Prediction-Based Joint Energy Optimization for Virtualized Data Centers

Motassem Al-Tarazi
Computer Science and Engineering Department
University of Nebraska-Lincoln
Lincoln, NE, USA
altarazi@unl.edu

J. Morris Chang
Electrical Engineering Department
University of South Florida
Tampa, FL, USA
chang5@usf.edu

ABSTRACT

Today's data centers tend to have tens to hundreds of thousands of servers to provide massive and sophisticated services. Statistically, data center and data center networks (DCNs) remain highly under-utilized which can be exploited for energy-saving. In this paper, we have studied energy-saving problem for network and server sides of the data center. The problem was formulated as a Mixed Integer Linear Program (MILP) that is solvable by an optimization software to jointly minimize the energy consumed by the servers and DCN. To overcome the optimization software high computational time, a heuristic algorithm to provide practical and efficient solution is introduced. The heuristic algorithm has two stages: first, it uses the virtual machines (VM) and the predicted servers resource utilization to provide VM consolidation algorithm and turn-off unused servers. The second stage uses an abstract performance-aware network flow consolidation that focused the traffic on subset of the network and turn-off the unused network devices. Simulation experiments using CloudsimSDN were conducted to validate the heuristic using real traces from Wikipedia in terms of energy consumption and average response time. The results show that the heuristic can save servers and network energy while maintaining performance.

CCS CONCEPTS

• **Networks → Data center networks**; • **Hardware → Power estimation and optimization**; • **Computer systems organization → Cloud computing**.

KEYWORDS

Data centers, Energy saving, Optimization, Prediction

ACM Reference Format:
Motassem Al-Tarazi and J. Morris Chang. 2020. Prediction-Based Joint Energy Optimization for Virtualized Data Centers. In *2020 ACM Southeast Conference (ACMSE 2020), April 2–4, 2020, Tampa, FL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3374135.3385279

ACMSE 2020, April 2–4, 2020, Tampa, FL, USA

ACM ISBN 978-1-4503-7105-6/20/03…$15.00
https://doi.org/10.1145/3374135.3385279

1 INTRODUCTION

In the era of cloud computing, data centers are growing in size leading toward the rise of large-scale data centers. One of the major concerns in cloud computing is the huge electricity consumption in the cloud data centers. According to the United States Environmental and Protection Agency (EPA) [16], the total electricity used by data centers in 2010 was about 1.3% of the all consumed electricity in the world and expected to reach 8% by 2020. Another report estimates the annual total energy costs of data centers in US alone to reach $13.7 billion by 2020 [8]. The major energy consuming categories in almost any data center includes servers, cooling systems, and data centers networks (DCNS) [15][12]. The energy consumption percentage for each category can be estimated as follows: servers (40-60%), cooling systems (15-30%), and DCNs (5-15%). Note that this percentage breakdown can change from a data center to another.

The architectural design of data centers is usually built to handle worst-case workload scenarios, which results in low average utilization for servers and rarely reaches its peak power. For example, Fan et al. [10] reported that, over the course of six months, a group of 5,000 servers under study at Google never exceeded 72% of their aggregate peak power. Due to this low utilization, server consolidation techniques have been proposed to increase server utilization and reduce energy consumption by putting unused servers to standby mode.

Similarly, the design of DCNs accommodates peak loads in most reliable way without taking energy saving into consideration. Data center networks are built with many redundant links and heavily over-provisioned link bandwidth to handle link failures and traffic bursts. Although current data center network design increases reliability, it also decreases energy efficiency since all network devices are powered-on all the time with minimal link utilization. Statistics showed that most of the network devices are under-utilized, where the typical utilization of a DCN is only 30% [17]. DCNs' over-provisioning and under-utilization can be exploited for energy saving research. Routes consolidation techniques are proposed to turn the network load to a minimal subset of network devices. Then it puts unused devices to sleep mode or shut them down to minimize the overall network power consumption. Most research efforts focus on power consumption of server within data center and power consumption of data center networks separately. Considering the power consumption of servers using server consolidation without taking DCN into account ignores the effects of virtual machine migration on DCN and increases the chances of traffic congestion which leads to network performance degradation. On the other

hand, considering DCN power consumption alone using routes consolidation ignores that network traffic might be affected by other events such as virtual machine migration.

In this paper, we studied the problem of saving servers and network energy consumption in virtualized data centers while maintaining their performance. We formulate the problem as a joint mix integer linear program to minimize the total servers and network energy as main objective. Moreover, the problem was constrained by network performance requirements, such as maximum link utilization and safety margin threshold for network links and servers resources.

The joint optimization is part of a framework that monitors the state of the data center by collecting and predicting run time utilization data for servers resources (CPU, memory, network, and disk) and network traffic. It uses them as an input for the joint optimization. The joint objective optimization will provide a new virtual machines placement and flow routing matrix that assure maximum data center energy saving while maintaining performance. Live migration commands will take place to adjust the placement of the virtual machines into their designated destinations based on the optimization solution. Finally, unused servers are moved to standby mode and unused switches links are turned off.

For large-scale data centers, the running time for the joint optimization, which is solvable via optimization software, is computationally inefficient. Thus, a heuristic algorithm for saving data center and network energy is proposed. The proposed heuristic has two stages; first at the servers side, the virtual machines initial placed using First Fit Decreasing. After that, the heuristic uses the resource predictions to solve resource utilization violations and save more energy. The second stage includes the use of an abstract performance-aware flow routing and consolidation to save network energy.

The proposed heuristic was evaluated using CloudsimSDN simulator using traces collected from Wikipedia page view statistic [1]. The results show that the proposed algorithm can save significant amount of energy in both servers and network sides while maintaining performance represented by average response time.

The rest of the paper is organized as follows. Section 2 reviews previous related works. Section 3 introduces the proposed system framework. Section 4 shows the prediction model used. Section 5 formulates the joint power saving problem. Section 6 shows the proposed heuristic algorithm. Section 7 presents the simulation results and Section 8 concludes the paper.

2 RELATED WORKS

Many approaches have been proposed to deal with the data center server-side energy saving using servers virtualization and virtual machines consolidation as servers are the most energy consuming devices, thus, providing great opportunity for energy saving [28][30][27][2][4]. For example, [2] propose a multi-objective optimization that consolidate the virtual machines into subset of servers taking into account the effect of virtual machine migration on network links. They propose a two stage heuristic algorithm where the first stage finds an initial feasible placement and the second stage triggered periodically to try to consolidate virtual machines into smaller set of servers to save more energy.

Other researchers focus on saving energy of the data center network (DCNs). They found optimization problems for current DCNs and propose different techniques and heuristics to solve them. The main idea is route consolidation, which try to switch the network traffic to a subset of switches and turn off unused devices. Many approaches use such technique such as ElasticTree [13], Carpo [26], REsPoNse [23], GreenTE [29], Merge network [6], and many others [20, 21, 24, 25].

ElasticTree [13] proposed a power manager that adjusts the active switches and links to satisfy dynamic traffic loads. Carpo [26] introduced a correlation-aware power optimization algorithm, it dynamically consolidates traffic loads into a minimal set of switches and links and shut down unused devices. REsPoNse [23] discussed the trade-off between optimal energy saving and scalability. It identifies a few critical routes offline, installs them to routing tables, then runs an online simple scalable traffic engineering to activate and deactivate network devices.

Recently, many researchers start to consider energy saving joint optimization for servers and DCNs. [14] studied the network routing and VM placement problem jointly to minimize traffic cost in DCN. They propose an online algorithm based on Markov approximation to find a near optimal solution within a feasible time.

VMPlanner [11] optimizes VM placement and network routing. They group VMs with high mutual traffic and assign them to the same rack. After that, traffic flows within a rack are consolidated to turn off unused switches.

PowerNets [31] considers finding optimal VM placement considering both servers and network resources and the correlation between VMs. They calculates the correlation coefficient between traffic flows and applied them for VM and traffic consolidation.

In this work, our proposed algorithm uses resource predictions for CPU, Mem, Dist,and network resources, jointly optimizing both servers and network sides of the data center.

3 SYSTEM MODEL

Figure 1 illustrates the major modules of the proposed framework: Resources measurements and prediction module, Joint Optimization module, and the Next placement and Power module. The resources

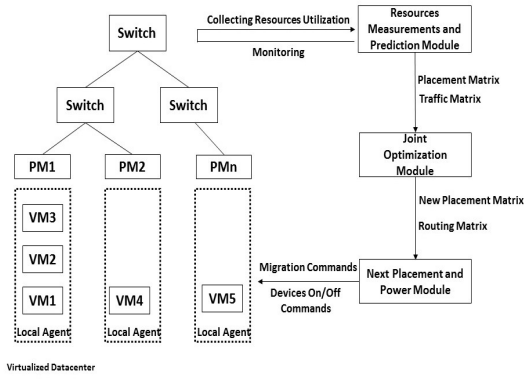

Figure 1: Proposed System Framework

measurements and prediction module is responsible for the continuous monitoring of the data center, collecting virtual machines and hosts resources utilization (CPU, memory, network, and disk) and getting predictions of future use for these resources from local agents. Note that local agents at each host calculates the resource utilization predictions for the host and each virtual machine. Furthermore, the Resources measurements and prediction module also extracts the network current traffic matrix.

The joint optimization module is a mix integer linear program (MILP) that takes the collected virtual machines and hosts resource utilization and predictions and the network traffic matrix as an input. The optimization will provide a solution that minimizes the energy consumed by the data center while maintaining performance. The results of this module are the new placement matrix, the flows routing matrix, and the devices and links power matrices.

The last module is the next placement and power module, which is responsible for sending live migration commands based on the solution provided by the optimization. After all live migration commands completed successfully, the module will turn unused servers to standby mode and turn the unused switches and links off.

4 RESOURCE MEASUREMENT AND PREDICTION

The first module in our proposed framework is the resource measurement and prediction module. It periodically collects hosts and virtual machines resources utilization as well as virtual machines predictions from local agents. Using these information, the module calculates hosts resource utilization predictions using linear regression for all resource types (CPU, Mem, Disk, and Net). Likewise, the local agent at each physical host collects virtual machine resource utilization and calculate prediction using linear regression.

Linear regression is a popular statistical approach to estimate the relationship between one or more input and one output. Linear regression approximate the regression function which represents a straight line. The regression function for the linear regression can be expressed as:

$$y = \beta_0 + \beta_1 x \tag{1}$$

Where β_0 and β_1 are the regression coefficient. They indicate the goodness of the fit and how well it predicts the output of y. The popular least square method is used to minimize the residuals.

The resource measurement and prediction module will categorize the hosts and virtual machines into one of the following: Overloaded, predicted to be overloaded, normal, predicted to be underloaded, and underloaded. These information and the resource utilization will be used as an input for the optimization and the heuristic algorithms to make better decisions.

Although there are many prediction models that are more complicated and might provide a much accurate resource predictions, using them in frameworks as our proposed framework is inefficient as they need more computational time and power to provide predictions.

5 PROBLEM FORMULATION

Consider a data center $G = (P \cup S, E)$ where P is the set of hosts, S is the set of switches and E is the set of links that connect switches and hosts and switches together. Each link $(i, j) \in E$ has a maximum capacity denoted by $C_{i,j}$, where C is the bandwidth capacity matrix.

The power consumed by a single physical machine follows the model proposed by [18][7], expressed by equation 2. The model shows that the server average power is approximately linear with respect to CPU utilization. The model has been proven to be accurate for large scale data centers.

$$\epsilon^{server} = \epsilon^{Pidle} + \epsilon^{Pdynamic} \cdot Util^{CPU} \tag{2}$$

Where ϵ^{server} is the total power consumed by the server, ϵ^{Pidle} is the power consumption when the CPU is idle, $\epsilon^{Pdynamic}$ is the dynamic power consumption of the CPU, and $Util^{CPU}$ is the average CPU utilization. A server $p \in PM$ can provide a set of resources bounded by an upper utilization threshold (U_p^{CPU}, U_p^{Mem}, U_p^{Net}, and U_p^{Disk}). Let VM be the set of all virtual machines, and $v \in VM$ must be hosted by a physical machine p, denoted by $H(p, v) = 1$. Each virtual machine v requests a specific amount of resources (denoted by vm_v^{CPU}, vm_v^{Mem}, vm_v^{Net}, and vm_v^{Disk}) to be consumed. A physical machine can host many virtual machines as long as its resource utilization thresholds are not violated. Note that in this paper we use host, physical machine, and server interchangeably.

A port link can be turned-off if there is no traffic on the link, and a switch can be turned-off if all its ports are turned-off. The power consumed by a single switch $s \in S$ consists of fixed power ϵ^{Sidle}, which consumed by components like (chassis, fans, etc), and ports power ϵ^{Port}. The power saving gained from turning off a single port is ϵ^{Port}, and from turning off an entire switch is $\epsilon^{Sidle} + \sum_{l \in N_i} \epsilon^{Port}$.

We use $On(i)$ and $On(i, j)$ as decision variables to denote that switch (or a server) i and link (i, j) are active or not. Assuming F is the set of all flows in DCN. A flow $f \in F$ consists of a source node $src(f)$, a destination node $dest(f)$, and the required bandwidth $bw(f)$. $route(f, (i, j)) = 1$ denotes that a flow f is using link (i, j), where $(i, j) \in E$.

With the notations summarized in Table I, we can formulate our problem as a mixed integer linear program that is solvable by an optimization software as the following: the *MILP* takes the data center $G = (P \cup S, E)$, utilization thresholds for links and host resources U_p^c, virtual machines current resource utilizations vm_v^c, virtual machines predicted resource utilizations $vm_v^{\hat{c}}$, the set of all flows F, the capacity matrix C, and the power specifications for the physical machines, virtual machines, and switches.

Equation 3 is the objective function. It minimizes the data center power consumption functions for hosts and switches.

$$Minimize \quad \sum_{p \in P} Hpow(p) + \sum_{s \in S} Spow(s) \tag{3}$$

The constraints are divided into three categories: servers constraints, network constraints, and links constraints. Equations 4-5 present servers power constraints, they state that a physical machine operates in active mode if and only if it needs to serve an active virtual machine. Specifically, equation 4 assures that a physical machine can be turned into standby mode only if there is no virtual machine placed on it. Likewise, equation 5 shows that a

Table 1: Definition of Important Symbols

Symbol	Definition
S	Set of all switches
P	Sets of all hosts
N	Set of all ports
E	Set of all links
D	Set of all traffic Demands
C	Capacity Matrix for all links
VM	Set of all virtual machines
N_i	Ports in switch i
i, j	A link connects two nodes i and j
$\epsilon^{Sidle}, \epsilon^{port}$	Energy consumed by an idle switch and port
$\epsilon^{Pidle}, \epsilon^{Pstandby}$	Energy consumed by a host in idle and standby modes
ϵ^{vm}	Energy consumed by a virtual machine
vm_v^c	Utilization of resource c by VM v
U_p^c	Threshold of resource c at host p
$On(i, j)$	A Link (i, j) is on or off
$On(j)$	A Switch or host (j) is on or off
$H(p, v)$	A VM v hosted by a host p
$route(f, (i, j))$	Flow f is routed using link (i, j)
$Hpow(*), Spow(*)$	Power consumed by host/switch
$u_{i,j}$	Utilization of link i, j

virtual machine can only be placed on an active physical machine.

$$On(p) \leq \sum_{v \in VM} H(p, v), \quad \forall p \in P \tag{4}$$

$$H(p, v) \leq On(p), \quad \forall v \in VM, p \in P \tag{5}$$

The power consumed by a physical machine is calculated in equation 6. It is based on the status of the physical machine and the virtual machines hosted on that machine.

$$Hpow(p) = \epsilon^{Pidle} On(p) + \epsilon^{standby}(1 - On(p)) + \epsilon^{vm} \sum_{v \in VM} H(p, v) vm_v^{CPU} \tag{6}$$

Equations 7 - 8 shows the placement constraints, they assure the correct placement of virtual machines on their designated physical machines. Equation 7 states that each virtual machine has to be and can only be served by one physical machine. Let $DH(v)$ be the potential destination hosts for a virtual machine v. Equation 8 ensures that a virtual machine can only be hosted by one of its potential destination hosts. The selection of the potential destination hosts for a virtual machine v is governed by the actual and predicted resources required by the virtual machine, the availability of these resources on the host, the host resource prediction (i.e. if the host is predicted to be overloaded or under-loaded), and the migration cost from the source to the destination node.

$$\sum_{p \in P} H(p, v) = 1, \quad \forall v \in VM \tag{7}$$

$$\sum_{p \in DH(v)} H(p, v) = 1, \sum_{p \in P \setminus DH(v)} H(p, v) = 0, \quad \forall v \in VM \tag{8}$$

Equations 9-10 introduce the actual and predicted resources utilization constraints. Since virtual machines deployed on a physical machine require some amount of resources, these resources should not exceed a specific utilization level from the resources offered by the physical machines (unless indicated, we consider the utilization threshold = 70%). In this formulation, c is the resource type. Note that all types of resources are considered (CPU, memory, network, and disk).

$$\sum_{v \in VM} (H(p, v) \times vm_v^c) \leq U_p^c, \quad \forall p \in P \tag{9}$$

$$\sum_{v \in VM} (H(p, v) \times vm_v^{\hat{c}}) \leq U_p^c, \quad \forall p \in P \tag{10}$$

Equation 11 calculates the power consumed by a switch. Which is the power consumed by its fixed components ϵ^{Sidle}, such as chassis, fans, line cards, ... etc., in addition to the power consumed by each active port ϵ^{Port}.

$$Spow(s) = \epsilon^{Sidle} \cdot On(s) + \sum_{n \in N_i} \epsilon^{Port} \cdot On(s, n) \tag{11}$$

Equations 12-13 present the flow constraints. Equation 12 states that a flow should always starts/ends at the host that contains the source/destination virtual machine. Equation 13 ensure that the virtual machines will use local bus if they were placed on the same server, otherwise the transmission should start at the server that hosts the source vm and ends at the server that hosts the destination vm.

$$\sum_{s \in S} (route(f, (p, s))) \leq H(p, src(f)), \quad \sum_{s \in S} (route(f, (s, p))) \leq H(p, dest(f)), \quad \forall p \in P, \forall f \in F \tag{12}$$

$$H(p, src(f)) - H(p, dest(f)) = \sum_{s \in S} (route(f, (p, s))) - \sum_{s \in S} (route(f, (s, p))), \quad \forall p \in P, \forall f \in F \tag{13}$$

Equations 14-17 show the links constraints. Equation 14 introduces the active link constraint. It states that an active link connects two active switches or a switch and a server.

$$On(i, j) \leq On(i), On(i, j) \leq On(j), \forall i, j \in E, \forall i \forall j \in S \tag{14}$$

Equation 15 states the bidirectional link power constraint which means both directions of a link (i, j) should have the same on/off

power status. Likewise, equation 16 ensures that for every active link $On(i,j) = 1$, both directions have the same capacity limits $C_{i,j}$.

$$On(i,j) = On(j,i)\,, \forall i,j \in E \tag{15}$$

$$On(i,j) \cdot C_{i,j} = On(i,j) \cdot C_{j,i}\,, \forall i,j \in E \tag{16}$$

Equation 17 introduces the satisfiability constraint. It shows that the summation of all traffic flow loads passing through link (i,j) is always less than or equal to the capacity limit of that link $C_{i,j}$.

$$\sum_{f \in F} (route(f,(i,j)) \cdot bw(f)) \leq On(i,j) \cdot C_{i,j}\,, \forall i,j \in E \tag{17}$$

Equation 18 shows the active switch constraint. Let $N_i \in N$ be the set of ports in a switch and $|N_i|$ is the cardinality of N_i, then equation 18 ensures that a switch will be turned off only if all its ports are turned off.

$$|N_i| \cdot (1 - On(i)) \leq \sum_{j \in N_i} (1 - On(i,j))\,, \forall i,j \in E, \forall i \in S \tag{18}$$

Equations 19-20 present utilization constraints. Equation 19 calculates the link utilization u for each link. Where link utilization is the summation of every traffic flow load passing link (i,j) to the capacity of that link. Equation 20 ensures that the utilization of every link is always less than or equal to a predefined upper link utilization threshold U^{upper}(unless indicated, we consider $U^{upper} = 0.80$).

$$u_{i,j} = \frac{\sum_{f \in F} (route(f,(i,j)) \cdot bw(f))}{C_{i,j}}\,, \forall i,j \in E \tag{19}$$

$$u_{i,j} \leq U^{upper}\,, \forall i,j \in E \tag{20}$$

Since mixed integer linear programming is NP-hard, the proposed formulation is not practical for large data centers.

6 HEURISTIC ALGORITHM

To overcome the exponential increase in the optimization software computation time, a heuristic algorithm solving the data center energy-saving problem was developed. In data center environment, traffic demands fluctuate frequently. For that reason, heuristic algorithm is preferred to solve our optimization model in real time. The algorithm takes input similar to the *MILP* and is divided into two main stages: virtual machine placement and consolidation, and network flow routing and consolidation.

The virtual machine placement and consolidation stage consists of two parts: initial virtual machine placement and dynamic virtual machine consolidation. The initial virtual machine placement stage targets finding a virtual machine placement on the hosts such that no resource violation occurs. Since no historical data available, no virtual machine nor host prediction can occur. The algorithm places the virtual machine using First Fit Decreasing (FFD) method, which is one of the most efficient algorithms to solve the bin-packing problem [9]. However, due to workload fluctuations, resource demands changes over time. So, the initial placement will not be efficient anymore and there is a need for a virtual machines consolidation to update the current placement and provide a more optimized solution which is presented in the second stage.

Algorithm 1 and 2 show the pseudocode of the proposed virtual machine consolidation for both overloaded and underloaded hosts. The algorithm runs periodically to solve any resource utilization constraint violation (Algorithm 1) and saving more energy by migrating *vms* from under utilized hosts to put them into standby mode (Algorithm 2).

The algorithm starts by categorizing the active hosts into three sets: OverloadedHosts, UnderloadedHost, and NormalHosts. The OverloadedHosts set consists of all hosts that violates one of its resource threshold constraint. UnderloadedHost is the set of hosts that none of its resource utilization exceeds 10% from the past execution. NormalHosts is the set of active hosts that are not overloaded nor underloaded and is not predicted to become overloaded in the near future. Algorithm 1 deals with each of the overloaded hosts by sorting their virtual machines in descending order of their total utilized resources. Starting with the virtual machine with the highest total utilized resources, the algorithm try's to find a target host from the set of normal hosts. To reduce the search space within the set of normal hosts, the virtual machine should be migrated to a normal host within its rack or pod. Furthermore, it should be assured that the migration will not cause the target server to become overloaded or predicted overloaded. If found, the virtual machine will be migrated to the target host. If the migration of the virtual machine solves the host resource violation (i.e. the host not overload any more), the host will be removed from the overloadHost list and it will be added to one of the other two lists. If the migration did not solve the resource violation, the process will be repeated with the second highest virtual machine and so on. If no more hosts in the normal hosts set able to handle the overloaded hosts virtual machines, the same procedure will be done using the set of underloadhosts. Finally, if no normal nor underloaded hosts able to handle hosting virtual machines from overloaded hosts, a new host will be turned on and the virtual machines will migrate to it.

Algorithm 2 aims to save more energy by migrating the virtual machines from underloaded hosts to put them into standby mode. Similar to the algorithm 1, it categorizes the hosts into the same three categories. Then, it sorts the hosts in ascending order of their resource utilization. For the host with the lowest required resource, the algorithm tries to find a target host or a set of hosts that can satisfy hosts virtual machines. The target host(s) should be: within the source host rack or pod to reduce the migration cost on the network, it's in the set of normal hosts, and the migration will not move the target host to be overloaded or predicted overloaded. If a target host(s) is found, all *vms* from the source host will be migrated and the host will be set into standby mode. If there is no more normal host available and there are more than 1 host in the set of UnderloadHosts, the algorithm searches for a target hosts within UnderloadHosts set. The migration conditions are similar to migration to normal hosts. The computational cost for this algorithm is $O(P.VM^2.\log VM)$.

For the network flow routing and consolidation stage we use our previous work [3], which is an abstract model that saves data center network energy while maintaining the network performance from traffic surges. we proposed a light-weight heuristic algorithm that combines setting-up safety margin threshold and load balancing technique together to save energy and maintain network performance to handle traffic surges.

Algorithm 1 VM Consolidation - Overloaded Hosts

Input: $P, VM, Hpred, VMpred, res(P), res(VM), On(p)$
$OverloadHosts = res(P) \mid$ "overloaded"
$UnderloadHosts = res(P) \mid$ "underloaded"
$NormalHosts = res(P) \mid$ "normal"
if $OverloadHosts = \phi$ **then**
    break
for $p \in OverloadHosts$ **do**
    Sort $vm \in p$ in descending order based on total utilized resources
    for $vm \in p$ **do**
        if $FindTargetHost(vm, Normal)$ **then**
            Migrate vm
            if $p \notin OverloadHosts$ **then**
                $OverloadHosts = OverloadHosts - p$
                next p
    for $vm \in p$ **do**
        if $FindTargetHost(vm, Underload)$ **then**
            Migrate vm
            if $p \notin OverloadHosts$ **then**
                $OverloadHosts = OverloadHosts - p$
                next p
    if $\exists p_{new} \in standbymode$ **then**
        Turn on p_{new}
        Migrate vm to p_{new}

Algorithm 2 VM Consolidation - Underloaded Hosts

Input: $P, VM, Hpred, VMpred, res(P), res(VM), On(p)$
$OverloadHosts = res(P) \mid$ "overloaded"
$UnderloadHosts = res(P) \mid$ "underloaded"
$NormalHosts = res(P) \mid$ "normal"
if $UnderloadHosts = \phi$ **then**
    break
$\forall p \in UnderloadHosts$ Sort them in ascending order of their total utilized resources
for lowest $p \in UnderloadHosts$ **do**
    if $FindTargetHosts(p, Normal) \neq false$ **then**
        Migrate all $vms \in p$
        Put p into standby mode
    else
        Continue
if $|UnderloadHosts| > 1$ **then**
    for lowest $p \in UnderloadHosts$ **do**
        $FindTargetHosts(p, UnderLoad)$

The heuristic algorithm starts by setting up predefined safety thresholds on each link capacity. Then, it continuously monitors the utilization of network links and balances the loads on active links using Valiant Load Balancing (VLB) mechanism [19]. A decision to turn on new switches or links can be taken if these thresholds are exceeded. Using this algorithm, the safety margins and the load balancing mechanism allow the network to handle traffic surges, while maintaining its performance. On the other hand, switches grouping and links consolidation will also take place if the loads on the networks switches and links are under-utilized. This will allow turning off some active ports and switches to lower network power consumption.

7 PERFORMANCE EVALUATION

This section presents the evaluation of our proposed heuristic algorithm. The evaluation is conducted to show that the algorithm achieves a considerable amount of energy saving while maintaining performance.

We compared the proposed heuristic to other algorithms including Most Full First (MFF) and Least Full First (LFF) bin packing algorithms for virtual machine placement. Most Full First (MFF) will assign a virtual machine to the most full host that can satisfy the virtual machine resource demands. On the other hand, Least Full First (LFF) assigns a virtual machine to the least full host. No virtual machine migration is implemented on these methods. Another method [2] that uses virtual machine consolidation to save energy and minimize the cost of live migration on the network. This method does not provide energy saving for network devices so we will refer to it as no network.

7.1 Simulation Setup

The proposed heuristic was implemented in CloudSimSDN [22]. It is an extension of the popular CloudSim simulator [5] that supports different software defined networks features. We consider a fat-tree data center with $k = 8$. The data center includes a 16 core switches, 32 aggregate switches, 32 access switches, and 128 hosts. Each host is equipped with 8 core CPU. Figure 2 shows Fat tree network topology with $k = 4$.

Cloud data center workloads fluctuate frequently. In this evaluation, we investigate the data center traces provided by Wikipedia data center, which is available for public. Specifically, we investigate the statistical Page view data for selected Wikimedia projects for one day [1]. For each hour, the traces consists of the page view count and the amount of bytes transmitted as a respond. Figure 3 and 4 shows the how number of requests and the bytes transmitted as a response varies each hour. For the experiments that include migration, the monitoring interval is set to 2 minutes to collect

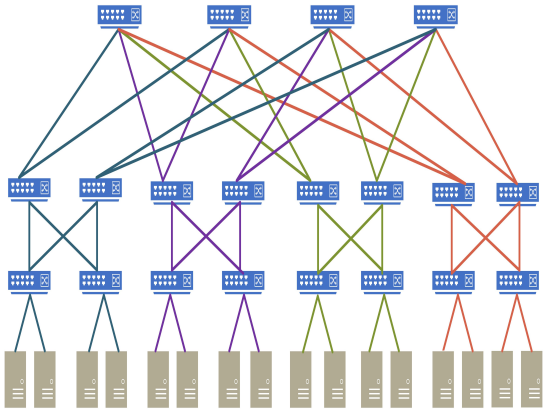

Figure 2: Fat Tree Network Topology with $k = 4$

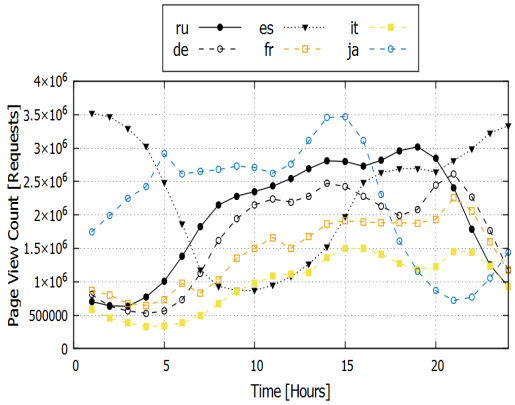

Figure 3: Wikipedia One Day Traces: Page View Requests

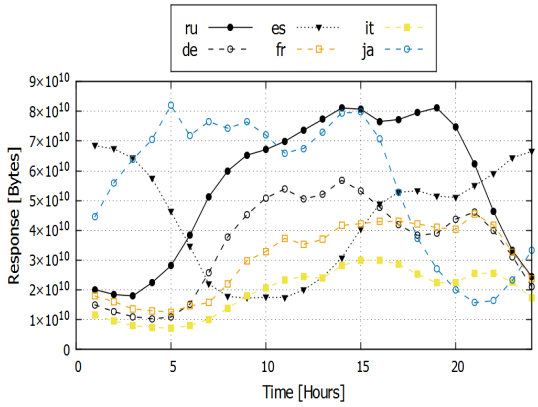

Figure 4: Wikipedia One Day Traces: Response Size in Bytes

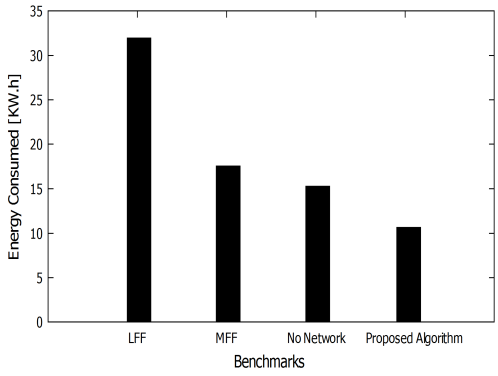

Figure 5: Servers Energy Consumption

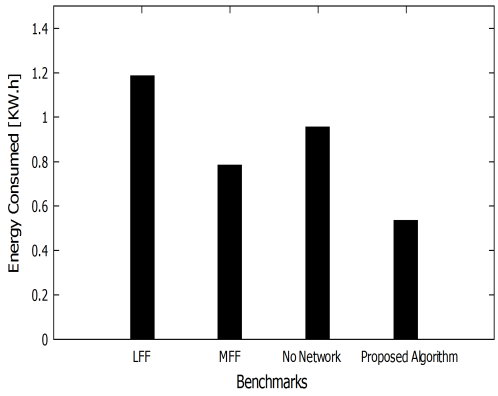

Figure 6: Switches Energy Consumption

the utilization of VMs, hosts, flows, and links. The migration is attempted every 20 minutes. The overall simulation time is 2 hours.

7.2 Simulation Results

In this section we present the simulation results to compare the proposed algorithm against MFF, LFF, and no network in terms of servers, switches, and total energy consumption and network time. Figure 5 shows the energy consumed by the servers. It can be seen that the proposed algorithm uses least energy compared to the other method. This is due to the quality of virtual machine consolidation process. The proposed algorithm uses hosts and virtual machines resource prediction which provide more information for better placement of the virtual machines. The no network uses only the current status of the resource utilization without any prediction. Both MFF and LFF do have any virtual machines migration, thus, it can't cop with workload fluctuations. LFF spread the workload over all hosts resulting in the worst energy consumption. MFF uses the best fit initially so it saves energy. But as the workloads varies over time, MFF can't cop with this variation and misses opportunities for more energy saving.

Figure 6 shows the energy consumed by switches. The results shows that the proposed algorithm outperforms all other algorithms. The proposed algorithm uses flow consolidation to move the flows into subset of the network devices and turn off unused ones. The proposed algorithm consumes 0.534 KW.h compared to 0.9557,0.7836, and 1.185 for the no network, MFF, and LFF, respectively. Likewise, Figure 7 show the total energy consumed by the data center.

Finally, we compare the proposed algorithm against other algorithms in terms on average response time. Figure 8 shows the average response time for all algorithms. The results state that the proposed algorithm has the best average response time. Specifically, the proposed algorithm has average response time of 1.248 seconds compared to 1.694,2.448, and 2.9296 for no network, MFF, and LFF, respectively.

8 CONCLUSION

The architectural design of data centers can be exploited for energy saving. Most research on literature focuses on optimizing energy saving for the servers and network separately. In this paper, we propose a joint optimization for minimizing the server-side and network side of the data center. The optimization is part of a framework that collect and predict servers and virtual machines resource

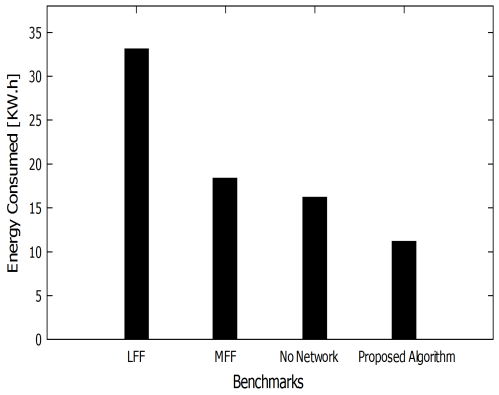

Figure 7: Total Energy Consumption

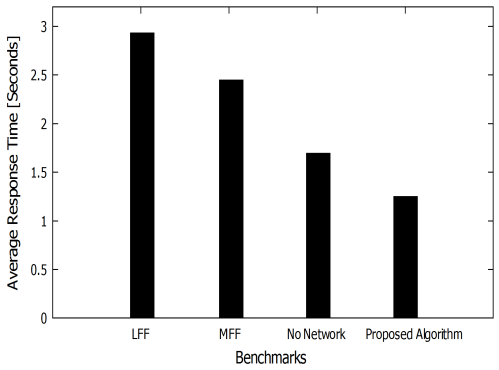

Figure 8: Average Response Time

utilization which will be used as an input for the joint optimization. The results of the joint optimization includes the new placement of virtual machines. For practical implementation on large scale data centers, a two stage heuristic algorithm is proposed. The heuristic first find near optimal solution for the server side then it uses an abstract performance model for saving network energy. The proposed algorithm was evaluated using CloudsimSDN with real Wikipedia traces. The results show that proposed algorithm saves servers and network energy while maintaining performance.

REFERENCES

[1] [n.d.]. Wikimedia Pageveiw. https://stats.wikimedia.org/EN/ Accessed: 2019-02-01.

[2] M. Al-Tarazi and J.M. Chang. 2018. Network-Aware Energy Saving Multi-Objective Optimization in Virtualized Data Centers. *Cluster Computing* (2018), 1–13.

[3] M. Al-Tarazi and J.M. Chang. 2019. Performance-Aware Energy Saving for Data Center Networks. *IEEE Transactions on Network and Service Management* 16, 1 (2019), 206–219.

[4] N. Bobroff, A. Kochut, and K. Beaty. 2007. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*. IEEE, 119–128.

[5] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.De Rose, and R. Buyya. 2011. CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

[6] A. Carrega, S. Singh, R. Bolla, and R. Bruschi. 2012. Applying Traffic Merging to Datacenter Networks. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*. 3.

[7] Y. Chou, B. Fahs, and S. Abraham. 2004. Microarchitecture Optimizations for Exploiting Memory-Level Parallelism. In *ACM SIGARCH Computer Architecture News*, Vol. 32. IEEE Computer Society, 76.

[8] P. Delforge. 2015. America's Data Centers Consuming and Wasting Growing Amounts of Energy. http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[9] G. Dósa. 2007. The Tight Bound of First Fit Decreasing Bin-Packing Algorithm is FFD (I)- 11/9OPT (I)+ 6/9. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 1–11.

[10] X. Fan, W. Weber, and L.A. Barroso. 2007. Power Provisioning for a Warehouse-Sized Computer. In *ACM SIGARCH computer architecture news*, Vol. 35. ACM, 13–23.

[11] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong. 2013. VMPlanner: Optimizing Virtual Machine Placement and Traffic Flow Routing to Reduce Network Power Costs in Cloud Data Centers. *Computer Networks* 57, 1 (2013), 179–196.

[12] A. Greenberg, J. Hamilton, D.A. Maltz, and P. Patel. 2008. The Cost of a Cloud: Research Problems in Data Center Networks. *ACM SIGCOMM computer communication review* 39, 1 (2008), 68–73.

[13] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. [n.d.]. ElasticTree: Saving Energy in Data Center Networks. In *NSDI*, Vol. 10. 249–264.

[14] J.W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. 2012. Joint VM Placement and Routing for Data Center Traffic Engineering. In *2012 Proceedings IEEE INFOCOM*. IEEE, 2876–2880.

[15] J.G. Koomey. 2008. Worldwide Electricity Used in Data Centers. *Environmental research letters* 3, 3 (2008), 034008.

[16] J. Koomey. 2011. Growth in Data Center Electricity Use 2005 to 2010. (2011).

[17] J. Liu, F. Zhao, X. Liu, and W. He. [n.d.]. Challenges Towards Elastic Power Management in Internet Data Centers. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*. IEEE, 65–72.

[18] D. Meisner and T.F. Wenisch. 2010. Peak Power Modeling for Data Center Servers with Switched-Mode Power Supplies. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*. ACM, 319–324.

[19] W. Ni, C. Huang, and J. Wub. 2014. Provisioning High-Availability Datacenter Networks for Full Bandwidth Communication. *Computer Networks* 68 (2014), 71–94.

[20] Y. Shang, D. Li, and M. Xu. [n.d.]. Energy-Aware Routing in Data Center Network. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*. ACM, 1–8.

[21] Y. Shang, D. Li, and M. Xu. 2013. Greening Data Center Networks with Flow Preemption and Energy-Aware Routing. In *Local & Metropolitan Area Networks (LANMAN), 2013 19th IEEE Workshop on*. IEEE, 1–6.

[22] J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, and R. Buyya. 2015. Cloudsimsdn: Modeling and Simulation of Software-Defined Cloud Data Centers. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 475–484.

[23] N. Vasi, P. Bhurat, D. Novakovi, M. Canini, S. Shekhar, and D. Kosti. [n.d.]. Identifying and Using Energy-Critical Paths. In *Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies*. ACM, 18.

[24] L. Wang, F. Zhang, C. Hou, J.A. Aroca, and Z. Liu. 2013. Incorporating Rate Adaptation into Green Networking for Future Data Centers. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*. IEEE, 106–109.

[25] T. Wang, Y. Xia, J. Muppala, and M. Hamdi. 2015. Achieving Energy Efficiency in Data Centers Using an Artificial Intelligence Abstraction Model. *IEEE Transactions on Cloud Computing* PP, 99 (2015), 1–1. https://doi.org/10.1109/TCC.2015.2511720

[26] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. [n.d.]. Carpo: Correlation-Aware Power Optimization in Data Center Networks. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, 1125–1133.

[27] Y. Wang and X. Wang. 2010. Power Optimization with Performance Assurance for Multi-Tier Applications in Virtualized Data Centers. In *2010 39th International Conference on Parallel Processing Workshops*. IEEE, 512–519.

[28] T. Yang, Y.C. Lee, and A.Y. Zomaya. 2014. Energy-Efficient Data Center Networks Planning with Virtual Machine Placement and Traffic Configuration. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 284–291.

[29] M. Zhang, C. Yi, B. Liu, and B. Zhang. [n.d.]. GreenTE: Power-Aware Traffic Engineering. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. IEEE, 21–30.

[30] Z. Zhang, C. Hsu, and J.M. Chang. 2015. Cool Cloud: a Practical Dynamic Virtual Machine Placement Framework for Energy Aware Data Centers. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 758–765.

[31] K. Zheng, X. Wang, L. Li, and X. Wang. 2014. Joint Power Optimization of Data Center Network and Servers with Correlation Analysis. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2598–2606.