An open Scheduling Framework for QoS resource management in the Internet of Things

Weipeng Jing , Qiucheng Miao and Guangsheng Chen
Department of Computer science and technology, Northeast Forestry University
Harbin, P.R.China
[e-mail: weipeng.jing@outlook.com]
*Corresponding author: Weipeng Jing

Received April 1, 2017; revised February 6, 2018; accepted March 19, 2018; published September 30, 2018

Abstract

Quality of Service (QoS) awareness is recognized as a key point for the success of Internet of Things (IOT).Realizing the full potential of the Internet of Things requires, a real-time task scheduling algorithm must be designed to meet the QoS need. In order to schedule tasks with diverse QoS requirements in cloud environment efficiently, we propose a task scheduling strategy based on dynamic priority and load balancing (DPLB) in this paper. The dynamic priority consisted of task value density and the urgency of the task execution, the priority is increased over time to insure that each task can be implemented in time. The scheduling decision variable is composed of time attractiveness considered earliest completion time (ECT) and load brightness considered load status information which by obtain from each virtual machine by topic-based publish/subscribe mechanism. Then sorting tasks by priority and first schedule the task with highest priority to the virtual machine in feasible VMs group which satisfy the QoS requirements of task with maximal. Finally, after this patch tasks are scheduled over, the task migration manager will start work to reduce the load balancing degree.The experimental results show that, compared with the Min-Min, Max-Min, WRR, GAs, and HBB-LB algorithm, the DPLB is more effective, it reduces the Makespan, balances the load of VMs, augments the success completed ratio of tasks before deadline and raises the profit of cloud service per second.

Keywords: QoS; Internet of Things; dynamic priority; load balancing; scheduling

This research was supported by Special Fund for Forest Scientific Research in the Public Welfare(201504307)

http://doi.org/10.3837/tiis.2018.09.001 **ISSN : 1976-7277**

1. Introduction

Full intelligentialize would be the ultimate goal of the Internet of Things. According to the forecasting application of the Internet of Things (IoT) made by [1], everyday devices will be connected to the Internet and share information with each other. Consequently new challenges arise in order to guarantee the effectiveness and the efficiency of the data processing, Cloud computing platform must be deployed at multiple scales and real time over various devices or servers, spread across IoT. this implies that it must establish a kind of “QoS-based contract” between IoT and cloud computing platform. Nevertheless, handling QoS requirements and load balancing for cloud computing platform is still essential.

Cloud computing is an internet-based computing services model which evolved from grid computing, distributed computing, parallel computing, virtualization and other technologies, it integrates enormous computing resources, storage resources and software resources via network to constitute some shared virtual resource pools, then various applications and files can be hosted on the cloud to enjoy the low-cost service which follows a “pay as you use” model. Using the cloud service, the customers are freed from massive software and hardware investment, they need not care for the upgrade of software and hardware, they just need pay money for the duration they has used the resource which they applied previously. The usual services that are provided by cloud computing can be classified into three levels: infrastructure-as-a-service(IaaS), platform-as-a-service(PaaS) and software-as-a-service (SaaS)[2]. So in the IoT application, the Cloud computing must provide high quality QoS. The completion time of all tasks and execution cost are two major factors of quality of service(QoS) which are interested by IoT devices. Then the cloud center should do it best to provide an high-speed and cheap service to satisfy various QoS requirement tasks.[4]

With the continuous exploration and research on cloud computing, it gradually shifts from "computer" as the center to "user" as the center. Due to the commercialization of cloud computing, it needs to pay more attention to the different needs of user tasks, that is to give priority to various QoS requirements of user tasks, such as completing the requirements of time, cost, energy consumption and reliability, and then concerned about the computer performance and response time. Therefore, using "user" as the center of resource allocation to improve user satisfaction and the utility value of user tasks, as the goal of resource allocation, is very suitable for solving the problem of theater allocation in cloud environment.

Task scheduling algorithm is the key to satisfy the data processing in IoT. Task scheduling algorithm in the scheduler is in charge of distributing tasks to suitable virtual machines(VMs) which are the processing units in the cloud. Scheduling of tasks in cloud computing is an NP-hard optimization problem. Scheduling algorithms are used mainly to minimize execution time and execution cost. A good scheduling algorithm should do it best to satisfy the users’ QoS requirement and utilize the available resources fully, so it should give the feasible priority to tasks and avoid load imbalance especially for the IoT data processing. The DPLB algorithm can effectively reduce the total task completion time and balancing the VMs’ load, and it also can archive a better success completion ratio and service profit per second in different task sets.

Priority based Task scheduling is one of the hot research point in cloud computing. Ghanbari et al. schedule priority based job by AHP method.[3] Gu et al. schedule priority based task on the Hadoop platform.[25] Traditional Min-Min or Max-Min scheduling

algorithm equal to give the smallest or largest task lowest priority or higher priority. Although the above algorithm reduce the overall task completion time, but did not consider the influence of value and deadline on priority, and Min-Min algorithm also prone to load imbalance. To avoid large task to be executed over a long time, G et al. draw dynamic priority into Min-Min algorithm, task priority increases with the waiting time.[5] In order to avoid task priority increases infinitely, literature[6] group tasks by priority, the tasks in the same group have the same priority, then first schedule the task with minimum deadline in the highest priority group. Although these two task scheduling algorithms increase the utilization ratio of resources, them still lack in considering the impact of execution cost on priority. Considering the residual value density of task and the urgency of the task execution, a real-time tasks scheduling algorithm based on dynamic priority in stand-alone environment was proposed in [7], but the algorithm is not available for distributed cloud environments.

Considering effective task scheduling algorithms should be able to balance the VM load in order to reduce the earliest completion time of all tasks.

The load balancing methods can be classified into dynamic load balancing methods and static load balancing methods. In the aspect of dynamic load balancing, the authors in [8] achieves load balancing on physical machine by VM migration.The authors in [9] achieves the dynamic load balancing on VM resources by exponential smoothing forecasting method. The authors in [10] achieves it by genetic algorithm based on the record of historical data and current state information on VMs. A dynamic load balancing algorithm HBB-LB was proposed in [11], in HBB-LB tasks on overloaded VMs are bees and the low loaded VMs are the food sources, imitates the foraging food behavior of honey bees, it implements tasks on overloaded VMs migration and achieve load balancing. A load balancing method which use Genetic algorithm to achieve task migration was proposed in [13], and literature achieve task migration by particle swarm optimization algorithm. The above dynamic load balancing strategies can balance the load of VMs, but the realizations are complex, and will increase the time-cost in tasks migration. In the aspect of static load balancing. In[14,19], the authors achieves load balancing by the ant colony algorithm. In [15], adds load constraint on the Max-Min algorithm. The authors in [16] first gets result of Min-Min algorithm, then adjusts the shortest tasks on overloaded VMs to other VMs which can reduce the ECT, finally schedules tasks to VMs. The realizations of above static load balancing algorithms are simpler relatively, but they are just fit for no wrong tasks set and steady cloud environment.

Considering the impact of value and deadline on task priority, and balancing the VMs load, this paper proposes a task scheduling strategy based on dynamic priority and Load Balancing (DPLB) in cloud environment. Dynamic priorities consist of task value density and the urgency of task execution, and over time, priorities increase to ensure that each task is executed in time.The scheduling decision variable ρ is composed of attractiveness considered earliest completion time (ECT) and load brightness considered load status information which by obtain from each VM by topic-based publish subscribe mechanism. Then sorting tasks by priority and first schedule the task with highest priority to the VM in feasible VMs group which satisfy the QoS requirements of task with maximal ρ. And this paper conducts a comparison among the Min-Min, Max-Min, WRR, GAs, and HBB-LB algorithm in the completion time of all tasks, load balancing degree, success completed ratio and service profit per second.

The main contributions of this paper are as follows: Proposing an new priority structure which considering the task value density and the urgency of the task execution, and proposing an new scheduling decision variable, which considers both ECT and VMs load. Using topic-based publish/subscribe mechanism to get VMs load status information for making

scheduling decision more accurate and designing a task migration manager to balance load further.

The remainder of this paper is organized as follows: Section 2 introduce the related model of DPLB algorithm. Section 3 describes DPLB algorithm including the cloud structure, some management strategies and the detail description of DPLB algorithm. Section 4 shows the experimental results and analysis. Finally, the conclusion is presented in Section 5.

2. Model

2.1 Task scheduling model

In this paper, we just consider the case of n uncorrelated independent subtasks scheduling, these tasks will be assigned to m VMs for performing $(m < n)$ .The set of n tasks is represented by $T(n) = \{t_1, t_2, ..., t_n\}$ $n \in N$, which t_i $(\mathrm{i} = 1, 2, \ldots, \mathrm{n})$ is the No. i task in the set, and the attributes of every task is represented by $t_i = (t^i_{id}, t^i_{mi}, t^i_{file}, t^i_{fee}, t^i_{deadline}, t^i_{memory}, t^i_{submit})$, where: t^i_{id} is the unique identification number of t_i . t^i_{mi} is the size of t_i , namely the number of million instructions (MI). t^i_{file} is the program file size of t_i . t^i_{fee} is the user' desired fee of t_i , user gives it based on the task' QoS requirements. $t^i_{deadline}$ is the user desired deadline for t_i . t^i_{memory} is the memory requirement of t_i . t^i_{submit} is the submission time of t_i .

The set of m virtual machine resources is represented by $VM(m) = \{vm_1, vm_2, ..., vm_m\}(m \in N)$, which vm_j $(j = 1, 2, \ldots, m)$ is the No. j virtual machine in the set, and the attributes of every VM is measured as $vm_j = (vm^j_{id}, vm^j_{\mathrm{capacity}}, B^j_{vm}, vm^j_{memory})$, where: vm^j_{id} is the unique identification number of VM in the data center. $vm^j_{capacity}$ is the processing capacity of VM, namely processing capacity for million instructions per second(MIPS). $B^j = \{b_{jk} \mid k = 1, ..., m\}$ is bandwidth between vm_j with other VMs. vm^j_{memory} is the memory size of VM.

There are some important identification in task scheduling field as follows:

(1) Expected Execution Time(ETC): ETC_{ij} is the expected execution time of t_i on vm_j ,then it can be expressed as Eq.(1):

$$ETC_{ij} = \frac{t^i_{mi}}{vm^j_{capacity}} \tag{1}$$

(2) Earliest Completion Time(ECT): be_j is the start time of t_i on vm_j , ECT_{ij} is the earliest completion time of t_i on vm_j , then it can be denoted as Eq.(2):

$$ECT_{ij} = be_j + ETC_{ij} \tag{2}$$

(3) The objective function and constraints: Makespan is the completion time of all tasks, it can be expressed as Eq.(3):

$$Makespan = \max\{ECT_{ij}\} \tag{3}$$

Minimizing the Makespan is the major purpose for most task scheduling algorithms, and it also keep some aspects of QoS requirements, then it can be expressed as Eq.(4):

$$\begin{cases} \min\{Makespan\} \\ t_{memory}^{i} \le vm_{memory}^{j} (i = 1, 2, ..., n) \\ t_{deadline}^{i} \le ECT_{ij} (j = 1, 2, ...m) \end{cases} \tag{4}$$

2.2. Dynamic priority model

The priority of task reflects the importance of the task, it should be take into account the fairness and efficiency to satisfy the QoS requirements of users, such as the cost of execution, the deadline of task etc. To achieve the purpose, this paper proposes a dynamic priority which considering the task value density and the urgency of the task execution.

(1) Task Value Density (TVD)

$$TVD_i = \frac{t_{fee}^{i}}{t_{mi}^{i}} \tag{5}$$

Due to the value of task can not reflect the real value of the task, it must consider the size of task simultaneously, so we define the TVD by the ratio of the user desired fee to the task size as Eq. (5) .

(2) The Urgency of Task Execution (UTE)

$$UTE_i = \frac{t_{wait}^{i}}{1 + t_{left}^{i}} \tag{6}$$

In Eq.(6), where t_{wait}^{i} is the waiting time of task, let $t_{current}$ is the current time, t_{left}^{i} is the left time, then t_{wait}^{i} can be expressed as $t_{wait}^{i} = t_{current} - t_{submit}^{i}$. Obviously with the increasing waiting time, the left time will decrease accordingly, then the UTE will increase rapidly, so it will satisfy the time constraints of tasks, the successful completion rate of tasks completed before the deadline reflects the priority dynamic characteristics.

(3) The Dynamic Priority of Task

In order to build dynamic priority, we need to normalize the $\mathrm{TVD_i}$ and $\mathrm{UTE_i}$. In this paper, the Z-score method which based on the mean of the original data and standard deviation is used in data normalization. Let st_{ik} denotes the normalization result of the No. k priority factor of t_i , we get the Eq.(7):

$$st_{ik} = (z_{ik} - \overline{z_k}) / \delta_k \tag{7}$$

Where $(k = 1, 2)$, z_{ik} is the element in $n \times 2$ order matrix Z which is denoted as $Z = \{TVD \quad UTE\}$, $\overline{z_k}$ is the average of No. k column of matrix Z which is calculated by

$\overline{z_k} = \frac{1}{n}\sum_{i=1}^{n} z_{ik}$, δ_k is the standard deviation of No. k priority factor which is calculated by $\delta_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_{ik} - \overline{z_k})}$.

Let $DP(t_i)$ denote the dynamic priority of t_i , then it can be defined by Eq.(8):

$$DP(t_i) = \omega_1 \times st_{i1} + \omega_2 \times st_{i2} \tag{8}$$

Where $\omega_1, \omega_2 \in [0,1]$, they are the weighting factors , and they satisfy the equation $\omega_1 + \omega_2 = 1$.

2.3. Mapped decision variable model

Tasks and the virtual machines is a mutual attraction process in tasks scheduling. In this process,we design two factors which consider ECT and the load status of VMs to schedule tasks.

(1) Time Attractiveness (TA)

The reciprocal of earliest expected completion time, it will be decreasing with the ECT reducing, the ECT is not zero, then it can be denoted as Eq. (9) :

$$TA_{ij} = \frac{1}{ECT_{ij}} \tag{9}$$

(2) Load Intensity(LI)

The reciprocal of VM load expect processing time(EPT_j), let L_j is the load on vm_j , then we can get $EPT_j = \frac{L_j}{vm_{capacity}^j}$, to avoid the processing time becoming zero, we let the EPT_j plus one as the denominator, then it can be denoted as Eq.(10):

$$LI_j = \frac{1}{1 + EPT_j} \tag{10}$$

In the initial moment, the EPT_j is zero, the value of LI is one, it is the maximum. With the increasing of load, the LI will decrease.

(3) Mapped decision variable (ρ_{ij})

Considering the TA and LI, we give the mapped decision variable as Eq.(11).

$$\rho_{ij} = TA_{ij} \times LI_j \tag{11}$$

Obviously the ρ_{ij} will increase with the increasing of TA and LI, namely the load of vm_j is lower with the reduction of ECT, the ρ_{ij} will more bigger. Finally, the task scheduler schedules the task to the available VM which can archive the biggest value of ρ_{ij} .

3. DPLB algorithm

3.1. Task scheduling cloud structure

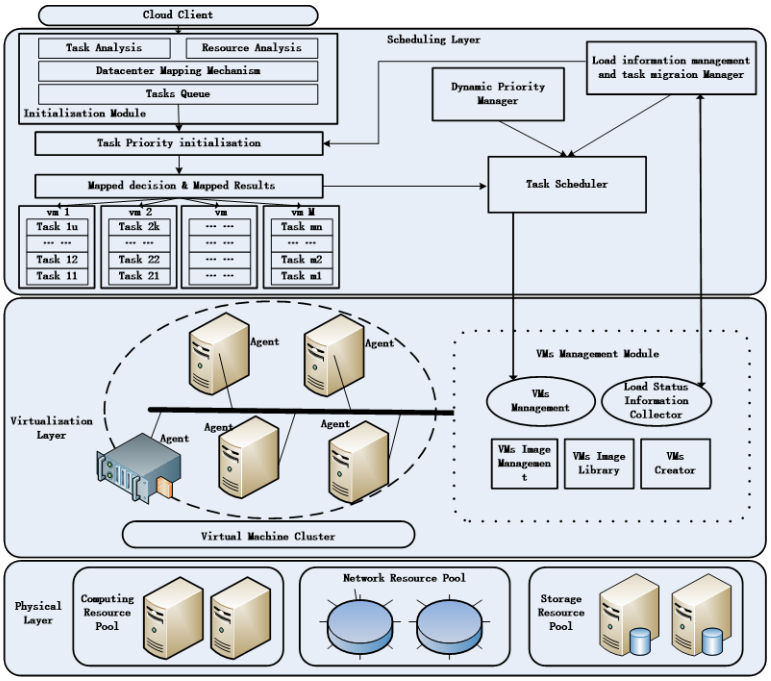

Fig. 1. The architecture of task scheduling model in cloud

Fig. 1 shows the task scheduling cloud architecture, it consists of physical layer, visualization layer, task scheduling layer and cloud client interface. The physical layer is comprised of massive different hardware resources, the same type of hardware resources constitute a resource pool such as computing resource pool, network resource pool and storage resource pool which providing basic facility services for virtualization layer. The virtualization layer is comprised of VMs cluster and VMs management modules. The virtual machine cluster includes massive VMs and VM agents, VM agent is an execution program which is deployed on every virtual machine to acquire the load status information of VMs periodically. VMs Management modules include image library component, image management component, VMs management component, VMs creator component and VMs load status information collector component, they function as follow **Table 1**.

Table 1. The component function of VMs management modules

Components	function
VMs Image Library	Storing and managing VMs image
VMs Image Management	
VMs Management	Creating VM instances and managing their whole lifecycle
VMs Creator	
Load Information Management	Collecting and analyzing the information submitted by monitoring agents

The task scheduling layer consists of initialization module, distribution results, dynamic priority manager, load information management and task migration manager and task scheduler.[21] The initialization module is responsible for sending task to the task queue in data center which can satisfy the task QoS requirements such as the memory requirement, capability requirement etc and has the local data that calculation required. The component of task priority initialization receives tasks from scheduling window in task queue, and calculates every task priority, and sorted the tasks by priority. The component of mapped decision and mapped results maps the tasks to the appropriate VM by decision variable, then records the mapped results between task and VM, and marks which task has been completed. The dynamic priority manager takes charge of sorting the tasks in VM's task queue by tasks priority. The task scheduler is responsible for sending the task to the virtual machine for execution. Load information management and task migration manager is responsible for checking the load information changes on the VMs, and motivating the task migration. The cloud client is the place where consumers submit their tasks and get the computing results.

3.2. management strategies

3.2.1. Task Queue management strategy

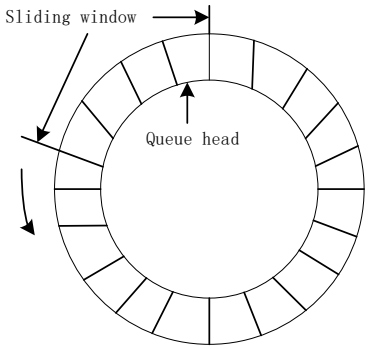

Fig. 2. The task queue management strategy diagram

In this paper, we organize tasks queue in initialization module into a circle, and set a sliding window on it, what looks like as **Fig. 2**. The tasks in sliding window will be scheduled immediately, the number of tasks in sliding window is set by the threshold. When the tasks scheduling in the sliding window is finished, the sliding window will slide down the task queue, and the queue head and queue rear will change correspondingly, the queue head position is on the sliding window head, the queue rear position is adjacent to the queue head but don't in the sliding window. Therefore, the increasing tasks submitted by the consumers will be scheduled rapidly through this task queue management strategy.

3.2.2. Dynamic priority manager strategy

Because of the priority tasks included in the time urgency factor, so the priority is growing along with the growth of time, task dynamic priority manager is responsible for task priority maintenance, specific maintenance strategies are as follows:

(1) Sorting tasks in VM's task queue by task priority.

(2) Maintaining task priority, when the task remaining time is zero, the task is recalculated value halved task priority, it will be marked and insert the ordered task queue task.

3.2.3. Load information management and task migration strategy

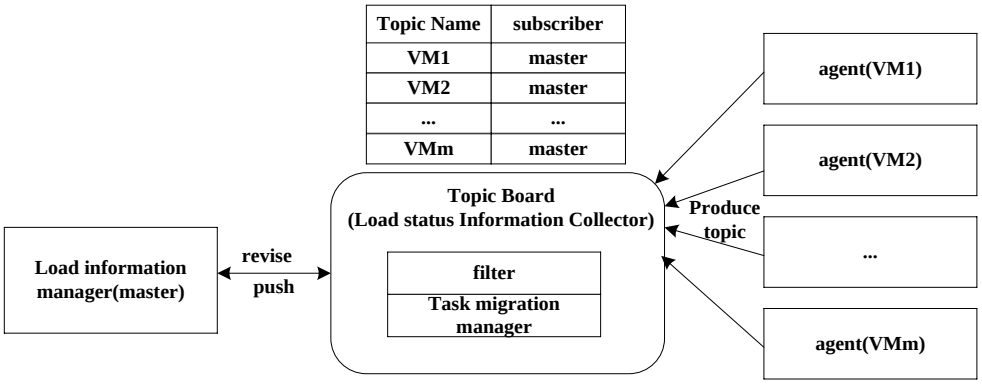

Fig. 3. VMs' load information management strategy diagram

The accuracy of load information that collected from VMs directly determines the quality of scheduling algorithm, while in the actual environment, it will meet some emergencies such as task should be re-executed because of program error occurs, VM server goes down etc.[23] that could lead to the VMs' load information change, therefore need a good load information collection mechanism to ensure the load information. In this paper, we use the distributed publish-subscribe messaging mechanism to achieve accurate load information from VMs. The block diagram of load information management strategy looks as **Fig. 3**.

The agents on VMs monitor the VMs' status and transmit topic messages which containing load status information flow to the topic board (namely the load status information collector) periodically.[21] The topic board is responsible for topic collecting and check the load information change, if the load reduction does not matches the corresponding VM's process performance in the period, it means some emergency happens on that VM, the corresponding topic message will be pushed to the subscriber who subscribe this topic. To avoid the master node cost too much source in management, the topic board is set on one of the VMs. The load information management and task migration manager in scheduling layer just need subscribe the topics on the topic board, and then check the message receiving queue periodically, if receive the topic message then update the load status information on the VM which matching the topic name.

After this patch of tasks are scheduled over, or some events that cause tasks to be recomputed occur on VMs and lead to the VM over the threshold (namely the VM is marked as over load VM), the task migration manager will start work, it will choose the tasks from the rear of task queue, and find the light load VM which can satisfy the requirement of task, and if the task is scheduled to the light load VM or exchanged with the task in the light VM can reduce the load and do not augment the Makespan, then schedules it to the light load VM. Its diagram shows as **Fig. 4**.

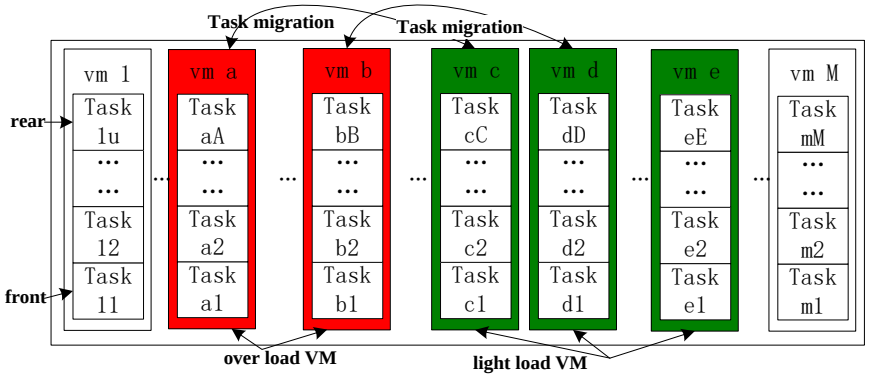

Fig. 4. Task migration strategy diagram

3.3. DPLB algorithm description

After submitting the task to the data center which can satisfy the task' QoS requirement and conform data locality, the task arrives at the task queue,[20] the DPLB algorithm will work as the following steps:

We first calculate the priority of all the tasks in the sliding window, prioritize the tasks, and sort the virtual machine resources by capacity. Then schedule the tasks in sliding window in order of priority, for every task, calculate its ECT on every VMs, and according to task's memory and capability(namely deadline) requirements $t^i_{memory} \le vm^j_{memory}$ and $ECT_{ij} \le t^i_{deadline}$, find task' available VMs group. Then check whether receive new load information topic messages which are pushed from topic board, updating the load information if have received and calculate the mapped decision variable ρ_{ij} between task and its available VMs. Then schedule the task to the VM which get the maximum value of the mapped decision variable ρ_{ij}, and update this VM's starting execution time be_j and load intensity LI_j. As above operation, schedule next task in sorted sliding window, until completed the last task scheduling, empty the sliding window and slide down the window, start a new round of tasks scheduling as above. A round of the algorithm process is as follows:

Algorithm DPLB algorithm

1: **Notations:** Let CT_i denote the available VMs group of t_i,
2: **Procedure:Init()**
3: **for**(int j=0;j<m;j++)
4: $be_j = 0$, $LI_j = 1$; // Initialization start execution time, LI
5: subscribe topics & set sliding window size;
6: **Procedure:DP**(t_i) // compute tasks' dynamic priority
7: **for** t_i in T **do**
8: set $DP(t_i)$ based on Eq.(9);
9: **end for**
10:**Procedure: Sort(T,VM)** // sort tasks and VMs
11:Sort T by $DP(t_i)$ in descending order;

12:Sort VM by $vm^{j}_{capacity}$ in descending order;
13:**Procedure: GetETC()** // calculate ETC_{ij}
14:**for** t_i in sorted T **do**
15: **for** vm_j in sorted VM **do**
16: set ETC_{ij} based on Eq.(1);
18: **end for**
19:**end for**
20:**Procedure: Schedule()**
21:check topic messages & update load information;
22:**for** t_i in sorted T
23: **if**($t^{i}_{left} \le 0$)
24: compute $DP(t_i)$ with half t^{i}_{fee} and insert t_i to sorted T ;
25: **if** (position of t_i in T changed) **then**
26: continue;
27:**for** vm_j in sorted VM // calculate ECT_{ij} and CT_i
28: set ECT_{ij} based on Eq.(2);
29: **if** （ $ECT_{ij} \le t^{i}_{deadline}$ && $t^{i}_{memory} \le vm^{j}_{memory}$ ）
30: $CT_i \leftarrow vm_j$;
31:**for** vm_j in CT_i // Calculate ρ_{ij} , Start scheduling
32: set ρ_{ij} based on Eq.(12);
33: assign t_i to the vm in CT_i with the $\max\{\rho_{ij}\}$;
34: update be_j and LI_j ;
35:**Procedure: migration()** // task migration
36:Find overload VMs and light load VMs;
37:**for** vm_j in overload VMs
38: **for** t_i in the vm_j' rear of task queue
39: **for** vm_k in light load VMs
40: **if**($(time^{jk}_{migration} + ECT_{ik}) <= Makespan$ && $lbd^{i}_{k} < lbd^{i}_{j}$)
41: $t_i --> vm_k$;
42: **for** t_u in the vm_k' rear of task queue
43: **if**($t^{u}_{size} < t^{i}_{size}$ && $(time^{jk}_{migration} + ECT_{ik}) <= Makespan$
44: && $lbd^{i}_{k} < lbd^{i}_{j}$ && $lbd^{u}_{j} < lbd^{u}_{k}$)
45: $t_i --> vm_k$, $t_u --> vm_j$;
46:**End**

4 Experimental Analysis

4.1. Experimental environment

To validate the effectiveness of DPLB proposed in this paper, we made the simulation experiment based on the CloudSim[17] platform. We schedule 50 − 450 independent tasks to 20 VMs on CloudSim, each virtual machine possesses one CPU, the processing capability of CPU in the range of 500 − 1000 (MIPS) , the memory capability in the range of 1000 − 2000 (MB) , the matching method between virtual machine and physical host is supported by Time-Shared algorithm. The size of tasks in the range of 10000 − 50000 (MI) , the file size of tasks in the range of 30 − 50MB, the memory requirement of tasks in the range of 800 − 1800 (MB) , the deadline requirement of tasks in the range of 50 − 1200 (s) ,the bandwidth between VMs in the range of 8 − 10, the priority weighting factor in Eq. (9) is set as $\omega_1 = \omega_2 = 0.5$ .The CloudSim platform running on a personal computer with CPU of AMD X2215 2.7GHz and memory of 2GB.

4.2. Assessment Indicators

In order to evaluate the performance of DPLB, four assessment indicators are defined in the experiments as follows:

Makespan: the completion time of all tasks

It is defined as Eq.(3), its value more smaller, the performance more better.

Load balancing degree: measure the degree of VMs cluster about load balancing

We can know the definition of EPT from Eq.(11), then the average expect processing time of all tasks on VMs can be denoted as $\overline{EPT} = \frac{1}{m}\sum_{j=1}^{m} EPT_j$. In this paper, we use the standard deviation of expected processing time to express the load balancing degree, then it can be denoted as Eq.(12), its value more smaller, the performance more better.

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(EPT_j - \overline{EPT})^2} \tag{12}$$

Success completion ratio: reflect the number of timely completion tasks

Let n_s is the number of tasks which are completed before the deadline, n is the number of all tasks, then the success ratio can be denoted as Eq.(13), its value more bigger, the performance more better.

$$SC = n_s / n \tag{13}$$

Service profit per second: evaluate the profit per second

We assume that the number of tasks which are completed before the deadline is k, and half the profit of tasks which are not timely completion, then the service profit per second can be denoted as Eq.(14), its value more bigger, the performance more better.

$$SP = (\sum_{i=1}^{k} t_{fee}^{i} + \frac{1}{2}\sum_{i=k}^{n} t_{fee}^{i}) / Makespan \tag{14}$$

4.3. results and analysis

Under the same experimental condition and environment, we implement the Min-Min algorithm, Max-Min algorithm, WRR algorithm, GAs algorithm, HBB-LB algorithm and DPLB algorithm in the CloudSim platform, and compare their performance in four indicators which are defined in section 4.2. In this paper, we design two type tasks sets to validate the performance of DPLB, there are smooth tasks sets which task can be completed one-time and recompleted tasks sets which some of the tasks need to be recomputed.

Part I Experiments with smooth tasks set, and Comparing Min-Min algorithm, Max-Min algorithm, WRR[18] algorithm, GAs[12] algorithm, HBB-LB[11] algorithm and DPLB algorithm.

Experiment 1 Makespan Comparison

Fig. 5 shows the comparison result of six algorithms on Makespan with different number of tasks, the X-axis represents number of tasks and the Y-axis represents the Makespan. From the **Fig. 5**, we can see that DPLB algorithm keeps a lower value, it is more fit to a large number of tasks, it becomes the lowest since 150 tasks, it is obvious better than the Min-Min algorithm and WRR algorithm which are prone to load imbalance, and it also better than Max-Min, GAs algorithm and HBB-LB algorithm, it proves that the DPLB algorithm can effectively reduce the overall task completion time.

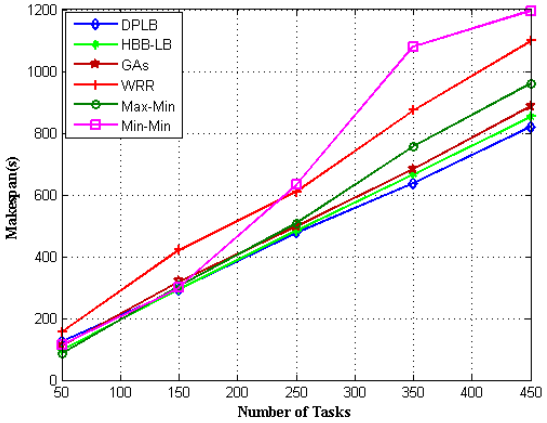

Fig. 6. Comparison of Makespan

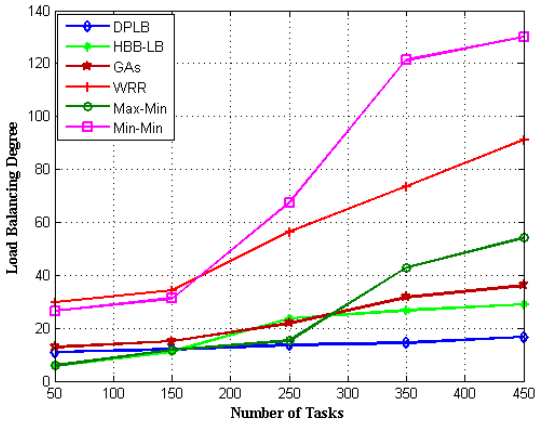

Fig. 6. Comparison of load balancing degree

Experiment 2 Load balancing degree comparison

Fig. 6 shows the comparison of load balancing degree, the X-axis represents number of tasks and the Y-axis represents the load balancing degree. From the **Fig. 6**, we can see that the DPLB algorithm keeps a lower value, it becomes the lowest since 150 tasks, it proves that the DPLB algorithm has a good load balancing characteristics. From the experiment, we also get the comparing of migration tasks' number as **Table 2**, and load balancing degree change of DPLB algorithm before and after running tasks immigration as **Table 3**, from them we can see that the migration tasks' number in DPLB algorithm is the fewest than others, it reduces the migration time cost and proves that the design of mapped decision variable which consider the load intensity can effectively balance VMs' load.

Table 2. the comparison of number of tasks migrated

No. of tasks	GAs	HBB-LB	DPLB
50	1	0	0
150	5	3	0
250	13	8	2
350	21	15	4
450	30	24	8

Table 3. The load balancing degree change of DPLB

No. of tasks	Before migration	After migration
50	10.94	10.94
150	12.06	12.06
250	17.28	13.56
350	22.33	14.46
450	28.25	16.70

Experiment 3 Success completion ratio comparison

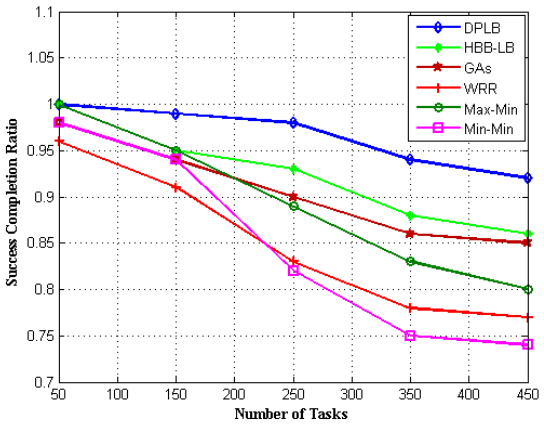

Fig. 7. Comparison of success completion ratio

Fig. 7 shows the comparison of success completion ratio, the X-axis represents number of tasks and the Y-axis represents the success completion ratio. From the **Fig. 7**, we can see that with the increasing number of tasks, the success completion ratio shows a downward trend, and DPLB algorithm keeps the highest success completion ratio all the time, namely the DPLB algorithm can ensure more tasks to be completed before deadline, it proves that the design of task's dynamic priority which considers the urgency of task execution has a good influence on DPLB algorithm, it insures that the task which has the shorter deadline can be first scheduled and completed.

Experiment 4 Service profit per second

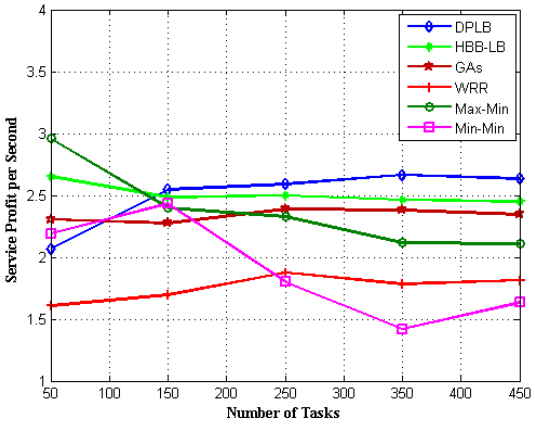

Fig. 8. Comparison of service profit per second

Fig. 8 shows the comparison of service profit per second, the X-axis represents number of tasks and the Y-axis represents the service profit per second. From the **Fig. 8** we can see that with the increasing number of tasks, the DPLB algorithm keeps the highest value since 150 tasks, namely the DPLB algorithm can bring more benefits per second.

Part II Experiments with recomputed task sets, and comparing dynamic balancing algorithm which includes GAs algorithm, HBB-LB algorithm, DPLB algorithm.

As we know, some error will happen in the actual cloud environment at times, it will cause to tasks need to be recomputed on the VM, and even lead to load imbalance. The dynamic load balancing algorithm is the key to solve this problem in actual cloud environment. In order to verify the dynamic load balancing performance of DPLB algorithm, we design a new task sets based on Part I 450 task set. In this new task sets, different number of tasks are marked as needed to be recomputed, and they are random distributed in the task set, and we assume that the recomputed task are checked to be recomputed in the end of its processing.

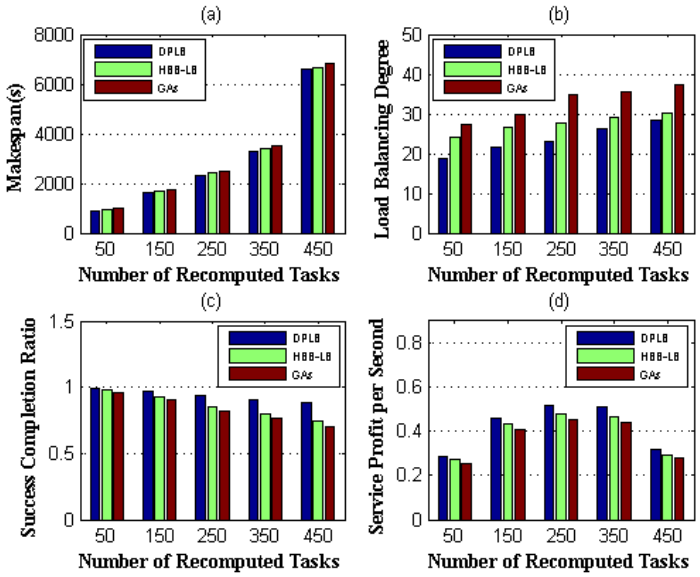

Fig. 9. Comparison of four indicators in recomputed task sets. (a) Comparison of Makespan; (b) Comparison of load balancing degree; (c) Comparison of success completion ratio; (d) Comparison of service profit per second.

Fig. 9 shows the comparison of four indicators in 450 task sets which with different number of recomputed tasks, the X-axis represents number of recomputed tasks in 450 task set and the Y-axis represents the Makespan, load balancing degree, success completion ratio and service profit per second in turn. From the **Fig. 9**, we can see that with the increasing number of tasks, the DPLB algorithm keeps the lowest value in Makespan and load balancing degree, keeps the highest value in success completion ratio and service profit per second, it proves that the DPLB algorithm also has a better performance in dynamic environment, it can effectively reduce the overall task completion time (Makespan) and load balancing degree, and effectively advance the success completion ratio and service profit per second.

From this experiment, we also get the get the comparing of migration tasks' number as **Table 4**, and load balancing degree change of DPLB algorithm before and after running tasks migration as **Table 5**, comparing with **Table 2** and **Table 3**, we can see that the recomputed tasks augment the migration tasks' number and the load balancing degree before load migration, and the DPLB algorithm also get a lower migration task number, it reduces the migration time cost.

Table 4. The comparison of number of tasks migrated

No. of tasks recomputed	GAs	HBB-LB	DPLB
50	3	2	2
150	8	5	4
250	16	12	6
350	24	19	12
450	38	30	18

Table 5. The load balancing degree change of DPLB

No. of tasks recomputed	Before migration	After migration
50	26.32	18.79
150	31.65	21.51
250	38.44	23.12
350	41.97	26.20
450	42.06	28.41

5. Conclusion

In this paper, we propose a task scheduling algorithm based on dynamic priority and load balancing in cloud computing environment (DPLB). The experimental result shows that the DPLB algorithm can effectively reduce the total task completion time and balancing the VMs' load, and it also can archive a better success completion ratio and service profit per second in different task sets. In future, we plan to improve this algorithm by considering other QoS factors such as security requirement and more complex tasks such as DAG task.

Acknowledgements

The work described in this paper is supported by National Natural Science Foundation of China （31770768, the Natural Science Foundation of Heilongjiang Province of China (F2017001), the Fundamental Research Funds for the Central Universities (2572017CB32) and China Forestry Nonprofit Industry Research Project （201504307）.

References

[1] *Com* Yunchuan Sun, Antonio Jara, "An extensible and active semantic model of information organizing for the Internet of Things," *Personal and Ubiquitous Computing, Volume 18, Issue 8*, 1821-1833, 2014. Article (CrossRef Link)

[2] Mishra A, Jain R, Durresi A, "Cloud computing: networking and communication challenges," *Cloud computing: networking and communication challenges, 50(9)*, 24-25, 2012. Article (CrossRef Link)

[3] J. Zhu, Y. Song, D. Jiang and H. Song, "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1-1, 2017. Article (CrossRef Link)

[4] Gu L, Tang Z, Xie G, "The Implementation of MapReduce Scheduling Algorithm Based on Priority," *Parallel putational Fluid Dynamics. Springer Berlin Heidelberg*, 100-111, 2014. Article (CrossRef Link)

[5] Liu G, Li J, Xu J, "An Improved Min-Min Algorithm in Cloud Computing," in *Proc. of Proceedings of the 2012 International Conference of Modern Computer Science and Applications. Springer Berlin Heidelberg, 53(4)*, 47-52, 2013. Article (CrossRef Link)

[6] Li Q, Ba W, "A group priority earliest deadline first scheduling algorithm," *Frontiers of Computer Science, 6(5)*,560-567, 2012. Article (CrossRef Link)

[7] Xia J L, Chen H, Yang B, "A real-time tasks scheduling algorithm based on dynamic priority," *Jisuanji Xuebao(Chinese Journal of Computers), 34(12)*, 2685-2695, 2012. Article (CrossRef Link)

[8] W. Wei, X. Fan, H. Song, X. Fan and J. Yang, "Imperfect Information Dynamic Stackelberg Game Based Resource Allocation Using Hidden Markov for Cloud Computing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1-1. Article (CrossRef Link)

[9] Ren X, Lin R, Zou H, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast," in *Proc. of Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on. IEEE*, 220-224, 2011. Article (CrossRef Link)

[10] Hu J, Gu J, Sun G, et al, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on. IEEE*, 89-96, 2010. Article (CrossRef Link)

[11] Venkata Krishna P, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, *13(5)*, 2292-2303, 2013. Article (CrossRef Link)

[12] Zhang W, Tan S, Lu Q, et al. "A Genetic-Algorithm-Based Approach for Task Migration in Pervasive Clouds," *International Journal of Distributed Sensor Networks*, 2015. Article (CrossRef Link)

[13] Ramezani F, Lu J, Hussain F K. "Task-based system load balancing in cloud computing using particle swarm optimization," *International Journal of Parallel Programming*, *42(5)*, 739-754 2014. Article (CrossRef Link)

[14] Li K, Xu G, Zhao G, et al. "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. of Chinagrid Conference (ChinaGrid), 2011 Sixth Annual. IEEE*, 3-9, 2011. Article (CrossRef Link)

[15] H. Zhang, D. Jiang, F. Li, K. Liu, H. Song and H. Dai, "Cluster-Based Resource Allocation for Spectrum-Sharing Femtocell Networks," *IEEE Access*, vol. 4, pp. 8643-8656, 2016. Article (CrossRef Link)

[16] Chen H, Wang F, Helian N, et al. "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in *Proc. of Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on. IEEE*, 1-8, 2013. Article (CrossRef Link)

[17] Calheiros R N, Ranjan R, Beloglazov A, et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, *41(1)*, 23-50, 2011. Article (CrossRef Link)

[18] Boloor K, Chirkova R, Viniotis Y, et al. "Dynamic request allocation and scheduling for context aware applications subject to a percentile response time SLA in a distributed cloud," in *Proc. of Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE*, 464-472, 2010. Article (CrossRef Link)

[19] Yunchuan Sun, Houbing Song, Antonio J. Jara, Rongfang Bie, "Internet of Things and Big Data Analytics for Smart and Connected Communities," *IEEE Access,Volume:4*, 766-773, 2016. Article (CrossRef Link)

[20] T. Qiu, K. Zheng, H. Song, M. Han and B. Kantarci, "A Local-Optimization Emergency Scheduling Scheme With Self-Recovery for a Smart Grid," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3195-3205, Dec. 2017. Article (CrossRef Link)

[21] Armbrust M, Fox A, Griffith R, et al. "A view of cloud computing," *Communications of the ACM*, *53(4)*, 50-58, 2010. Article (CrossRef Link)

[22] Jiguo Yu, Wenchao Li, Xiuzhen Cheng, Mohammed Atiquzzaman, Hua Wang, Li Feng. "Connected dominating set construction in cognitive radio networks," *Personal and Ubiquitous Computing*,*20(5)*,757-769, 2016. Article (CrossRef Link)

[23] Li Feng, Jiguo Yu, Xiuzhen Cheng, Mohammed Atiquzzaman. "A novel contention-on-demand design for WiFi hotspots," *Personal and Ubiquitous Computing*, *20(5)*, 705-716, 2016. Article (CrossRef Link)

[24] Yunchuan Sun, Hongli Yan, Cheng Lu, Rongfang Bie, Zhangbing Zhou. "Constructing the web of events from raw data in the Web of Things," *Mobile Information Systems. Volume 10, No. 1*, 105-125, 2014. Article (CrossRef Link)

[25] Ghanbari S, Othman M. "A priority based job scheduling algorithm in cloud computing," *Procedia Engineering*, *50*, 778-785, 2012. Article (CrossRef Link)

[26] Mao Y, Chen X, Li X. "Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing," in *Proc. of Proceedings of International Conference on Computer Science and Information Technology. Springer India*, 457-465, 2014. Article (CrossRef Link)

WEIPENG JING (M'17) received the Ph.D. degree from the Harbin Institute of Technology of China. He is currently an Associate Professor with Northeast Forestry University, China. His research interests include modeling and scheduling for distributed computing systems, fault tolerant computing and system reliability, cloud computing, and spatial data mining. He has published over 50 research articles in refereed journals and conference proceedings, such as CPC, PUC, and FGCS.

QIUCHENG MIAO received the B.S. degree in electrical and information engineering from Shihezi University, Shihezi, China, in 2016. He is currently pursuing the master's degree with Northeast Forestry University. His current research interests include edge computing, differential privacy and named data networking. He is the member of the ACM.

GUANGSHENG CHEN, He is currently an Professor with Northeast Forestry University, China. His research interests include Forestry High Performance Computing, Forestry Big Data Applications. He has published over 50 research articles in refereed journals and conference proceedings, such as Advanced Materials Research, International Workshop on Database Applications, GIS: Proceedings of ACM International Symposium on Advances in Geographic Information Systems.