Power-Efficient Virtual Machine Replication in Data Centers

Payman Khani, Bin Tang, Jiaochao Han, and Mohsen Beheshti
Department of Computer Science, California State University Dominguez Hills, Carson, CA 90747, USA
Email: {pkhani,btang,jhan,mbeheshti}@csudh.edu

Abstract—**By replicating virtual machines (VMs) and placing replica copies into data centers, not only does it distribute the requests to virtual machines into different physical machines, thus reducing server load, but also it achieves fault tolerance in risks of server failures by placing multiple copies of a VM on different servers. This paper studies the virtual machine replication problem (VMR) in data centers, with the goal of minimizing the total power consumption in this process. To guarantee that each VM is available in the event of server failure, it replicates R copies of each VM and place them into different physical machines (PMs) in the data centers, where R depends upon the server failure probability. We show that VMR is equivalent to the minimum cost flow problem, which can be solved efficiently and optimally. We further reduce the power consumption in the data center by consolidating PMs that store VM replicas and turning off inactive ones. In addition, we design two time-efficient heuristic algorithms to solve VMR. Via extensive simulations and analysis, we compare the VM replication algorithms under different data center scenarios, and show that our consolidation algorithm could further consolidate 50 PMs upon the VM replication algorithm in a data center of 1028 PMs.**

Keywords – Power-Efficiency, Virtual Machine Replication, Server Consolidation, Data Center

I. **Background and Motivation**

Data centers consist of tens of thousands of server machines that support a large number of Internet services such as social networks, video streaming, and search engines [17]. Recently more and more data centers utilize server virtualization technologies for resource sharing as well as operating cost reduction. Virtualization technologies such as VMware, Xen and Microsoft Virtual Servers consolidate applications previously running on multiple physical servers into a single physical server by enabling multiple OS environments on the same physical machine. By doing so, some of the servers can be turned off and power consumption of data centers can be dramatically reduced. In particular, the hardware resources of a PM such as CPU cycles, memory, and bandwidth are divided into several smaller isolated computing units, known as virtual machines, which can be rented to tenants in a pay-as-you-go manner (e.g., Amazon EC2 and Microsoft Azure).

Failures are quite common in current data centers [8], [12]. There are various server, link, switch, rack failures due to hardware, software, and power outage problems, as well as human errors. As the size of the data center grows, individual server and switch failures may become the norm rather than exception in data centers. To achieve fault tolerance, redundancy of hardware and software is usually adopted. For example, virtual machine replication and placement in data centers has recently drawn attentions in research community (e.x., [10], [15]). By replicating virtual machines and placing replica copies into data center networks, not only does it distribute the requests to virtual machines into different physical machines, thus reducing server load, but also it achieves fault tolerance in risks of server failures by placing multiple copies of a VM on different servers. Since virtual machines depend on the physical devices and platforms to function, a failure of a hosting server becomes a serious problem in virtualization-based cloud computing data centers. Therefore virtual machine replication is critical to the smooth operation of data centers.

In addition, power consumption is also a big concern in any data center. Studies have shown that two largest chunks of power consumption in a data center go to equipment (including servers, storages, and network devices) and cooling, each consuming around 45% of the total power consumption in data center [7]. The network devices in data centers include various switches, routers, and links. They usually cost one third to half of power consumption of the servers and storages. Another study [1] shows that if the system is underutilized and the servers are fully energy-proportional (that is, consuming almost no power when idle and gradually consume more power as the activity level increases), then the network devices will consume nearly 50% of overall power in a data center.

However, most of the existing work of virtual machine replication only focus on the energy consumption on server machines, failing to consider the power consumption spent on network devices [10], [15] (please refer to Section II for more detailed discussion). In contrast, our work considers power consumptions on servers as well as switches inside the data center networks. In this paper, we propose to make redundant deployment of VMs in anticipation of hosting physical machine failures. In particular, we aim to create R copies of each VM and place the replica copies in the data center networks under the storage capacity constraint of PM, where no two copies of the same VM are stored at the same PM (for the purpose of fault tolerance). Our goal is to minimize the power consumption on the switches of the data center networks. We refer to this problem as *virtual machine replication problem (VMR)*. With intricate transformation of the data center network to a flow network, we show that VMR is equivalent to the minimum cost flow problem [2], [18], which can be solved optimally and efficiently.

With each VM having R copies in the data center, each copy stored at one of the PMs, we further reduce the power consumption in data center via *physical machine consolidation*. The goal of PM consolidation is to move the VM replica copies into smaller number of PMs, while still satisfying aforesaid constraints and maintaining the minimum cost flow power consumption. This way, we can turn off more PMs that do not store any VMs to save power consumption. The power consumption saved in this part is on the physical machines of the data center. We design power-efficient PM consolidation algorithm and show that it further reduces the power consumption in VMR. Finally, we design two power-efficient and time-efficient heuristic VM replication algorithms and compare all the algorithms via extensive simulations.

Paper Organization. The rest of the paper is organized as follows. Section II gives an overview of the related literature, and introduces the data center topology and the minimum cost flow problem. In Section III, we formulate the VMR problem. Section IV presents the different algorithms for VMR, including minimum cost flow-based optimal VM replication algorithm and two heuristic algorithms. We present the PM consolidation algorithm in Section V to further improve the results of VMR. In Section VI, we compare all the proposed algorithms, discuss the results in details, and present some insights. Section VII concludes the paper and discuss possible future work.

II. **Background**

Related Work. Power efficiency has become one of the most active topics in data center research. To our surprise, however, virtual machine replication for fault tolerance in data centers has not been actively researched. There are only a few publications that are reviewed below. Goudarzi and Pedram [10] presented an approach that first creates multiple copies of VMs and then uses dynamic programming and local search to place these copies on the physical servers in order to minimize the energy cost. Machida et al. [15] proposed a VM placement method to establish a redundant configuration against physical machine failures with less physical machines. In particular, they studied redundant VM placement that survives at any k physical machine failures while minimizing the number of physical machines. Both approaches only focused on the energy consumption on server machines, without considering the power consumption spent inside the data center networks such as on network links and switches. This paper considers energy consumptions on servers as well as in the data center network when placing replicated virtual machines.

VM placement has been an active research in recent years. Meng et al. [16] proposed using traffic-aware virtual machine placement to improve the network scalability, by assigning VMs with large mutual bandwidth usage to host machines in close proximity. Alicherry et al. [5] considered the problem of optimal placement of computational nodes and presented algorithms for assigning virtual machines to data nodes that minimize various latency metrics under different constraints. Li et al. [14] considered the cost caused by network traffics as well as the cost caused by the utilization of physical machines, and focused on the optimized placement of VMs to minimize the combination of both costs. However, none of the existing VM placement work addresses the issue of fault tolerance, which is critical in any large-scale data center. We tackle this challenge by replicating VMs inside data center in a power-efficient manner.

Minimum Cost Flow Problem. The minimum-cost flow problem (MCF) [2] is to find the cheapest possible way of sending a certain amount of flow through a flow network. In a flow network, some nodes are supply nodes and some are demand nodes, and the total supply equals the total demand. MCF is to send flows from supply nodes to demand nodes with minimum cost while the capacity constraint of each edge is satisfied. MCF can be solved efficiently and optimally [2]. It has been recently used to model data center network virtualization [11] as well as efficient resource allocation for MapReduce Applications in the Cloud [13]. We use minimum cost flow to model a totally different problem, which is to replicate VMs while incurring minimum energy consumption within data center networks. To the best of our knowledge, this is the first work that introduces minimum cost flow technique to solve VM replication problem in data center optimally.

Data Center Topology [4]. We focus on fat-tree networks [4] as they are widely adopted in data centers to interconnect commodity Ethernet switches. Fat tree is a variation of three-stage Clos networks [6], which is rearrangeably non-blocking with 1:1 oversubscription ratio [4]. *Rearrangeably non-blocking* means all the bandwidth available to the end hosts can always be saturated for any communication patterns. *Over-subscription* is the ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology. An oversubscription of 1:1 indicates that all hosts may potentially communicate with arbitrary other hosts at the full bandwidth of their network interface.

A k-ary fat-tree is shown in Fig. 1 with $k = 4$, where k is the number of ports of each switch. There are three layers of switches: edge switch, aggregation switch and core switch from bottom to top. Core switches handles huge amount of traffic across the entire data center, therefore consuming lots of energy power. In contrast, aggregate switches and edge switches transmit less amount of traffic therefore consume less power. The lower two layers are separated into k pods. *Pods* are modular units of compute, storage, and networking resources that are designed together as a unit in data center, each containing $k/2$ aggregation switches and $k/2$ edge switches, while forming a complete bipartite graph in between. In particular, each edge switch is directly connected to $k/2$ physical machines; and each of its remaining $k/2$ ports is connected to each of the $k/2$ aggregation switches from the same pod. There are $\frac{k^2}{4}$ k-port core switches, each of which is connected to each of k pods. In general, a fat-tree built with k-port switches supports $\frac{k^3}{4}$ physical machines. In the small

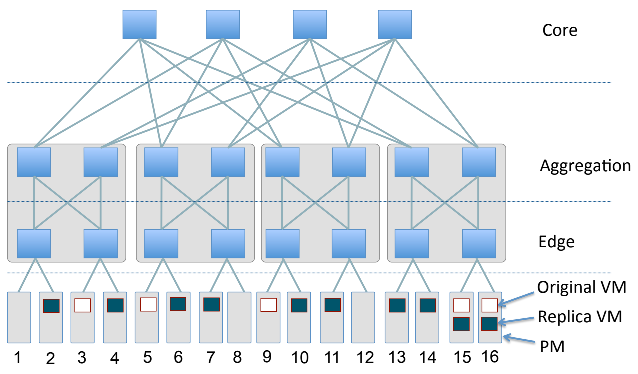

Fig. 1. A k-ary fat tree topology with $k = 4$ and 16 physical machines (PMs). There are $p = 5$ original virtual machines (VM): (v_1, v_2, ..., v_5) from left to right, located at PM 3, 5, 9 15, 16, respectively.

data center of Fig. 1, there are 16 physical machines.

One observation is that in data center network, switch power consumption varies. According to [7], a high-end switch such as Cisco Catalyst 6500 series consume 400 to 4,000 Watts and an edge switch such as Dell Power Connect series consumes about 100 Watts only. Accordingly, we consider that different level of switches cost different amount of power in our problem formulation and solution. Our cost model generalizes the cost model in [16], in which the cost definition is the number of switches on the path between two PMs.

III. Virtual Machine Replication Problem (VMR)

Problem Models. We model a data center as a graph $G(V, E)$, where $V = \{1, 2, ..., |V|\}$ is a set of $|V|$ PMs and switches, and E is a set of edges. Let V_s denote the set of PMs, V_e the set of edge switches, V_a the set of aggregate switches, and V_c the set of core switches. Thus, $V = V_s \cup V_e \cup V_a \cup V_c$. Without loss of generality, let $V_s = \{1, 2, ..., |V_s|\}$. Each edge represents a physical link that exists either between the switches or between switches and PMs.

In our model, there is initially a set of p distinct *original virtual machines* $V_m = \{v_1, v_2, ..., v_p\}$ in the data center network, as shown in Fig. 1. Each VM is of unit size and is stored in its *source physical machine* $S(j) \in V_s$ (a source PM can have multiple original VMs initially). Let m_i be the number of units (each unit can store one VM) of initial free storage of PM i. If i is a source PM, the available space after storing its original VMs is thus $m_i - |\{1 \leq j \leq p | S(j) = i\}|$.

Let r_e, r_a, and r_c denote the power consumption in the edge, aggregate, or core switches by relaying one VM. Let $c_{i,j}$ denote the minimum power consumption of transmitting one VM from one PM $i \in V_s$ to another PM $j \in V_s$, and assume along the way there are x edge switches, y aggregate switches, and z core switches. Therefore, $c_{i,j} = x * r_e + y * r_a + z * r_c$. For example, in Fig. 1, if $i = 1$ and $j = 2$, then $x = 1$, $y = z = 0$; if $i = 1$ and $j = 16$, then $x = y = 2$, $z = 1$. When $r_e = r_a = r_c$, $c_{i,j}$ equals the number of switches on the path between PMs i and j.

Replication Constraint of VMs. Replica constraint of VMs refer to the requirement that multiple replica copies of the same VM can not be stored at the same PM, for the purpose of fault-tolerance. This has two consequences. First, the number of replica copies of each VM $R \leq |V_s|$; otherwise, replication constraint can not be satisfied. Second, each PM (including the source PM) is allowed to store at most p distinct VMs, even though its storage capacity could possibly be larger than the total size of the p virtual machines. We therefore define *effective storage capacity* as follows.

Definition 1: **(Effective Storage Capacity of a PM.)** The *effective storage capacity* of PM i, denoted as m_i^e, is the maximum storage capacity of i that can be used to store virtual machines in VM replication. m_i^e is the minimum of the available storage at i and the total size of the VMs that can be stored at i (except its own original VMs). That is, $m_i^e = \min\{m_i - |\{1 \leq j \leq p | S(j) = i\}|, p - |\{1 \leq j \leq p | S(j) = i\}|\}$. □

Problem Formulation of VMR. Denote the k^{th} replica copy of virtual machine $v_j \in V_m$, where $1 \leq j \leq p$ and $k \in \{1, 2, ..., R\}$, as $v_{j,k}$. We define *replication function* $r : V_m \times \{1, 2, ..., R\} \to V_s$, indicating that $v_{j,k}$ is transmitted from its source physical machine $S(j) \in V_s$ to *destination physical machine* $r(j, k) \in V_s$ via the minimum power consumption path between them. Here we assume that the original copy of v_j in $S(j)$ is its primary replica copy and it is not moved, i.e. $r(j, 1) = S(j)$. VMR is to find a replication function r to minimize the *total replication cost*, which is the sum of the minimum power consumption of transmitting all the $R - 1$ replica copies (excluding the original copy) of all the p VMs:

$$\sum_{j=1}^{p} \sum_{k=2}^{R} s_j \times c_{S(j),r(j,k)},$$

under the storage capacity constraint of each PM:

$$|\{1 \leq j \leq p | r(j, k) = i\}| \leq m_i^e, \forall i \in V_s, 1 \leq k \leq R,$$

and the replication constraint of VMs:

$$r(j, k) \neq r(j, k'), 1 \leq j \leq p, k \neq k'.$$

IV. Algorithmic Solutions of VMR

In this section we will present the minimum cost flow based optimal algorithm and two other efficient heuristic algorithms for VMR.

Minimum Cost Flow (MCF) Solution. Below we show that the VMR is equivalent to well-known minimum cost flow problem with below transformation.

Transformation. We first transform the data center network $G(V, E)$, shown in Fig. 1, into a new flow network $G'(V', E')$, shown in Fig. 2, following below five steps.

i). $V' = \{s\} \cup \{t\} \cup V_m \cup V_s$, where s is the supply node and t is the new demand node in the flow network.

ii). $E' = \{(s, i) : i \in V_m\} \cup \{(j, t) : j \in V_s\} \cup \{(i, j) : i \in V_m, j \in V_s\} - \{(v_1, S(1)), (v_2, S(2)), ..., (v_p, S(p))\}$. Note

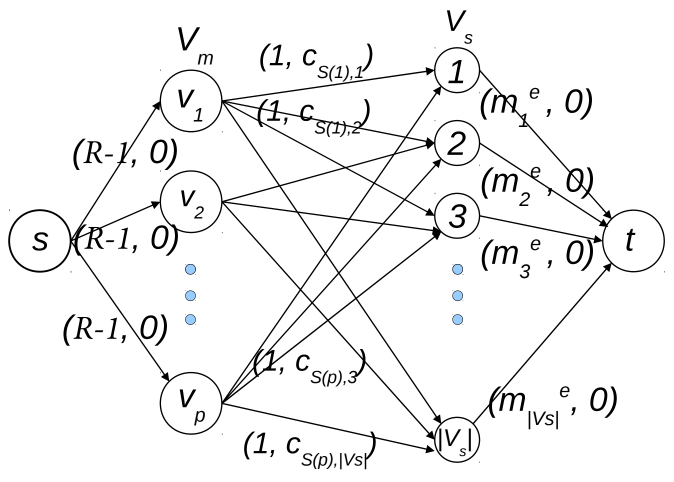

Fig. 2. VMR is equivalent to minimum cost flow problem. In each parenthesis, the first value is the capacity of the edge and the second the cost of the edge.

that there does not exist an edge between VM node v_i and $S(i)$, the source physical machine of v_i. This is because for each VM, only its $R-1$ replica copies, not the original copy, are transmitted.

iii). For each edge (s, v_i), set its capacity as $R-1$ and its cost 0. For each edge (j, t), set its capacity as m_j^e and its cost 0.

iv). For each edge (v_i, j), $v_i \in V_m, j \in V_s$, set its capacity as 1, and its cost as $c_{S(i),j}$, the minimum energy cost sending one VM replica from its source PM $S(i)$ to its destination PM j.

v). Set the supply at s and the demand at t as $p \cdot (R-1)$, which signifying that all together $p \cdot (R-1)$ replica copies must be transmitted inside the data center.

MCF Algorithm. The technique used above is similar to that used in [19] for a closely related problem of data availability in intermittently connected sensor networks. Next, the flow network above (Fig. 2) is passed to a minimum cost flow algorithm discussed below, which calculates the VM replication function r. That is, for each of $R-1$ replica copies of each of the p original VM, the MCF outputs its destination PM and the path in the data center network along which the replica copy is transmitted.

Time Complexity of Optimal VMR Algorithm. VMR algorithm includes both above transformation and the minimum cost flow algorithm. In the transformation, finding the minimum energy consumption path between any two PMs takes $O(|V_s|)^3$, constructing $G'(V', E')$ takes $p + p \cdot (|V_s| - 1) + |V_s| = (p+1) \cdot |V_s|$, which is $O(p \cdot |V_s|)$. Therefore it takes $O(|V_s|^3 + p \cdot |V_s|)$ for transformation. For the minimum cost flow, we adopt the algorithm proposed in [9], which is based on scaling push-relabel method and its implementation works well over a wide range of problem classes. For any flow network, the algorithm has the time complexity of $O(a^2 b \log(ac))$, where a, b, and c are the number of nodes, number of edges, and maximum edge capacity in the flow network. Above transformation gives that $|V'| = |V_s| + p + 2$ and $|E'| = p + p \cdot (|V_s| - 1) + |V_s| = (p+1) \cdot |V_s|$. Therefore the time complexity of the minimum cost flow is $O\big((|V_s| + p)^2 \cdot (p \cdot |V_s|) \log(|V_s + p| \cdot c)\big)$, where $c = \max\big\{R-1, \max_i^{|V_s|}\{m_i^e\}\big\}$.

After all the $R-1$ copies of each VM are placed in the data center following above minimum cost flow algorithm, we next turn off the inactive physical machines defined below.

Definition 2: (**Active Physical Machine (APM) and Inactive Physical Machine (IPM).**) A physical machine is active if it has at least one VM (either original or replica) stored in its local storage; otherwise, it is considered as inactive and therefore can be turned off. □

EXAMPLE 1: Fig. 1 shows the final placement of replica VMs following minimum cost flow solution, with $p = 5$ and $R = 3$. Specifically, replica VMs of v_1 are placed at PM 2 and 4; replica VMs of v_2 placed at PM 6 and 7; replica VMs of v_3 placed at PM 10 and 11; replica VMs of v_4 placed at PM 16 and 13; and replica VMs of v_5 placed at PM 15 and 14. PMs 1, 8, and 12 are IPMs thus turned off. □

Theorem 1: VMR is equivalent to minimum cost flow problem, therefore can be solved optimally and efficiently.

Proof: We show that with above transformation and minimum cost flow algorithm, $R-1$ copies of each VM are created and placed with minimum total replication cost, while satisfy replication constraint of VMs and storage constraint of PMs.

First, according to the transformation, since the amount of supply at s is $p \cdot (R-1)$ (step v), and the capacity of each edge (s, v_i) $(1 \le i \le p)$ is $R-1$ (step iii) , a valid flow of amount $p \times (R-1)$ from s to t must be divided into $R-1$ amount on edge (s, v_1), $R-1$ amount on (s, v_2), ..., and $R-1$ amount on (s, v_p). Therefore $R-1$ amount of flow must come out of v_i, signifying that $R-1$ replica copies of each VM must be created.

Next, since the edge capacity of each edge (v_i, j), $v_i \in V_m, j \in V_s$ is 1 (step iv), and there does not exist an edge between VM node v_i and $S(i)$ (step ii), it must be that each of the $R-1$ flows that arrive at v_i travels toward a different PM other than v_i's source PM. This satisfies the replication constraint of VMs. Meanwhile, that the edge capacity of edge (j, t) is m_j^e designates that no more than m_j^e amount of flow come out of node $j \in V_s$, meaning PM node j can not store more VM copies than what its effective storage capacity allows. This satisfies the storage constraint of PMs.

As for the cost, note that the edge cost of (v_i, j), $v_i \in V_m, j \in V_s$ is $c_{S(i),j}$, the minimum power consumption between PMs $S(i)$ and j, while all other edges have cost zero. This reflects the fact that only replication cost is considered in VMR. Finally, minimum cost flow algorithm gives the minimum cost of sending $p \cdot (R-1)$ amount of flow from s to t, showing that the corresponding replication cost obtained is indeed minimum. ■

Heuristic Algorithms for VMR. In addition, we propose two other time-efficient heuristic VM replication algorithms viz.

First Fit and **Greedy**, and compare them with the optimal minimum cost flow solution via simulations.

First Fit. Suppose that all the available PMs are ordered from left to right in the fat tree data center topology. The algorithm starts to replicate each of the original VMs and place their $R-1$ replica copies on the first available PM, the second available PM, so on and so forth, until all the VMs have their replica copies placed in the data center. Note that each placement has to satisfy the storage constraint of each PM as well as the replication constraint of VMs. Checking storage constraint takes constant time while replication constraint takes $O(\bar{m})$, where $\bar{m}$ is the average storage capacity of a PM. Therefore it takes $p \cdot (R-1) \cdot \bar{m}$. Since $R = O(|V_s|)$, the time complexity of FirstFit is $p \cdot |V_s| \cdot \bar{m}$.

Greedy. In this algorithm, for each VM, it places each of its replica copies to the closest PM in terms of power consumption. This continues until all the replica copies of all the VMs are placed. Finding all the minimum energy consumption paths between all pair of PMs takes $O(|V_s|^3)$, placing all the replica copies takes $O(p \cdot |V_s| \cdot \bar{m})$. Therefore the time complexity of Greedy is $O(|V_s|^3 + p \cdot |V_s| \cdot \bar{m})$.

V. Server Consolidation Algorithm

Recall that at the end of VM replication algorithm, it turns off all the inactive physical machines (IPMs) based on the VM replica placement, as shown in Example 1. In this section, we further consolidate all the left active physical machines, in order to obtain and turn off more inactive physical machines. This is indeed possible. To obtain more inactive PMs, the key observation is that we can move the VM replica copies from one active PM to another to "empty" some of the active PMs. During consolidation, however, it needs to maintain the total replication cost yielded by the minimum cost algorithm as well as to satisfy the storage and replication constraints. For example, in Fig. 1, the two replica VMs that are placed at PM 13 and 14 can be placed together in one PM, say PM 13, while maintaining the same total replication cost and satisfying the replication constraint of VMs (since they are replica copies of different original VMs). We can therefore turn off PM 14, which further saves energy power of the data centers. In server consolidation process, the original VMs will not be moved, because they served as the "anchor" nodes in the minimum cost VM replication process. If they are moved around, the obtained minimum cost replication cost is no longer achieved. Before presenting the server consolidation algorithm, we first give following definitions.

Definition 3: (**Target Physical Machine (TPM) of a Replica VM** $v_{j,k}$**.**) Recall that for replica VM $v_{j,k}$, its source PM is $S(j)$ and its destination PM (from minimum cost flow) is $r(j,k)$. A PM i is a TPM of $v_{j,k}$ if a) $c_{S(j),r(j,k)} = c_{S(j),i}$, that is, it has the same replication cost to $S(j)$ as $r(j,k)$ does, b) it has enough storage to store $v_{j,k}$, and c) it does not store original VM v_j or any replica VM of v_j, that is, for any VM $v_{j',k'}$ PM i stores, $j' \neq j$. □

Definition 4: (**Consolidating Physical Machine (CPM).**) A physical machine is CPM if it is active and it is not a source PM (i.e., it does not store any original VM). CPMs are PMs that can be potentially turned off and made inactive. □

Algorithm 1 works as follows. It first finds all the CPMs that has only one replica VM, and finds the TPMs for this replica. If at least one such TPM exists, it moves this replica VM to any one of them and turns off this CPM and marks it IPM. We assume that once a PM is identified as TPM, it can no longer be turned off. Next, it finds all the CPMs that has two replica VMs, tries to find their TPMs and turns off these CPMs. This takes place until all the CPMs are checked.

Algorithm 1: Server Consolidation Algorithm.
Input: VM replica placement from VM replication algorithm
Output: Number of IPMs.

```
0.  Notations:
    $m$: largest number of replica VMs a CPM stores
    $N_{ipm} = 0$: number of IPMs
1.  for ($i = 1$ to $m$)
2.      for (each of the CPMs that has $i$ replica VMs)
3.          flag = true;
4.          for (each of the replica VMs)
5.              if (it can find a TPM)
6.                  move the replica VM to the TPM
7.              else
8.                  flag=false;
9.                  break;
10.             end if;
11.         end for;
12.         if (flag == true)
13.             $N_{ipm}++$; /*This CPM can be turned off */
14.     end for;
15. end for;
16. RETURN $N_{ipm}$. /*Return number of IPMs */
```

We emphasize that server consolidation does not incur extra energy consumption on top of VM replication. To achieve that, the VM replicas are not physically placed in the data center until server consolidation is done, at which point the final placement of all the VM replicas is decided. The total energy consumption for VM replication and PM consolidation is still the one yielded by the minimum cost flow based VM replication algorithm.

Time Complexity. Finding all the minimum energy consumption paths between all pair of PMs takes $O(|V_s|^3)$. Finding the largest number of replica VMs a CPM stores takes $O(|V_s|)$. There are three *for* loops in Algorithm 1. For the outermost and innermost ones, each takes $O(\bar{m})$, where $\bar{m}$ is the average storage capacity of a PM. For the *for* loop in between, it takes $O(|V_s|)$. To check if a replica VM has a TPM or not takes $O(|V_s| \cdot \bar{m})$. Therefore the total time complexity of Algorithm 1 is $O(|V_s|^3 + \bar{m}^3 \cdot |V_s|^2)$.

VI. Performance Evaluation

Simulation Setting. In this section, we compare the performances of the three VM replication algorithms viz. **Optimal**,

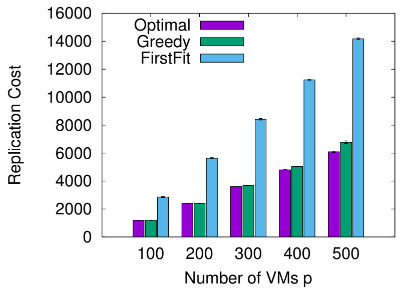

Fig. 3. Performance comparison by varying p, number of original VMs. Here, $k = 8$ and $R = 7$.

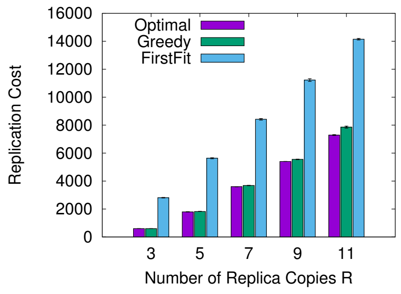

Fig. 4. Performance comparison by varying R, number of replica copies of each VM. Here, $k = 8$ and $p = 300$.

Greedy, and **FirstFit**, as well as the performances between Optimal and server consolidation algorithm. We generate data centers of different sizes: $k = 8$, a small data center with 128 PMs; and $k = 16$, a large data center with 1028 PMs. The original VMs are randomly generated on the PMs. The size of each VM (and its replica copies) is 1 unit. The storage capacity of each PM is 30, which means each PM can store maximum 30 VMs. This is consistent with real-world scenario. For example, 30 VMs could run on the same host virtualized with Microsoft Hyper-V on Cisco UCS blades that offer 256 GB RAM and two eight-core Intel E5-2665 CPUs. Recall that r_e, r_a, and r_c denote the power consumption of transmitting one VM on the edge, aggregate, and core switches respectively. Throughout the simulation, we set $r_e = r_a = r_c = 1$ (that is, the power consumption is measured as number of switches a VM traverses [16]). We also vary them to reflect the fact that high-end core switches consume more power than aggregation and edge switches. In all the simulation plots, each data point is the average of five runs, and the error bars indicate 95% of confidence interval.

VM Replication Algorithms. Fig. 3 and Fig. 4 compare the three algorithms in a data center of 128 PMs, by varying number of original VMs p and number of replica copies of each VM R, respectively. It shows that with the increase of either p or R, the replication cost of all three algorithms increase. In all cases, however, Optimal performs the best, while Greedy performs very closely as Optimal. Both algorithms cost roughly half of the power consumption that FirstFit incurs. Table I further shows that the number of active PMs resulted from the three algorithms. It shows that when p is small (100 or 200), FirstFit yields least number of active PMs among the three algorithms; while when p gets large, all the PMs are active in all three algorithms. This is consistent with previous finding that FirstFit performs well in terms of server consolidation [3], even though it incurs the highest VM replication costs.

Study of Scalability. Since the performances of Optimal and Greedy are comparable, next we only focus on these two. Table II shows the replication costs of both algorithms in a small 128-PM data center and a large 1028-PM data center. For fair comparison, we set all the parameters such that after replication, the entire storage capacities of both data centers

TABLE I
NUMBER OF ACTIVE PHYSICAL MACHINES BY VARYING p, NUMBER OF ORIGINAL VMS. HERE, $k = 8$ AND $R = 7$.

Algorithms	$p = 100$	$p = 200$	$p = 300$	$p = 400$	$p = 500$
Optimal	126	128	128	128	128
Greedy	124	128	128	128	128
FirstFit	80	113	123	127	128

are half occupied. The last column, *Improvement Percentage*, is calculated as the replication cost difference between Optimal and Greedy divided by replication cost of Greedy. It shows that in small data center, Optimal improves upon Greedy by 2.2% while in large data center, it is 11.2% improvement. This shows that Optimal algorithm performs better than Greedy in large data centers therefore is more scalable.

Varying r_e, r_a, and r_c. We set $r_e = 10$, $r_a = 5$, and $r_c = 1$ to reflect the observation that core switches consume more power than aggregation switches, which consume more power than edge switches. Fig. 5 shows the replication cost of FirstFit is five- to ten-times higher that those in Optimal and Greedy. Comparing with Fig. 3, this shows that the performance difference between FirstFit and the other two algorithms is much larger in this case. This is because the placement of replica VMs in FirstFit goes through the core switches more frequently than the other two, which costs more power when r_e is 10 as opposed to r_e is 1.

Server Consolidation Algorithm (Algorithm 1). Finally we study the performance of the proposed server consolidation algorithm in a 1028-PM data center, and show the number of inactive physical machines (IPM) obtained from this algorithm. Fig. 6 shows that with the increase of R, the number of IPMs decreases. This is because more VM replica copies spread into data center, making the consolidation more

TABLE II
STUDY OF SCALABILITY. FOR 128-PM DATA CENTER, $p = 300$ AND $R = 7$. FOR 1028-PM DATA CENTER, $p = 1500$ AND $R = 10$.

Data Center Size	Optimal	Greedy	Improvement Percentage (%)
128	3600	3684	2.2
1024	37352.8	42086.4	11.2

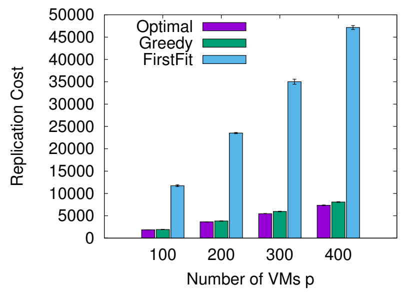

Fig. 5. Performance comparison by varying p, number of original VMs. Here, $r_e = 10$, $r_a = 5$, and $r_c = 1$, $k = 8$ and $R = 5$.

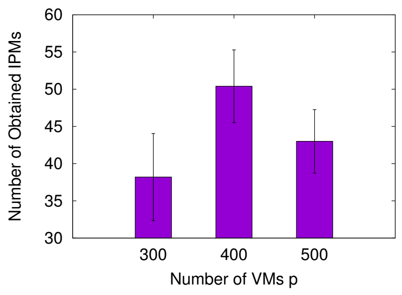

Fig. 7. Server consolidation by varying p, number of initial VMs. Here, $k = 16$, $R = 5$.

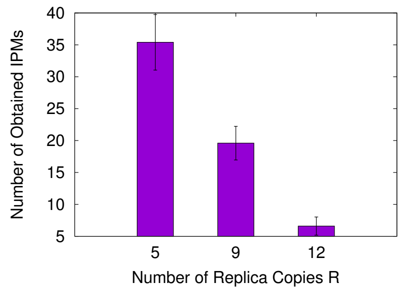

Fig. 6. Server consolidation by varying R, number of replica copies of each VM. Here, $k = 16$ and $p = 200$.

difficult. Fig. 7 shows the number of IPMs by varying p, number of initial VMs. The reason that $p = 400$ has the highest number of 50 IPMs is because in this case, replica copies of the same VMs could spread to PMs under different edge switches, giving rise of the scenario that replica copies from different VMs could be consolidated on the same PMs.

VII. Conclusion and Future Work

We designed power-efficient optimal and heuristic VM replication algorithms. VM replication is an important measure to overcome server failure in data centers, and provides good service to cloud-based data center users. We consider conserving power consumption inside the data center networks such as switches and links, as well as the power consumptions on the physical machines. For the former, we designed an optimal and efficient virtual machine replication algorithm. For the latter, we proposed an efficient server consolidation algorithm that turns off unused physical machines. Extensive simulation results showed that our algorithms perform well under different data center scenarios. Currently, our server consolidation algorithm is a heuristic algorithm. As one of the future work, we would like to study the hardness of the server consolidation problem, and propose optimal or approximation algorithms.

VIII. Acknowledgement

This work was supported in part by NSF Grant CNS-1419952.

References

[1] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. In *Proceedings of ISCA*, 2010.
[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.
[3] Y. Ajiro and A. Tanaka. Improving packing algorithms for server consolidation. In *Proceedings of the 33rd International Computer Measurement Group Conference*, 2007.
[4] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, 2008.
[5] M. Alicherry and T.V. Lakshman. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *Proceedings of IEEE INFOCOM*, 2013.
[6] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.
[7] S. Fang, R. Kanagavelu, B. Lee, C. H. Foh, and K. M. M. Aung. Power-efficient virtual machine placement and migration in data centers. In *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013.
[8] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, 2011.
[9] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algorithms*, 22:1–29, 1997.
[10] H. Goudarzi and M. Pedram. Energy-efficient virtual machine replication and placement in a cloud computing system. In *Proceedings of IEEE Cloud*, 2012.
[11] C. Guo, G. Lu, H. J. Wang., S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *Proceedings of CoNEXT*, 2010.
[12] 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*, 2008.
[13] M. Li, D. Subhraveti, A. R. Butt, A. Khasymski, and P. Sarkar. Cam: A topology aware minimum cost flow based resource manager for mapreduce applications in the cloud. In *Proceedings of HPDC*, 2012.
[14] X. Li, J. Wu, S. Tang, and S. Lu. Let's stay together: Towards traffic aware virtual machine placement in data centers. In *Proceedings of IEEE INFOCOM*, 2014.
[15] F. Machida, M. Kawato, and Y. Maeno. Redundant virtual machine placement for fault-tolerant consolidated server clusters. In *Proceedings of the 12th IEEE/IFIP NOMS, mini conference*, 2010.
[16] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of IEEE INFOCOM*, 2010.
[17] J. Mudigonda and P. Yalagandula. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of USENIX NSDI*, 2010.
[18] C.H. Papadimitriou and K. Steiglitz. Combinatorial optimization: Algorithms and complexities. *Prentice Hall*, 1982.
[19] B. Tang, N. Jaggi, and M. Takahashi. Achieving data k-availability in intermittently connected sensor networks. In *Proceedings of the IEEE ICCCN*, 2014.