Applied Mathematics, Modeling and Computer Simulation
C.-H. Chen et al. (Eds.)

doi:10.3233/ATDE221107

Research on Deadline Division and Scheduling of Batch Scientific Workflow in Cloud Environment

Qing ZHAO [a,1], Kuo FENG[b], Yancui SHI[a] , Chengkui ZHANG[a], Le SUN[a], Mengxiang ZHANG[a]
[a] *Tianjin University of Science and Technology, Tianjin, China*
[b] *Tianjin No.1 Middle School Binhai School, Tianjin, China*

Abstract. Data intensive batch processing scientific workflow is a typical application model in the era of big data. A reasonable scheduling method can improve the resource utilization rate and reduce the rental cost on the premise of meeting the deadline requirements. In this paper, an iterative floating interval allocation method suitable for batch scientific workflow is proposed in the deadline allocation stage, and then in the resource mapping stage, a task scheduling algorithm considering the utilization of time gaps and minimizing the cost of renting virtual machines is proposed, which weighted the expected utilization of time slices. Experiments show that compared with similar scheduling algorithms, in the specified deadline time, it can not only better solve the data transmission bottleneck, but also better improve the execution efficiency and reduce the total rental cost.

Keywords. Batch processing scientific workflow, Task scheduling, Deadline Division

1. Introduction

The resource allocation and task scheduling of cloud platform for scientific workflow tasks has always been a research hotspot. It is an important means to optimize the execution efficiency, resource cost and energy consumption of multi tasks with mutual dependencie.

In the era of big data, with the increasing scale of data processing and the increasing computing complexity, a new scientific workflow, batch scientific workflow, has begun to attract attention. It is usually used in data intensive applications. Because the massive data must rely on the distributed or cloud computing environment to speed up, some task nodes are modeled as batch task groups containing a large number of independent tasks of the same type. At present, this kind of scientific workflow has important applications in many big data science fields, such as signal processing, medical image processing, astronomical scientific computing and so on. Therefore, the research of more mature and perfect cloud scheduling algorithm for batch scientific workflow is imminent.

1 Corresponding Author: Qing Zhao, Tianjin University of Science and Technology, Tianjin City, China. E-mail: zhaoqing@tust.edu.cn.

The scheduling of scientific workflow with limited execution time is the most common problem in reality, and at present, some achievements exist, such as the fuzzy set prediction model, M/M/m queuing theory model and other methods [1-2]. Abrisami[3] proposed a PCP scheduling algorithm considering time gap, and Ghafarian[4] proposed a sub workflow partition scheduling algorithm. Visheratin et al. [5] proposed deadline genetic algorithm (CDCGA). These algorithms have their own emphasis on scheduling results, but there are still some problems. For example, the PCP algorithm ignores the gap between tasks due to the partial order relationship, and does not fully consider the global time gap; The improved IC-PCPD2 algorithm [6] proposed later allocates the gap according to the execution proportion of the task, but does not consider the allocated time gap in the previous path, which often leads to the problem that the scheduling space of the previous task is too loose and the subsequent is too tight. In addition, these time proportional allocation methods are oriented to general scientific workflow. For batch scientific workflow, the estimated execution time of the task group is related to the selection of virtual machines and the execution parallelism of subtasks, which requires additional strategies to estimate. Therefore, this paper proposes an iterative floating interval division method for batch scientific workflow, which completes the division of workflow deadlines by reasonably allocating time gaps.

After the deadline is divided, the dynamic scheduling of tasks can be carried out according to the local deadline allocation of subtasks. In the existing workflow scheduling algorithm, Zhu et al. [7] proposed to schedule scientific workflow to the leased cloud computing cluster to maximize the utilization of the cluster. Vahid et al. [8] proposed two workflow scheduling algorithms: proportional deadline constrained PDC and deadline constrained critical path DCCP. Therefore, this paper proposes a TGbMF task scheduling algorithm considering the utilization of time gap, so as to optimize the overall execution efficiency and resource rental cost of batch workflow.

2. Virtual Machine Type Initialization and Workflow Deadline Division

2.1. Determine the Initialization Virtual Machine Type and Parallelism of the Task Group

In the deadline division stage, we need to consider the task volume of each task group, which is related to its task type and quantity. We need to initialize the virtual machine type and parallelism for each task group before we can estimate the execution time, and then reasonably allocate the deadline.

Suppose that the cloud platform provides three types of virtual machines, including balanced, high CPU and high memory. Each batch group is divided into the most appropriate category according to its task type, and can only be adjusted at m different levels under this category. We are based on this common sense: when the computer performance is enough to support task operation, simply doubling the configuration will generally double the rental unit price, but the increase in execution efficiency is often less than twice. Therefore, the initialization method of virtual machine type is to select the configuration with the lowest performance of the virtual machine type.

Expected runtime etc is an n×m matrix, in which each a represents the expected execution time of a single task in the task batch ti on the virtual machine of level j under its category.

For the selection of the number of virtual machines in the task batch, we consider: according to the current virtual machine level selection, query the ETC matrix to determine the estimated execution time etc_{ij} of a single task in the task batch. Let L be the minimum rental charging duration of the virtual machine. There are $taskN_i$ independent subtasks of the same type on the task batch ti. Calculate the number of virtual machines with the highest parallelism based on making full use of the minimum rental duration resources of the virtual machine as follows:

$$x = \left\lceil \frac{taskN_i}{\left\lfloor \frac{L}{etc_{ij}} \right\rfloor} \right\rceil \tag{1}$$

2.2. *Workflow Deadline Division*

2.2.1 Task Relevance Clustering

In this paper, before dividing the deadline, the local adjacent tasks are clustered by the correlation degree, so that the tasks with frequent data transmission can be scheduled as a whole, so as to optimize the global data communication. At present, task clustering algorithms include hypergraph segmentation, BEA matrix transformation and DAG Graph Segmentation, but their time complexity is high. Therefore, this paper clusters the two tasks with a large amount of data transmission in order to reduce the cross-node communication overhead. Taking figure 1 as an example, two task packages {v1, v2, v5} and {v3, v4, v6} are formed.

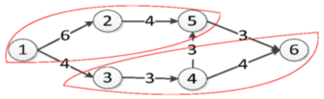

Figure 1. Data association degree clustering

2.2.2 Task Deadline Partition Based on Time Gap for Batch Scientific Workflow

Assigning time gaps to subtasks can make local task scheduling obtain greater relaxation space, and thus have the opportunity to choose a scheduling scheme with lower cost. A reasonable redistribution strategy can reduce the cost of resources in a balanced way in the overall scope of the workflow. This paper presents an iterative allocation method suitable for batch scientific workflows.

The process of time gap allocation is a traversal process, and different traversal sequences will lead to different allocation results. This section adopts the method of iterative generation of critical path based on traversal sequence proposed by Abrisami et al. [3,6]. Let $HP = \{HP_{[1]}, HP_{[2]}, \cdots, HP_{[l]}\}$ represent the current critical path, HP[i] is the ith task on the path, and l is the total length of the critical path.

Then, time gap allocation is carried out. Suppose that the floating-point interval of the task is the sum of the total processing time of the task package and the time gap allocated to the task package, and let $f_{v_i}^{float}$ be the floating interval of task v_i. The derivation method is:

$$Time_y = \left\lceil \frac{taskN_y}{p} \right\rceil \cdot etc_{yu} \tag{2}$$

$$f_{v_i}^{float} = Time_y + \max_{i \in \beta_j} \{L_{i,j} / B\} + T_u^n + T_{S_i} \tag{3}$$

Where $Time_y$ is the time required for the y-th task batch to execute tasks using j virtual machines of u level under corresponding classes, where virtual machine type u and parallelism p are calculated from the initialization method in Section 2.2. β_j represents the direct precursor set of task package v_j, S_i represents the software unit of task v_i, whose loading time is T_{S_i}, and $L_{i,j}$ represents the amount of data to be transmitted from task v_i to v_j.

Let EFT, FFT and LFT represent the earliest start time, the earliest completion time and the latest completion time of the floating interval task package. If $FFT_{v_k} + f_{v_k}^{float} = LFT_{v_k}$, v_k belongs to the fixed task package. V^{fix} is the set of fixed task packages. The total time gap on the critical path is calculated as:

$$f_{HP}^{float} = \sum_{HP_{[k]} \in HP'} \left\{ EFT_{HP_{[k+1]}} - FFT_{HP_{[k]}} \right\} + LFT_{HP_{[l]}} - FFT_{HP_{[l]}} \tag{4}$$

$LFT_{HP_{[l]}} - FFT_{HP_{[l]}}$ represents the time gap of the critical path, where $HP' = HP / V^{fix} / \{HP_{[l]}\}$. The latest completion time of $HP_{[l]}$ in the algorithm is generated by the deadline. The overall time gap of the whole path is allocated to non-fixed task groups according to the workload proportion of each task group. Considering that the application oriented here is batch scientific workflow, it is not allocated according to the proportion of the execution time of a single task as in the traditional scheme, but is divided according to the proportion of the estimated execution time of the task group (see equation 2) calculated under the virtual machine type and parallelism initialization mode in Section 2.1. The specific time gap allocation formula is:

$$f_{v_i}^{dis} = f_{HP}^{float} \times f_{v_i}^{float} / \sum_{v_k \in HP/V^{fix}} \left\{ f_{v_k}^{float} \right\} \tag{5}$$

If $FFT_{v_i} + f_{v_i}^{dis} > LFT_{v_i}$, the time gap allocated to v_i is $f_{v_i}^{dis} = LFT_{v_i} - FFT_{v_i}$. The floating interval of v_i is $f_{v_i}^{float} = f_{v_i}^{float} + f_{v_i}^{dis}$, and the earliest and latest completion times of all subsequent tasks of v_i are updated. When all non-fixed task groups have completed the allocation of time gap, update the earliest and latest completion time of all tasks. If $f_{HP}^{float} < 0$, reassign the critical path until all time gaps are no longer updated.

3. Multi Rule Fusion (TGbMF) task Scheduling Algorithm based on Local Deadline Assignment

3.1 Depth Based Task Scheduling Order

Define ℓ as a set of all schedulable task packages, initialize ℓ as {v0}, and set the depth of the task package v_i as the minimum number of tasks from v0 to v_i. If ℓ is not empty, divide ℓ into multiple subsets according to the depth of the task package. Select the subset with the minimum depth, and then select the task package with the largest and earliest end time as the next task package to be scheduled. After scheduling the task package each time, update the set of all scheduled precursor packages in ℓ.

3.2 TGbMF Task Scheduling Method

Based on the previous deadline division, this chapter considers three aspects: the reuse of the remaining time slice, the cheapest execution cost and the utilization rate of the new lease time slice, and obtains a better effect of improving the execution efficiency and reducing the lease cost through weighted integration.

FNPA (Fewest amount of newly leased time priority algorithm): when scheduling, give priority to reusing the remaining time slices as much as possible [9], so as to improve the utilization of the leased time interval. Define $\Re_{v_i}^t$ and $\overline{\Re}_{v_i}^t$ as the number of lease intervals that v_i is scheduled to lease and the maximum possible time interval to lease, respectively. The specific algorithm is as follows:

$$\overline{\Re}_{v_i}^t = (T_{i,\lambda_t} + T_{v_i,t} + T_t + T_{i,t}^S) / L + 1 \tag{6}$$

The virtual machine type corresponding to time slot t is λ_t. T_{i,λ_t} represents the task processing time required by task package v_i when selecting a virtual machine of type λ_t, $T_{v_i,t}$ represents the data transmission time required when task package v_i is allocated to time slot s, T_t represents the installation time of the virtual machine, $T_{i,t}^S$ represents the installation time of the corresponding software, and L is the length of the billing interval. After normalization, the smaller the value of $\gamma_{v_i}^t$, the higher the priority, because it represents the smaller the interval that needs to be leased. The calculation method of $\gamma_{v_i}^t$ is:

$$\gamma_{v_i}^t = \Re_{v_i}^t / \overline{\Re}_{v_i}^t \tag{7}$$

LACA (Least actual cost priority algorithm): in some cases, scheduling tasks to time fragments first will reduce the execution efficiency and increase the cost. For example, when allocating the remaining time slice with high CPU configuration to memory complex tasks, we must choose between time slice reuse and high execution efficiency. Therefore, the virtual machine instance with lower cost is preferred first. The processing cost of each task package is calculated as follows:

$$M_{v_i}^{t} = (T_{i,\lambda_t} + T_{v_i,t} + T_t + T_{i,t}^{S} + T_b) \times P_\lambda / L \tag{8}$$

Where P_λ represents the price of the virtual machine, L represents the length of the billing interval, and $\varphi_\omega^{v_i}$ represents the set of all time slots in the time interval EFT_{v_i} to D_{v_i} on the virtual machine set in ω , and the algorithm for minimizing the processing cost is as follows:

$$\hbar_{v_i}^{t} = \frac{M_{v_i}^{t}}{\max\limits_{t' \in \varphi_\omega^{v_i}} M_{v_i}^{t'}} \tag{9}$$

NLEA (New lease expectation algorithm): when a new time slice needs to be leased, the lease priority can be determined by calculating the expected utilization of the new time slice. Let $\tau_{v_i}^{t}$ be the length of the newly leased time fragment when v_i is scheduled to time slot t, then the weighted priority on each time gap can be calculated as:

$$\Psi_{v_i}^{t} = \frac{\tau_{v_i}^{t}}{2 \times L} \tag{10}$$

The smaller the $\Psi_{v_i}^{t}$ value, the higher the scheduling priority. Iterate the allocation process until all current tasks are scheduled.

In this paper, by means of weighted fusion, the above three different factors are comprehensively considered, and the selection of weights is finally determined through the experiment of specific application scenarios, as shown in formula 11. Where, t is the time slot.

$$\Gamma_{v_i}^{t} = \gamma_{v_i}^{t} \times a + \hbar_{v_i}^{t} \times b + \Psi_{v_i}^{t} \times c \tag{11}$$

4. Experimental Results and Analysis

This paper uses the Workflow Generator to generate the test sets of montage and cybershake workflows. These workflows contain 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 tasks respectively. The real virtual machine service provided by Amazon EC2 is used for virtual machine modeling, and the billing unit is hour.

4.1 Weight Selection Experiment

a. b and c weights are tested by single factor analysis. Each time, assume that the two weights are 1, and test the effect of the third weight increasing from 0 to 10000. Figure 2 and 3 shows the influence of the value of parameter a on TGbMF algorithm.

It can be seen from figure 2 that the rent cost optimization becomes better and better when the value of a rises from 0 to 1, which shows that the algorithm FNPA has a good optimization effect. As the value of a continues to increase, the rental cost optimization is not obvious. This is because, with the continuous rise of value a, the algorithm is very dependent on using the leased time slot, but the leased time slot is limited, and it will ignore whether the virtual machine matches the task.

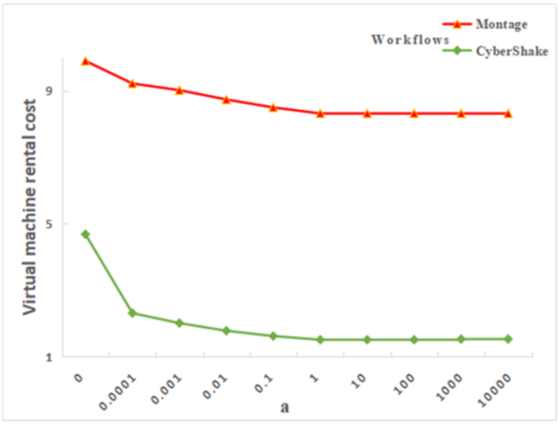

Figure 2. Weight a Test Results of Workflow Montage and CyberShake

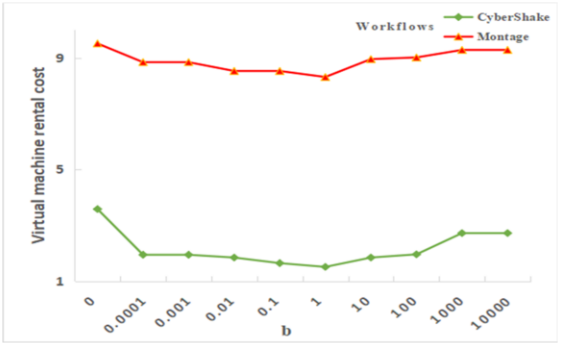

Figure 3. Weight b Test Results of Workflow Montage and CyberShake

Figure 4 shows the influence test results of parameter c. At this time, both a and b are taken as 1. It can be seen that the performance of c=1 and c=10 algorithms is the best. Therefore, this paper selects two groups of parameter combinations {a=1, b=1, c=10} and {a=1, b=1, c=1} to compare the experimental results again. The results show that when a=1, b=1 and c=10, the result of parameter combination optimization is better, therefore, it is selected in the subsequent experiments of this paper.

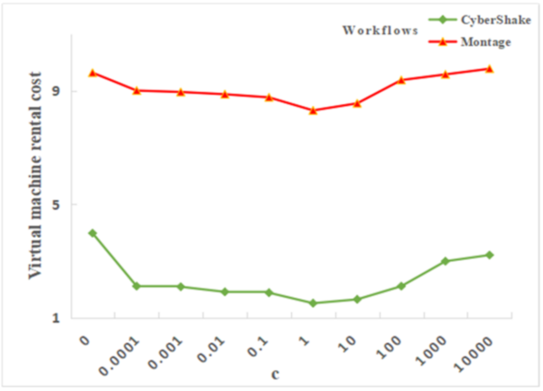

Figure 4. Weight c Test Results of Workflow Montage and CyberShake

4.2 Comparison Experiment with Reference Algorithms

This paper compares the proposed TGbMF algorithm with the IC-PCP algorithm using time gap proposed by Abrishami et al. [3,6] and the scheduling algorithm MRH using time gap integration proposed by Li[10]. In order to be consistent with the factors considered in this paper, IC-PCP algorithm adds virtual machine loading time and software installation time, and MRH algorithm adds task data correlation clustering operation.

Figure 5 shows the comparison diagram of deadline generated by comparing the TGbMF algorithm with parameter values, Li's MRH algorithm and abrishami's IC-PCP algorithm under the same number of tasks and the same considerations.

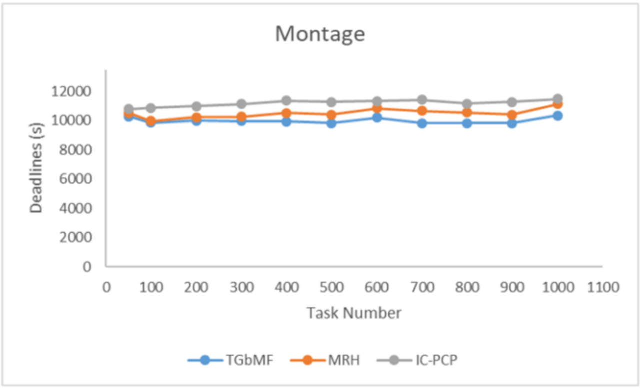

Figure 5. Comparison chart of deadline

It can be seen from figure 5 that the total time of deadline of TGbMF algorithm is 1.3% to 7.9% lower than that of Li's MRH algorithm, with an average of 4.8%, and 4.5% to 13.8% lower than that of Abrishami's IC-PCP algorithm, with an average of 10.6%.

Figure 6 shows the total rental cost comparison chart generated by comparing TGbMF algorithm, Li's MRH algorithm and Abrishami's IC-PCP algorithm with the same number of tasks and the same considerations.

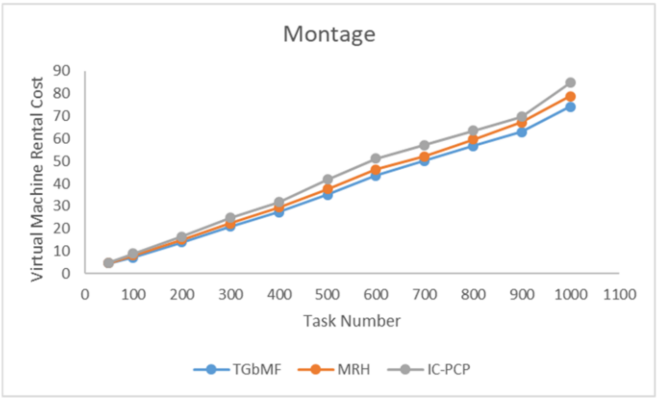

Figure 6. Total service rental cost comparison chart

As can be seen from figure 6, the overall rental cost of TGbMF algorithm is 1.9% to 11.1% lower than Li's MRH algorithm, with an average of 6.14%, and 4.1% to 18.3% lower than Abrishami's IC-PCP algorithm, with an average of 13.1%.

5. Conclusion

Aiming at the characteristics of batch scientific workflow with time constraints in cloud environment, an iterative dealine partition strategy with floating interval and a multi strategy heuristic algorithm TGbMF integrating time slice utilization are proposed. Experiments show that compared with similar scheduling algorithms, it can not only better solve the data transmission bottleneck within the specified dealine time, but also better improve the execution efficiency and reduce the total rental cost.

Acknowledgment
This research was supported by the Natural Science Foundation of Tianjin City (18JCQNJC69800), Research Plan Project of Tianjin Municipal Education Commission (2017KJ035), the National Natural Science Foundation of China (11803022, 61702367), and Tianjin Science and Technology Commissioner project (No. 20YDTPJC00560).

References

[1] Kong Xiang-zhen, Lin Chuang, Jiang Yi-xin, et al. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction[J]. Journal of Network & Computer Application, 2011, 34 (4):1068-1077.
[2] Cao Jun-wei, Kai Hwang , Li Ke-qin, et al. Optimal multiserver configuration for profit maximization in cloud computing[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(6):1087-1096.
[3] Abrishami S, Naghibzadeh M, Epema D. Cost-driven scheduling of grid workflows using partial critical paths[J]. IEEE Transactions on Parallel & Distributed Systems, 2012, 23(8):1400-1414.
[4] Ghafarian T, Javadi B. Cloud-aware data intensive workflow scheduling on volunteer computing systems[J]. Future Generation Computer Systems, 2015, 51:87-97.
[5] Visheratin Alexander A, Melnik Mikhail, Nasonov Denis. Workflow scheduling algorithms for hard-deadline constrained cloud environments[J]. Procedia Computer Science, 2016, 80: 2098-2106.
[6] Abrishami S, Naghibzadeh M, Epema D. Deadline constrained workflow scheduling algorithms for iaas clouds[J]. Future Generation Computer Sytems, 2013, 29(1):158-169.
[7] Zhu Meng-xia, Wu Qi-shi, Zhao Yang. A cost effective scheduling algorithm for scientific workflows in clouds[C] //Proc of the 31st IEEE International Performance Computing and Communications Conference (IPCCC), 2013: 256-265.
[8] Vahid Arabnejad, Kris Bubendorfer, Bryan Ng. Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources[J]. Future Generation Computer Systems , 2017, 75:348-364.
[9] Cai Zhi-cheng. Resource provisioning methods for cloud workflow applications[D]. Nanjing City: Southeast University, 2015. (in Chinese)
[10] Li Xiao-ping, Cai Zhi-cheng. Elastic resource provisioning for cloud workflow applications[J]. IEEE Transactions on Automation Science & Engineering, 2017, 14(2):1-16.