JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 30, 1537-1552 (2014)

An Integrated Dynamic Resource Scheduling Framework in On-Demand Clouds*

LEI XU, ZONGHUI WANG AND WENZHI CHEN
College of Computer Science and Technology
Zhejiang University
Hangzhou, 310027 P.R. China
E-mail: {leixu; zjuzhwang; chenwz}@zju.edu.cn

The biggest advantage of employing virtualization in cloud is the ability to provision resource flexibly which makes "pay-as-use" model possible. However, since the workload of virtual machine constantly changes, it is still a challenge that efficiently schedule resource by migrating virtual machines among lots of hosts, especially with multi-objective to meet, like power and QoS constraints. In this paper, we present a dynamic resource scheduling solution, named Smart-DRS, which can well determines ***when*** to schedule, ***which*** to schedule, and ***where*** to schedule. It is an integrated framework that aims to deal well with hotspot mitigation, load balancing and server consolidation which are classic problems to solve in on-demand clouds. The experimental results show that our framework strikes a balance between efficiency, overhead and instantaneity.

Keywords: cloud, dynamic resource schedule, hotspot mitigation, load balancing, server consolidation

1. INTRODUCTION

The ability of virtualization technology which is an important enabler for cloud computing brings immense benefits in terms of reliability, efficiency and scalability. Virtualization, coupled with VM (Virtual Machine) migration capability, enables the cloud data centers to flexibly provision resource which makes the "computing-as-a-service" vision of on-demand clouds possible. This "pay-as-use" model enables user pay only for the resources they really used in which compute resources are made available as a utility service – an illusion of availability of as much resources (*e.g.*, CPU, memory, and network bandwidth) as demanded by the user. It is the most important feature of IaaS (Infrastructure as a Service) and also the biggest push of cloud computing.

However, implementing such an on-demand cloud data center requires an excellent DRS (Dynamic Resource Scheduler), a power and QoS aware scheduler. As we know, the energy consumption of datacenter is an increasingly sharp problem. So we need to power off several PMs (Physical Machines) by consolidating all VMs together to certain selected PMs when the workload is low. Meanwhile, to keep high QoS performance of cloud service, we need to well manage resource to automatically balance load and mitigate high load machine in time.

For achieving aforementioned goals, in this paper, we present a three-step dynamic resource scheduling strategy, named Smart-DRS, which is an integrated scheduling solu-

Received December 14, 2012; revised November 8, 2013; accepted November 16, 2013.
Communicated by Ruay-Shiung Chang.

* This work has been presented in 2012 International Workshop on Power and QoS Aware Computing (P-QoSCom'12) in conjunction with IEEE Cluster'12, Sept. 2012, Beijing, sponsored by IEEE, IEEE Computer Society, IEEE TCSC, Intel, VMware, NVidia, *etc.*

tion that proved to have a good performance by experimenting in the prototype system we developed. We start the scheduling for three steps in general: (1) Firstly, we employ a prediction technique based on Single Exponential Smoothing algorithm to determine *when* to schedule; (2) Then, we use holistic approach (define a score model) to determine *which* VM should be scheduled; (3) At last, in the most important process of scheduling, we use a novel methodology based on vector projection arithmetic to decide *where* to allocate the potential VMs.

The rest of this paper is organized as follows: Section 2 describes related work. Section 3 details the algorithms using in Smart-DRS. Finally, in Section 4, we implement a prototype system to evaluate the performance of Smart-DRS. A summary and plan of our future work are described in Section 5.

2. RELATED WORKS

Dynamic resource scheduling is a primary problem in virtual environment management. It needs dynamically balances computing capacity across a collection of hardware resources aggregated into logical resource pools. When a VM experiences an increased load, it should automatically allocated additional resources by redistributing VM among the PMs in the resource pool. This problem has attracted a lot of attention from the research community in the last few years.

In [1-3], the authors discusses respectively about scheduling scheme making and executing. They treat VM deployment as a *Bin Packing* problem and use the *Bin Packing* dynamic programming algorithm to solve. This method traverses different number of backpacks to find the least number of backpacks that can accommodate all items. However, in fact, VM scheduling problem is more complicated than the *Bin Packing* problem, because VM has more than two-dimension factors to care about when place it to the target PM. That means deploying this method on cloud datacenter has several limitations.

Minimize the traffic when schedule the VMs has also been studied. [4] makes use of the network footprints to generate a good mapping for the data center. The goal is to minimize the communication cost between all the VMs. [5] also aims to minimize the communication traffic in a data center. They consider the topology of a data center and set communication cost as the number of switches between two VMs.

Reducing power consumption is also often considered. The goal is to reduce the number of active PMs. There are already some approximate algorithms proposed to solve this. The most widely used is first-fit [6-8], which can generate a near-optimal placement. But we think that first-fit has many limitations. In a production data center, VMs are entering and leaving continuously and redeployment are required periodically. First-fit ignores the migration cost and may generate unnecessary overhead. We need to take the migration cost into account.

Besides above all, Wood *et al.* [9] develop an integrated VM scheduling system named Sandpiper which is a XEN based automated provisioning system for monitoring and detecting hotspots. Thus, it detects when a VM is under-provisioned and either allots more resources locally or migrates the VM to a new PM which is capable of supporting the VM. However, Sandpiper is easy to choose a wrong target PM, because Sandpiper takes VM migration decision based on a metric, which it refers to as volume. That means it converts the three dimensional resource information of PMs into a single dimension metric (which is volume) and then uses this single dimension metric for worst fit in a three dimensional scenario. In this process, the information about the shape of the resource utilization is lost.

Compared to these works mentioned above, our work has the following main contributions:

- We present an integrated scheduling solution which contains the major processing procedures of resource management in cloud, such as when to schedule, which to schedule and where to schedule.
- We apply SES algorithm to predict workload and score model to choose the migrated VM. In addition, more importantly, we employ a novel algorithm based on vector projection arithmetic to select the potential target PM which could retain the shape of the multi-dimensional object representing VM or PM.
- We develop a prototype system running in a small scale cluster with 32PMs and 3200VMs to evaluate the performance of Smart-DRS.

3. DETAILS OF SMART-DRS

In this section, we discuss in detail about our three-step dynamic resource scheduling framework, Smart-DRS. As we know, there are three classic problems [8] to solve in the field of cloud resource management which towards simultaneously minimizing power consumption and maximizing QoS performance:

Hotspot mitigation: The goal of hotspot mitigation is to avoid a condition in which VM runs in a hot PM which has inadequate resource to meet its service level agreements (SLA) requirements. VM migration can mitigate this hotspot, so as to improve the QoS performance.

Load balancing: The goal of load balancing is to avoid a situation where there is a large discrepancy in resource utilization levels of the PMs. VMs can be migrated from the high workload host to the low workload one, so as to achieve this balance and improve the QoS performance.

Server consolidation: The goal of consolidation is to avoid server sprawl – many PMs host low-resource-usage VMs. VMs on lightly loaded host can be packed onto fewer machines to meet resource requirements, so as to reduce power consumption.

However, for each of the three goals mentioned above, we need to address three important questions: when to schedule, which to schedule and where to schedule. In the following, we will discuss our methods respectively.

3.1 *When* to Schedule?

As we know, most of traditional DRS strategies are based on the threshold which means the physical machine surpasses the upper boundary of threshold, then to trigger a migration. Therefore, a VM migration could be triggered by instantaneous peak load, which leads to the unnecessary waste of migration expenditure. However, Smart-DRS employs prediction technique based on SES (Single Exponential Smoothing) algorithm, which observes *n* future load values, if at least *k* value is above the threshold, we think the next load value is above the threshold.

SES algorithm is a widely used forecasting method and an efficient technique that can be applied to time series data, either to produce smooth data for presentation. SES algorithm is very suitable for our model, because in our system, the time series data are a sequence of historical monitoring data of resource utilization. And the observed phenomenon may be an essentially random process, or it may be an orderly but noisy process. Whereas in the SES the past observations are weighted equally, exponential smoothing assigns exponentially decreasing weights over time.

SES removes random perturbations of time series data, then the form of SES is given by the formula (1), while the sequence of observations begins at time $t = 0$.

$$\overrightarrow{y_{t+1}} = \overrightarrow{y_t} + \alpha(y_t - \overrightarrow{y_t}) \tag{1}$$

Where $\overrightarrow{y_{t+1}}$ is the prediction value at time $t+1$, $\overrightarrow{y_t}$ is the prediction value at time t, while y_t is the real value at time t, α is the smoothing factor, and $0 < \alpha < 1$. Formula (1) can change further to formula (2).

$$\overrightarrow{y_{t+1}} = \alpha y_t + (1-\alpha)\overrightarrow{y_t} \tag{2}$$

Through the observation, the above equation is kind of a recursion equation, which can be expanded to formula (3).

$$\overrightarrow{y_{t+1}} = \sum_{i=0}^{t-1} \alpha(1-\alpha)^i y_{t-i} + (1-\alpha)^t \overrightarrow{y_1} \tag{3}$$

As we can see from formula (3), exponential smoothing forecast value is a weighted sum of all the previous real observational values. That means SES makes use of all the historical data, so it has more stability and regularity.

In addition, we can't avoid that there would be some error information in the collected historical monitoring data which will affect the accuracy of the result. Therefore, we need process the data in advance to delete the information that is not appropriate.

$$\sigma = \sqrt{\tfrac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}, \bar{y} = \tfrac{1}{n}\sum_{i=1}^{n} y_i \tag{4}$$

y_i would be arranged in order statistics. If y_i satisfies $|y_i| > k\sigma$, then y_i is a bad value, which should be eliminated, and k is Grubbs criterion coefficient. Besides defining the error information model, we also need to determine the value of α and $\overrightarrow{y_1}$ to calculate the predictive value.

The Value of $\boldsymbol{\alpha}$: The value of α determine the degree of smoothing and how responsive the model is to fluctuation in the time series data. The value of α is arbitrary and is determined both by the nature of the data and the feeling by the forecaster as to what constitutes a good response rate. A smoothing constant close to zero leads to a stable model while a constant close to one is highly reactive. To our knowledge, it is good to set a small value of α in order to increase the weight of historical data when the time series data doesn't have fluctuations. On the contrary, it's good to set a big value of α to increase the weight of recent prediction value when the time series data has obvious fluctuations.

The Value of $\vec{y_1}$: In fact, the smaller value of α, the more sensitive our prediction value will be on the selection of this initial smoother value $\vec{y_1}$. In our method, we define $\vec{y_1}$ to be initialized to y_1 when the number of series data is more than an experiential value 15 [10]. While the number is less than 15, we define $\vec{y_1}$ to be the average value of the series data. As shown in formula (5).

$$\vec{y_1} = \begin{cases} \sum_{i=1}^{k} y_i / k, & k < 15 \\ y_1, & k \geq 15 \end{cases} \tag{5}$$

3.2 *Which* to Schedule?

Selecting one or more VMs for migration is a crucial decision of the resource management. The migration process not only makes the VM unavailable for a certain amount of time but also consumes resources like CPU, memory and network bandwidth on source and destination PMs. Performance of other VMs that are hosted on source and destination PMs are also affected due to increased resource requirements during migration. Many existing VM selection approaches are straightforward that selecting the candidate VM for migration because of resource constrained. The VM whose resource requirements can't be locally fulfilled is selected. However, Smart-DRS employs a more holistic approach that CPU utilization, memory usage and net bandwidth usage are all considered before selecting the candidate VM.

Reducing the memory data quantity that needs to be migrated is very important. Because that live migration mechanism is a way that iteratively copy VM memory pages to the target host as well as tracking those pages that have been modified in the copy process. It should intercept all memory access of the migrating VM which will seriously affect the application performance running in this VM. By reducing the amount of data copies in the network can minimize the total migration time and also mitigate the impact on application performance. Meanwhile, the VM whose CPU utilization rate or net bandwidth consumption is high means it has a great help to mitigate hotspot after migration. So Smart-DRS defines a model to balance these factors:

$$Score(cpu, mem, net) = F(workload(cpu, net), cost(mem)). \tag{6}$$

Where $Score(cpu, mem, net)$ takes the three most important factors into consideration and it's easy to extend to more dimensions. The score of VM is a function about workload and cost. Meanwhile, the workload is a function about CPU and network, and the cost is a function about memory.

$$\begin{cases} workload_i = \sqrt{cpu_i * net_i} \\ cost = \dfrac{1}{mem_i} \end{cases} \tag{7}$$

Where cpu_i represents the CPU utilization rate of VM_i, net_i is its usage of network bandwidth, mem_i is its usage of memory. It is clear that we want to choose the VM with both the highest workload value and the cost value. For solving this problem, we define:

$$Score(cpu_i, mem_i, net_i) = \frac{workload_i * cost_i}{workload_i + cost_i}. \tag{8}$$

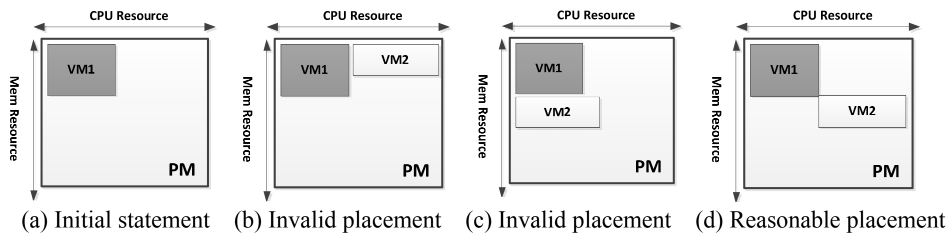
Score is the harmonic mean of *workload* and *cost*, and it would be arranged in descending order, Smart-DRS chooses the one with highest score value as the migrated VM. From formula (8) we can see that this VM has both the highest workload and cost' which means it has the most CPU utilization rate and network bandwidth but the least memory usage. In other words, the VM of the highest score value utilize the most physical resource but easy to migrate.

3.3 *Where* to Schedule?

After previous steps, we should choose out the target PM that has enough resources so that it can support the incoming migrating VM. To the best of our knowledge, many existing methods (including heuristic methods) treat this problem to be similar to multi-dimensional *Bin Packing* problem. However, while this method may seem fine on the surface, certain drawbacks and anomalies can be uncovered when we analyze deeper.

3.3.1 Multi-dimensional *bin packing* algorithm

Actually the problem of placing resource-aware VMs among PMs looks similar to the multi-dimensional bin packing problem. Existing literatures [7, 9-11] all tried to solve VM placement problem with bin packing algorithm and its variant. However, they are not exactly the same. Existing methods consider VM as a three dimensional object and PM as a three dimensional box. They try to pack as many objects in the boxes as possible, as well as the number of boxes required are minimized. However, while packing objects into a given box, two objects can be placed side by side or one on top of the other. But if we consider VMs as the objects, we can't place VMs like that. This is because once a resource is utilized by a VM, it can't be reused by any other VM. Fig. 1 shows the main difference.



(a) Initial statement (b) Invalid placement (c) Invalid placement (d) Reasonable placement

Fig. 1. The difference between VM placement and bin packing problem.

For simplicity, we show this scenario in 2D model. As we can see form Fig. 1, when VM2 should be placed in this PM, the only allowed position is the one shown in Fig. 1 (d). While in bin packing algorithm, the positions as shown in Figs. 1 (b) and (c) are also allowed. Due to this shortage of bin packing, we employ a novel algorithm to solve the VM placement problem based on Vector Projection.

3.3.2 Vector projection algorithm

Vector Projection (VP) model, presented in paper [12], is a novel methodology that can make the process of choosing PMs easier. Based on it, we proposed vector projection algorithm. In this algorithm, firstly, we choose three major resources available with the PM, namely CPU, MEM and IO. These resources form the three dimensions of an abstract object. We normalize the resources along each axis. Thus, the total available resource can be represented as a unit cube which is called Normalized Resource Cube (NRC). Secondly, we should express the resource related information of potential VMs and potential target PMs as a vector within the NRC, as shown in Fig. 2. The total capacity of PM is expressed as a vector from the origin of the cube (0, 0, 0) to point (1, 1, 1). This vector is identified as Total Capacity Vector (TCV). Resource Utilization Vector (RUV) represents the current utilization of resources of a PM The vector difference between TCV and RUV represents the Remaining Capacity Vector (RCV), which essentially captures how much capacity is left in the PM. The resource requirement of a VM is represented by Resource Requirement Vector (RRV) which is the vector addition of normalized resource requirement vectors of each resource type. In order to measure the degree of imbalance of resource utilization of a PM, we define a Resource Imbalance Vector (RIV) of PM. It is the vector difference between RUV's projection on TCV and RUV. RIV of a VM is defined in a similar way: it is the vector difference between RRV's projection on TCV and the RRV.

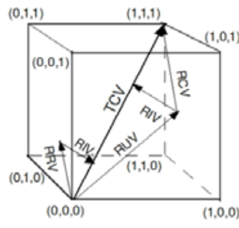


Fig. 2. Depiction of vectors in NRC.

Table 1. Vectors' name of Aronym.

Acronym	Name
TCV	Total Capacity of PM Vector
RUV	Resource Utilization of PM Vector
RCV	Remaining Capacity of PM Vector
RRV	Resource Requirement of a VM Vector
RIV	Resource Imbalance of PM/VM Vector

Then, we project NRC on a plane perpendicular to the principal diagonal of the cube as shown in Fig. 3. It is easy to see that this would result in a regular hexagon on the said projection plane. After that, we take the projection of the resource vectors onto the projection plane to make the resource information of 3D space reduce to 2D plane. We refer to this as Planar Resource Hexagon (PRH). Corners (0, 0, 0) and (1, 1, 1) of the NRC map to the center of the PRH. Other six corners maps to the six vertices of the hexagon as marked in the figure. Then we take the projection of tip of the resource vectors TCV, RUV (of PM) and RRV (of VM) onto the projection plane. The projection points of these three vectors would be inside the regular hexagon. For simplicity, the projection point of RUV of PM (RRV of VM) will be referred to as position of PM (VM) in the PRH. Note that the projection of tip of TCV is the center of the PRH.

By carefully analysis, we can know the regular hexagon on the projection plane is divided into six regular triangles. Shown in Fig. 4, we named these triangles like that: Δ CI defines the region where the projection of tip of RUV whose *CPU* > *MEM* > *IO*. Likewise, ΔMC represents the region where the projection of tip of RUV whose *MEM* > *CPU* > *IO*. Other triangles can be identified by similar inequalities. Further analysis shows that the VMs and PMs whose resource vectors projected in the same triangle or axis have an affinity. Now, we group all of the potential VMs and potential target PMs based on the rules that the RRV of a VM and RCV of a PM are in the same triangle or the same axis and in the opposite direction. That means they have the characteristics of complementary resources. Our ultimate goal is to find a best complementary target PM for a potential VM in a group, whose RIV is of same magnitude as the VM's RIV and is in the opposite direction. The groups are shown in Table 2. After comparison, we found

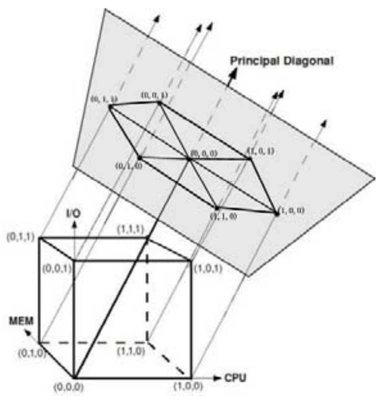


Fig. 3. Resource vector projection.

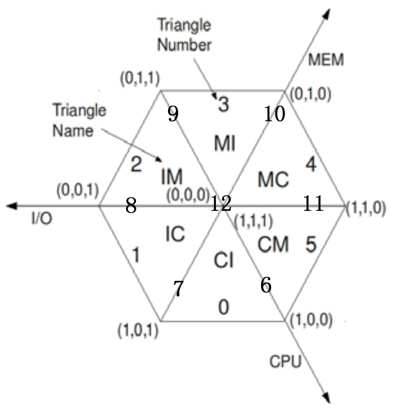


Fig. 4. The planar resource hexagon.

Table 2. Vector projection location groupings.

Group	Name	Group Feature
0	CI	Imbalanced in CPU
1	IC	Imbalanced in IO
2	IM	Imbalanced in IO
3	MI	Imbalanced in MEM
4	MC	Imbalanced in MEM
5	CM	Imbalanced in CPU
6	CPU axis	Especially imba in CPU
7	IC/CI	Imbalanced in CPU & IO
8	IO axis	Especially imba in IO
9	MI/IM	Imbalanced in MEM & IO
10	MEM axis	Especially imba in MEM
11	MC/CM	Imbalanced in mem&cpu
12	Origin P	Resource balancing

that the triangle is imbalanced in the resource which the closest resource axis represents. If a triangle has no projection of tip of vectors, we will make it with adjacent triangles into a new group, because adjacent triangles are with similar characteristics of re-sources. Loop this process until every group has resource vectors.

3.3.3 VP algorithm elaboration

This segment details VP algorithm which is a VM placement algorithm. In section 3.2, we have introduced how to choose a VM to migrate. And VM placement problem is how to choose a target PM for this VM. The target PM for this VM can be chosen based on one of the three goals: hotspot mitigation, load balancing, and server consolidation.

Algorithm 1: getMigrated_VM (goal)

```
1.  switch (goal){
2.  case hotspot mitigation:
3.    for all VMs running in this hotspot PM
4.      find the highest value of score(x)
5.    return VMx
6.  case load balancing:
7.    order_AllPMlist();
        /* according to the resource utilization
        of every PM in descending order. */
8.    P=the PM in the top of AllPMlist
9.    find the highest value of score(x) in P
10.   return VMx
11. case server consolidation:
12.   order_AllPMlist();
        /* according to the resource utilization
        of every PM in ascending order. */
13.   P=the PM in the top of AllPMlist
14.   find the highest value of score(x) in P
15.   return VMx
16. }
```

Algorithm 2: getTarget_PM (VM)

```
1.  named the triangle T which contains
    VM.RRV
2.  for all PMs do
3.    if PM.RCV locates in T
4.      add this PM into PotentialPMlist
5.    end if
6.  end for
7.  if no PM.RCV locates T    then
8.    T=T+left triangle of T+right triangle of T
9.  end if
10. Remove PMs from PotentialPMlist which
    can't support the VM
11. then
12. choose the most complementary PM in
    PotentialPMlist whose RIV is of the most
    closest magnitude (is the most slightly less)
    as the VM's RIV
13. return this PM
```

Algorithm 1 gets the VM which should be migrated away firstly based on three different goals. Then algorithm 1 would pass this VM as a parameter to algorithm 2. Algorithm 2 firstly generates the *potentialPMlist* by adding those PMs whose RCV locates in the same triangle with the VM's RRV. At last, it chooses the most appropriate PM in the *potentialPMlist* whose RIV is of same magnitude as the VM's RIV and is in the opposite direction.

4. EXPERIMENTAL RESULTS

For evaluating the performance of Smart-DRS, we have implemented a prototype system as discussed in Section 3. Besides, we build a small-scale cluster whose detail information is listed as follows:

Table 3. Experiments environment.

PM Numbers	VM Numbers	CPU Frequency	Memory	Network	VMM Version
32	3200	3.3GHz	4GB	100Mb/s	Xen 3.0.3

The major goal of our design is to evaluate the performance of Smart-DRS strategy while applied in data center environment. In order to avoid time-consuming and complicated implementation in code without knowing potential effects of the modification, we choose some mature open source projects in our system, such as Ganglia in monitor module and Xen Motion in execution module. The detailed system architecture is described in Fig. 5.

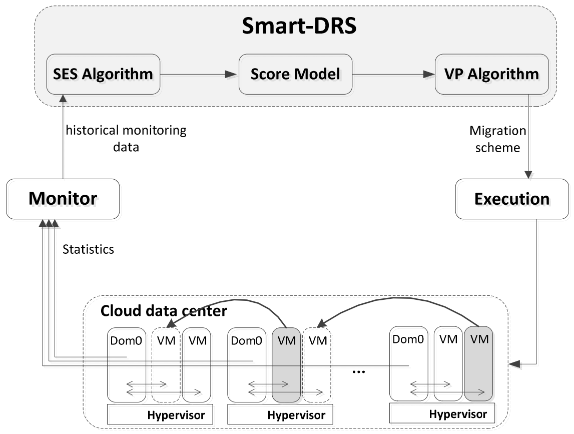


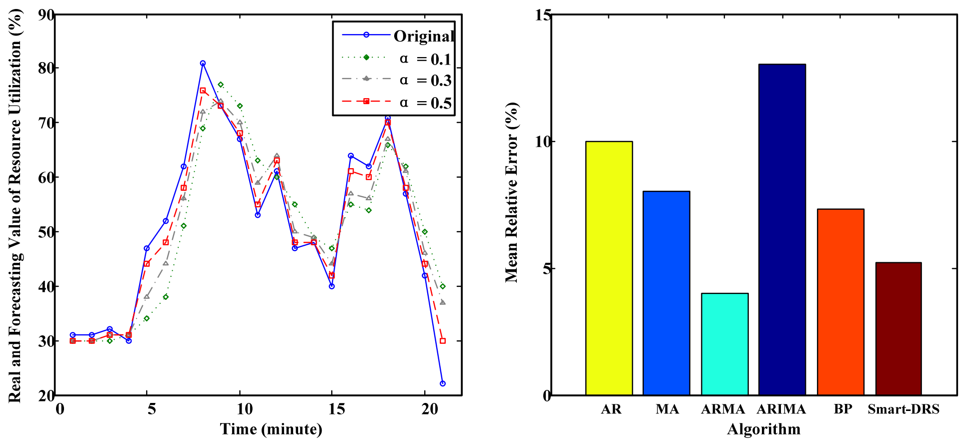
Fig. 5. The architecture of experiment prototype system.

4.1 Evaluation of *When* to Schedule

In order to evaluate the performance of predicting algorithm, we need some of convictive host load samples. Fortunately, Dinda and O'Halloran from Carnegie Mellon University offered us plenty of load samples by long-term tracing with many kinds of machines in a cluster system. We choose the day's collection of load time series on August 18, 2010 as our test samples.

In the first experiment, as shown in Fig. 6 (a), we continuously monitor 32 PMs for 21 minutes and record the real load value. After that, we respectively calculate the forecasting value for average when $\alpha = 0.1, 0.3, 0.5$, the Mean Relative Error (MRE) is correspondingly 13.6%, 9.7% and 7.3%. That means SES algorithm has an acceptable predicting accuracy while applied in a small-scale datacenter and $\alpha = 0.5$ is a better situation in our experiment environment.

In the second experiment, as shown in Fig. 6 (b), we compare Smart-DRS with other classic forecasting algorithms such as AR (Autoregressive), MA (Moving Average), ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average) and BP (Back Propagation) neural network. We can see that ARMA prediction algorithm has the least mean relative error, which means it has the most accuracy, be-



(a) Compared with itself of prediction efficiency. (b) Compared with other prediction methods.
Fig. 6. Evaluation of "when to schedule" performance.

cause it combines AR's advantage with MA's. Then our Smart-DRS follows, it has more accuracy that BP neural network which is a kind of continuous practicing algorithm. Even Smart-DRS is a relatively simple model, it has a relatively good performance, and the mean relative error of it is acceptable.

4.2 Evaluation of *Which* to Schedule

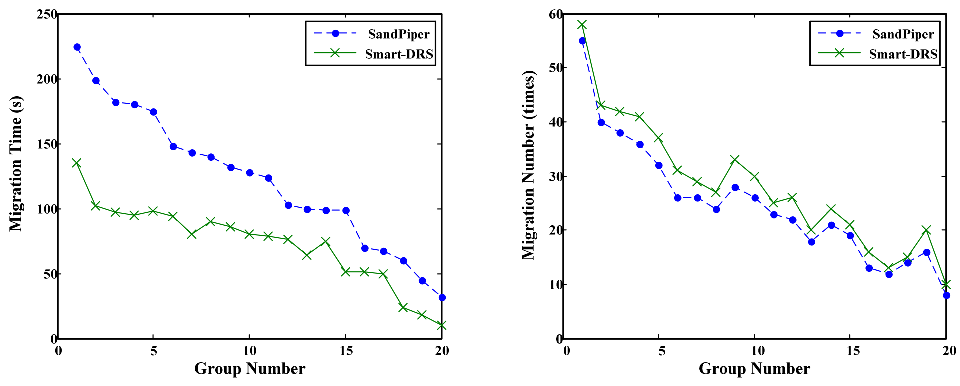
SandPiper is a Xen based automated provisioning system [6]. It takes VM migration decision based on a metric, which refers to as *sand_volume.*

$$sand\_volume = \frac{1}{1-cpu_i} * \frac{1}{1-mem_i} * \frac{1}{1-net_i} \tag{9}$$

Where *cpu*, *mem* and *net* are the corresponding normalized utilizations of the resources (in this paper, for simplicity, we just consider PM as a 3D object). According to our experience, in many experiments, authors would like choose the VM whose *sand_volume* is the highest as the migrated VM. Because *sand_volume* value represents the resource utilization of the VM and migrating the highest one helps a lot to mitigate the resource utilization rate. However, different VM choosing strategy will affect the migration time and migration numbers.

In this experiment, we carry out a series of 20 groups tests including different initial statements of our experiment cluster, such as hotspot, load unbalancing, low resource utilization and so on. We count the migration time and migration numbers from initial state to the final state between SandPiper model and our *score* model of Smart-DRS.

Fig. 7 (a) tells us that Smart-DRS can save much more migration time while just a little high in migration numbers compared to SandPiper shown in Fig. 7 (b). Because Smart-DRS would like to choose the VM which utilizes high cpu and net bandwidth but low memory resource. Smart-DRS can reduce much more migration time at the sacrifice of a little more migration numbers.



(a) Migration Time experiment. (b) Migration Number experiment.
Fig. 7. Evaluation of "which to schedule" performance.

4.3 Evaluation of *Where* to Schedule

In this experiment, we try to evaluate the performance of VM placement. First of all, we should claim that we have defined *Balance*$_i$ to represent the degree of load balancing of each machine and *SYS*$_{balance}$ to represent the degree of load balancing of the whole system (a higher value is better).

$$Balance_i = \sqrt{[\delta_{cpu_i}{}^2 + \delta_{mem_i}{}^2 + \delta_{io_i}{}^2]/2} \tag{10}$$

$$\begin{cases} \delta_{cpu_i} = cpu_i - \sum_{i=1}^{n} cpu_i / n \\ \delta_{mem_i} = mem_i - \sum_{i=1}^{n} mem_i / n \\ \delta_{io_i} = io_i - \sum_{i=1}^{n} io_i / n \end{cases} \tag{11}$$

$$SYS_{balance} = \sum_{i=1}^{n} Balance_i n \tag{12}$$

This experiment compares the performance of Smart-DRS with two classical Bin Packing algorithms used in dynamic resource scheduling, *Best Fit Decreasing* (BFD) and *First Fit Decreasing* (FFD). We measure the algorithm execution time and *SYS*$_{balance}$ of each algorithm. As shown in Fig. 8, the BFD and FFD consume much more execution

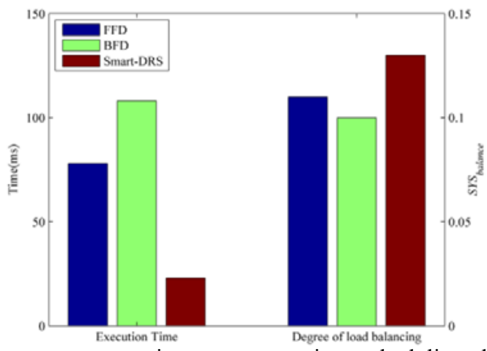
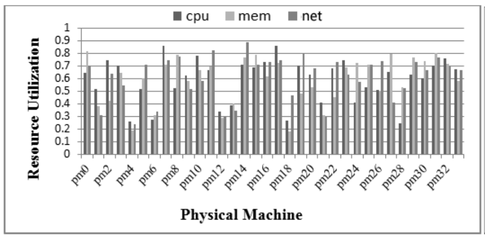


Fig. 8. Performance comparisons among various scheduling algorithms.

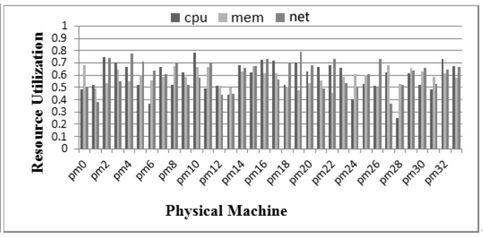
time and achieve no better performance in $SYS_{balance}$ than Smart-DRS. The reason is that they have to sort all of the potential VMs and target PMs firstly according to their resource utilization.

4.4 Overall Evaluation

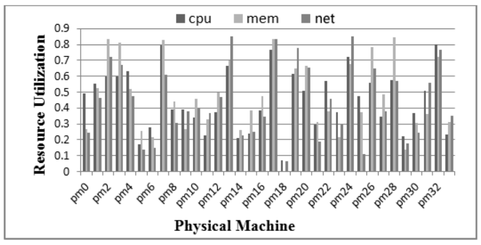
In the last experiment, we assess the performance of Smart-DRS in overall situation. By continuously monitoring the resource utilization of *cpu*, *mem*, *net* of 32 PMs, we could observe the difference between before scheduling and after scheduling. In order to carry out this experiment, we have set 0.8 as the upper threshold and 0.2 as the lower threshold.



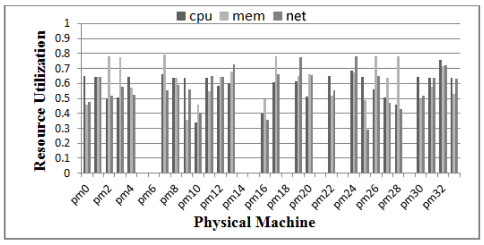
(a) Before scheduling (load balancing).



(b) After scheduling (load balancing).



(c) Before scheduling (server consolidation).



(d) After scheduling (server consolidation).

Fig. 9. The overall evaluation between before and after scheduling.

Fig. 9 shows that our scheduling algorithm really works. It is clear that the load of system is imbalanced and the resource utilization of several machines exceeds 0.8 before scheduling as shown in Fig. 9 (a). While in Fig. 9 (b), after scheduling, the load of system is more balanced and none of resource utilization exceeds the upper threshold. Similarly, in Fig. 9 (c) we can see that the resource utilization of several machines is under 0.2 while in Fig. 9 (d), these low utilization machines are taken offline and the others are load balancing.

5. CONCLUSION

In this paper, we studied the resource management method in on-demand cloud environment and we have presented an integrated dynamic resource scheduling framework named Smart-DRS. It employs SES algorithm to predict the resource utilization of PMs in order to avoid tiny and temporary load peak value triggering unnecessary migration. The experiment result shows that prediction value is pretty close with the real value. Then Smart-DRS creates score model to determine which VM to choose. And the result tells us that it can save much more migration time which is very important to improve QoS performance. At last, Smart-DRS employs VP algorithm to make the migration scheme. The reason why we choose VP algorithm is that it's a novel theory which can be used to make the process of choosing PMs easier and more appropriate. The experiment result tells us that it works really well and it's a kind of low cost and efficient method.

Nevertheless, our framework has some limitations that we plan to address in the future: (1) Firstly, we can't employ Smart-DRS on a large scale cloud data center, so we don't know its scalability; (2) Then, we will comment on the computation and space complexity of vector projection algorithm in future work; (3) At last, we don't talk about the user priority. In the real world, it is very likely that some high priority users or users who pay more for the resources need to occupy the whole physical machines. We can't move other VMs to these machines even if they are actually idle.

In addition, various other measurements and optimization strategies will need to be explored in the future.

REFERENCES

1. F. Hermenier, X. Lorca, J. M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *Proceedings of ACM SIGPLAN International Conference on Virtual Execution Environments*, 2009, pp. 41-50.
2. C. Hyser, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," *Hewlett Packard Laboratories*, 2007, pp. 189-195.
3. R. Nathuji and K. Schwan, "VirtualPower: Coordinated power management in virtualized enterprise systems," in *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles*, 2007, pp. 265-278.
4. J. D. Sonnek, J. B. S. G. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," in *Proceedings of International Conference on Parallel Processing*, 2010, pp. 228-237.
5. X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of International Conference on Computer Communications*, 2010, pp. 1154-1162.
6. T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, Vol. 53, 2009, pp. 2923-2938.
7. M. Cardosa, M. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*, 2009, pp. 327-334.
8. A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of hpc applications," in *Proceedings of the 22nd Annual International Conference on Super-Computing*, 2008, pp. 175-184.
9. T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-

box strategies for virtual machine migration," in *Proceedings of the 4th ACM/USENIX Symposium on Networked Systems Design and Implementation*, 2007, pp. 229-242.
10. M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *Communications Magazine*, Vol. 50, 2012, pp. 34-40.
11. F. Hermenier, S. Demassey, and X. Lorca, "Bin repacking scheduling in virtualized datacenters," in *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming*, 2011, pp. 27-41.
12. N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Proceedings of the 10th International Symposium on Integrated Network Management*, 2007, pp. 119-128.
13. M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proceedings of IEEE International Conference on Computer Communications*, 2011, pp. 71-75.
14. M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodogies and their migration using a novel vector based approach," in *Proceedings of IEEE 4th International Conference on Cloud Computing*, 2011, pp. 275-282.
15. J. Li, J. Peng, Z. Lei, and W. Zhang, "An energy-efficient scheduling approach based on private clouds," *Journal of Information Computational Science*, Vol. 8, 2011, pp. 716-724.
16. L. Xu, W. Chen, Z. Wang, and S. Yang, "Smart-DRS: A strategy of dynamic resource scheduling in cloud data center," in *Proceedings of IEEE International Conference on Cluster Computing Workshops*, 2012, pp. 120-127.
17. T. Imada, M. Sato, and H. Kimura, "Power and qos performance characteristics of virtualized servers," in *Proceedings of the 10th International Conference on Grid Computing*, 2009, pp. 232-240.

Lei Xu (徐磊) is now a Ph.D. student in the College of Computer Science and Technology, Zhejiang University. His current research interests include operating system, virtualization and cloud infrastructure.

Zonghui Wang (王总辉), born in March 1979, is a Lecturer in the College of Computer Science and Engineering at Zhejiang University in Hangzhou, China. He received his Ph.D. in the College of Computer Science and Technology at Zhejiang University in 2007. His research interests focuse on cloud computing, distributed system and computer architecture.

Wenzhi Chen (陈文智) born in 1969, received his MS and Ph.D. degrees from Zhejiang University, Hangzhou, China. He is now a Professor and Ph.D. supervisor of College of Computer Science and Technology of Zhejiang University. His areas of research include computer architecture, system software, embedded system and security.