CoolCloud: A Practical Dynamic Virtual Machine Placement Framework for Energy Aware Data Centers

Zhiming Zhang
Iowa State University
Ames IA, USA
Email: zhiming@iastate.edu

Chan-Ching Hsu
Iowa State University
Ames IA, USA
Email: cchsu@iastate.edu

Morris Chang
Iowa State University
Ames IA, USA
Email: morris@iastate.edu

***Abstract*—With the continuing growth of cloud computing services, power consumption has become one of the most challenging issues in data center environments. With the support of today's virtualization technology, the efficiency and flexibility of data center management can be greatly enhanced, creating great energy saving opportunities. However, effective energy aware design is a non-trivial task, considering the size of the data center, the dynamic fluctuation of workloads and the variation of computing resource requests. In this paper, we propose CoolCloud: a practical solution for managing 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. Experiment result demonstrates our design can effectively improve data center energy efficiency and scales well to large size data centers. Comparing with industry leading product VMware's Distributed Resource Scheduler (DRS), our design offers better performance in both load balancing and power consumption.**

***Keywords*-Energy aware computing; virtualization; data center**

I. Introduction

Virtualized data center environment provides a shared hosting infrastructure to customers who need server resources to run their applications. All the applications run inside virtual machines (VMs) 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.

Virtualized data center management has brought a great amount of research interest. Most of the extant approaches [1], [2], [3], [4] 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. Note that some of the objectives may conflict with each other when not handled carefully, 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) left out.

In this paper, we tackle the above discussed challenges with the goal of designing a practical virtual machine placement framework that can be applied to real world enterprise data centers. We name our design $CoolCloud$ given its capability of cooling down the data center and providing a more energy efficient cloud. We formulate the VM placement problem into an ILP optimization problem with the objective of maximizing cluster energy savings. Due to that the optimization is NP hard, a heuristic approach is further proposed to reduce computation complexity and make our design scale well to the size of enterprise data centers. VM Live migration (with its cost considered) is used to move VMs from one server to another when placement decisions are made. A real testbed data center implemented with industry product VMware vSphere 5 is used to evaluate the proposed framework. The main contributions of our work are:

- Our optimization design can achieve maximum energy savings with all resource constraints (CPU, memory, network and storage) and VM live migration costs taken into account.
- Our framework is a practical solution that can be applied to enterprise data centers. The computation efficient heuristic design provides fast placement solutions given workload fluctuations.
- Our design is implemented and evaluated within a real testbed built from industry leading platform. Experiment result suggests that CoolCloud can effectively improve data center energy efficiency and is highly scalable for large size data centers.

The remaining of the paper is organized in the following sequence. We provide our VM placement optimization model in Section II. Section III introduces the heuristic

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.105

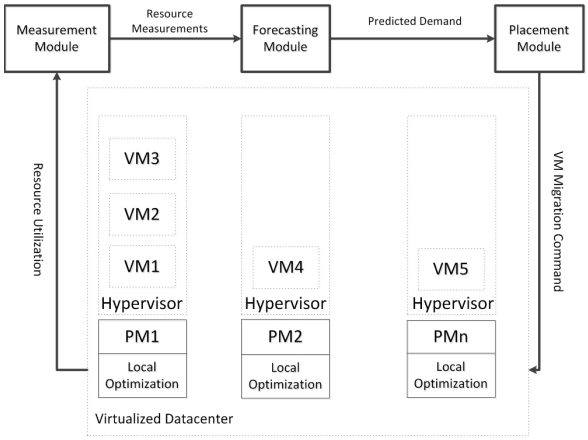

Figure 1. VM Placement Framework (CoolCloud)

design. The implementation of our design is provided in section IV. Section V demonstrates the experiment results. Related work is given in Section VI and Section VII concludes this paper.

II. System Model

The proposed CoolCloud design in Figure 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 an integer linear programming optimization model (a heuristic approach is proposed later for practical deployment) 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.

A. VM Placement Problem Formulation

The following sections presents the optimization model for the virtual machine replacement problem. The problem is to minimize the system energy consumption, denoted as $\mathcal{E}$, by deploying VMs in active mode on physical machines (PMs) 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.

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 practical to assume that a physical machine operateing actively ($o_m = 1$) consumes energy at a much higher level than in sleep mode.

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.

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

Live virtual machine migration is put into action if a virtual machine is decided to be placed on a physical machine n different from the one it is currently residing on before the optimal solution is provided. Namely, for a machine m, its next placement, $l_{mn} = 1$ and its current placement, $l'_{mn} = 0$, which is given 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}.

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

3) Resource Constraints: The service level agreements are categorized by four aspects: CPU, memory, hard disk and network bandwidth utilization. Essentially, the deployed resources of a physical machine cannot exceed a specified utilization level. In this paper, we assume if the resource constraint of each VM can be satisfied, the applications' SLA can be satisfied as well. The constraints are as follows, where U_m indicates the utilization of virtual machine m:

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

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

Table I
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), $\mathcal{O} = (o_n)_{1\times N}$

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

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

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 (25% headroom is given in our design). Energy can be consumed at diverse power levels and the execution time are different from virtual machines.

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 1 for PM m.

$$o_n \le \sum_{m\in\mathbb{N}_{\mathbb{VM}}} l_{mn}, \forall n \in \mathbb{N}_{\mathbb{PM}}, \tag{7}$$

$$l_{mn} \le o_n, \forall m \in \mathbb{N}_{\mathbb{VM}}, n \in \mathbb{N}_{\mathbb{PM}}, \tag{8}$$

Constraint (7) and (8) together satisfy the conditions that a physical machine operates in active mode if and only if it needs to host an active VM. Otherwise, the physical machine will be turned into sleep mode.

5) Objective Function: We define the energy consumption as a summation of the virtual machine execution energy, migration energy, active physical machine energy and sleeping physical machine energy, which are termed below respectively. The next expression presents the energy consumed as a whole, where *Period* is the predefined execution time period.

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

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.

III. HEURISTIC DESIGN FOR PRACTICAL DEPLOYMENT

The proposed ILP design provides optimal VM placement solutions, however it is NP hard and unpractical for large size data centers. We develop a heuristic approach which solves the formulated problem to avoid the exponential

growth in the computation time. The devised algorithm has the goal of offering suboptimal solutions and low computational complexity.

The pseudocode is shown in Algorithm 1, given the same input with what the model will take into account, and is implemented using Java programming language. The output includes the placement of virtual machines, l_{mn}, virtual machine migration, g_{mn} and operation mode of physical machines, o_n.

A. Algorithm Design Principle

The initial placement is taken by the algorithm as a preliminary solution to improve upon. The algorithm first looks for a solution which does not violate the resource constraints (3)-(6). Because of the substantial gap between the operation energy usages of active and sleep modes of a physical machine, turning physical machines to sleep mode when possible can save energy. The attempt is to seek a new solution with an improved energy consumption value by consolidating virtual machines to less physical machines so that physical machines that are originally active can be switched into sleep mode. The algorithm is devised to solve the problem in a timely manner so that suboptimal solutions can be reached to respond to the granularity of fluctuation of the workload. Overall, the computation complexity of the algorithm is $\mathcal{O}(MN(\log M)^2 \log N)$ in the worst case.

B. Algorithm Description

We divide the heuristic into two working stages, feasible solution initialization and virtual machine consolidation. At the first stage, the algorithm works on looking for feasible solution where all the constraints are satisfied. In the case of no constraint violations, the algorithm proceed to the next stage of the heuristic method, taking the initial placement as a feasible solution; or otherwise the initial placement causes one or more constraint violations, in which case, the initial placement is not viable as a feasible solution.

Making the attempt to find an effective solution, the heuristic essentially migrates around VMs onto different physical machines with the fundamental principle of reducing respective resource utilization by firstly, moving VMs requiring large amount of respective resource to another physical machine which is able to accommodate with sufficient resource. If the constraint is still violated, the algorithm begins a procedure that switches virtual machines to physical machines until either the constraints are all satisfied or no more moves can be made to produce a possible feasible solution, which usually means keeping the original placements.

The second stage of the heuristic serves the primary purpose of consolidating VMs in order to sleep more PMs, reducing energy consumption. The resource utilization of PMs are summed up to draw a comparison between origin to destination. PMs are chosen in ascending order as prospective candidates if the VMs residing on them could be potentially migrated to the remaining PMs. Once the candidate PMs are selected, the heuristic chooses one out of the rest of active PMs to host incoming VMs.

Then the VMs currently residing on a given candidate PM migrates tentatively to check whether the following conditions are satisfied: no resource utilization constraints are violated and the after-migration energy consumption is less than the initial placement energy consumption. As long as one of the conditions fails, the given solution will not be sufficient to improve with the tentative candidate consideration. Along with the consolidation process, when encountering a resource constraint violation, the algorithm will start from the resource exceeding the most ($\mathbf{H}^{\mathbf{CPU}}$, $\mathbf{H}^{\mathbf{MEM}}$, $\mathbf{H}^{\mathbf{HD}}$ or $\mathbf{H}^{\mathbf{BW}}$), and attempt to reach a solution without exceeding the limits down the road. All the energy terms in Eq. (9) are considered in this phase of the algorithm.

Algorithm 1: Energy-saving VM Placement

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

```
STAGE 1: Feasible Solution Initialization
while There exists a resource constraint violation do
    Perform virtual machine migration to
    find a feasible solution;
    if A feasible solution cannot be found then
        Adopt the alternative for operation;
        break;
    end if
end while
STAGE 2: Virtual Machine Consolidation
repeat
    Seek a better solution to consume
    energy at a lower level;
until The solution cannot be improved.
return L, G and O.
```

IV. Testbed Implementation

We have built a real virtualized data center testbed to evaluate our design and ensure it can be practically applied to real world data centers. Currently we have four physical servers to host virtual machines, each configured with i7 3770 CPU, 16GB 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 equipped with 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 manages 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. The following provides detailed information about the hardware and software setup of the testbed:

- 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.

We configure a Hadoop cluster built with Apache Hadoop 2.2.0 on top of our testbed data center to perform execution of MapReduce benchmarks. The cluster is composed of one master node and 20 computing nodes running Ubuntu 12.04. Each node has 4 GB of memory and 40 GB hard disk. The Hadoop configuration uses the default settings and runs with Oracle JDK 1.7. The estimation of resource usage of a specific VM is based on the VM's history resource usage as all applications have program phases [5] that last for a period of time. With the characterization of the workloads and benchmark suite used in this paper, the workload fluctuations range from seconds to minutes, which are actively monitored by the CoolCloud data center. In this design, we use one minute as the threshold for a stable program phase and the threshold for initiating VM migration/remapping.

V. 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 include: Apache ab, Phoronix Test Suite [6] and HiBench [7]. HiBench is a widely-used benchmark suite for Hadoop provided by Intel to characterize the performance of MapReduce based data analysis running in data centers. While the benchmark programs are running, our dynamic VM placement software will keep monitoring the VMs and servers to make migration decisions when necessary. At the same time, we keep record of each physical server's resource utilization and power consumption for a one hour period. We run all experiments three times and use the average as the result.

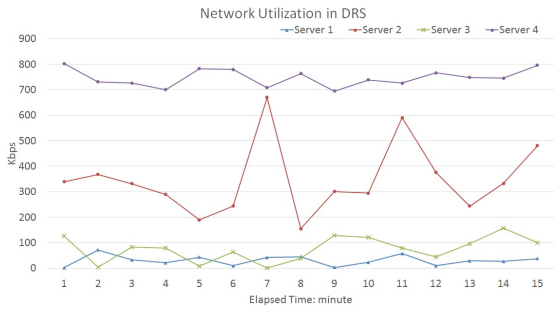

Figure 2. Network Utilization in DRS

For the evaluation of energy saving capabilities of VMware DRS [4] 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 [8], [9], 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.

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

In eq (10), 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 [10].

A. Evaluation on Testbed

In the following experiment result charts, note that $CoolCloud$ is the proposed optimization design, where $CoolCloud(I)$ represents the ILP design and $CoolCloud(H)$ represents the heuristic design.

Figure 2 shows the network utilization of each server while the testbed data center is managed by VMware DRS. As we can see there are big differences of network bandwidth consumption of each server. For example, within the 15 minutes examining 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

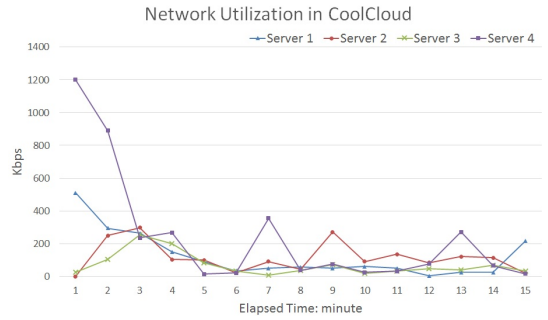

Figure 3. Network Utilization in CoolCloud

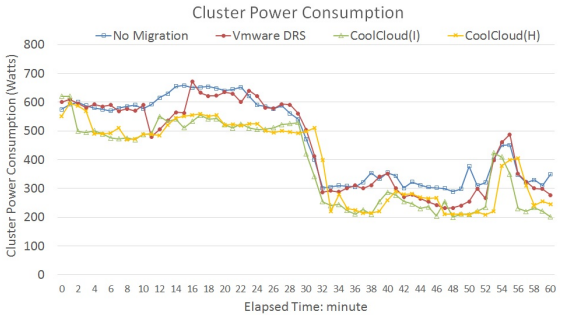

Figure 4. Power Consumption Comparison

across servers. This is especially harmful when several VMs that all require high network bandwidth are placed on the same server. This design flaw causes resource wastage: due to the bottle neck of one resource, other resources can not be fully utilized. For example, in the case of unbalanced network utilization, if a PM runs out of network bandwidth, even if it still has large amount of remaining CPU or memory resource, it is unlikely to accommodate any more VMs.

On the other hand, Figure 3 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 CoolCloud 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.

Figure 4 shows the power consumption of the data center managed by $No\ Migration$, DRS, $CoolCloud(I)$ and $CoolCloud(H)$. Each case is monitored in a 60 minutes

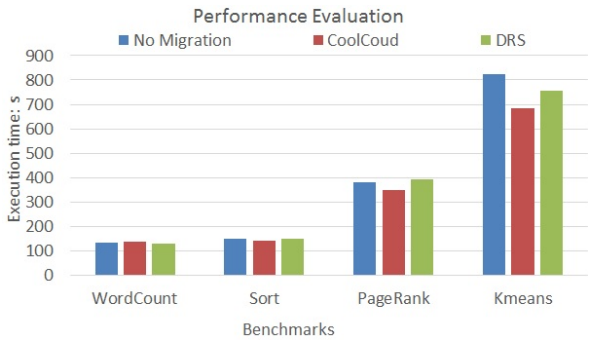

Figure 5. CoolCloud Performance Evaluation

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 power savings. DRS can provide 15.5% power savings on average compared to the settings where no management scheme is used at all. $CoolCloud(I)$ and $CoolCloud(H)$ achieved 28.6% and 28.3% power savings respectively when comparing with the case of no management scheme used, and this is over 15% gain of power savings compared with DRS. The power consumption measured here is the result of taking all costs including the cost of live migration into consideration.

Both DRS and our design provide power savings by turning off under utilized servers, however our design is capable of achieving the maximum power 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 proactive fashion, thus responding quickly to the workload fluctuations and seizing every energy saving opportunities.

Figure 5 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

B. 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.

1) VM Profiling: The goal of VM profiling is to generate large numbers of VMs with runtime information and feed these VMs' runtime info 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 1000 VMs in total and provide the simulation result in the following.

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.

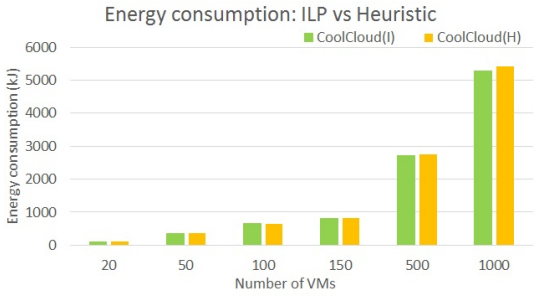

Figure 6. Energy consumption for ILP and Heuristic

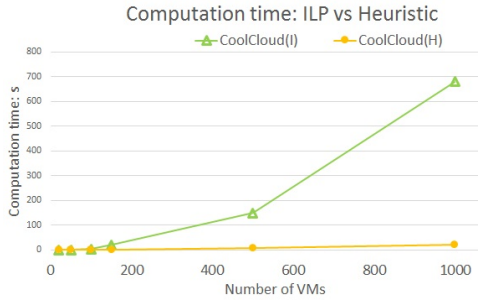

Figure 7. Computation time for ILP and Heuristic

Figure 6 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 1000. In the case of 1000 VMs, with the management of ILP, the data center energy consumption is 5280kJ and this number is 5401kJ 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.

Figure 7 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 1000 VMs, the computation time for ILP and Heuristic are 680s and 22s 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 highly computational efficient making it a practical solution for large-scale data centers.

VI. Related Work

Earlier work mostly focuses on improving resource utilization and load balancing of VMs across physical servers [7]. Timothy et al. [7] 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 specific 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.

Static VM placement methods [11], [12] are effective for initial VM placements. However, these approaches do not consider future dynamic workload changes that may need VM remappings. Jing et al. [11] propose a multi-objective virtual machine placement algorithm that simultaneously minimize power consumption, resource wastage and thermal dissipation. Xin et al. [12] also considers physical resources as multi-dimensional and propose a multi-dimensional space partition model to determine the mapping of VMs and PMs.

Furthermore, Bobroff et al. [13] uses first-fit approximation to solve the VM placement problem focusing on CPU utilization. Fabien et al. [14] formulate VM placement into a constraint satisfaction problem to minimize the number of physical machines. This work considers uni-processor computers and assumes each PM can only host one active VM. Hien et al. [15] extends [14] and considers CPU and RAM as constraints.

VII. Conclusion

This paper presents a dynamic virtual machine placement framework to manage the mappings of VMs to physical servers. This framework tackles the problem of finding the most energy efficient way (least resource wastage and least power consumption) of placing the VMs considering their fluctuating workloads and resource requirements. With all resource constraints and migration cost taken into account, the design is implemented and evaluated against a real testbed. It is proven that CoolCloud is highly scalable and can be practically applied to enterprise data centers for great energy efficiency.

References

[1] 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.

[2] 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.

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

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

[5] Z. Zhang and J. Chang, "A cool scheduler for multi-core systems exploiting program phases," *Computers, IEEE Transactions on*, vol. 63, no. 5, pp. 1061–1073, May 2014.

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

[7] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The hibench benchmark suite: Characterization of the mapreduce-based data analysis," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, March 2010, pp. 41–51.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.