Journal of Computers Vol. 32 No. 5, 2021, pp. 232-243
doi:10.53106/199115992021103205020

Node Resource Management Model of Hierarchical Edge Computing

Zhi-Bo Wang*

China Shenhua Energy Company Limited Shenshuo Railway Branch, Shenmu, 719316, China
11255894@chnenergy.com.cn

Received 1 August 2021; Revised 1 September 2021; Accepted 8 October 2021

Abstract. This article focuses on hierarchical edge computing in the industrial Internet of Things scenario, and studies the static resource deployment of hierarchical edge computing networks. When deploying a hierarchical edge computing network with new equipment, the allocation of computing capacity between layers is one of the hot is-sues. This paper proposes a method for the allocation of edge computing node capacity between layers based on the M/M/1/c queue model, delay and call loss are performance indicators, and the optimal inter-layer capacity allocation algorithm is designed and implemented. This algorithm can reduce the global average delay of the sys-tem under the premise of meeting the requirement of call loss rate. Simulation verification shows that the optimal inter-layer capacity allocation algorithm can effectively reduce the system's global average delay and call loss rate under the condition of a certain total system cost.

Keywords: edge computing, Industrial Internet of Things, network deployment

1 Introduction

1.1 Research Background

In recent years, with the development of Internet of everything, low latency applications and other needs, the concept of edge computing has emerged. Edge computing means that the computing function is deployed at the edge of the network, so it has a better response time, can ensure the delay requirements of low-latency applications, and at the same time inhibit the spread of the original and large amount of data in the network, and then it provides new possibilities for reducing the bandwidth pressure of the core network and improving the level of data security and privacy.

As early as 2012, Flavio Bonomi et al. [1] proposed the concept of fog computing and analyzed its role and application in the Internet of Things. Its core idea is to decentralize cloud computing capabilities, and use distributed equipment to deploy computing capabilities in a network closer to the user side, so as to solve cloud computing's latency and mobility issues. S. Yi et al. [2] further discussed the definition, goals and challenges of fog computing, and carried out examples and practices based on actual scenarios. M. Chiang et al. [3] and A. V. Dastjerdi et al. [4] analyzed the advantages and problems of fog computing in IoT scenarios.

For edge computing, W. Shi et al. [5-6] combined actual scenarios to put forward a wealth of requirements, including delay, device battery life, bandwidth pressure, security and privacy, etc., and clarified the goals and characteristics of edge computing. In the research of edge computing, its application scenarios are not only for the Internet of Things, 5G is also one of the key application scenarios, so the mobile edge computing that focus-es on the 5G scenario is proposed. Y. Mao et al. [7] focused on the analysis of MEC's design goals, key issues and some typical application schemes. P. Mach et al. [8] focused on investigating the structural design and task offloading of MEC. S. Wang et al. [9], N. Abbas et al. [10] and A. Ahmed et al. [11] discussed the key technologies and future directions in

* Corresponding Author

MEC, related research and technology development, and promising application scenarios. M. Satyanarayanan described [12] with the popularity of edge computing, the improvement of service quality it can bring in different application scenarios.

1.2 Typical Application Scenarios in Edge Computing

Edge computing has a variety of application scenarios, such as MEC, Internet of Vehicles, Industrial Internet of Things, and so on. In scenarios such as MEC, edge nodes are often inseparable from base stations. Therefore, scholars have adopted the practice of adding servers to base stations to make them into small edge clouds; In the scene of Internet of Vehicles, edge nodes are often provided by on-board computers or roadside equipment. Due to the mobility of automobiles, the Internet of Vehicles also poses new challenges to the mobility of corresponding edge computing; In the industrial Internet of Things, edge computing is mainly composed of smart gateways, smart routers or small servers set up. The industrial Internet of Things is relatively mobile, but has a large amount of data from non-smart sensors or smart terminals. This requires that the edge computing network needs to have good carrying capacity and will not cause rapid expansion of system delay due to congestion.

D. A. Chekired et al. [13] highlighted the advantages of hierarchical fog computing in industrial IoT data scheduling. They first analyzed the industrial IoT scene in a large factory and built a hierarchical computing network. Using probabilistic analysis, they concluded that the hierarchical structure is more efficient than the flat structure. They designed a queue strategy and an offloading plan, and finally performed a simulation analysis using data from the real environment, which proved that compared with the traditional flat edge computing network, the performance of the hierarchical structure is better.

1.3 Research Content and Main Work

This paper is oriented to the industrial Internet of Things scene in edge computing, and studies the static deployment of hierarchical edge computing networks. The main content of the work is as follows: Firstly, the M/M/1/C queuing model closer to the characteristics of the scene is used to solve the problem of inter layer computing capacity allocation; Secondly, a method to calculate the global average delay of the system is proposed to optimize the global average delay of the system; Thirdly, the use of the stable distribution of Markov chains to calculate the global call of the system is proposed. The loss rate scheme has better statistical characteristics; Finally, an optimal inter-layer capacity allocation algorithm is proposed to find an approximate solution to the calculation of the inter-layer capacity allocation problem, and to reduce the call loss rate and global average of the system Time delay.

2 Related Basic Theories

2.1 Markov Chain

Markov chain can be divided into continuous-time Markov chain (CTMC) and discrete-time Markov chain (DTMC). It is a series of relational representations of state spaces, and it is also a random process. The most important thing about the Markov chain is the Markov property, that is, in a random process, it is first assumed that his current and past historical moments are known, and the probability distribution of the various states that it may transfer to in the future is only related to the state at the present moment, and has nothing to do with the state in the past. Markov chains may have properties such as recurrence, irreducibility, and periodicity.

Fig. 1 is a simple Markov chain. The Markov chain in the figure has two states, 0 and 1. Among them, state 0 will transition to state 1 with probability p, and it will still be state 0 after transition with probability 1-p. State 1 will transition to state 0 with probability q, and will remain in state 1 after transition with probability 1-q. Obviously, the sum of the probabilities of the paths transitioning from a state should be 1. In the figure, the sum of the probabilities of transition from state 0 to the outside and transition from state 1 to the outside are both 1.

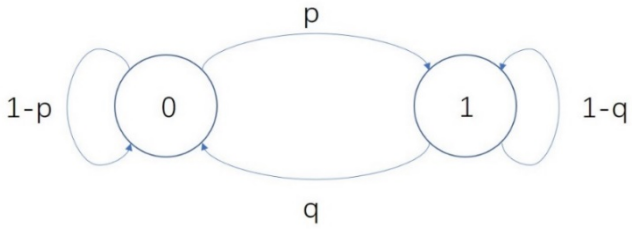

Fig. 1. Example of Markov Chain

If a Markov chain has the characteristics of irreducibility, normal return, and aperiodic at the same time, then this Markov chain has ergodicity and is also called an ergodic chain. The ergodic chain has steady-state characteristics on a long-term scale.

The stationary distribution of a Markov chain is defined as if there is a probability distribution $\pi = \pi(S)$ in each state, and the conditions in Fig. 2 are met, then π is a stationary distribution.

$$\forall s_j \in \boldsymbol{s}:\ \pi(s_j) = \sum_{s_i \in \boldsymbol{s}} \pi(s_i) P_{i,j} \quad \Longleftrightarrow \quad \boldsymbol{\pi} = \boldsymbol{\pi P}, \quad 0 < \pi(s_i) < 1,\ \|\boldsymbol{\pi}\| = 1$$

Fig. 2. Conditions of stationary distribution of Markov Chain

2.2 Queuing Theory

Queuing theory is a branch of operations research, focusing on the work process of random service systems, so it is also called the theory of random service systems. The main content of queuing theory is to study the relation-ship between variables such as waiting time, service time, waiting queue length and the system structure, customer arrival rate, and service rate that the system can provide in a random service system. The analysis results of queuing theory with statistical characteristics provide accurate and good support for the analysis of system performance.

3 Inter-layer Computing Capacity Allocation Method for Hierarchical Edge Computing Nodes

3.1 Inter-layer Capacity Allocation Model Based on M/M/1/c Queue

The model is shown in Fig. 3. The edge computing node in the figure is divided into two layers. Among them, the first layer of nodes, we can call it shallow nodes, shallow nodes are closer to the edge of the network; the second layer of nodes, that is, deep nodes, are located deeper in the network, closer to the cloud. Deep nodes and shallow nodes may be homogeneous devices or heterogeneous devices. μ represents the total computational capacity cost that can be allocated in a pair of shallow-deep connections, and m represents the number of shallow nodes directly connected to a deep node in the initial situation, m = 2 in the figure. For μ, we have the following conditions.

$$\mu_1 = (1-\alpha)\mu \tag{1}$$

$$\mu_2 = \alpha m \mu \tag{2}$$

$$\lambda = \lambda_1 \tag{3}$$

λ represents the arrival rate of a data stream connected to this edge computing network, and λ_1 is the arrival rate of the first layer node, that is, the shallow node. α is the capacity allocation coefficient, which represents the plan result and implementation plan for calculating the capacity allocation. Obviously we have the following constraints.

$$0 < \alpha < 1 \tag{4}$$

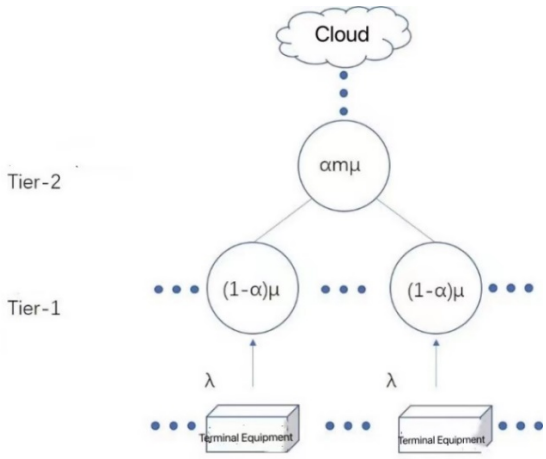

Fig. 3. Two-Layer model of Hierarchical Edge Computing

Let's study the model of a single edge computing node. Before introducing the practical significance of each parameter, first give an overview of the node's own queue model, as shown in the figure below.

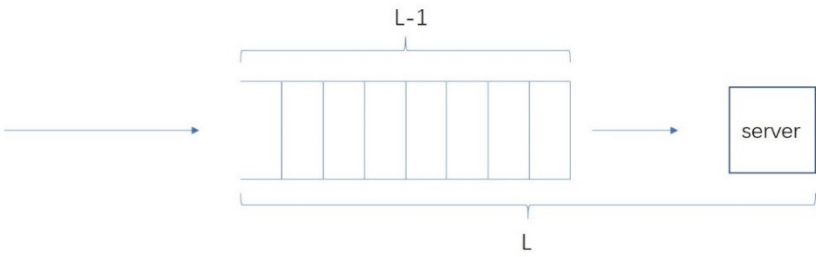

Fig. 4. Queueing Model of Edge Computing Node

In the Fig. 4, L-1 represents the queue length provided by the node itself, and L represents the total capacity of the node. N represents the number of tasks stuck in the node. t_1 and t_2 represent the residence time of tasks in nodes of different layers. Use D to represent the maximum delay that can be accepted in the system. If the detention time of the task at the end of the full cache queue is to be less than the maximum delay of the system, the following constraints apply.

$$\frac{L}{\mu} \le D \tag{5}$$

Considering the link delay t_L caused by routing and forwarding during task offloading, assuming that C_L is the maximum number of offloads allowed in the system, the buffer queue length of the deep-level node needs to meet the following conditions.

$$\frac{L}{\mu} + C_L t_L \le D \tag{6}$$

Set the node cache queue length in the two layers to be the same size, and stipulate that only one offload is allowed from the shallow layer to the deep layer. So L can be determined.

$$L \le \mu(D - t_L) \tag{7}$$

3.2 Calculation Method of System Call loss Rate and Global Average Delay Based on Stable Distribution

The discrete-time Markov chain of the node model is shown in the Fig. 5 below. The parameters are represented by the first-level node as an example.

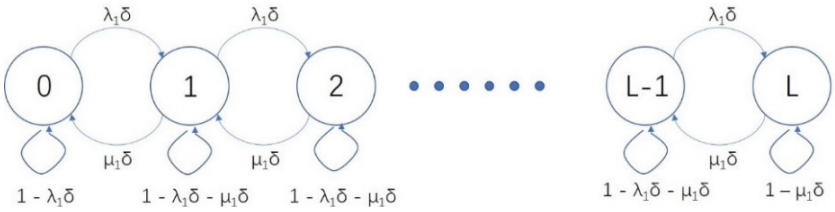

Fig. 5. Discrete-Time Markov Chain of Node Model

Each circle in the figure represents a state, and the value in it represents the number of tasks in the system. δ represents a very small time interval, and $\lambda_1\delta$ and $\mu_1\delta$ indicate the probability of a new arrival task and the probability of completing a task. According to Fig. 5, the local equilibrium equation of this Markov chain and the recursive representation of each state can be obtained.

$$P_k = \frac{\lambda_1}{\mu_1} P_k - 1 \tag{8}$$

$$P_k = \left(\frac{\lambda_1}{\mu_1}\right)^k P_0 \tag{9}$$

Where P_k represents the steady-state probability of state k, and λ_1 and μ_1 are the node task arrival rate and service rate. Substituting the recurrence representation of each state into the probability constraint condition can solve P_0, and P_k can be expressed by P_0 and the recurrence relationship.

$$P_k = \frac{\lambda_1^{\ k}}{\mu_1^{\ k}} = \frac{\lambda_1^{\ k}}{\mu_1^{\ k}} \cdot \frac{1 - \frac{\lambda_1}{\mu_1}}{1 - \left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \tag{10}$$

In the same way, similar expressions can be derived for the second-level nodes.

$$P_k' = \frac{\lambda_2^{\ k}}{\mu_2^{\ k}} P_0' = \frac{\lambda_2^{\ k}}{\mu_2^{\ k}} \cdot \frac{1 - \frac{\lambda_2}{\mu_2}}{1 - \left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \tag{11}$$

In the first-level node, the probability of task offloading is $P_{offload1}$, which is also the probability of the first-level node being cached full. So $P_{offload1}$ can be represented by the P_L state of the Markov chain.

$$P_{offload1} = P_L = \frac{\lambda_2^{\ L}}{\mu_2^{\ L}} \cdot \frac{1 - \frac{\lambda_1}{\mu_2}}{1 - \left(\frac{\lambda_1}{\mu_1}\right)^{L+1}} \tag{12}$$

Substituting the actual values of λ_1 and μ_1, the function of $P_{offload1}$ with respect to α is obtained.

$$P_{offload1}(\alpha) = \frac{\lambda^{\ L}}{1 - (\alpha)^L \mu^L} \cdot \frac{1 - \frac{\lambda}{(1-\alpha)\mu}}{1 - \left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \tag{13}$$

In the same way, the $P_{offload2}$ of the second-level node can be represented by the P'_L state.

$$P_{offload2} = P'_L = \frac{\lambda_2^{\ L}}{\mu_2^{\ L}} \cdot \frac{1-\frac{\lambda_2}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \tag{14}$$

Substituting the actual values of λ_1 and μ_1, the function of $P_{offload2}$ with respect to α is obtained.

$$P_{offload2}(\alpha)_L = \frac{\lambda_2^{\ L}}{\mu_2^{\ L}} \cdot \frac{1-\frac{\lambda_2}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}}$$

$$= \left(\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right) \cdot \frac{1-\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}}}{1-\left(\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right)^{L+1}} \tag{15}$$

The average number of tasks of the first-level nodes can be calculated from the distribution of P_k.

$$N_1 = \sum_{k=0}^{L} k \cdot P_k = \sum_{k=0}^{L} \frac{k\lambda^k}{(1-\alpha)^k \mu^k} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \tag{16}$$

According to Little's theorem, there are the following relationships.

$$N_1 = \lambda(1-P_{offload1}) \cdot t_1 \tag{17}$$

Call $\lambda(1\text{-}P_{offload1})$ the effective arrival rate of the first-level node. Therefore, the average task retention time in the first-level nodes can be expressed.

$$t_1 = \frac{N_1}{\lambda(1-P_{offload1})} \tag{18}$$

$$t_1(\alpha) = \sum_{k=0}^{L} \frac{k\lambda^k}{(1-\alpha)^k \mu^k} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \cdot \left(\lambda - \frac{\lambda^{L+1}}{(1-\alpha)^L \mu^L} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right)^{-1} \tag{19}$$

In the same way, the average number of tasks of the second-level nodes can be calculated from the distribution of P'_k. From Little's theorem, formula (21) can be obtained.

$$N_2 = \sum_{k=0}^{L} k \cdot P'_k = \sum_{k=0}^{L} \frac{k\lambda_2^{\ k}}{\mu_2^{\ k}} \cdot \frac{1-\frac{\lambda}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \tag{20}$$

$$t_2 = \frac{N_2}{\lambda_2(1-P_{offload2})} \tag{21}$$

The residence time of the task in the second-level node can be expressed.

$$t_2(\alpha) = \sum_{k=0}^{L} \frac{k{\lambda_2}^k}{{\mu_2}^k} \cdot \frac{1-\frac{\lambda_2}{\mu_2}}{1-\left(\frac{\lambda_2}{\mu_2}\right)^{L+1}} \cdot (\lambda_2 - \lambda_2 \cdot P_{offload2})^{-1}$$

$$= \sum_{k=0}^{L} k \left(\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right)^k \cdot \frac{1-\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}}}{1-\left(\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right)^{L+1}}$$

$$\cdot \left(\frac{m\lambda^{L+1}}{(1-\alpha)^L \mu^L} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \cdot (1-P_{offload2}(\alpha)) \right)^{-1} \tag{22}$$

After expressing the unloading probability and loss rate of each layer node as a function of calculated capacity distribution coefficient between layers α, the global call loss rate of the system can be obtained.

$$P_{loss}(\alpha) = P_{offload1}(\alpha) \cdot P_{offload2}(\alpha)$$

$$= \left(\frac{\lambda^{L+2}}{\alpha(1-\alpha)^{L+2} \mu^{L+2}} \right)^L \left(\frac{1-\frac{\lambda}{(1-\alpha)\mu}}{n\left(1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}\right)} \right)^{L+1} \cdot \frac{1-\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}}}{1-\left(\frac{\lambda^{L+1}}{\alpha(1-\alpha)^L \mu^{L+1}} \cdot \frac{1-\frac{\lambda}{(1-\alpha)\mu}}{1-\left[\frac{\lambda}{(1-\alpha)\mu}\right]^{L+1}} \right)} \tag{23}$$

According to the unloading rate and the system call loss rate of each layer node, the probability that the task is distributed in the whole system can be expressed. Thus, the global average delay of the system can be expressed.

$$T(\alpha) = P_{offload1} \cdot (t_L + t_2) + (1 - P_{offload1}) \cdot t_1 \tag{24}$$

3.3 Optimal inter-layer Capacity Allocation Algorithm

In this section, the optimal interlayer capacity allocation algorithm is designed, the task objective is divided into two sub problems and solved in order, as shown in Table 1 and Table 2, respectively. First, an improved gradient descent method is used to solve the reasonable range of α. Algorithm 1 shows the solution process.

Table 1. Constraints Conversion based on Gradient Descent

Algorithm 1. Constraint conversion based on gradient descent

Input: function $P_{loss}(\alpha)$, step size r, boundary point ε, limit the number of iterations n

Output: α_start, α_end

$P'(\alpha) = \mathrm{d}P_{loss}(\alpha)/\mathrm{d}\alpha$

$\alpha = \varepsilon$

k = 0

while $P_{loss}(\alpha)$ > 10-3

$\alpha = \alpha - 1 * P'(\alpha)$

k++

if k > n

Throws an exception: Cannot find a qualified α.

end

while $P_{loss}(\alpha)$ < 10-3

$\alpha\_old = \alpha$

$\alpha = \alpha - r$

end

$\alpha\_start = \alpha\_old$

$\alpha = 1 - \varepsilon$

while $P_{loss}(\alpha)$ > 10-3

$\alpha = \alpha - r * P'(\alpha)$

end

while $P_{loss}(\alpha)$ < 10-3

$\alpha\_old = \alpha$

$\alpha = \alpha + r$

end

$\alpha\_end = \alpha\_old$

Secondly, in order to minimize the global average delay of the system, corresponding algorithms need to be designed. In this section, a gradient descent method is designed to search from around 0 point, and the constraint condition of the independent variable α is added to find the minimum value of T(α).

Table 2. Gradient Descent with Constraints

Algorithm 2. Gradient descent method with additional constraints

Input: Time delay function $T(\alpha)$, α_start, α_end step length r, allowable error ε

Output: α _solution

$T'(\alpha) = dT(\alpha)/d\alpha$

$\alpha = \alpha\_start$

$\alpha\_old = 0$

while $\alpha \geq \alpha\_start$ && $\alpha \leq \alpha\_end$ && $|\alpha - \alpha_{old}| > \varepsilon$

$\alpha\_old = \alpha$

$\alpha = \alpha - r * T'(\alpha)$

end

α _solution = α

The two algorithms of the constraint conversion based on gradient descent shown in Table 1 and the gradient descent method with increasing constraint conditions shown in Table 2 together form the optimal inter-layer capacity allocation algorithm.

4 Simulation Process and Result Analysis

The main task of this section is to simulate the designed algorithm and analyze the results. There are two main measurement indicators: system global average delay and system call loss rate. The main variables of the simulation in this section are μ/λ, the size of the node cache L, the connection ratio m between the nodes of the system network layer, and the link delay t_L. Measure the impact of the above variables on the system's global average delay and system call loss rate through simulation.

In the Fig. 6, it can be seen that in terms of the system's global average delay and call loss rate, the optimized calculation capacity allocation scheme is better than the node equalization without allocation. Similarly, with the increase of μ/λ, the global average delay of the system and the system call loss rate both decrease, which is also intuitive.

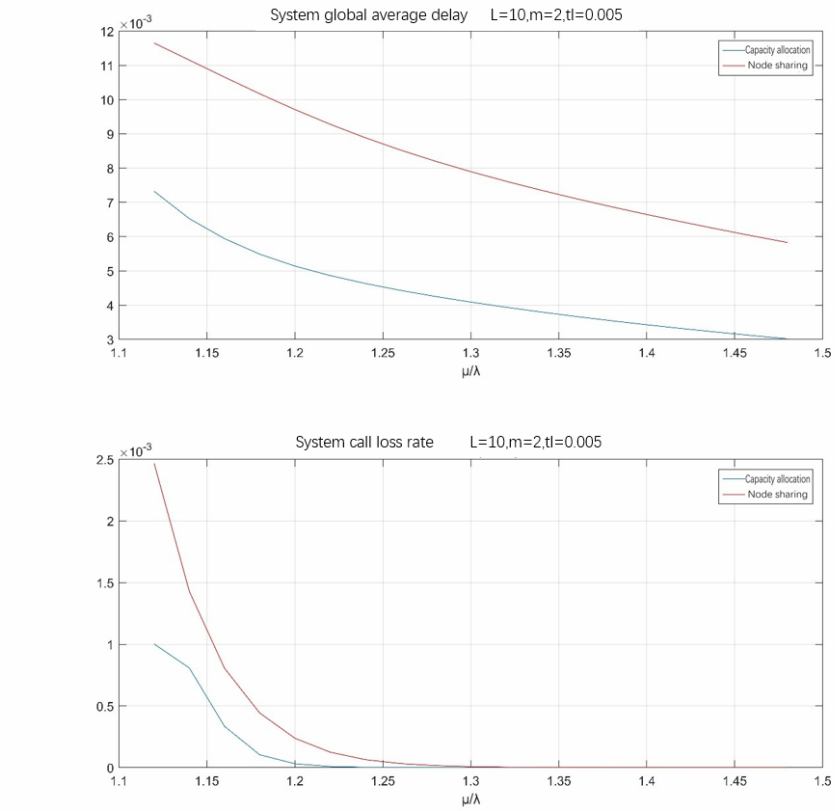

Fig. 6. Relation between Global Average Delay & Loss and μ / λ

As shown in Fig. 7, within the allowable range of system delay, the larger the cache, the larger the global average delay of the system, which is mainly caused by the queuing delay of tasks in the node. However, if the cache is too small, it will affect the call loss rate, because the expectation of the arrival rate of the task is stable, it is not a definite distribution. When the instantaneous arrival rate fluctuates, the load capacity of the cache is too small to cope with the peak, serious task loss will occur.

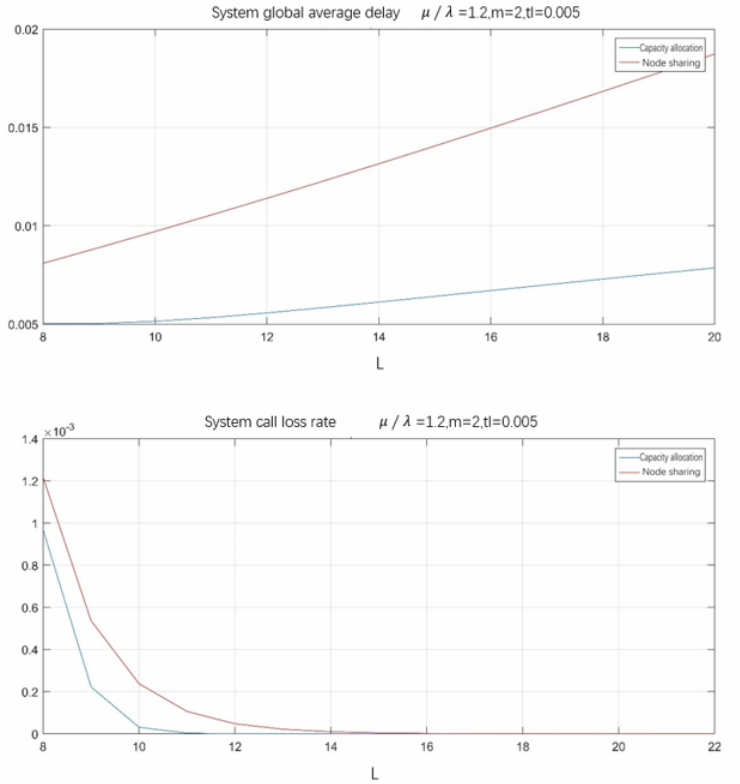

Fig. 7. Relation between Global Average Delay & Loss and *L*

The effect of inter-layer node connection ratio on the system's global average delay and call loss rate is completely different. As Fig. 8 indicates, with m increases, the system's global average delay decreases. Intuitively, it can be understood that more nodes participate in resource recovery. This makes the deployment of global hardware resources more ideal.

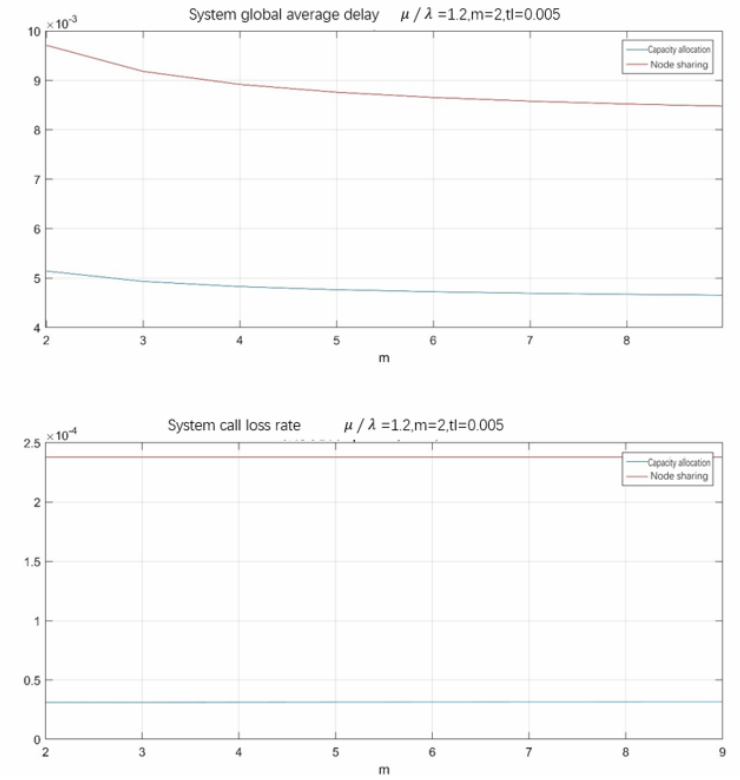

Fig. 8. Relation between Global Average Delay & Loss and *m*

As illustrated in Fig. 9, obviously, the link delay should not affect the call loss rate of the system, which has also been verified in the simulation. On the other hand, it can be found that the system global average delay of the capacity allocation scheme is more insensitive to the link delay. With the increase of link delay, the growth rate of the system global average delay of the node equalization is faster, and the growth rate of the system global average delay of the capacity allocation scheme is lower.

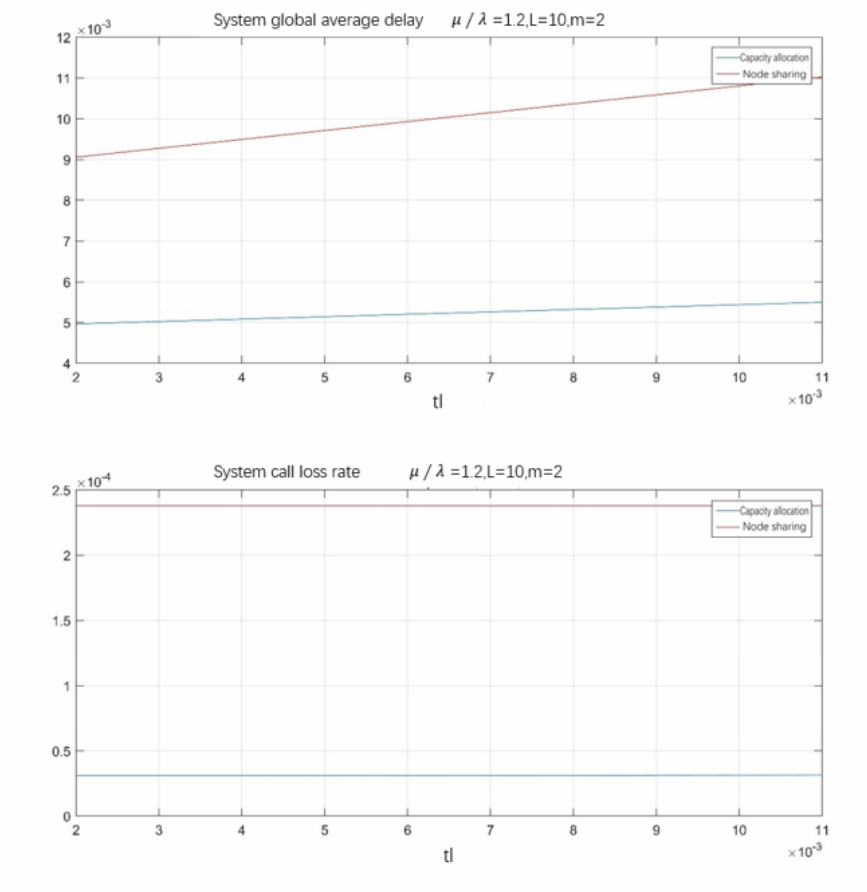

Fig. 9. Relation between Global Average Delay & Loss and t_L

5 Conclusion

This paper uses the Industrial Internet of Things as an application scenario, and aims at the static resource deployment problem of hierarchical edge computing, and proposes a new method for the allocation of computing capacity between hierarchical edge computing nodes. Firstly, an interlayer capacity allocation model based on M/M/1/C queue is established; Secondly, the calculation method of system call loss rate and global average delay based on stationary distribution is proposed, and the global average delay and call loss rate are calculated accurately; Finally, the optimal interlayer capacity allocation algorithm is proposed, which decomposes the problem into two sub-problems, and designs an algorithm to solve the inter layer capacity allocation scheme based on the gradient descent method. The experimental results show that after the optimization of capacity allocation, the system global average delay and call loss rate of the hierarchical edge computing network are reduced.

Acknowledgements

This research was supported by the Technology innovation project of Shenshuo Railway Branch, China.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proc. Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012.

[2] S. Yi, Z. Hao, Z. Qin and Q. Li, Fog Computing: Platform and Applications, in: 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies, 2015.

[3] M. Chiang, T. Zhang, Fog and IoT: An Overview of Research Opportunities, IEEE Internet of Things Journal 3(6)(2016) 854-864.

[4] A. V. Dastjerdi, R. Buyya, Fog Computing: Helping the Internet of Things Realize Its Potential, Computer 49(8)2016 112-116.

[5] W. Shi, S. Dustdar, The Promise of Edge Computing, Computer 49(5)(2016) 78-81.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, IEEE Internet of Things Journal 3(5)(2016) 637-646.

[7] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, A Survey on Mobile Edge Computing: The Communication Perspective, IEEE Communications Surveys & Tutorials 19(4)(2017) 2322-2358.

[8] P. Mach, Z. Becvar, Mobile Edge Computing: A Survey on Architecture and Computation Offloading, IEEE Communications Surveys & Tutorials 19(3)(2017) 1628-1656.

[9] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, W. Wang, A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications, IEEE Access (5)(2017) 6757-6779.

[10] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile Edge Computing: A Survey, IEEE Internet of Things Journal 5(1)(2018) 450-465.

[11] A. Ahmed, E. Ahmed, A survey on mobile edge computing, in: 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016.

[12] M. Satyanarayanan, The Emergence of Edge Computing, Computer 50(1)(2017) 30-39.

[13] D. A. Chekired, L. Khoukhi, H. T. Mouftah, Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory, IEEE Transactions on Industrial Informatics 14(10)(2018) 4590-4602.