

QoS-Constrained, Reliable and Energy-Efficient Task Deployment in Cloud Computing

Zhenghui Zhang, Yuqi Fan

School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui, 230000, China

Email: zhenghuizhang@mail.hfut.edu.cn

Abstract

Reliability, QoS and energy consumption are three important concerns of cloud service providers. Most of the current research on reliable task deployment in cloud computing focuses on only one or two of the three concerns. However, these three factors have intrinsic trade-off relationships. The existing studies show that load concentration can reduce the number of servers and hence save energy. In this paper, we deal with the problem of reliable task deployment in data centers, with the goal of minimizing the number of servers used in cloud data centers under the constraint that the job execution deadline can be met upon single server failure. We propose a QoS-Constrained, Reliable and Energy-efficient task replica deployment (QSRE) algorithm for the problem by combining task replication and re-execution. For each task in a job that cannot finish executing by re-execution within deadline, we initiate two replicas for the task: main task and task replica. Each main task runs on an individual server. The associated task replica is deployed on a backup server and completes part of the whole task load before the main task failure. Different from the main tasks, multiple task replicas can be allocated to the same backup server to reduce the energy consumption of cloud data centers by minimizing the number of servers required for running the task replicas. Specifically, QSRE assigns the task replicas with the longest and the shortest execution time to the backup servers in turn, such that the task replicas can meet the QoS-specified job execution deadline under the main task failure. We conduct experiments through simulations. The experimental results show that QSRE can effectively reduce the number of servers used, while ensuring the reliability and QoS of job execution.

Keywords: *Cloud Computing; Task Deployment; Reliability; Quality of Service; Energy Consumption*

1 INTRODUCTION

Cloud computing has gradually become an indispensable computing model for various scientific, commercial and data-intensive applications. With the accelerated development of cloud computing services, the infrastructure scale of cloud data centers expands, and the system architecture becomes increasingly complex. At the same time, the uncertainty of various resources is rapidly increasing, which results in frequent failures of the system during operation, with 100,000 processors experiencing a failure every couple of minutes^[1]. A cloud service provider usually signs a Service Level Agreement (SLA) with the users in advance, specifying the quality of service (QoS) provided to the users. However, failures of cloud computing systems prolong the job completion time, which will result in significant losses for the cloud service providers^[2]. In a survey of 63 data centers, the average down-time cost of each data center rose from \$500,000 in 2010 to \$740,357 in 2020^[3].

Task replication is a viable way to provide fault tolerance of job execution in cloud data centers at the cost of using more servers, which reduces the efficiency of computing resources utilization and increases the energy consumption of cloud data centers. Garraghan et al. presented a failure analysis on a large-scale heterogeneous cloud environment, and the results show that the re-submissions of jobs caused by crash failures of tasks and servers will lead to a significant waste of computing resources^[4]. Therefore, it is important to study how to improve energy efficiency of cloud data

centers while ensuring the reliability of cloud computing systems^[5]. Task deployment strategies significantly affect the reliability, QoS and energy efficiency of cloud computing systems. Current task deployment strategies usually focus on one or two of these three objectives^[6,7,8]. However, these three objectives have intrinsic trade-off relationships. Accordingly, there is an urgent need for effective task deployment to improve the energy efficiency while guaranteeing the reliability and QoS of job execution in cloud data centers.

The existing studies show that load concentration can reduce the number of servers and hence effectively save energy^[9]. Load concentration employs an energy-aware strategy that concentrates task loads on as few servers as possible, such that some servers are idle and can be shut down to reduce energy consumption. Chase et al. validated this strategy in a network environment and their experimental results showed that the load concentration technique can achieve energy savings of more than 29%^[10].

The scenario of single server failure has been attracting much attention from industries and academia^[11,12,13], since typical failure modes are single server failures^[14,15]. Therefore, we consider the scenario of single server failure. In this paper, we combine task replication and re-execution to provide fault tolerance and ensure QoS through an efficient task deployment strategy with the objective of minimizing the number of servers used. The contributions of this paper are as follows:

- (1) We study the problem of reliable task deployment in cloud data centers to minimize the number of servers required for task execution to reduce the energy consumption of cloud data centers, under the constraint that the job execution deadline can be met upon single server failure.
- (2) We propose a QoS-Constrained, Reliable and Energy-efficient task replica deployment (QSRE) algorithm to minimize the number of servers required for task execution and guarantee that the task replicas can be completed within the QoS-specified job execution deadline under the main task failure.
- (3) We evaluate the performance of algorithm QSRE by simulations, and the results demonstrate the effectiveness of QSRE.

2 RELATED WORK

Checkpointing is the most typical fault tolerance method. The use of checkpoint technology in cloud computing systems ensures system reliability, but the frequent creation of checkpoints during the operation of the system causes significant time overhead, and a lot of storage resources will be consumed by the storage of checkpoints^[16]. However, if the frequency of checkpoint creation in the system is reduced, it will take more time to re-execute the failed tasks after a program failure. Therefore, in large-scale cloud computing systems, the checkpoint schemes cannot provide services of high quality to users while ensuring system reliability.

Task replication is another typical fault tolerance technique to provide fault resilience and promote the performance of job completion time for cloud data centers. Limam et al. proposed a dynamic replication strategy that simultaneously satisfies the availability and performance requirements of users; a task replica is created only when the service cannot reach the specified availability target or when the user query response time is greater than the threshold specified in the service level agreement^[17]. Chiang et al. proposed a three-layer cloud dispatching architecture and a task scheduling algorithm to reduce job completion time by assigning each task to a suitable server in a server cluster^[7].

The energy consumption for running servers incurs a huge operational cost on cloud service providers. Load

concentration can reduce the number of servers and hence effectively save energy^[9]. Zhang et al. designed an algorithm to achieve the balance of reliability and energy consumption; the algorithm establishes different priorities for tasks, uses dynamic voltage and frequency scaling (DVFS) to select an energy-saving frequency for servers to execute each task, and assigns tasks to appropriate servers to obtain higher system reliability with low energy consumption^[18]. Fan et al. proposed a reliability-aware and energy-efficient task replicas assignment algorithm that uses task replicas running at a low speed and allows multiple task replicas to share the same server computing resources to minimize the energy consumption of cloud data centers, while providing fault tolerance for task execution^[6].

Reliability, QoS and energy consumption are three important concerns of cloud service providers. Most of the research on task deployment focuses on only one or two objectives of reliability, QoS and energy efficiency. In this paper, we deal with the problem of reliable task deployment in data centers with the objective of minimizing the energy consumption of cloud data centers by minimizing the number of servers required for running the task replicas, while providing fault tolerance for task execution and meeting job execution time requirement.

3 PROBLEM FORMULATION

Each job processed in cloud data centers is usually composed of multiple tasks which are deployed on different servers to execute in parallel to speed up the completion of the jobs. The job is expected to finish running in QoS-specified time D . During the execution of a job, a single server failure is usually much more likely to occur than multiple simultaneous failures, since typical failure modes are single server failures^[14,15]. Therefore, the scenario of single server failure has been attracting much attention from industries and academia^[11,12,13]. Accordingly, we consider the scenario of single server failure. In this paper, we combine task replication and re-execution to provide fault tolerance for cloud data centers. If the execution time of a task is no more than $D/2$, the task can be completed within the specified time requirement D by re-executing it after its failure. If the execution time of a task is longer than $D/2$, a task replica needs to be initiated. That is, there will be two instances of the task: the main task and the task replica. Each main task runs on an individual server at the server's full running rate, and the task replica associated with the main task simultaneously runs on a backup server. In order to reduce the number of servers used, multiple task replicas can be deployed on the same backup server within the server computing capacity. The task replicas on a backup server run at different rates.

In this paper, we assume that all the servers have the same failure probability. The successful execution of the main task will trigger the termination of the associated task replica. When the main task is executed, the task replica runs at a low speed. If the main task fails, the execution speed of the associated task replica is immediately increased to the maximum processing speed of the backup server, while the other task replicas on that backup server is terminated. Note that the failure of a backup server will not affect the execution of the main task.

The later the main task failure, the longer the task replica execution time. In particular, if a main task t_i fails at the last moment, the remaining execution time of task replica t_i^S is $D - s(t_i)$. We call the time as the reserved execution time for deploying task replica t_i^S , which is denoted as $R^S(t_i) = D - s(t_i)$.

In order to ensure that task t_i is completed within D , the task replica must have been executed for time $s^S(t_i)$ before its main task failure, where $s^S(t_i) = s(t_i) - R^S(t_i) = s(t_i) - (D - s(t_i)) = 2s(t_i) - D$.

When $s(t_i) \leq D/2$, the time that the task replica needs to be executed is $s^S(t_i) \leq 0$. That is, the main task does not require a task replica, since the task can be finished within D by re-execution after failure.

When $s(t_i) > D/2$, a task replica needs to be initiated and assigned to a backup server. The longer the execution time of main task t_i , the shorter the reserved execution time of the associated task replica. Note that a task replica t_i^S will run exclusively on the backup server and the other task replicas on the backup server will be terminated, after the associated main task t_i fails. It is also easy to see that the main task with more running time makes the job execution deadline harder to be guaranteed. Therefore, for a backup server v_k running multiple task replicas, we set the reserved time on v_k , i.e. R_k , as the minimum reserved execution time of the task replicas deployed on v_k .

For a job composed of $|T|$ tasks with QoS-specified completion time D , we create $|T|$ main tasks and the associated task replicas if needed. The total number of task replicas is at most $|T|$. We deploy the task replicas on the backup servers with the goal of minimizing the number of backup servers used, while ensuring the job can be completed within D . That is, our objective is to:

$$\text{Minimize } \sum_{1 \leq k \leq N} x_k \quad (1)$$

Subject to

$$x_k = \min \left(1, \sum_{1 \leq i \leq |T|} x_{i,k}^S \right) \quad \forall v_k \in V \quad (2)$$

$$\sum_{1 \leq k \leq N} x_{i,k}^S = 1 \quad \forall t_i \in T \quad (3)$$

$$s^S(t_i) = 2s(t_i) - D \quad \forall t_i \in T \quad (4)$$

$$R^S(t_i) = s(t_i) - s^S(t_i) \quad \forall t_i \in T \quad (5)$$

$$R_k = \min_{t_i \in T} x_{i,k}^S \cdot R^S(t_i) \quad \forall v_k \in V \quad (6)$$

$$\sum_{1 \leq i \leq |T|} x_{i,k}^S \cdot s^S(t_i) + R_k \leq D \quad \forall 1 \leq k \leq N \quad (7)$$

Eq. (2) determines whether a server v_k is used or not. Eq. (3) dictates that each task replica is deployed on one and only one server. Eq. (4) signifies the time required for task replica t_i^S to execute at the maximum speed of the server. Eq. (5) defines the reserved execution time for deploying task replica t_i^S . Eq. (6) signifies the reserved execution time of the backup server is the minimum reserved execution time of all the task replicas deployed on the server. Eq. (7) ensures the sum of the execution time of task replicas on backup server v_k and the reserved execution time of server v_k cannot exceed the job execution time requirement D .

4 TASK REPLICA DEPLOYMENT ALGORITHM

The problem of deploying task replicas on one server is similar to the bin packing problem which is an NP -complete problem. The task deployment problem in this paper requires deploying task replicas to multiple servers to minimize the number of servers used, and the amount of computing resources usable on the servers are different. Obviously, the problem in this paper is more complex. Therefore, the problem in this paper is also an NP -complete problem.

In this section, we propose a simple and effective QoS-Constrained, Reliable and Energy-efficient task replicas

deployment algorithm (QSRE) to allocate the task replicas on the least number of backup servers, while providing fault tolerance for task execution and meeting the job completion requirement, as shown in Algorithm 1. Before executing Algorithm 1, the execution time of each main task and the reserved execution time of each task replica are calculated. The task replicas are then sorted in the non-ascending order of the associated main task execution time.

The task replicas are deployed in an iterative way by Algorithm 1. In each iteration, Algorithm 1 tries to allocate the task replicas with the longest and shortest execution time in the task replica set T^S on the backup server to fully exploit the server computing resources. Specifically, for backup server v_k , Algorithm 1 selects the task replica with the longest execution time and tries to deploy it on server v_k . If the server has enough computing capacity, the reserved execution time of server v_k is set as the reserved execution time of the task replica. Algorithm 1 then determines whether the task replica with the shortest execution time can be allocated on server v_k . If no, the next server v_{k+1} will be used. Otherwise, Algorithm 1 determines whether the task replica with the second longest execution time can be deployed on server v_k . If yes, the algorithm tries the task replica with the second shortest execution time. The process continues until the task replicas with longer execution time cannot be deployed on server v_k . Algorithm 1 will then keep trying to allocate the task replicas with shorter execution time on server v_k , until server v_k cannot accommodate task replicas any more. In this case, the task replica deployment on backup server v_k is finished, and the algorithm will use a new backup server, if any other task replicas are to be deployed.

Algorithm 1: QSRE

Input: Sorted $|T|$ task replicas list by the non-ascending order of the task replica execution time, the reserved time of each task replica, job completion time D .

Output: Number k of servers used to run all the task replicas.

1. **Initialize** $k \leftarrow 1$, $\Delta \leftarrow D$, $Flag \leftarrow False$, $head \leftarrow 1$, $tail \leftarrow |T|$;
 2. **while** $head \leq tail$ **do**
 3. **if** $s^S(t_{head}) \leq \Delta$ **then**
 4. | $\Delta \leftarrow \Delta - s^S(t_{head})$;
 5. | **if** $Flag = False$ **then**
 6. | | $\Delta \leftarrow \Delta - R^S(t_{head})$, $Flag \leftarrow True$;
 7. | $head \leftarrow head + 1$;
 8. **if** $s^S(t_{tail}) \leq \Delta$ **then**
 9. | $\Delta \leftarrow \Delta - s^S(t_{tail})$;
 10. | $tail \leftarrow tail - 1$;
 11. **else if** $Flag$ **then**
 12. | $k \leftarrow k + 1$, $\Delta \leftarrow D$, $Flag \leftarrow False$;
 13. **return** k ;
-

5 PERFORMANCE EVALUATION

We evaluate the proposed algorithm QSRE against algorithm Greedy^[19] and the state-of-art algorithm RER^[6]. In algorithm RER, the execution speeds of all the task replicas are the same, which are α ($0 \leq \alpha \leq 1$) times the maximum processing speed of the server. The task replicas can be mapped onto dedicated backup servers or assigned to

the servers on which the main tasks are running. A small α leads to long completion time of a task. We set α as 0.5, 0.75 and 1.0, respectively, in the simulations. If algorithm RER assigns a task replica on a server where a main task is running, the task replica will not be able to execute in parallel with the main task, which leads to an increase in the job completion time. Therefore, the task replicas are mapped on the dedicated backup servers for algorithm RER in the simulations. Algorithm Greedy deploys the task replicas on the backup servers in a greedy way and also uses the same reservation time as algorithm QSRE. Consequently, algorithm Greedy can guarantee the tasks are completed within the specified time requirement. However, the time complexity of algorithm Greedy is $O(|T|^2)$, which is larger than that of algorithm QSRE.

The performance metrics used in the simulations are the number of backup servers to deploy task replicas and the number of job execution timeouts. In each simulation, a main task is randomly selected to fail, and the associated task replica is accelerated to the maximum processing speed of the backup server. We record the completion time of jobs and determine whether the jobs are timed out. In the simulations, the size of each task is randomly generated within a certain range, and the maximum processing speed of the servers is set as 1. We run each algorithm for 2000 times and the average value of the running results is taken as the final result.

5.1 Impact of Job Execution Time Requirements

The job execution time requirement D has a direct impact on the deployment of task replicas. In the simulations, each job consists of 1000 tasks, and the size of each task is randomly generated in the range of $[0.5, 0.8]$. D cannot be less than the maximum task completion time, and hence we make D vary in $[0.8, 1]$.

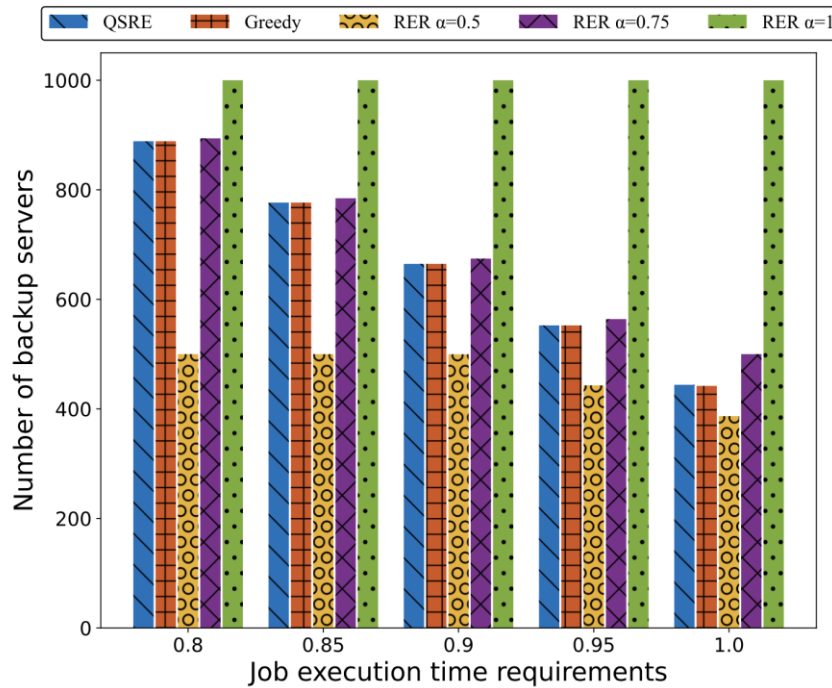


FIG 1 THE IMPACT OF JOB EXECUTION TIME REQUIREMENTS ON THE NUMBER OF BACKUP SERVERS

Fig 1 illustrates the impact of job execution time requirements on the number of backup servers used for each algorithm. With the increase of D , the computing resources that can be provided by the servers also increase. Accordingly, more task replicas can be run on the servers, and the number of backup servers used for each algorithm decreases. The numbers of backup servers used by algorithm Greedy and algorithm QSRE are very close. The number of backup servers used by algorithm RER ($\alpha = 0.75$) is more than that used by algorithm QSRE from 0.6% to 12.6%. Algorithm

RER ($\alpha = 0.5$) uses the least number of backup servers, since the task replicas run at a very low speed and hence more task replicas can be deployed on the same server. Each task replica is deployed on the backup server with the full running speed by algorithm RER ($\alpha = 1.0$), and hence it requires the most number of backup servers. Algorithm QSRE uses 12%-53% fewer backup servers than algorithm RER ($\alpha = 1.0$).

Table 1 shows the number of timeouts for the algorithms under different job execution time requirements when the main task fails at a random moment. From the table, it can be observed that algorithms QSRE, Greedy, and RER ($\alpha = 1.0$) can guarantee the jobs are completed within the specified time requirement. In contrast, algorithm RER experiences a high number of timeouts, when $\alpha = 0.5$ and $\alpha = 0.75$. The result shows that the algorithm at $\alpha = 0.5$ and $\alpha = 0.75$ requires more time to run the task replicas after the main task failure, such that the job execution cannot meet the deadline.

TABLE 1 THE NUMBER OF TIMEOUTS FOR EACH ALGORITHM UNDER DIFFERENT JOB EXECUTION TIME REQUIREMENTS WHEN THE MAIN TASK FAILS AT A RANDOM MOMENT

Job Execution Time Requirement	0.8	0.85	0.9	0.95	1.0
QSRE	0	0	0	0	0
Greedy	0	0	0	0	0
RER $\alpha = 0.5$	961	691	501	358	289
RER $\alpha = 0.75$	152	128	178	208	219
RER $\alpha = 1.0$	0	0	0	0	0

The simulations in this section demonstrate the performance of different algorithms with various job execution time requirements. In general, algorithms QSRE, Greedy, and RER ($\alpha = 1.0$) are able to finish the job execution within the QoS-specified deadline when the main tasks fail. On the contrary, there are a significant amount of timeouts with algorithm RER at $\alpha = 0.5$ and $\alpha = 0.75$. Algorithm RER at $\alpha = 1.0$ uses full task replication, and hence it requires the most number of backup servers. Algorithms Greedy and QSRE achieve the similar performance in terms of the number of backup servers used. However, the time complexity of Greedy is larger than that of QSRE. In general, we find that algorithm QSRE achieves the best performance of the number of servers used under the constraint of job execution deadline.

5.2 Impact of Task Execution Time

We evaluate the impact of task execution time on the performance of different algorithms. Each job consists of 1000 tasks. The job execution time requirement D is set as 1. When the execution time of a task is less than $1/2$, the task can be re-executed after failure, which can ensure the SLA of the job execution. Therefore, the minimum execution time of the tasks is set as 0.5, and the maximum execution time of the tasks varies in $[0.8, 1]$.

Fig 2 illustrates the impact of task execution time on the number of backup servers. As the task execution time increases, the tasks consume more computing resources, and hence the deployment of task replicas requires more backup servers. Algorithms QSRE and Greedy achieve very close performance with Greedy using up to 0.4% fewer backup servers than QSRE. Algorithm RER uses 2%-12% and 35%-55% more backup servers than algorithm QSRE, when $\alpha = 0.75$ and $\alpha = 1.0$, respectively. A higher α value means that each task replica runs at a faster speed, and hence the task replica consumes more computing resources.

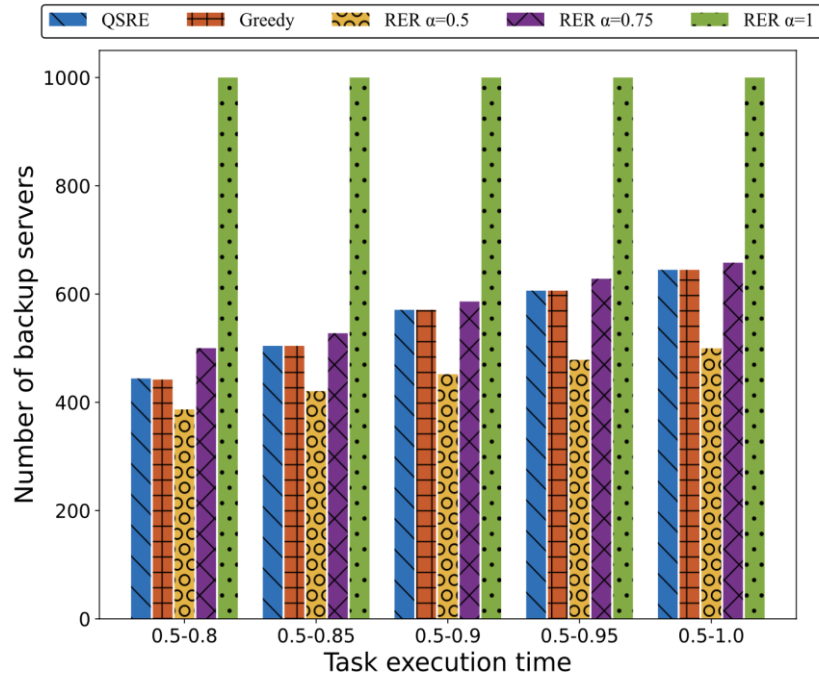


FIG 2 THE IMPACT OF TASK EXECUTION TIME ON THE NUMBER OF BACKUP SERVERS

Table 2 shows the number of timeouts for each algorithm under different task execution time when the main task fails at a random moment. The number of timeouts for algorithm RER at $\alpha = 0.5$ and $\alpha = 0.75$ increases gradually with more task execution time, due to the increasing workload of the jobs. Algorithm RER at $\alpha = 0.5$ and $\alpha = 0.75$ experiences a high number of timeouts. Timeouts occur in 15%-33% jobs with algorithm RER ($\alpha = 0.5$), when the main task fails at a random moment. 10%-15% jobs cannot be completed within the job execution time requirement for algorithm RER ($\alpha = 0.75$), when the main task fails at a random moment. Algorithm RER at $\alpha = 1.0$ runs a full replica for each main task, and hence algorithm RER experiences no job execution timeouts. There are no job timeouts with algorithms QSRE and Greedy either, since they use the reserved time on the server to guarantee the job execution suffers from no timeouts.

TABLE 2 THE NUMBER OF TIMEOUTS FOR EACH ALGORITHM UNDER DIFFERENT TASK EXECUTION TIME WHEN THE MAIN TASK FAILS AT A RANDOM MOMENT

Tasks Execution Time	0.5~0.8	0.5~0.85	0.5~0.9	0.5~0.95	0.5~1.0
QSRE	0	0	0	0	0
Greedy	0	0	0	0	0
RER $\alpha = 0.5$	289	339	466	605	665
RER $\alpha = 0.75$	219	266	276	280	293
RER $\alpha = 1.0$	0	0	0	0	0

The simulations in this section demonstrate the performance of different algorithms by varying the task execution time. On the whole, algorithms QSRE, Greedy, and RER ($\alpha = 1.0$) can guarantee the jobs are completed within the specified time requirement, while there are a large amount of timeouts with algorithm RER at $\alpha = 0.5$ and $\alpha = 0.75$. Algorithms Greedy and QSRE achieve very close performance in terms of the number of backup servers used. However, the time complexity of algorithm Greedy is higher than that of algorithm QSRE. QSRE uses fewer backup servers than algorithms RER at $\alpha = 0.75$ and $\alpha = 1.0$. In general, the simulations in this section also show that

algorithm QSRE achieves the best performance of the number of servers used, while meeting the constraint of job execution deadline.

6 CONCLUSION

There is an urgent need for efficient task deployment strategies to improve the energy efficiency while guaranteeing the reliability and the QoS of job execution in cloud data centers, since these three factors are important concerns of cloud service providers. Most of the research on task deployment focuses on only one or two of these three factors. However, they have intrinsic trade-off relationships. In this paper, we tackled the problem of reliable task deployment in data centers, with the goal of minimizing the energy consumption of cloud data centers while providing fault tolerance for task execution and guaranteeing job execution timelines. We proposed a QoS-Constrained, Reliable and Energy-efficient task replica deployment (QSRE) algorithm by combining task replication and re-execution. For each task in a job that cannot finish executing by re-execution within deadline, we initiate two replicas for the task: main task and task replica. Each main task runs on an individual server at the full speed. Multiple task replicas can share the computing resources of a backup server and run at different speeds on the server. Algorithm QSRE deploys task replicas iteratively. In each iteration, the algorithm tries to deploy the task replicas with the longest and shortest execution time on the backup server in turn to exploit the computing resources of the server. Compared to the existing algorithms, QSRE can not only guarantee the job completion deadline under the task failure, but also achieve superior performance in terms of the number of backup servers used.

REFERENCES

- [1] Engelmann C, Geist A. Super-scalable algorithms for computing on 100,000 processors[C]//International Conference on Computational Science. Springer, 2005: 313-321.
- [2] Cui X, Mills B N, Znati T, Melhem R G. Shadows on the cloud: An energy-aware, profit maximizing resilience framework for cloud computing[C]//CLOSER, 2014: 15-26.
- [3] Cost of data center outages 2020[EB/OL]. <https://pitcch.org/uei6upi6/1c0972-cost-of-data-center-outages-2020>.
- [4] Garraghan P, Moreno I S, Townend P, Xu J. An analysis of failure-related energy waste in a large-scale cloud environment[J]. IEEE Transactions on Emerging topics in Computing, 2014, 2(2): 166-180.
- [5] Luo L, Wu W, Zhang F. Energy modeling based on cloud data center[J]. Journal of Software, 2014, 25(7): 1371-1387.
- [6] Fan Y, Wang C, Wu W, Znati T, Du D. Slow replica and shared protection: Energy-efficient and reliable task assignment in cloud data centers[J]. IEEE Transactions on Reliability, 2021, 70(3): 931-943.
- [7] Chiang M L, Huang Y F, Hsieh H C, Tsai W C. Highly reliable and efficient three-layer cloud dispatching architecture in the heterogeneous cloud computing environment[J]. Applied Sciences, 2018, 8(8): 1385.
- [8] Tan Y, Zeng G, Wang W. Policy of energy optimal management for cloud computing platform with stochastic tasks[J]. Journal of software, 2021, 23(2): 266-278.
- [9] Liu X, Zhan Z, Deng J D, Li Y, Gu T, Zhang J. An energy efficient ant colony system for virtual machine placement in cloud computing[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(1): 113-128.
- [10] Chase J S, Anderson D C, Thakar P N, Vahdat A M, Doyle R P. Managing energy and server resources in hosting centers[J]. ACM SIGOPS operating systems review, 2001, 35(5): 103-116.
- [11] Softchris. Reliability metrics - Training[EB/OL]. <https://learn.microsoft.com/en-us/training/modules/cmu-data-center-design/7-reliability-metrics>.
- [12] Guo Z, Li J, Ramesh R. Optimal management of virtual infrastructures under flexible cloud service agreements[J]. Information Systems Research, 2019, 30(4): 1424-1446.
- [13] Sahoo R K, Squillante M S, Sivasubramaniam A, Zhang Y. Failure data analysis of a large-scale heterogeneous server environment[C]//International Conference on Dependable Systems and Networks, IEEE, 2004: 772-781.
- [14] Eisler M, Corbett P, Kazar M, Nydick D S, Wagner J C. Data ONTAP GX: A scalable storage cluster[C]//FAST, 2007: 23-23.

- [15] Masuda S, He F, Kawabata A, Oki E. Distributed server allocation model with preventive start-time optimization against single failure[C]//2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR), IEEE, 2020: 1-6.
- [16] Pedram M. Energy-efficient datacenters[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(10): pp. 1465-1484.
- [17] Limam S, Mokadem R, Belalem G. Data replication strategy with satisfaction of availability, performance and tenant budget requirements[J]. Cluster Computing, 2019, 22(4): 1199-1210.
- [18] Zhang L, Li K, Xu Y, Mei J, Zhang F, Li K. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster[J]. Inform Sciences, 2015, 319: 113-131.
- [19] Taal A, Wang J, de Laat C, Zhao Z. Profiling the scheduling decisions for handling critical paths in deadline-constrained cloud workflows[J]. Future Generation Computer Systems, 2019, 100: 237-249.

AUTHORS

¹**Zhenghui Zhang**, Master's degree student, School of Computer and Information Technology, Hefei University of Technology. Research Interests: Cloud computing and edge computing.

²**Yuqi Fan**, Associate Professor, School of Computer and Information Technology, Hefei University of Technology. Research interests: Artificial Intelligence and Blockchain.