The Resource Configuration Method with Lower Energy Consumption Based on Prediction in Cloud Data Center

Liang Quan
School of Information Sciences and Engineering, Fujian University of Technology, Fuzhou 350108, China
Email: liangquanlq@gmail.com

Liang Jiumei
School of Chemical Engineering, Hunan Institute of Engineering, Xiangtan 412001, China
Email: 379486951@qq.com

Zou Fumin
School of Information Sciences and Engineering, Fujian University of Technology, Fuzhou 350108, China

Abstract—**The cloud computing data center have numerous hosts as well as application requests. In future, the short response time and user Qos are required, and the lower electricity power consumption to build the low-carbon green network is an irrevocable trend. The paper first puts forward a reconfiguration framework based on the request prediction of Double Exponential Smoothing, On the basis, work out in advance the allocation scheme which can improve the resource utilization ratio as well as lower energy consumption. The paper also present a concept of Utility Ratio Matrix (URM) to represent allocations of hosts and Virtual Machines (VMs) and a reconfiguration algorithm. The algorithm can separate the reconfiguration computing from the real allocation so that it can avoid a time delay, and can also reduce the energy consumption in data center. The corresponding analysis and experimental results show the feasibility of the reconfiguration algorithm in this paper.**

Index Terms—**Cloud Data Center; Request Prediction; Utility Ratio Matrix; Resource Reconfiguration**

I. INTRODUCTION

How to meet the requests of a huge number (up to millions of or even more) of application as well as to guarantee QoS is one of the main challenge for cloud data center, whereas the virtualization technology is a key lying in data center to coping with that challenge [1, 2]. Virtualization provides the necessary abstraction so that the underlying fabric (raw compute, storage, network resources) can be unified as a pool of resources and resource overlays (e.g. data storage services, Web hosting environments) can be built on top of it. However, cloud data center's virtualization is confronted with an actual problem when it deals with resource allocation. A dynamic resource configuration in accordance with the requirement variation and the resource status is an efficient way to cope with allocations on demands. Nevertheless, during the process of dynamic configuration, a variation of the demands and data center environment including the type and quantity of both requests and virtual machines, load of nodes and status of resource increases the complexity of the reconfiguration algorithm, which leads to a consequence that configuration is usually later than request variation. Considering the time consumed in the adjustment of VMs, nodes, resources and so on, it will aggravate the time delay and fail to provide a reliable QoS guarantee. Among many application scenarios such as web application, cluster system and distributed computing, the similar dynamic configuration policy of resources have been adopted. The researches mentioned above all invariably show the ubiquitous time delay [3, 4, 5].

In addition, data center is usually in possession of a huge resource storage, and a lot of servers in execution, which will consume a large quantity of electricity power and lead to a great electricity waste, especially while the deployed servers are in a peak utilization. Therefore, the power cost is a critical factor that limits the scale and efficiency of cloud data center. The adoption of an efficient and reliable deployment policy of VMs and resources so as to improve the utilization ratio of resource meanwhile lower the power consumption is another problem that cloud data center confronts, it will prove significant for building an energy-efficient green network environment [6].

The main contributions of the paper lie in the following: (1) A reconfiguration framework based on a request prediction method of Double Exponential Smoothing is provided so as to cope with the development of VMs and resources in cloud data center; (2) A data structure called Utility Ratio Matrix (URM) is presented to help to reduce the energy-comsuption; (3) A reconfiguration algorithm of VMs and resources is put forward in the paper. The innovation of the algorithm lies in the following: based on the prediction of application requests, the algorithm separates the computation of configuration results from the real configuration, namely, it makes out a specific configuration policy in advance before the real deployment executing, which can avoid a time delay of configuration results to the varied requirements.

II. Related Work

Refernece [7] gives a comparatively comprehensive illustration on the evolution, technological problems and existent challenges of data center. It is desirable to understand the aspects of their design that are worthy of carrying forward, as well as existing or upcoming shortcomings and challenges that would have to be addressed. The paper also define a layered model for such data centers and provide a detailed treatment of state of the art and emerging challenges in storage, networking, management and power/thermal aspects.

To traditional digital data center, a resource on-demand approach is proposed for Web applications, which can efficiently online reconfigure clusters in response to time-varying resource requirements. It can also dynamically decide the number of running nodes and virtual machines deployed on them [8]. For dynamic resource provisioning in large-scale enterprise data centers, researchers proposed a scalable algorithm that can produce within 30 seconds high-quality solutions for hard placement problems with thousands of machines and thousands of applications [9]. Another reference [10] also introduces and evaluates a middle ware clustering technology capable of allocating resources to web applications through dynamic application instance placement. It defines application instance placement as the problem of placing application instances on a given set of server machines to adjust the amount of resources which available to applications in response to varying resource demands of application clusters. Reference [11] proposes a resource on-demand approach for Web applications, which can efficiently online reconfigure clusters in response to time-varying resource requirements. It can also dynamically decide the number of running nodes and virtual machines deployed on them. It first predicts the future workloads of the applications with Brown’s quadratic exponential smoothing method to make reconfiguration catch up with demands. Bobroff and et al put forward a dynamic VMs migrating method, in which the unnecessary nodes will be shut down. In this method, Linear Time Series Prediciton is applied to predict the VM’s demands on resources, and the VMs are listed in a descending order according to their demands. Then, apply First-fit Knapsack Algorithm to deploy VMs on proper nodes [12]. Kusic and et al put forward a dynamic resource allocation framework based on Limited Control Predition. The framework through a two-layer control architecture can work out the number of VM duplicates that should be set on, the position of the node where the duplicate VMs lay as well as the resource amount allocated to VMs on a same node. Although the method can improve the resource utilization ratio by shutting down the unnecessary nodes, yet its computation complexity is extremely exponential [13]. Based on the analysis on topology characteristics and traffic patterns of data centers, reference [14] presents a novel approach called VM Planner for network power reduction in the virtualization based data centers. The basic idea of VM Planner is to optimize both virtual machine placement and traffic flow routing so as to turn off as many unneeded network elements as possible for power saving. Reference [15] proposes a coordinated cooling-aware job placement and cooling management algorithm which is Highest Thermostat Setting (HTS). HTS is aware of dynamic behavior of the Computer Room Air Conditioner (CRAC) units and places jobs to reduce cooling demands from the CRACs. HTS also dynamically updates the CRAC thermostat set point to reduce cooling energy consumption. Buyya et al. also made continuously deeper researches on energy-consumption of cloud data center and put forward some good ideas and methods [16, 17, 18, 19]. And Dougherty et al. put forward methods of green cloud computing infrastructure to facilitate to obtain a lower energy consumption [20, 21]. In addition, there are many researches on how to reduce the energy consumption in data center, we are not going to repeat them.

All researches mentioned above against the resource allocation of data center put forward different solutions. Through a thorough study about the above, we can draw a comparison between our work and the above, finding some differences. Firstly, while solving the problem, we have the different objectives. We focus on a real-time resource configuration, optimization and lower energy consumption while the above are only confined to one or two aspects. Secondly, we specially aims at cloud data center while the above are for other application scenarios. Though similar, they are quite different. Thirdly, all the above researches lack prediction step or taking different prediction methods which leads to a different working method and actual result. Fourthly, by experimental results, the configuration algorithm and its efficiency of this paper is comparatively superior to the above researches.

III. Predicition-based Configuration Framework of VMs and Resources

As mentioned above, there is a time delay of configuration computation to the variation of application requirements. Another related issue is the carbon dioxide (CO_2) emission that is detrimental for the physical environment due to its contribution in the greenhouse effect [22]. All these problems require the development of efficient energy-conscious provisioning policies at VM, host and resource level. Therefore, we put forward a configuration framework of VMs and resources based on Request Prediction (CFVmR-RP), as shown in figure 1.

After the occurrence or during the execution of applications, Request Predict Module (RPM) follows the certain predicting strategy to predict the variation trend of requests according to both the feature of application requests and variation of cloud data center environments.

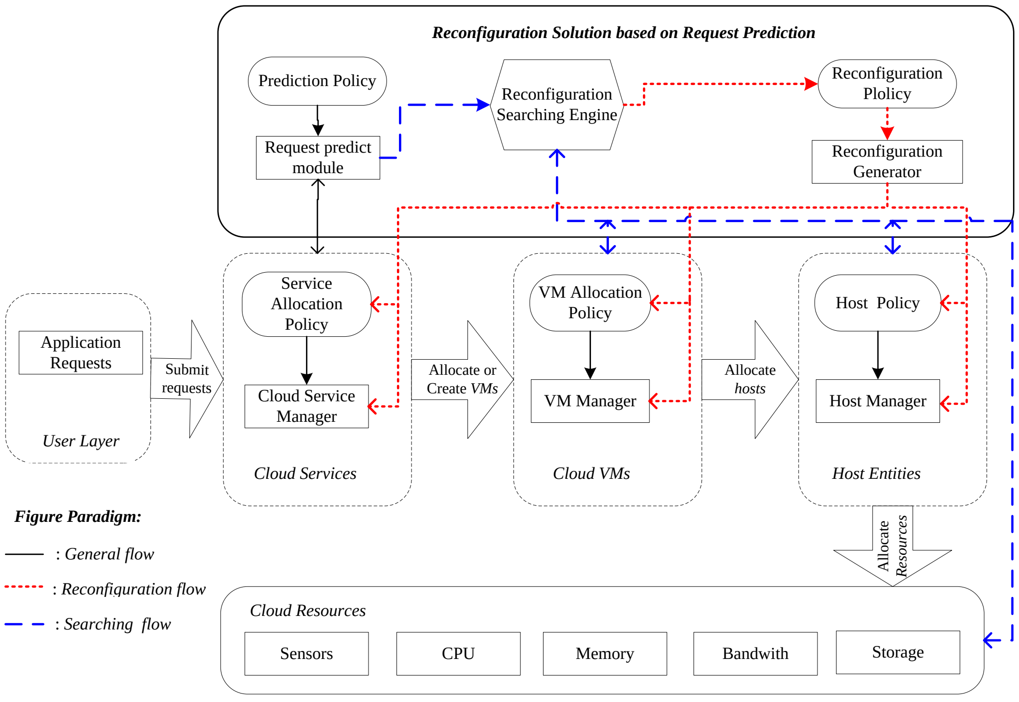

Figure 1. The configuration framework of VMs and resources

The prediction result will be sent to Reconfiguration Searching Engine (RSE) which selects the relatively optimal configuration by a through search among VMs, hosts and physical resources managed by cloud data center so as to adapt varied requirements of application. Later, according to the relatively optimal configuration from RSE, modify the reconfiguration strategy and carry out a real-time adjustment among VMs, hosts and physical resources. The modifies include VM started or released, hosts started or shut down and corresponding physical resources' deployment increased or decreased. As the relatively optimal configuration already been predicted before requests change, therefore the situation in which configuration results being later than request variations is avoidable. The configuration based on the request prediction not only optimizes the number of both VMs and hosts started but also improves the utilization ratio of resources. The next section will give a detailed description about the specific process of reconfigurations.

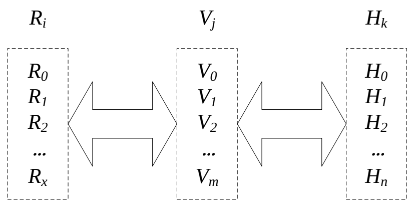
R_i
V_j
H_k
R_0 R_1 R_2 ... R_x
V_0 V_1 V_2 ... V_m
H_0 H_1 H_2 ... H_n

Figure 2. Multiple-multiple relationships of application requests, VMs and hosts

There actually are a multiple-mutiple relationship among applications, VMs and hosts. Namely, any application can be dispatched to several VMs and any VM can be deployed on several hosts which is in charge of resources of all sorts, as shown in figure 2. The reconfiguration based on the premix that application requirements are guaranteed aims to have a request prediction so as to reduce VMs and hosts as much as possible to improve the utilization ratio of resource as well as to lower power consumption.

R_i, V_j and H_k respectively stand for application requests, VMs and hosts. Suppose $\mathrm{Req}_{ij}=1$, it means a request R_i is dispatched to a VM V_j, while $\mathrm{Req}_{ij}=0$ dispatching not happen. Suppose $q_j=\sum_{i=0}^{x} Req_{ij}$, it presents the total sum of requests diaspatched to V_j, while $q_j=0$ means no any request is dispatched to V_j, thus at this moment, V_j can be released. Likewise, the total sum of requests received by cloud data center can be described as $q_{center}=\sum_{i=0,j=0}^{i=x,j=m} Req_{ij}$ (here, the same request when dispatched to different VMs is regarded as different requests, nevertheless it does not affect the deployment of VMs and hosts). Suppose $q_{center}=0$, it means any request is not received in the data center, at this moment all hosts and VMs on standby or idle can therefore be released or shut down.

Let $h_{jk}=1$, it means host h_k has been deployed a VM V_j, while $h_{jk}=0$ means no deployment has been done. Suppose $v_k=\sum_{j=0}^{j=m} h_{jk}$, it means the total sum of VMs deployed on host h_k, while $v_k=0$ means no VM has been deployed on host h_k, so at this moment, h_k is on standby or idle and therefore can be shut down or set to an energy-saving mode.

For a further understanding, VM utility ratio, resource utilization ratio and power consumption value are defined as the followings.

Definition 1. Utility Ratio δ_j

$R(V_j)$ represents the total amount of resources that can meet the demand of V_j; $R(Req_{ij})$ stands for the resource amount that is needed by the request R_i dispatched on V_j. Then utility ratio of VM V_j can be described as $\delta_j = \frac{\sum_{i=0}^{q_j} R(Req_{ij})}{R(V_j)}$, $\delta_j \le 1$, q_j is the total request amount received by V_j.

Definition 2. Resource Utilization Ratio α_k

$Resp_{max-k}$ represents the maximum amount of resources that host h_k can accept VM deployments, suppose $\{\beta_j \mid j = 0,1,2,\cdots,m\}$ be the probability of each VM ($V_j, j = 0,1,2,\cdots,m$) being deployed on host h_k, then let the resource utilization ratio of host h_k at T moment be $\alpha_k = \frac{\sum_{j=0}^{m}(\beta_j \bullet R(V_{jk}))}{Resp_{max-k}}$, $\alpha_k \le 1$, $R(V_{jk})$ is the resource amount required by VM V_j which is deployed on h_k. If $v_k = 0$, it means no VM is deployed on h_k and the utilization ratio is zero.

IV. Configuration Method of VMs and Resources based on Request Prediction

A. The Prediction Method of Double Exponential Smoothing

In cloud data center, new application requests occur constantly and resources are also released continually. Therefore, in order to configure resources in a highly efficient way and guarantee no time delay between configuration programs and varied requirements of applications, a prediction about the future application request is needed so as to know the demand of applications in advance. We adopt a method of Double Exponential Smoothing to predict the application request at the time $t+T$. Suppose the time serie $\{q_t\}$ has a linear change in trend from some time, then the linear trend predicting model is as follows

$$\hat{q}_{t+T} = a_t + b_t T \qquad (1)$$

a_t and b_t are the smoothing factors, in which,

$$\begin{cases} a_t = 2S_t^{(1)} - S_t^{(2)} \\ b_t = \dfrac{a}{1-a}(S_t^{(1)} - S_t^{(2)}) \end{cases} \qquad (2)$$

$S_t^{(1)}$, $S_t^{(2)}$ are respectively the first and the second smoothing values. Therefore $\hat{q}_{t+T}$ is the prediction value at the moment $t+T$. The detail prediction process can be seen in related literatures.

B. The Configuration Algorithm

Definition 3. Utility Ratio Matrix $I_{utility}$

Utility ratio matrix $I_{utility}$ is used to describe the load of each VM and host, which is shown as follows:

$$I_{utility} = \begin{array}{c c} & \xrightarrow{\qquad\qquad\qquad} k(Hosts) \\ \downarrow & \left(\begin{array}{ccc} \delta_j / \alpha_{k_x} & \cdots & \delta_j / \alpha_{k_y} \\ \vdots & \ddots & \vdots \\ \delta_{j_z} / \alpha_{k_x} & \cdots & \delta_{j_z} / \alpha_{k_y} \end{array} \right)_{m \times n} \\ j(VMs) & \end{array}$$

The elements within matrix are called utility ratio: $U_{load} = \delta_j / \alpha_k$, δ_j is the utility ratio of V_j, α_k is the resource utilization ratio of host h_k. In matrix $I_{utility}$, δ_j is descending sort by rows while α_k is descending sort by columns. The sorting result makes VMs of high utility ratio and hosts of high resource utilization ratio gather at the upper-left part of the matrix. As searching in matrix usually starts from low to high(e.g. 0~*m* or *n*), it is helpful to bring down the searching time. In addition, $I_{utility}$ obviously demands dynamic modification.

The configuration algorithm includes three parts which described as follows.

```
Algorithm 1. App_VM_Reconfiguration( )
Input: $I_{utility}$ of the result after shiftings finished
Output: New $List\_app\_VMs$ and $List\_VM\_Hosts$
{ Assignment_Shifting( );
Deployment_Shifting( );
While $j \le max\_j$ do {Place Apps into $V_j$ according to $List\_app\_VMs$; j++;}
While $k \le max\_k$ do {Place VMs into $H_k$ according to $List\_VM\_Hosts$; k++;}
Return ( $List\_app\_VMs$, $List\_VM\_Hosts$ );
}
```

```
Algorithm 2. Assignment_Shifting( )
{ Input: $I_{utility}$;
Output: $List\_app\_VMs$;
While $j \le max\_j$ do { $R(V_j) \times (1-\delta_j)$ ;write the residual into $List\_VMs\_residual$; j++;}
If (new App) then { Find the first VM who satisfy the App's requirement and assign it;
If (no VM satisfy the App) then {create a new VM and assign it;}
Re-sort $List\_VMs\_residual$; //Here can use method of Quick sort
write $List\_app\_VMs$;}
While $j \le max\_j$ do {Find VMs $j_x$ with smaller utility and stop its apps;
For $j = max\_j$ to $j_x + 1$ do {Assign the apps of $j_x$;}
If (the assignment accomplishes) then {stop $j_x$; write $List\_app\_VMs$;}}
Re-sort $I_{utility}$; Re-sort $List\_VMs\_residual$;
Return( $List\_app\_VMs$ );
}
```

```
Algorithm 3. Deployment_Shifting( )
{Input: $I_{utility}$, $List\_app\_VMs$;
Output: $List\_VM\_Hosts$;
While $k \le max\_k$ do
{ $Resp_{max-k} \times (1-\alpha_k)$ ; write the residual into $List\_Hosts\_residual$; j++;}
If (Hosts not satisfy VM's requirement) then {Start one or more hosts and deploy it;}
//VMs of the host with smaller utilization shifts into other hosts with higher utilization;
While $k \le max\_k$ do {Find host $k_y$ with smaller utilization and stop its VMs;
For $k = max\_k$ to $k_y + 1$ do {Place the VMs of $k_y$;}
If (the placement accomplishes) then {stop host $k_y$; write $List\_VM\_Hosts$;}}
Re-sort $I_{utility}$; Re-sort $List\_Hosts\_residual$;
Return ( $List\_VM\_Hosts$ );
}
```

The specific process of the reconfiguration algorithm can be divided into 3 stages. The first stage is Application Assignment Shifting (AAS). Based on request prediction and new application requirements, assign applications to VMs, and each assignment is respectively recorded in $List\_app\_VMs$ which records the application and its corresponding VMs. The shifting process includes: (1) Compute resource residual of each VM according to matrix $I_{utility}$ and keep the results in $List\_VMs\_residual$. A residual of VM V_j is $R(V_j) \times (1-\delta_j)$, since δ_j in $I_{utility}$ is listed in a descending order, the computed results are listed in an ascending order in $List\_VMs\_residual$. (2) To new application R_i, $R(Req_{ij})$ stands for the resource required by this application assigned to V_j. Search within $List\_VMs\_residual$ and assign the new application R_i to the VM with the first residual fitful for $R(Req_{ij})$, then record this in $List\_app\_VMs$. Update the sorting of $List\_VMs\_residual$: carry out a quick sort in matrix $I_{utility}$ so as to ensure the descending sort order of δ_j because the utility ratio has changed. The method can not only improve the utility ratio of VM but also lower the complexity of search. (3) To the assigned application, try to stop all or partial applications that has assigned on the VM with low utility ratio. For example, stop b applications on V_j after calculating out their resource amount (namely, $\sum_{i=0}^{b} R(Req_{ij})$) and then assign them to one or more VM with higher utility ratio. If all applications on V_j are stopped, then destroy V_j. The final assignment results are kept in $List\_app\_VMs$, update the sorting of both $List\_VMs\_residual$ and matrix $I_{utility}$.

The second stage is called VM Deployment Shifting (VDS). Shifting process includes: (1) According to matrix $I_{utility}$, calculate the resource residual of hosts. The resource residual of a host H_k is $Resp_{max-k} \times (1-\alpha_k)$, whose results are kept in $List\_Hosts\_residual$. Likewise, $List\_Hosts\_residual$ is listed in an ascending order. (2) Try to stop all or partial VMs that have deployed on the host who has a low utility ratio. For example, stop c VMs on host H_k after calculating their resource amount (namely, $\sum_{j=0}^{c}(R(V_j) \times \delta_j)$), deploy them to one or more hosts who has a higher utility ratio refer to $List\_Hosts\_residual$. If all VMs on H_k are stopped, then shut down H_k or let it be on standby. The final deployment results are kept in $List\_VM\_Hosts$ (this table records the location that which host the VM are deployed on). Update the sorting of $List\_Hosts\_residual$ and matrix $I_{utility}$.

The third stage is reconfiguration. Based on the results of two stages above, that is, $List\_app\_VMs$ and $List\_VM\_Hosts$, re-adjust the relatively optimal position of applications, VMs and hosts.

Special notes should be pointed out that the shifting locations of applications and VMs are only computed but real placement dose not happen in the former 2 stages. The real placement happens in the third stage in which the adjustment of relative positions of applications, VMs,and hosts are really done according to $List\_app\_VMs$ and $List\_VM\_Hosts$. It is a critical design of this reconfiguration algorithm, which can effectively lower the complexity of algorithm.

V. Algorithm Analysis and Experiment Results

A. The Algorithm Complexity Analysis

In Algorithm 2 (Assignment_Shifting), the time complexity of calculating VM residual is $o(m)$, the complexity of assigning new application and dealing with applications shifting is $o(m+2m\log^{m})$, the complexity of re-sorting $I_{utility}$ and $List\_VMs\_residual$ is $o(m \bullet n + m\log^{m})$. Therefore, the complexity of Algorithm 2 is $o(m \bullet (2+n+3\log^{m}))$. Similarly, the time complexity of Algorithm 3 (Deployment_Shifting) is proximately $o(n \bullet (1+m+3\log^{n}))$. Thus the whole time complexity of Algorithm 1 (Reconfiguration Algorithm App_VM_Reconfiguration) should be $o(2m+n+2m \bullet n+4\log^{m}+4\log^{n})$, which is actually superior to the time complexity $o(N^{2.5})$ in refernce [8] and nearly the same as $o(MN)$ in reference [7].

B. The Experiment Environment and Design

This section elaborates the test and analysis of the method put forward in the paper, the main test and analysis includes: prediction accuracy, the performance and comparison of reconfiguration algorithms and so on.

Our experimental environment is well qualified to simulate cloud computing data center. At present, FJUT computing center adopts HPP (Hyper Parallel Processing) architecture which has absorbed the advantages of both computers as Cluster and MPP. In computing center, the application server is Tomcat7.0, the operating system is Red Hat 2.6, and the management platform of virtual resources is VMWare Workstaiton 8.04. In addition, the computing nodes contain 92 blade servers (Dawning CB65-F) which can provide 736 processor cores of 2.6GHz and an internal memory of 1.5TB. The storage subsystem contains 4 data servers, an metadata server, a set of first-class real-time storage of 12T and a set of second-class duty storage. In addition, the experiment platform adds an extra 50 hosts(whose CPU is AMD Athlon(TM) 64 X2 3600+ 2.8GHz) as auxiliary computing nodes.

The computing center does more than just accepting daily service requests in order to facilitate carrying out related experiments, we specially developed a software of User Simulator which can generate service requests complying with Poisson Distribution so as to simulate the users' access to the data center.

C. Experiment Results

(1) Tests of the prediction accuracy

We choose a daily record of users' access to FJUT data center in one day as the experimental data. In order to obtain a larger amount of users' access, User Simulator is used to generate evenly partial data accumulated to this daily recorded data. The daily data are collected from 00:00AM~24:00PM. Every 20 minutes as a sample data is taken, so total 73 samples are taken in one day. The predicting results are shown in figure 3, the experiment result shows that users’ access amount increases significantly during both periods of 12:00~14:00PM and 21:00~24:00PM. Figure 3 indicates that the predicting value is fairly close to the real value so that it can accurately predict varied users' requests.

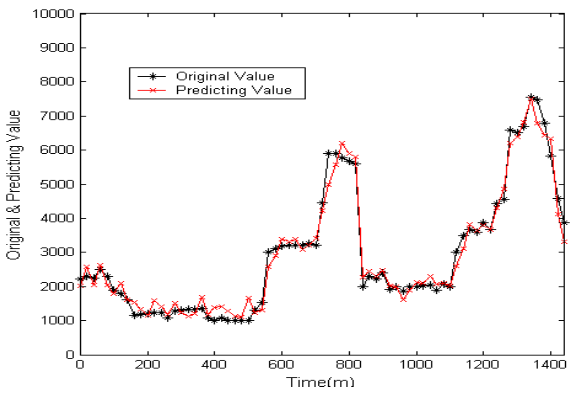

Figure 3. Original Value and Predicting Value of Requests

(2) Execution performance of the reconfiguration algorithm

The experiment first test the execution time of reconfiguration algorithm *App_VM_Reconfiguration* (AVMR) under the condition in which computing nodes and request amount increase, then compare *App_VM_Reconfiguration* with Application Placement Controller (APC) in reference [9], the experimental results are shown in figure 4. Within 142 computing nodes of FJUT data center, run the reconfiguration algorithm. As the number of hosts increases, the time consuming is all less than 30ms which can be ignored. The relatively smooth time curve demonstrates a stable performance of the algorithm which is not much affected by an increase in hosts. Compared to APC, AVMR is about equal at performance because of a twisted time curve in figure 4. And when there is an increase in request amount, the excuting time of the reconfiguration algorithm gradually increase but the time curve has a steady rising. No abrupt change occurs during the process, which indicates that the algorithm AVMR is quite stable. Compared to algorithm APC, when the request amount is small (≤ 1000), both the time curve and performance of AVRM are nearly identical with those of APC. But when the requests increase gradually, the algorithm AVRM presented in this paper starts to show some superiority over algorithm APC with its relatively better time performance. In addition, the more requests there are, the smoother the time curve is, because the request prediction enables a reconfiguration in advance to relieve the time lag of the configuration results.

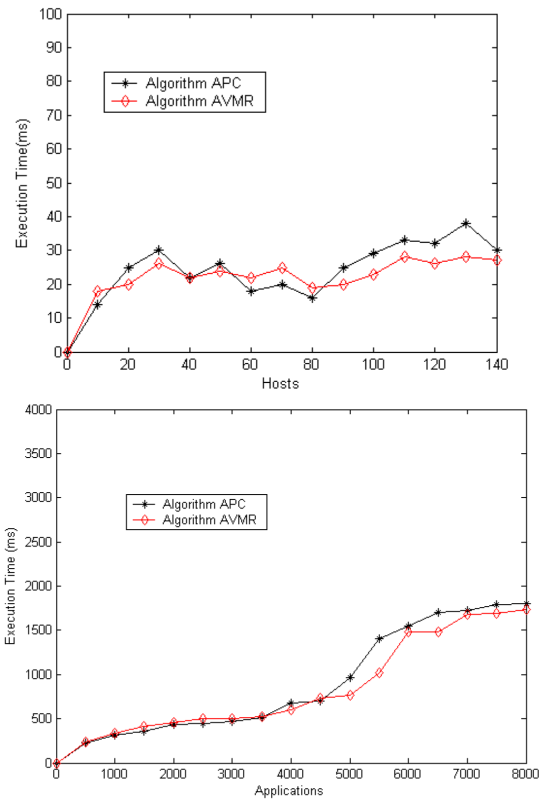

Figure 4. Execution time with re-configuration

To test the effectiveness of the reconfiguration algorithm AVMR, get the demand satisfaction ratio and the information of application placement changes when application requests and computing nodes vary. And then compare algorithm AVMR with algorithm APC, the comparison results are shown in figure 5. As application increases, the demand satisfaction ratio falls to some extent, yet still keeps above 0.94 which is an acceptable range of values. At 21:00~23:00PM when application requests in FJUT data center turns out to be quite large in amount, the algorithm AVMR detects application placement changes which limited in amount whose occurrence become smooth as the hosts increase in number. It shows that the algorithm can decide in a stable and efficient way whether a change is needed in a deployment of applications. Compared to algorithm APC, the placement change occurs evidently much less in algorithm AVMR, and the difference of these two algorithms tends to become lager and larger as the hosts increase in number.

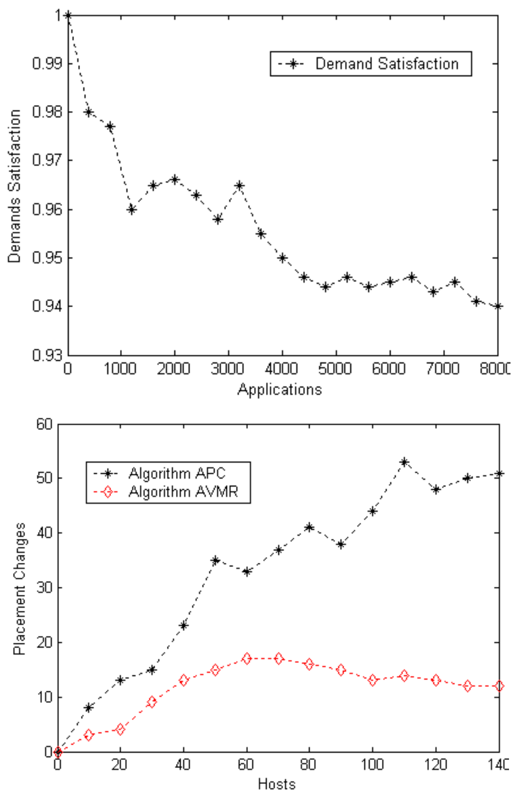

Figure 5. The demand satisfaction and placement changes with re-congfiguration

To observe algorithm AVMR manipulating physical hosts during its execution, 50% computing nodes (or 71 hosts) are set on while the other 71 nodes are shut down or on standby during the experiment. Likewise select the period 21:00~23:00PM, during which time the application requests in FJUT data center are relatively larger, add up the average time of hosts being shut down or started when the algorithm AVMR is executed, whose results are shown in figure 6. The experimental results show that during the period 21:00~22:00PM, among 71 computing nodes, 15 hosts are shut down meanwhile another 7 new computing nodes are started. Around 22:00PM, no hosts are shut down, instead 20 new computing nodes are started because it is the peak time when students access to web at night. What is more special, during this peak time, a large number of application requests are on-line video on demand which leads to a rapid increase of computing amount in data center. And when close to 23:00PM, there is a rapid decrease in operation of starting computer. In contrast, the operation of shutting down hosts increases, for the access to web decreases gradually.

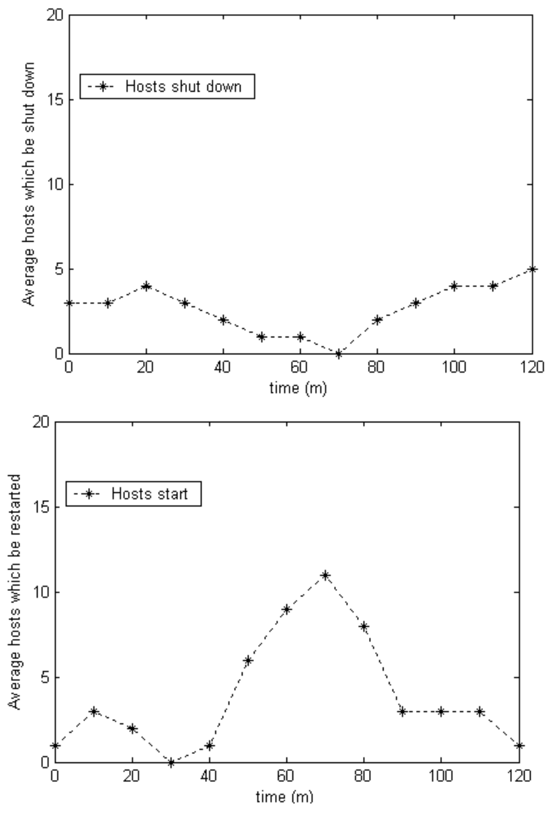

Figure 6. Average hosts which shut down and start

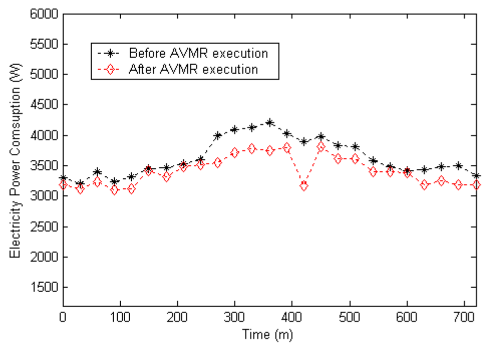

Figure 7. Comparison of electricity power comsuption

Finally we test the power consumption of data center. The comparison of electricity power consumption of FJUT data center before and after running of reconfiguration algorithm AVMR is made, the result is shown in figure 7. After a 12-hour track of electricity power consumption and statistics of every hour, we find some differences of power consumption before and after the running of AVMR. The electricity power consumption after the running of AVMR is less than that before the running of AVMR, which demonstrates that the reconfiguration algorithm is helpful to lower the electricity consumption of the data center.

VI. CONCLUSIONS

The future cloud data center will be a critical part of cloud computing application. The paper puts forward a reconfiguration framework based on request prediction according to the technological process of VMs and resource configuration in cloud computing data center. The innovativeness of this algorithm lies in the following two aspects: (1) Predict the application requests; (2) Separate the computing of the configuration program from real configurations implementing. For a better illustration of the innovative works, the paper puts forward some new definitions for the first time such as Utility Ratio Matrix which can well represent the utilization ratio of VMs and hosts in same a data structure.

As the results of the request prediction is the foundation of computing reconfiguration program, the accuracy of request prediction is of great importance. How to select a better prediction method calls for further researches. In addition, the optimization computing of the relatively optimized configuration is a critical and tough task, and what is the most proper optimization objective of the relatively optimized configuration and how to work out these objectives are worthy of an in-depth research in the future.

ACKNOWLEDGEMENTS

We would like to acknowledge and extend our heartfelt gratitude to our colleagues for their generous assistance, my special thanks also go to generous financial supports of the National Natural Science Foundation of China (under Grant No. 61101139), the Natural Science Foundation of Fujian Province (under Grant No.2012J01244, No.2012J06015, No.2012J01243 and No.2013J01214), and Hunan Provincial Project of Science and Technology (No. 2013FJ3090)

REFERENCES

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

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 25(6) pp. 599-616, Elsevier Science, June 2009.

[3] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, A. Tantawi. Dynamic placement for clustered Web applications. *In: Proc. of the 15th Int'l Conf. on World Wide Web.* 2006. 593−604. [doi: 10. 1145/1135777. 1135865].

[4] T. Mukherjee, A. Banerjee, G. Varsamopoulos, Sandeep K. S. Gupta, S. Rungta. Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. *Computer Networks,* 2009, 53(17) pp. 2888-2904.

[5] P. Padala, K. G. Shin, X. Y. Zhu, M. Uysal, Z. K. Wang, S. Singhal, A. Merchant, K. Salem. Adaptive control of virtualized resources in utility computing environments. *ACM SIGOPS Operating Systems Review,* 2007, 41(3) pp. 289−302. [doi: 10. 1145/1272996. 1273026]

[6] C. LIN, Y. TIAN, M. YAO. Green Network and Green Evaluation: *Mechanism, Modeling and Evaluation. Chinese Journal of Computers*, 2011, 34(4) pp. 593-612.

[7] K. Kant. Data center evolution: A tutorial on state of the art, issues, and challenges. *Computer Networks,* 2009, 53(17) pp. 2939-2965.

[8] H. B. MI, H. M. WANG, G. YIN, D. X. SHI, Y. F. ZHOU, L. YUAN. Resource On-Demand Reconfiguration Method for Virtualized Data Centers. *Journal of Software*, 2011, 22(9) pp. 2193−2205.)

[9] C. Q. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A Scalable Application Placement Controller for Enterprise Data Centers. *The International World Wide Web Conference Com -mittee (IW3C2).* WWW 2007, May 8. 12, 2007, Banff, Alberta, Canada. ACM 978 -1 -59593 -654 -7/07/0005.

[10] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic Placement for Clustered Web Applications. *The International World Wide Web Conference Committee (IWC2),* WWW2006, May 22–26, 2006, Edinburgh, UK. ACM 1595933239/06/0005.

[11] G. Pacifici, W. Segmuller, M. Spreitzer, M. Steinder, A. Tantawi, and A. Youssef, "Managing the response time for multi-tiered web applications," *IBM, Tech. Rep. RC 23651*, 2005.

[12] N. Bobroff, A. Kochut, K. Beaty. Dynamic placement of virtual machines for managing SLA violations. *In: Proc. of the 10th IFIP/IEEE Int'l Symp. on Integrated Network Management.* 2007. 119−128. [doi: 10. 1109/INM. 2007. 374776]

[13] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, G. F. Jiang. Power and performance management of virtualized computing environments via look ahead control. *Journal of Cluster Computing*, 2009, 12(1) pp. 1−15.

[14] W. W. Fang, X. M. Liang, S. X. Li, L. Chiaraviglio, N. X. Xiong. VMPlanner: Optimizing Virtual Machine Placement and Traffic Flow Routing to Reduce Network Power Costs in Cloud Data Centers. Computer Networks, Available online 23 September 2012.

[15] A. Banerjee, T. Mukherjee, G. Varsamopoulos, Sandeep K. S. Gupta. Integrating cooling awareness with thermal aware workload placement for HPC data centers. *Sustainable Computing: Informatics and Systems,* 2011, 1(2) pp. 134-150.

[16] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, *Univ. of Melbourne, Tech. Rep.* CLOUDS-TR-2010-3, 2010.

[17] R. Buyya, A. Beloglazov, and J. H. Abawajy, Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges, *Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2010), Las Vegas,* USA, July 12-15, 2010., vol. abs/1006. 0308, 2010.

[18] K. H. Kim, A. Beloglazov, R. Buyya. Power-aware provisioning of virtual machines for real-time Cloud services. *Concurrency and Computation: Practice and Experience,* 2011, 23(13) pp. 1491-1505.
[19] A. Beloglazov, J. Abawajy, R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer System*, 2012, 28(5) pp. 755-768.
[20] S. K. Garg, C. S. Yeo, R. Buyya. Green Cloud Framework for Improving Carbon Efficiency of Clouds. Euro-Par 2011 Parallel Processing Lecture Notes in Computer Science Volume 6852, 2011, pp 491-502.
[21] B. Dougherty, J. White, D. C. Schmidt. Model-driven auto-scaling of green cloud computing infrastructure. *Future Generation Computer System,* 2012, 28(2) pp. 371-378.
[22] "Report to Congress on Server and Data Center Energy Efficiency," *U. S. Environmental Protection Agency,* 2007.

Liang Quan graduated from Southwest Jiaotong University, China, in 1996. He received the M. S. degree from Central South University of Forestry Science and Technology in 2004 and the Ph.D. degree from Beijing University of Science and Technology, China, in 2008. He is currently an associate professor at the school of Information Science and Engineering, Fujian University of Technology. He has published about 60 refereed journal and conference papers, in which proximately 40 papers are indexed by SCI or Ei. His research interests include network computing and sensor networks.

He received research award from Science Foundation of the national and Fujian province He is a reviewer or PC member of international journal of Computational Information System, Journal of Grid Computing, Journal of Network and Computer Application, ICECE 2010 and ICCSIT 2011.