A Fine Grained Dynamic Virtual Machine Placement Framework for Energy Aware Data Centers

Zhiming Zhang, Chan-Ching Hsu and J. Morris Chang, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
Iowa State University

Abstract—With the continuing growth of cloud computing services, power consumption has become one of the most challenging issues in data center environments. Virtualization technology enables one physical machine to host several virtual machines (VMs) which helps to achieve much higher per-server utilization. With the support of today's virtual machine migration technology, the efficiency and flexibility of data center management can be further enhanced, creating great energy saving opportunities. In this paper, a fine grained virtual machine placement framework is proposed to manage the mappings of VMs to physical servers. This framework solves the problem of finding the most energy efficient way (least resource wastage and least power consumption) of placing the VMs considering their resource requirements. We formulate the mapping problem into an ILP optimization problem to which the optimal solution is able to be found with featured software packages. As the formulated problem is NP hard, in order to improve the computational complexity, a heuristic approach is proposed to solve the problem efficiently. A real testbed data center implemented with industry product VMware vSphere 5 is used to evaluate the proposed framework. Experiment result demonstrates our design can effectively improve data center energy efficiency with each VM's resource requirement satisfied. Our design performs better than VMware's Distributed Resource Scheduler (DRS) in respect of load balancing and power consumption.

Index Terms—Energy aware computing, data center, cloud computing, virtual machine migration, VMware.

1 Introduction

Rapid growth of cloud computing services have led to creation of large scale enterprise data centers. These data centers consumed an estimated 108 billion kWh (108 tWh) in 2010 [1], 2.8% of the total U.S. electricity consumption. This power consumption is now expected to double in five years. In a typical data center, servers consume about 44% of the power usage [1]. How to reduce energy related costs in a data center has now become one of the most challenging issues for cloud computing service providers.

Virtualization technology has been adopted to enhance efficiency and flexibility in data center management and resource provisioning. With the support of virtualization, resources of a single server can be divided into multiple isolated execution environments. This allows one physical machine to host several virtual machines (VMs), which helps to achieve higher per-server utilization and results in higher energy efficiency.

The virtualized data center environment in this paper provides a shared hosting infrastructure to customers who need server resources to run their applications. All the applications run inside virtual machines which are provisioned and managed on-demand. VMs' resource utilization (CPU, memory, network, etc.) must be constantly monitored and the data center manager must respond to the changing on-demand resource requests from applications and determine which physical server a VM should be placed on. This is a time consuming task that can not be performed by human operators in a timely fashion considering the complexity and size of the data center.

A naive approach for management of VMs may hurt application performance, e.g., when physical resources are not sufficiently allocated. On the other hand, if the VMs are over provisioned, physical resources will be poorly utilized and this results in waste of energy. Questions like how to fully utilize resources, achieve maximum power savings, eliminate hot spots must be taken care of before VM placement decisions can be made.

In the past few years, virtualized data center management has brought a great amount of research interest. Earlier work mostly focuses on improving resource utilization and load balancing of VMs across physical servers [2], [3], [4], [5]. Recently, more work is being devoted to energy saving optimizations. Several approaches propose to use VM consolidation to provide energy savings. The main idea of VM consolidation is to place VMs onto a smaller number of physical servers, and turn off the remaining servers

Zhiming Zhang, Chan-Ching Hsu and J. Morris Chang are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, 50010 USA e-mail: (zhiming @iastate.edu and morris @iastate.edu).

to save energy.

Most of the extant approaches [6], [7], [8], [9] only consider solving one specific problem or focus on one aspect of optimization, e.g., balancing VMs across servers, eliminating hot spots, minimizing power consumptions, maximizing resource utilizations, etc. However, these goals or optimizations should be considered together to build a well-performing data center. Notice that some of the objectives may conflict with each other, making the optimization problem more complicated. Another issue is that past work only focuses on one or two server resources at the same time, e.g., CPU, memory or network bandwidth. These solutions usually perform well on the server resource(s) being considered and leave potential performance bottlenecks on the resource(s) being left out. Live VM migration has been used in a few work to facilitate dynamic VM placements. However, these proposed approaches are limited by their over simplified models that do not consider the runtime fluctuation of workloads or the VM migration costs.

In this paper, we propose a fine grained dynamic virtual machine placement framework which is capable of building a real energy aware data center. The proposed framework design includes three major components. The first component is responsible of collecting runtime resource utilizations of each VM. These resources include CPU, memory, network and storage. The second component is an integer linear programming (ILP) optimization model that provides the optimal VM placement solution. The objective function of this model is to minimize data center energy consumption without affecting each VM's performance. The model takes each VM's resource requirements as its constraints to guarantee performance. The third component is a commander responsible of sending out VM migration commands based on the placement solution from the optimization model. Live migration is used here so service of each VM will not be interrupted during dynamic placements. Our design also considers the VM migration cost which is often ignored in past works.

We implement our design with VMware Vsphere 5 and build a real testbed to evaluate our work. Experiment result demonstrates our design provides better performance than VMware's Distributed Resource Scheduler (DRS) in regard of load balancing and power consumption. The main contributions of our work are:

- To the best of our knowledge, this is the first work that provides the dynamic VM placement framework with the objective of minimizing energy consumption. We consider all aspects of resources including CPU, memory, network and storage which were not touched altogether in any of previous work. Also the cost of live migration is firstly introduced in this work.
- Our design is a complete solution that constantly monitors the workload changes over time in a data center and pro-actively provides the VM placement solutions by solving the formulated ILP model, achieving VM consolidation to guarantee VMs' performance and save energy.
- Our design is implemented with industry product and tested on a real testbed data center, which gives a solid performance evaluation in the practical aspect. The experimental result suggest that our model is able to improve the energy consumption of data centers significantly, and the proposed heuristic approach can improve the computation complexity and scale well to large scale server clusters.

The remaining of the paper is organized in the following sequence. Related work is given in Section 2. We provide our VM placement optimization model in Section 3. Section 4 introduces the complete framework. The implementation of our design is provided in section 5. Section 6 demonstrates the experiment results and Section 7 concludes this paper.

2 Related Work

Virtualized data center management has gathered a great amount of research interests in the past few years. Recent studies focus on improving server resource utilizations, meeting power budgets, balancing workloads among servers and reducing any energy related costs. We have done an extensive study of past works to inspire our design. A brief discussion of past achievements and limitations is given as follows.

Timothy et al. [10] propose a VM mapping algorithm called Sandpiper to detect hotspots and relocate VMs from overloaded servers to under-utilized ones. When a migration between two servers is not directly feasible, Sandpiper can identify a set of VMs to interchange in order to free up sufficient amount of resources on the destination server. This approach is able to solve simple replacement issues but requires extra memory space for interim hosting of VMs. This process also needs extra rounds of migration and may affect system performance. Singh et al. [11] propose a load balancing algorithm called VectorDot for handling the hierarchical and multi-dimensional resource constraints in virtualized data centers. The proposed algorithm is evaluated on a real data center testbed built with VMware ESX servers and network attached storages.

Grit et al. [12] consider some VMs replacement issues for resource management policies in the context of a system for on-demand leasing of shared networked resources in server clusters. When a migration is not directly feasible, due to sequence issues, the VM is suspended using suspend-to-disk. Once the destination server becomes available, the VM resumes. In our work, when migration is not feasible, we first try to find the delay-tolerant VM and suspend it to release

server resource for other time-sensitive VMs. When server resource becomes available, the suspended VM resumes. Electricity price is also considered for determining when to resume the suspended VM in order to minimize cost.

Jing et al. [13] propose a multi-objective virtual machine placement algorithm that simultaneously minimize power consumption, resource wastage and thermal dissipation. Xin et al. [14] also consider physical resources as multi-dimensional and propose a multi-dimensional space partition model to determine the mapping of VMs and PMs. [15], [16], [17], [13], [14] are static VM placement methods that only consider the initial VM placement and do not consider future dynamic workload changes that may need VM remappings. Shrivastava et al. [18] consider the inherent dependencies between VMs comprising a multi-tier application. They propose a scheme called AppAware to determine VM placement that can greatly reduce network traffic considering the interaction between applications running on different VMs.

Meng et al. [19] consider the network traffic and bandwidth as factors that may affect system performance. They optimize VM placement based on traffic patterns and communication distances. VMs with mutual banwidth usage are assigned to PMs. Cost-aware workload placement is also gathering wide interest in data center operations. [20], [21], [22], [23], [24], [25] propose to reduce data center operational cost by exploiting electricity price differences across regions. Workload can be migrated to a data center where resource is sufficient and energy price is low. However, migration cost and service level agreement are not considered. Furthermore, short running workloads do not make sense in this scenario. [26] focus on how to use green energy to power data centers.

The works we studied above all take advantage of server virtualization and provides improved and balanced resource utilization. However, these works do not consider the cost of VM migration while making VM remapping decisions which may lead to undesirable results. In our work, we take this actual cost of live migration into account, and we put this cost in terms of time and energy consumption into our model to decide whether migration is indeed beneficial. Another weakness is that some works use simulation to evaluate their proposed algorithm or schemes which oversimplifies the actual dynamic changes of workloads in a real data center. In our work, we build a real virtualized data center testbed comprised of VMware ESXi servers, ISCSi network attached storages which provides a comprehensive evaluation of our design.

Bobroff et al. [27] use first-fit approximation to solve the VM placement problem focusing on CPU utilization. Fabien et al. [28] formulate VM placement into a constraint satisfaction problem to minimize the number of physical machines. This work only considers uni-processor computers and assumes each PM can only host one active VM. The VM placement problem is oversimplified. In our work, the VM placement problem is not constrained by uni-processor computers. Instead, the constraints come from whether available resource on PM can satisfy VM SLA or resource requirement. Hien et al. [29] extends [28] and considers CPU and RAM as constraints. However their implementation assumes these two inputs are known in advance. In our work, we use VM history to predict its future resource needs. We also plan to study the autocorrelation and periodic attributes of VM history for more accurate prediction.

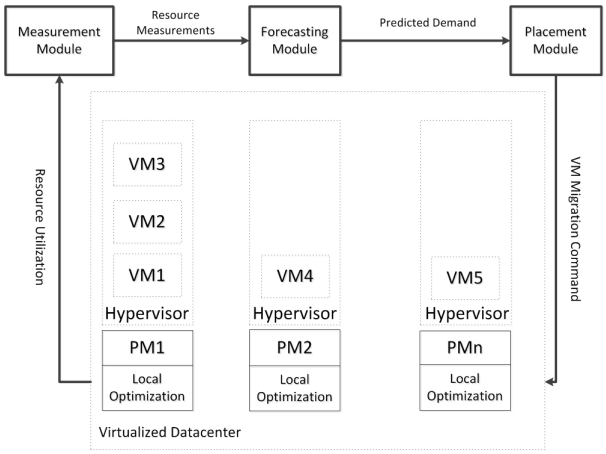


Fig. 1. VM Placement Framework

3 System Model

The proposed design shown in Fig. 1. includes three major components. The first component is responsible of collecting runtime resource utilizations of each VM. These resources include CPU, memory, network and storage. The second component is a linear programming optimization model that provides the optimal VM placement solution. The objective function of this model is to minimize data center energy consumption without affecting each VM's performance. The model takes each VM's resource requirements as its constraints to guarantee performance. The third component is a commander responsible of sending out VM migration commands based on the placement solution from the optimization model. Live migration [30] is used here so service of each VM will not be interrupted during dynamic placements. Our design also considers the VM migration cost which is often ignored in past works.

3.1 Live Migration

One major benefit of virtualization is that it enables server consolidation and provides better multiplexing of data center resources across VMs. Carefully placing VMs on physical servers (PSs) based on their resource

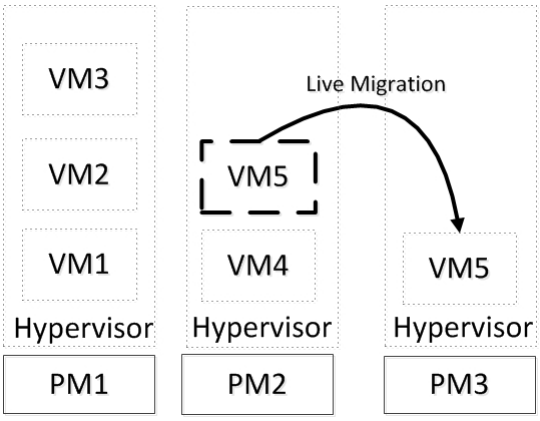


Fig. 2. Virtual Machine Migration

needs can greatly reduce data center management cost and energy consumption. VM live migration is an emerging technology that enables virtual machines to migrate from one physical server to another with little service downtime. As shown in Fig. 2., VM5 is migrated from PM2 to PM3. Live migration allows us to dynamically remap VMs to PSs based on the changing workload. For example, when a PS is overloaded, VMs can be moved out and migrate to a less loaded PS to guarantee performance. On the other hand, when there are several under-utilized PSs, VMs can be consolidated to a fewer number of PSs thus providing energy savings.

However, live migration does not come free. Live migration works by transferring VM's architecture states and memory data from its original host to its destination host. This process consumes extra CPU and network resource and may harm performance thus must be taken into consideration before migration is applied. Virtual machine placement algorithms have been proposed to enhance the performance and energy efficiency of a data center.

3.2 VM Placement Problem Formulation

The following sections presents the optimization model for the virtual machine replacement problem. Let $\mathbb{N}_{\mathbb{VM}} = \{VM_0, \ldots, VM_{M-1}\}$ be the set of virtual machines with cardinality $|\mathbb{N}_{\mathbb{VM}}| = M$. Similarly, let $\mathbb{N}_{\mathbb{PM}} = \{PM_0, \ldots, PM_{N-1}\}$ be the set of physical machines with cardinality $|\mathbb{N}_{\mathbb{PM}}| = N$. The problem is defined as follows:

Given (1) the power and time requirements of virtual machine m to run on physical machines n, P_{mn} and T_{mn}, (2) the migration cost of each virtual machine in power, $P_{mn}^{migrate}$ and in time, $T_{mn}^{migrate}$, (3) the physical machine capacity in terms of cpu, memory, hardware and bandwidth utilization, H_n^{CPU}, H_n^{MEM}, H_n^{HD} and H_n^{BW}, (4) the virtual machine utilization requirements for CPU, U_m^{CPU}, memory, U_m^{MEM}, hard disk, U_m^{HD} and bandwidth, U_m^{BW} (5) the active and sleep mode power use of the physical machines, P_{active}^n and P_{sleep}^n, the problem is to minimize the system energy consumption, denoted as $\mathcal{E}$, by placing virtual machines on physical machines PMs, deploying VMs in active mode with the consideration of the migration cost. The output includes the virtual machine destinations, migration indicators and operation mode of physical machines specified as, l_{mn}, g_{mn} and o_n.

3.2.1 Decision Variables

The decision variables of the placement problem are presented by two matrices $\mathcal{L}$, $\mathcal{G}$ and a vector $\mathcal{O}$. $\mathcal{L} = (l_{mn})_{M\times N}$ is the virtual-physical machine incidence matrix, or the *placement matrix* and $\mathcal{G} = (g_{mn})_{M\times N}$ is the virtual machine migrating incidence matrix, or the *migration matrix*; $\mathcal{O} = (o_n)_{1\times N}$ is the physical machine activation incidence vector, or the *operation mode vector*. The value of a decision variable is determined as follows.

$$l_{mn} = \begin{cases} 1, \text{VM } m \text{ is placed on PM } n, \\ \quad \forall m \in \mathbb{N}_{\mathbb{VM}}, n \in \mathbb{N}_{\mathbb{PM}}; \\ 0, \text{otherwise}, \end{cases}$$

$$g_{mn} = \begin{cases} 1, \text{VM } m \text{ is migrated to PM } n, \\ \quad \forall m \in \mathbb{N}_{\mathbb{VM}}, n \in \mathbb{N}_{\mathbb{PM}}; \\ 0, \text{otherwise}, \end{cases}$$

$$o_n = \begin{cases} 1, \text{PM } n \text{ is in active mode}, \forall n \in \mathbb{N}_{\mathbb{PM}}; \\ 0, \text{otherwise}, \end{cases}$$

When l_{mn} is asserted, the physical machine n serves the virtual machine m, provisioning with the power, P_{mn}; o_m has to set a value of one correspondingly. If VM m does migrate to PM n, the migration cost involves power, $P_{mn}^{migrate}$ and time, $T_{mn}^{migrate}$ for the procedure. It is conceivable that a physical machine operateing actively ($o_m = 1$) consumes energy at a much higher level than in sleep mode.

3.2.2 Placement Constraint

Each virtual machine can be served by only one physical machine, and it must be placed on one of the physical machines to have the resource granted toward it. The following constraint essentially set up this condition to satisfy.

$$\sum_{n\in\mathbb{N}_{\mathbb{PM}}} l_{mn} = 1, \forall m \in \mathbb{N}_{\mathbb{VM}}, \tag{1}$$

For an individual virtual machine m, if the result of $\sum_{m\in\mathbb{N}_{\mathbb{VM}}} l_{mn}, \forall n \in \mathbb{N}_{\mathbb{PM}}$ is equal to 0, it means the physical machine n operates in sleep mode and no virtual machines will run on it.

The idea of virtual machine migration is put into action if a virtual machine is placed on a physical machine n different from the one by which it is currently served before the optimal policy is made. Namely, for a machine m, its next placement, $l_{mn} = 1$ and its current placement, $l'_{mn} = 0$, which is given

TABLE 1
Definitions of Important Symbols

Symbol	Definition
N	Number of physical machines to serve virtual machines
M	Number of virtual machines
P_{active}	Basic power level of physical machines in active mode
P_{sleep}	Power level of physical machines in sleep mode
Period	Time period for which the solution pertains
$P_{mn}^{migrate}$	Power level for VM m migrating to PM n
$T_{mn}^{migrate}$	Time for VM m migrating to PM n
H^{CPU}	Limit on CPU utilization of physical machines
H^{MEM}	Limit on memory utilization of physical machines
H^{HD}	Limit on hard disk utilization of physical machines
H^{BW}	Limit on network bandwidth utilization of physical machines
$\mathbb{N}_{\mathbb{VM}}$	Set of VMs, $\lvert\mathbb{N}_{\mathbb{VM}}\rvert = M$
$\mathbb{N}_{\mathbb{PM}}$	Set of PMs, $\lvert\mathbb{N}_{\mathbb{PM}}\rvert = N$
$\mathbf{U^{CPU}}$	Virtual machine CPU utilization, $\mathbb{U}^{\mathbb{CPU}} = \{U_m^{CPU}, \forall m \in \mathbb{N}_{\mathbb{VM}}\}$
$\mathbf{U^{MEM}}$	Virtual machine memory utilization, $\mathbb{U}^{\mathbb{MEM}} = \{U_m^{MEM}, \forall m \in \mathbb{N}_{\mathbb{VM}}\}$
$\mathbf{U^{HD}}$	Virtual machine hard disk utilization, $\mathbb{U}^{\mathbb{HD}} = \{U_m^{HD}, \forall m \in \mathbb{N}_{\mathbb{VM}}\}$
$\mathbf{U^{BW}}$	Virtual machine network bandwidth utilization, $\mathbb{U}^{\mathbb{BW}} = \{U_m^{BW}, \forall m \in \mathbb{N}_{\mathbb{VM}}\}$
$\mathcal{E}$	Total system energy consumed
$\mathcal{L}$	Placement matrix (decision variable), $\mathcal{L} = (l_{mn})_{M\times N}$
$\mathcal{G}$	Migration matrix (decision variable), $\mathcal{G} = (g_{mn})_{M\times N}$
$\mathcal{O}$	Operation mode vector (decision variable), $\mathcal{O} = (o_n)_{1\times N}$

information initially, are compared to represent that the virtual machine m is migrated to the physical machine n from other physical machine. Constraint (2) gives the value of g_{mn} by the comparison of the two states, l_{mn} and l'_{mn}.

$$l_{mn} - l_{mn} \cdot l'_{mn} = g_{mn}, \forall m \in \mathbb{N}_{\mathbb{VM}}, n \in \mathbb{N}_{\mathbb{PM}}, \quad (2)$$

3.2.3 Resource Constraints

There are constraints which define resource utilization in the physical machine domain. Virtual machines on each physical machine in the solution need guaranteed service levels. The service level are categorized by four aspects: CPU, memory, hard disk and bandwidth utilization, which haven't considered in the past work and firstly are jointly discussed here. Essentially, the deployed resources of a physical machine cannot exceed a specified utilization level. Based on service level requirements, the resource utilization of physical machines varies. The constraints are as follows, where U_m indicates the utilization of virtual machine m:

$$\sum_{m\in\mathbb{N}_{\mathbb{VM}}} (l_{mn} \cdot U_m^{CPU}) \leq H_n^{CPU}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \quad (3)$$

$$\sum_{m\in\mathbb{N}_{\mathbb{VM}}} (l_{mn} \cdot U_m^{MEM}) \leq H_n^{MEM}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \quad (4)$$

$$\sum_{m\in\mathbb{N}_{\mathbb{VM}}} (l_{mn} \cdot U_m^{HD}) \leq H_n^{HD}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \quad (5)$$

$$\sum_{m\in\mathbb{N}_{\mathbb{VM}}} (l_{mn} \cdot U_m^{BW}) \leq H_n^{BW}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \quad (6)$$

For physical machines, the resource utilization level is limited to be not over a specific value, for example, H^{CPU} is the maximum CPU utilization level, which could be a measurement of percentage. A set-up percentage less than 100% leaves the margin for new arrival tasks. Energy can be consumed at diverse power levels and the execution time are different from virtual machines.

3.2.4 Operation Mode Constraints

The following two constraints define in which mode a physical machine will run. If it is an physical machine operating in active mode, o_m is equal to one for PM m.

$$o_n \leq \sum_{m\in\mathbb{N}_{\mathbb{VM}}} l_{mn}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \quad (7)$$

$$l_{mn} \leq o_n, \forall m \in \mathbb{N}_{\mathbb{VM}}, n \in \mathbb{N}_{\mathbb{PM}}, \quad (8)$$

Constraint (7) and (8) together satisfy that a physical machine operate in active mode as long as in charge of supplying service to any virtual machine, or the physical machine is able to run inactively with no virtual machines to serve.

3.2.5 Objective Function

We define the energy consumption as a summation of the virtual machine execution energy, migration energy, active physical machine energy and sleep

physical machine energy, which are termed below respectively. The next expression presents the energy consumed as a whole, where *Period* is the predefined time period.

$$\begin{aligned}\min \mathcal{E} = & \sum_{m\in\mathbb{N}_{\mathbb{VM}}, n\in\mathbb{N}_{\mathbb{PM}}} (P_{mn}\cdot \mathit{Period}\cdot l_{mn}) \\ + & \sum_{m\in\mathbb{N}_{\mathbb{VM}}, n\in\mathbb{N}_{\mathbb{PM}}} (P_{mn}^{migrate}\cdot T_{mn}^{migrate}\cdot g_{mn}) \\ & + \sum_{n\in\mathbb{N}_{\mathbb{PM}}} (P_{active}^{n}\cdot \mathit{Period}\cdot o_n) \\ & + \sum_{n\in\mathbb{N}_{\mathbb{PM}}} [P_{sleep}^{n}\cdot \mathit{Period}\cdot (1-o_n)].\end{aligned} \tag{9}$$

Eq. (9) is the objective function optimizing the total energy consumption of the data center. P_{mn} denotes the required power level of the virtual machine m to operate on the physical machine n. The required power levels and time for migration are related to the resource usage, different from virtual machine to virtual machine. The idea is to find out how to place virtual machines onto physical servers in the way that energy cost is minimized. The optimization model takes advantage of the migration feature and consolidates virtual machines onto less number of active physical servers. For each virtual machine, its resource needs are satisfied to guarantee performance by aforementioned constraints. The resource utilization on physical machines is tend to be utilized fully so less servers need to be active, saving energy.

4 Approach to Solve the Problem

Since the formulated problem in Section 3 is ILP, which is generally NP hard, obtaining the optimal solution can result in an exponential increase in computation time as the problem size grow, underlying the issue of scalability. Therefore, as a countermeasure against such adversity, we develop a heuristic algorithm which has the goal of offering a suboptimal solution and ease computational intensity. On the other hand, in the considered energy-aware data center environment, because the number of nodes (i.e., physical and virtual machines) changes and resource requirements vary (i.e., CPU, memory, hard disk, and network bandwidth) frequently, a less expensive and time-consuming approach is desired to optimize energy consumption in response to the dynamics; hence, a heuristic approach is favored to be implemented for solving the proposed optimization model in practice. The pseudocode is shown in Algorithm 1, and is implemented using Java programming language. It takes the same input with what the optimization model takes into account; the algorithm decides the placement of virtual machine, virtual machine migration, and operation mode of physical machine. In the following, we discuss first the algorithm design principles and then the details about the heuristic.

Algorithm 1: Energy-saving VM Placement

Input: $\mathit{Period}, \mathcal{L}', \mathbb{N}_{\mathbb{VM}}, \mathbb{N}_{\mathbb{PM}}, \mathbf{P}, \mathbf{P_{active}}, \mathbf{P_{sleep}}, \mathbf{U^{CPU}}, \mathbf{U^{MEM}}, \mathbf{U^{HD}}, \mathbf{U^{BW}}, \mathbf{T^{migrate}}, \mathbf{H^{CPU}}, \mathbf{H^{MEM}}, \mathbf{H^{HD}}, \mathbf{H^{BW}}$ and $\mathbf{P^{migrate}}$.
Output: $\mathcal{L}, \mathcal{G}$ and $\mathcal{O}$.

```
Create resource utilization monitor;
L = L', G = 0, and given O;
STAGE 1: Feasible Solution Initialization
while The solution is not feasible do
    for PMs with high utilization do
        if The constraint is violated then
            for VMs utilizing more resources do
                Migrate VMs to PMs with lower utilization if constraints met;
            if The constraint still not satisfied then
                for VMs utilizing more resources do
                    for PMs with lower utilization do
                        Exchange VMs if constraints satisfied;
if A feasible solution not found then
    Adopt the alternative for operation and leave;
STAGE 2: Virtual Machine Consolidation
Create energy cost monitor;
repeat
    for Active PMs with lower total utilization do
        for Another active PMs with lower total utilization do
            Migrate all VMs on first PM to second PM;
            if Constraints violated then
                for Violated constraints with higher utilization do
                    for VMs utilizing more resources do
                        for Another active PMs with lower total utilization do
                            Migrate VMs from second PM to third PM if not exceed the maximum;
            if Constraints violated then
                No solution updated;
until The solution not improved.
return L, G and O.
```

4.1 Design Principle

Two stages are devised in the algorithm with separate objectives. In the first stage, we want to determine whether a feasible solution exist prior to performing virtual machine consolidation, which is the second

stage. Since the model specifies the maximum resource utilization of physical machines on each category (i.e., resource constraints (3)-(6)), the algorithm checks if the current operation (i.e., l'_{mn}) violates the constraints. If a constraint is violated, the algorithm starts to find a feasible solution by performing virtual migration. The objective in this stage is to obtain a feasible solution as an initial solution to begin virtual machine consolidation. If no such feasible solution can be found, the algorithm determines no consolidation is necessary, and an alternative will be adopted when the problem is infeasible. In the following stage, the objective is looking for a better virtual machine placement decision in terms of energy consumption. Considering the difference between the operation energy in active and sleep mode of physical machines, switching physical machines into sleep mode results in reducing energy consumption. With such idea, placing virtual machines to a less number of physical machines is necessary to produce a better solution. Therefore, the attempt is to reach a new solution with an improved energy consumption by consolidating virtual machines to physical machines.

The algorithm is designed to be executed whenever the context changes, for example, virtual machines are done with its tasks, or more virtual machines are requested to perform more tasks, showing the cases where input values have to be updated and the need of producing new solutions. However, the computation overhead might appear to be high, if the algorithm needs to perform constantly. To lower such overhead, *Period* can be used to control the time period to run the heuristic, instead of producing solutions dynamically. Overall, the computation complexity of the algorithm is $\mathcal{O}(MN(\log M)^2 \log N)$ in the worst case.

4.2 Algorithm Details

Before going into the first-stage of the algorithm, we create resource utilization monitors in order to keep track of the resource utilization status on physical machines. As the algorithm goes along, the these monitors are updated whenever a better solution is found. The utilization level is calculated based on the cumulative resource allocated to virtual machines on each physical machine, namely, $\mathbf{U^{CPU}}$, $\mathbf{U^{MEM}}$, $\mathbf{U^{HD}}$ and $\mathbf{U^{BW}}$. The current operation strategy is taken as the initial solution, i.e., $\mathcal{L} = \mathcal{L}'$, $\mathcal{G} = 0$, and $\mathcal{O}$ is given. The algorithm compares the monitored resource utilization levels with the maximum resource utilization values, i.e., $\mathbf{H^{CPU}}$, $\mathbf{H^{MEM}}$, $\mathbf{H^{HD}}$ and $\mathbf{H^{BW}}$. If any of constraints (3)-(6)) is not satisfied, the solution is not feasible, and a feasible solution needs to be found before entering the next stage of the algorithm.

In order to find a feasible solution, the heuristic tries virtual machines migration, reducing physical machine resource utilization. It first puts together and sorts utilization level in all categories of physical machines. Starting from the physical machine that consuming the highest utilized resource more than the capacity (can be CPU, memory, hard dish or bandwidth), on that physical machine we try two methods (line 4-7 and 8-11, respectively). Firstly, we find that on the physical machine, which virtual machines utilize more respective resource, and find destination physical machines with lower utilization level. The algorithm then calculates the new utilization levels for physical machines involved migration, e.g., source and destination physical machines. If after the above migration attempts, however, the constraint still cannot be satisfied, the heuristic will try the following migration strategy. Similarly, starting with virtual machines that utilize more respective resources, we find physical machines with a lower resource utilization level. Then virtual machines on the selected physical machine will be picked for migration. The algorithm intends to migrate virtual machines with higher resource utilization so that the opportunity of finding a feasible solution with less migration cost is higher. As the resource monitor keeps track utilization level, as long as all constraints are satisfied, the program can move into the second stage. However, after the procedure in the first stage, if no feasible solution can be found, we determine there is no feasible solution to work on for energy-saving procedure, and leave the routine.

The second stage of the heuristic serves the purpose of consolidating virtual machines in order to sleep physical machines, reducing energy consumption. Before performing the energy-saving procedure, we create the energy cost monitor to keep track of energy consumption of physical machines, migration and entire system. In addition to respective resource utilization, the total resource utilization of physical machine is summed up. Starting from physical machines having lower total utilization (first physical machine in line 17, second in 18), the algorithm migrates virtual machines on the first physical machine to the second one, in order to switch off the first. However, if a constraint is violated after this migration, virtual machines will be migrated to other physical machines until constraints are satisfied, or all other physical machines have been tried with no feasible solution reached. In the latter case, the solution will not be updated, and the next lower physical machines will be selected as a candidate machine to turn off. When multiple constraints are violated, the heuristic begins with the resource utilization that exceeds the allowed maximum the most, migrating virtual machines utilizing more respective resource to other low total utilization physical machines.

5 Testbed Implementation

We have built a real virtualized data center testbed to evaluate our work. Fig.3. provides an overview of

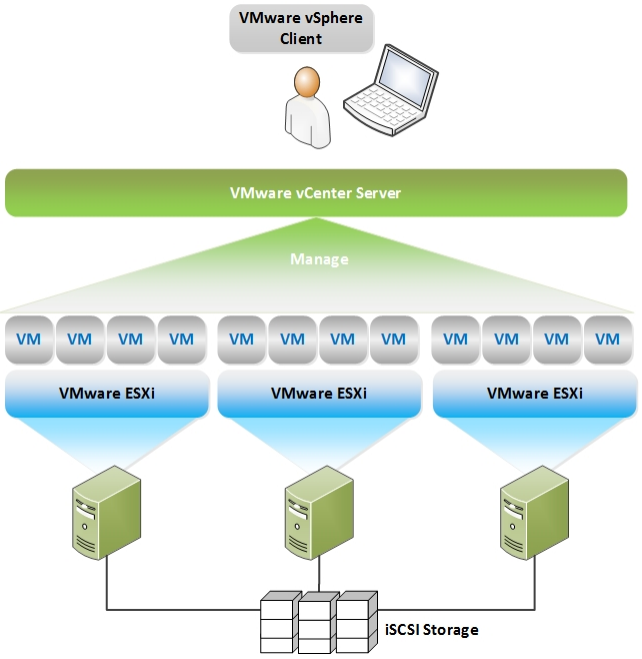


Fig. 3. Testbed Data Center Implementation

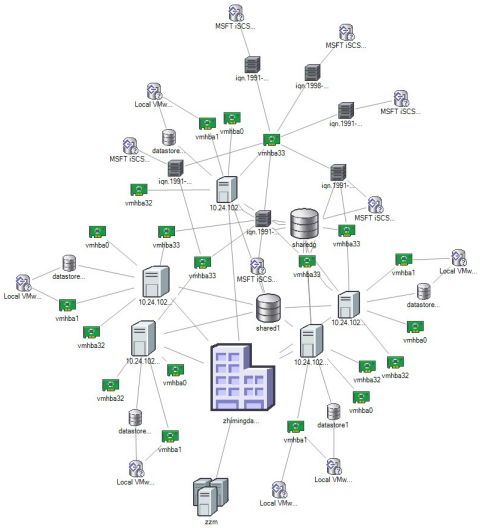


Fig. 4. Datacenter Topology

the implementation of the testbed. Currently we have two physical servers to host virtual machines, each configured with i7 3770 CPU, 32GB DDR3 memory. Each physical server is virtualized using VMware ESXi5 hypervisor. We have deployed 20 Ubuntu 12.04 lts Linux virtual machines in this data center. Each VM is given an ISCSi network storage and can be accessed by every physical server. A third server is used to host the heart of the data center VCenter, which controls all the VMs and hypervisors. VCenter is also responsible for sending out migration command once VM placement decisions are made. Two additional virtual servers are used to provide DNS, Active Directory Domain and network storage services. Fig. 4. shows the topology of the datacenter which gives an overview of connections among ESXi servers, shared ISCSi storages and vCenter, etc. The hardware and software setup of the testbed follows:

- vCenter Server 5.0: Intel Core i7-3770@3.40GHz, 4 GB RAM, runs 64-bit Windows 2008 Server R2.
- vCenter Database: Intel Core i7-3770@3.40GHz, 4 GB RAM, runs 64-bit Windows 2008 Server R2 and Microsoft SQL Server 2005.
- ESX 5.0 Servers: Intel Core i7-3770@3.40GHz, 32 GB RAM.
- Network: 1Gbps vMotion network configured on a private LAN.
- Storage: 2 1TB iSCSI storage hosted by 2 Windows 2008 Server R2, shared by all the hosts.

Live VM migration has become available several years ago on modern virtualization platforms, e.g., VMware, Xen, which further enhances the efficiency and flexibility of data center management. Live VM migration is a technology for moving an active VM from one physical host to another without shutting down the VM. Through this process, users will only feel little or even no interruption of their service. The actual VM downtime (usually in milliseconds) during live migration depends on the implementation. This downtime is considered the cost of migration, which might affect system performance thus must be taken account while designing dynamic VM placement algorithms [30]. For example, *pure stop and copy* usually suffers long downtime (tens of seconds) since this approach suspends the VM at the beginning of migration. The entire memory contents and architectural state is then transferred to the destination host. After this, the VM is reinstantiated and the applications resume.

Another migration mechanism is called $pre-copy$. This approach reduces service downtime considerably by transferring memory contents to the destination host while the VM continues to execute on the source host. This is an iterative process. During the period when the active memory of the VM is transferred to the destination, the copy of the VM that is still executing will dirty some of the transferred pages by rewriting on them. The hypervisor memory management unit tracks the dirty pages, which are then resent to the destination in subsequent migration iterations. The iterative process continues until a very small working set size is reached or until an iteration count limit is reached. At that point, the migration execution changes from the $precopy$ to $downtime$ phase, during which the VM is stopped and the remaining active memory of the VM and the architectural state, such as register contents, are transferred to the destination host. Since most of the memory of the VM has been transferred to the destination beforehand, the $downtime$ is typically minimal (in tens of milli-

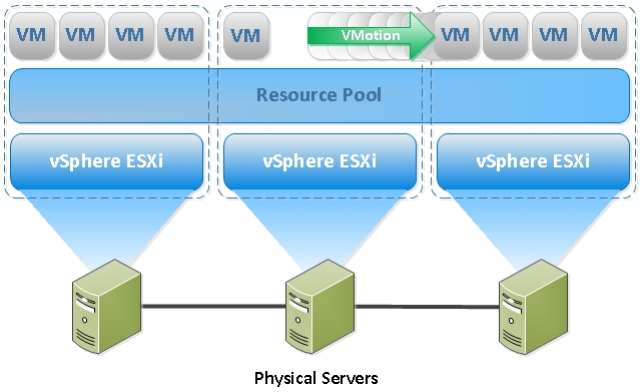


Fig. 5. Live Migration: vMotion

seconds). Today's most popular commercial products, e.g., VMware, Xen, and Kernel Virtual Machine are all based on precopy-based live migration. In this implementation, vMotion as shown in Fig. 5 is used to provide virtual machine migrations among ESXi servers.

5.1 Software Design

We implement our design in VMware Vsphere 5 using their API and SDK. Our software design includes three major components and communicates with Vcenter to manage the testbed data center. The first component is responsible of collecting runtime resource utilizations of each VM. These resources include CPU, memory, network and storage. This process is distributed to each VM where statistics are collected and sent to the data collector. The data collector then sends the VM statistics to the second component which is the optimization model mentioned in Section III that provides the optimal VM placement solution. The third component is the VM migration commander which is responsible of sending out VM migration commands based on the placement solution from the optimization model. This component communicates with Vcenter and places each VM onto the optimal server.

Migrating a VM from one PS to another PM requires extra CPU, memory and network bandwidth. We studied the live migration cost by migrating VMs to different servers in our data center testbed. Main observations are: 1. VM migration adds about an extra 10% to 20% CPU utilization; 2. the duration of migration mostly depends on the amount of VM memory size and network bandwidth. These observations suggest us to provide 10% CPU headroom for migration. The migration duration can be modeled as: $\Delta t = 10 + M(GB)$ (s) where Δt is the migration duration in seconds and M represents the VM's memory size in GB.

5.2 Comparison with VMware DRS and DPM

Distributed Resource Management (DRS) is designed by VMware to efficiently manage allocation of server resources to virtual machines in a cluster. As this is a commercial proprietary product, there is only limited information regarding the internal design. The core function of DRS is to provide dynamic load balancing of VMs across servers based on VMs' resource needs (CPU and memory). DRS uses a greedy hill-climbing algorithm to check if a VM migration could reduce the cluster Imbalance Score (Ic), which is defined as the standard deviation over all hosts. Migration would be applied if Ic can be reduced. This move-selection step is repeated until no additional beneficial moves remain which is a pre-defined Ic. DRS is invoked periodically (by default, every 5 minutes) to calculate the current Ic and determine whether migration is necessary.

Distributed Power Management (DPM) is an extended feature of DRS aiming to save energy. DPM periodically checks resource utilizations of each ESX server and recommends VM evacuation and powering off ESX servers if the resource utilization is lower than a threshold.

Compared to VMware DRS and DPM solution, our design is more concise and integrated. The objective of our design is to minimize cluster energy consumption and the constraints themselves will provide the load balancing features. Our design pro-actively checks for optimization opportunities triggered by VM workload fluctuations or new VM placements. The server resources considered in our design are not limited by CPU and memory, we also consider network and storage utilization, eliminating potential bottlenecks in these resources.

6 Experiment Result

We first evaluate the energy conserving capability of our optimization framework (the ILP design) to demonstrate that optimal dynamic VM placement can be achieved. Secondly, we demonstrate that the heuristic design is capable of achieving near optimal results. Thirdly, we use simulation to demonstrate that the heuristic design can scale well to large scale server clusters.

In order to thoroughly examine whether the dynamic VM placement decisions could effectively result in a balanced and energy aware data center, long-running and fluctuating workloads are required to trigger the VM migration. These workloads are generated using several popular web server benchmarks including: Apache ab, Phoronix Test Suite and RuBBoS [31]. While the benchmark programs are running, our dynamic VM placement software will keep monitoring the VMs and servers and make migration decisions when necessary. At the same time, we keep

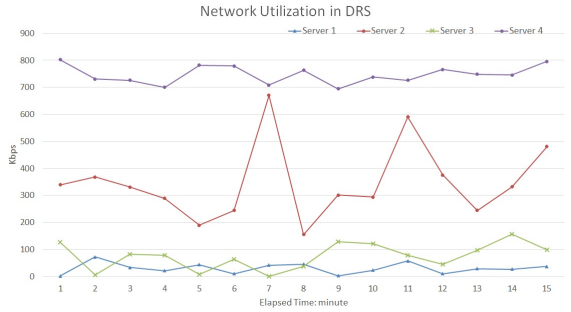


Fig. 6. Network Utilization in DRS

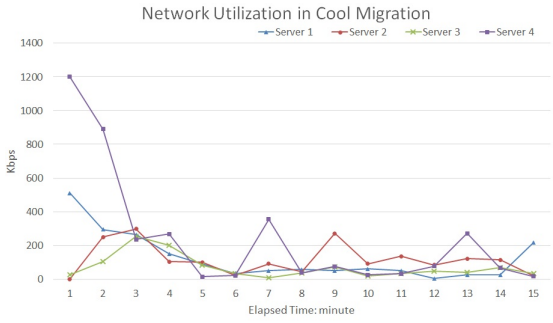


Fig. 7. Network Utilization in Cool Migration

record of each physical server's resource utilization and power consumption for a one hour period.

For the evaluation of energy saving capabilities of VMware DRS and our heuristic algorithm, the same testbed and workloads are used. We compare these three designs in regard of their abilities to balance workloads, server resources and their energy saving abilities. The results of network utilization, imbalance score and power consumption of each design are compared to demonstrate their overall performance.

The power consumption of each physical server is measured based on the work in [32], [33], where the full-system average power is approximately linear with respect to CPU utilization as given in eq.(10). It has proven to be an accurate way of measuring server power consumption especially in a data center environment where the total power consumption is an aggregation over a large number of servers.

$$P_{Total} = P_{Dynamic} \cdot U_{Avg} + P_{Idle} \tag{10}$$

where P_{Total} is the total power consumption of the server, $P_{Dynamic}$ is the dynamic power consumption of the CPU, U_{Avg} is the average CPU utilization and P_{Idle} is the power consumption when CPU is idle. In our experiment, all the metrics on the right side of eq.(10) are measured using the Intel Power Gadget [34].

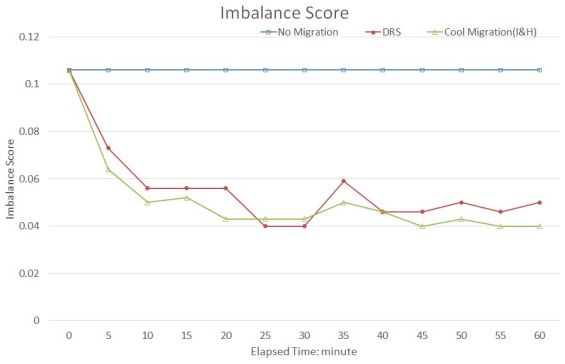


Fig. 8. Imbalance Score Comparison

6.1 Evaluation on Testbed

In the following comparison results, note that $Cool\ Migration$ is the proposed optimization design, where $Cool\ Migration(I)$ represents the ILP design and $Cool\ Migration(H)$ represents the heuristic design.

Fig. 6. shows the network utilization of each server while the testbed data center is managed by VMware DRS. As we can see there is big difference of network bandwidth consumption of each server. For example, within the 15 minutes time period, server 1 only consumes less than 100 Kbps of bandwidth. Server 4 on the other hand, consumes more than 700 Kbps of bandwidth. This is because DRS does not balance the network resource utilizations across servers. This is especially harmful when several VMs that all require high network bandwidth are placed on the same server.

On the other hand, Fig. 7. shows the network utilization of each servers while the testbed data center is being managed by our dynamic VM placement design. To demonstrate the effectiveness of our design and to save space at the same time, we only show the result for Cool Migration ILP, since the result for Heuristic is similar. The network utilization starts unbalanced with server 4 having heavy network traffic (1200 Kbps) and server 2 having very little network traffic (0 Kbps). Our optimization model quickly detects this imbalance and provides the optimal placement solution. In about 3 minutes, the migrations are complete and the network bandwidth consumptions are balanced across all servers. This demonstrates our design solves the unbalanced issue in DRS, eliminating potential network bandwidth bottlenecks.

Fig. 8. shows the imbalance score of the data center. Imbalance score is defined as the standard deviation of resource utilizations over all physical servers in a cluster. A lower score means a more balanced resource utilizations across all servers. $No\ Migration$ represents when no management software is applied to the data center, DRS represents when the data center is managed by VMware DRS and $Cool\ Migration$ is

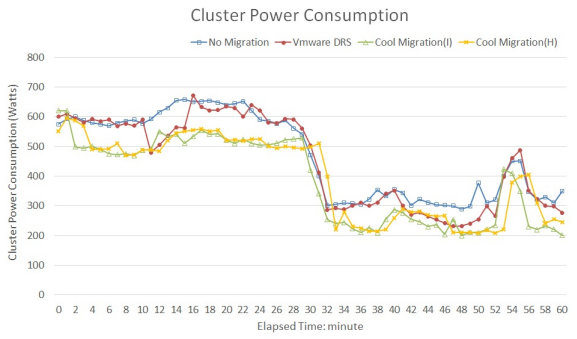


Fig. 9. Power Consumption Comparison

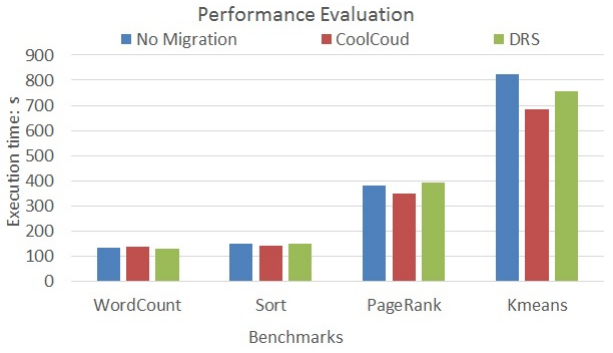


Fig. 10. CoolCloud Performance Evaluation

when our design is used to manage the data center. We purposely start the data center with an imbalanced fashion, i.e., several servers heavily loaded with VMs and other servers lightly loaded. The initial imbalance score is 0.106 and both DRS and our design are capable of bringing down the imbalance score to around 0.05 which is the target score for DRS. Notice that our design could achieve a lower imbalance score (0.04) since we do not set a target score but let the optimization model balance the data center as much as possible. This result demonstrates our design is capable of creating a more balanced data center than VMware DRS. Also note that in this experiment, the heuristic design delivers the same imbalance score as the ILP design in the whole run of the benchmark programs, this means the heuristic design is equally effective in balancing workloads in the cluster.

Fig. 9. shows the power consumption of the data center managed by $No\ Migration$, DRS, $Cool\ Migration(I)$ and $Cool\ Migration(H)$. Each case is monitored in a 60 minutes time period. The data center starts with the same workload and initial VM placement. The result shows both DRS and our design are capable of achieving energy savings. DRS can provide 15.5% energy savings on average compared to the case where no management software is used at all. $Cool\ Migration(I)$ and $Cool\ Migration(H)$ achieve 28.6% and 28.3% energy savings respectively when comparing with the case of no management software used, and this is over 15% gain of energy savings compared with DRS.

Both DRS and our design provide energy savings by turning off under utilized servers, however our design achieves more energy savings. This is because DRS mainly focuses on balancing CPU resource, and it only periodically (every 5 minutes) checks if any server is under utilized. This periodic checking may miss some energy saving opportunities due to the fluctuation of workloads. Further more, DRS does not provide the balancing of memory or network bandwidth across servers. This implies that some servers cannot be turned off due to resource wastage which leads to waste of energy. On the other hand, our design has an objective of minimizing energy consumption and all aspects of server resources are being considered. This creates a well balanced data center in regard of all resources, thus more servers can be turned off to achieve more energy savings. Notice that our design constantly monitors the server resource utilization in a pro-active fashion, thus responding quickly to the workload fluctuations and seizing every energy saving opportunities.

Figure 10 provides the performance evaluation of CoolCloud. In this experiment, we measure the execution time for four benchmark programs from HiBench, i.e., WordCount, Sort, PageRank and Kmeans. For WordCount, the execution time is about the same across all three configurations, i.e., 132s for $No\ Migration$, 138s for $CoolCloud$ and 130s for DRS. $CoolCloud$ requires slightly longer time to complete execution due to the overhead of live migration. However for PageRank and Kmeans (825s for $No\ Migration$, 684s for $CoolCloud$ and 756 for $Kmeans$), $CoolCloud$ demonstrates significant lower execution time compared to $No\ Migration$ and DRS. This is because $No\ Migration$ can not resolve the resource contention issue experienced by VMs, and DRS only reacts to this issue every 5 minutes. On the other hand, $CoolCloud$ is able to detect the resource contention proactively and respond quickly by initiating VM migration to resolve this issue. Note that the cost for live migration is fully considered in both the optimization model and the heuristic. The time duration for live migration typically ranges from several seconds to tens of seconds depending on the memory footprint of the VM. The small performance degradation comes from the live migration overhead and affects the applications performance running on that specific VM. CoolCloud prioritizes VMs' with smaller memory footprints for migration thus significantly reduces the overall migration overhead

6.2 Evaluation through Simulation

The evaluation result of the ILP and heuristic designs against the test bed data center has proven to provide

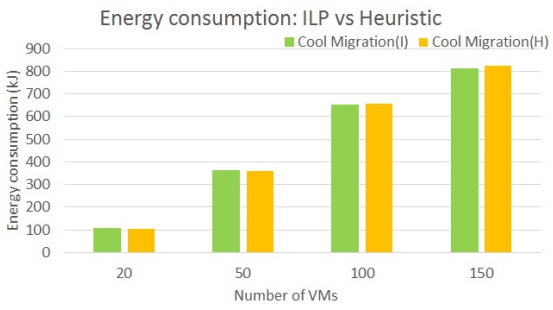


Fig. 11. Energy consumption for ILP and Heuristic

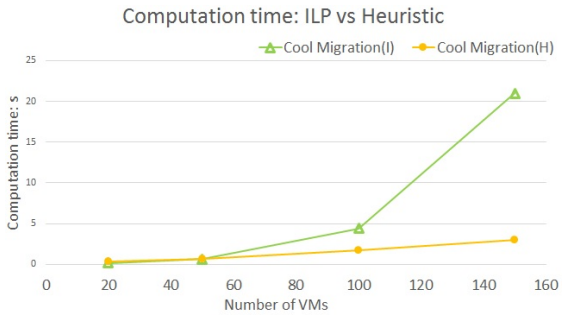


Fig. 12. Computation time for ILP and Heuristic

better energy conservations compared to VMWare's Dynamic Resource Planning design. In this section, we demonstrate that the heuristic design can be effectively applied to large-scale clusters to provide energy savings. To thoroughly evaluate the heuristic design, we designed an hybrid approach that combines profiling VM data from the test bed and simulating a large-scale cluster using the collected data.

6.3 Evaluation through Simulation

The evaluation result of the ILP and heuristic designs against the test bed data center has proven to provide better energy conservations compared to VMWare's Dynamic Resource Planning design. In this section, we demonstrate that the heuristic design can be effectively applied to large-scale clusters to provide energy savings. To thoroughly evaluate the heuristic design, we designed an hybrid approach that combines profiling VM data from the test bed and simulating a large-scale cluster using the collected data.

6.3.1 VM Profiling

The goal of VM profiling is to generate large numbers of VMs with runtime information and feed these VMs as inputs to the ILP and heuristic to evaluate their performance in respect of energy saving capability and computation complexity. The VM profiling is accomplished while running benchmark programs on the 20 VMs in the data center testbed. To accelerate the profiling process and not lose the fluctuation of workloads, each profile lasts 30 minutes. The profile includes VM's CPU, memory, network, harddisk utilization and power consumption footprint. Since each profile represents 20 VMs and requires 30 minutes to generate, we need 60 minutes to generate profiles for 40 VMs, 150 minutes for 100 VMs, 240 minutes for 160 VMs and so forth. In this paper, we generate profiles for 150 VMs in total and provide the simulation result in the following.

6.3.2 Performance Comparison of CPLEX and Heuristic Algoritm

The simulation for the ILP design and the heuristic design are carried out with the same system configuration: A server with 2 Intel Xeon x5650 CPUs which has 24 virtual cores, Red Hat Enterprise Linux Workstation 6.6 (Santiago) with 2.6 kernel, and the total amount of memory is 47 GB. The optimization ILP formulation is solved by IBM CPLEX 12.5.

Fig. 10 displays the energy consumption result for the ILP design and the Heuristic design when the number of virtual machines in the data center ranges from 20 to 150. In the case of 150 VMs, with the management of ILP, the data center energy consumption is 810kJ and this number is 828kJ for applying the heuristic design. This means the solution provided by the heuristic design only differs 2.3% from the optimal result. Overall, this result demonstrates that the heuristic design can provide solutions with only slight degradation on energy savings compared to the optimal ILP design.

Fig. 11 displays the computation time of ILP and Heuristic. In the case of 20 VMs, ILP and Heuristic take comparable time for calculation with 180ms and 375ms respectively. At the point of 50 VMs, the computation time is about the same with 630ms and 661 respectively. However when there are more than 50 VMs, the computation time for solving ILP grows dramatically as the number of VMs increases. In the case of 150 VMs, the computation time for ILP and Heuristic are 21s and 3s respectively. This result demonstrates that the heuristic design is highly computational efficient when it comes to large-scale clusters.

Overall, the simulation result shows that the heuristic design can provide near optimal solutions for energy savings and it is greatly computational efficient and highly scalable for practical large-scale data centers.

7 Conclusion

This paper presents a fine grained dynamic virtual machine placement framework to manage the mappings of VMs to physical servers. This framework solves the problem of finding the most energy efficient way (least resource wastage and least power consumption) of placing the VMs considering their

resource requirements. We formulate the problem as a ILP problem which is the core of the framework. We study the opportunity of energy minimization in data center while meeting a set of constraints. However, finding the best placement policy is expensive. Therefore, a heuristic method is designed to obtain a near-optimal solution with significantly less computation time. The proposed framework design includes three major components: runtime resource utilization collector, VM placement optimization model and the live migration commander.

What makes our design unique is that it is a complete solution that constantly monitors the workload changes over time in a data center and pro-actively provides VM placement solutions to guarantee performance and save energy. Our work considers all aspects of resources including CPU, memory, network and storage. It also takes into account of the cost of live VM migration which is often ignored in previous works. A real testbed data center implemented with industry product VMware vSphere 5 is used to evaluate the proposed framework. Experiment result demonstrates our design can effectively improve data center energy efficiency with each VM's resource requirement satisfied. The heuristic design is proven to provide high scalability for large scale server clusters. Our design performs better than VMware's Distributed Resource Scheduler (DRS) in respect of load balancing and power consumption.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments and feedbacks that have tremendously helped in improving the quality of this paper. We would also like to thank our colleagues and the ECpE department for their valuable support on our research.

REFERENCES

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010. technical report," *Analytics Press*, vol. (http://www.koomey.com/research.html), 2011.

[2] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: Integration and load balancing in data centers," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, Nov 2008, pp. 1–12.

[3] Y. Zhao and W. Huang, "Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, ser. NCM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 170–175. [Online]. Available: http://dx.doi.org/10.1109/NCM.2009.350

[4] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*, Dec 2010, pp. 89–96.

[5] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in *Web Information Systems and Mining*, vol. 6318. Springer Berlin Heidelberg, 2010, pp. 271–277.

[6] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010. [Online]. Available: http://dx.doi.org/10.1007/s13174-010-0007-6

[7] F. P. Tso, G. Hamilton, K. Oikonomou, and D. P. Pezaros, "Implementing scalable, network-aware virtual machine migration for cloud data centers," in *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, ser. CLOUD '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 557–564. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2013.82

[8] A. Gulati, "Towards proactive resource management in virtualized datacenters," *http://www.vmware.com*, 2013.

[9] A. G. Ganesha Shanmuganathan, "Vmware distributed resource management: Design, implementation, and lessons learned," *http://www.vmware.com*, 2012.

[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[11] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: Integration and load balancing in data centers," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, 2008, pp. 1–12.

[12] L. Grit, D. Irwin, A. Yumerefendi, and J. Chase, "Virtual machine hosting for networked clusters: Building the foundations for "autonomic" orchestration," in *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, ser. VTDC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 7–. [Online]. Available: http://dx.doi.org/10.1109/VTDC.2006.17

[13] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, 2010, pp. 179–188.

[14] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Mathematical and Computer Modelling*, no. 0, pp. –, 2013. [Online]. Available: http://www.sciencedirect.com

[15] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855610.1855620

[16] K. Mills, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, ser. CLOUDCOM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 91–98. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2011.22

[17] M. Cardosa, M. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, pp. 327–334.

[18] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 66–70.

[19] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1154–1162. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833690

[20] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.04.017

[21]

[22] M. Mazzucco and D. Dyachuk, "Optimizing cloud providers revenues via energy efficient server allocation," *Sustainable*

Computing: Informatics and Systems, vol. 2, no. 1, pp. 1 – 12, 2012.

[23] A. Verma, P. Ahuja, and A. Neogi, "pmapper: Power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, ser. Middleware '08. New York, NY, USA: Springer-Verlag New York, Inc., 2008, pp. 243–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496950.1496966

[24] K. Le, R. Bianchini, M. Martonosi, and T. Nguyen, "Cost-and energy-aware load distribution across data centers," *Proceedings of HotPower*, 2009.

[25] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '11. New York, NY, USA: ACM, 2011, pp. 233–244. [Online]. Available: http://doi.acm.org/10.1145/1993744.1993767

[26] R. Bianchini, "Leveraging renewable energy in data centers: present and future," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '12. New York, NY, USA: ACM, 2012, pp. 135–136. [Online]. Available: http://doi.acm.org/10.1145/2287076.2287101

[27] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 119–128.

[28] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, ser. VEE '09. New York, NY, USA: ACM, 2009, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/1508293.1508300

[29] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud, "Autonomic virtual resource management for service hosting platforms," in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, ser. CLOUD '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2009.5071526

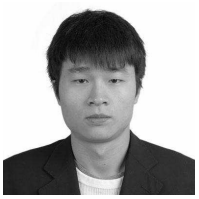
[30] C. C. Keir, C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, 2005, pp. 273–286.

[31] Phoronix Media, *Phoronix Test Suite*. http://www.phoronix-test-suite.com/.

[32] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007, pp. 13–23. [Online]. Available: http://doi.acm.org/10.1145/1250662.1250665

[33] D. Meisner and T. F. Wenisch, "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*, ser. ISLPED '10. New York, NY, USA: ACM, 2010, pp. 319–324. [Online]. Available: http://doi.acm.org/10.1145/1840845.1840911

[34] *White Paper. Measuring Processor Power*. Intel Corporation., April 2011.



Zhiming Zhang received the B.S. degree in computer engineering from University of Electronic Science and Technology of China in 2009. He is currently working toward the Ph.D. degree in computer engineering with the Department of Electrical and Computer Engineering, Iowa State University, Ames. His research interests include computer architecture and energy aware computing.

Chan-Ching Hsu received the B.S. degree in in Taiwan. He is currently working toward the Ph.D. degree in computer engineering with the Department of Electrical and Computer Engineering, Iowa State University, Ames. His research interests include wireless network and energy aware computing.

J. Morris Chang is an associate professor at Iowa State University. Dr. Chang received his Ph.D. degree in computer engineering from North Carolina State University. His industry experience includes positions at Texas Instruments, Microelectronic Center of North Carolina and AT&T Bell Laboratories. Dr. Chang's research interests include: Computer Architecture, Performance study of Java Virtual Machines (JVM), and Wireless Networks. Currently, he is a handling editor of Journal of Microprocessors and Microsystems and the Middleware & Wireless Networks subject area editor of IEEE IT Professional. He is a senior member of IEEE.