RESEARCH ON TASK SCHEDULING STRATEGY BASED ON THE TRUSTWORTHINESS OF MAPREDUCE

QIN Jun[1,2], SONG Yanyan[3] and ZONG Ping[4,5]

[1]Communication University of China, Nanjing, China
[2]Nanjing University of Posts and Telecommunications, China
[3]Communication University of China, Nanjing, China
[4]Nanjing University of Science and Technology Zijin College, China
[5]Nanjing University of Posts and Telecommunications, China

ABSTRACT

With the rapid development and popularization of information technology, cloud computing technology provides a good environment for solving massive data processing. Hadoop is an open-source implementation of MapReduce and has the ability to process large amounts of data. Aiming at the shortcomings of the fault-tolerant technology in the MapReduce programming model, this paper proposes a reliability task scheduling strategy that introduces a failure recovery mechanism, evaluates the trustworthiness of resource nodes in the cloud environment, establishes a trustworthiness model, and avoids task allocation to low reliability node, causing the task to be re-executed, wasting time and resources. Finally, the simulation platform CloudSim verifies the validity and stability of the task scheduling algorithm and scheduling model proposed in this paper.

KEYWORDS

Cloud Environment, Failure Recovery Mechanism, Task Scheduling Algorithm .

1. INTRODUCTION

With the extensive use of Internet, e-commerce and other application technologies, how to use data analysis technology to extract resources with commercial and application value from massive data resources is a problem worthy of in-depth study. MapReduce programming model is distinguished by its advantages of good scalability, fault tolerance and large-scale parallel processing, and has become the key technology of big data processing and analysis [1]. Google company proposed to use MapReduce programming model to deal with parallel computing of large-scale data sets. MapReduce uses the design ideas of functional programming language and vector programming language for reference, and proposes a simple parallel programming method, which realizes basic parallel computing tasks by using map and reduce function programming. MapReduce distributes the task of large-scale dataset operation to each sub node under the management of the master node to complete together, and then obtains the final result by integrating the intermediate results of each sub node, so as to realize the reliable execution and fault tolerance mechanism of the task [2].

One of the purposes of MapReduce programming model design [3,4] is to use a large number of working nodes to process massive amounts of data, so MapReduce must be able to quickly handle failed machines [5]. In the MapReduce programming model, the Job Tracker node periodically pings each Task Tracker node. If the Task Tracker node does not respond within a specified time, the node will be marked as invalid. All tasks completed on the failed node will be set to an unexecuted state and assigned to be executed again on other Task Tracker nodes.

This paper makes the following assumptions about the cloud environment platform:

(1) The cloud environment platform is heterogeneous.
(2) All nodes in a heterogeneous platform have and only have two working states: normal operation state and failure state. When a node fails, it is in a failed state.
(3) The failure state of any node has nothing to do with other nodes and will not affect other nodes in a normal state.
(4) If a node is in an idle state when it fails, the node will be replaced by another node in the standby state, which does not affect the schedulability of the task. If there are tasks being executed on the failed node, the failure recovery mechanism and replacement mechanism will be used to ensure that the node is restored to a normal operating state.

2. Reliability Task Scheduling Strategy with Introduction of Failure Recovery Mechanism

Trustworthiness is a parameter to evaluate the reliability of a system or product [6,7]. Here reliability means that a certain system or product is trustworthy or trustworthy. Usually, trustworthiness metrics are as follows.

(1) Trustworthiness

Trustworthiness refers to the probability that the system completes the specified task within a specified time t according to user requirements. It is expressed as a time function:

$$R(t) = P(T > t) 0 < t < \infty$$

T represents the working time before the failure. Untrustworthiness and trustworthiness have a complementary relationship. Untrustworthiness is also called failure probability. Failure probability F is also a function of time.

$$F(t) = P(t < T) 0 < t < \infty$$
$$F(t) = 1 - R(t)$$

(2) Failure rate

The failure rate refers to the probability that the product or system has not failed at the moment of work until the time t, and the system will fail in the next unit time after the time t.

Assuming that the failure rate of a system or product obeys an exponential distribution, the relationship between trustworthiness and failure rate is:

$$R(t) = e^{-\lambda t}$$

(3) Mean Time to Failure (MTTF)

Suppose that under the same test conditions, there are N irreparable systems or products whose failure time is:

$$MTTF = \frac{1}{N}\sum_{i=1}^{N} ti$$

For some systems or products that fail to be repaired, the average time before failure is their average lifespan. If the probability of system failure obeys an exponential distribution, there is

$$MTTF = \int_0^{\infty} e^{-\lambda t}\, dt = \frac{1}{\lambda}$$

(4) Mean Time Between Failures, MTBF

The mean time between failures refers to how long the system or product runs on average before a failure occurs. The continuous time of each work is calculated as t_1, t_2, ... t_N, and the mean time between failures is:

$$MTBF = \frac{1}{N}\sum_{i=1}^{N} ti$$

(5) Mean Time To Repair (MTTR)

The average repair time refers to the average repair time of a system or product that can be repaired. The repair time of a system or product is not a certain time. Assuming the repair rate is μ, and obeys the exponential distribution, the average repair time can be expressed as:

$$MTTR(t) = \int_0^{\infty} tue^{-ut}\, dt = \frac{1}{u}$$

The task scheduling strategy that aims at maximizing the reliability of task scheduling and minimizing the total response time of the task is a common task scheduling model in the cloud environment [8].

Definition 1: Node model. Assuming that the nodes in the cloud environment are heterogeneous, there are differences in the performance of each node Nj in N. The node model to be studied is described as an undirected graph G(T, E), the node set: N={N_1,N_2……,Nm}, m is the number of nodes, E represents the edge set of the graph, and mainly represents the number of nodes Interrelationships. Expressed by an n*m matrix RT, RTij represents the running time of task i on node j. A matrix TI of order m*m is used to represent the communication volume between nodes, where the communication volume between node a and node b is represented as TIab. In the undirected graph G(T, E), the virtual machines are independent of each other and have no dependencies.

Definition 2: Task response time T_{ij}. It mainly includes task waiting time WT_{ij}, task transmission time CT_{ij} and task execution time RT_{ij}.

$$T_{ij} = WT_{ij} + CT_{ij} + RT_{ij} \qquad (1)$$

The time when all tasks are completed, we use ST to indicate.

$$ST = max \quad T_{ij} \qquad (2)$$

Definition 3: Trustworthiness. The trustworthiness of the node is mainly considered from the two aspects of node failure rate and failure repair rate [9]. The failure of the node is mainly the communication link between the node and the node and whether the node itself fails. Assuming that the probability of node failure and the probability of failure of the communication link between nodes obey the Poisson distribution with parameters σ and ξ respectively, the probability of node failure p times within the time interval [0, t] is the same. It can be seen that the communication link failure probability $\lambda_p(t) = e^{-\sigma t}/p!$ within the time interval [0, t]. Assume that the probability of a node failure and recoverable obeys the Poisson distribution with parameter $\theta_p(t) = e^{-\xi t}/p!$, where the communication link cannot be recovered when the communication link fails. Assume the probability $\mu_p(t) = e^{-\varepsilon t}/p!$ that the node is repaired p times in the time interval [0, t].

Define $P_j(t)$ as the probability that there is no failure (ie p=0) on node j in the time interval [0,t], and its probability value is

$$P_j(t) = e^{-\sigma_j t} = e^{-\sigma_j RT_{ij}} \qquad (3)$$

Define $C_j(t)$ as the probability of no failure (ie p=0) on the communication link between node a and node b in the time interval [0, t], and its probability value is

$$P_j(t) = e^{-\xi t} = e^{-\xi_j TI_{ab}/Net_{ab}} \qquad (4)$$

In equation 4, TI_{ab}/Net_{ab} represents the communication time between nodes on the communication link.

Define the probability that $R_j(t)$ does not repair on node j in the time interval [0, t] (ie p=0), and its probability value is

$$R_j(t) = e^{-\varepsilon t} = e^{-(1-I_j)\varepsilon_j RT_{ij}} \qquad (5)$$

According to the above introduction, the probability of node j completing task i is K_{ij}.

$$K_{ij} = P_{ij} \times C_{ij} \times R_{ij} = e^{-\sigma_j RT_{ij}} \times e^{-\xi_j TI_{ab}/Net_{ab}} \times e^{-\varepsilon_j RT_{ij}} = e^{-\sigma_j RT_{ij} - \xi_j TI_{ab}/Net_{ab} - \varepsilon_j RT_{ij}} \qquad (6)$$

In order to improve the parallelism of the application, a job is often divided into multiple tasks and executed in parallel on multiple nodes at the same time. A task is executed on only one node. When all tasks return the task execution results, it indicates that the job is successfully completed. Assuming that N(j) is the set of nodes performing the task, the reliability of the task can be expressed as

$$Trust(N) = \prod_{i=1}^{n}\prod_{j=1}^{m} K_{ij} = \prod_{i=1}^{n}\prod_{j=1}^{m} e^{-(\sigma_j RT_{ij} + \xi_j TI_{ab}/Net_{ab} + \varepsilon_j RT_{ij})} = \prod_{i=1}^{n}\prod_{j=1}^{m} e^{trust_{ij}} \qquad (7)$$

According to formula 2 and formula 7, in order to maximize the reliability of the scheduling strategy and minimize the total response time of the task, the objective function is set as formula 8.

$$MinF = -x\,ln(Trust(N)) + ST \qquad (8)$$

Since the value of the total response time of the task is larger than the value range of the reliability, x can be used as a scale factor to coordinate the proportion of reliability and time cost to prevent the time cost from controlling the objective function value.

3. Reliability Task Scheduling Algorithm Introducing Failure Recovery Mechanism

Aiming at the task scheduling problem in the cloud environment, the trust evaluation model considering the failure recovery mechanism is introduced into the ant colony simulated annealing algorithm, and the ant colony simulated annealing algorithm considering the failure recovery mechanism is proposed.

This paper proposes an ant colony algorithm based on SA (Ant Clony Optimization Simulated Anealling, ACOSA) [10]. Its principle is to find the local optimal task scheduling solution for tasks T_i and node N_j through ACO, and then use SA for local optimization. Thereby assigning tasks to appropriate heterogeneous resource nodes for execution. Among them, ACO sets the initial pheromone concentration to a constant before the ant searches, which will increase the ant's search space:

$$\tau_j(0) = c$$

With the increase of time, the concentration of pheromone becomes higher and higher, and the ant can choose the appropriate path according to the concentration of the pheromone on the path that the ant walked last time.

When the task is scheduled to be executed on the resource node, the trustworthiness reflects the reliability of the service provided by the target resource node. The ACOSA algorithm fully considers the heterogeneous characteristics of resource nodes but does not consider the impact of the trustworthiness of the resource nodes on the task scheduling results. For this reason, we take the maximization of credibility and the minimization of time cost as the objective function, and the Function as a heuristic function of ACO.

$$\eta_{ij} = \frac{1}{-x\ln(K_{ij}) + RT_{ij}}$$

In the actual cloud environment, when a node fails, the tasks assigned to the node will often be re-executed. The advantage of introducing a failure recovery mechanism is that for those failures that can be recovered, the node can recover the tasks that have stopped executing by running the failure recovery program.

Execution steps based on reliability task scheduling strategy:

(1) Initialization parameters. Set the initial temperature T_{max}, the maximum number of iterations Ite_{rmax}, and the initial pheromone τ_{ij}.

(2) Construct a feasible solution. Ant j selects the appropriate node according to the selection migration rule, adds the selected node to the taboo table, and instructs all tasks to be allocated to appropriate node resources.
(3) Update pheromone. When all ants complete the path search, a local optimal solution is generated, and the local optimal solution is used to update the local pheromone.
(4) SA performs partial optimization. According to the local optimal solution obtained by ACO, use SA to optimize the local optimal solution to obtain a new solution.
(5) Metropolis guidelines. According to Metropolis criterion, judge whether the new solution constructed by SA will be accepted.
(6) Termination criteria. Cool down the current temperature and determine whether the termination criterion is met. If it is met, execute (7), otherwise return to (4).
(7) Global pheromone update. Update the global pheromone according to the candidate solution generated by SA, the number of iterations is increased by 1. If the number of iterations is greater than $Iter_{max}$, all steps are terminated, otherwise, return to (2).

4. SIMULATION

In the simulation experiment, the influence of the failure recovery mechanism and its parameters on the evaluation of the trustworthiness of the resource node is discussed, and the performance of the algorithms is compared in the case of different nodes and the number of tasks.

The experimental environment parameters are set as follows: task Rcc is 0.1, 1, 10 respectively; the number of tasks is 100, the number of nodes is 20, and the number of communication links is 20. Set up two kinds of nodes with low trustworthiness, which account for 20% and 30% of the total number of nodes respectively, and the probability of execution failure of the two types of nodes is 80% and 50% respectively. The experimental results are shown in Figure 1.

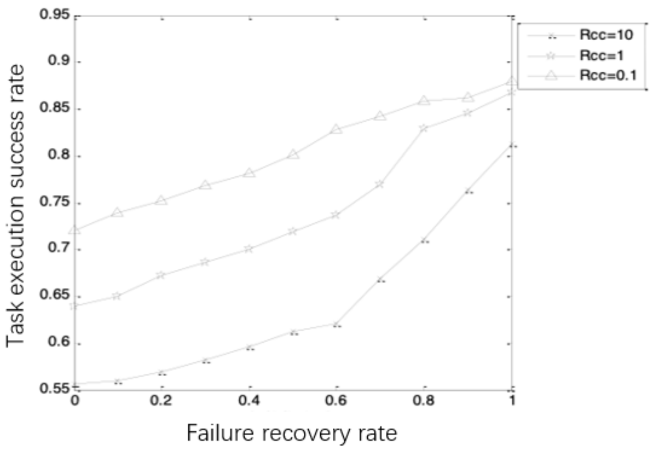

Figure 1. Task execution success rate under different failure recovery rates (without limiting the maximum number of recovery times)

It can be seen from Figure 1 that when there is no failure recovery mechanism (that is, uk=0), the probability of successful completion of the three types of tasks is relatively low. As the failure recovery rate u_k increases, the probability of successful execution of the three types of tasks slowly increases. When u_k is 1, the probability of successful task execution does not reach 100%. This is because although recoverable failures can be recovered through the application, some failures are not recoverable. The two curves with Rcc of 0.1 and 10 in Figure 1 represent calculation-intensive tasks and communication-intensive tasks, respectively. The communication-intensive tasks use the communication link for a relatively long time, and the probability of communication link failure is calculated. Intensive tasks are high. Since communication link failures are unrecoverable, the probability of successful completion of communication-intensive tasks is greater than that of computationally-intensive tasks.

In order to study the effect of failure recovery rate on task execution time, this article conducted experiments on three different types of tasks, and the experimental results are shown in Figure 2.

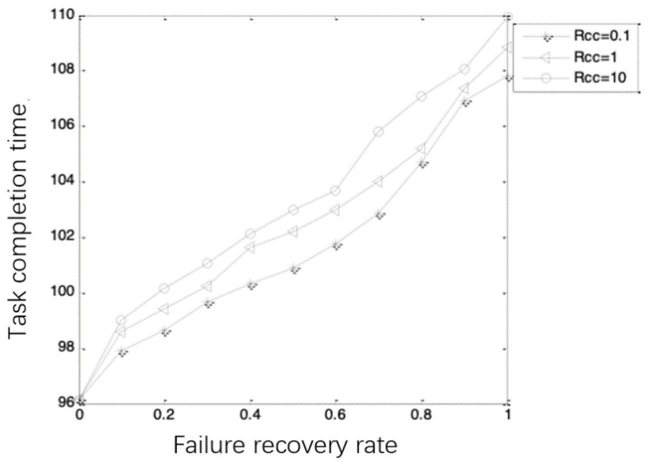

Figure 2. Task completion time corresponding to different failure rates

It can be seen from Figure 2 that different failure recovery rates correspond to different task execution times. As the failure recovery rate increases, the task completion time also gradually increases. This is because the service overhead generated during the failure recovery process causes the task execution time to increase. In an ideal situation, the node has no failure u_k=0, all tasks can be successfully executed according to the assigned node, and the task completion time is the shortest. In Figure 2, when the failure recovery rate is less than 0.6, the task completion time increases slowly, but when the failure recovery rate exceeds 0.6, the task completion time increases sharply. This is because the failure recovery time will increase when the node fails to recover. Frequent failure recovery will increase the task completion time. It can be seen that choosing an appropriate failure recovery rate has an important impact on the task completion time.

Experiments have proved that the introduction of a failure recovery mechanism does increase the probability of successful task execution, indicating that the failure recovery mechanism is effective, but time and resource overhead will also be incurred during the failure recovery process, so we have to choose an appropriate failure recovery probability. In the case of comparing the number of different tasks, compare the task execution success rate and the value of the objective function. Compare the FCFS algorithm (First Come First Service, FCFS) and ACOSA algorithm, where the failure recovery rate u_k is set to 0.6.

Figures 3 and Figures 4 respectively show different task execution success rates corresponding to different task numbers and the objective function values corresponding to different tasks.

It can be seen from Figure 3 that when the number of tasks gradually increases, the task execution success rate of the FCFS algorithm and the ACOSA algorithm is gradually decreasing, but the task execution success rate of the ACOSA algorithm with the introduction of a failure recovery mechanism is significantly higher than that of the FCFS algorithm.

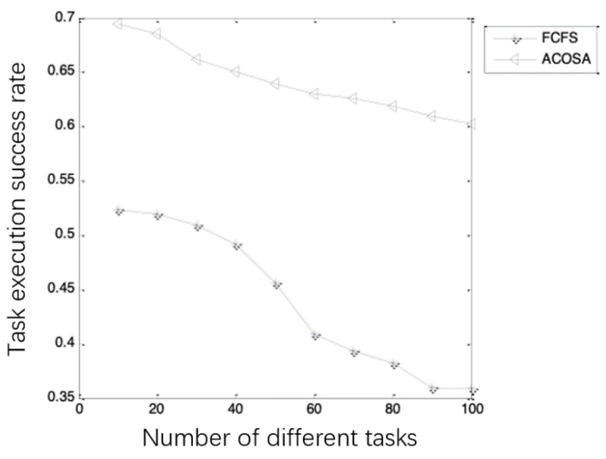

Figure 3. Different task numbers correspond to different task execution success rates

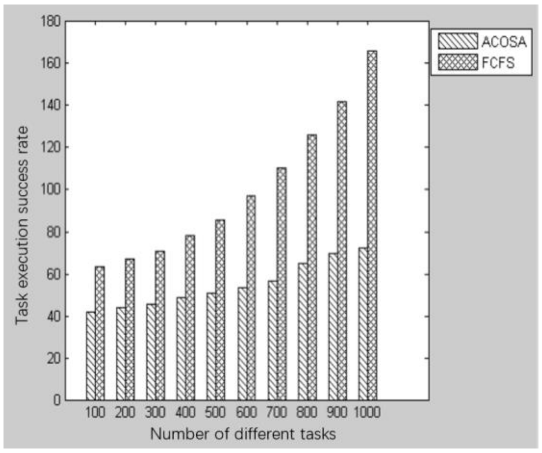

Figure 4. Objective function values corresponding to different tasks

It can be seen from Figure 4 that the task completion time of the ACOSA algorithm with the introduction of the failure recovery mechanism is shorter than the task completion time of the FCFS algorithm, which proves that the introduction of the failure recovery mechanism can not only improve the execution success rate of the task, but also select the appropriate failure recovery The performance of the ACOSA algorithm with the introduction of a failure recovery mechanism is better than that of the FCFS algorithm in the case of low rate.

5. CONCLUSION

A cluster composed of a large number of resource nodes in a cloud environment has the characteristics of heterogeneity, dynamics, and uncertainty. Reliability refers to the probability that a task submitted by a user is successfully completed. In a cloud environment with a large number of resource nodes, node failure events are inevitable. Aiming at the defects of the MapReduce fault tolerance mechanism, this paper proposes a reliability task scheduling strategy that introduces a failure recovery mechanism to evaluate the trustworthiness of resource nodes in the cloud environment. Establish a trustworthiness model to avoid assigning tasks to nodes with low reliability, causing tasks to be re-executed, and wasting time and resources. Finally, the simulation platform CloudSim verifies the validity and stability of the task scheduling algorithm and scheduling model proposed in this paper.

ACKNOWLEDGEMENTS

The authors would like to thank the support from General project of natural science research in universities of Jiangsu Provincial Department of Education (No. 19KJD520007).

REFERENCES

[1] CHENG Yan, ZHANG Yun. (2018) “Improvement and research on Apriori algorithm based on MapReduce -HBase”. Journal of Nanjing University of Posts and Telecommunications. Vol.38, No. 5, pp91-99.

[2] LI Jun. （2021） “Design of online aggregation optimization of big data based on MapReduce”. Journal of Hebei University, Vol. 41, No. 2, pp212-217.

[3] Jianjiang Li, Yajun Liu, Peng Zhang, Wei Chen, Lizhe Wan. (2020) “Map-Balance-Reduce: An improved parallel programming model for load balancing of MapReduce”. Future Generation Computer Systems. Vol. 4, No. 1. pp150-157.

[4] Zhao Xincan, Zhu Yun, Mao Yimin. (2020) “Incremental mining algorithm of Apriori based on MapReduce”. Application Research of Computers. Vol. 37, No. S2, pp73-75+79.

[5] WU Lizhen, KONG Chun, CHEN Wei. (2021) “Short-term load forecasting based on linear regression under MapReduce framework”. Journal of Lanzhou University of Technology. Vol. 47, No.1, pp97-104.

[6] ZHANG Jie. (2016) “The Trust of Networked Software Measurement Model to Optimize the Simulation Analysis”. Computer Simulation. Vol. 33, No. 10, pp278-281+299.

[7] JIANG Jing, YU Yonghong, ZHAO Weibin. (2020) “Research on A Trust Model Based on the QoS and Malicious Node Deletion”. Computer & Digital Engineering. Vol. 48, No. 1, pp98-105.

[8] JIANG Qiang, YI Chun-lin, ZHANG Wei, GAO Sheng. (2021) “The Multi-objective Path Planning for Mobile Robot Based on Ant Colony Algorithm”. Computer Simulation. Vol. 38, No. 2, pp318-325.

[9] WU Chun, YOU Xiaojian, LYU Tao. (2018) “Research on multipath secure routing based on confidence degree”. Journal of Northeast Normal University (Natural Science Edition). Vol. 50, No. 4. pp66-72.

[10] LIU Pengfei, MAO Yingchi, WANG Longbao. (2019) “Task assignment method based on cloud-fog cooperative model”. Journal of Computer Applications. Vol. 39, No.1, pp8-15.

AUTHORS

Qin Jun, Professor, graduated from Nanjing University of Posts and telecommunications. He has presided over and participated in more than 30 scientific research projects, won more than 10 awards, published more than 80 academic papers in academic journals and international conferences, and published 5 teaching materials.