information

MDPI

Article

Task Staggering Peak Scheduling Policy for Cloud Mixed Workloads

Zhigang Hu, Yong Tao, Meiguang Zheng * and Chenglong Chang

School of Software, Central South University, Changsha 410075, China; zghu@csu.edu.cn (Z.H.); yongtao@csu.edu.cn (Y.T.); changchenglong@csu.edu.cn (C.C.)

* Correspondence: zhengmeiguang@csu.edu.cn

Received: 7 November 2018; Accepted: 16 December 2018; Published: 18 December 2018

Abstract: To address the issue of cloud mixed workloads scheduling which might lead to system load imbalance and efficiency degradation in cloud computing, a novel cloud task staggering peak scheduling policy based on the task types and the resource load status is proposed. First, based on different task characteristics, the task sequences submitted by the user are divided into queues of different types by the fuzzy clustering algorithm. Second, the Performance Counters (PMC) mechanism is introduced to dynamically monitor the load status of resource nodes and respectively sort the resources by the metrics of Central Processing Unit (CPU), memory, and input/output (I/O) load size, so as to reduce the candidate resources. Finally, the task sequences of specific type are scheduled for the corresponding light loaded resources, and the resources usage peak is staggered to achieve load balancing. The experimental results show that the proposed policy can balance loads and improve the system efficiency effectively and reduce the resource usage cost when the system is in the presence of mixed workloads.

Keywords: cloud computing; mixed workloads; task scheduling; load balancing; performance counters

1. Introduction

Cloud computing [1–3] is an emerging business application providing on-demand access, ubiquitous access characteristics, and elastically scalable information technology (IT) resource usage [4]. A variety of applications [5,6] have been deployed in cloud computing platforms, some application tasks have higher CPU requirements, and some require higher storage, while others require frequent I/O access or larger bandwidth resources. Different cloud tasks have great differences in resource requirements due to their own constraints, diversity, and complexity [7].

Recently, task scheduling for mixed workloads has attracted plenty of attention [8,9]. Mixed workloads typically include compute-intensive, memory-intensive and data-intensive tasks. Most of the compute-intensive tasks are used for CPU operations such as computation or logic judgment. Memory-intensive tasks take up a large number of storage resources. The main execution features of data-intensive tasks are large amounts of data transmission, processing, and I/O access. For the task sequences of massive mixed workloads submitted by user, it is Non-deterministic Polynomial Complete (NP-C) to reasonably allocate cloud resources and ensure the load balancing and performance efficiency of the whole system. The relevant research results of task scheduling have showed that: (1) Different types of tasks (compute-intensive, memory-intensive and data-intensive) have significant differences in resource requirements [10,11]. (2) Heterogeneity of node resources and dynamic nature of load. When scheduling tasks of different types, the load size of resources such as CPU, memory, and I/O on the node resources at the current time needs to be considered [12,13]. Unreasonable task scheduling policy might lead to a load imbalance of resources, which not only

seriously affects node resource utilization and performance, but also increases task completion time and cost.

In this work, we propose a Staggering Peak Scheduling by Load-aware Model (SPSLAM) when researching the task scheduling problem of cloud mixed workloads. The SPSLAM mainly considers the task characteristics and resource load status, which aims to balance load, improve efficiency, and optimize resource usage cost. The main contributions of our work are as follows:

- The task sequences of mixed workloads are divided into multiple types of queues by the fuzzy clustering algorithm based on different task characteristics.
- We introduce the Performance Counters (PMC) mechanism to dynamically monitor the node resource load status and respectively sort the resources by CPU, memory, and I/O load size.
- We develop a policy SPSLAM from the perspective of task types and resource load for mixed workloads and by doing so, the system can realize load balancing and improve efficiency.

The rest of the paper is organized as follows. In Section 2, the related work is discussed. In Section 3, we introduce the system model and give the definition of related concepts. Section 4 presents the staggering peak scheduling policy. Experiment results and performance analysis are presented in Section 5. Section 6 concludes the paper.

2. Related Work

To address the issue of scheduling mixed workloads in cloud computing, Khorandi et al. [14] presented a high-performance technique for virtualization-unaware scheduling of compute-intensive synchronized tasks in virtualized high performance computing systems. Virtual machines (VMs) were in turn assigned/reassigned to clustered physical machines based on CPU load. In this way, it can minimize the performance and scalability degradation of high-performance computing applications. However, its performance was extremely unstable due to the large dynamic nature of the "intermediate data" generated during executing I/O-intensive tasks. In some test cases, there may even be a 30% reduction in system energy efficiency. To optimize system performance, Wang et al. [15] found that computation-intensive tasks and bandwidth-intensive tasks together created a severe bottleneck in cloud-based distributed interactive applications. They presented an interference-aware solution to smartly allocate workloads and dynamically assign capacities across VMs based on their arrival/departure patterns. However, the scalability of the solution required further verification when it came to the interference between the computation-intensive and bandwidth-intensive tasks. It can be seen that the task characteristics and resource load status have an important impact on the scheduling effect.

For the heterogeneity of node resources and the dynamic nature of load in cloud computing, most of the research has focused on resources pre-process and static analysis of resources load status. Resources were clustered based on cloud computing resources set on the service level of user, which made tasks with different preferences to be selected in different clusters to reduce the task scheduling range [11]. However, the method assumed that the computing and communication capabilities of resources were stable for a certain period of time when the resources were clustered, and the changes in resource computing and communication capabilities were not considered. After a period of time, it was easy to cause load imbalance and efficiency reduced. To realize flexible management of multi-resource cloud task scheduling performance and energy, Mao et al. [16] proposed a multi-resource cloud task scheduling algorithm for Energy Performance Trade-Offs. The energy and performance of a cloud system can be flexibly managed and controlled by setting a parameter to reconfigure the weight of performance and energy consumption. But the factors influencing the performance and the energy consumption of VM, host, and data center were diversified. The way to realize better trade-offs was not only about tuning the static parameter but also designing an algorithm that dynamically changed the parameter based on the current workload situation. Therefore, for cloud task scheduling issues of mixed workloads, it is necessary to consider the dynamic nature of resource

load to select the appropriate scheduling policy after analyzing the task characteristics. This requires further research in this paper.

The load balancing technology [17] in cloud computing has always been a research focus. Chen et al. [18] measured the load status by setting the number of task request connections for the node resources and assigned the task to the node with the smallest number of connections. However, this method did not reflect the real load of the node resources. When the load request increased rapidly, the scheduling efficiency decreased instantly and the load was easily imbalanced. In cloud computing, Zhao et al. [19] used the Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications (SWITCH) project to address the urgent industrial need for developing and executing time critical applications. Mobile edge computing (MEC) [20] was to push mobile computing, network control, and storage to the network edges (e.g., base stations and access points) so as to enable computation-intensive and latency-critical applications at the resource-limited mobile devices. To satisfy the quality of offloading service, Zhang et al. [21] proposed a hierarchical cloud-based Vehicular Edge Computing (VEC) offloading framework, where a backup computing server in the neighborhood was introduced to make up for the deficit computing resources of MEC servers. Offloading computation to a mobile cloud was a promising approach for enabling the use of computationally intensive applications by mobile devices. To address the allocation of cloud and wireless resources among mobile devices, Josilo et al. [22] provided a polynomial time algorithm to optimize the equilibrium cost and provide good system performance.

Based on the existing study, our work focuses on the diversity of tasks, the constraints of tasks themselves, and the relationship with the requirement of resources. When analyzing the resource load status, static performance indicators such as CPU computing ability, memory, and bandwidth are mostly used. However, in the dynamic environment such as cloud computing, these indicators are uncertain and atypical, and it is necessary to dynamically monitor the resources load status during task scheduling. This paper proposes the SPSLAM that mainly considers task characteristics and resources load status. The goal is to achieve load balancing, improve efficiency, and optimize resource usage cost.

3. Model Description

System Framework Model

The system framework of SPSLAM proposed in this paper is shown in Figure 1. It includes three parts: task sequences classification, resources load sorting, and scheduling scheme generation. For the task sequences submitted by the user, firstly, based on the characteristics of each task and the different requirements of resources, the Fuzzy C-Means (FCM) clustering algorithm is used to divide the task sequences of the mixed workloads into multiple types of queues. Secondly, the classification and sorting of resources load is achieved by introducing the PMC mechanism. Finally, the task execution queues and resource sorting information are taken as input, and the final scheduling scheme is generated based on SPSLAM.

The applications submitted by the user in this paper can be represented as the task sequences $V = \{V_{ij} | V_{ij} = \langle C_{ij}, M_{ij}, O_{ij} \rangle\}$, where V_{ij} represents the jth task of the ith sequence, C_{ij} represents the computational amount of V_{ij} (unit: $MIPS$, the number of instructions), M_{ij} is the memory resource required by V_{ij}, and O_{ij} is the I/O resource required by V_{ij}.

The node resources in cloud computing can be defined as mainly composed of CPU, memory, and I/O resource. These resources are noted as $R = \{R_1, R_2, \ldots, R_m\}$, where $R_j = \langle C_j, M_j, O_j \rangle$. Three parameters of the resources respectively represent CPU utilization, available memory size, and I/O latency wait time. The data transfer rate between the two node resources is noted as the matrix B, where the matrix element $b_{i,j}$ represents the data transfer rate between R_i and R_j.

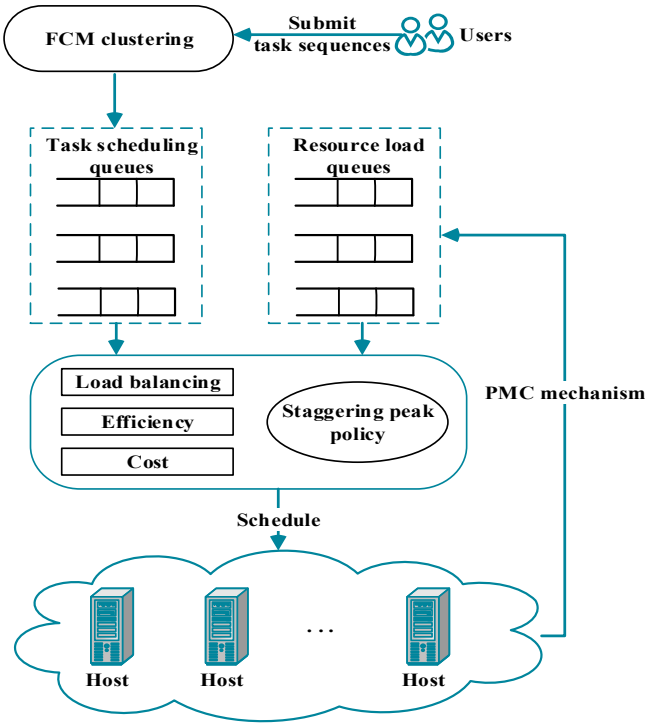


Figure 1. System framework model.

4. Staggering Peak Scheduling Policy

The SPSLAM proposed in this paper consists of three parts: task sequences classification, resources load sorting, and scheduling scheme generation.

4.1. Task Sequences Classification

In cloud computing, the requirements of tasks are difficult to describe accurately, and the dynamic nature of cloud resources makes it impossible to correctly describe the load status in a specific time period. Therefore, the matching scheduling of tasks and resources also has some ambiguity.

Due to the different characteristics of tasks, the demand for resources is different. For example, compute-intensive tasks require strong CPU computing ability, memory-intensive tasks require large amounts of memory, and data-intensive tasks require frequent I/O access and bandwidth resources. In this paper, the FCM algorithm is used to divide the task sequences into three queues: CPU-intensive, memory-intensive, and I/O-intensive based on the diversity requirements of tasks. The FCM algorithm is an improved scheme of K-means by applying fuzzy theory. It belongs to the soft clustering algorithm and promotes the hard membership to soft membership, which makes the iteration of the central point easier to achieve global optimization. At the same time, the FCM algorithm is superior to the traditional hard clustering algorithm in clustering time and accuracy. The specific clustering process based on FCM algorithm is described as follows:

Step 1. Form an initialization sample matrix $T_{n\times 3}$ based on attribute values $V_{ij} = \langle C_{ij}, M_{ij}, O_{ij} \rangle$ of n tasks, where T_{ij} represents the jth dimension attribute of the ith task. Since the three parameter units of V_{ij} are not on the same scale, the attribute values need to be respectively standardized and normalized when classifying the task sequences. We standardize T_{ij} to get T'_{ij} by Equation (1),

$$T'_{ij} = \frac{T_{ij} - \overline{T_j}}{S_j} \tag{1}$$

where $\overline{T_j} = \frac{1}{n}\sum_{j=1}^{n} T_{ij}$, and S_j is the standard deviation of V_{ij} in the jth dimension attribute. Then we normalize T'_{ij} to compress the data between $[0, 1]$ by Equation (2),

$$T''_{ij} = \frac{T'_{ij} - T'_{jmin}}{T'_{jmax} - T'_{jmin}} \tag{2}$$

where T'_{jmax} and T'_{jmin} represent the maximum and minimum values in $\left\{T'_{1j}, T'_{2j}, \ldots, T'_{nj}\right\}$, respectively.

Step 2. Set the number of task clusters to 3 and initialize the membership matrix (U) with a random number whose value is between $[0, 1]$. The initialization condition can be formally presented as Equation (3), where u_{ij} indicates the degree to which V_{ij} belongs to the fuzzy group (c_j). The larger the u_{ij} is, the higher the probability the V_{ij} belongs to c_j.

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, 2, \ldots, n \tag{3}$$

Step 3. In order to ensure the minimum dissimilarity between the cloud task and its cluster center in each cluster, we use the weighted exponential product of u_{ij} and each task to the Euclidean distance (d_{ij}) of each class center as the optimization objective function. The function is defined as

$$J(U, c_1, c_2, \ldots, c_n) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} \tag{4}$$

where $d_{ij} = \|V_{ij} - c_j\|$, and $m = 2$ which is a weighted index in this paper. To get the optimal extreme value of the clustering function, the lattice multiplier method can be used to derive the iterative formula of c_k and U.

$$c_k = \frac{\sum_{i=1}^{n} u_{ki}^{m} V_{ii}}{\sum_{i=1}^{n} u_{ki}^{m}}, k = \{1, 2, 3\} \tag{5}$$

$$u_{ki} = \frac{1}{\sum_{p=1}^{c} \left(\frac{d_{ki}}{d_{pi}}\right)^{2/(m-1)}} \tag{6}$$

Step 4. Calculate c_k and $J(U, c_1, c_2, \ldots, c_n)$ by Equations (4) and (5) respectively. If $J < \varepsilon$ (ε is the threshold), it indicates that the cluster ends. Otherwise, update U by Equation (6) and return to step 3.

In this paper, the algorithm of task sequences classification based on FCM is listed in Algorithm 1.

Algorithm 1: The algorithm of task sequences classification

Input: $c = 3, task_{list}, \varepsilon$
Output: $Q_{VCPU}, Q_{VMem}, Q_{VIO}$
1: Initialize the sample matrix $T_{n\times 3}$ through the task attribute values C_{ii}, M_{ii} *and* O_{ii};
2: **for** $T_{ij} \epsilon\ T_{n\times 3}$ **do**
3: Standardize and normalize T_{ij} to get T'_{ij};
4: **End for**
5: Initialize the membership matrix U;
6: **Do** calculate three cluster centers;
7: Calculate the value function $J(U, c_1, c_2, \ldots, c_n)$;
8: Recalculate the membership matrix U;
9: **Until** $J(U, c_1, c_2, \ldots, c_n) < \varepsilon$
10: Return $Q_{VCPU}, Q_{VMem}, Q_{VIO}$;

After clustering the task sequences, these sequences have been divided into three queues: CPU-intensive queue (Q_{VCPU}), memory-intensive queue (Q_{VMem}), and I/O-intensive queue (Q_{VIO}), which are defined as

$$\begin{cases} Q_{VCPU} = \{V_1, V_2, \ldots, V_a\} \\ Q_{VMem} = \{V_{a+1}, V_{a+2}, \ldots, V_{a+b}\} \\ Q_{VIO} = \{V_{a+b+1}, V_{a+b+2}, \ldots, V_n\} \end{cases} \tag{7}$$

Among them, Q_{VCPU} contains a task, the length of the Q_{VMem} is b, and the length of Q_{IO} is $n - a - b$. The total length of the three queues is n.

4.2. Resources Load Sorting

For the load status of node resources, we have introduced different types of PMC [23] mechanisms to evaluate it. Recent research results have shown that the PMC mechanism can accurately and intuitively reflect the use of hardware components during system operation. By reading PMC data, it can accurately measure system energy consumption and monitor resource activity with little overhead [23,24].

In general, VMs deployed on each node resource trigger various types of PMC events during execution. From the hardware perspective, PMC events represent that some node resources are active. Based on the PMC mechanism, this paper monitors the number of resources activity events such as CPU (P^{CPU}), Mem (P^{Mem}), Disk (P^{Disk}), and Net (P^{Net}) triggered by the execution of VMs, and respectively records the PMC event set associated with the specific resource device. Combining the proportion of various events in the overall PMC event of the system can sort the load status of the node resources in a specific period of time, and can be formulated as

$$f_i^x = \sum_{X \epsilon \{CPU, Mem, IO\}} \frac{P_i^X(t_1, t_2)}{P^X(t_1, t_2)}, i = \{1, 2, \ldots, n\} \tag{8}$$

where $P_i^X(t_1, t_2)$ represents the PMC event related to resource X triggered by the execution of VM_i during the time period $[t_1, t_2]$, and $P^X(t_1, t_2)$ represents the total PMC event related to the resource X. Using the proportional relationship between the two, we indicate three indicators (f_i^{CPU}, f_i^{Mem}, and f_i^{io}) respectively represent the CPU usage, memory usage, and I/O latency wait time when VM_i is executed. Since the I/O access process during task scheduling is closely related to disk and network bandwidth activity events, P_i^{IO} can be used to represent these two sets of PMC events. P_i^{IO} is formulated as

$$P_i^{IO} = P_i^{Disk} + P_i^{Net} \tag{9}$$

Therefore, the resources load sorting based on the PMC mechanism is described by the proportion of PMC events triggered by VM to the total PMC events. The relevant data can be obtained through the PMC programming interface. The three indexes of f_i^{CPU}, f_i^{Mem}, and f_i^{io} are respectively sorted from small to large for all resources. The three queues are formulated as

$$\begin{cases} Q_{CPU} = \{C_{1CPU}, C_{2CPU}, \ldots, C_{mCPU}\} \\ Q_{Mem} = \{C_{1Mem}, C_{2Mem}, \ldots, C_{mMem}\} \\ Q_{IO} = \{C_{1IO}, C_{2IO}, \ldots, C_{mIO}\} \end{cases} \tag{10}$$

It should be noted that cloud resources are only sorted without static classification after the resources load status is distinguished due to the dynamic nature. All the resources are included in the three queues, unlike the task scheduling queues.

4.3. Staggering Peak Policy

Based on the diversity requirements of cloud tasks, mixed workloads are generally divided into three types: compute-intensive, memory-intensive, and data-intensive tasks. Most of compute-intensive tasks include calculations, logical judgments, and other CPU actions that consume CPU resources, resulting in high CPU usage peak. Memory-intensive tasks mainly take up a large number of storage resources. Data-intensive tasks involve frequent network transfers and large data reading and writing of I/O devices such as disks. This type of task consumes very little CPU and spends most of its time on I/O processing of data and operations waiting for I/O operations to complete, because I/O operations are much slower than CPU and memory.

When scheduling the task sequences, it is necessary to consider the different characteristics and the diversity requirements. During the execution of cloud tasks, the use of CPU, memory, and I/O resources do not conflict with each other. For example, when the cloud platform performs a task requesting a large number of CPU computing resources, the memory and I/O resources are generally in an idle state, and the non-CPU-intensive task can be scheduled for the resource to balance the load, improve scheduling efficiency, and optimize cost. This paper proposes the SPSLAM algorithm based on the analyzed task sequences classification and resources load sorting to make full use of these idle resources in cloud computing. The SPSLAM is presented in Algorithm 2.

Algorithm 2: SPSLAM

Input: $List_{schedule}, \{Q_{VCPU}, Q_{VMem}, Q_{VIO}\}, \{Q_{CPU}, Q_{Mem}, Q_{IO}\}$
Output: $bestSchedule$
1: **for** $V_{ii} \in List_{schedule}$
2: **if** $prec(V_{ii})$ has been scheduled **then**
3: **if** $V_{ii} \in Q_{VCPU}$ **then**
4: Schedule V_{ii} into Q_{CPU};
5: **End if**
6: **if** $V_{ii} \in Q_{VMem}$ **then**
7: Schedule V_{ii} into Q_{Mem};
8: **End if**
9: **if** $V_{ii} \in Q_{VIO}$ **then**
10: Schedule V_{ii} into Q_{IO};
11: **End if**
12: **End if**
13: **End for**

The SPSLAM algorithm proposed in this paper distinguishes the types of tasks and the load status of node resources during task scheduling. If V_{ii} is a data-intensive task, it should be scheduled to Q_{IO}.

The complexity of the SPSLAM algorithm is composed of three parts: task sequences classification, resources load sorting, and staggering peak policy. The time complexity of the task sequences classification algorithm based on FCM is $O(t \times N)$, where t is the number of iterations, and N is the number of tasks. The resources load sorting complexity is $O(MlogM)$, where M is the number of the resources. The time complexity of the staggering peak policy is $O(N)$. Therefore, the complexity of the SPSLAM is $O(t \times N + MlogM)$. Due to the fact that the task sequences classification and the resources load sorting can be executed at the same time, it will not cause too much extra time overhead for the system. Actually, the SPSLAM complexity is less than $O(t \times N + MlogM)$.

5. Experiments and Evaluation

In this section, we discuss the experimental results to evaluate the performance of SPSLAM. We used a workflow simulator (WorkflowSim [25]) and performed simulation scheduling experiments on the simulator. The task sequences with a different number of tasks were generated by the generator (WorkflowGenerator [26]). The simulator and generator were the open source software developed by

the Pegasus group at the University of Southern California. The experimental results were compared with fuzzy clustering and two level based task scheduling (FCTLBS) [11], multi-queue interlacing peak scheduling method (MIPSM) [13], and least connection (LC) [18].

5.1. Workloads and Setup

In this paper, to address the issue of scheduling mixed workloads in cloud computing, three scientific workflow task sequences (Montage [27], CyberShake [28], and Epigenomics [29]) were selected for the experiments, which were respectively derived from astronomy, seismology and human genetics.

We used the above three scientific workflows to generate task sequences combinations with a different number of tasks. The combinations formed three sets of test workloads (Montage-25, CyberShake-30, Epigenomics-24), (Montage-50, CyberShake-50, Epigenomics-50), (Montage-100, CyberShake-100, Epigenomics-100), which were respectively recorded as WL#1, WL#2, and WL#3. The characteristics of some of the tasks in these three scientific workflows are shown in Table 1. For example, the task fastqSplit in the workflow Epigenomics is a memory-intensive task, which requires plenty of storage resources.

Table 1. Scientific workflow characteristics.

Name	Category	Type-intensive	Count	Runtime (s)	Inputs (MB)	Outputs (MB)
mConcaFit	Montage	CPU	1	13.60	0.03	0.02
mAdd	Montage	CPU	1	30.34	357.28	330.86
extract_agt	CyberShark	I/O	19	355.31	38786.64	441.97
fastqSplit	Epigenomics	Mem	2	41.78	462.20	462.20
map	Epigenomics	I/O	146	9635.01	2964.24	0.58

All experiments were run on Dell machines with the Inter(R) Core(TM) i5-6500 processor, 3.20 GHZ, 4 GB memory, and Windows7 64-bit operating system. We used the WorkflowSim to generate a data center with 600 hosts, each with 10 VMs. The features of VM referred to the type of Amazon EC2 VM [27]. The specific parameters of the five types of VMs are shown in Table 2.

Table 2. Five kinds of virtual machine (VM) types.

VM	CPU (MIPS)	RAM (GB)	BW (M/S)	Price ($/h)
1	2500	0.85	750	4.65
2	2000	3.75	500	4.03
3	1000	1.70	980	3.71
4	750	1.25	1043	3.67
5	500	0.61	558	3.24

5.2. Experiment Metrics

The experiment metrics of the cloud staggering peak scheduling policy for mixed workloads proposed in this paper mainly include load balancing, efficiency, and cost optimization. The efficiency is mainly represented by the total completion time of the execution task sequences and the task success rate.

Load balancing (LB) can be expressed by the mean square error of each resource load in this paper, which refers to the degree of load balancing between resources. The LB is formulated as

$$\mathrm{LB} = \frac{1}{m}\sum_{j=1}^{m}\left(\mathrm{Load}_j - \mathrm{Load}_{avg}\right)^2 \tag{11}$$

where $Load_j$ represents the load of R_j, and $Load_{avg}$ represents the average load of each resource. The smaller the LB is, the more balanced the system load system gets in cloud computing.

The task completion time (Makespan) is the total execution time of all task sequences submitted by the user. The Makespan is formulated as

$$\text{Makespan} = \max_{i \in N}\{\text{Makespan}_i\} \tag{12}$$

where N represents the number of task sequences, and $Makespan_i$ represents the completion time of the ith task sequence.

The task success rate (V_s). If the completion time of V_{ij} is less than the deadline, V_{ij} will be considered to be successfully executed under the deadline constraint. The V_s can be formulated as

$$V_s = \frac{n_s}{N} \times 100\% \tag{13}$$

where n_s is the number of successfully executed tasks in N tasks.

Resource usage cost (SumCost). This paper introduced the time-based charging method in the Amazon EC2 cloud environment [30], which was charged in hours. Therefore, the execution cost of each task sequence is the sum of the cost of each VM. The SumCost can be formulated as

$$\text{SumCost} = \sum_{i=1}^{n}\sum_{j=i}^{k}\text{Cost}(VM_{i,j}) \tag{14}$$

where $Cost(VM_{i,j}) = [EFT(V_{end}, VM_{i,j}) - EST(V_{start}, VM_{i,j})] \times P(VM_{i,j})$. Among them, $VM_{i,j}$ represents the *jth* VM on the *ith* node resource, $P(VM_{i,j})$ represents the cost per unit time, $EFT(V_{end}, VM_{i,j})$ represents the completion time of the last executed task on $VM_{i,j}$, and $EST(V_{start}, VM_{i,j})$ represents the starting time of the earliest execution of the task on $VM_{i,j}$. k represents the number of VMs on each node resource.

5.2.1. Load Balancing

Figure 2 shows the performance of the four algorithms in load balancing by LB where the total number of tasks is different in all task sequences. The smaller the LB is, the more balanced the load between cloud resources gets. As is shown in Figure 2, as the number of tasks increases, the load of each VM resource also increases, causing the host to be overloaded and a decrease in LB. The LB of the SPSLAM algorithm has been relatively small and has little change, and the MIPSM algorithm is second. The FCTLBS algorithm performed the worst because it only statically clustered resources, and did not consider the dynamic nature of resources load, which led to imbalanced distribution of node load. The LC algorithm got better load balancing effect because it always prioritized the resources with lighter load during scheduling. However, when the number of tasks was too much, the effect became worse. The experimental results showed the advantages of the staggering peak scheduling policy based on task diversity and the dynamic nature of resources.

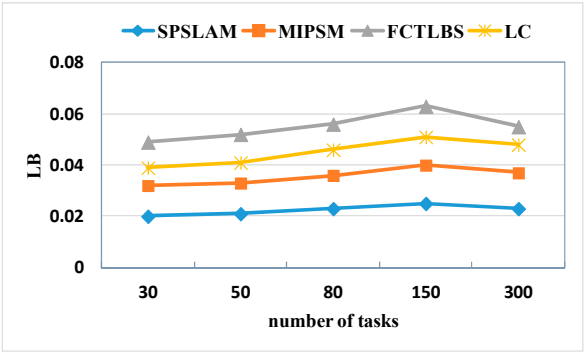


Figure 2. Load balancing degree of four scheduling algorithms.

5.2.2. Scheduling Completion Time

Figure 3 shows the scheduling completion time of the four algorithms in the case where the number of workflows is increased from 3 to 15. When the number of workflows is 3, the makespan of the four algorithms is not much different. With the increase of the number of workflows, the makespan of SPSLAM is significantly lower than FCTLBS, MIPSM, and LC. The reason was that SPSLAM staggered the resource usage peak and reduced the delay waiting time of the task when scheduling the workflow task sequences. In this way, it reduced the scheduling completion time and finally improved the task scheduling efficiency.

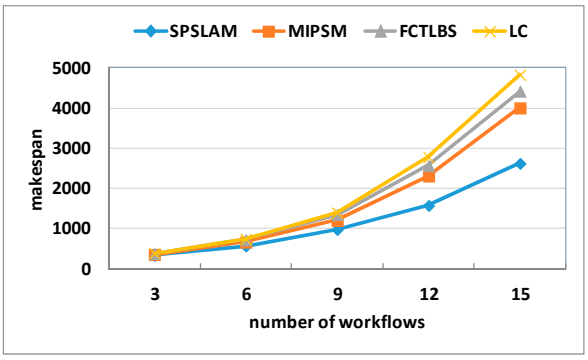


Figure 3. Schedule completion time under different workflow numbers.

5.2.3. Task Success Rate

Figure 4 shows the scheduling effect by V_s. The abscissa is the total number of tasks in all task sequences, and the ordinate represents the success rate of different algorithms. As is shown in Figure 4, as the number of tasks increases, the resource competition occurs, and the V_s of tasks decreases correspondingly. The LC algorithm did not consider task characteristics and its diversity when scheduling mixed workloads, and the performance was worse. The FCTLBS algorithm only pre-processed and statically clustered resources, which led to poor performance. The MIPSM algorithm only performed hard clustering on tasks, and did not consider the fuzziness of matching scheduling between task and resource, which had certain limitations. The SPSLAM algorithm had a task success rate of 90%, which was within an acceptable range. The experimental results showed that this policy had obvious advantages for mixed workloads.

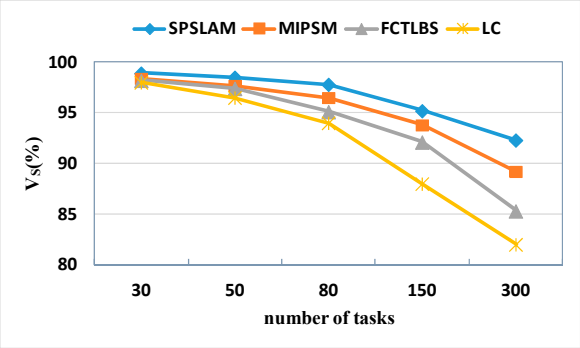
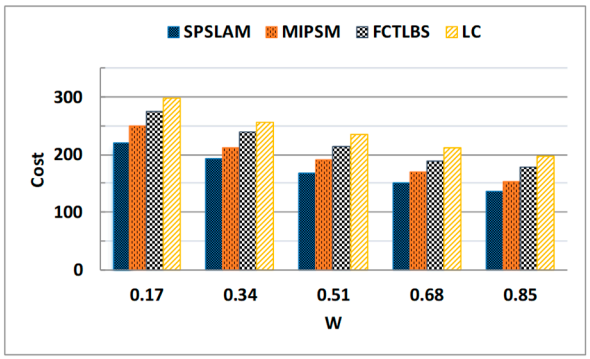


Figure 4. Success rate of tasks under different task numbers.

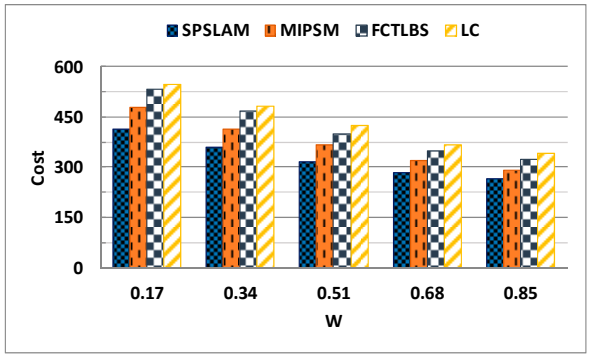
We finished two sets of experiments to show scalability of the proposed framework by using system efficiency metrics. The first experiment verified the scheduling completion time of the four algorithms in the case where the number of workflows was increased from 3 to 15. When the number of workflows was 3, the makespan of the four algorithms was not much different. With the increase of the number of workflows, the makespan of SPSLAM was significantly lower than FCTLBS, MIPSM, and LC. This proved that the proposed framework reduced the delay time of task scheduling and the scheduling completion time. The second experiment verified the scheduling effect by V_s. As the number of tasks submitted by users increased, the resource competition occurred, and the V_s of tasks decreased correspondingly. However, the proposed framework still had a task success rate of 90%, which was within an acceptable range. It can be concluded that the SPSLAM algorithm had relatively high V_s and throughput when the task workloads submitted by users were increased. It improved the system efficiency effectively and was of good scalability.

5.2.4. Resource Usage Cost

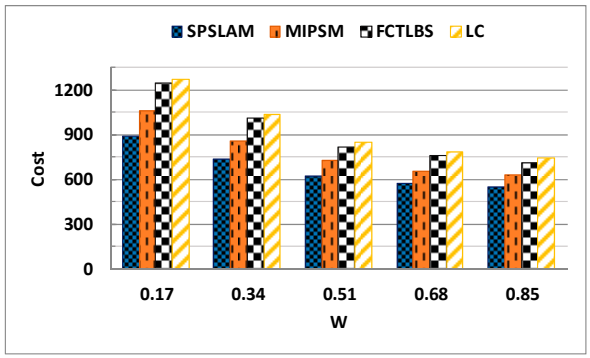
Figure 5 shows the scheduling cost of the four algorithms under different deadline constraints. The experiment was divided into three groups, and each group executed different test workloads (WL#1, WL#2, and WL#3). We selected the five values in the minimum and maximum completion time intervals as the deadline constraints, deadline $= t_{min} + W \times (t_{max} - t_{min})$, where W was the weight parameter, and $W \in \{0.17, 0.34, 0.51, 0.68, 0.85\}$. As is shown in Figure 5, when the constraint on the deadline is reduced, the scheduling scheme that satisfies the time constraint is gradually increased and the cost of the four scheduling algorithms is reduced. Among the four algorithms, the resource usage cost generated by SPSLAM was the lowest. The reason was that the SPSLAM algorithm considered the characteristics of each task for mixed workloads and used the FCM algorithm to divide the task sequences into three categories according to the resources requirements such as CPU, memory, and I/O, which reduced the blindness of tasks scheduling. The resources were sorted by CPU size, memory, and I/O load size by monitoring the load status to narrow the resource selection range. SPSLAM considered the task characteristics and resources load status when scheduling multiple task sequences, which improved the task scheduling efficiency and reduced the resource usage cost.



(**a**) WL#1



(**b**) WL#2



(**c**) WL#3

Figure 5. Scheduling cost under different deadlines.

6. Conclusions

To address the issue of scheduling mixed workloads in cloud computing, a novel cloud task staggering peak scheduling policy SPSLAM based on task characteristics and resource load status was proposed. This policy consisted of three parts: task sequences classification, resources load sorting and scheduling scheme generation. First, based on different task characteristics, the task sequences were divided into queues of different types by the fuzzy clustering algorithm. Second, PMC mechanism was introduced to dynamically monitor the load status of resource nodes and respectively sort the resources by the metrics of CPU, memory, and I/O load size. Finally, the task sequences of specific type were scheduled for the corresponding light loaded resources, and the resources usage peak was staggered to achieve load balancing. The experimental results showed that the proposed policy was superior to other algorithms in terms of load balancing and system efficiency, and reduced resource usage cost.

In the future, there are still many open issues that can be further explored in this new-born system. First, we will research for better task clustering and resource state monitoring methods in cloud computing. In addition to research strategies for system efficiency and load balancing, energy efficiency optimization can also be studied. Second, the computation-intensive and bandwidth-intensive tasks may seriously interfere with each other in cloud computing. We will present an interference prediction model from the proposed SPSLAM framework that mitigates the interference effects and greatly improves the system performance. Finally, we will apply the SPSLAM framework to more practical applications such as the field of mobile edge computing. We will further prove its superiority and application value in MEC. These are problems that are worth taking further study.

Author Contributions: Conceptualization, Y.T. and M.Z.; Formal analysis, Y.T. and C.C.; Funding acquisition, Z.H.; Software, C.C.; Supervision, C.C.; Validation, Z.H.; Visualization, Z.H.; Writing—original draft, Y.T.; Writing—review & editing, Y.T. and M.Z.

Funding: This research received no external funding.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (No. 61602525 and 61572525). The authors are grateful to the anonymous reviewers for their helpful comments and feedback.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hameed, A.; Khoshkbarforoushha, A.; Ranjan, R.; Jayaraman, P.P.; Kolodziej, J.; Balaji, P.; Zeadally, S.; Malluhi, Q.M.; Tziritas, N.; Vishnu, A.; et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* **2016**, *98*, 751–774. [CrossRef]
2. Yousafzai, A.; Gani, A.; Noor, R.M.; Sookhak, M.; Talebian, H.; Shiraz, M.; Khan, M.K. Cloud resource allocation schemes: Review, taxonomy, and opportunities. *Knowl. Inf. Syst.* **2017**, *50*, 347–381. [CrossRef]
3. Madni, S.H.H.; Latiff, M.S.A.; Coulibaly, Y.; Abdulhamid, S.M. Resource scheduling for infrastructure as a service (IaaS) in cloud computing. *J. Netw. Comput. Appl.* **2016**, *68*, 173–200. [CrossRef]
4. Kapil, D.; Tyagi, P.; Kumar, S.; Tamta, V.P. Cloud Computing: Overview and Research Issues. In Proceedings of the International Conference on Green Informatics, Fuzhou, China, 15–17 August 2017.
5. Wang, R.; Shang, P.; Zhang, J.; Wang, Q.; Liu, T.; Wang, J. MAR: A Novel Power Management for CMP Systems in Data-Intensive Environment. *IEEE Trans. Comput.* **2016**, *65*, 1816–1830. [CrossRef]
6. Mon, E.E.; Thein, M.M.; Aung, M.T. Clustering based on task dependency for data-intensive workflow scheduling optimization. In Proceedings of the Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS), Salt Lake City, UT, USA, 14 November 2016.
7. Yang, J.; Meng, Q.; Wang, S.; Li, D.; Huang, T.; Dou, W. Energy-Aware Tasks Scheduling with Deadline-constrained in Clouds. In Proceedings of the IEEE International Conference on Advanced Cloud and Big Data, Chengdu, China, 13–16 August 2016.
8. Ye, X.; Liang, J.; Liu, S.; Li, J. A Survey on Scheduling Workflows in Cloud Environment. In Proceedings of the International Conference on Network and Information Systems for Computers, Wuhan, China, 23–25 January 2015.
9. Tan, Y.; Wu, F.; Wu, Q.; Liao, X. Resource stealing: A resource multiplexing method for mix workloads in cloud system. *J. Supercomput.* **2016**, *6*, 1–17. [CrossRef]
10. Hanani, A.; Rahmani, A.M.; Sahafi, A. A multi-parameter scheduling method of dynamic workloads for big data calculation in cloud computing. *J. Supercomput.* **2017**, *73*, 1–27. [CrossRef]
11. Wang, X.; Wang, Y.; Hao, Z.; Du, J. The Research on Resource Scheduling Based on Fuzzy Clustering in Cloud Computing. In Proceedings of the International Conference on Intelligent Computation Technology & Automation, Nanchang, China, 14–15 June 2016.
12. Liu, L.; Mei, H.; Xie, B. Towards a multi-QoS human-centric cloud computing load balance resource allocation method. *J. Supercomput.* **2016**, *72*, 1–14. [CrossRef]
13. Zuo, L.; Dong, S.; Shu, L.; Zhu, C.; Han, G. A Multiqueue Interlacing Peak Scheduling Method Based on Tasks' Classification in Cloud Computing. *IEEE Syst. J.* **2018**, *12*, 1518–1530. [CrossRef]
14. Khorandi, S.M.; Sharifi, M. Scheduling of online compute-intensive synchronized jobs on high performance virtual clusters. *J. Comput. Syst. Sci.* **2016**, *85*, 1–17. [CrossRef]

15. Wang, H.; Li, T.; Shea, R.; Ma, X.; Wang, F.; Liu, J.; Xu, K. Toward Cloud-Based Distributed Interactive Applications: Measurement, Modeling, and Analysis. *IEEE/ACM Trans. Netw.* **2018**, *26*, 3–16. [CrossRef]
16. Mao, L.; Li, Y.; Peng, G.; Xu, X.; Lin, W. A Multi-Resource Task Scheduling Algorithm for Energy-Performance Trade-offs in Green Clouds. *Sustain. Comput. Inf. Syst.* **2018**, *19*, 233–241. [CrossRef]
17. Dave, A.; Patel, B.; Bhatt, G. Load balancing in cloud computing using optimization techniques: A study. In Proceedings of the International Conference on Communication & Electronics Systems, Coimbatore, India, 21–22 October 2017.
18. Chen, Y.; Zhang, Z.; Ren, J. Improved Minimum Link Load Balancing Scheduling Algorithm. *Comput. Syst. Appl.* **2015**, *24*, 88–92.
19. Zhao, Z.; Martin, P.; Jones, A.; Taylor, I.; Stankovski, V.; Salado, G.F.; Suciu, G.; Ulisses, A.; de Laat, C. Developing, Provisioning and Controlling Time Critical Applications in Cloud. In Proceedings of the European Conference on Service-oriented & Cloud Computing, Oslo, Norway, 27–29 September 2017.
20. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
21. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the IEEE International Conference on Communications, Paris, France, 21–25 May 2017.
22. Jošilo, S.; Dán, G. A game theoretic analysis of selfish mobile computation offloading. In Proceedings of the IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017.
23. Dargie, W. A Stochastic Model for Estimating the Power Consumption of a Processor. *IEEE Trans. Comput.* **2015**, *64*, 1311–1322. [CrossRef]
24. Jijun, W.; Hua, C. Computer Power Estimation Model Based on Performance Events. *Appl. Res. Comput.* **2017**, *34*, 734–738.
25. Chen, W.; Deelman, E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In Proceedings of the IEEE, International Conference on E.-Science, Chicago, IL, USA, 8–12 October 2012.
26. Bharathi, S.; Chervenak, A.; Deelman, E.; Mehta, G.; Su, Me.; Vahi, K. Characterization of scientific workflows. In Proceedings of the Third Workshop on Workflows in Support of Large-Scale Science, Austin, TX, USA, 17 November 2008.
27. Montage: An Astronomical Image Engine. Available online: http://montage.ipac.caltech.edu (accessed on 30 March 2018).
28. Southern California Earthquake Center. Available online: http://www.scec.org (accessed on 30 March 2018).
29. Illumina. Available online: http://www.illumina.com/ (accessed on 30 March 2018).
30. Panwar, R.; Mallick, B. Load balancing in cloud computing using dynamic load management algorithm. In Proceedings of the International Conference on Green Computing and Internet of Things, Noida, India, 8–10 October 2015.