International Journal of Knowledge and Language Processing
Volume 4, Number 2, **2013**

www.ijklp.org
KLP International ©2013 ISSN 2191-2734
pp.12–22

A Novel Energy Efficient Algorithm for Cloud Resource Management

Jing SiYuan
School of Computer Science
Leshan Normal University, Leshan, 614000, China
Jingsiyuan_628@126.com

Received April 2013; revised April 2013

ABSTRACT. *The size of cloud data center is growing rapidly. The larger the data center is, the more energy it consumes. Therefore, more and more researchers focus on energy efficient resource management techniques in the data center. The most popular way is a dynamic on-demand resource provision which allows turning off part of idle servers to save energy. But numerous part of the relevant research just considers how to maximize the resource utilization, i.e. minimize the required servers, without considering the overhead of a virtual machine (abbreviated as a VM) placement change. In this work, we propose a new method to minimize the energy consumption and VM migration at the same time; moreover we also design a network-flow-theory based approximate algorithm to solve it. The simulation results show that, compared to existing work, the proposed method can slightly decrease the energy consumption but greatly decrease the number of VM placement change (nearly 75%).*

Keywords: Cloud computing; Virtual machine; Energy efficiency; Resource management; Approximate optimization

1. **Introduction.** Recently, cloud computing [1] has attracted considerable attention and is believed to become one of the most important future computing and service paradigm. Everything in cloud computing is regarded as a service, such as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service), etc. These cloud-based services integrate globally distributed resources into seamless computing platforms and make them available on a subscription basis using pay-as-you-use model to customers, regardless of their locations. This new paradigm brings growing demand for high performance computing infrastructures and leads to construction of large-scale computing data center, called cloud data center (abbreviated as CDC).

CDC has several key features which should be considered in efficient resource management. (1) CDC is large scale. A typical CDC commonly hosts tens of thousands of servers and serves hundreds or even thousands of web-based application service (e.g. SaaS) at the same time. (2) Resources (e.g. CPU, memory, disk, bandwidth, etc.) in CDC are virtualized. Virtualization technique abstracts lower-level server hardware resource and

provides on-demand resource for the upper-level services which are capsulated in virtual machines (VM). Moreover, modern VM software, like Xen [2] and VMware [3], supports live migration which allows VM instance migration from one server to another. (3) Service level agreement (SLA) cannot be violated. The cloud computing system must provide reliable QoS which can be defined in terms of SLA that describes such characteristics as minimal throughput, maximal response time or latency delivered by the deployed system.

It is obvious that such a complex system cannot be efficiently managed by manual operation and the most popular way is an adaptive resource management technique [4]. One of the important steps in such technique is dynamical resource provisioning which periodically monitors the data center and collects data including workload, QoS etc. And estimate the workload of each service in the next time period and then make a decision about VM resource reconfiguration and placement change [5].

In this work, we assume that the workload of each service in the next time period have been estimated and just explore an efficient method to decide the solution about re-allocating resources to each VM meanwhile changing the placement of VM. Two aims are considered in our solution. The first is minimizing energy consumption because high energy consumption not only translates to high energy and maintenance cost, which will reduce the profit margin of IaaS providers, but also high carbon emissions which is not environmentally sustainable [4]. The second is minimizing the number of VM placement change. This is due to that no matter VM live migration or VM start/stop leads to extra resource overhead and system performance degradation [2] [3].

The rest of this paper is organized as follows. Section II introduces the related works. Section III models the system and formulates the problem. Section IV introduces the designed algorithm. We explain the simulation and performance evaluation results in section V. The paper is finally concluded in Section VI.

2. **Related Works.** Recently, the technique of adaptive resource management in the cloud data center or virtualized data center attracts many researchers working on this hot topic.

X. Y. Sun et al. [5] propose an integrated architecture for efficient resource management in virtualized systems. The method handles automated capacity and workload management at three different scopes, i.e. (1) on the shortest time scale (second), node controllers dynamically adjust resource allocations to the VM which aims to aims to satisfy the SLO (Service Level Objective) of individual applications (or service). (2) On a longer time scale (minutes), pod controllers manage pods by adjusting the placement of workloads on nodes within a pod; (3) On the longest time scale (hours), pod sets controllers study the resource consumption history of many workloads and determines whether the data center has enough resource capacity to satisfy workload demands, places compatible workloads onto nodes, and group nodes into pods.

A similar idea is proposed by A. Beloglazov and R. Buyya in [6]. They design three-level architecture to meet the requirement of large scale resource management. The local-manager node is responsible for making decisions about resource allocation and VM placement change. In our work, we adopt this model.

K. H. Kim et al. explore the problem of power-aware allocation of VM in DVS-enable cloud data centers for application services based on user QoS requirements such as the deadline and budget constraints [7]. In their research, the system is composed of a set of cloud data centers. When a user submits a request to the system, a global broker would ask each data center to calculate a price of provisioning resource (i.e. VM). Then the broker chooses the data center which provides the lowest price to provide service. Three policies are proposed for scheduling real-time VM to reduce the energy consumption, while meeting deadline constraints and maximizing the acceptance rate of provisioning requests, called Lowest-DVS, δ-Advanced-DVS and Adaptive-DVS respectively.

G. Laszewski et al. [8] proposed a power-aware algorithm for scheduling virtual machine in a DVS-enable compute cluster. They regard the virtual machine as a group of requests and model it in terms of required processor frequency and required executing time, just as most of the power-ware tasks scheduling works, and schedule VM from high frequency server to that which frequency is low to achieve load balance. However, these works only consider the dynamic power but ignore the fixed energy consumption.

D. Kusic et al. [9] implement and validate a dynamic resource provisioning framework for virtualized server environments wherein the provisioning problem is posed as one of sequential optimization under uncertainty and solved using a lookahead control scheme. Their objective is to maximize the service provider's profit by minimizing both energy consumption and SLA violation. Switching servers on / off as well as resizing and dynamic consolidation of VM via VM migration are applied as power saving mechanisms. Two key challenges are considered in the model, i.e. (1) quickly changing workload and (2) the cost for switching hosts and VM on/off. However, the proposed model requires simulation-based learning for the application specific adjustments. Moreover, due to the complexity of the model the optimization execution time reaches 30 minutes even for a small experimental setup (15 servers), which is not suitable for large-scale real-world systems.

Y. C. Lee and A. Y. Zomaya [10] proposed two energy-conscious task consolidation heuristics, which aim to maximize resource utilization and explicitly take into account both active and idle energy consumption.

In [11], A. Verma1 et al. propose a model for VM placement considering energy and migration cost. Different with this work, we do not estimate the cost of VM migration because it is complex and cannot be accurately pre-defined [2].

3. **System Model and Problem Formulation**

3.1. **Control loop for self-adaptive resource management.** The control loop is widely used in resource management of data center [5]. Such type of period controlling method can continually optimize the system performance on the basis of the current state of the system. Each round of the controlling is comprised of four sub-phases. The first phase is that it continually collects the information data of the system, commonly including service workload, quality of service and resource utilization of each server etc. This information will be used for making decisions of resource management. The second phase is that it

forecasts workload of each service during next controlling round [12]. The methods which perform well in this work include Kalman filter and data regression. The third phase is analyzing the information and calculating the new solution for resource allocation and VM placement. It aims to optimize the resource provisioning without SLA violation. This work has two objectives of optimization, i.e. reducing the entire energy consumption and the VM placement change. The VM placement changes are caused by VM creation, migration and destroy. The final phase is command execution.

3.2. **Energy model.** A recent study [13] has indicated that the energy consumption of server is approximately linear to the CPU utilization. This result is applicable in both DVFS enabled server and DVFS disabled server. The reasons behind this are (1) the modern server is equipped multi-core processor or multi-processor thus the CPU is the major energy consumer in servers; (2) although CPU can adjust its frequency to lower the power consumption, the number of frequency's states is limited moreover DVFS is not applied to other server components apart from the CPU.

Furthermore, the study has shown that on average, an idle server consumes approximately 70% of the power consumed by the server running at the maximal speed. This result proves that dynamic power management technique is effective which set idle servers to the sleep mode to reduce idle power.

In this work, we use the power model [13] defined in (1)

$$P(u) = k \cdot P_{\max} + (1-k) \cdot P_{\max} \cdot u \tag{1}$$

The parameter $P(u)$ denotes current power consumption, $P_{\max}$ is the maximal power consumed when the utilization is maximal, u is the resource utilization and k is the fraction of power consumed by idle server.

Based on the power model, we can define the energy model in this work (see formula (2)).

$$E = \int_{t_0}^{t_1} P(u(t))dt \tag{2}$$

3.3. **Problem Formulation.** In section 3.1, we have introduced the control-loop-based dynamical resource management technique. Our work just focuses on the third step, namely consider that workload of each service during the next time period has been forecasted then re-allocate resources to services. Moreover, we consider that if current resource cannot satisfy the requirement, we need to increase the total capacity (i.e. turned on servers) and if the resource is abundant, we also consider turning off part of them to save energy. In addition, the accuracy of forecasting is not considered in this work.

The work considers that there are service and n active servers in the system. The resource allocation solution is described by matrix L where $L_{i,j}$ denotes the resource allocated to service i on server j. We use matrix denote the placement of service's VM instance. It is obvious that the matrix I is dependent on the matrix L (this will be explained in later problem formulation). Furthermore, we assume server in the system is homogeneous and define the resource capacity of each server is C and the resource

required by service i during next time period is R_i. In addition, we should consider the basic resource used for VM running. This is denoted as γ.

The aim is to find a new resource allocating solution which can minimize the total energy consumption and minimize the placement change of services' VM instances (including VM start/stop/migration) meanwhile satisfy all the constrains, e.g. the resource requirement of services must be satisfied and the resource allocation on each server cannot exceed the resource capacity.

Before the mathematically formulate the problem, we need to give some explanation: (1) we assume the resource (i.e. server) is infinite, this is reasonable because the size of current data center is huge and the resource management framework adopted by this work support dynamically scale the domain which is the second -level managing environment. (2) We just consider CPU in this work and it can be easily extent to multi-dimension resource.

The problem is formulated as follows:

$$
\begin{aligned}
&(\mathrm{I}) \quad \min \ \textstyle\sum_{j=1}^{n} E_j \\
&(\mathrm{II}) \quad \min \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n}\left|I_{i,j} - I^{*}_{i,j}\right| \\
&s.t. \\
&\forall i, \forall j, \ \ I_{i,j} = 0 \text{ or } I_{i,j} = 1 \\
&\forall i, \forall j, \ \ I_{i,j} = 0 \Leftrightarrow L_{i,j} = 0 \\
&\forall i, \forall j, \ \ I_{i,j} = 1 \Leftrightarrow L_{i,j} > \gamma \\
&\forall i, \forall j, \ \ L_{i,j} = 0 \text{ or } L_{i,j} > \gamma \\
&\forall i \qquad \textstyle\sum_{i=1}^{m} L_{i,j} \le C \\
&\forall j \qquad \textstyle\sum_{j=1}^{n}\left(L_{i,j} - \gamma\right) = R_i \\
&\forall j \qquad \textstyle\sum_{j=1}^{n} I_{i,j} \ge 1 \\
&i \in [1..m], \ j \in [1..n]
\end{aligned}
\tag{3}
$$

Here, the symbol $I^{*}_{i,j}$ denotes current placement. The first objective denotes minimization of energy consumption. The second objective denotes minimization of placement change of services' VM instances. Constraints 1~4 denote the relationship between matrix I and L moreover the resource allocated to each VM instance must be higher that a given basic resource requirement γ. This constraint also avoids too much VM instances running on a same server. Constrain 5 means the sum of allocating resource on each server cannot exceed the capacity. Constraint 6 means the allocating resource apart that for VM running must satisfy the service's resource requirement. The final constraint keeps each service has at least one VM instance in the system.

4. **Algorithm.** We can see that the problem is a two-objective combinatorial optimization

problem with multiple constraints. Such types work has been proven NP hard and we cannot find the optimal solution in polynomial time. In this work, we study an existing works [14] [16] [17] and propose an approximate algorithm which iteratively optimize the solution based on a max-flow and min-cost-flow method (called NFT-DRP).

The algorithm is described as follows.

Algorithm : **N**etwork-**F**low-**T**heory-based **D**ynamical **R**esource **P**rovisioning (**NFT-DRP**)

0. Input: A set of servers, current VM placement matrix $\boldsymbol{P}$, resource demand $\boldsymbol{D}$
1. Output: resource provision solution $\boldsymbol{R}$
2. | **N =** estimate_upper_bound_of_required_servers($\boldsymbol{D}$);
3. | $\boldsymbol{P}$=select_a_set_of_servers(**N**);
4. | while true
5. | | $\boldsymbol{R}$ = search_resource_allocation_based_on_max_flow($\boldsymbol{P}$, $\boldsymbol{D}$);
6. | | if satisfy_the_demand($\boldsymbol{R}$) == true break;
7. | | $\boldsymbol{R}$ = optimize_solution_based_on_min_cost_flow($\boldsymbol{R}$);
8. | | $\boldsymbol{R}$ = allocate_resource to_residual_demand_by_heuristic ($\boldsymbol{R}$, $\boldsymbol{D}$);
9. | | if satisfy_the_demand($\boldsymbol{R}$) == true break;
10. | | $\boldsymbol{P}$ = cancel_an_arc_from_graph($\boldsymbol{P}$);
11. | end
12. | turned_off_unused_servers($\boldsymbol{R}$);
13. | return $\boldsymbol{R}$;

At first, the algorithm calculates the upper bound of required servers. Existing work of bin-packing problem has indicated that heuristic method (e.g. FFD or BFD) uses no more than $OPT \cdot 11/9 + 1$ bins (where OPT is the number of bins given by the optimal solution) [15].

Then, the algorithm selects a set of servers for resource allocation. If required servers is more than current servers, the system will turn on (N-N') servers which are in sleep mode or other energy saving mode (N means required servers and N' means current running servers). If required servers are less than current servers, the system will turn off (N'-N) servers to save energy. The server selection follows an intuitive idea which is shown in the formula below.

$$
\begin{aligned}
& a \cdot \frac{\sum_{i=1}^{m} service\_value_i \cdot I_{i,j}}{\sum_{i=1}^{m} I_{i,j}} + b \cdot server\_value_j \\
& service\_value_i = \frac{service_i - service_{\min}}{service_{\max} - service_{\min}} \\
& service_i = resource\_demand_{i,new} - resource\_demand_{i,old} \\
& server\_value_j = \frac{server_j - server_{\min}}{server_{\max} - server_{\min}} \\
& server_j = \sum_{i=1}^{m} I_{i,j}
\end{aligned}
\tag{4}
$$

We can get an appraisal value for each server by this formula. It is comprised of two

parts. The left part estimates the potential of services which are hosted in server j. High potential means it requires more resource in the next period so we cannot turn off the corresponding server to increase the efficiency in next step meanwhile decrease the chance of VM placement change. The right part calculates how many VM hosted on server j. The more VM are, the lower the value is. It can decrease the number of VM placement change. In addition, parameter a and b represent the weight of two parts.

After the selection, algorithm enters into iteration and it will return if all the demand is satisfied. In each round, the algorithm search an initial solution for resource allocation based on current VM placement matrix P and resource demand D. This progress is based on the maximum flow method (we adopt the basic Ford-Fulkerson algorithm in this work). Then, we adopt a good idea from Tang's work [15] which use the minimum cost flow to let free resource centralized which can make the algorithm more efficient. After that, we apply a FFD heuristic method to dispatch the residual resource demand for suitable servers. If there is no residual resource demand in the system, namely all the resource demand is satisfied, the algorithm returns the solution, else it will cancel an arc from the graph (it means remove a VM from current system) and compete again. The worst case is that the flow degenerates to the simplest, namely there is no edge in between the service nodes and server nodes.

5. **Experimental Results and Discussion.** This work adopts the resource management model proposed in [6]. Thus each local manager is responsible for dynamic resource provision and VM migration/start/stop. As the targeted environment is an IaaS, a Cloud computing environment that is supposed to create a view of infinite computing resources to users, it is essential to evaluate the proposed strategy on a large-scale virtualized data center infrastructure. However, it is extremely difficult to conduct repeatable large-scale experiments on a real infrastructure, which is required to evaluate and compare the proposed algorithm to different algorithms with the same conditions. Therefore, a simulation has been chosen to evaluate the proposed method. We design a series of experiments in simulation. In these experiments; first of all, we test the feasibility of the proposed method by two groups of different resource request. Secondly, we compare the performance of the proposed method results with existing works. Finally, we test the computational overhead by groups of data with different size. We adopt Matlab 2012a to perform the simulation and the machine is equipped with Intel i5-2300 CPU 2.8GHz, 3G RAM.

5.1. **Availability.** First of all, we plan to test the availability of the proposed method with two groups of data (see figure 1 and 2). These two kinds of data are randomly generated and the difference between them is; the resource demand in the first group of data is not severely fluctuated and the second group of data is contrary to first group of data i.e. resource demand fluctuates dynamically. In this experiment, we compare the proposed method with an existing heuristic method in literature [6]; in this author just consider the energy efficiency.

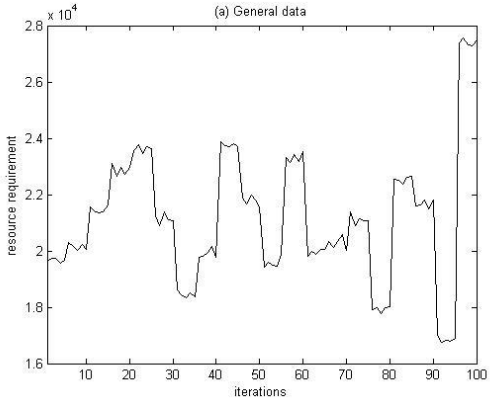

FIG. 1 DATA SET (A)

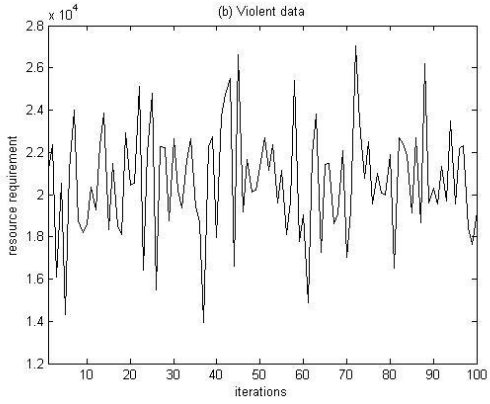

FIG. 2 DATA SET (B)

The experimental results are shown in figure 3~6. It can be seen from Figure 3, energy consumption of three methods by using a first group of data (In this experimental result we don't used any energy saving module in system thus servers will not be turned off even if they are idle) and figure 4 shows energy consumption of three methods by using a second group of data. It can be seen from the figure that our method consumes slightly less energy than heuristic method in both experiments.

Figure 5 and 6 show the results of VM placement change's time of two methods (our method and heuristic). It can be seen from figures that heuristic method generates result far more VM placement change than our method (about 5 times in the first experiment and 2.5 times in the second experiment). Heuristic's VM placement change curve is similar to the data's curve, namely VM placement change time depends on the total resource demand on the system. On the contrary with heuristic, VM placement change in our method is affected by specific data. We explain this conclusion by analyzing the data. In our experiment, the data (i.e. Resource demand of each service) follows a Poisson distribution which is reasonable and the average value follows uniform distribution. The difference between two data groups is the time interval of average value change, e.g. the first group changes average value every 5 rounds and the second one changes every 2 rounds. Therefore, we can be seen, that the VM placement change time has sharply grown every 5 rounds. Moreover, in second experiment, due to the resource demand of each service change quickly, the mean value of VM placement change time increases. This result is consistent with our prediction. It can be believed that, in a real system, the VM placement change may be much less than the designed experiment because, most of the time, the changes of services' resource demand does not greatly fluctuate.

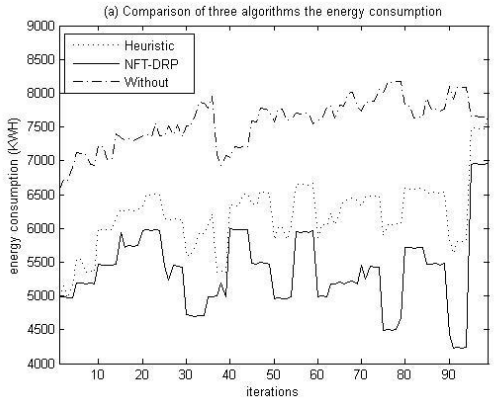

FIG. 3 COMPARISONS OF ENERGY CONSUMPTION (A)

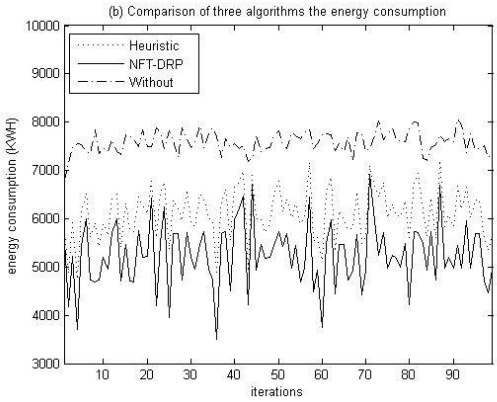

FIG. 4 COMPARISONS OF ENERGY CONSUMPTION (B)

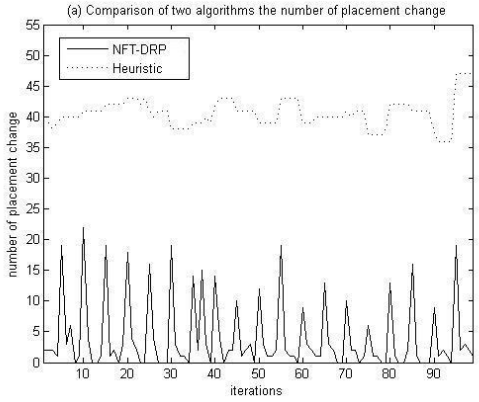

FIG. 5 COMPARISONS OF VM PLACEMENT CHANGE(A)

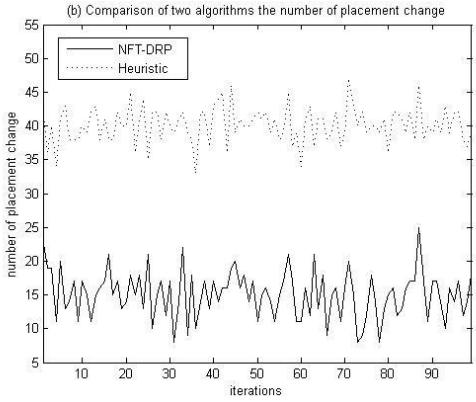

FIG. 6 COMPARISONS OF VM PLACEMENT CHANGE(B)

5.2. **Efficiency.** We compare the proposed method with heuristic with respect to the difference in energy efficiency, and as well as the VM placement change. We achieve this aim by applying ten groups of data with different size. The number of services in data increases from 10 to 100. The experimental results are shown in figure 7 and 8. It can be seen clearly from the figures, the proposed method's energy saving efficiency is better than heuristic method. On the other hand, in the VM placement change, the proposed method is much better than heuristic. When we used just 10 services in experimental data, then average VM placement change number of our method is 2, while this number is 25 for heuristic. Furthermore, when there are 100 services in experimental data, then average VM placement change number of our method is 51 and for the heuristic this number is nearer to 200.

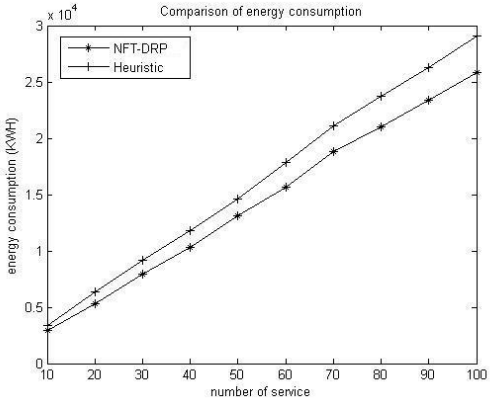

FIG. 7 COMPARISONS OF ENERGY CONSUMPTION

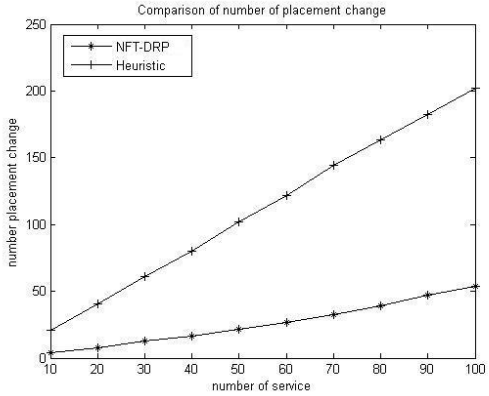

FIG. 8 COMPARISONS OF VM PLACEMENT CHANGE

5.3. **Computational overhead.** Finally, we test the computational overhead of the proposed algorithm. The simulation is carried out 12 times and the data size (number of services) is incremental in each round, i.e. 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200. The results are shown in figure 9. It can be seen from the figure, the computational time is about 70 seconds when the data set contains 100 services, furthermore, the computational time is up to 238.77 seconds when there are 150 services in the experimental data and finally computational time is 598 seconds when there are 200 services in the experimental data. Because, in the resource management model, resource provisioning solution need to re-compute every few minutes, this result can provide clear guideline when apply this method to real system.

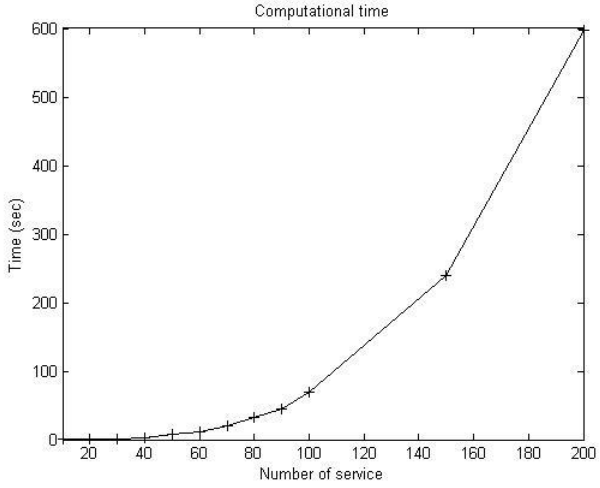

FIG. 9 COMPUTATIONAL OVERHEAD OF NFT-DRP

6. **Conclusion.** In this work, we reconsider the dynamic resource provision in the cloud data center. We do not only consider the energy efficiency which is the most concerned issue for cloud providers, but also consider the overhead of VM placement change which cannot be ignored in the cloud systems. We model the problem and propose a network-flow-theory-based approximate optimal algorithm. A series of simulations are performed to test the availability, efficiency and computational overhead of the proposed

method. The results show that it can greatly reduce the VM creation, destruction and migration operation compared to the existing method.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, et al. A view of cloud computing. Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] P. Barham, B. Dragovic, K. Fraser, et al. Xen and the art of virtualization. *Proc. of the 19th Symposium on Operating Systems Principles*, pp. 164-177, 2003.

[3] VMware. http://www.vmware.com/

[4] S. Jing, Shahzad. Ali, K. She, Y. Zhong. State-of-the-art research study for green cloud computing. Journal of Supercomputing, 2011.

[5] X. Zhu, D. Young, B. J. Watson, et al. 1000 islands: an integrated approach to resource management for virtualized data centers. Cluster Computing, vol. 12, no. 1, pp. 45-57, 2009.

[6] A. Beloglazov, R. Buyya. Energy efficient allocation of virtual machines in cloud data centers. *IEEE International Conference on Cluster, Cloud and Grid*, pp. 577-578, 2010.

[7] K. H. Kim, A. Beloglazov, R. Buyya. Power-aware Provisioning of Cloud Resources for Real-time Services. *Proc. of the 7th International Workshop on Middleware for Grids, Clouds and e-Science (MGC'09)*, Champaign, USA, 2009.

[8] G. Laszewski, L. Wang, A. J. Younge, et al. Power-aware scheduling of virtual machines in dvfs-enabled clusters. *IEEE International Conference on Cluster Computing and Workshop*, pp. 1-10, New Orleans, LA, 2009.

[9] D. Kusic, J. O. Kephart, J. E. Hanson, et al. Power and performance management of virtualized computing environments via lookahead control. Cluster Computing, vol. 12, no. 1, pp. 1-15, 2009.

[10] Y. C. Lee, A. Y. Zomaya. Energy efficient utilization of resources in cloud computing systems. Journal of Supercomputing, March, 2010.

[11] A. Verma1, P. Ahuja, A. Neogi. pMapper: power and migration cost aware application placement in virtualized systems. *Middleware*, pp. 243-264, 2008.

[12] E. Kalyvianaki, T. Charalambous, S. Hand. Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters. *Proc. of the 6th international conference on Autonomic computing*, pp. 117-126, 2009.

[13] X. Fan, W. D. Weber, L. A. Barroso. Power provisioning for a warehouse-sized computer. *Proc. of the 34th annual international symposium on Computer architecture*, pp. 13-23, 2007.

[14] C. Tang, M. Steinder, M. Spreitzer, et al. A scalable application placement controller for enterprise data centers. *Proc. of the 16th international conference on World Wide Web*, pp. 331-340, 2007.

[15] M. Yue. A simple proof of the inequality FFD (L)<11/9OPT(L)+1, for all l for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica (English Series), vol. 7, no. 4, pp: 321-331, 1991.

[16] P. Lindberg, J. Leingang, D. Lysaker, et al. Comparison and Analysis of Eight Scheduling Heuristics for the Optimization of Energy Consumption and Makespan in Large-Scale Distributed Systems. Journal of Supercomputing, vol. 59, no. 1, pp. 323-360, 2012.

[17] S. U. Khan, P. Bouvry, T. Engel. Energy-efficient High-Performance Parallel and Distributed Computing. Journal of Supercomputing, vol. 60, no. 2, pp. 163-164, 2012.