A Memory Configuration Method for Virtual Machine Based on User Preference in Distributed Cloud

Shukun Liu[1*], Weijia Jia[2], Xianmin Pan[1]
[1] Department of Information Technology, Hunan Women's University, 410004 Changsha, China
[e-mail: liu_shukun@csu.edu.cn]
[2] Faculty of Science and Technology, University of Macau, 999078 Macau, China
[e-mail: weijiaj@gmail.com]
*Corresponding author: Shukun Liu

Received February 18, 2018; revised April 30, 2018; revised June 2, 2018; accepted June 19, 2018; published November 30, 2018

Abstract

It is well-known that virtualization technology can bring many benefits not only to users but also to service providers. From the view of system security and resource utility, higher resource sharing degree and higher system reliability can be obtained by the introduction of virtualization technology in distributed cloud. The small size time-sharing multiplexing technology which is based on virtual machine in distributed cloud platform can enhance the resource utilization effectively by server consolidation. In this paper, the concept of memory block and user satisfaction is redefined combined with user requirements. According to the unbalanced memory resource states and user preference requirements in multi-virtual machine environments, a model of proper memory resource allocation is proposed combined with memory block and user satisfaction, and at the same time a memory optimization allocation algorithm is proposed which is based on virtual memory block, makespan and user satisfaction under the premise of an orderly physical nodes states also. In the algorithm, a memory optimal problem can be transformed into a resource workload balance problem. All the virtual machine tasks are simulated in Cloudsim platform. And the experimental results show that the problem of virtual machine memory resource allocation can be solved flexibly and efficiently.

Keywords: Virtual memory block usability, user preference, configuration method, user task requirement, resource allocation

This work was supported in part by the Natural Science Foundation of Hunan Province (2018JJ2193, 2016JJ3051), National Natural Science Foundation of China (Grant No. 61502163,No.61532013)and was also supported by the research project of teaching reform in ordinary universities of Hunan province in 2017 (No. 579). The work was also supported by supported by DCT-MoST Joint-project No. (025/2015/AMJ); University of Macau Funds Nos: CPG2018-00032-FST & SRG2018-00111-FST; National China 973 Project No. 2015CB352401; Shanghai Scientific Innovation Act of STCSM No.15JC1402400 and 985 Project of Shanghai Jiao Tong University: WF220103001. We would like to express our thanks to the editors and reviewers for their valuable comments and suggestions to improve the presentation of our manuscript.

1. Introduction

Nowadays, virtual machine technology has played an important role in areas of cloud computing, big data and so on. The importance of virtual technology is that it can significantly improve the utilization of resources. So all the virtual machines can use hardware at the same time, but the quantities of the hardware do not change a lot. In this way, users and service providers can save much time and many costs in some extent.

Security isolation, high reliability and smaller grain size of computer system which can be provided by virtual machine technology are all very important to users [1, 2]. But now the main researches of memory allocation have been paid more attention to only single physical machine environment, few of them pay attention to multi-virtual machine environments which should be a hot research issue. Now many researches of memory allocation are mainly focus on the view of operation system, for example, distributed sharing memory system [3, 4], network memory [5, 6] and so on. Resource utility is a key factor to evaluate the excellence degree of cloud platform. If users do not pay more attention to it, more and more resource will be wasted. So user satisfaction maybe be decreased quickly too. Today the most important problem is how to construct high efficient global resource management frame and achieve the goal that all kinds of resource can be allocated properly on cross-physical machines platform.

We have found that whether the memory can be virtualized successfully or not is a key factor of hardware resource utilization, for example, the utilization of CPU and I/O devices. If the memory requirement is not be satisfied, all the machines cannot work normally [7]. So virtual memory has already been a bottleneck of success of high utilization for other resource. How to improve memory utilization and allocate memory resource properly is a hot issue today. The main contribution of this paper is as follow:

(1)In this paper, the concepts of user service preference and virtual memory blocks usability are proposed firstly.

(2)A memory resource allocation mechainsm is proposed according to virtual machine priority and virtual memory blocks usability in distributed cloud.

(3)Furthermore, an improved memory allocation optimization algorithm based on user preference requirement ,makespan and virtual memory blocks usability is proposed.

2. Background

This paper discusses the optimizations method for multi-virtual machine from the view of user and resource balance. The assignment methods and the adjusting aim are similar in the technology view. But our optimization method for memory allocation is in advantage of the achieving the resource balance during the memory allocation.

The aim of virtual memory optimization mechanism is to obtain the memory useful situation and allocate the memory resource properly. We can reclaim memory resource from the virtual machines which occupy overmuch memory space and reallocate them to the virtual machines which have no enough memory resource to use. Memory optmization is a special task of virtual machine. Memory running situation can be monitored in a proper way. In some extent, the change sates of overload for all the virtual machines can be estimated. So we can know whether the assignment states are properly. Now the main memory optimization technology can be classified into page reuse technology, dynamic adjustment mechanism of memory and multiple virtual machine memory workload balance technique. Within limits

assignment mechanism of memory can bring in the optimation perfamance for the whole system.

2.1 Research of Memory Workload

Weiming Zhao[8] implemented a memory balance tool named MEB with the function of monitoring the application situation of virtual machine. And it can predict the memory requirements and allocate the active memory periodically. MEB has two effective prediction units which can reclaim the memory space without degrading of the performance obviously. The result shows that the efficiency will become higher and the throughput capacity will be improved greatly. Katsunori Sato[9] proposed an idea of monitoring the running situation of virtual memory according to the real-time allocation situation of memory. They implemented a tool named VMMB(Virtual Machine Monitor Balancer)with a new mechanism of balance every memory state of virtual machine. And simultaneously, Waldspurger [10] proposed several mechanisms too. For example, it can estimate the memory using situation when the virtual machine did not participate with memory allocation with the method of statistical sampling. In every sampling period, only few pages are selected and monitored, so it can judge whether they are visited or not. If the pages are used, they estimated all the pages which are visited by virtual machines. But the disadvantage is that the predicated domain is limited by the scale of the allocation space.

Galloway in literature [11] introduced an online greedy algorithm, in which physical machines can be dynamically turned on and off but the life-cycle of a virtual machine is not considered. Jianhua Gu et al. [12] stated an algorithm named Genetic, which calculates the history data and current states to choose an allocation.

2.2 Typical Technology of Page Reuse

Typical technology of page reuse includes host swapping and page sharing. Cellula Disco system firstly proposed the idea that swapping the part of physical memory pages with part of disk space of host operating system. So the useful virtual memory of virtual machine can overtake the machine memory. An idea that swapping the part of physical memory pages with part of disk space of host operating system is proposed in the Cellula Disco system few years ago firstly. WaldSpurger [10, 13] and Sugerman [14] have achieved the swapping virtual technology in VMware ESX Server and VMware workstation separately. Memory sharing technology among virtual machines based on page content comparison is proposed in literature [15]. And its main method is using hashing table. Identify the same page content which will be used by different virtual machines. In this method, the memory space is saved greatly. And Gupta [16] combined page sharing with page compressing and page patching which can be used to improve the memory utilization.

2.3 Typical Technology of Adjusting Memory Dynamically

Nowadays, balloon technology is the mainstream of memory allocation. In the system of VMware, using balloon mechanism the memory which is not used in some virtual machines can be reclaimed. And some virtual machine can apply the memory that is occupied by other virtual machines again. As we know that the balloon mechanism is supported in XEN and KVM. When the memory using situation is changed, hot-plug technology can cheat the interfaces of management of operating system, so the memory can be allocated freely again. To us, the virtual machine memory space has the ability of scalability.

2.4 Classification of Memory Balancing Technology

Memory balancing technology can be classified into two types: single physical machine technology and multi-physical machine technology.

Literature [17] proposed memory balancing methods based on black box technology and grey box technology. Magenheimer [17] proposed the mechanism of self balloon which is based on Xen balloon driver. Zhao [8] proposed a memory prediction method and a memory allocation mechanism which are based on missing page rate curve. In literature [18] a dynamic memory reflection model which can combine para-virtualization, shadow page and hard-aided virtualization with virtual storage technology and memory sharing is proposed. Literature [19] brings a two-layer address space mechanism and constructs a virtual memory optimization frame of crossing multi-physical machines.

Page reuse technology which is researched in [9, 20] and dynamic adjusting memory mechanism in literature [3] has provided the basic support method, but they can not solve the problem of when and how to allocate the memory space. Literature [21] and literature [22] are absent of the global management system frame and assisted scheduling methods. In order to solve the above problems, Yaqiong Li et al.[22] proposed a global memory optimization frame of cloud computing platform. In this pattern, logic address and global address construct a two-layer address mapping. So an abstraction of physical resource can be used at any time. In 2011, Weizhe Zhang [7] proposed an idea that virtualization technology has made the dynamic arrangement of the resource and security isolation can be achieved easily. And in his paper, the concept of spontaneous adjustment and global adjustment are proposed. Rich state of memory resource and shortage state of memory are defined too. And the corresponding algorithm is achieved at the same time [7].

From the analysis above we found that most of existing research does not consider user service satisfaction, priority of virtual machines and real-time of virtual machine allocation. In this paper we will address this issue. And the memory allocation situation will be more balanced according to users real memory requirement.

3. Problem Statements

Virtualization technology plays an important role in distributed cloud platform. Now the key point of virtualization and the essence of virtualization is the maximize resource allocation with a proper way. In the way of time-sharing and multi-machine reuse technology, CPU and other I/O devices can be used high efficiently. To memory resource, the above methods cannot solve the problem of memory optimization. Now the utility of memory problem has become a bottleneck of how to solve other key problems. So the proper utility of memory and allocation is a special important problem which is waiting to be solved urgently.

We know physical memory can be allocated by operating system with a balanced method. So the key and main problem is that we will pay attention to how to improve user service satisfaction in our paper.Resource over being allocated and wasted resource can all be avoided in the process of virtualization memory allocation. In cloud platform, all virtual machines can obtain their proper memory resource in proper times which is the basic process of multi-machine memory allocation. And at the same time, all the tasks can be finished in the shortest time. The goal of allocation of memory is using the least cost and the shortest time of users in a balancing state. The problem of memory resource allocation has been proved as a NP-hard problem. In this paper, in order to solve the resource allocation problem, a makespan mechanism based on user preference is proposed with a high efficient way.

3.1 Problem Description

In general, all physical resources are limited. Let there are m virtual machines in a single cloud platform, all the resources will be assigned properly to all virtual machines. And we will be sure that virtual machines can finish their tasks in a short time as far as possible. Let the sequence number of every virtual machine is $VM_1, VM_2, VM_3, \ldots\ldots, VM_m$. This situation is equal to that the $PMith$ (i<1<n)memory is assigned to $V_jth(1<j<m)$ virtual machine. That is to say, m virtual machines are scheduled to n memory blocks in some orders. And we treated every task as a virtual machine in this paper.

In general cloud platform can be constructed with many different computing nodes. And all the nodes maybe isomorphism or heterogeneous. The memory of every device can be treated as different memory blocks which have different capacity with each other. In virtual machine monitor layer, the problem will be solved easily if all the memory nodes can be formed into a logical view. Then sometime the logical view can be described into m logical memory units (for example, if the whole space is N, then every unit space is N/m), we can simulate the memory unit into m dependent memory processors. And the m virtual machines can be simulated into m different tasks. So we can see virtual machines as tasks which are assigned to n different memory processors. So the problem is transformed into a load balance problem (In the real situation, because of multi-virtual machines, all states of the different virtual machines are not same. So the memory is very hard to be assigned to the virtual machine because of the complex situations. So it is easy to make the memory resource allocation in an unbalance state, and the memory efficiency is very low.)

Now in order to solve the problem of allocating the different memory units to different virtual machines, many different methods are proposed. But the weakness is that the efficiencies of the above algorithms are very low. This paper has abstracted the whole process from other view which has changed the traditional recognition. We improve the memory utilization efficient from the view of memory optimization.

Now how to assign different virtual machines to different pseudo processors (every virtual memory block is supposed to be a processor) is the key problem. During the allocation process, we should try our best to balance every processor (In this paper, we suppose the address space of every processor is big enough.).

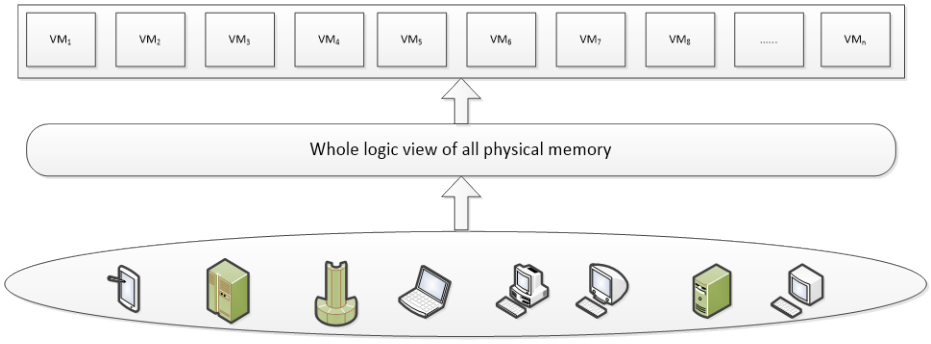

Fig. 1. Distribution of devices

3.2 Related Symbols and Definitions

In general, operating system will create different virtual machines and allocate hard resource to every virtual machine according to different requirements. In this paper, only memory

optimization mechanism is discussed. The methods of how to assign resource properly to every virtual machine can not only satisfy the memory requirement of every virtual machine but also save memory resource with limited time and price cost. This problem can be seen as that how to allocate multi-virtual machine to the fixed memory at the same time which can be defined as a typical makespan problem based on user service preference.

DEF 1：User preference

The cost and time which are consumed during the process of cloud service are key factors of user preference. The less time and cost is the main pursuit goal of all the tasks for different users. The perference of $user_i$ can be described below:

$$\begin{aligned} &\mathrm{P}(\mathrm{u}_i) = [\eta * time(j) + (1-\eta) * \cos t(j)]^{-1} \\ &(\eta \subseteq [0,1], P(\mathrm{u}_i) \subseteq [0,1]) \end{aligned} \tag{1}$$

η is the harmonic coefficient, $cost()$ is a price function, $Time()$ is a time function. i denotes the user number and j denotes the task number.

$$Time(j) = \left[\left[S(u_i) \right]^{-1} - (1-\eta) * \cos t(j) \right] \Big/ \eta \tag{2}$$

DEF 2: User Preference Set

Preference set is a collection of all the initial values which depends on the task assigning degree and the interval time of task finishing. The interval time is a sub of start time and end time. Let $\min(s)$ be the min value and $\max(s)$ be the max value of an attribute of Preference set. All the values of the Preference in the domain is [0,1]. So we adopt a method of minimum & maximum normalization to make linear transformation.

$$\begin{aligned} &v' = [(v - \min(s)) / (\max(s) - \min(s))] * \\ &(new\_\max(s) - new\_\min(s)) + new\_\min(s) \end{aligned} \tag{3}$$

Through the mapping, all the Preference value can be transformed into the domain of $[new\_\max(s) - new\_\min(s)]$. So the parameters formalization will be easy.

DEF 3: Threshold of Satisfaction

$$Threshold\left(P(u_i)\right) = Avg(P(u)) \tag{4}$$

$$Sum\left(P(u_i)\right) = \sum_{i=1}^{n} P(u_i) \tag{5}$$

$$\text{So } Avg(P(u)) = \sum_{i=1}^{n} P(u_i) / \mathrm{n} \tag{6}$$

$Goal(U_i) = Min(\mathrm{Max}_{1 \le i \le n} P(u_i))$ means that most of the users can get the basic equal preference under the condition of that the biggest satisfaction of all the users tries to keep the smallest, that is to say that all the differences of the satisfactions keep small.

If we want to make the preference(satisfaction) of every user keep max as far as possible, the key factor is that the running time on every memory block should be less. Suppose that the cost of every user and task size is proportional to each other. So it is fair to every virtual machine on the factor of cost. And the only key factor of user satisfaction is time except cost.

We describe every physical block with number $M_1, M_2, M_3, \ldots\ldots, M_n$, and then the *ith* memory block running time is $TR(m_i)$. Because many virtual machines maybe will be assigned to the same block (Let the virtual block space be big enough), the final goal now turned into $\min(max(TR(m_i))$.

DEF 4: Symbol Definition Set

ZET_{ij}: The expected execution time of tasks T_i on virtual machine node VM_j.

$ZET_{ij} = \infty$ denotes that task T_i can be run on the node VM_j (of course sometimes it maybe be hung up or be short of resource)

R_i: denotes the arriving time of task T_i.

ZZT_j: denotes the early processing time of node VM_j.

FT_{ij}: denotes the expected ending time of task T_i on processing node of VM_j, in general $FT_{ij} = ZZT_j + ZET_{ij}$.

When virtual machine V_i is assigned to node VMP_j, FT_i denotes the finishing time of task T_i, then running finishing time of task set $Makespan(T) = Max_{ni \in T}(T_i)$.

RFT_{ij}: denotes the fact running time of task V_i in virtual machine which is running on the node VM_j. Nowadays the goal of scheduling algorithms is to get the minimum $Makespan(T)$ [23, 24].

In this paper, we try our best to look for some conditions to satisfy user's requirements in the view of users. According to the above analysis, the problem of common user satisfaction can be transformed into the problem of user satisfaction makespan based on time to solve.

Table 1. The symbol table

Signal	Meaning
ZET_{ij}	the expected running time of virtual task on virtual node VM_j
R_i	the arriving time of task T_i
ZZT_j	the early using time of node VM_j
FT_i	the finishing time of task T_i
FT_{ij}	the expected time of task T_i on node VM_j
RFT_{ij}	the fact running time of virtual machine task V_i on node VM_j

DEF 5:User Preference Task Priority

User task priority depends on the whole cost of every unit $P(VM_i) = \mu * cost(j)/n$, where μ is a constant, and n denotes the quantity of all the tasks which were submitted by users. Without loss of generality, the higher of cost the higher of user task priority.

In the paper, VMP ($VMP = \{VMP_1, VMP_2, \ldots, VMP_m\}$) denotes the set of all the processor nodes of VMM (Virtual Machine Monitor, VMM). In order to assign memory resource to different virtual machines balanced, there should be a safe hardware and software running environments. Any error of hardware and software can lead to the running environments be disrupted. And the computing task cannot be finished normally. The stability of hardware and software lead to the different running environment of task implementing. Every node has useful state and unavailable state. If the processing node is on the unavailable state (the

physical memory address which is mapping by virtual memory has been allocated up), the task cannot be run on the node, and even the running information maybe is missing. The unavailable times divide the whole times running on computing node is called node efficiency. Let j denote the failure rate of memory node. And we suppose that all the unavailable processes of processing nodes are independent.

V ($V=\{v_1, v_2,\ldots,v_m\}$) denotes the set of virtual machines. We suppose that all the virtual machines are independent with each other. We abstract all the virtual machines into many independent tasks. So in essence, the scheduling problem is how to assign m virtual machines ($V=\{v_1, v_2,\ldots,v_m\}$) to n processing virtual nodes $VMP=\{VMP_1,VMP_2,\ldots,VMP_n\}$. Task scheduling result can be denoted by a two-dimension matrix, in which $v_{ij}=1(v_{ij}\in X)$ denotes V_i will be assigned to node VMP_j and running normally, otherwise $v_{ij}=0$. Task assignment matrix is denoted as bellow:

$$AssignmentTask[X]=\begin{pmatrix} v_{11} & v_{12}\cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{m1} & v_{m2}\cdots & v_{mn} \end{pmatrix} \tag{7}$$

DEF 6: Capacity-makespan

In any allocation of virtual machine (VM) requests to physical machines, we can let $VMRE(i)$ denote the collection of VM requests allocated to machine PM_i. Under this allocation situation, machine PM_i will have total workload.

$$TWL=\sum_{j\in VMRE(i)} m_j t_j \tag{8}$$

m_j is the memory requests of VM_j and t_j is the span of request j (i.e., the length of processing time of request j).The aim of load balancing is to minimize the maximum load (capacity makespan) on any physical machine.

In this paper, the definition of makespan is the same as traditional way in all the references, so the capacity makespan of all the physical machines maybe be defined as bellow:

$$Capacity\_makespan = max_i\left(TWL_i\right) \tag{9}$$

DEF 7: Imbalance degree of virtual machine

The imbalance degree of virtual machine reflects the allocation balance degree of memory resource and CPU resource. So the value of it depends on virtual memory mapping situation and logical virtual CPU unit using states. The variance is a popular concept which is always used to measure how far a collection of values are spread out from each other in domain of statistics. Using variance, an imbalance degree of virtual memory and CPU is defined as bellow:

$$IBD_{memory}=\sum_{j=0}^{n}(Memory_j-Memory_{avg})^2 \Big/ n \tag{10}$$

$$IBD_{cpu}=\sum_{j=0}^{n}(CPU_j-CPU_{avg})^2 \Big/ n \tag{11}$$

The factor of the imbalance degree of virtual machine is a complex indicator about memory and CPU, so $IBD_{VM} = IBD_{Memory} + IBD_{cpu}$.

In order to allocate the memory to virtual machine, the virtual memory block should be assigned at any time[25]. In our paper, the definition of usability of virtual memory block is first proposed to be used in memory load balance. In essence, the usability of virtual memory block is a definition which is based on task scheduling efficient. If there is no a virtual memory block can be used, any virtual machine cannot be sure that all the tasks can be finished in the most shortest and proper time with limited memory resource. It can be used to describe the ability probability of system providing the key memory space to virtual machine. In this paper, the concept is involved to the domain of computing platform and to support the memory cooperation of multi-virtual machines. It also can be used to show the running ability of virtual machine to obtain the normal memory resource. The definition of usability of memory block is as below.

DEF 8: Usability of Virtual Memory Block

In the environment of cloud computing, the probability of supporting the normal running store space for series of virtual memory nodes is named as memory block usability.

Let $U(V_i, VMP_j)$ denotes the probability of task V_i which can run on the processing node VMP_j . This probability depends on $DEM(V_i)$ which is assigned to the virtual machine and the real usability quantity of the memory block which is denoted as $R(VMP_j)$.

$U(V_i, VMP_j) = 1$ denotes that the memory unit is available(if $R(VMP_j) - Dem(V_i) > 0$)

$U(V_i, VMP_j) = 0$ denotes that the memory unit is not available(if $R(VMP_j) - Dem(V_i) <= 0$)

Let $Y = Sum(t_i)$ denotes the quantity of available memory units when $time = t_i$. According to the description above, the probability is $exp(-\lambda_j t)$ when the normal task can be running during the time t on the processing node [25]. Because the task cannot run until all the processing nodes are on normal states. So we can obtain the below result:

$$U(V_i, VMP_j) = exp(\delta_j * RFT_{ij}) \tag{12}$$

Let $U(V, P, X)$ denotes the executing usability of m tasks which depends on the scheduling result on n virtual memory nodes. Then,

$$U(V, VMP, X) = \exp\left(-\sum_{i=0}^{i=n}\sum_{j=0}^{j=m}\delta_i \times V_{ij} \times FT_{ij}\right) \tag{13}$$

In order to avoid the virtual machine task being assigned to the virtual memory block whose usability is low, we let the usability be the scheduling goal. In this way, the waiting time of virtual machine can be reduced and the running speed can be improved greatly. The normal usability of virtual machine running can be higher, so the efficiency of virtual machine being allocated can be improved also. At the same time, the resource utility is improved too.

Let $C(V_i, VMP_j) = \delta_i * RFT_{ij}$, then if we want to improve $U(V_i, VMP_j)$, $C(V_i, VMP_j)$ should be decreased.

In the same principle, let $C(V, VMP, X) = \sum_{i=0}^{i=n}\sum_{j=0}^{j=m}\delta_i \times V_{ij} \times RFT_{ij}$. If we want to increase the value of $U(V, VMP, X)$, then $C(V, VMP, X)$ should be decreased. So the optimization mechanism

takes the max memory block usability as the goal should make sure that the value of $C(V,VMP,X)$ would be small enough [29].

4. System Model and Problem Solving Method

The formulation of this problem can be described as follows. Given a set of identical virtual machines memory blocks (VMMBs). We denote it as a set with the name VMMB. And a set of n user requests. Each request has a processing time (consider only CPU processing for example), the objective of load balance is to assign each request to one of the VMMBs so that the loads placed on all memory blocks are balanced as far as possible.

4.1 System Model

In this paper, we can simplify the problem into a daily physical problem: let there are m cups which capacity is different and be full of water. Now we want to inject water of m cups into n same cups. Some cups are empty and some of them are full of water. Now we want to pour water of the m cups into n cups, and all the water quantity of the cups is equal to each. This can be seen as a load balance problem. But in many literatures, this problem is proved as a NP-hard problem. So the only method is the approximate. But now the difference is that we have a parameter that is memory block usability. First let us recall the general definition of makespan problem which is defined as below. The goal function is that making the max running time of the task to be min as far as possible. The makespan model is defined as below:

$Makespan(I_Q, P_{Q,} F_{Q,} OPT_Q)$

I_Q : the set of tuples $T = \{< t_1, t_2, \ldots, t_n >; m\}$, where t_i is the processing time for the *ith* cup-job and m is the number of identical beakers.

P_Q : $P_Q(T)$ is the set of partitions $B = (b_1, \ldots, b_m)$ of the numbers $\{t_1, t_2, \ldots, t_n\}$ into m beakers subsets.

F_Q : $F_Q(T, B)$ is equal to the processing time of the largest subset in the partition B, that is, $F_Q(T,B) = Max_i\{\sum_{t_j \in T_i} t_j\}$

OPT_Q : $Min(F_Q(T,B))$.

Task scheduling is a NP-hard problem[15,16], so the general algorithm is heuristic algorithm which requires a local goal function. The scheduling goal is closed to the goal function. In order to implement a heuristic task scheduling algorithm, the local goal function based on scheduling aim should be assured first. So in order to implement the multi-goal Makespan memory scheduling algorithm which is based on user preference set and virtual memory block usability, the aim function should be created first. The goal function should make sure that the memory block usability will be strong. The concrete problem formulation can be described as bellow.

User-Preference- $Makespan(I_Q, P_{Q,} F_{Q,} OPT_Q)$

I_Q: the set of tuples $SU = \{< su_1, \ldots, su_n >; m\}$, where it is the virtual user preference (satisfaction) for the *ith* virtualization machine job and m is the number of identical logical memory processors.

P_Q : $P_Q(T)$ is the set of partitions $V = (v_1, \ldots, v_m)$ of the numbers $\{t_1, t_2, \ldots, t_n\}$ into m subsets.

$F_Q : F_Q(SU,\ V)$ is equal to the processing time of the largest subset in the partition V, that is, $\mathrm{P}(\mathrm{u}_i) = [(1-\eta)*\cos t(j) + \eta * Time(j)]^{-1}$, $(\eta \subseteq [0,1], P(\mathrm{u}_i) \subseteq [0,1])$ and $F_Q(\mathrm{SU},V) = \max_i\{\sum_{suj \in PU_i} pu_j\}$.

OPT_Q: $Min(F_Q(SU,V))$

4.2 Problem Solving Method

During the process of cloud computing scheduling, because the fact running time of task on computing node cannot be predicted generally, the running time of virtual machine need to be obtained before they are running. So the local goal functions during the scheduling mostly focus on the prediction of running time. Today many prediction methods of network grid task running time are researched, and some famous prediction methods are proposed [26, 27].

In nowadays, there are many heuristic algorithms of dependent task, such as min-max algorithm , genetic algorithm and so on. The goal of traditional heuristic scheduling algorithm is to obtain the shortest task finishing time (makespan). In this paper we call this algorithm as heuristic scheduling algorithm which is based on traditional min time makespan (for example, Graham-Schedule Algorithm[28]). In order to obtain the shortest task finishing time, the task of shortest finishing time will be selected from the task which will be assigned to processing nodes to run. And the only factor of this algorithm is time.

By using a data structure such as a 2-3 tree, we can organize the m processors using their loads as the keys, we can always find the lightest loaded processor, update its load, and reinsert it back to the data structure in time $O(logm)$. With this implementation, the algorithm Graham-Schedule runs in time $O(nlogm)$. In order to get the high efficiency of task scheduling algorithm, in this paper, the priority of virtual machine, usability of virtual memory block and user satisfaction are combined together for constructing the new goal function. So the new local function is proposed as bellow.

$$F_Q(SU,V) = Max_i\{\sum_{i=1}^{m}{}_{pu_j \in PU_i} \psi\, pu_j * (1-\psi)\mathrm{pri(vi)}\} \tag{14}$$

The local function above is determined by the parameter of user satisfaction and the parameter of virtual machine priority. We call the algorithm as improved satisfaction and priority scheduling algorithm (ISAPS) which is based on user satisfaction and priority only. Because of the conflict of the factor of virtual machine block and user satisfaction, we cannot get the best result at the same time. So we select a parameter as $\psi (0 \le \psi \le 1)$. In order to adjust the different goals, user can change the value of the parameter which can change the weight of the goal function. When the value of ψ increased, the weight of the first item will be increased, so the scheduling will pay more attention to the user satisfaction, otherwise, the scheduling algorithm will pay attention to the priority of virtual machine. When $\psi = 0$, ISAPS will change into an algorithm which only considers the factor of virtual machine priority, and when $\psi = 1$, ISAPS will change into an algorithm which only considers user satisfaction preference. The ISAPS algorithm is described as bellow which is an improved edition of SAPS algorithm:

Algorithm 1: ISAPS algorithm

Input: $I = < v_1, v_2, \ldots\ldots, v_n; m >$, all integers

Output: a scheduling of the n tasks of processing time of $v_1, v_2, \ldots\ldots, v_n$

For all VMs in scheduling tasks vi(in a random order)

Sort all tasks according to user preference and $pri(v_i)$

Save all the tasks in a queue according to the sorted tasks

For i=1 to m

$$f_Q(pu_i, vmp_i) = \max_i \{\sum_{i=1}^{m} {}_{pu_j \in PU_i} \psi pu_j * (1-\psi) pri(v_i)\}$$

Do until all tasks in VM are mapped in all the virtual machines

For each task in v_i find the maximum earliest completion time and corresponding processing node

Find the task t_i with the maximum satisfaction $f_Q(su_i, vmp_i)$

Assign task v_i to the memory node vmp_i that give the max $f_Q(su_i, vmp_i)$

Delete t_i in V

Update all the ready v_i of each vmp_i assigned a new t_i

Update all $f_Q(su_i, vmp_i)$ for all i

5. Performance Evaluations

5.1 Experimental Setup and Simulation

In this section, we compare simulation results of different scheduling algorithms regarding load balance. We use cloud computing simulation platform cloudsim3.0 which is developed by University of Melbourne. We have implemented the algorithms that we discussed above. In the experiments, we used 100 physical nodes to construct the cloud computing environments. And about 1000 independent virtual machine tasks set are researched. We will show the experimental results of ISAPS, Random Algorithm, Round-Robin, List Scheduling algorithm and FCFS algorithm. All simulations are conducted on an Inter(R) Core(TM) i5-3317U computer with 1.7GHz CPU and 4GB memory. We compare the results of our proposed algorithm ISAPS with other four existing algorithms below:

(1) Random Algorithm: a general scheduling algorithm by randomly allocating the VM requests to the VPM which can provide the logic resource required.

(2) Round-Robin algorithm: a traditional load balancing scheduling algorithm by allocating the VM request one by one to each virtual memory block in turn that can provide resource required.

(3) List Scheduling algorithm: One of the well-known online traditional load-balancing algorithm, it selects the available PM with the lowest current workload to allocate virtual machine [29].

(4) FCFS algorithm: according to the arriving time sequence of VMs, first come first serve.

For simulation, to be realistic, we adopt the log data at Lawrence Livermore National Lab [30]. That log contains months of records collected by a large Linux cluster. Each line of data in that log file includes eighteen elements. In our simulation experimental environment, we only need the request ID, start time, duration time, and processor requirement. To enable those data be fit with our simulation, some conversions are needed, like we convert the units from seconds in Lawrence Livermore National Lab log file into minutes, because we set a minute as a time slot length mentioned in previous section[30]. Another conversion is that processor number needed in Lawrence Livermore National Lab log file has been changed into eight types of VM requests. To simplify the simulation, three types of heterogeneous PMs and eight types of VMs are considered (can be dynamic configured and extended). We do the simulation with enough physical machines which could satisfy all the virtual machine requests and VM numbers vary from 200 to 800 (each type approximately one eighth). The simulations for different algorithms are based on the same environment with same VM requests.

Table 2. Types of virtual machine

Status	Cloud id	Memory	Cpu unit	storage	Start time	Finish time	Vm-id
T	0	1.8	2	180	0.01	0.21	1
T	5	2.4	3	240	0.01	0.24	2
T	1	3	3	300	0.01	0.24	3
T	2	7.5	12	500	0.01	0.06	4
T	4	18	14	1024	0.01	0.04	5
T	7	36	24	1000	0.01	0.03	6
T	6	70	50	2050	0.01	0.02	7
T	9	8	9	870	0.01	0.32	8
T	3	28	20	2020	0.01	0.21	9
T	8	2.6	4	400	0.01	0.24	10
T	101	54	30	1400	0.01	0.11	11
T	106	1.7	2	180	0.01	0.22	12

Table 3. Eight types of logical PM suggested

Type id	Cpu units	Memory(G)	Storage(T)
1	16	30	1T
2	64	135	2T
3	14	27	3T
4	98	180	3T
5	65	140	4T
6	260	500	4T
7	130	200	4T
8	600	1024	6T

The only difference lies in the scheduling process of each algorithm, including all PMs are turned on at the beginning in other algorithms. While in ISAPS, PMs are turned on one by one according to the VM requests. To be fair, if the actual total number of turn-on physical machines is not the same for different algorithms, all metrics such as capacity makespan, and imbalance degree are adjusted by timing a co-efficiency.

5.2 Experimental Results

With the increasing of the workload and adjusting the values of the parameters (lambda and DT), we first obtain the comparison of work efficiency. The compared results are showed in **Fig. 2**.

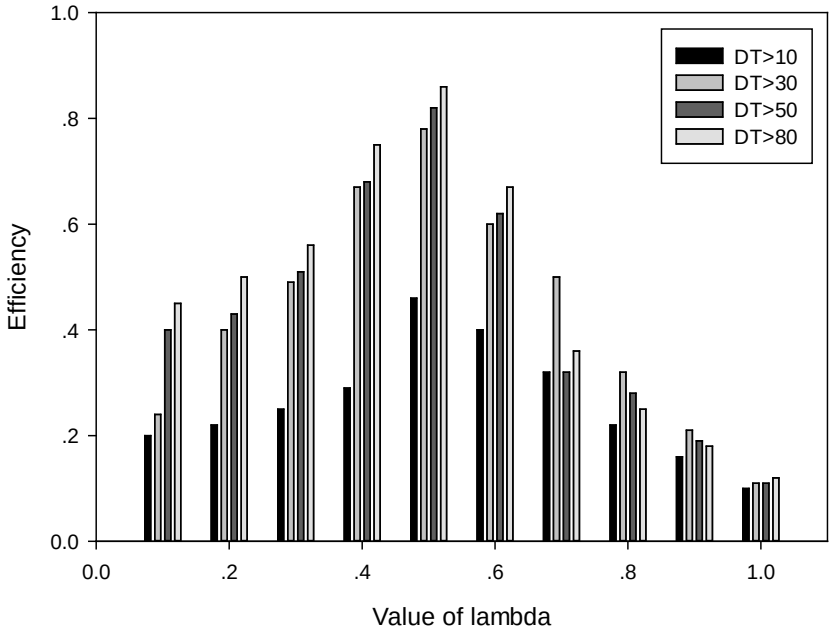

Fig. 1. Efficiency with different values of the parameters (lambda and DT)

Strongly divisible capacity of jobs and machines denote the capacity of all VMs with a divisible sequence, for example, the sequence of distinct capacities , taken on by VMs (the number of VMs of each capacity is at random) is such that for all , exactly divides , the largest item capacity in L exactly divides the capacity C[26].

The corresponding relationship is as follow. We use VM type 1 2 3 for PM1 and PM2, VM type 4 5 6 for PM3 and PM4, VM type 7 8 9 for PM5 and PM6, VM type 10 11 12 for PM7 and PM8.

Fig. 3 - **Fig. 5** as follow show the imbalance degree, makespan, satisfaction of makespan and capacity makespan results respectively when fixing total number of VM requests as 400 but changing the max duration of the virtual machines. The results are the average value of six times simulation of the same inputs. From these figures, we notice that ISAPS algorithm shows the best performance in imbalance degree, makespan, capacity makespan, satisfaction capacity makespan compared with other four algorithms.

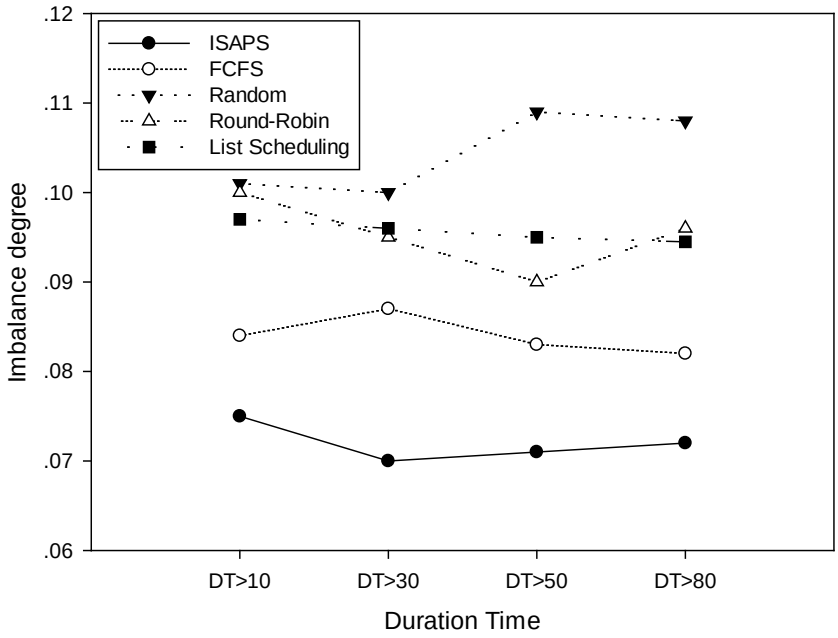

Fig. 2. Imbalance degree comparison when varying duration of VMs

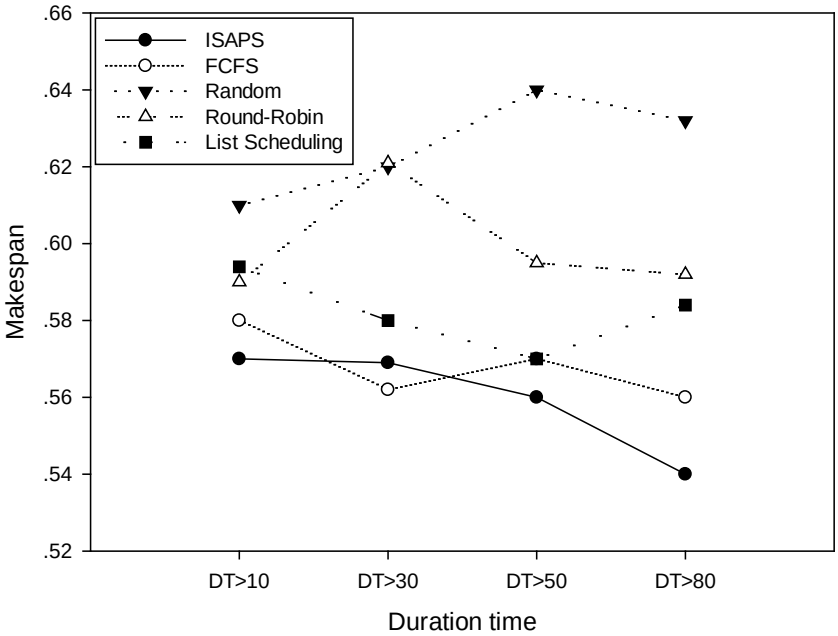

Fig. 3. Makespan comparison when varying duration of VMs

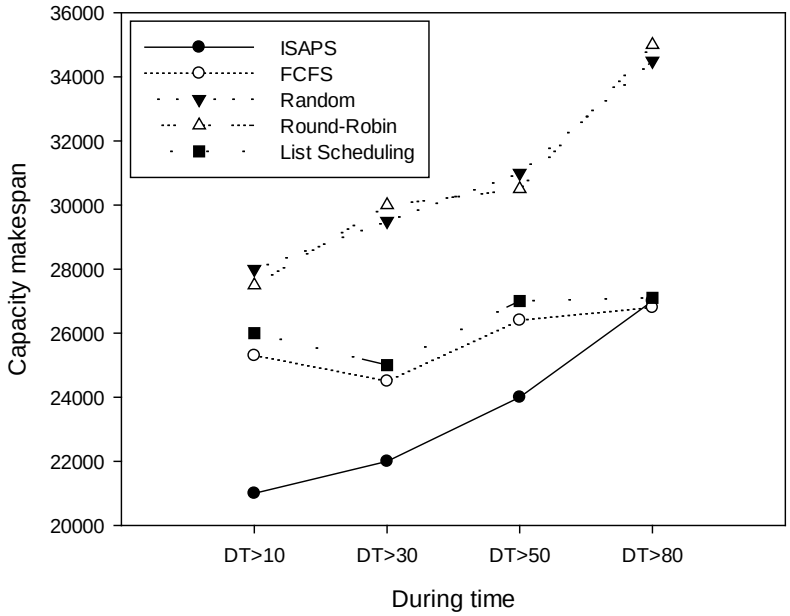

Fig. 4. Capacity makespan comparison when varying duration of VMs

6. Conclusions

As is well-known, all kinds of physical resource should be used in a proper way. Resource allocation with a balancing way is a key factor which can plays an important role in cloud data centers. In this paper, in order to solve resource allocation problem and to reflect capacity sharing property in cloud data centers, we propose an algorithm named as improved satisfaction and priority scheduling algorithm. The algorithm is based on user satisfaction and priority with new metrics such as user preference (satisfaction), virtual machine priority, capacity makespan and so on. In section 5, cloud computing simulation platform cloudsim3.0 which is developed by University of Melbourne is used in our experiment. And at the same time we have implemented all the algorithms that we discussed above. In a nutshell, simulations results have shown that ISAPS has better performance than a few existing algorithms at imbalance degree, capacity makespan as well as user preference (satisfaction) of service.

References

[1] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, et al., "Ananta: cloud scale load balancing," in *Proc. of Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 207-218. Article, 2013. Article (CrossRef Link)

[2] F. Hao, M. Kodialam, T. Lakshman, and S. Mukherjee, "Online allocation of virtual machines in a distributed cloud," in *Proc. of INFOCOM, 2014 Proceedings IEEE*, pp. 10-18, 2014. Article (CrossRef Link)

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, et al., "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, pp. 164-177, 2003. Article (CrossRef Link)

[4] Y. Zhou, L. Iftode, and K. Li, "Performance evaluation of two home-based lazy release consistency protocols for shared virtual memory systems," *ACM SIGOPS Operating Systems Review*, vol. 30, pp. 75-88, 1996. Article (CrossRef Link)
[5] K. Qureshi, B. Majeed, J. H. Kazmi, and S. A. Madani, "Task partitioning, scheduling and load balancing strategy for mixed nature of tasks," *The Journal of Supercomputing*, vol. 59, pp. 1348-1359, 2012. Article (CrossRef Link)
[6] S. Chandara and R. Abdullah, "A study on implementation of idle network memory virtualization for cloud," in *Proc. of Open Systems (ICOS), 2011 IEEE Conference on*, pp. 53-58, 2011. Article (CrossRef Link)
[7] Z. W.-Z. Z. Hong-Li and Z. D. C. Tao, "Memory cooperation optimization strategies of multiple virtual machines in cloud computing environment," *Chinese Journal of Computers*, vol. 12, p. 003, 2011. Article (CrossRef Link)
[8] W. Zhao, Z. Wang, and Y. Luo, "Dynamic memory balancing for virtual machines," *ACM SIGOPS Operating Systems Review*, vol. 43, pp. 37-47, 2009. Article (CrossRef Link)
[9] K. Sato, M. Samejima, and N. Komoda, "Dynamic optimization of virtual machine placement by resource usage prediction," in *Proc. of Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pp. 86-91, 2013. Article (CrossRef Link)
[10] C. A. Waldspurger, "Memory resource management in VMware ESX server," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 181-194, 2002. Article (CrossRef Link)
[11] C. Min, I. Kim, T. Kim, and Y. I. Eom, "Vmmb: Virtual machine memory balancing for unmodified operating systems," *Journal of Grid Computing*, vol. 10, pp. 69-84, 2012. Article (CrossRef Link)
[12] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," *Journal of Computers*, vol. 7, pp. 42-52, 2012.
[13] T. Harvey, C. Newell, and C. Laplace, "Cooperative memory resource management for virtualized computing devices," ed: Google Patents, 2014. Article (CrossRef Link)
[14] J. Sugerman, G. Venkitachalam, and B.-H. Lim, "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor," in *Proc. of USENIX Annual Technical Conference, General Track*, pp. 1-14, 2001. Article (CrossRef Link)
[15] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *Proc. of INFOCOM, 2013 Proceedings IEEE*, pp. 355-359, 2013. Article (CrossRef Link)
[16] M. Gahlawat and P. Sharma, "Survey of virtual machine placement in federated Clouds," in *Proc. of Advance Computing Conference (IACC), 2014 IEEE International*, pp. 735-738, 2014. Article (CrossRef Link)
[17] D. Magenheimer, "Memory overcommit without the commitment," *Xen Summit*, pp. 1-3, 2008. Article (CrossRef Link)
[18] H. Chen, X. Wang, Z. Wang, B. Zhang, Y. Luo, and X. Li, "DMM: A dynamic memory mapping model for virtual machines," *Science China Information Sciences*, vol. 53, pp. 1097-1108, 2010. Article (CrossRef Link)
[19] K. Govil, D. Teodosiu, Y. Huang, and M. Rosenblum, "Cellular disco: resource management using virtual clusters on shared-memory multiprocessors," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, pp. 229-262, 2000. Article (CrossRef Link)
[20] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, et al., "Difference engine: Harnessing memory redundancy in virtual machines," *Communications of the ACM*, vol. 53, pp. 85-93, 2010. Article (CrossRef Link)
[21] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *NSDI*, pp. 17-17, 2007. Article (CrossRef Link)
[22] Y.-Q. Li, Y. Song, and Y.-B. Huang, "A memory global optimization approach in virtualized cloud computing environments," *Chinese Journal of Computers*, vol. 34, pp. 684-693, 2011. Article (CrossRef Link)

[23] H. Liu, H. Jin, X. Liao, W. Deng, B. He, and C.-z. Xu, "Hotplug or ballooning: A comparative study on dynamic memory management techniques for virtual machines," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, pp. 1350-1363, 2015. Article (CrossRef Link)
[24] L. Wang, H. Wang, L. Cai, R. Chu, P. Zhang, and L. Liu, "A Hierarchical Memory Service Mechanism in Server Consolidation Environment," in *Proc. of Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pp. 40-47, 2011. Article (CrossRef Link)
[25] S.-p. Wang, X.-c. Yun, and X.-z. Yu, "Research on multi-objective grid task scheduling algorithms based on survivability and Makespan," *Journal of China institute of communications*, vol. 27, p. 42, 2016. (Article (CrossRef Link)).
[26] L. Gong, X.-H. Sun, and E. F. Watson, "Performance modeling and prediction of nondedicated network computing," *Computers, IEEE Transactions on*, vol. 51, pp. 1041-1055, 2002. Article (CrossRef Link)
[27] J. Noudohouenou and W. Jalby, "Using static analysis data for performance modeling and prediction," in *Proc. of High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pp. 933-942, 2014. Article (CrossRef Link)
[28] J. ChEn, "Introduction to Tractability and Approximability of Optimization problems," *Lecture Notes*, Department of Computer Science, Texas A&M University, pp. 833-847, 2002. Article (CrossRef Link)
[29] Minxian Xu, Wenhong Tian. "An Online Load Balancing Algorithm for Virtual Machine Allocation with Fixed Process Intervals," *Journal of Information & Computational Science*, 11(3), pp. 989-1001, 2014. Article (CrossRef Link)
[30] Hebrew University, Experimental Systems Lab, www.cs.huji.ac.il/labs/parallel/workload, 2014. Article (CrossRef Link)

Shukun Liu received his Ph.D. degree in Computer Science and Technology from Central South University, PR China, in 2016. His major research interests include cloud computing, virtualization technology, performance analysis, computer networks, database technology, data mining, and software engineering. He has published nearly twenty papers in related journals, and he is a member of CCF and ACM.

Prof. Weijia Jia is a Professor in the Faculty of Science and Technology of University of Macau He joined the German National Research Center for Information Science (GMD) in Bonn (St. Augustine) from 1993 to 1995 as a research fellow. During 1995–2013, he joined the Department of Computer Science, City University of Hong Kong as a professor. He has served as editor of the IEEE TPDS and PC chair, and a member/keynote speaker for various prestigious international conferences. He is also a Senior Member of the IEEE and a Member of ACM.

Prof. Xianmin Pan is a Professor in the Department of Information Technology, Hunan Women's University. His major research interests include cloud computing, virtualization technology, performance analysis, computer networks, database technology, data mining, and software engineering. He has published nearly twenty papers in related journals.