Computing Capacity Allocation for Hierarchical Edge Computing Nodes in High Concurrency Scenarios Based on Energy Efficiency Evaluation

Ziheng Zhou[1], Zhenjiang Zhang[1](✉), Jianjun Zeng[2], and Jian Li[1]

[1] Beijing Jiaotong University, No.3 Shangyuancun, Beijing, China
zhangzhenjiang@bjtu.edu.cn
[2] Beijing Li'antong Information Technology Company, Beijing, China

Abstract. Edge computing could play an important role in Internet of Things (IoT). Computing capacity allocation has been researched a lot in mobile edge computing, which is task oriented. However, hierarchical edge computing also needs computing capacity allocation which is node oriented. This paper focusses on capacity allocation of nodes in hierarchical edge computing. We take energy efficiency and loss in high concurrency scenarios into consideration and work out a method to do allocation by weighing loss and energy efficiency. Simulation is under circumstances that nodes overload, which means that loss is inevitable. A new inspiration of deployment is also given after simulation.

Keywords: Edge computing · Energy efficiency · Computing capacity

1 Introduction

Edge computing should be deployed at the edge of the network in order to get better latency and reduce the pressure of core network. However, edge hold less computing capacity compared to cloud, which means that we need strategy to make full use of edge rather than appending servers like cloud. Network structure optimization might be a good idea before deploying. With appropriate deployment, we could use less power and weaker devices to satisfy the latency and data processing requirements.

A. Kiani, N. Ansari and A. Khreishah proposed a hierarchical structure for fog computing [1], they made a lot of work to prove and simulate. Inspired by them, we will move on to find a more specific solution of deployment in hierarchical edge computing. We focus on loss ratio and energy efficiency to make a strategy, so as to adjust a fixed-structure hierarchical edge network by modifying their computing capacity allocation. Assuming that we need some CPU cycles to deal with data which arrives continuously, different layers in this hierarchical network will provide diverse and distinct computing capacity, namely different CPU frequencies or service rate. By weighing loss and energy efficiency, we figure out a plan to do capacity allocation before deployment. Simulations are also given.

2 Related Work

A. Kiani et al. [1] investigated how and where at the edge of the network the computation capacity should be provisioned, and proposed a hierarchical model. They use queueing theory to analyze latency and solve the problem by stochastic ordering and upper bound-based techniques.

Y. Li et al. [2] studied the edge server placement problem in mobile edge computing. They focus on energy consumption and devise a PSO (particle swarm optimization) [3, 4] to find the solution. Z. Liu et al. [5] found that cooperation can contribute to the sum-capacity of a hybrid network. P. Yuan et al. [6] studied the capacity of edge caching systems. Y. Lin et al. [7] proposed a problem that how to allocate the traffic and capacity in the management plane in mobile edge computing. Queueing theory and latency are concerned. M. Noreikis et al. [8] focus on edge layer capacity estimation and provide a capacity planning solution which considers QoS requirements. H. Badri et al. [9] proposed a risk-based optimization of resource provisioning in mobile edge computing. M. Liu et al. [10] proposed a price-based distributed method to manage the offloaded computation tasks.

3 System Model

Hierarchical edge coming contains two types of computing nodes. Shallow nodes are deployed near the devices while deep nodes contact with a set of shallow nodes. Data generated from devices flow from shallow nodes to deep nodes, which probably reach cloud eventually.

Data from devices could need to be computed and each piece of such data can be casted to a task. Computing service could be provided from shallow nodes and deep nodes in the meantime because there could be an offloading scenario on the shallow nodes.

For shallow node i, λ_i is the arrival rate. If shallow nodes could not process the task timely, this task will be casted to deep nodes so that shallow nodes could handle the next task. So that the whole server rate is divided into two parts. Assuming that we need μ_t in total, $(1-\alpha)\mu_t$ is deployed on shallow node and $\alpha\mu_t$ is on deep node. It is noted that the shallow rate is $(1-\alpha)\mu_t$ but deep one is not. A deep node is linked to several shallow nodes, and it should be $\alpha n \mu_t$ while n is the number that is linked. α is distribution coefficient.

$$\mu_s = (1-\alpha)\mu_t \tag{1}$$

$$\mu_d = \alpha n \mu_t \tag{2}$$

μ_s is the rate of shallow nodes while μ_d is the rate of deep ones. Regarding one deep node and its shallow nodes as a cluster, we focus on this cluster to solve the problem.

3.1 Loss

As previously noted, the arrival rate has an effect on the loss ratio. We assume that λ_i is known and certain over a period of time so that the problem is simplified, because we mean to decrease the loss rather than seeking an accurate solution of loss ratio. With a certain arrival rate λ_i and server rate μ_s,we can find the offloading work load λ_{ji} as a part of arrival rate of deep node j in time stationary, which is from shallow node i.

$$\lambda_{ji} = max[\lambda_i - (1-\alpha)\mu_t, 0] \tag{3}$$

Notice that a deep node is linked to several shallow nodes, the total arrival rate of deep node j is definite. C_{ij} is a coefficient to indicate whether shallow node i is linked to deep node j. In addition, it is clear that $\sum_j C_{ij} = 1$.

$$\lambda_j = \sum\nolimits_{\mathrm{i}} \{C_{ij} * max[\lambda_i - (1-\alpha)\mu_t, 0]\} \tag{4}$$

Focusing on this deep node and ignoring the situation that there is no loss, the lost portion is transformed into r. Thus, the lost portion on deep nodes can be expressed.

$$\begin{aligned} r &= \sum\nolimits_{\mathrm{i}} \{C_{ij} * max[\lambda_i - (1-\alpha)\mu_t, 0]\} - \alpha n \mu_t \\ &= \sum\nolimits_{k=1}^{m} [\lambda_k - (1-\alpha)\mu_t] - \alpha n \mu_t \\ &= \sum\nolimits_{k=1}^{m} \lambda_k - (1-\alpha) m \mu_t - \alpha n \mu_t \end{aligned} \tag{5}$$

$\sum_{k=1}^{m} \lambda_k$ is independent of α so that we could regard it as a constant K. Therefore, a function is built after normalization.

$$\begin{aligned} r(\alpha) &= \frac{K - (1-\alpha) m \mu_t - \alpha n \mu_t}{\sum \lambda} \\ &= \frac{K - m\mu_t + (m-n)\mu_t \alpha}{\sum \lambda} \end{aligned} \tag{6}$$

With $r(\alpha)$, we can take its derivative easily.

$$r'(\alpha) = \frac{(m-n)\mu_t}{\sum \lambda} \tag{7}$$

It is clear that $r'(\alpha)$ is always no more than 0, which means that $r(\alpha)$ is a non-increasing function. We could also get the similar conclusion with intuition. With more provisioning or capacity allocated to deep node, the extent of multiplexing goes higher among these edge nodes so that the lost portion will be reduced.

3.2 Energy Efficiency

M. Dayarathna et al. have proposed that CPU consumes the most power in a certain device [11]. And the energy consumption of a server in idle state accounts for more than 60% in full state [11, 12]. We could also get the similar result when it comes to some lightweight devices such as Raspberry Pi and other IoT devices. Thus, some algorithms try to decrease amounts of idle server and succeed.

However, the problem in this paper talks about a different case. Those excellent methods are not appropriate because we could not modify the location or amounts of edge nodes. The whole network structure is definite and we could only adjust capacity allocation to optimize energy efficient. The total power should consist of idle power and working power, where working power is generated from CPU utilization.

$$P_{total} = P_{idle} + P_{working} \tag{8}$$

P_{idle} is certain once we determine the network. $P_{working}$ can be reduced by adjusting CPU utilization. In other words, adjusting its capacity, which means CPU frequency as known as server rate μ, can decrease power consumption. The relation of CPU frequency and power is sometimes approximately linear in some cases. However, we need more accurate evaluation but not approximation because $P_{working}$ is the only thing that we can optimize.

We can regard CPU as a set of FET (Field Effect Transistor) or CMOS (Complementary Metal Oxide Semiconductor). And we can try to figure a function of power [13].

$$P = CV^2 f \tag{9}$$

P is the power consumption and we use C to indicate some other constant which is independent of frequency f. V is the supply voltage. It seems that f and P is linear, but noting that there is a factor that V does influence f. Frequency f could not keep going higher without the increase of V because of gate delay. To simplify the problem, we can construct a function to measure power consumption. The argument is μ which is affected by f and V. And C is just a constant to amplify so that we can ignore it.

$$p' = \mu^3 \tag{10}$$

It should be noted that p' is a parameter to measure the level of power consumption but not itself. Thus, we have a function to indicate energy efficiency as long as we use allocated capacity to replace μ.

$$\begin{aligned} p' &= n\mu_s^3 + \mu_d^3 \\ &= n(1-\alpha)^3\mu_t^3 + \alpha^3 n^3 \mu_t^3 \end{aligned} \tag{11}$$

μ_t is definite at the beginning so it is not a variable. In order to simplification and normalization, assume that $\frac{p'}{n^3\mu_t^3} = p$.

$$p(\alpha) = n^{-2}(1-\alpha)^3 + \alpha^3 \tag{12}$$

It is obvious that $p(\alpha)$ is a convex function.

3.3 Capacity Allocation

With $r(\alpha)$ and $p(\alpha)$, we can do allocation easily. In order to weigh loss and energy efficiency, we create a new function using another coefficient β.

$$\begin{aligned} G(\alpha) &= \beta * r(\alpha) + (1-\beta) * p(\alpha) \\ &= \beta * \frac{K - m\mu_t + (m-n)\mu_t \alpha}{\sum \lambda} + (1-\beta) * [n^{-2}(1-\alpha)^3 + \alpha^3) \end{aligned} \tag{13}$$

β indicates the balance of loss and energy efficiency which ranges from 0 to 1. It is clear that we could find α to make $G(\alpha)$ minimum with given β, n, k, K, μ_t by taking derivative because $G(\alpha)$ is the sum of a non-increasing function and a convex function (Fig. 1).

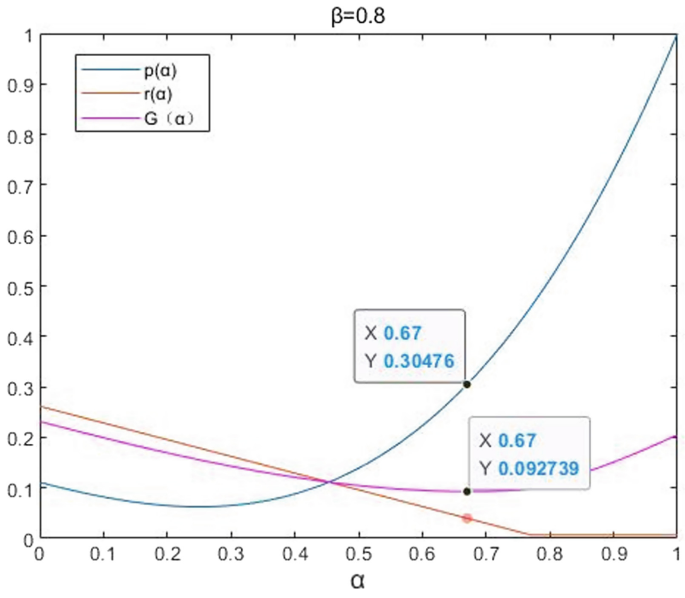

Fig. 1. $\boldsymbol{\beta} = 0.8, \boldsymbol{n} = 3$.

4 Simulation and Result Analysis

Taking $\mu_t = 1$ for instance, the upper bound capacity of this cluster is 3 because $\mu_{cluster} = n\mu_s + \mu_d$. In order to reveal loss, we make that $\sum \lambda > \mu_{cluster}$ so that there is always loss. λ is [1.57 0.23 1.22] in this experiment and $\sum \lambda = 3.02 > 3$.

When $\alpha = 0.67$, loss equals 0.03. Meanwhile, $p(\alpha)$ equals 0.30476, namely costing 30% power compared to methods without energy efficiency optimization.

With more nodes in one cluster, energy efficiency goes well when α is low, which makes solution of α shifts left. And $p(\alpha)$ did not increase. Moreover, there is inspiration that increasing β and n will help to reduce loss and energy cost.

5 Conclusion and Future Work

We find a method to simplify the problem and propose a model to do capacity allocation considering of weighing loss and energy efficiency in hierarchical edge computing. Simulation proves that it is easy to regulate parameter in accordance with the circumstances. Finally, we suppose a pattern to optimize hierarchical edge network structure.

In future, we will still work on hierarchical edge computing and try to figure out a method to adjust hierarchical structure dynamically.

Acknowledgements. This work was supported by the National Key Research and Development Program of China (grant number 2018YFC0831304).

The National Natural Science Foundation of China under Grant 61701019.

References

1. Kiani, A., Ansari, N., Khreishah, A.: Hierarchical capacity provisioning for fog computing. IEEE/ACM Trans. Netw. **27**(3), 962–971 (2019)
2. Li, Y., Wang, S.: An energy-aware edge server placement algorithm in mobile edge computing. In: 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, pp. 66–73 (2018)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN 1995), pp. 1942–1948 (1995).
4. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization for integer programming. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), pp. 1582–1587 (2002)
5. Liu, Z., Peng, T., Peng, B., Wang, W.: Sum-capacity of D2D and cellular hybrid networks over cooperation and non-cooperation. In: Proceedings of 7th International ICST Conference on Communications and Networking, China, pp. 707–711 (2012)
6. Yuan, P., Cai, Y., Huang, X., Tang, S., Zhao, X.: Collaboration improves the capacity of mobile edge computing. IEEE Internet Things J. **6**(6), 10610–10619 (2019)
7. Lin, Y., Lai, Y., Huang, J., Chien, H.: Three-tier capacity and traffic allocation for core, edges, and devices for mobile edge computing. IEEE Trans. Netw. Serv. Manag. **15**(3), 923–933 (2018)
8. Noreikis, M., Xiao, Y., Ylä-Jaäiski, A.: QoS-oriented capacity planning for edge computing. In: 2017 IEEE International Conference on Communications (ICC), Paris, pp. 1–6 (2017)
9. H. Badri, T. Bahreini, D. Grosu and K. Yang: Risk-Based Optimization of Resource Provisioning in Mobile Edge Computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, pp. 328–330. (2018).
10. Liu, M., Liu, Y.: Price-based distributed offloading for mobile-edge computing with computation capacity constraints. IEEE Wirel. Commun. Lett. **7**(3), 420–423 (2018)
11. Dayarathna, M., Wen, Y.G., Fan, R.: Data center energy consumption modeling: a survey. IEEE Commun. Surv. Tutor. **18**(1), 732–794 (2016)
12. Wang, S., Liu, Z., Zheng, Z., Sun, Q., Yang, F.: Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In: Proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2013), pp. 102–109 (2013)
13. Texas Instruments: CMOS Power Consumption and Cpd Calculation. SCAA.35B (1997)