the three-phase integrated resource allocator, IRA, is introduced that exploits the min-cut algorithms to form the VM clusters. Finally, another version of IRA, called B-IRA, is proposed that uses approximation algorithms and explores a smaller space for solution. Three phases of IRA can be seen in Fig. 4

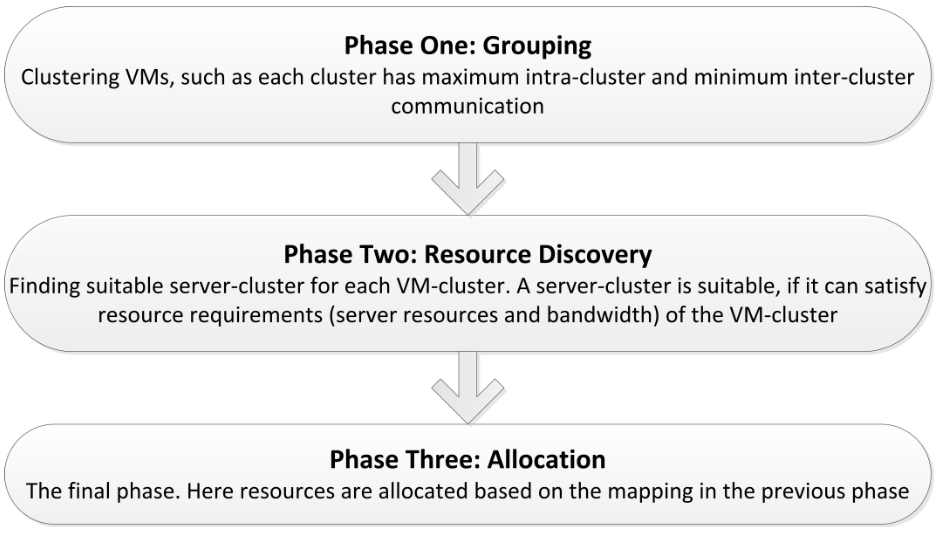

Phase One: Grouping
Clustering VMs, such as each cluster has maximum intra-cluster and minimum inter-cluster communication
Phase Two: Resource Discovery
Finding suitable server-cluster for each VM-cluster. A server-cluster is suitable, if it can satisfy resource requirements (server resources and bandwidth) of the VM-cluster
Phase Three: Allocation
The final phase. Here resources are allocated based on the mapping in the previous phase

Fig. 4 Three Phases of IRA

The *first phase* of IRA is only applied when the resource requirement of VDC is more than one server-cluster. All the servers in a server-cluster are attached to the same edge-switch. Therefore, when more than one server-cluster is required it means that there will be communication between clusters. Because of the high cost of this kind of communication compared with intra-cluster one, the IR tries to cluster the VMs of VDC in a way that inter-cluster is as low as possible. It is obvious that when the VDC can be fit in a single server-cluster, there is no need for VM-clustering. The *second phase* tries to find a valid mapping for each VDC on physical resources. The process of mapping for each VDC is successful if: first, there is at least one sever-cluster for each VM-cluster of VDC that can satisfies the resource demand of the VM-cluster and second, there is enough bandwidth among the selected server-clusters that can handle the communication between VM-clusters. Finally in the *third phase*, when the two previous phases are done successfully, the allocation routine places the VM-clusters onto corresponding server-clusters.

For evaluating the proposed technique, a set of simulations is conducted. The physical infrastructure considered is a three-tier hierarchal architecture with 192 server-clusters and 9216 servers. Regarding the number of switches, there are eight core switches, 32 aggregation switches and 192 edge-switches and each edge-switch represents a server-cluster. For evaluating the performance, three metrics are selected:

Percentage of accepted VDCs: how many of requested VDCs are accepted and placed on infrastructure

Cost: here the cost is the inter-cluster communication. When a VDC is split into several VM-clusters, the amount of communication between these parts is the cost of VDC placement.

Partition size: by changing the size of VDCs, the effect of VM-clustering is observed.

The main assumption in [39] is that network switches are the bottleneck in datacenter network and that the datacenter network topology is fat-tree. They have designed two algorithms called traffic aware embedding algorithm (TAE) and first fit virtual link mapping (FFLM) to map the requests for VDC on the physical infrastructure. They claim that proposed approach can increase the chance of placing VDCs and reduce the network cost.

Virtual Clusters (VC) are entities like VDCs that provide isolated resources for a job and are configured per job in runtime. A job that runs on VC has several sub-jobs and each of them runs on a VM in VC. The proposed approach in [40]first extracts the communication pattern among the VMs in VC and makes Communication Skeleton (CS). It then tries to place the VC on infrastructure by considering the resource demand of VMs as well as derived CS in order to increase resource (CPU and network) utilization. For reaching this goal, they introduce partial VC mapping which is in contrast to full VC mapping. They claim the former can lead to better resource utilization and less placement complexity and Topology-aware partial VC mapping (TOP-VCM) algorithm is designed to fulfill the concept. TOP-VCM tries to map VMs and Virtual Links between them on physical resources and links in order to increase resource utilization. It also leads to decrease in response time of tasks which are processed by job that is running on VC due to better communication performance.

The majority of works done just has considered the case where there is one infrastructure and distributed infrastructures are not considered. So Ref. [41] tries to solve the VDC embedding problem in the case of distributed infrastructures by considering energy efficiency and environmental impacts. The proposed solution comprises two phases: VDC partitioning and partition embedding. VDC partitioning phase splits the VDC to several partitions such that bandwidth between partitions is minimized. The partition embedding phase then lists the datacenters that are able to host the partition and chooses the best one based on constraints such as cost and location.

3.3.3 Hypervisor Enhancement

Virtualization technology has a promising effect on resource utilization in datacenters. Since the main concern at the time of designing current hypervisors such as Xen was the computing-intensive applications so they suffer from poor network I/O performance. For example, Single-root I/O virtualization, which is the current standard for network virtualization, is based on interrupt and since handling each interrupt is costly, the performance of network virtualization depends on the resource allocation policy in each hypervisor.

For conquering aforementioned problem, [42] proposes a packet aggregation mechanism that can handle the transfer process in a more efficient and rapid manner. However, the aggregation step itself introduces a new source of delay and must be addressed. Hence, the queuing theory has been used to model and dynamically tune the system in order to achieve the best tradeoff between delay and throughput.

Regarding the resource allocation policy problem the Credit-Based scheduler, the de facto resource scheduler in Xen hypervisor, cannot handle the I/O-intensive workloads properly since it is not aware of different behavior of various VMs and handle all of them in the same way. So, the I/O-intensive VMs don't earn enough credit for handling network interrupts and hence retrieve the data in a slow manner that leads to high latency and response time.

To tackle the mentioned problem and eliminate the bottleneck which is caused by scheduler, [43] introduces a *workload-aware network virtualization model* that monitors the behavior of VMs and divides them based on their behavior in two categories: I/O-intensive and CPU-intensive and then handles them by *Shared Scheduling* and *Agile Credit Allocation*. When an I/O-intensive VM faces burst traffic, the shared scheduling gives it more credit so it could be able to handle the traffic and agile credit allocation is responsible for adjusting the total credit based on number of I/O-intensive VMs in order to reduce the wait time for each I/O-intensive VM.

Ref [44] first introduces a semantic gap that exists between VMM and VMs. The gap is that VMM is unaware of processes inside VMs so it cannot schedule the VMs efficiently. As an example, when a VM sends a request to another co-located VM, the co-located VM must earn vCPU so it can process the request and provide the response ASAP but the current scheduling in VMM doesn't consider this matter. As a result, the response latency increases and it may lead to QoS violation. Moreover, it gets worsen when the co-located VMs are CPU or IO intensive because the competition for CPU increases dramatically. Fig. 5 illustrates the mechanism of inter-VM communication in current Hypervisors.

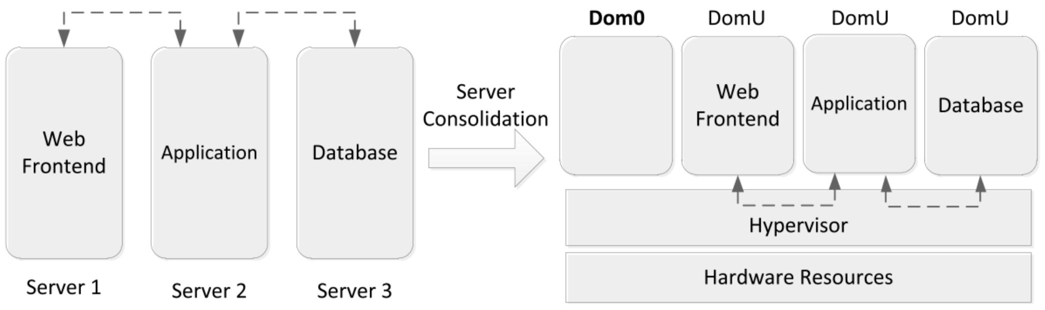


Fig. 5 Inter-VM communication in VMMs

To resolve the problem, they propose the CIVSched algorithm which is aware of communication among co-located VMs. CIVSched monitors the packets that are send through the network and identifies the target VM and schedules the VM in a way to reduce response latency. The CIVSched prototype is implemented on Xen Hypervisor.

For each DomU guest (VM) there is a virtual front-end driver that VM sends the requests for I/O operations to it. Then these requests are sent to back-end driver which is in Dom0 guest. And finally, the back-end driver sends the captured requests to real device driver and returns the responses to the front-end driver. As an example, Fig. 6 depicts the way of a packet sent by any VM through the Xen hypervisor in standard manner.

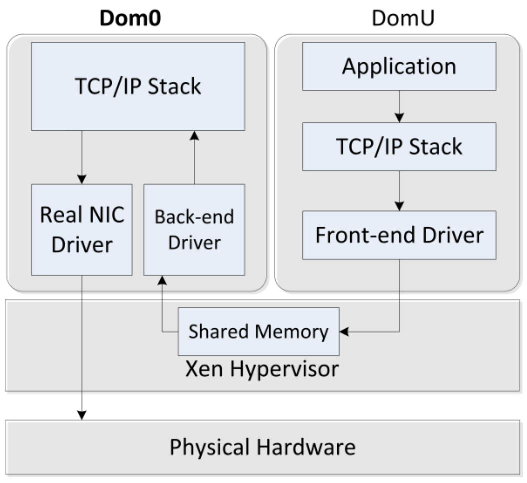


Fig. 6 Xen I/O Overview

The CIVSched must abide by two design principles: low latency for inter-VM and low latency for the inner-VM process. These two design principles help CIVSched to decrease the inter-VM latency. For realizing the two above mentioned requirements, the CIVSched adds five modules to the Xen I/O mechanism. The *AutoCover* (Automatic Discovery) module finds the co-located VMs and stores their MAC address and IDs in a mapping table. The *CivMonitor* checks all the packets transmitted by VMs and when finds an inter-VM packet, informs the *CivScheduler* about it. Then, *CivScheduler* gives more credit to target VM so it can handle the packet as fast as possible. Until now, the first design principle (low latency for inter-VM) is satisfied but the other one still needs attention. Regarding the second principl, *CivMonitor* identifies the process of target VM that will receive the packet via TCP/UDP port number within the packet and passes the information to the target VM. Finally, *PidScheduler* and *PidTrans* modules inside the guest VM schedule the target process with respect to decreasing latency. Fig. 7 illustrates the added modules by CIVSched to the Xen hypervisor.

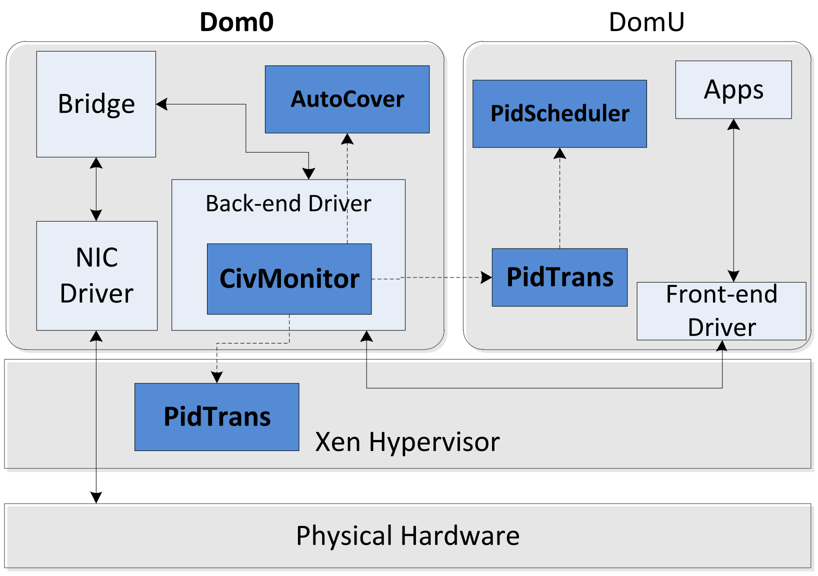


Fig. 7 Overview of CIVSched

For evaluating the CIVSched, it has been implemented on Xen hypervisor version 4.1.2 and is compared with XenandCo [45] scheduler (another proposed scheduler for Xen) and Credit scheduler which is the base scheduler in Xen. For comparing the *Network latency*, experiments consist of a ping-pong test, a simulation test and a real world we application scenario but with synthetic benchmarks. *Fairness Guarantees* is also evaluated because the fairness of scheduler directly affects the fairness of CPU resources allocated to each VM. The UnixBench suite 4.1.0 is adopted for evaluating the *performance overhead* of CIVSched on host's performance. *Performance overhead* is measured at two levels: when there are just two VMs on the host (light consolidation) and when there are seven VMs running simultaneously on the host (heavy consolidation)

3.3.4 Topology Enhancement

There are two major categories for datacenters network architecture: **switch-centric and server-centric**. In *server-centric architectures* the servers not only act as computing resources, but also have the responsibility for packet forwarding. In *switch-centric architectures* however, the packet forwarding and communication is guaranteed by switches and servers do not have any role in packet delivery.

Among the switch-centric architectures, 3-level fat tree topology is the most common one. In this topology, there are several paths between two nodes so in the case of link failure there are alternative routing ways. One dominant problem by fat tree is that the number of nodes that can be connected is restricted. If one needs to connect more nodes, there are two way for that: replacing the switches with bigger ones that need to reconfigure the whole connections which is costly or increasing the number of levels that leads to increase in network diameter. For solving the mentioned problems, [46] proposes a new approach for constructing large datacenter networks with fixed size switches. They first construct hyper graphs using the hyper graph theory and then convert them to indirect hyper graphs. This approach makes it possible to connect more nodes together with fixed size switches compared with fat tree topology.

In server-centric architectures, the delay of a server-to-server direct hop and a server-to-server-via-a-switch hop is considered equal; however with the fast growing capabilities of servers for packet forwarding, this assumption might be invalid in the future. Moreover, until today, it is believed that BCN [47] can connect the most number of dual port servers for a fixed number and size of switches, while the authors claim that DPillar [48] architecture can connect more servers than BCN under the same configuration.

Regarding two aforementioned points, [49] proposes three new server-centric architectures for dual port servers. The architectures try to answer this question: "*what is the maximum number of dual-port servers that any architecture can accommodate at most, given network diameter* ***d****, and switch port number* ***n***" [49] and approximate the upper bound of possible dual port servers.

The first architecture, based on generalized hypercube [50], is called SWCube. Under different conditions, this architecture either can connect more servers than DPrill or less. The two other architectures, SWKautz and SWdBrujin, which are based on Kautz graph [51]and de Brujin graph [52]respectively, always proper better answers compared with DPrill.

With the help of dual port servers, [47] proposes two new server-centric network structures called Hierarchical irregular Compound Network (HCN) and Bidimensional Compound Network (BCN) for datacenters which are of server degree 2. These structures have low diameter and high bisection and can be expanded easily. Fig. 8 and Fig. 9 depict these two new structures and more details about them will be presented in the following.

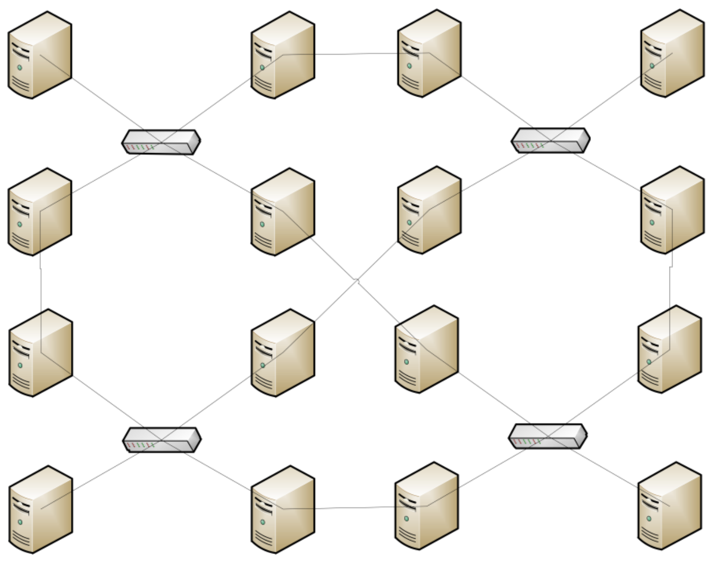


Fig. 8 Structure of HCN (4,1) [47]

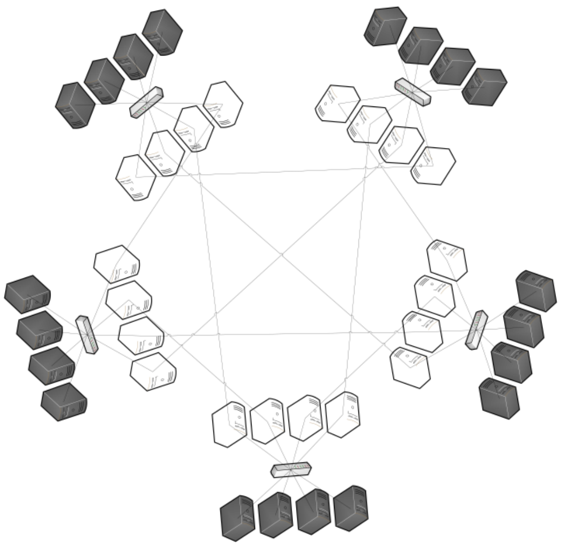


Fig. 9 BCN Network Architecture in the first dimension [47]

A level-h HCN is denoted as *HCN(n,h)*. Here *n* stands for number of dual-port servers in the smallest module and number of ports in miniswitch that connects the servers together in that module. A HCN(n,h) consists of n of HCN(n,h-1) modules that are connected via a complete graph. Here the second port of servers is used to connect the smallest modules together (the first port is connected to miniswitch). The smallest module is denoted as HCN(n,0). Generally, we can say that: $for\ i\ \geq 0\ ,$ $HCN(n,i)\ is\ formed\ by\ n\ HCN(n\ ,i-1)$

α stands for number of master servers and β determines the number of slave servers in BCN(α,β,h) where h shows the level of BCN in the first dimension. The sum of α and β denotes the number of dual-port servers and number of ports in miniswitch again in smallest module or building block like HCN. A general BCN is denoted by BCN (α,β,h,γ) and γ stands for level of BCN in the second dimension. It worth to mention that there is really no master/slave relation between servers and these names are just for simplifying the presentation of structure. The master servers, black ones in Fig. 9,are used to expand the BCN from outer section or first dimension and slave servers, white servers in Fig. 9, make it possible to expand the BCN from inner section or second dimension.

Since good support for one-to-one traffic routing leads to good all-to-one and one-to-all support, two algorithms are presented for routing one-to-one traffic in BCN. First, the single path routing without failure is studied and then it has extended to parallel multipath routing. Finally, the failures are also considered and addressed by using multipath between servers. *Network order* (number of servers in network), *bisection width* and *path diversity* are the parameters that BCN and HCN are compared with FiConn based on them. The results show that BCN can surpass the FiConn and yield better performance regarding aforementioned parameters.

Network Power Effectiveness (NPE) indicates the tradeoff between power consumption and throughput of a network and also shows the bit-per-second per watt (bps per watt) for a network. This parameter is important in today's datacenters since the main concern of recent proposed architectures for datacenter network is throughput and power consumption is rarely considered, however the network consumes a big portion of overall power in a typical datacenter. Ref [53] does a comprehensive study on dominant advanced network architectures in datacenters like Fat-tree and BCube regarding their NPE. It also studies the effect of different parameters such as traffic load, power-aware routing, traffic pattern and topology size on the NPE and compares the switch-centric architectures (which that try to propose a new architecture based on switches or improve tree architecture) like VL2 and Fat-tree against server-centric architectures (that use servers with multiple NICs and each server is involve in the packet forwarding flow) like DCell [54]and FiConn [55].

3.3.5 Traffic & Flow Engineering

Various methods are proposed that aim to reduce just energy consumption of network. Ref [56] aims to reduce power consumption of core network in distributed cloud for different kind of applications. For content delivery services, they design a mixed integer linear programming model. Based on this model, they conclude that replicating popular content on different clouds can reduce power consumption significantly compared to centralized content delivery model because of power reduction of network switches. For storage as a service (SaaS), they suggest that migrating content based on its access frequency can be beneficial. And finally, for VM placement, they propose to break the large VMs into several smaller VMs and place them on different clouds to facilitate the access of users from different locations and reduce the power consumption of network.

Despite the common approaches that try to optimize the network performance and energy via sole traffic-engineering, [57] tries to consider both traffic-engineering as well as network features such as topology and end-to-end connectivity. The proposed approach deeply explores both application characteristics and network features and then based on these observations does the VM assignment. This assignment will lead to favorable conditions in datacenter network that will be used for next step which is traffic-engineering.

One common way for reducing the power consumption of switches is to reduce the number of active switches by flow aggregation and then put the rest of them in “sleep on idle” state. Although the flow aggregation works fine for application-limited flows, where the amount of data that need to be transmitted is low and one network link can be shared between several flows to transmit their data, it cannot handle the network-limited flows like MapReduce applications. In network-limited flows, the application produces lots of data in a short period of time and then the capacity of network as well as the number of flows that compete on the bottleneck link determines the throughput for each flow. Consequently, however the number of active switches is reduced by aggregation technique, their uptime increases due to limited capacity for each flow.

Ref [58] addresses above issue by modeling the problem and considering both reducing the number of active switches as well as deadline and size of each flow. They design “willow” which uses software defined network (SDN) technique and schedules the flows in terms of energy consumption. For achieving online scheduling, a greedy approximation algorithm is presented. This algorithm tries to use all the idle ports in a switch and then reduce the number of active switches until the running duration of network is not increased.

Example in Fig. 10 shows that current techniques for scheduling flows, that just consider either minimizing the number of active switches or maximizing the network throughput, are not suitable for network-limited flows regarding energy efficiency. In this example, there are two flows: f1 from server 1 to server 3 and f2 from server 2 to server 4. Here, the following assumptions are considered: size of each flow = $\mathbf{Z}$, the capacity of each link = $\mathbf{B}$, power consumption of each switch = $\mathbf{P}$ and deadline of each flow = $\frac{(\mathbf{2}*\boldsymbol{z})}{\boldsymbol{B}}$.

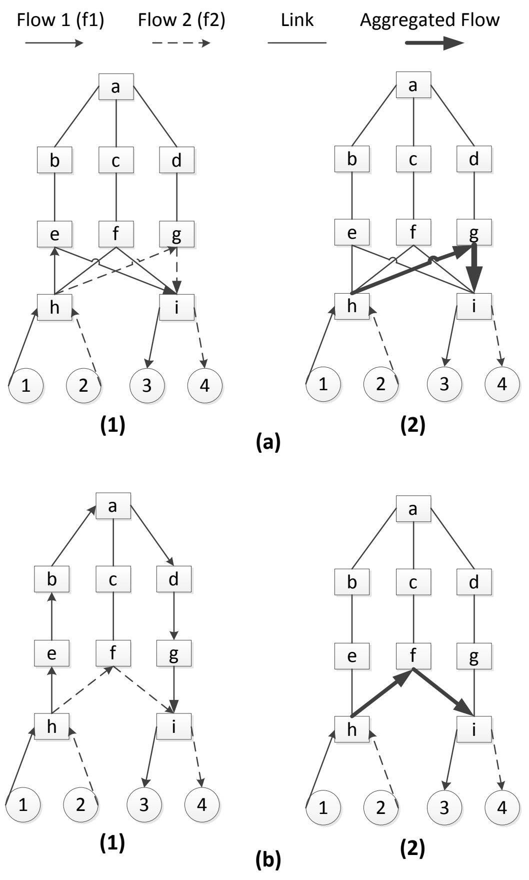


Fig. 10 Illustrative example for showing performance of different flow scheduling algorithms

The scheduling algorithm in (1)(a) aims to maximize the network throughput so routes the flows as is depicted. Here, four switches are active for **Z/B** time, so the total network energy consumption is **(4*Z*P)/B**. meanwhile, the scheduling algorithm in right side tries to minimize the number of active switches. the result is three active switches with **(2*Z)/B** uptime and hence the network energy consumption is **(6*Z*P)/B**. as can be seen, while the second algorithm use one less switch compared with first one, it uses more energy than that of first one. However, in this scenario both scheduling schemas meet the deadline.

In scenario B, it is assumed that links h-g and e-i are faulty and out of service. Again, the left side figure illustrates the path that flow scheduling maximizing throughput selects and the energy consumption of network for such routing would be **(8*Z*P)/B**. The energy consumption of network for path that second algorithm i.e. minimizing the active switches, chooses is **(6*Z*P)/B**. we can see that despite the first scenario where the maximizing throughput gave better energy consumption, in this scenario minimizing the number of active switches is more successful regarding network energy consumption. This example clearly demonstrates that considering either network throughput or number of active switches is not sufficient and both of them must be considered simultaneously in order to achieve a successful scheduling under various conditions and the Willow algorithm is proposed to do it.

There are three basic ideas behind willow design and it has been developed based on them.

SDN based Flow Scheduling: willow uses SDN framework to collect information about flows such as their size and deadline or computing the routing path for each of them.

Routing Path Selection: willow is not topology-dependent and can work with different topologies such as fat-tree or Bcube.

Differentiation between Elephant Flows and Mice Flows: since the elephant flows dominate the traffic in datacenter networks [59], willow focuses on them and first schedules them, then reuses the computed paths for elephant flows in a random way for mice flows.

For evaluating Willow regarding network energy consumption reduction, simulation and testbed experiments are conducted. In the simulation setup, for network topology both fat-tree and blocking fat-tree are used. The MapReduce computation trace from 50 mappers and 20 reducers is considered for workload trace. The evaluation metric for comparing willow against rival algorithms i.e. simulated annealing and particle swarm optimization is network energy ration. For evaluating willow in a real setup, a testbed with 16 servers in a fat-tree network is considered.

3.3.6 Communication-aware Consolidation

As the amount of PUE goes down by using advanced approaches for reducing the non-IT equipment power consumption, the power consumption of IT equipment (servers and switches) becomes dominant in datacenters. A wide variety of techniques and algorithms are proposed until now for reducing the quota of servers in power consumption [13-15, 19, 60-63] but there are lots of opportunities for decreasing the switches' power consumption [58].

Joint approaches like [64, 65] consider both inter VMs communication as well as VM consolidation. However, these works suffer from too simplistic and inaccurate models of communication time and its effect on application runtime and energy consumption. They have tried to group the VMs that communicate with each other and then place them on one or more servers to reduce network traffic, but did not take into account VMs communication inside a group, and more importantly, did not consider the communication structure among the servers.

To address the aforementioned issues regarding joint approaches, we have developed a new approach and will describe it here.

One of our contributions in this work is that we consider all inter-VM communication even inside a group and place VMs based on them so the network traffic is less compared to other approaches. Another advantage of our work is that we compute the communication time for each VM using a detailed network simulator, NS2, which improves the accuracy of results. A number of other communication-aware VM consolidation techniques exist, but they optimize only based on either the communication volume [66] or network congestion [18], both of which are relevant but indirect and inaccurate indicators of the actual communication time on the network. As an example, consider a unit volume of data communicated between two servers; it takes different amounts of time if the two servers are in the same rack compared to when they are in different racks, and even more if in different clusters. Thus, the distribution of communications endpoints as well as the structure of racks and clusters, in addition to communication volume, play important roles to determine communication time, and hence, communication energy consumption. We improve state of the art by showing this shortcoming in existing approaches and their resulting sub-optimal placement, and by providing an approach to accurately consider network communication time.

An important contribution of our approach over prior works is that we consider the structure of the datacenter, in terms of placement of servers in racks as well as racks in clusters, when placing the VMs. Prior art have only concentrated on consolidating VMs on servers without paying attention to the reality that most, if not all, cloud computing implementations take advantage of a datacenter in the backend, and hence, the actual place and connection structure of those servers is an important factor in total communication time, and the final energy consumption. We show and quantify the significance of this point by our experiments in Section 6.

Another contribution of ours is revealing the inaccuracies in prior work by considering actual network structure and elements when evaluating the energy consumption and VM consolidation outcome. To evaluate our as well as rival techniques, we use NS2 simulator for real-world network simulation, which is far more accurate than approaches that abstract out most network properties and only count the number of switches between servers, such as [67] where a switch can become host spot, and consequently, the actual communication time is effectively more than they estimate. We prove superiority of our proposal by comprehensive experiments in this elaborate simulation environment. Experimental results show that our approach reduces the amount of communication up to 71.9% in synthetic benchmarks, and 77% in a real world benchmarks compared to an improved version of our closest rival. Consequently, the overall energy consumption is improved by up to 17.8% in synthetic benchmarks, and 79% in the real world benchmark, by our technique compared to that improved rivals.

4. Our Approach

In this section we first show the flow of our work. Since pure VM placement problem is similar to the bin-packing problem in computer science, we use one of corresponding algorithms, the first-fit algorithm, as a baseline algorithm to compare to.

4.1 Our Optimization and Evaluation Flow

Fig. 11 gives an overview of our optimization and evaluation flow. First we use a VM placement algorithm to place the VMs on servers (the *Placement Phase* in the figure). We present two algorithms in this paper (see Section 5) corresponding to this step. Then, the results of the placement are evaluated to report here and compare to related work (the *Evaluation Phase* in the figure). There are many real-world cases even at large scales where the VMs are basically static and the goal is to optimally place them for the lowest energy. The flow in Fig. 11 is sufficient to address the needs of such cases, but for other cases where the nature of VMs is semi-static or dynamic, we re- run our VM-placement algorithm (see Fig. 12) at certain intervals if new VMs arrive or existing VMs finish execution during that interval. We assume 30-minutes for this re-execution interval in this work. While the choice of this time period for re-placement interval may not be optimal, it is short enough to address most dynamic behaviors, while it is long enough to cover rather long boot up/shut down latency of several new/finished VMs, and furthermore, it is long enough to reduce the energy and time overhead of re-running the placement algorithm—see Section 6.3. Deciding the optimal interval for VM re-placement is an interesting objective and is part of our future work.

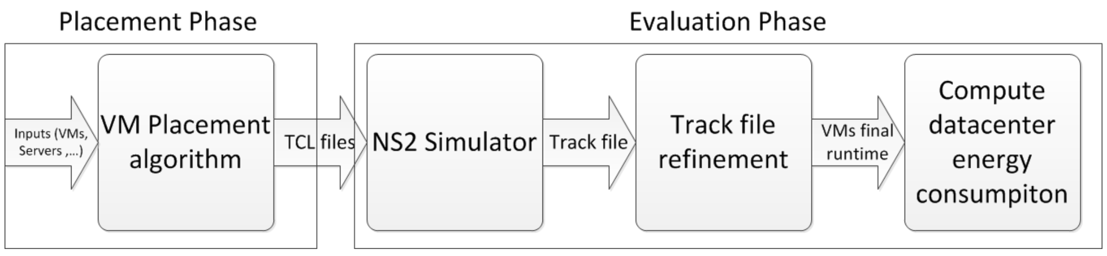
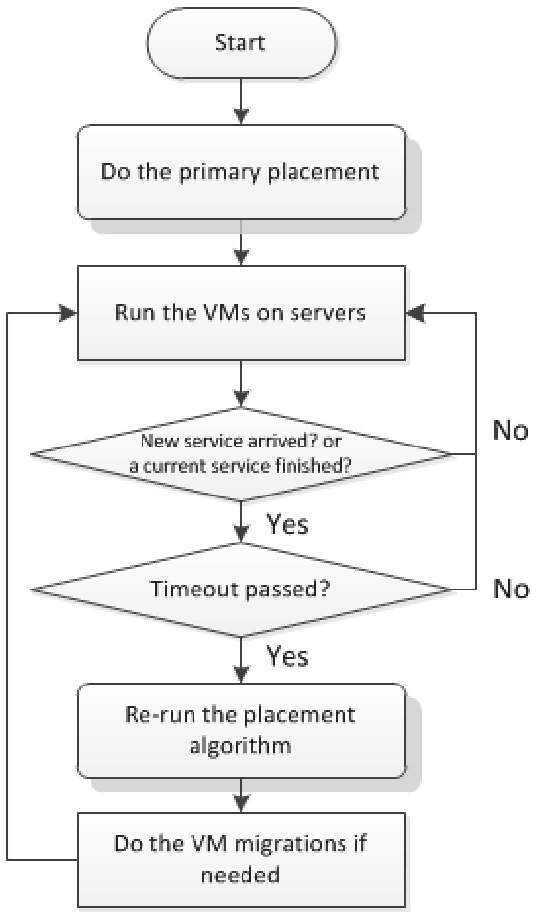


Fig. 11. Our optimization and evaluation overall flow



Start
Do the primary placement
Run the VMs on servers
New service arrived? or a current service finished?
No
Yes
Timeout passed?
No
Yes
Re-run the placement algorithm
Do the VM migrations if needed

Fig. 12. Placement and re-placement phases of our technique for dynamic environments

After the placement phase is accomplished, it is time to evaluate the proposed VM placement so we go on to the evaluation phase in Fig. 11. Since the placement algorithm is re-run at fixed intervals, we evaluate its advantages in a single interval in our experiments in Section 6. The overhead of live migration of a number of VMs, if the re-placement decides so at the beginning of a new interval, is analyzed in Section 6.3. Based on the VM placement determined by the algorithm, we generate TCL script files that are the input of the NS2 simulator for network traffic assessment. TCL files consist of information such as network topology and amount of data that transfers between the VMs. After NS2 has finished simulating the network operation, it produces track files. In a track file, the sender, the receiver, the send time and the receive time of all packets are specified and we use this information to calculate the communication overhead of each VM and add it to the VM total execution time. Finally we compute energy consumption of datacenter IT equipment based on the VMs updated execution time.

5. Problem Formulation and Algorithms

In this section we give notations and formulate the optimization problem. After that, the proposed placement algorithms are described.

5.1 Energy Model

Servers consume a lot of energy in datacenters. Server power consumption is proportional to CPU utilization. Some researches indicate that an idle server consumes around two-thirds of its peak load consumption. The remaining one-third changes almost linearly with the increase in the level of CPU load. According to this, we use the following server power model [68], [18], [69]:

$$P_{server} = P_{idle} + (P_{max} - P_{idle}) * U$$ (~~1~~2)

where P_{max}, P_{idle}, and U are server power consumption at maximum utilization, server power consumption when idle, and server CPU utilization respectively. It is noteworthy that since the assumption in our benchmarks is that when a CPU is assigned to a VM, that single CPU is fully utilized by that VM, the U parameter for each server is defined here as the number of active cores of the server divided by the total number of cores deployed in the server.

Switches energy consumption is also significant specially when there is a lot of communication between servers. Energy consumption of some powerful switches is more than regular servers, and hence, it is important to consider energy consumption of them in evaluation. Here we use the Eq. (1)

for switches power consumption.

5.2 Notations

We assume that the resource demand for each VM is determined and fixed, and does not change during the run time of the VM. For example a VM needs two cores and 3GB RAM and these values are constant during the runtime of VM and do not change. We also assume that the communication matrix of VMs is given. Each element of this matrix shows the amount of communication between two VM. This information can be obtained by profiling the VMs during a test period prior to optimization. The above specifications of all VMs are given parameters to the algorithm. We define the notation listed in Table 1.

Table 1. Notation used in our problem formulation

Parameter	Meaning/ description
M	Number of Servers
N	Number of VMs
S	Number of Switches
$VMCore_i$	Required Number of Cores for VM i
$VMRAM_I$	Required Amount of Memory for VM i
$VMComTime_i$	Communication Time of VM i
$ServerCore_i$	Available Number of Cores in Sever i
$ServerRam_i$	Available Amount of RAM in Server i
$VMtoVMcom_{i,j}$	Amount of Communication between VM i and VM j in KB
$StoScom_{i,j}$	Amount of Communication between Server i and Server j in KB
$VMonServer_{i,j}$	Binary variable indicating whether VM i is placed on Server j or not (1 = is placed , 0 = is not placed)
$isServerUp_i$	Binary variable indicating whether Server i is up or not (1 = server is up, 0 = server is not up)
$ServerEC_i$	Energy Consumption of Server i
$SwitchEC_i$	Energy Consumption of Switch i

5.3 Problem Formulation

Using the notations given in Table 1, the optimization problem is defined as follows:

“For a given set of datacenter structural parameters (i.e., *m*, *S*, *ServerCore* vector, *ServerRam* vector, and the structure of the datacenter network) and a given set of VM parameters (i.e., *n*, *VMCore* vector, *VMRAM* vector, *VMtoVMcomm* matrix), minimize total datacenter energy consumption, covering servers and switches, by assigning each VM to one and only one server while considering the communication delay among the VMs as well as capacity constraints on server resources. "

Formal equations are listed below:

Equation (3) designates the objective function which is total energy consumption of all servers and switches. Note that in the placement phase (see Fig. 11) we do not compute this equation; our algorithms take the servers and racks structure into account when placing the VMs to minimize the energy, but the amount of this energy is computed in the evaluation phase, where we run NS2 simulations for obtaining the communication time of each VM to finally compute the energy consumption.

Our goal is to reduce the amount of energy that servers and switches consume. But we should pay attention to some constraints while we are trying to achieve this goal. The first and second constraints that are mentioned in (4) and (5) deal with constraints on resources of servers. Equation (4) is about number of cores that is available in a server and states that required number of cores of VMs that are placed on each server must be less than or equal to available cores of server. Equation (5) is similar to (4) but instead of cores count, it focuses on memory volume. Each VM must be placed on one and only one server and (6) shows this constraint.

After placing VMs on servers, the communication volume between servers is calculated by (7) and then by using NS2 simulator, the communication time of each VM is calculated in (8). Note that the NS2 simulation written as $NS2(StoScomm_{k,j})$ in (8) is not a mathematical model since it involves running the simulator and processing its outputs. It merely clarifies the process taken to obtain communication time after placement. We also count the number of up servers by (9). Reducing energy consumption of servers and switches has a direct relationship with results of (7) and (9). If a placement algorithm can reduce the number of up servers and also communication between servers, then the energy consumption of datacenter will be reduced consequently.

Finally, the energy consumption of each server and switch is calculated by (10) and (11). As explained above, these two equations are not part of the *placement phase* but belong to the *evaluation phase*. In these equations we use the power models that are introduced in (1) and (2), as well as communication time vector *VMComTime* which is calculated by NS2 simulator *after* the placement phase. VMComTime indicates the time of the last packet that is sent or received by a VM. At the end of NS2 simulation phase, it reports the sender, receiver, and the time for each packet. Using this information, we calculate the VMComTime for each VM as

VMComTime = max (send or receive time of all packets that this VM has sent or received).

CalculateServersEnergy function in (10) works as follows: for each server in each interval, the function checks to see how many of the VMs placed on the server are running in the current interval based on their total running time (communication time + execution time). After obtaining the number of running VMs, the function calculates the utilization of server (U parameter) based on the number of cores that are active by these VMs. Finally it calculates the energy consumption of the server in the interval by equation (10). It is noteworthy that the server is turned off when all the VMs on it are finished. The sum of the energy consumption in each interval gives the total energy consumption of server.

In CalculateSwitchesEnergy function in equation (11), in order to determine which top of rack switches are ON in an interval, the function checks the servers that are connected to each of the switches (by using the switch id of servers). Note that if none of the servers are ON (meaning that there is no running VM on the servers) then the switch is turned off. However, if one or more servers are turned on, the switch is also turned on and its energy consumption contributes to total energy consumption of the switches. For layer 2 as well as core switches, the process is the same as above except that instead of servers, now the lower layer switches are taken into account to determine on and off switches. Note also that the datacenter network topology is considered in this step. It is the topology that determines which servers are connected to which top of rack switches as well as the connection between different layers of switches. In fact we use (10) and (11) to evaluate our approach and compare it with previous ones.

$$Min \quad \sum_{i=1}^{m} ServerEC_i + \sum_{i=1}^{s} SwitchEC_i \tag{3}$$

$$\sum_{j=1}^{n} VMCore_j * VMonServer_{j,i} \leq ServerCore_i \ , \quad i = 1 \dots m \tag{4}$$

$$\sum_{j=1}^{n} VMRam_j * VMonServer_{j,i} \leq ServerRam_i \ , \quad i = 1 \dots m \tag{5}$$

$$\sum_{j=1}^{m} VMonServer_{i,j} = 1 \tag{6}$$

$$StoScom_{i,j} = \sum_{k=1}^{n} \sum_{l=1}^{n} VMonServer_{k,i} * VMonServer_{l,j} * VMtoVMcom_{k,l} \quad , \quad i,j = 1 \dots m \tag{7}$$

$$VMComTime_i = NS2\left(StoScom_{k,j}\right), \qquad i = 1 \dots n\,, \quad k,j = 1 \dots m \tag{8}$$

$$isServerUp_i = \begin{cases} 1\ , & \sum_{j=1}^{n} VMonServer_{j,i} > 0 \\ 0\ , & \sum_{j=1}^{n} VMonServer_{j,i} = 0 \end{cases} \qquad i = 1 \dots m \tag{9}$$

$$ServerEC_i = isServerUp_i * CalculateServersEnergy\,(VMComTime \quad , P_{server}) \ , \quad i = 1 \dots m \tag{10}$$

$$SwitchEC_i = CalculateSwitchesEnergy(VMComTime \quad , P_{switch}) \ , \ i = 1 \dots s \tag{11}$$

5.4 Proposed VM Placement Algorithms

In this section we describe our proposed VM placement algorithms. The above-explained VM placement problem is NP-hard, and consequently, we propose a meta-heuristic and a heuristic algorithm to solve it. First we introduce our algorithm that is based on the Simulated Annealing technique. Then we describe a heuristic algorithm we have designed for the same problem.

5.4.1 Simulated Annealing Based VM Placement (SABVMP)

Our Simulated Annealing Based VM Placement (SABVMP) technique takes advantage of the first-fit algorithm in its loop. First-fit consolidates VMs in this way: it puts VMs and servers in two queues and starts from the first VM in the VMs queue. Then checks every server from the head of servers queue to see if the server has enough resources and the VM can be placed on that server. If all the VMs are placed, it terminates successfully, but if there is no suitable server for a VM, it aborts. SABVMP iteratively changes the order of VMs in the queue (line 7,Table 2), and then uses first-fit to place the VMs from the queue onto the servers (line 8,Table 2), and then calculates total output traffic of servers (line 9). It also calculates the amount of traffic between each pair of racks, and multiplies this value with a constant representing the cost (time) of communication between those two racks, and then adds this amount to the calculated traffic of servers. The reason behind this formula (lines 4 and 9) is that our experiments show that communication volume among racks has a strong effect on VMs communication time. Note the effect of the topology of the datacenter network in this formula; the constant coefficient mentioned above corresponds to the above topology, or in other words, the structure of the available connections among racks. In the rest of the algorithm (line 10), if the above mentioned amount of total traffic among servers is accepted over previous iterations (line 10), algorithm saves this value as the new best value of simulated annealing and chooses the current order of VMs in queue as the best one.

The function *Accept* in line 10 of algorithm compares the amount of VM communication (among servers in one rack as well as among different racks) in the new order of VMs (O`) against the old one (O) and determines whether to accept O' as the new order or not. If O' reduces the above metric, it is always accepted as the new order; but even if it does not, it may still be accepted by a probability based on *temperature* so as to avoid being stuck in *local* minima. In other words, complying with the Simulated Annealing philosophy of operation, although O` is actually worse than O, this temporary upward move is accepted hoping that future moves find a better minima. This probability of acceptance reduces with *temperature* so that at the beginning, larger parts of the design space are initially explored, but the moves are mostly downward at later stages near the end of the algorithm. By each iteration, the temperature is reduced (line 11) and so does the above probability. The algorithm continues until the temperature reaches its lowest value, *Tl*. The VMs will be placed on servers according to the final order of VMs in the queue. Table 2 shows the SABVMP pseudo code.

Table 2. Pseudo code of our SABVMP algorithm

```
Algorithm 1 – VM Placement with SABVMP
1.  Set the highest temperature to Th, and the coolest temperature to Tl
2.  Initialize VMs order in queue ,X
3.  Do VM placement using first fit
4.  Calculate total output communication of servers (servers traffic + C * racks traffic) , O
5.  T= Th
6.  While temperature is higher than Tl
7.     Randomly change the order of VMs in queue ,X`
8.     Do VM placement using first fit algorithm
9.     Calculate total output communication of servers (servers traffic + C * racks traffic), O`
10.        if ( Accept(O',O, temperature) ) then X = X` and O = O`
11.    Decrease the temperature T by multiplying it by a constant smaller than one
12. End
```

5.4.2 Communication-Aware VM Placement (CAVMP)

In terms of processing capacity of physical machines, the pure VM placement problem is similar to the bin-packing problem in computational complexity theory where objects of different volumes must be packed into a finite number of bins of a certain capacity in a way that minimizes the number of bins used. However, our above communication-aware VM placement problem has other characteristics, most importantly existence of communication among VMs, which necessitates designing new algorithms. Nevertheless, algorithms developed for the basic bin-packing problem can still be used in part. Our heuristic algorithm also partially uses the first-fit technique, which is among bin-packing algorithms. In this heuristic algorithm we try to find VMs that are in the same group and then put them on servers. One of the-differences between CAVMP and SABVMP is that unlike SABVMP, CAVMP does VM placement only in one pass, and hence, is much faster than SABVMP and can be even used online. Table 3 contains the pseudo code of our CAVMP algorithm. CAVMP works in this way: it first chooses a VM randomly from the set of VMs (called **X** in Table 3) and inserts it in the queue X` (line 2) and assigns a *Group ID* to it (line 3). After that, it searches for a VM among VMs in X that has the most communication with VMs that are in the X` queue. This new VM then is appended to the tail of the X` queue and the same Group ID as the previous one is assigned to it. If there is no VM in X that has communication with current VMs in X`, but X still is not empty, it indicates that all the VMs in a group have already been detected and the remaining VMs do not belong to this group. Thus, the Group ID is increased and again a VM is randomly relocated from X to X`. This process repeats until every VM in X is transferred to the X` queue. At this state, all the VMs that are in the same group have been detected and have the same Group ID (lines 4 to 13). Now it is time to place the VMs on servers. The placement phase in the CAVMP algorithm has three steps. At the first step, the algorithm tries to place all the VMs of a group on a single server, so it checks the resource demand of each group against available resources of each server to find a suitable match (line 14). After finishing step one, it is expected that some groups are not placed because their resource demand is more than available resources of all individual servers. Now it is time for step two. In this step CAVMP tries to place the VMs of a group on a single rack. It also tries to place the VMs of a group that have more communication with each other on the same server (line 15). The goal of this step is first, to avoid communication between racks and second, to reduce the communication inside a rack. Finally at the third step, first-fit algorithm is used to place the remaining VMs in X` queue onto servers (line 16). This step is used when the resource demand of a group is more than available resource of all individual racks. This may happen when a group is very large and need a lot of resources, or when the previous groups of VMs have consumed some part of availed resources of racks and consequently, no rack has enough resources left. At this step CAVMP uses a topology-aware variation of the first-fit algorithm to place the VMs such that the VMs with more communication among them are put in the same server, then the same rack, and then the topologically-neighbor racks (this is accomplished by selecting the VMs from the head of the above queue in which the VMs with more communication are put next to each other). Note that this hierarchal placement approach is also the other place where the topology of the datacenter network is taken into account; VMs are assigned to servers and racks considering their topological neighborhood.

Table 3. Pseudo code of CAVMP algorithm

```
Algorithm 2 – VM Placement with CAVMP
X : initial set of VMs in arbitrary order
X` : queue that contains final order of VMs
Group ID : indicates group ID of each VM (It is initialized to zero)
1.  Select a VM randomly from X and remove it
2.  Insert this VM in X`
3. Assign Group ID to this VM
4.  While X is not empty
5.      Find the VM in X that has most communication with VMs in X`
6.      If there is no VM in X that has communication with current VMs in X`
7.          Choose a VM randomly from X
8.          Increase Group ID
9.      End
10.     Assign Group ID to new VM
11.     Append  this new VM to the tail of X`
12.     Remove the selected VM from X
13. End
14. Try to place all the VMs of a group on the same server
15. Try to place the remaining VMs from previous step in the same rack
16. Finally Use topology-aware first-fit to place remaining VMs in X` on servers
```

6. Experimental Results

In this section we first describe the topology and specification of the datacenter IT equipment, and then the benchmarks are described and finally the results of our algorithms are compared to the competitors. We compare results of our two algorithms, SABVMP and CAVMP, to three rivals: a baseline communication-unaware first-fit algorithm as well as an improved version of the SCAVP technique, which we call SCAVP+, where Integer Linear Programming (ILP) is used to find the best placement of VMs within racks based on the same criteria that SCAVP uses. Note that SCAVP [65] is the most relevant communication-aware VM placement work we are aware of. Another rival algorithm with which we compare our algorithms, is the VMFlow approach [64], which is a communication aware algorithm as well, and is introduced in the related work section.

6.1 Datacenter Network Topology

Three-tier tree topology is a common architecture in today datacenters [70], so we used this architecture for evaluation (see Fig. 13); Core switches are at the topmost layer of datacenter network. Next level consists of aggregate switches. Finally at the lowest level, the top-of-rack switches have been placed and servers are connected to them. We assumed 192 servers for the datacenter where every 24 servers are packed in one rack, so we have eight racks in total, and hence, eight top-of-rack switches. There are also four aggregate switches and two core switches in the datacenter as shown in Fig. 13. Each top-of-rack switch is connected to two aggregate switches and each aggregate switch is connected to both core switches. One Gigabit Ethernet (1GE) links are used for connecting servers to top-of-rack switches. All other links between switches are 10GE.

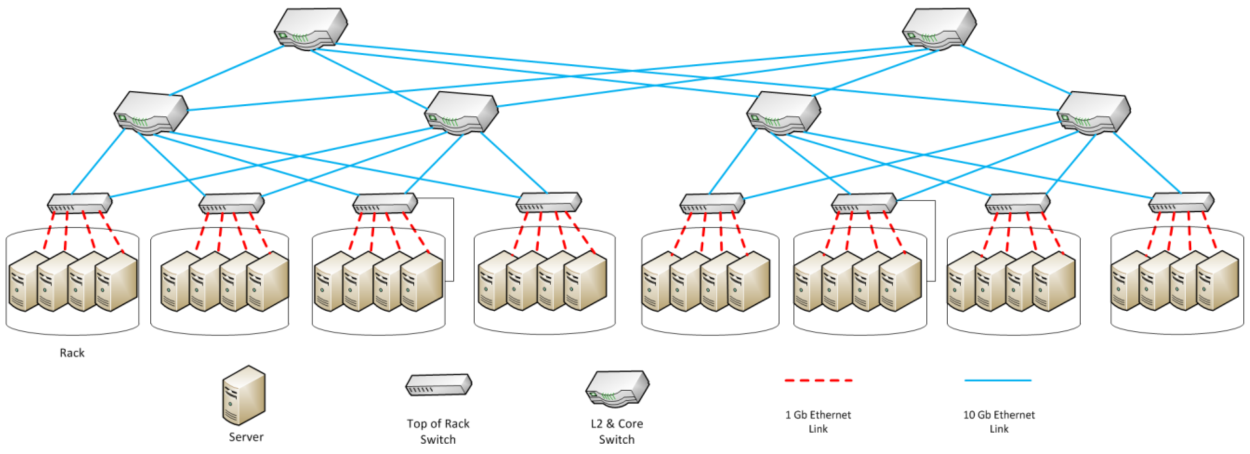


Fig. 13. Three-tier tree topology used in most datacenters and our experiments

In simulations on synthetic benchmarks, we used servers with 8 cores and 12 GB RAM, and obtained the values for power consumption of servers from [71], and for switches from [23]. All other values we used for calculating total energy consumption are listed in Table 4. In the real-world benchmark, the VM specifications are based on the available online data and are given separately in Section 6.2.4. Since top-of-rack switches are different from layer 2 and core switches, power values for them are given in different rows in the table. Each switch in our experiments, both in synthetic as well as real world benchmarks, has one linecard and each linecard has 24 ports, thus we have $\boldsymbol{n_{linecard}} = \mathbf{1}$ and $\boldsymbol{n_{ports}} = \mathbf{24}$.

Table 4. Values used for parameters to calculating energy consumption

Parameter	Value (Watt)
P_{idle}	128
P_{max}	247
$P_{chassis}$ (top of rack switch)	90
$P_{linecard}$ (top of rack switch)	30
P_r (top of rack switch)	1
$P_{chassis}$ (layer 2 and core switches)	520
$P_{linecard}$ (layer 2 and core switches)	35
P_r (layer 2 and core switches)	2

6.2 Results

We used two benchmark classes in our work: a set of synthetic benchmarks, as well as two real world benchmarks. We also used NS2 version *ns-allinone-2.35* in our experiments and Ubuntu 10.04 was the operating system of machine that we used for simulations. The results are presented and discussed below.

6.2.1 Synthetic Benchmarks

The first class of experiments is on ***synthetic benchmark*** where we generated random values for the number of groups, the amount of communication among the VMs in each group, and also for the resource demands of the VMs. For producing these random values, the random generator function of C++ was used which produces *uniformly distributed* random values. Different seed values were used for each benchmark instance of a certain category of benchmarks when producing the random values. Details of these benchmarks are given in Table 5.

The *Pure Runtime* field in Table 5 is pure computation time of each VM and does not include its communication time which experiments show to be roughly 10 minutes so as to be close to the 30-minute period assumed for re-executing the placement algorithm—see Section 4.1. We calculate the communication time for each VM by the NS2 simulator and then add it to the above pure runtime to obtain the actual runtime of each VM. Note that the actual communication time depends on the communication pattern as well as placement of VMs, and hence, due to the randomness of the benchmarks, it is not possible, nor needed, to more accurately tune them to absolutely total 30 minutes of execution time. It is also noteworthy that in all experiments, a homogeneous datacenter is assumed so the pure runtime of VMs does not change on different servers.

Table 5. Specifications of synthetic base point benchmarks used in the experiments

Number of Instances	Pure Runtime (min.)	Number of VMs	Number of Groups	Random values			
				Number of VMs in each Group	Communication between VMs (KB)	Number of Cores for each VM	Amount of RAM for each VM (GB)
10	20	100	40	1 to 4	50,000 to 100,000	1 to 4	1 to 3

We assume that the VMs are provided in groups and that the VMs in a group only communicate to one another and have no communication with the VMs in other groups. This scenario corresponds to most multi-tier implementations of internet-scale services, where machines form multiple groups or tiers each of which serves a specific part needed for accomplishment of the overall task. Furthermore, in many dynamic environments such as Infrastructure-as-a-Service (IaaS) cases, independent services run separately and do not communicate with one another.

Total energy consumptions of the five algorithms are given in Table 6 for each benchmark as well as averaged over all of them. As the table shows, our CAVMP algorithm improves total energy consumption from 15.1% up to 21.0% (16.6% on average) compared to the first-fit algorithm, and from 5.8% up to 13.0% (9.2% on average) compared to the SCAVP+ algorithm. Compared with VMFlow, the CAVMP improves the energy consumption from 5.9% up to 15.1% (10.8% on average). One can see from these results that SCAVP+ is more successful than VMFlow in improving energy consumption. The cause of better improvements in benchmarks 4, 8 and 9 is that in these benchmarks, the amount of communication between VMs is more than other benchmarks. As these numbers are randomly generated, in the mentioned benchmarks these numbers are greater than others and consequently our algorithms have achieved more improvements in energy consumption.

Table 6. Total energy consumption (KWh) comparison of the algorithms on synthetic benchmarks

	Algorithm	Benchmarks										
		1	2	3	4	5	6	7	8	9	10	Average
Total Energy Consumption (KWh)	First-Fit	5.172	5.077	5.107	5.229	5.143	5.11	5.296	5.23	4.963	5.409	5.174
	SCAVP+	4.632	4.518	4.609	4.975	4.627	4.523	4.986	4.695	4.842	5.101	4.751
	VMFlow	4.950	4.782	4.909	4.942	4.831	4.443	4.948	4.866	4.575	5.104	4.835
	SABVMP	4.505	4.312	4.398	4.457	4.444	4.336	4.628	4.388	4.557	4.834	4.486
	CAVMP	4.338	4.118	4.316	4.434	4.361	4.183	4.484	4.131	4.212	4.552	4.313
Improvement (%)	CAVMP vs. First-Fit	16.1	18.9	15.5	15.2	15.2	18.1	15.3	21	15.1	15.9	16.6
	CAVMP vs. SCAVP+	6.4	8.9	6.3	10.9	5.8	7.5	10.1	12	13	10.8	9.2
	CAVMP vs. VMFlow	12.4	13.9	12.1	10.3	9.7	5.9	9.4	15.1	7.9	10.8	10.8

More details of the experimental results on these benchmarks are given in Table 7 to Table 8. In Table 7, breakdown of total power consumption into that of servers and switches is provided. The results indicate that CAVMP improves the energy consumption of servers from 10.3% up to 22.3% (16.8 % on average) and energy consumption of switches from 15.8% up to 19.6% (17.7% on average) compared to first-fit algorithm. Improvement of CAVMP compared to SCAVP+ for energy consumption of severs is from 8% up to 15.9% (10.4% on average) and for energy consumption of switches is from 0.1% up to 15.7% (5.9% on average).

Table 8 gives network activity details of the benchmarks after being placed on physical servers by the algorithms. The results show that CAVMP reduces number of created packets from 46.2% up to 71.7% (55.4% on average) compared to first-fit algorithm, and from 44.5% up to 71.9% (54.3% on average) compared to SCAVP+ algorithm. For average communication time, CAVMP improves it from 54.5% up to 80.1% (62.7% on average) compared to first-fit, and from 42.6% up to 75% (55.1% on average) compared to SCAVP+ algorithm. The results clearly prove our claims earlier in the paper that too simplistic models for communication overhead are inaccurate and mislead prior placement algorithms.

In section 3.3.6 we claimed that paying attention only to communication volume is not enough and this factor alone cannot properly show the effectiveness of a VM placement approach. Now here, Table 8 demonstrates validity of this claim. As an example consider the results of benchmark 4 in

Table 8. In this benchmark SABVMP algorithm produces fewer packets than CAVMP (8306332 vs. 8852241 packets respectively) while its average communication time is more than CAVMP algorithm (6.76s vs. 5.63s respectively).

Table 7. Breakdown of energy consumption of servers and switches for the algorithms

		Algorithms														
		Energy Consumption of Servers (KWh)					Energy Consumption of Switches (KWh)					Total Energy Consumption (KWh)				
		First-fit	SCAVP+	VMFlow	SABVMP	CAVMP	First-fit	SCAVP+	VMFlow	SABVMP	CAVMP	First-fit	SCAVP+	VMFlow	SABVMP	CAVMP
Benchmarks	1	3.589	3.299	3.295	3.173	3.007	1.583	1.333	1.655	1.332	1.331	5.172	4.632	4.950	4.505	4.338
	2	3.454	3.185	3.207	2.983	2.789	1.623	1.333	1.576	1.329	1.329	5.077	4.518	4.782	4.312	4.118
	3	3.485	3.275	3.252	3.066	2.985	1.622	1.334	1.657	1.332	1.331	5.107	4.609	4.909	4.398	4.316
	4	3.605	3.435	3.326	3.125	3.102	1.624	1.54	1.616	1.332	1.332	5.229	4.975	4.942	4.457	4.434
	5	3.561	3.294	3.255	3.113	3.029	1.582	1.333	1.576	1.331	1.332	5.143	4.627	4.831	4.444	4.361
	6	3.487	3.19	3.079	3.005	2.853	1.623	1.333	1.364	1.331	1.33	5.11	4.523	4.443	4.336	4.183
	7	3.672	3.446	3.372	3.295	3.152	1.624	1.54	1.576	1.333	1.332	5.296	4.986	4.948	4.628	4.484
	8	3.607	3.361	3.248	3.056	2.826	1.623	1.334	1.618	1.332	1.305	5.23	4.695	4.866	4.388	4.131
	9	3.341	3.264	3.085	3.102	2.882	1.622	1.578	1.490	1.455	1.33	4.963	4.842	4.575	4.557	4.212
	10	3.785	3.563	3.527	3.418	3.219	1.624	1.538	1.577	1.416	1.333	5.409	5.101	5.104	4.834	4.552

Table 8. The number of packets and average communication time in synthetic benchmarks

		Algorithms									
		Number of Created Packets					Average Communication time (Minutes)				
		First-fit	SCAVP+	VMFlow	SABVMP	CAVMP	First-fit	SCAVP+	VMFlow	SABVMP	CAVMP
Benchmarks	1	16660024	16334225	12968821	9389696	7315590	12.98	10.63	10.47	8.11	4.92
	2	16202921	16340557	10994352	8249222	4998043	12.82	10.73	9.07	6.71	3.35
	3	16372259	15868397	10511741	10153529	7715610	12.77	10.82	9.32	8.59	5.57
	4	16702210	16344139	12493124	8306332	8852241	13.15	10.78	10.1	6.76	5.63
	5	15979576	15475093	10798503	8224744	6830880	12.95	10.62	8.95	7.35	4.63
	6	17237219	16576796	11677803	8079757	8863314	13.18	10.45	9.28	7.27	6
	7	16343172	16008757	12834956	9653977	7560433	12.88	10.67	9.64	7.87	5.09
	8	14640857	14773285	10408409	6521748	4146000	13.59	10.82	9.25	5.69	2.71
	9	15688719	15191705	11438370	7457326	7630100	12.89	11.28	9.25	7.36	4.73
	10	15935169	15447530	12085616	9145935	8566678	12.78	11.03	9.9	8.01	5.73

6.2.2 Analysis of sensitivity to the number of groups

Each group of VMs represents a service that is providing a service to the users. Assuming that the services work independently, no communication happens among different groups whereas the VMs in each group may communicate to one another to provide the designated service. The number of these groups, or services, can affect the outcome of the placement algorithms. Thus, we designed two more sets of experiments to evaluate this effect. Similar to the base point case in previous section, each set of experiments consists of 10 benchmarks with random values as in Table 5 produced with different seed values; the two sets of experiments contain 30 and 50 groups respectively.

Comparing the energy saving results of the cases for 30, 40, and 50 groups in Fig. 14, it is clearly seen that our technique performs more effectively when the number of groups increases for the same number of total VMs. This shows the significance of considering inter-VM communication even when placing VMs among servers inside a single rack; our technique considers this issue whereas SCAVP (and hence SCAVP+) suffice to only inter-rack communication, and hence, our algorithm better packs groups on individual servers to reduce inter-server traffic.

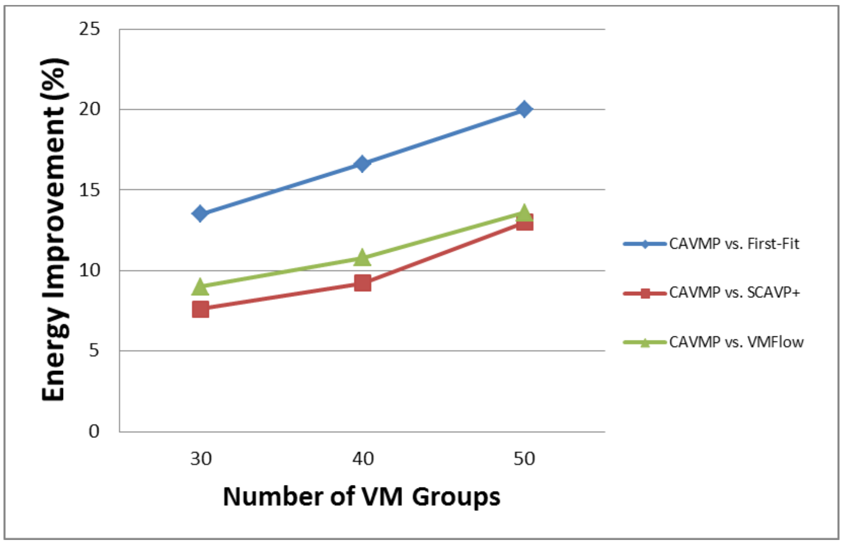


Fig. 14. The energy saving of our algorithm improves with the number of VM groups (i.e., the number of services)

6.2.3 Analysis of sensitivity to the amount of communication

The volume of communication among the VMs in each group is another important factor that can affect how much energy can be saved by the placement techniques. We conducted another two sets of experiments, again with 10 randomly generated benchmarks as before, but with two other ranges of communication volume among their VMs: one set with communication in the range of 5,000 KB to 10,000 KB of communication between each VM pair, and another one with 100,000 KB to 200,000 KB communication size. The summary of results is shown in Fig. 15. As expected, when there is little communication among servers, no more energy can be saved compared to SCAVP+ algorithm since the differentiating factor of our technique is not seen in the benchmark. However, our technique shines with higher communication volume among VMs.

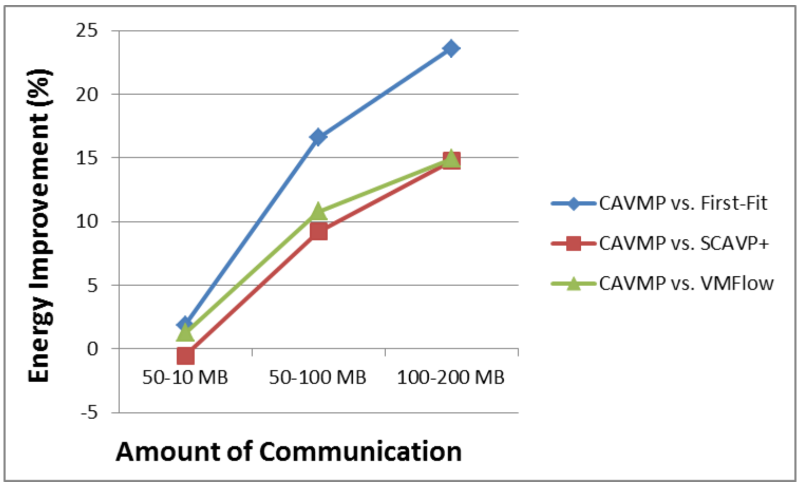


Fig. 15. The energy saving of our algorithm improves with the amount of communication among VMs.

Another interesting observation in this set of experiments is that our saving is marginally, 0.5%, below that of SCAVP+ at 5-10 MB communication size range. This is because we applied ILP technique in SCAVP+ to improve SCAVP for the purpose of best fitting VMs among servers in a rack, and hence, it outperforms our own heuristic CAVMP placement when there is little to no communication among VMs. However note that our CAVMP takes only 0.015s to run on an Intel Corei3 machine (OS: Windows 7, 4 GB of RAM, CPU frequency: 2.93 GHz) in this case whereas SCAVP+ takes two orders of magnitude more time, 3.62s, for the same case Thus, our technique is a better choice for *online* usage even in this case.

It is noteworthy that in all the experiments we considered the CPU utilization of VMs, not servers, to be zero while communicating and 100% while computing, but the utilization of servers depends on the number of VMs placed on them and can be any valid value U.

6.2.4 Real World Benchmark: Wikipedia servers

The second class of benchmarks we used is two ***real world benchmarks*** obtained from Wikipedia servers [72] and other sources [73-77] which we call *RB2 (real-world benchmark 2)* from now on for ease of reference. The Wikipedia servers are monitored using Ganglia infrastructure which provides the information of each server such as memory and CPU usage and the amount of communication. In this benchmark, we assumed each physical machine of Wikipedia is a virtual machine. Wikipedia actual physical machines are of various types and have different resources; for example, one of their machine has12 CPUs with 2.00GHz frequency and its CPU utilization is 5%, while another one has 8 cores with 2.66GHz clock frequency and CPU utilization of 9 % [72].Thus, we had to convert them to our available cores of VMs to be able to use them in our experiments. Consequently, after the conversion we assume that each virtual core of VMs works at full utilization. Note that the actual CPU utilization of the physical server depends on the number of VMs assigned to that server, and hence the server cores are not necessarily 100% utilized. Each machine belongs to a module of the application that is running on Wikipedia servers. We considered each module as a group and assumed that each physical machine mainly communicates with other physical machines inside that group.

For each physical machine, Ganglia provides only aggregate input and output traffic volume but does not give the amount of communication between every pair of machines. In order to create the communication matrix, that indicates the amount of communication between every two machines in each interval, we used the simple gravity model [78]. In this model for calculating the traffic between two machines, *i* and *j*, the following equation is used:

$$C_{i,j} = (out_i * in_j) / (\sum_k in_k)$$

where, out_i is output traffic of physical machine *i* and in_i is input traffic of physical machine *j* [64]. It is noteworthy that although this model cannot accurately determine actual communication among machines since in reality some machines only serve independent requests and do not communicate with other nodes, but due to lack of any further details of communication, there is no more accurate alternative and we had to suffice to this model.

The Wikipedia benchmark has 100 VMs and RB2 has 132 VMs, and the pure runtime (not covering communication time) of all the VMs is 600 minutes in Wikipedia benchmark and 300 minutes in the RB2 one. The number of cores that each VM needs in these benchmarks is more than our synthetic ones, so we used bigger servers in simulations of these benchmarks. Here each server has 16 cores and 64 GB memory and for the power parameters, we have P_idle=283 W, P_max=388 W [79]. The power parameters for switches are the same as Table 4.

First, we describe the Wikipedia benchmark. This benchmark consists of 10 groups and each of these groups has its own number of VMs. A variety of resource demands are found in these VMs but their most influential difference is the pattern of communication among these VMs. As can be seen in Table 9, some groups have little communication inside themselves despite their big size, but some other small ones such as 4 and 5 have very high communication. This is important because the energy consumption results that are presented later heavily depend on this communication pattern.

The RB2 benchmark has 19 groups with total 132 VMs. In this benchmark, unlike the previous one, we can only see group No. 9 with small size and high communication and it will affect the energy consumption that we will see later.

Table 9. Details of the communication pattern of our real world benchmarks

Wikimedia benchmarks			RB2 benchmark					
Group No.	**Number of VMs**	**Average amount of communication between VMs in each interval (KB)**	**Group No.**	**Number of VMs**	**Average amount of communication between VMs in each interval (KB)**	**Group No.**	**Number of VMs**	**Average amount of communication between VMs in each interval (KB)**
1	37	11,000	1	4	500,000	11	7	700,000
2	31	12,500	2	15	4,000	12	2	220,000
3	4	58,000	3	2	50,000	13	15	60,000
4	4	5,000,000	4	8	5,000	14	6	64,000
5	4	3,200,000	5	7	25,000	15	16	22,000
6	4	130,000	6	3	50,000	16	6	400,000
7	6	10,000	7	4	15,000	17	4	60,000
8	3	950,000	8	4	14,000	18	16	6,000
9	3	40,000	9	5	1,000,000	19	4	210,000
10	4	230,000	10	4	200,000			

The first parameter to compare is the number of up servers. In the Wikipedia benchmark, SCAVP+ and SABVMP algorithms used 31 servers while the first-fit and CAVMP used 32 servers. The difference is more in the RB2 benchmark where SCAVP+ uses the least number of servers with 23 servers, and after that is SABVMP with 24 servers. The third rank in terms of fewer numbers of servers is for first-fit with 26 servers and then comes VMFlow with 28 ones. As we expected, here again the CAVMP uses the most servers by turning on 37 ones. The reason that in the RB2 benchmark we see a bigger difference among the number of used servers by various algorithms is the very high diversity of VMs in terms of demand for resources; this leads to different results in different algorithms depending on how the algorithm decides which server to turn on. Note that CAVMP again uses more servers than SCAVP+ since SCAVP+ takes advantage of an efficient bin-packing based placement.

To assess how much traffic each algorithm transmits, we count the number of transmitted packets that are transferred over the network of the datacenter. Fig. 16 shows the number of packets that each algorithm transmits. The size of each packet is 8KB. As can be seen, CAVMP transmits fewer packets than any other algorithm and significantly improves the number of transmitted packets by 77% compared to SCAVP+ and 53% compared to VMFlow in the Wikipedia benchmark. The reason for this dramatic decrease in the number of transmitted packets is that CAVMP places all the VMs in groups 4, 5 and 8 on the same servers, and since each of these groups have huge communication inside themselves; this communication volume is offloaded from the datacenter network. In the RB2 benchmark, we can see that the difference between CAVMP and the rival algorithms is less than Wikipedia benchmark since groups such as 4, 5 and 8 from Wikipedia do not exist in this benchmark. Here the CAVMP decreases the communication volume by 49% and 39% compared to SCAVP+ and VMFlow respectively. The number of transmitted packets for SCAVP+ and first-fit are close to each other, however since SCAVP+, unlike first-fit, tries to reduce the communication among the racks, most of its communication is between servers inside the same rack; we will shortly see the effect of it on energy consumption.

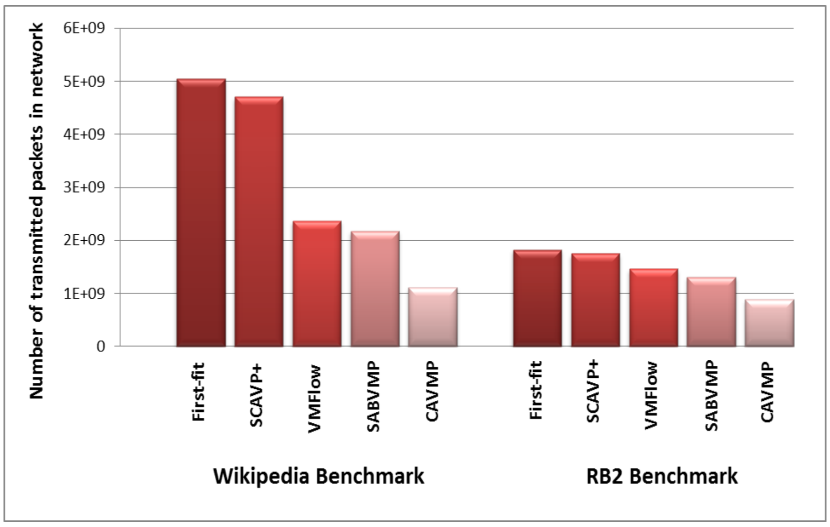


Fig. 16. Number of packets each algorithm transmits over the datacenter network in real world benchmarks

Average communication times for VMs in each algorithm are compared in Fig. 17. As the figure shows, first-fit performs the most poorly because it is totally communication-unaware. In the Wikipedia benchmark, CAVMP shows the best operation and its average VM communication time is about 93% less than first-fit, about 92% less than SCAVP+ and 93.5% less than VMFlow. We observe that while the amount of communication in VMFlow was less than SCAVP+, the communication time is more. This observation perfectly shows the disability of VMFlow to simultaneously reduce the communication time as well as communication volume. The VMFlow algorithm is partially successful in decreasing the communication volume, but since it does not propose an intelligent placement, this reduction does not improve the communication time. For example if a group with a lot of inter-VM communication has five members, this algorithm puts four of them in a server in a rack and the fifth one in another rack. In such case, while the amount of communication between the four VMs is eliminated, the communication between the fifth one and others cause all the switches and servers that are involved in the communication to remain ON and consume energy. We will provide the experimental results of its effects later.

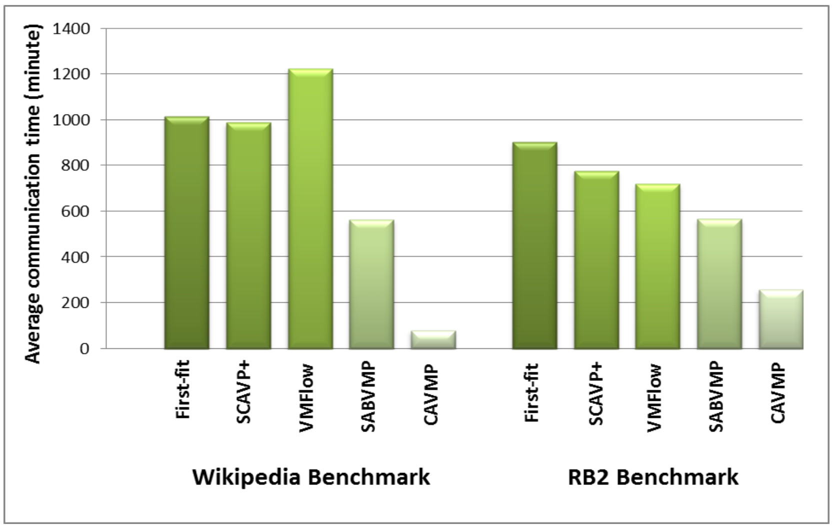


Fig. 17. Average VM communication time in each algorithm in our real world benchmarks

The downside of concentrating all communication in a few servers is the increase in packet drops. Since first-fit distributes communication across the servers, percentage of packets that are dropped is less than three others. Other algorithms try to compact the communication and it results in more dropped packets. Although the percentage of packet drop in other algorithms is more than first-fit, it is still marginal. It is observable in Table 10 that while an algorithm tries to compact the traffic on fewer servers, the amount of dropped packets increases. Further note that since the network protocol used in the experiments is TCP, the dropped packets are retransmitted, and hence, the final result returned by NS2 covers this retransmission time as well.

Now we compare the amount of energy that the placement suggested by each algorithm consumes. As mentioned before, in the experiments we only consider the energy that the IT equipment (servers and switches) consume in the datacenter but do not take into account the energy consumption of other parts such as the cooling system. We calculate the amount of energy that each individual switch and server consumes and according to them calculate the total energy consumption. Results are depicted in Fig. 18, and full details are given in Table 13.

As in synthetic benchmarks, we considered full CPU utilization for VMs during run-time and zero utilization during communication time; utilization of physical servers is based on the number of VMs running on them at each time. For example if all the VMs on a server are solely communicating in a specific period, the utilization of that server is then considered zero and the server power consumption is only its idle power.

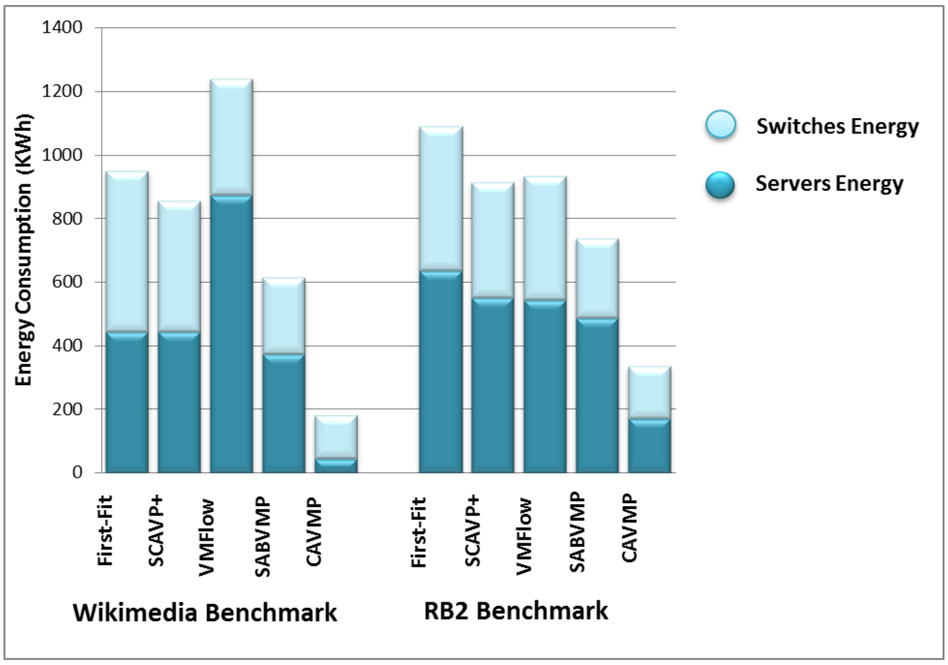


Fig. 18. Energy consumption of each algorithm under the real world benchmarks

Table 10. Summary of experimental results for real world benchmarks

	Wikimedia Benchmark					RB2 Benchmark				
	First-fit	*SCAVP+*	*VMFlow*	*SABVMP*	*CAVMP*	*First-fit*	*SCAVP+*	*VMFlow*	*SABVMP*	*CAVMP*
Number of Up servers	32	31	31	31	32	26	23	28	24	37
Number of transmitted packets	5.1E+09	4.7E+09	2.4E+09	2.2E+09	1.1E+09	1.8E+09	1.8E+09	1.5E+09	1.3E+09	8.9E+08
Average VM communication time (minute)	1015	987	1223.68	563.36	79.32	901.939	777.705	720.03	565.947	258.697
Packet drop (%)	0.246	0.4263	0.249	0.8163	1.9755	0.027	0.0277	0.0303	0.0383	0.0497
Servers energy consumption (KWh)	504.805	411.061	367.375	239.077	138.99	453.971	362.156	389.164	247.461	164.099
Switches energy consumption (KWh)	445.623	444.099	872.136	374.078	43.8826	635.131	551.286	544.087	487.481	169.17
Total energy consumption (KWh)	950.428	855.16	1239.51	613.156	182.872	1089.1	913.442	933.25	734.942	333.27
Energy reduction compared to SCAVP+ (%)	-11	0	-45	29	79	-19	0	-2	20	64

As a result, CAVMP provides the best placement and reduces the energy consumption by 81% compared to first-fit in Wikipedia benchmark and 69% in RB2 benchmark. Comparison of SCAVP+

and CAVMP indicates that CAVMP has better results and can save energy consumption by 79% and 63% in Wikipedia benchmark and RB2 benchmark respectively compared to SCAVP+. This clearly shows the importance of considering inter-VM communication (compare to first-fit), and more importantly, the significance of considering the structure of server as well as racks (compared to SCAVP+), when placing VMs in order to reduce total energy consumption.

Please note that the reason for such a big difference between the improvement of synthetic benchmarks and real world ones is the huge amount of inter-VM communication in real world benchmarks. The higher the inter-VM communication, the better the benefits of our approach. Note that if we increase the amount of communication in synthetic benchmarks, the same results as real world benchmarks will be obtained

6.2.5 Algorithm Execution Time

Table 11 gives execution time of SABVMP and CAVMP algorithms when run on an Intel Core i3 2.93GHz machine with 4GB of memory. Execution time of CAVMP is two orders of magnitude less than SABVMP. Indeed CAVMP execution time is very low and so it can be effectively used as an online algorithm. Due to absence of any pre-processing at the placement time (such as grouping the VMs that other algorithms do), First-fit algorithm has the least execution time.

Table 11. Execution time of algorithms in real world benchmark in seconds

	First-fit	***SCAVP***	***VMFlow***	***SABVMP***	***CAVMP***
Ganglia	0.088 s	3.58 s	0.481 s	36.069 s	0.444 s

6.3 Overhead of Re-running VM placement

In the above experiments we considered a single time interval after placing the VMs, or a semi-static environment with some constant set of services and VMs, such as the real-world Wikipedia servers. However for more dynamic environments such as IaaS cases, in which new services may start and current services may terminate from time to time, we run the VM placement algorithm periodically as in Fig. 12 and find the new placement based on the new status of VMs. This may impose two overheads that should be assessed here: the overhead of executing the placement algorithm, and the overhead of migrating VMs among servers if needed. The execution time of our online algorithm, CAVMP, is negligible (as detailed above and quantified in Table 11), and hence, it has virtually no overhead. The newly run placement algorithm may decide a new place for a number of existing VMs to improve energy consumption in the new interval, and hence, these VMs should be live migrated to other servers. Thus, it is important to evaluate the energy and time overhead of these migrations. The actual number of migrations depends on the actual case and the number and communication pattern of newly arrived VMs as well as that of finished ones. We provide a worst-case analysis here to show the overhead is negligible even if all VMs have to undergo a live migration at the beginning of the new interval.

The required time to do a live migration can be calculated as the VM's size of memory divided by available network bandwidth [80]. This is because the data and images of VMs are usually stored on a Network Attached Storage (NAS) so there is no need to move them among servers [80]. Considering n VMs with M GB of memory each (worst case in Table 5), n.M GB should be moved among m servers which provide m Gbps bandwidth assuming 1Gbps Ethernet links. Thus, it takes 8×M×n/m seconds to migrate all n VMs. In other words, for the values in synthetic benchmarks as in Table 5, it takes around 8×3×100/30=80 seconds to do all migrations if in the worst case all VMs were decided to be moved. This represents worst case overhead of negligibly 4.4% if the re-placement interval is 30 minutes as assumed in Section 4.1. We repeat that deciding the optimal interval for VM re-placement is an interesting objective that we intend to explore as a future direction for further research.

7. Conclusion and Future Work

Datacenters' voracious appetite for energy has motivated researchers around the globe to find techniques and mechanisms for quenching it. Cooling system and IT equipment are of paramount importance regarding energy consumption in datacenters. IT equipment itself can be split into computing resources and communication resources. The focal point of this chapter is energy efficiency of networking equipment in datacenters. The chapter has surveyed and categorized various approaches in this area and tried to present them in a clear picture. Finally, a new approach by the authors of chapter is illustrated that tries to reduce the energy consumption of datacenters by considering communication among the VMs.

In the following, we indicate some future directions regarding reducing usage of equipment which is the main concern of this chapter. In the VDC placement, one important challenge that needs to be addressed is the resource fragmentation. The current approaches just consider the directly connected nodes instead of the whole topology of physical infrastructure. So, the resource fragment problems are inevitable in current state. Further research in this area and trying to investigate the network topology can lead to better placement and better use of resources.

Regarding hypervisor enhancement, one future direction can be considering multi-core systems instead of single core ones. Simplifying the problem of resource scheduling among guest domains to cases where there is just one physical core can’t satisfy the modern systems with several cores. Combining software solutions such as aggregating flows with hardware advances like multiqueues devices can further improve the network performance of virtualized systems. So, it can be another direction for future research in the scope of hypervisor enhancement. Finally, making the network I/O virtualization approaches adaptive to heterogeneous workloads can be another avenue for subsequent researches.

New topologies that try to enhance the network performance in datacenters need more attention. Although some routing algorithms are proposed for them, there is still space for proposing more efficient and practical routing algorithms. Various aspects of these newly proposed architectures such as bisection bandwidth or their incremental nature need more investigations. There are also opportunities for designing new architectures based on servers that have more than two NICs.

Considering the heterogeneous network equipment and try to utilize them in an efficient way by traffic engineering and co-location different flows can be a direction for future researches. Some of the current approaches just consider the state of switches (being ON/OFF) or their active time. However, the bit rate of ports as well as the number of used/unused ports of switch also can affect the decision making in traffic engineering and bring some valuable opportunities for energy saving.

Also, there are a number of avenues of research we propose to follow as future directions regarding our approach and other communication aware consolidation approaches. One open challenge in this work is to find the optimal solution of the problem. To reach this goal it is necessary to involve an estimation of network communication time in the placement phase. Results of NS2 simulation cannot be directly used here since the execution time of each NS2 simulation is very high (up to several hours). Finding a way to estimate the communication time without needing NS2 simulations per iteration is necessary here.

Finally, considering and evaluating the effects of other datacenter structural features such as cooling structure and thermal effects (e.g. rack-based vs. row-based vs. room-based cooling), network topologies, power distribution mechanisms (e.g. modular distributed UPS vs. centralized UPS), as well as application behaviors (e.g. burst network usage in cases such as shuffling phase of MapReduce jobs) and their effects and interaction with VM placement are other interesting questions to seek answers for.

Acknowledgment

This work was supported by the Iran Telecommunication Research Center [500/12188/t].

References

[1] Google, http://www.google.com/about/datacenters/efficiency/internal, accessed November 10th, 2014.

[2] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, pp. 243-256, 2010.

[3] V. K. Arghode and Y. Joshi, "Modeling Strategies for Air Flow Through Perforated Tiles in a Data Center," *IEEE Transactions on Components, Packaging and Manufacturing Technology,* , vol. 3, pp. 800-810, 2013.

[4] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach," *IEEE Transactions on Parallel and Distributed Systems,* vol. 19, pp. 1458-1472, 2008.

[5] A. Sansottera and P. Cremonesi, "Cooling-aware workload placement with performance

constraints," *Performance Evaluation,* vol. 68, pp. 1232-1246, 2011.

[6] Facebook, https://www.facebook.com/notes/facebook-engineering/designing-a-very-efficient-data-center/10150148003778920, accessed November 10th, 2014.

[7] D. Wong and M. Annavaram, "Scaling the Energy Proportionality Wall with KnightShift," *IEEE Micro,* vol. 33, pp. 28-37, 2013.

[8] L. Tan, C. Minghua, and L. L. H. Andrew, "Simple and Effective Dynamic Provisioning for Power-Proportional Data Centers," *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, pp. 1161-1171, 2013.

[9] L. Minghong, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic Right-Sizing for Power-Proportional Data Centers," *IEEE/ACM Transactions on Networking,* vol. 21, pp. 1378-1391, 2013.

[10] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, "NapSAC: design and implementation of a power-proportional web cluster," *SIGCOMM Computer Communication Review,* vol. 41, pp. 102-108, 2011.

[11] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 119-128, 2007.

[12] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer,* vol. 40, pp. 33-37, 2007.

[13] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 41-50, 2009.

[14] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in Clouds through gossiping," in *IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM),* pp. 1-6, 2011.

[15] X. Wang and Z. Liu, "An Energy-Aware VMs Placement Algorithm in Cloud Computing Environment," in *International Conference on Intelligent System Design and Engineering Application (ISDEA),* pp. 627-630, 2012.

[16] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based Optimization of Power and Migration Cost in Cloud Computing," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 172-179, 2012.

[17] S. Weiming and H. Bo, "Towards Profitable Virtual Machine Placement in the Data Center," in *IEEE International Conference on Utility and Cloud Computing (UCC)*, pp. 138-145, 2011.

[18] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: Data Center Energy-Efficient Network-Aware Scheduling," in *IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing Green Computing and Communications (GreenCom),(CPSCom),* pp. 69-75, 2010.

[19] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on OpenStack Cloud," *Future Generation Computer Systems,* vol. 32, pp. 118-127, 2014.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Computer Communication Review*, vol. 39, pp. 68-73, 2008.

[21] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th annual international symposium on Computer architecture (ISCA)*, pp. 338-347, 2010.

[22] L. A. Barroso, J. Clidaras, and U. Hlzle, *the Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, Morgan & Claypool Publishers, 2013.

[23] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," *NETWORKING*, pp. 795-808, 2009.

[24] C. Eddington, "InfiniBridge: an InfiniBand channel adapter with integrated switch," *IEEE Micro*, vol. 22, pp. 48-56, 2002.

[25] B. Dickov, M. PericÃ s, P. M. Carpenter, N. Navarro, and E. AyguadÃ, "Analyzing Performance Improvements and Energy Savings in Infiniband Architecture using Network Compression," in *Proceedings of the IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, pp. 73-80, 2014.

[26] V. Sundriyal, M. Sosonkina, A. Gaenko, and Z. Zhang, "Energy saving strategies for parallel applications with point-to-point communication phases," *Journal of Parallel and Distributed Computing*, vol. 73, pp. 1157-1169, 2013.

[27] C. Decusatis, "Optical interconnect networks for datacom and computercom," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, pp. 1-33, 2013.

[28] D. Larrabeiti, P. Reviriego, J. A. Hern, Ndez, J. A. Maestro, M. Urue, "Towards an energy efficient 10 Gb/s optical ethernet: Performance analysis and viability," *Optical Switching and Networking*, vol. 8, pp. 131-138, 2011.

[29] R. Danping, L. Hui, and J. Yuefeng, "Power saving mechanism and performance analysis for 10 Gigabit-class passive optical network systems," in *IEEE International Conference on Network Infrastructure and Digital Content*, pp. 920-924, 2010.

[30] P. Mahadevan, S. Banerjee, and P. Sharma, "Energy proportionality of an enterprise network," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pp. 53-60, 2010.

[31] C. Yiu and S. Singh, "Merging traffic to save energy in the enterprise," in *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, pp. 97-105, 2011.

[32] A. Carrega, S. Singh, R. Bolla, and R. Bruschi, "Applying traffic merging to datacenter networks," in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, Article No. 3, 2012.

[33] Q. Yi and S. Singh, "Minimizing Energy Consumption of FatTree Data Center Networks," *SIGMETRICS Performance Evaluation Review*, vol. 42, pp. 67-72, 2004.

[34] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)," *IEEE Transactions on Computers*, vol. 57, pp.

448-461, 2008.

[35] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pp. 29-34, 2010.

[36] Q. Xiao, J. Hong, A. Manzanares, R. Xiaojun, and Y. Shu, "Communication-Aware Load Balancing for Parallel Applications on Clusters," *IEEE Transactions on Computers,* vol. 59, pp. 42-52, 2010.

[37] Z. Jidong, S. Tianwei, H. Jiangzhou, C. Wenguang, and Z. Weiming, "Efficiently Acquiring Communication Traces for Large-Scale Parallel Applications," *IEEE Transactions on Parallel and Distributed Systems,* vol. 22, pp. 1862-1870, 2011.

[38] D. M. Divakaran, L. Tho Ngoc, and M. Gurusamy, "An Online Integrated Resource Allocator for Guaranteed Performance in Data Centers," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, pp. 1382-1392, 2014.

[39] S. Luo, H. Yu, L. Li, D. Liao, and G. Sun, "Traffic-aware VDC embedding in data center: A case study of fattree," *China Communications,* vol. 11, pp. 142-152, 2014.

[40] W. Xiaohui, L. Hongliang, Y. Kun, and Z. Lei, "Topology-Aware Partial Virtual Cluster Mapping Algorithm on Shared Distributed Infrastructures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, pp. 2721-2730, 2014.

[41] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual Data Center Embedding across Distributed Infrastructures," *IEEE Transactions on Cloud Computing,* vol. 1, pp. 36-49, 2013.

[42] M. Bourguiba, K. Haddadou, I. El Korbi, and G. Pujolle, "Improving Network I/O Virtualization for Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, pp. 673-681, 2014.

[43] G. Haibing, M. Ruhui, and L. Jian, "Workload-Aware Credit Scheduler for Improving Network I/O Performance in Virtualization Environment," *IEEE Transactions on Cloud Computing,* vol. 2, pp. 130-142, 2014.

[44] G. Bei, W. Jingzheng, W. YongJi, and S. U. Khan, "CIVSched: A Communication-Aware Inter-VM Scheduling Technique for Decreased Network Latency between Co-Located VMs," *IEEE Transactions on Cloud Computing,* vol. 2, pp. 320-332, 2014.

[45] S. Govindan, C. Jeonghwan, A. R. Nath, A. Das, B. Urgaonkar, and S. Anand, "Xen and Co.: Communication-Aware CPU Management in Consolidated Xen-Based Hosting Platforms," *IEEE Transactions on Computers,* vol. 58, pp. 1111-1125, 2009.

[46] G. Qu, Z. Fang, J. Zhang, and S. Zheng, "Switch-Centric Data Center Network Structures Based on Hypergraphs and Combinatorial Block Designs," *IEEE Transactions on Parallel and Distributed Systems,* vol. 26, pp. 1154-1164, 2015.

[47] G. Deke, C. Tao, L. Dan, L. Mo, L. Yunhao, and C. Guihai, "Expandable and Cost-Effective Network Structures for Data Centers Using Dual-Port Servers," *IEEE Transactions on Computers,* vol. 62, pp. 1303-1317, 2013.

[48] L. Yong, Y. Dong, and G. Lixin, "DPillar: Scalable Dual-Port Server Interconnection for Data Center Networks," in *Proceedings of 19th International Conference on Computer*

Communications and Networks (ICCCN), pp. 1-6, 2010.

[49] D. Li and J. Wu, "On Data Center Network Architectures for Interconnecting Dual-Port Servers," *IEEE Transactions on Computers,* early access, under press.

[50] L. N. Bhuyan and D. P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Transactions on Computers,*vol. C-33, pp. 323-333, 1984.

[51] G. Panchapakesan and A. Sengupta, "On a lightwave network topology using Kautz digraphs," *IEEE Transactions on Computers*,vol. 48, pp. 1131-1137, 1999.

[52] F. T. Leighton, *Introduction to parallel algorithms and architectures: array, trees, hypercubes*: Morgan Kaufmann Publishers Inc., 1992.

[53] Y. Shang, D. Li, M. Xu, and J. Zhu, "On the Network Power Effectiveness of Data Center Architectures," *IEEE Transactions on Computers* early access, under press.

[54] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *Proceedings of the ACM SIGCOMM conference on Data communication*, pp. 75-86, 2008.

[55] L. Dan, G. Chuanxiong, W. Haitao, K. Tan, Z. Yongguang, and L. Songwu, "FiConn: Using Backup Port for Server Interconnection in Data Centers," in *IEEE INFOCOM*, pp. 2276-2285, 2009.

[56] A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Distributed Energy Efficient Clouds Over Core Networks," *Journal of Lightwave Technology*, vol. 32, pp. 1261-1281, 2014.

[57] W. Lin, Z. Fa, J. Arjona Aroca, A. V. Vasilakos, Z. Kai, H. Chenying, L. Dan, and L. Zhiyong, "GreenDCN: A General Framework for Achieving Energy Efficiency in Data Center Networks," *IEEE Journal on Selected Areas in Communications,*vol. 32, pp. 4-15, 2014.

[58] D. Li, Y. Yu, W. He, K. Zheng, and B. He, "Willow: Saving Data Center Network Energy for Network-limited Flows," *IEEE Transactions on Parallel and Distributed Systems,* early access, under press.

[59] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pp. 19-19 , 2010.

[60] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on OpenStack Cloud," *Future Generation Computer Systems*, vol. 32, pp. 118-127, 2014.

[61] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, pp. 1027-1034, 2011.

[62] M. Cao Le Thanh and M. Kayashima, "Virtual machine placement algorithm for virtualized desktop infrastructure," in *IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 333-337, 2011.

[63] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual

machines," *Future Generation Computer Systems*, vol. 28, pp. 469-477, 2012.

[64] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "VMFlow: leveraging VM mobility to reduce network power costs in data centers," in *Proceedings of the international IFIP TC 6 conference on Networking - Volume Part I*, pp. 198-211, 2011.

[65] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement," in *IEEE International Conference on Services Computing (SCC)*, pp. 72-79, 2011.

[66] W. Meng, M. Xiaoqiao, and Z. Li, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proceedings IEEE INFOCOM*, pp. 71-75, 2011.

[67] M. Xiaoqiao, V. Pappas, and Z. Li, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," in *Proceedings IEEE INFOCOM*, pp. 1-9, 2010.

[68] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pp. 337-350, 2008.

[69] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the conference on Power aware computing and systems*, pp. 3-3, 2008.

[70] OL-11565-01, *Cisco data center infrastructure 2.5 design guides*, 2007.

[71] Dell PowerEdge R710 Featuring the Dell Energy Smart 870W PSU and Intel Xeon E5620, *ENERGY STAR Power and Performance Data Sheet*.

[72] Ganglia monitoring tool, available online at http://ganglia.wikimedia.org/, accessed July 6th , 2014.

[73] http://neos-server.org/ganglia/, accessed July 6th, 2014.

[74] http://ganglia.it.pasteur.fr/, accessed July 6th, 2014.

[75] http://www.meteo.unican.es/ganglia/, accessed July 6th, 2014.

[76] https://ganglia.surfsara.nl/, accessed July 6th, 2014.

[77] https://dev.monitor.orchestra.med.harvard.edu/, accessed July 6th, 2014.

[78] Zhang, Y., Roughan, M., Duffield, N., Greenberg, A. Fast accurate computation of large-scale IP traffic matrices from link loads. *Proceedings of ACM SIGMETRICS Performance Evaluation Review*, pp. 206-217, 2012.

[79] IBM Power 710 Express and IBM Power 730 Express (8231-E2B), *ENERGY STAR Power and Performance Data Sheet*.

[80] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, pp. 755-768, 2012.