Edge computing task scheduling strategy based on load balancing

Lintan Sun, *Zigan* Li[*], *Jingxian* Lv, *Chenfei* Wang, *Yajuan* Wang, *Long* Chen and *Dong* He

National Grid Co., Ltd. Customer Service Center Information Operation and Maintenance Center, China

Keywords: Edge calculation, Load balancing, Secondary allocation.

Abstract. With the rapid development and wide application of the Internet of Everything, in order to cope with the increasing amount of data and computational scale of mobile terminal processing, and the imbalance of existing scheduling algorithms and low resource utilization, this paper proposes a task scheduling algorithm based on business priority. The algorithm firstly divides the service according to the priority of the service. Secondly, the standard deviation of the computing task group is used to determine the proportion of long and short services, and the dynamic selection model is established. Finally, according to the idea of secondary allocation, the task of heavy load is assigned to the scheduling strategy of light load resources to execute, and the service redistribution model is established. The simulation results show that compared with the typical algorithm, the proposed algorithm achieves the result of comprehensive consideration of Makespan and load balancing to improve system efficiency.

1 Introduction

In the era of the Internet of Everything, the Internet of Everything generates massive amounts of data, and the network bandwidth and computing resources of the existing cloud computing model cannot process these data efficiently. So there is edge computing technology. The mobile edge computing technology sends the service to be processed to the edge computing server closer to the user terminal, which solves the problem that the processing power of the mobile terminal is insufficient, the hardware condition is poor, and the distance to the cloud is too long, resulting in excessive delay response.

In the whole process of the application of the edge computing technology, completing a business calculation requires communication resources and computing resources of the edge computing server. Therefore, the time and system loss required in the calculation process can be reduced by optimizing the computational resource allocation strategy, as shown in Figure 1. In [3], the problem of optimization of resource allocation is solved by using the mutual iteration of the game algorithm and the Hungarian algorithm, thereby reducing the calculation energy consumption and delay. In [4], the task unloading problem is expressed as a mixed

[*] Corresponding author: gwkfzx_xxywzx@163.com

integer nonlinear computing process, which converts the reduced latency problem into two sub-problems: task unloading placement problem and resource allocation. Literature [5] combines user fees with task deadlines to construct reasonable user priorities, and uses improved discrete particle swarm optimization to achieve optimal scheduling of cloud computing tasks in a short period of time.

Based on the min-min scheduling algorithm and max-min scheduling algorithm and related research, this paper makes improvements, divides the task set according to different priorities, designs the standard deviation matching function of long and short task sets, and proposes an improved algorithm. Combining the advantages of the two scheduling algorithms, the benign matching of the resource tasks is realized, and the secondary allocation mechanism is added after the pre-allocation is completed, and the idle resources in the pre-allocation phase are fully utilized, thereby further reducing the computing time and improving the load balancing.

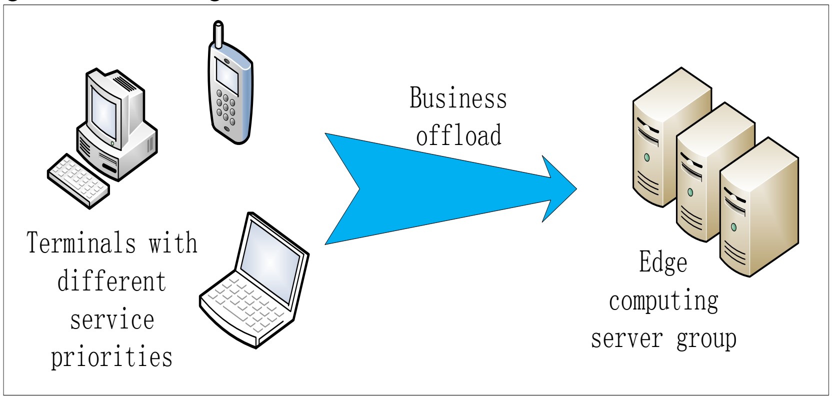
Business offload
Terminals with different service priorities
Edge computing server group

Fig. 1. Edge calculation flow chart.

2 System modeling

2.1 Business wait queue model

When the mobile terminal transmits the service to be offloaded to the edge server, the service is first stored in the service scheduler. The service scheduler divides the services according to the importance degree and sorts them from large to small, and forms the task scheduling queues D{S_1, S_2, S_3, S_4, S_5,…,S_l}} of the different priority areas according to the difference of the minimum completion time. The specific parameters of the service T_n after sorting the priority and the minimum completion time are set as T_n={ID_{Tn}, Da_{Tn}, W_{Tn}, P_{Tn} }. Where: ID_{Tn} is the service number of the service; Da_{Tn} is the data volume of the service; W_{Tn} is the waiting time in the service dispatch queue after the service is transmitted to the edge server; P_{Tn} is the importance of the service. The higher the importance of the service, the higher the priority of the service scheduling.

2.2 Server resource scheduling model of edge computing

This paper considers the distributed computing scenario of multi-tasking and multi-edge servers. Set up one server on the edge computing network, using the set S={ S_1, S_2, S_3, S_4, S_5,…,S_l}. For any server S_m in the S set, the specific parameter configuration of the server can be represented by the set S_m={ID_m, C_m, P_m, W_m }, wherein the specific meanings of each parameter are as follows: ID_m indicates the edge The numbering flag of the server serves as the basis for distinguishing the server; C_m indicates the computing power of the edge computing server; P_m indicates the priority of the edge computing server; W_m indicates the ready time of the edge computing server.

It can be seen from the parameters that the time $t_n = Da_{Tn}/C_m$ consumed by the edge calculation server for calculating the task T_n . The resource scheduler sorts the edge calculation servers according to the priority P_m to form a resource scheduling queue, which facilitates resource matching for subsequent transmission services.

3 Edge computing task scheduling strategy based on standard deviation and secondary allocation

3.1 One-pre-scheduled resource allocation algorithm based on standard deviation

In this paper, the number of edge computing servers in the research scenario is smaller than the number of mobile terminals, that is, the number of services, and the working state of all servers in the initial state is idle. The service set with the highest priority is assigned. First, calculate the minimum completion time standard deviation of all services in the service set, and have the shortest completion time of the i-th service = the ready time of the edge calculation server + the service calculation time. The formula is as follows:

$$ETC_i=W_m+t_i \tag{1}$$

where in, ETC_i represents the shortest completion time of the i-th service; W_m represents the ready time of the m-th edge calculation server; $t_i = Da_{Ti}/C_m$ represents the time consumed by the m-th edge calculation server to calculate the task T_i .

Therefore, the shortest completion time standard deviation SD of all services in the business concentration can be expressed as:

$$SD = \sqrt{\frac{\sum_{i=1}^{s} ETC_i - aveETC}{s}} \tag{2}$$

where s represents the number of services concentrated in the service; ETC_i represents the shortest completion time of the i-th service; $aveETC$ represents the average value of the shortest completion time of all services in the service set.

Based on the description of the algorithm, a primary pre-scheduled resource allocation algorithm based on standard deviation is described as follows:

Input: task set T_i, edge calculation server S_m, ETC matrix

Output: task set completion time, working time of each edge server

```
1) FOR for each business in the task set
2)     FOR for each edge server
3)         $ETC_i = W_m + t_i$
4)     END FOR
5) END FOR
6) Find the minimum expected completion time for each business, and sort by increasing time size
7) Calculate the standard deviation SD
```

8) finding the difference between the expected execution time of two consecutive services in the service set to be completed is greater than the position p of the standard deviation SD, determining whether p is in the first half of the service set, if it is jumping to step 9); if not jump to Step 10)

9) The min-min algorithm is used to allocate the service, and the service is deleted from the to-be-assigned service set.

10) Allocating the service by using the max-min algorithm, and deleting the service from the set of services to be allocated

11) Determine whether all services have been assigned, if it is to step 12); if not jump to step 1)

12) End

3.2 Resource allocation algorithm based on redistribution edge computing server

After completing the service priority grouping and pre-scheduling, the problem of load imbalance may still be faced. The algorithm on the edge computing server with the longest execution time tries to allocate to other resources through the secondary allocation algorithm. For other resources, you can reduce the completion time and then allocate it to achieve load balancing. The specific operation of the secondary allocation is as follows:

(1) The services of the heavy load edge computing server are arranged in ascending order according to the execution time.

(2) If the service with the least execution time on the heavy load edge calculation server is allocated to other edge calculation servers, the load time of all edge calculation servers is updated.

According to the description of the algorithm, the resource allocation algorithm based on redistribution is described as follows:

Input: task set T_i, edge calculation server S_m, ETC matrix

Output: task set completion time, working time of each edge server

```
1) FOR for all services on the edge computing server with the longest execution time
2)     FOR for other m-1 edge computing servers
3)         IF (If the allocation is completed, the system completion time does not increase, and the execution time of each edge calculation server is not greater than the system completion time)
4)             Rescheduling the business set
5)             Update the working time of each edge computing server $W_m$
6)             END IF
7)     END FOR
8) Whether to traverse all services, if yes then jump to step 9); otherwise jump to step 1)
9) END FOR
```

After outputting the "each edge server working time", calculate the standard deviation of working time of each edge computing server, as an indicator of load balancing, the formula is as follows:

$$SD = \sqrt{\frac{\sum_{j=1}^{M} W_j - aveW}{M}} \tag{3}$$

where M represents the total number of edge computing servers; W_j represents the ready time of the jth edge computing server; $aveW$ represents the ready time average of all edge computing servers.

4 Simulation and analysis

4.1 Simulation condition

In order to deal with the complexity and variability of the actual network environment, this paper should fully consider the heterogeneity of the service and edge computing server resources to the very important impact of the edge computing business scheduling algorithm when simulating the real network environment. That is, fully consider the performance of the scheduling algorithm under the interaction of the two. In this paper, the simulation of the scheduling strategy is completed by the MATLAB experimental platform. The comparison experiments are carried out on 6 groups of services with 30-80 data, and the relevant parameters of the simulation platform are set according to the empirical values.

Based on the offloading of services to the edge computing server, the proposed algorithm is compared with the traditional min-min scheduling algorithm and the high priority (TPMM) scheduling strategy. The TPMM scheduling algorithm allocates resources according to the priority of the service in descending order. It can always ensure that the higher priority service can calculate enough processing resources and perform a certain degree of load balancing optimization. In the simulation, the calculation time consumed by different scheduling strategies and the load of the edge computing server with different computing capabilities are compared when processing the same amount of data.

4.2 Result analysis

On the premise of low heterogeneity services and server computing resources, the time consumed by the three service scheduling strategies and the load balancing standard deviation are shown in Figure 2. By comparison, the scheduling algorithm adopted by the main algorithm consumes the least time during the whole process of increasing the number of services from 30 to 80, followed by the min-min algorithm, and the TPMM algorithm consumes the most time. However, in terms of server load balancing, the TPMM algorithm performs better than the min-min algorithm. This is because the algorithm proposed in this paper considers the shortest completion time standard deviation of the service waiting queue, so that the proportion of the long and short tasks in the entire service set can be determined, so that the min-min or max-min algorithm can be dynamically used in a pre-scheduling to deal with different situation. At the same time, the secondary scheduling mechanism is supplemented to further reduce the delay and optimize the load balancing.

Under the premise of high heterogeneity business and server computing resources. The time consumed by the three service scheduling strategies and the standard deviation of the load balancing are shown in Figures 3. In this case, the min-min algorithm performs poorly, the load balancing standard deviation increases sharply, and even several edge computing servers are idle for a long time, and the resource utilization rate is seriously degraded. The TPMM algorithm allocates services based on the minimum delay while ensuring load balancing, and adapts the service priority. Therefore, the algorithm will increase in the time delay index, but get a lower load balance standard deviation. The algorithm proposed in this paper improves the resource utilization and reduces the working time because of the dynamic selection of the pre-allocation algorithm and the secondary allocation strategy.

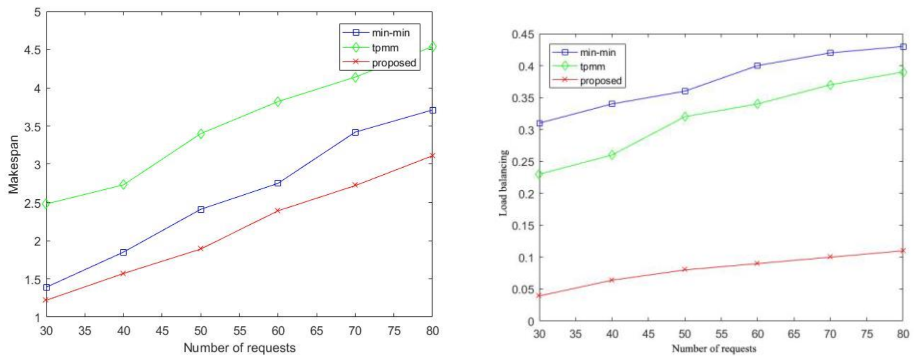

Fig. 2. Comparison chart under low heterogeneity conditions:(a)Makespan(b)Load balance.

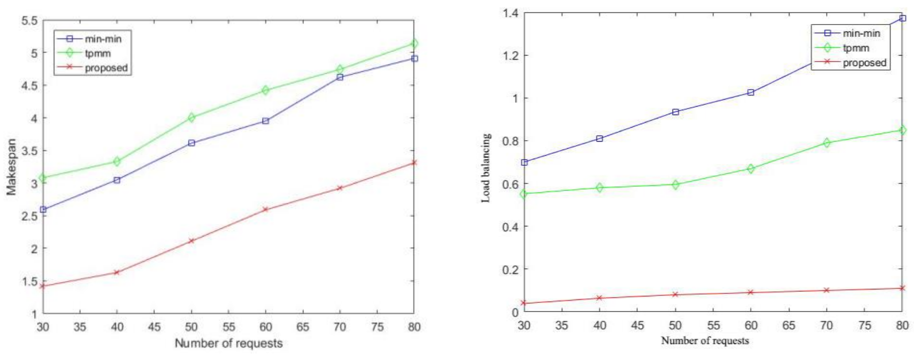

Fig. 3. Comparison chart under high heterogeneity conditions: (a)Makespan(b)Load balance.

5 Conclusion

Edge computing integrates network, computing, storage, etc. on the edge of the network to provide intelligent services. The task scheduling algorithm in the edge computing environment is the core problem in the research of edge computing technology, which has received special attention from researchers. This paper focuses on the performance of grid task scheduling algorithm in terms of computational performance and load balancing. Based on the shortcomings of min-min scheduling algorithm, a service scheduling algorithm based on task grouping and quadratic allocation is proposed. Experimental results demonstrate the effectiveness of the proposed algorithm. The experimental results show that the algorithm achieves the effect of reducing the service scheduling delay, reducing the server load balance standard deviation and improving the resource utilization rate by optimizing twice before and after.

References

1. Lin Tao, Qin Dongyang, Ma Tongkuan, et al. Research on Task Scheduling of Mobile Edge Computing Based on Game Theory[J]. Computer Simulation, 35(11): 399-403.
2. Liu Guoqiang. Research on task unloading strategy based on mobile edge computing [D]. Harbin Institute of Technology.

3. Juan Liu, Yuyi Mao, Jun Zhang, et al. Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing System [C].IEEE International Symposium on Information Theory (ISIT) . Barcelona: IEEE, 2016: 1451-55.
4. Olga Munoz, Antonio Pascual-Iserte, and Josep Vidal. Joint Allocation of Radio and Computational Resources in Wireless Application Offloading [C].Future Network and Mobile Summit. Lisboa: IEEE, 2013: 1-10.
5. Jing Zhang, Weiwei Xia, Feng Yan, Lianfeng Shen. Joint Computation Offloading and Resource Allocation Optimization in Heterogeneous Networks With Mobile Edge computing [J].IEEE Access, 2018, volume6: 19324-19337.
6. Qiao Nannan, You Jiali. A Multi-Directional Particle Swarm Optimization Algorithm for Network Edge Task Scheduling Problem[J]. Journal of Computer Applications and Software(4).