symmetry

MDPI

Article

Dynamic Job Scheduling Strategy Using Jobs Characteristics in Cloud Computing

Mohammed A. Alsaih [1,*,†], Rohaya Latip [1,*,†], Azizol Abdullah [1], Shamala K. Subramaniam [1] and Kamal Ali Alezabi [2]

1 Department of Communication Technology and Networking, Faculty of Computer Science, University Putra Malaysia, Selangor 43400, Malaysia; azizol@upm.edu.my (A.A.); shamala_ks@upm.edu.my (S.K.S.)
2 Institute of Computer Science and Digital Innovation, UCSI University, Cheras, Kuala Lumpur 56000, Malaysia; kamal@ucsiuniversity.edu.my
* Correspondence: mohammedalsaih@yahoo.com (M.A.A.); rohayalt@upm.edu.my (R.L.)
† These authors contributed equally to this work.

Received: 23 August 2020; Accepted: 12 September 2020; Published: 6 October 2020

Abstract: A crucial performance concern in distributed decentralized environments, like clouds, is how to guarantee that jobs complete their execution within the estimated completion times using the available resources' bandwidth fairly and efficiently while considering the resource performance variations. Formerly, several models including reservation, migration, and replication heuristics have been implemented to solve this concern under a variety of scheduling techniques; however, they have some undetermined obstacles. This paper proposes a dynamic job scheduling model (DTSCA) that uses job characteristics to map them to resources with minimum execution time taking into account utilizing the available resources bandwidth fairly to satisfy the cloud users quality of service (QoS) requirements and utilize the providers' resources efficiently. The scheduling algorithm makes use of job characteristics (length, expected execution time, expected bandwidth) with regards to available symmetrical and non-symmetrical resources characteristics (CPU, memory, and available bandwidth). This scheduling strategy is based on generating an expectation value for each job that is proportional to how these job's characteristics are related to all other jobs in total. That should make their virtual machine choice closer to their expectation, thus fairer. It also builds a feedback method which deals with reallocation of failed jobs that do not meet the mapping criteria.

Keywords: job scheduling; reallocation; virtual machines; job characteristics; QoS; cloud computing

1. Introduction

Recently, cloud computing has turned out to be one of the most capable distributed computing models [1,2]. A job scheduling scheme is one of the essential and challenging concerns in a cloud computing structure [3]. Clouds are on-demand computing platforms where usually there are a big number of "centralized" resources, for instance, storage and processors devices. The key advantages of cloud computing systems over other distributed computing systems, mainly grids, are scalability, reliability, flexibility and sustainability [4]. Alternatively, a set of cloud resources is employed by the users (i.e., dedicated and provided to a user) when required in a resource-as-a-service manner. A main method that facilitates the recognition of clouds is virtualization. This technology helps the vendors of cloud services to supply virtually any kind of computing scheme, eliminating the complexity and burden of sustaining varied sets of both software and hardware mechanisms to assemble users' requirements. Broadly implemented virtualization technologies incorporate Xen [5], VirtualBox [6], and VMWare Server [7]. Contrasting of most grids, clouds are possessed by commercial IT service vendors (e.g., GoGrid, Amazon, and Google) and controlled by a pay-per-use or subscription-based dealing structure. For instance, Amazon's Elastic Compute Cloud (EC2) rents its computers in its cloud to users on an hourly basis [8]. In cloud computing, conventional job scheduling methods merely consider how to increase the efficiency of job scheduling or how to guarantee the required QoS for the users; they hardly ever offer a scheme that reflects on mixing both of these two features simultaneously. Quite a few job scheduling techniques have been implemented at this point as indications in cloud computing areas [9–16].

The majority of them could be categorized in the environment of clouds by highlighting efficiency [10,12–17]. Parts of them contemplate on the fairness between users heavily in a way that would affect the efficiency [9,12]. Cloud resources, or partly those allocated to a specific user, are regularly homogeneous, reliable, and tightly coupled. Additionally, the characteristics, the number of resources, and the types to be reserved can be assorted to the requests of users. These features appoint the resources of clouds as an excellent environment that is suitable for uncertainty-handling systems. On the other hand, because of financial costs coupled with the exploit of cloud resources, some of these systems are not completely convenient.

Particularly, advanced reservation is not proper to transact with performance variations revealed from cloud resources. Migration could be a superior option, except it is not simply incorporated into heterogeneous computing environments like clouds, and it heads for engaging in time consuming and complex processes like check-pointing. Replication is an unfeasible alternative as it tends to use resources redundantly for its breaded replicas. This leads us to focus on reallocation as an acceptable secondary support technique for scheduling mechanisms in a cloud computing environment. Dynamic scheduling algorithms are constantly referred to as making one of the best current choices in problem solving, which can be described as it generates the optimal solution but locally. A dynamic algorithm cannot only be the overall optimal solution for all problems, but for a wide range of many of the problems it could produce an optimal solution or a near-optimal one. In this paper, we propose a dynamic algorithm in a cloud environment. This algorithm contains three stages; the stage of classification based on QoS, as shown in Figure 1. We classify jobs in the first stage into two classes of jobs; the completion time jobs (CT) and the bandwidth jobs(BW). Then, we assign a general expectation value for each job based on their characteristics related to all other jobs in the same class (e.g., job length, job expected execution time, jobs expected bandwidth). For resources, different parameters are also considered, such as CPU, memory, available bandwidth. Then, the priority value is assigned to each job at each type and jobs will be mapped to a specific resource using the scheduling strategy. After that, the fairness justice evaluation will be calculated by a function we call Justice Evaluation Function "JEF" [9,12,17], hence we can get the final results about fairness. Finally, we add a simple method to deal with the jobs that do not meet the mapping criteria, which in this case, will adjust the job's general expectation value according to that jobs characteristics and available resources characteristics to prepare it for reallocation stage. A substance enhancement in bandwidth and CPU utilization is realized. The experimental results demonstrate a substantial decrease in the jobs' completion time within an increase in users' satisfaction.

2. Related Work

Job scheduling in the cloud and grid has a tremendous narration in research. A bunch of algorithms, techniques and strategies have been proposed for this, which include the Berger model [9], Bags-of-Tasks [10], Optimal Workflow Scheduling [17], Multi-QoS Cost-Based Scheduling [11], Utility-Based Scheduling [15], Priority Scheduling [12,13]. Another priority scheduling algorithm has been proposed in [14], and it employs a job scheduling algorithm according to the quality of service (QoS) parameters, for example average execution, load balancing, and makespan. Initially, depending on the QoS, they compute the priorities of the jobs, and then jobs with higher priority are mapped first on the Vm. However, there is no feedback strategy to adjust the expectation vector of the Jobs; besides, it consumes a lot of time to search for multiple values of multiple matrices that represent different properties of the jobs.

The work in [9] demonstrates a job scheduling strategy founded on the Berger model of the social theories of distributive justice. This framework creates a "job scheduling strategy that employees dual fairness constraints. It gives the jobs an expected state and compares it with an actual state". The simulation result illustrates a positive enhancement on users' fairness to a considerable degree with a deficiency in efficiency. This scheduler uses this algorithm to establish dual fairness constraints. The first one is classifying users based on QoS preferences, then using the general expectation function with job classification to restrain the resources fairness in the process of selection. The second one is to define a function for resource fairness justice to judge the resource fairness allocation.

The algorithm in [10] proposes a scheduling technique that deals with various applications sorted as a group of identical and independent jobs on a heterogeneous master-client policy. The goal is to reduce the maximum gap between the actual time an application has exploited in the system and the time this application would have exploited if performed singlehandedly. An Optimal Workflow based Scheduling (OWS) method [17] was implemented to get a resolution that satisfies the user-preferred parameters of Quality of Service (QoS). This work concentrates on scheduling several types of workflows in clouds. Exploiting this manner substantially enhanced the utilization of the CPU resources. Different methods have been adopted in [18] to handle the jobs and processes for the same or different systems. Those processes or jobs are running simultaneously in the web application/server. A new scheduling strategy called Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW) has been introduced in [19]. The QoS is taken into account in this scheduling strategy when many workflows are scheduled. In [20], a job scheduling system based on an M/G/1 queuing model was presented after analyzing the QoS requirements of resources and users' jobs in cloud computing systems to effectively use those resources.

From the user viewpoint and to meet user requirements on time and with low cost, another job scheduling based on QoS has been proposed in [21], where two factors have been considered; completion time and cost. However, this scheduling system does not take the cost of the queue of waiting jobs into account. Different scheduling policies such as delay or FIFO scheduling in the fair scheduler have been used in [22], where the problems of job scheduling have been discussed due its importance in the cloud computing. Unlike the traditional job scheduling, the work in [23] has been presented by a job scheduler to schedule job groups. The proposed scheduler is based on job cost and computation performance since each job differs from the other in cloud resources. The work in [24] presented a job scheduler that is used by the Independent Software Vendors (ISV) for software distribution in cloud computing. A smart scheduler based on Quality of Service (QoS) was implemented in [25]. This scheduler uses a backfill strategy based lightweight Virtual Machine Scheduler to dispatch jobs, and better performance and optimal resource utilization were the goals of this work.

To get an overview and basic concepts of cloud computing and as a good foundation for understanding this technology, the reader can refer to [26,27]. The services of cloud computing [28] that are provided to remote users who have different requirements have been described; in addition, they present a scheduling approach that showed which workloads consist of workflows or queries. On the other hand, several studies have focused on particle swarm workflows scheduling and optimization [29,30]. Different parallel job scheduling strategies have been proposed in [31,32] to optimize the mapping between jobs and resources. Those methods report different strategies used in these fields. The Backfilling Strategy [33] is an extension of FCFS, which provides a fair resource allocation. This strategy solves the problems of waiting time and starvation by using two algorithms, conservative backfilling and an Extensible Argonne Scheduling System (EASY) but it does not consider priority in scheduling the same as conservative backfilling, and it has less fairness. Backfilling performance is like cutting, sometimes called "queue jumping", which violates the term of social justice [34,35].

A greedy scheduling algorithm is proposed in [36] to enhance the work in the Berger model and the result showed a better fairness via reducing the job execution time. For the sake of scheduling jobs based on their priority, the authors in [37] have proposed a workflow scheduling strategy. The scheduler calculates the jobs priority when the scheduling begins and ends. To define fairness, the authors in [34] present two metrics; the first one is fair start time, which is the early start time, and the second one is fair share of resources, which means what fair share is, is the main concern in this work. The justice evaluation function can be used to estimate the fairness [9].

Recently, the authors in [38] proposed a new job scheduling method. The cellular automata technique has been employed to carry out the scheduling mechanism. This method considers the cost of jobs and the hardness factor, it aims to increase the profitability of the service providers of the cloud, where more jobs are executed in the specified deadline.

In addition, the authors in [39] have presented a cloud job scheduling using the concept of fuzzy logic to attain the fairness of the resource allocation consistent with the quality of service. The projected tactic solves two categories of issues, the first issue is how to choose the best Vms to bind the job, whereas, the second issue employs a static justification of the job using assumption values. This procedure displays an improved mathematical connotation among the parameters to adjust the expectation vector of the sorted jobs. However, using such assumptions statically and not considering the jobs characteristics might affect the jobs mapping to Vms, hence affecting the fairness Judgment.

The proposed DTSCA algorithm uses dynamic calculation as the initial values of users' jobs expectation vector to realize the user's satisfaction. In addition, it sorts and classifies both users' jobs and available Vms according to users' jobs types and appropriate Vms. Furthermore, it adds a dynamic feedback and reallocation methods that adjust the expectation jobs vector dynamically and make sure all jobs are mapped to the most preferred Vm.

3. Proposed Model

3.1. Job Classification

Assigning a general expectation vector for a type of job is used in the Berger model and proved to be inefficient. In this work, each job has a general expectation value based on their characteristics related to all other jobs, which are all collected in a general expectation values list (JEV). The proposed scheduling algorithm primarily contains three stages and includes the stage of classification based on QoS as shown in Figure 1. We classify jobs in the first stages into two classes of jobs; the completion time jobs (CT) and the bandwidth jobs (BW). We then assign a general expectation value for each job based on their characteristics related to all other jobs in the same class, for example, job length, job expected execution time and jobs expected bandwidth. Also, for resources, different parameters are considered, such as CPU, memory and available bandwidth. Each job will then be mapped to a specific resource using the scheduling strategy.

Justice evaluation will be calculated by a function we call Justice Evaluation Function (JEF) [9], hence we can get the final results about fairness. Finally, we add a simple method to deal with the jobs that do not meet the mapping criteria, which in this case will prepare that job according to their characteristics and available resources characteristics to prepare it for reallocation stage. A fixed amount and types of execution times have been assumed by researches for each job, but in the real world of cloud computing it is not the case. When jobs are submitted by the users in the job classification part, it will dynamically compute the attributes of each job and QoS properties, then it calculates the initial expectation vector for each job in each class of the jobs types, and after that it places these values in the job attributes vector, which includes the following:

- **Class-Type:** the job class type can be either time type or bandwidth type.
- **Start-Time:** the starting time for each job.
- **Finish-Time:** the time needed for a job to complete execution.
- **Expected-Time:** the job's expected finish time.

- **Expected-Bandwidth:** the expected bandwidth needed for each job.
- **Priority:** the priority value assigned to each job
- **J-Value:** the justice evaluation value for each job amongst other jobs (at the same job type).

Afterward, the jobs-classifier categorizes the job based on attributes and initial values of QoS (completion time and bandwidth). Then, it assigns a priority value for each available virtual machine Vm. Finally, it sends jobs to different scheduler sections, which will then send these jobs to the most appropriate available resource for execution (virtual machines).

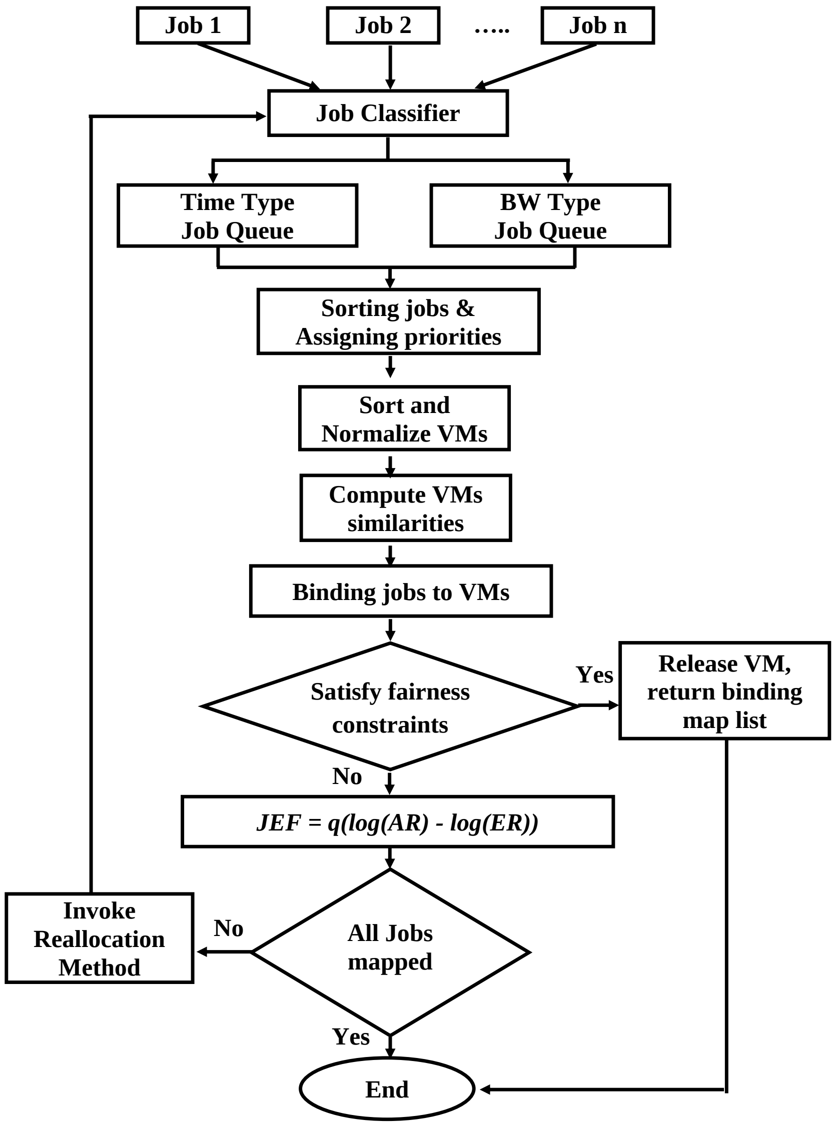

Figure 1. Scheduling and reallocation model.

3.2. Jobs and Resources Description

In the cloud computing, virtualization is exploited to allocate host resources to the layer of virtual machines. Scheduling is to map jobs to resources using assured optimization rules. Suppose the attributes of the resource set of virtual machines Vm_i is:

$$A_{ti} = (A_{i1}, A_{i2}, ..., A_{in}), n = 1, 2, 3 \tag{1}$$

where A_{i1}, A_{i2}, A_{i3} represents CPU, memory and bandwidth, respectively. The vector of performance of Vm_i is:

$$Vm_i = [ET_{i1}, ET_{i2}, ET_{i3}] \tag{2}$$

where ET_i is the related performance value of attributes ET_i of Vm_i. In order to establish fair resources selection, the jobs general expectation function is established. This function uses a variety of specifications of users QoS jobs and adjusts the ratio of performance of the selected virtual machines accordingly. This expectation function can be represented as:

$$e_r = e_{r1}, e_{r2}, e_{r3} \tag{3}$$

and

$$\sum e_{rj} = 1 \tag{4}$$

3.3. Generation of Expectation Initialization and Priority

In this phase, we develop a method that generates an expectation value for each job at each job type (time-type jobs and bandwidth-type jobs), which is proportional to how a job's characteristics are related to other jobs in total; this method is called "Atomization", according to the following formula:

$$J_A = (J_{Ai}/sum(J_{An})) \tag{5}$$

where J_A represents the job length in MIPS for time-type jobs and job-required bandwidth for the bandwidth-type jobs. For example, job 4 in Table 1 would have an expected bandwidth value of 0.2439 as 0.2439 = 2000/(2000 + 3000 + 1200 + 2000). Priority then will be assigned to each job of each type according to the atomization process and jobs of each type will be sorted in an ascending manner.

3.4. Scheduling Algorithm

After the jobs are classified into their corresponding types, the priority lists of each type of them are created, the initial expectation value of each job in each type has been generated using the atomization method, and the available resources are collected and sorted in a descending manner in a virtual machines list Vm_i List, and the jobs mapping to resources start by selecting a virtual machine from Vm_i according to the priority assigned to the job using the general expectation value initialization. In this work, the resource selection process employs the general expectations to achieve the fairness constraints using the justice evaluation function (JEF) as follows:

$$JEF = \theta(log(AR) - log(ER)) \tag{6}$$

where denotes the constant. $0 < \theta < 1$, AR (Actual Resource) is the sum of resources that job J_i attains after mapping. ER (Expected Resource) is the sum of resources that job J_i expected to acquire. When AR equals ER; JEF = 0, thus the fairness is achieved. Each job J_i can be mapped to any V_m in the available Vm_i list as long as its execution requirements do not exceed the number of available Vm_i.

Then, normalization collects the virtual machines performance parameters into a list called p[i] to [0, 1], in order to carry on the comparison with the vector of general expectations called $e_r[i]$. Assuming that M = M_1,..., M_j, j = 1, 2, 3 is the corresponding set of performance parameters for Vm_i. The normalized value is calculated as follows:

$$NP_{ij} = ((pCur - pMin)/(pMax - pMin)) \tag{7}$$

where *pCur* is the performance parameters of the current value, *pMin* is the performance parameters minimum value and *pMax* is the performance parameters maximum value. After that, the resulted performance parameters list NP_{ij} of Vm_i can be used to match the expectations function of job e_r (Equation (4)).

Then, the similarity between the general expectation function $e_r[i]$ and the actual allocation vector of normalized Vm_i parameters NP_{ij} is used to restrain the convergence to produce the maximum similarity of both vectors. The similarity between the initial general expectation vector and the performance vector can be calculated via the following formula: Compute the similarity by equation

$$S = (1 - (NP[i] * e[i]) / \sqrt{((NP[i] * e[i])^2 * (NP[i] * e[i]))^2)} \tag{8}$$

Finally, the jobs are mapped to the appropriate virtual machine in an iterative manner until all job mapping is completed. If one of these jobs invalidate the previous constraints the initial general expectation value will be adjusted as in Equation (5) and the job will be mapped again. In the last round, the algorithm will check whether all jobs are mapped or not, if not, the reallocation method is invoked. A description of the model including the job scheduling algorithm and the reallocation method can be found in Algorithms 1 and 2, respectively.

Algorithm 1: Jobs scheduling algorithm.

```
begin while Submitted Jobs list<>0 do
    Calculate initial GEV for both types of jobs using;
    a. List of $A_{i1}$
    b. List of $A_{i2}$
    c. List of $A_{i3}$
    Atomize($A_{i1}, A_{i2}, A_{i3}$) // Time-Type-Jobs, using Equation (5)
    Atomize($A_{i1}, A_{i2}, A_{i3}$) // Bandwidth-Type-Jobs, using Equation (5)
    Assign priority to each job on each type.
    Produce a candidate collection of Vmi.
    Sort available Vms in an Ascending manner.
    Select a Vm from Vmi according to the GEV preference. using Equation (6)
    Normalize the Vm performance parameters to [0, 1]. using Equation (7)
    Compute the similarity between erj and NPij. using Equation (8)
    Bind job to Vm with higher similarity.
    if binding satisfies fairness constraints then
        a.Return binding map list;
        b.Release Vm.
    else
        Adjust the initial GEV (go to Calculate initial GEV)
    end
end
for all submitted jobs do
    if all jobs are mapped then
        return true
    else
        call Reallocation method.
    end
end
Exit ().
```

Algorithm 2: Reallocation method.

```
begin Retrieve the failed jobs from each type
for all failed bandwidth type jobs do
    Sort jobs according to their bandwidth in an ascending manner.
    Pick the job that has the largest length (hence needs the largest available Vm's BW)
    Sort Vms according to their remaining available bandwidth in an ascending style.
    Select the 1st Vm in the sorted bwVmlist and mark it as preferred Vm.
    if preferred Vm available remaining bandwidth matches the BW required to execute the largest job then
        Map that job to that Vm
    else
        Move the job to the next Vm in the preferredVmlist
    end
end
for all failed time type jobs do
    Sort jobs according to their length in a descending manner.
    Pick the job that has the smallest length (hence needs the smallest available Vm MIPS)
    Sort Vms according to their remaining MIPS in an ascending style.
    Select the 1st Vm in the sorted timeVmlist and mark it as preferred Vm.
    if preferred Vm available remaining capacity matches the MIPS required to execute the largest job then
        Map that job to that Vm
    else
        Divide them in half and move to the next preferred Vm
    end
end
Return the new scheduled list for each job type.
```

4. Simulation and Results

We extended the CloudSim platform simulator [1] to implement the DTSCA Algorithm. Here, after jobs classification, and according to the job's general expectation vector initialization and priority assignment, the priority and the initial value of the general expectation vector of each job is generated and assigned dynamically using the atomization phase of the DTSCA algorithm, which will generate more sufficient initial values that are better related to user job's characteristics and cloud available resources properties—unlike the Berger model and fuzzy logic algorithms that applied the assumption concept and used an astatic initial value of general expectation vector and generated job's priorities according to them.

Moreover, the available resources were sorted and classified according to their characteristics (CPU, memory and bandwidth) in order to make the jobs binding to them faster and more efferent for both time-jobs-type and bandwidth-type jobs. Furthermore, the reallocation method in DTSCA is combined with the scheduling algorithm to guarantee that all jobs are mapped successfully to the resources in a manner that satisfies both job type restrictions and available resource specifications, which led to better resource utilization and better users' satisfaction.

The description of jobs' parameters are listed in Section 3.1 and the values are illustrated in Table 1. In addition, the resources (virtual machines) parameters used in this experiment are listed in Table 2. The simulation results are depicted in Figures 2–5. To validate the proposed DTSCA algorithm, its results are compared with the other two scheduling algorithms, which are named the job scheduling algorithm based on the Berger model [9] and the fuzzy logic algorithm [39] using the CloudSim simulator.

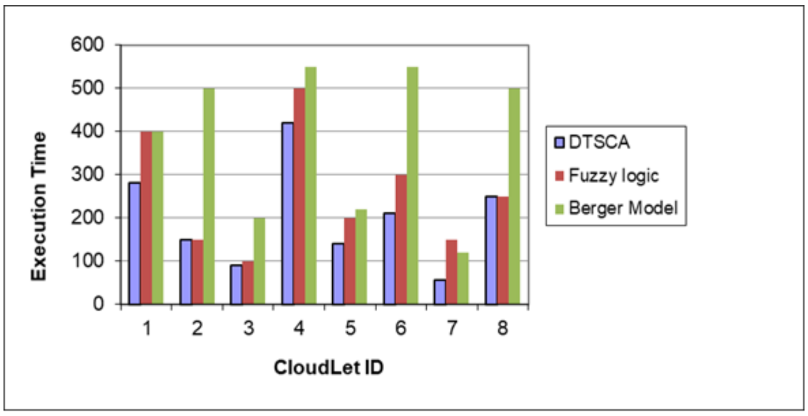

Figure 2. Jobs execution time comparison.

Figure 2 shows that the job execution time in the DTSCA algorithm is better than both other algorithms (the Berger model algorithm and fuzzy logic algorithm). Generally, the efficiency of the jobs execution of the DTSCA algorithm is better than the efficiency in both other algorithms since it reduced the jobs required execution time. The completion time of the DTSCA algorithm is better than the completion time in both algorithms.

Table 1. Jobs parameters.

Cloudlet ID	Class Type	Length	File Size	Output Size	Expected Time	Expected Bandwidth
0	1	4000	2500	500	400	-
1	1	3000	2000	400	200	-
2	1	2000	800	300	150	-
3	1	5000	5000	2000	500	-
4	2	2000	800	300	-	2000
5	2	3000	2000	400	-	3000
6	2	800	300	300	-	1200
7	2	2500	1000	500	-	2000

Table 2. Vm parameters.

VmId	CPU	Memory	Bandwidth
0	4	2048	2000
1	2	1024	3000
2	2	1024	1200
3	1	512	2500

Figure 3 shows the justice values resulted from the comparative experiments. In terms of meeting user expectations, DTSCA is more efficient compared to other algorithms since the user satisfaction is better when the J value approaches zero (total fairness achieved when the J value of the job = 0).

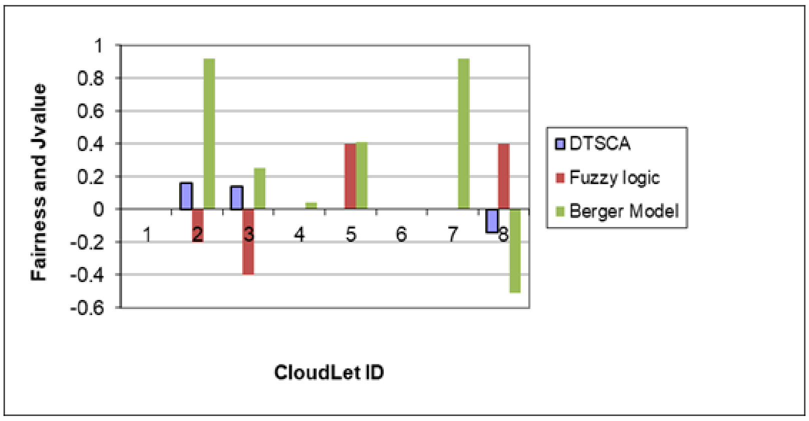

Figure 3. Fairness comparison.

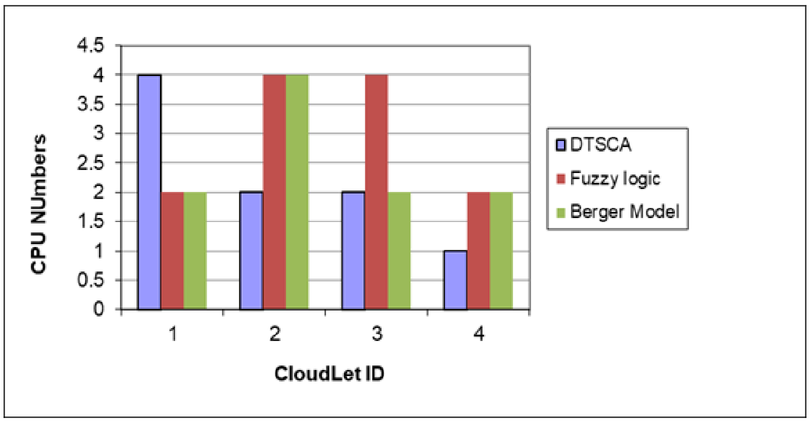

Figure 4. Comparison of first class of jobs.

Figure 4 depicts the first type of job resulted from comparative experiments. DTSCA and fuzzy logic algorithms enable job 1–4 (Time-Type-Jobs) to obtain good computing power (CPU utilization) with better fairness, while the Berger algorithm depicts poorer results.

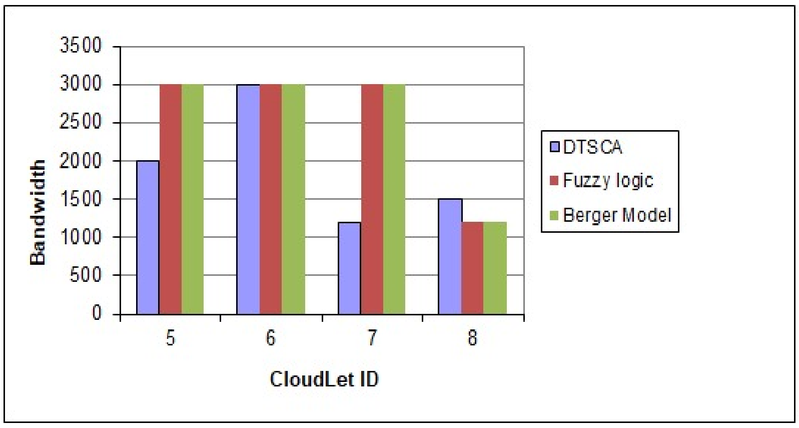

Figure 5. Comparison of second class of jobs.

Figure 5 shows the comparison between the allocated virtual machines bandwidth for the second type of job (Bandwidth-Type-Jobs). DTSCA enables jobs 5–8 to consume almost the same expected amount of BW, which indicates a better fairness since bandwidth for each job equals the expected bandwidth for each one of them with a slight difference for job 7.

5. Conclusions and Future Work

In this work, a dynamic job scheduling algorithm using users' jobs characteristics is developed to deal with jobs mapping to the most appropriate available resources in a manner that meets users QoS expectations and at the same time achieves resource utilization through scheduling and reallocation methods. The dynamic job scheduling algorithm DTSCA has been implemented and validated by extending the CloudSim simulator and compared the results of the simulation with the job scheduling algorithm based on the Berger model and fuzzy logic-based job scheduling algorithm. In addition, the proposed DTSCA algorithm has been proved as an effective strategy that achieved better completion time, better resource utilization, and better user satisfaction. Moreover, DTSCA achieved better performance in terms of fairness for the whole system in the cloud computing environment. Furthermore, as future work, a non-linear probability mapping strategy between jobs' QoS requirements and resource characteristics will be considered to obtain an optimized and fairer scheduling model. Another future work direction is improving the proposed algorithm to be applicable to the Internet of Things (IoT) technology.

Author Contributions: Conceptualization, M.A.A.; methodology, M.A.A., R.L.; software, M.A.A., R.L.; validation, M.A.A., R.L., A.A. and S.K.S.; formal analysis, M.A.A., R.L.; investigation, M.A.A., R.L., A.A. and S.K.S.; resources, M.A.A., R.L. and K.A.A.; data curation, M.A.A., R.L.; writing—original draft preparation, M.A.A.; writing and editing, M.A.A., R.L. and K.A.A.; visualization, M.A.A., R.L., K.A.A.; supervision, R.L.; project administration, R.L.; review, M.A.A., R.L., A.A., S.K.S. and K.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was supported by the Universiti Putra Malaysia.

Acknowledgments: The authors would like to thank and acknowledge the financial support and facilities provided by Universiti Putra Malaysia and Ministry of Education Malaysia (MOE) for the execution, completion and publication of this paper.

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

References

1. Buyya, R.; Ranjan, R.; Calheiros, R.N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In Proceedings of the 2009 International Conference on High Performance Computing & Simulation, Leipzig, Germany, 21–24 June 2009; pp. 1–11.
2. Hewitt, C. ORGs for scalable, robust, privacy-friendly client cloud computing. *IEEE Internet Comput.* **2008**, *12*, 96–99. [CrossRef]
3. Arunarani, A.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
4. Radu, L.D. Green cloud computing: A literature survey. *Symmetry* **2017**, *9*, 295. [CrossRef]
5. Xen Project. Available online: https://xenproject.org/2015/01/14/xen-project-announces-4-5-release/ (accessed on 20 January 2020).
6. Virtual Box. Available online: https://www.virtualbox.org/ (accessed on 10 May 2020).
7. VMware Server. Available online: https://www.vmware.com/ (accessed on 4 August 2020).
8. Amazon Elastic Compute Cloud. Available online: https://aws.amazon.com/ec2 (accessed on 21 August 2020).
9. Xu, B.; Zhao, C.; Hu, E.; Hu, B. Job scheduling algorithm based on Berger model in cloud environment. *Adv. Eng. Softw.* **2011**, *42*, 419–425. [CrossRef]
10. Benoit, A.; Marchal, L.; Pineau, J.F.; Robert, Y.; Vivien, F. Offline and online master-worker scheduling of concurrent bags-of-tasks on heterogeneous platforms. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–8.
11. Dutta, D.; Joshi, R. A genetic: Algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology, Mumbai, India, 25–26 February 2011; pp. 422–427.
12. Ghanbari, S.; Othman, M. A priority based job scheduling algorithm in cloud computing. *Procedia Eng.* **2012**, *50*, 778–785.

13. Potluri, S.; Rao, K.S. Quality of service based task scheduling algorithms in cloud computing. *Int. J. Electr. Comput. Eng.* **2017**, *7*, 1088. [CrossRef]
14. Wu, X.; Deng, M.; Zhang, R.; Zeng, B.; Zhou, S. A task scheduling algorithm based on QoS-driven in cloud computing. *Procedia Comput. Sci.* **2013**, *17*, 1162–1169. [CrossRef]
15. Yang, B.; Xu, X.; Tan, F.; Park, D.H. An utility-based job scheduling algorithm for cloud computing considering reliability factor. In Proceedings of the 2011 International Conference on Cloud and Service Computing, Hong Kong, China, 12–14 December 2011; pp. 95–102.
16. Chang, R.S.; Lin, C.Y.; Lin, C.F. An adaptive scoring job scheduling algorithm for grid computing. *Inf. Sci.* **2012**, *207*, 79–89. [CrossRef]
17. Varalakshmi, P.; Ramaswamy, A.; Balasubramanian, A.; Vijaykumar, P. An optimal workflow based scheduling and resource allocation in cloud. In Proceedings of the International Conference on Advances in Computing and Communications, Kochi, India, 22–24 July 2011; pp. 411–420.
18. Gupta, P.K.; Rakesh, N. Different job scheduling methodologies for web application and web server in a cloud computing environment. In Proceedings of the 2010 3rd International Conference on Emerging Trends in Engineering and Technology, Goa, India, 19–21 November 2010; pp. 569–572.
19. Xu, M.; Cui, L.; Wang, H.; Bi, Y. A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, China, 10–12 August 2009; pp. 629–634.
20. Li, L. An optimistic differentiated service job scheduling system for cloud computing service users and providers. In Proceedings of the 2009 Third international conference on Multimedia and Ubiquitous Engineering, Qingdao, China, 4–6 June 2009; pp. 295–299.
21. Huang, Q.Y.; Huang, T.l. An optimistic job scheduling strategy based on QoS for cloud computing. In Proceedings of the 2010 International Conference on Intelligent Computing and Integrated Systems, Guilin, China, 22–24 October 2010; pp. 673–675.
22. Ge, Y.; Wei, G. GA-based task scheduler for the cloud computing systems. In Proceedings of the 2010 International Conference on Web Information Systems and Mining, Sanya, China, 23–24 October 2010; Volume 2, pp. 181–186.
23. Selvarani, S.; Sadhasivam, G.S. Improved cost-based algorithm for task scheduling in cloud computing. In Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 28–29 December 2010; pp. 1–5.
24. Ejarque, J.; Micsik, A.; Sirvent, R.; Pallinger, P.; Kovacs, L.; Badia, R.M. Job scheduling with license reservation: A semantic approach. In Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, Ayia Napa, Cyprus, 9–11 February 2011; pp. 47–54.
25. Guin, R.B.; Chakrabarti, S.; Tarafdar, C. *Modelling & Simulation of a Smarter Job Scheduling System for Cloud Computing Service Providers and Users*; Computer Science & Engineering Department Kalyani Government Engineering College: Kalyani, India, 2011.
26. Kalapatapu, A.; Sarkar, M. Cloud computing: An overview. *Cloud Computing: Methodology, Systems, and Applications*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 3–29. [CrossRef]
27. Varghese, B.; Buyya, R. Next generation cloud computing: New trends and research directions. *Future Gener. Comput. Syst.* **2018**, *79*, 849–861. [CrossRef]
28. Paton, N.; De Aragão, M.A.; Lee, K.; Fernandes, A.A.; Sakellariou, R. Optimizing utility in cloud computing through autonomic workload execution. *Bull. Tech. Comm. Data Eng.* **2009**, *32*, 51–58.
29. Farid, M.; Latip, R.; Hussin, M.; Abdul Hamid, N.A.W. A Survey on QoS Requirements Based on Particle Swarm Optimization Scheduling Techniques for Workflow Scheduling in Cloud Computing. *Symmetry* **2020**, *12*, 551. [CrossRef]
30. Ambursa, F.U.; Latip, R.; Abdullah, A.; Subramaniam, S. A particle swarm optimization and min–max-based workflow scheduling algorithm with QoS satisfaction for service-oriented grids. *J. Supercomput.* **2017**, *73*, 2018–2051. [CrossRef]
31. Feitelson, D.G.; Rudolph, L.; Schwiegelshohn, U. Parallel job scheduling—A status report. In *Workshop on Job Scheduling Strategies for Parallel Processing*; Springer: New York, NY, USA, 2004; pp. 1–16.
32. Liu, X.; Zha, Y.; Yin, Q.; Peng, Y.; Qin, L. Scheduling parallel jobs with tentative runs and consolidation in the cloud. *J. Syst. Softw.* **2015**, *104*, 141–151. [CrossRef]

33. Baraglia, R.; Capannini, G.; Pasquali, M.; Puppin, D.; Ricci, L.; Techiouba, A.D. Backfilling strategies for scheduling streams of jobs on computational farms. In *Making Grids Work*; Springer: New York, NY, USA, 2008; pp.103–115.
34. Raz, D.; Avi-itzhak, B.; Levy, H.; Levy, H. *Fairness Considerations in Multi-Server and Multi-Queue Systems*; RUTCOR: Piscataway, NJ, USA; Rutgers University: New Brunswick, NJ, USA; Citeseer: Princeton, NJ, USA, 2005.
35. Jasso, G. The theory of the distributive-justice force in human affairs: Analyzing the three central questions. In *Sociological Theories in Progress: New Formulations*; Sage: Newbury Park, CA, USA, 1989; pp. 354–387.
36. Li, J.; Feng, L.; Fang, S. An greedy-based job scheduling algorithm in cloud computing. *JSW* **2014**, *9*, 921–925. [CrossRef]
37. Jayadivya, S.; Bhanu, S.M.S. Qos based scheduling of workflows in cloud computing. *Int. J. Comput. Sci. Electr. Eng.* **2012**, *1*, 2315–4209.
38. Zekrizadeh, N.; Khademzadeh, A.; Hosseinzadeh, M. An Online Cost-Based Job Scheduling Method by Cellular Automata in Cloud Computing Environment. *Wirel. Pers. Commun.* **2019**, *105*, 913–939. [CrossRef]
39. Pandey, P.; Singh, S. Fuzzy logic based job scheduling algorithm in cloud environment. *Comput. Model New Technol.* **2017**, *21*, 25–30.