Balancing on the Edge: Fast Service or No Service

Pei Peng
Nanjing University of Posts and Telecommunications
Nanjing, China
Email: pei.peng@njupt.edu.cn

Emina Soljanin
Rutgers, The State University of New Jersey
Piscataway, USA
Email: emina.soljanin@rutgers.edu

2023 IEEE 23rd International Conference on Communication Technology (ICCT) | 979-8-3503-2595-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCT59356.2023.10419405

***Abstract*—Edge computing operates between the cloud and end-users, and strives to provide computing services at a high rate. Since the computing and storage resources are limeted, directing more resources to some computing jobs will block (and pass to the cloud) the execution of others. We evaluate the system performance with two metrics: job computing time and job blocking probability. Edge nodes often operate in highly unpredictable environments, and replicating job execution when resources allow improves the job mean execution time. We show that the job computing time increases with the number of groups, but the job blocking probability does not. That is, there is a tradeoff between job computing time and blocking probability. This paper adopts the average system time as a single system's performance indicator to evaluate the tradeoff. We conclude that the optimal number of groups that maximizes the system service rate changes with arrival rate and cloud time.**

***Index Terms*—Edge computing, blocking system, distributed computing, resource allocation.**

I. Introduction

With the rapid increase in IoT applications, such as smart cities and homes, autonomous vehicles, and artificial intelligence, billions of IoT devices are coming to our everyday lives [1], [2]. The demand for low latency storage and computing services is increasing to accommodate novel IoT platforms (e.g., deep learning) [3]–[5]. Some applications, for example, connected and autonomous vehicles, smart healthcare, and ocean monitoring, require fast service or no service at all. Cloud services are inefficient in responding to such applications.

Edge computing is an inter-layer between the cloud and the end-user. It provides storage and computing infrastructure at the node located one or two network hops from the end-user [6]. Therefore, in the edge system, the bottleneck of the computing service is no longer the communication delay. As shown in Fig. 1, the edge computing system receives and processes jobs from the end-users. However, the edge computing system operates on a much smaller storage and computing resources scale than the cloud. Service requests get sent to the cloud when all edge workers are busy, and communication time becomes a significant part of the delay.

This paper addresses two problems. The first problem is how to increase the computing speed of each job. As a computing system, computing speed, in terms of computing time or service rate, is a classical and essential performance metric [7], [8]. The second problem is processing more jobs in the edge system instead of sending them to the cloud. Due to the limited storage and computing resources, the edge system may be unable to process all jobs by itself. The jobs sent to the cloud will experience higher latency. Therefore, it is necessary to serve more jobs in the edge system to take advantage of the geographic benefit [9].

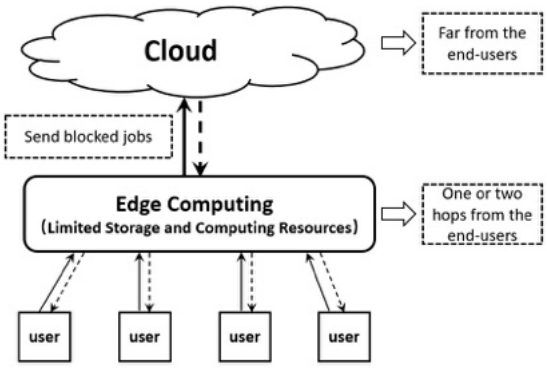

Fig. 1. Edge computing system deployed between the users and the cloud processes the jobs sent by the users.

To address the first problem (decreasing the expected computing time), we need to eliminate the adverse effects of job execution straggling. We model the edge computing system as a distributed computing system where a controller node manages a computing cluster of workers [10]. In such systems (and distributed computing in general), random fluctuations in task computing times lead to some tasks, known as stragglers, taking much longer to execute [11]. Therefore, mitigating stragglers increases the computing speed. Many studies have shown that introducing redundancy, in the form of replication or coding, is an effective method of mitigating stragglers, see, e.g., [12], [13] and references therein.

Using resources redundantly is a known strategy to decrease the mean computing time, which is our first goal. However, with the higher resource usage, the system will send more jobs to the cloud for execution, which goes against our second goal to execute more jobs locally. An edge computing system that sends jobs to the cloud once the resources become unavailable acts locally as a blocking system. We need to use resources strategically in the edge computing system to address both problems. Task scheduling is one of the methods that has been well studied [14], [15]. Strategies for performance metrics are proposed, such as offload latency, power consumption, and energy efficiency. Unlike these studies, we consider the task

scheduling method in the following way. When a job arrives, the edge system replicates it to several workers. Here, we refer to the workers processing the same job as the replication group and its size as the replication factor. Since the total number of workers in the system is limited, we have to decrease the number of groups when we increase the replication factor. The edge computing system with a higher replication factor may have a higher blocking probability. That is, it can process fewer jobs simultaneously. Therefore, processing more jobs in the edge may decrease the computing speed for each job.

Based on the above discussion, we aim to analyze how the job computing time and the job blocking probability changes with the number of groups, and find the optimal number of groups that minimizes the average system time. The paper is organized as follows. In Sec. II, we describe the edge computing system architecture, job arrival, and computing time. In Sec. III, we theoretically and numerically analyze the job computing time and the job blocking probability changes with the number of groups. In Sec. IV, we analyze the optimal number of groups (or replication factor) that minimizes the average system time changes with the parameters of the job arrival rate. The conclusions are given in Sec. V.

II. System Model

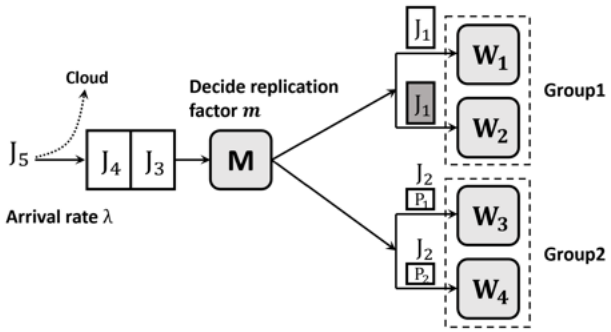

Fig. 2. Edge Computing System Model

System Architecture: We consider the edge computing system model shown in Fig. 2 as a combination of the distributed computing system and the blocking system. The edge computing system has limited storage and computing resources [2]. It consists of a single front-end controller node and multiple computing servers, which we refer to as workers. The single controller node manages the entire computing cluster of nodes and divides the workers into several groups. When a job arrives, the controller node will assign the job and its copies to a group of workers. The workers process the jobs and send the completed jobs to the controller node. There is a queue at the controller node. Due to the limited storage and computing resources, the queue's space is also limited. When all workers are busy, the new arrival job will be blocked and sent to the cloud (see Fig. 2 job J_5). Of course, the blocked job has to experience a significant communication time between the edge and the cloud. Furthermore, this communication time may be much longer than the job computing time. Therefore, our goals for the edge computing system are to increase the computing speed and decrease the number of blocked jobs.

Redundancy and Task Scheduling: In an edge computing system, the job computing time T_{job} is a crucial performance metric. However, task straggling is a fundamental problem in edge computing, which significantly affects the system performance. To solve this problem, replication is an effective technique to introduce redundancy to mitigate stragglers. In Fig. 2, the controller node sends the J_1 and its copy to workers 1 and 2 separately. Compared to J_2 without redundancy, even though one worker processes the job for a long time, the controller node can still receive the result from the other worker. Thus, the system can mitigate one straggler.

As a blocking system, the blocking probability P_b is an effective performance metric to evaluate the probability that a job will be processed locally. To improve the performance, we need to solve the task scheduling problem. In this paper, we mainly focus on finding the optimal way for the controller node to allocate computing resources. In an edge computing system with N workers, when a job arrives, the controller node can assign m workers to this job. When the controller node assigns one worker, the edge system can simultaneously process N jobs; When the controller node assigns N workers, the system can process only one job. Fig. 2 shows an example that the controller node assigns J_1 and J_2 two workers separately. Due to limited resources, the controller node must block some new arrival jobs and send them to the cloud. Thus, the communication delay must be taken into account.

Job Arrival and Computing Time: M/M/1 or M/M/n queues have been used in task allocation models in edge systems [16], [17]. Here, we assume the job arrivals follow a Poisson process with a rate λ, which allows us to model the blocking system as an M/G/c/k queue. Analyzing the blocking probability of an M/M/c/k (or M/G/c/k) queue that models the blocking operation of the system in Fig. 2 is a highly complex problem. To better understand it, we consider the Erlang B model, the M/M/c/c queue, where the queue length equals the number of groups and the computing time follows $\text{Exp}(\mu)$.

Meanwhile, since the worker may process a task that is part of a job, the computing time should change with the task size. Therefore, we will adopt a server-dependent time scaling model in [10]. The assumption here is that the straggling effect depends on the server. Assume that the worker processes the entire job following $\text{Exp}(\mu)$. Thus, if the size of a task is $1/i$ of the job, the worker will process this task following $\text{Exp}(i\mu)$.

Problem formulation: We consider the edge computing system with N workers. The controller node will assign a group with m workers to each job, and the system has a total of c groups. Here, we have $c = N/m$ (c, N and m are integers). Since we use replication to introduce redundancy, the replication factor is equal to m.

In this paper, we mainly evaluate two performance metrics, the expected job computing time $\mathbb{E}[T_{\text{job}}]$ and the job blocking probability P_b. Then we will analyze the optimal number of groups c (or replication factor m) that minimizes $\mathbb{E}[T_{\text{job}}]$ and P_b. Then, we use the average system time to evaluate the

tradeoff between these two metrics. The average system time is defined as

$$\mathbb{E}[T_{\text{sys}}] = (1 - P_b)\,\mathbb{E}[T_{\text{job}}] + P_b\,\mathbb{E}[T_{\text{cl}}] \tag{1}$$

Where T_{cl} is time that a job spends in the cloud.

Parameters and Notations:

N	–	number of workers in the system
m	–	replication factor, number of workers in a group
c	–	number of groups, $c = N/m$
T_{job}	–	job computing time
T_{cl}	–	time that a job spends in the cloud (cloud time)
P_b	–	job blocking probability
T_{sys}	–	system time
λ	–	job arrival rate
μ	–	rate parameter of Exp distribution

III. Performance Metrics Analysis

The job computing time measures how much time the job spends in the system occupying resources. Since the system uses m-fold replication and the worker computing time follows $\text{Exp}(\mu)$, the expected job computing time is given by

$$\mathbb{E}[T_{\text{job}}] = \frac{1}{m\mu}. \tag{2}$$

Observe that $\mathbb{E}[T_{\text{job}}]$ reaches its minimum at $m = N$. Moreover, T_{job} follows the exponential distribution with the rate parameter $m\mu$. See, e.g., [18].

A. Job Blocking Probability

For an Erlang B blocking system with c groups and the job arrives as a Poisson process with the rate λ, the job blocking probability is

$$P_b(c, \rho) = \frac{(\rho)^c/c!}{\sum_{j=0}^{c}(\rho)^j/j!} \tag{3}$$

where $\rho = \lambda\,\mathbb{E}[T_{\text{job}}]$. The above expression shows that for a given c, P_b increases with ρ. From (2), we know that $\mathbb{E}[T_{\text{job}}]$ is a function of m, in which $m = \frac{N}{c}$. Then we will take $\rho = \frac{\lambda}{m\mu}$. Thus, P_b is a function of c and we can rewrite (3) in the following,

$$P_b(c) = \frac{(Kc)^c/c!}{\sum_{j=0}^{c}(Kc)^j/j!} \tag{4}$$

where $K = \frac{\lambda}{N\mu}$ is a constant. We find the m that minimizes the job blocking probability P_b in Theorem 1.

Theorem 1. *For the blocking system with Poisson(λ) arrivals and* $\text{Exp}(\mu)$ *computing time, the job blocking probability P_b increases with the number of groups c and reaches the minimum at $c = N$ (i.e., $m = 1$).*

Proof. Assume $c_1 > c_0$, from (4), the blocking probability is

$$P_b(c_1) = \frac{(Kc_1)^{c_1}/c_1!}{\sum_{j=0}^{c_1}(Kc_1)^j/j!} = \frac{1}{\sum_{j=0}^{c_1}\frac{c_1!/j!}{(Kc_1)^{c_1-j}}}$$
$$= \frac{1}{\sum_{j=0}^{c_1}\prod_{i=1}^{j}\frac{c_1-j+i}{Kc_1}} < \frac{1}{\sum_{j=0}^{c_1}\prod_{i=1}^{j}\frac{c_0-j+i}{Kc_0}}$$
$$< \frac{1}{\sum_{j=0}^{c_0}\frac{c_0!/(c_0-j)!}{(Kc_0)^j}} = \frac{(Kc_0)^{c_0}/c_0!}{\sum_{j=0}^{c_0}(Kc_0)^j/j!} = P_b(c_0)$$

Since $c \in [1, N]$, P_b reaches its minimum at $c = N$.

B. Numerical Analysis

We evaluate (2) and (4) for $\mathbb{E}[T_{\text{job}}]$ vs. c and P_b vs. c. We consider a system with $N = 24$ workers, the job arrives following the Poisson distribution with the rate $\lambda = 1$, and the computing time for each worker follows the exponential with $\mu = 0.1$. The results are plotted in Fig. 3. We observe that

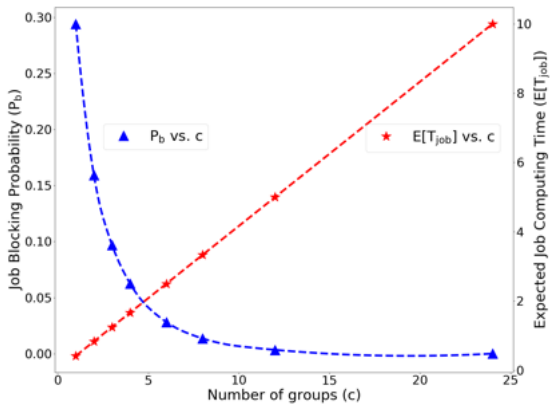

Fig. 3. Expected job computing time $\mathbb{E}[T_{\text{job}}]$ and job blocking probability P_b as a function of c.

$\mathbb{E}[T_{\text{job}}]$ increases linearly with c, which means that introducing more redundancy provides a higher computing speed; P_b decreases with increasing c, which means introducing more redundancy leads to more jobs sent to the cloud. We also observe that P_b is close to 0 and decreases slowly when c is large; when $c \leq 6$, P_b decreases sharply. These observations are consistent with the theoretical analysis of the optimal number of groups. Considering both job computing time and job blocking probability, we find that each metric requires a very different optimal c.

IV. Average System Time

We conclude from the above analysis that simultaneously minimizing the job computing time and blocking probability is impossible. Interestingly, some minimal replication significantly reduces computing time with almost no blocking probability change. Therefore, we adopt the average system time, as a single performance indicator describing the tradeoff between the job computing time and the job blocking probability. For a system with N workers and c groups, the expression of the average system time is

$$\mathbb{E}[T_{\text{sys}}(c)] = (1 - P_b(c))\,\mathbb{E}[T_{\text{job}}] + P_b(c)\,\mathbb{E}[T_{\text{cl}}] \tag{5}$$

Claim 1. *If $\mathbb{E}[T_{job}] \geq \mathbb{E}[T_{cl}]$, the job should always be sent to the cloud.*

In this paper, we assume that the edge system can provide a better computing performance by reducing the communication time between the users and the system. When $\mathbb{E}[T_{\text{job}}] \geq \mathbb{E}[T_{\text{cl}}]$, this means that the edge system does not have enough computing resources for each job. Then it is better for the users to send the job to the cloud directly. From (2), the edge system

will spend $\frac{1}{\mu}$ time to process the job with minimum computing resource, and $\frac{1}{N\mu}$ time to process the job with a maximum computing resource. Therefore, from the Claim 1, we can conclude that the job should always be sent to the cloud when $\mathbb{E}[T_{\text{cl}}] \leq \frac{1}{N\mu}$.

When $\mathbb{E}[T_{\text{cl}}] > \frac{1}{N\mu}$, it may be faster to process the job in the edge system. Then we should make use of the edge system to reduce the average system time. However, from the conclusion of Fig. 3, it is impossible to reduce the job computing time and the blocking probability simultaneously. The main purpose of this section is to find the optimal c (or m) that minimizes the $\mathbb{E}[T_{\text{sys}}]$.

Firstly, we consider the scenario that $\mathbb{E}[T_{\text{cloud}}] \in [\frac{1}{N\mu}, \frac{1}{\mu}]$. According to Claim 1, when $m \leq \frac{1}{\mu\,\mathbb{E}[T_{\text{cl}}]}$, the job should be sent to the cloud; otherwise, the job should be sent to the edge. That is, the optimal m should be in the region $[\frac{1}{\mu\,\mathbb{E}[T_{\text{cl}}]}, N]$. In the following theorem, we find the conditions that the average system time $\mathbb{E}[T_{\text{cl}}]$ reaches the minimum at $c = 1$ ($m = N$).

Theorem 2. *For the edge system with Poisson(λ) arrivals and* $\text{Exp}(\mu)$ *computing time, when* $\mathbb{E}[T_{cl}] \in [\frac{1}{N\mu}, \frac{1}{\mu}]$*, the average system time* $\mathbb{E}[T_{sys}]$ *reaches the minimum at* $c = 1$ *($m = N$) under the conditions* $\lambda \geq N(N-2)\mu$ *or* $\lambda \leq \frac{N\mu}{N-2}$.

Proof. When $c = 1$ and $m = N$, the average system time is

$$\mathbb{E}[T_{\text{sys}}(1)] = (1 - P_b(1))\frac{1}{N\mu} + P_b(1)\,\mathbb{E}[T_{\text{c}}]$$

then, for any $c \geq 2$ we have

$$\begin{aligned}
&\mathbb{E}[T_{\text{sys}}(1)] \leq \mathbb{E}[T_{\text{sys}}(c)] \\
&\Leftrightarrow \frac{1 - P_b(1)}{N\mu} + P_b(1)\,\mathbb{E}[T_{\text{cl}}] \leq \frac{1 - P_b(c)}{m\mu} + P_b(c)\,\mathbb{E}[T_{\text{cl}}] \\
&\Leftrightarrow \mathbb{E}[T_{\text{cl}}](P_b(1) - P_b(c)) \leq \frac{1}{\mu}(\frac{1}{m} - \frac{1}{N} + \frac{P_b(1)}{N} - \frac{P_b(c)}{m}) \\
&\Leftrightarrow \mathbb{E}[T_{\text{cl}}] \leq \frac{1}{\mu}\frac{\frac{1}{m} - \frac{1}{N} + \frac{P_b(1)}{N} - \frac{P_b(c)}{m}}{P_b(1) - P_b(c)}
\end{aligned}$$

Let the above inequality holds when the right hand side is larger than $\frac{1}{\mu}$, we can have

$$P_b(c) \geq \frac{(1 - \frac{1}{N})P_b(1) - \frac{1}{m} + \frac{1}{N}}{1 - \frac{1}{m}} \tag{6}$$

Since $P_b(c) \geq 0$, the inequality (6) holds when $(1-\frac{1}{N})P_b(1) - \frac{1}{m} + \frac{1}{N} \leq 0$. As we know $P_b(1) = \frac{K}{1+K}$, then we have

$$(1 - \frac{1}{N})\frac{K}{1+K} - \frac{1}{m} + \frac{1}{N} \leq 0 \Leftrightarrow K \leq \frac{1 - \frac{1}{N}}{m - 1} - \frac{1}{N}$$

Apparently, the left-hand side increases with decreasing m. Therefore, $K \leq \frac{1}{N-2}$, that is, $\lambda \leq \frac{N\mu}{N-2}$.

According to (3),

$$P_b(c) \geq \frac{K^c}{\sum_{j=0}^{c}(K)^j} = 1 - \frac{1 - K^c}{1 - K^{c+1}} \geq 1 - \frac{1}{K}$$

The inequality (6) holds when $1 - \frac{1}{K} \geq \frac{(1-\frac{1}{N})P_b(1) - \frac{1}{m} + \frac{1}{N}}{1 - \frac{1}{m}}$. The right-hand side reaches the maximum at $m = \frac{N}{2}$. Then we have $K \geq N - 2$, that is, $\lambda \geq N(N-2)\mu$.

Therefore, $\mathbb{E}[T_{\text{sys}}]$ reaches the minimum at $c = 1$ ($m = N$) under the conditions that $\lambda \geq N(N-2)\mu$ or $\lambda \leq \frac{N\mu}{N-2}$.

Next, we consider the scenario that $\mathbb{E}[T_{\text{cl}}] > \frac{1}{\mu}$. In this scenario, the edge system is always the first choice for each job. The job will only be sent to the cloud when the edge system is busy. Similar to the conclusion of Theorem 2, the optimal c that minimizes the average system time changes with different system parameters. However, we can still draw some certain conclusions in the following theorem.

Theorem 3. *For the edge system with Poisson(λ) arrivals and* $\text{Exp}(\mu)$ *job computing time, when* $\mathbb{E}[T_{cl}]$ *is sufficiently large, the average system time* $\mathbb{E}[T_{sys}]$ *reaches the minimum at* $c = N$ *($m = 1$).*

Proof. When $c = N$ and $m = 1$, the average system time is

$$\mathbb{E}[T_{\text{sys}}(N)] = \frac{1 - P_b(N)}{\mu} + P_b(N)\,\mathbb{E}[T_{cl}]$$

$$\begin{aligned}
&\mathbb{E}[T_{\text{sys}}(N)] \geq \mathbb{E}[T_{\text{sys}}(c)] \\
&\Leftrightarrow \frac{1 - P_b(N)}{\mu} + P_b(N)\,\mathbb{E}[T_{\text{cl}}] \leq \frac{1 - P_b(c)}{m\mu} + P_b(c)\,\mathbb{E}[T_{\text{cl}}] \\
&\Leftrightarrow \mathbb{E}[T_{\text{cl}}](P_b(c) - P_b(N)) \geq \frac{1}{\mu}(1 - \frac{1}{m} - P_b(N) + \frac{P_b(c)}{m}) \\
&\Leftrightarrow \mathbb{E}[T_{\text{cl}}] \geq \frac{1}{\mu}\frac{1 - \frac{1}{m} - P_b(N) + \frac{P_b(c)}{m}}{P_b(c) - P_b(N)}
\end{aligned}$$

Apparently, the right-hand side of the above inequality is finite. Therefore, $\mathbb{E}[T_{\text{sys}}]$ reaches the minimum at $c = N$ ($m = 1$) when $\mathbb{E}[T_{\text{cl}}]$ is sufficiently large.

A. Numerical Analysis

In Fig. 4, we evaluate (2) to see how the average system time changes with c. We consider a system with $N = 24$ workers, and the computing time for each worker follows the exponential with $\mu = 0.1$. In the upper subfigure, the job arrives following the Poisson distribution with different values of $\lambda \in \{0.1, 1, 10, 40\}$, and the cloud time $\mathbb{E}[T_{\text{cl}}] = 8$ is smaller than the maximum job computing time. In the lower subfigure, we consider different values of $\mathbb{E}[T_{\text{cl}}] \in \{15, 50, 100\}$. The job arrives following the Poisson distribution with $\lambda = 2$.

The upper subfigure shows that when $\lambda = 0.1$ is small enough, the average system time $\mathbb{E}[T_{\text{sys}}]$ reaches its minimum at $c = 1$, which is consistent with the result in Theorem 2. To decrease c and increase the replication factor m will significantly reduce $\mathbb{E}[T_{\text{sys}}]$. When $\lambda = 1$ or 10, increasing the number of groups leads to a smaller $\mathbb{E}[T_{\text{sys}}]$. When $\lambda = 40$ is sufficiently large, $\mathbb{E}[T_{\text{sys}}]$ reaches its minimum at $c = 1$ again. However, the optimal value of c does not provide a significant reduction of $\mathbb{E}[T_{\text{sys}}]$. From the results, we conclude that when fewer jobs arrive, almost all jobs can be processed in the edge system, so it is better to increase the computing speed. When more jobs arrive, the system should reduce the computing speed and provide the computing resources for more jobs. When there are too many jobs, the resource allocation strategy can not change the system performance significantly.

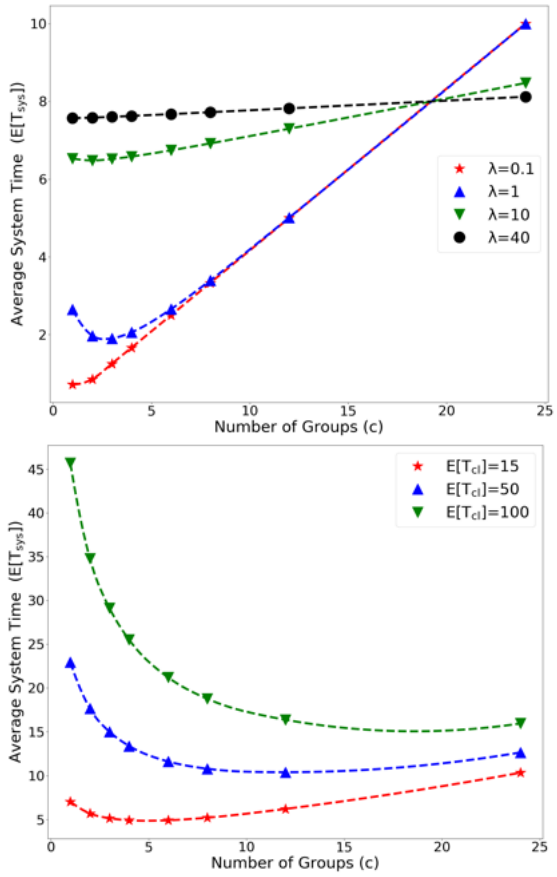

Fig. 4. Average system time $\mathbb{E}[T_{\text{sys}}]$ vs. the number of groups c for different values of λ (upper) or $\mathbb{E}[T_{\text{cl}}]$ (lower).

The lower subfigure shows that when $\mathbb{E}[T_{\text{cl}}] = 15$, $\mathbb{E}[T_{\text{sys}}]$ reaches its minimum at $c = 4$; when $\mathbb{E}[T_{\text{cl}}] = 50$, $\mathbb{E}[T_{\text{sys}}]$ reaches its minimum at $c = 12$; when $\mathbb{E}[T_{\text{cl}}] = 100$, $\mathbb{E}[T_{\text{sys}}]$ reaches its minimum at $c = 24$. We observe that the optimal number of groups c increases with $\mathbb{E}[T_{\text{cl}}]$. From the results, we have the following conclusions. Generally, we should balance the tradeoff between the job computing time and the blocking probability. However, when $\mathbb{E}[T_{\text{cl}}]$ is sufficiently large, we should focus on reducing the blocking probability and assigning resources for more jobs.

V. Conclusions

We addressed the questions concerning the number of groups that optimizes the job computing time and the job blocking probability. We find that it is impossible to simultaneously minimize these two performance metrics. Therefore, we use the average system time to evaluate the tradeoff. We find that the optimal number of groups that minimizes the average system time changes with the parameters of the job arrival rate and the cloud time. Further questions of interest include, e.g., the analysis for other computing time probability distributions (Pareto and bimodal) and replication strategies. The optimal number of groups may behave differently for light and heavy-tailed distributions since their computing cost vs. time tradeoffs are qualitatively different [19].

Acknowledgment

This work was supported in part by the University Science Research Project of Jiangsu Province (Grant No. 23KJB120009), the Natural Science Research Start-up Foundation of Recruiting Talents of Nanjing University of Posts and Telecommunications (Grant No. NY222008) and the NSF-BSF Award FET-2120262.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things J.*, vol. 3, pp. 637–646, 2016.
[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, pp. 2322–2358, 2017.
[3] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.
[4] J. Hochstetler, R. Padidela, Q. Chen, Q. Yang, and S. Fu, "Embedded deep learning for vehicular edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 341–343.
[5] B. Li, P. Chen, H. Liu, W. Guo, X. Cao, J. Du, C. Zhao, and J. Zhang, "Random sketch learning for deep neural networks in edge computing," *Nature Computational Science*, vol. 1, no. 3, pp. 221–228, 2021.
[6] 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 (HotWeb)*. IEEE, 2015, pp. 73–78.
[7] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel comp." *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 3, pp. 7–11, 2015.
[8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. on Inform. Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.
[9] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[10] P. Peng, E. Soljanin, and P. Whiting, "Diversity/parallelism trade-off in distributed systems with redundancy," *IEEE Trans. on Information Theory*, vol. 68, no. 2, pp. 1279–1295, 2021.
[11] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[12] J. Wang, C. Cao, J. Wang, K. Lu, A. Jukan, and W. Zhao, "Optimal task allocation and coding design for secure edge computing with heterogeneous edge devices," *IEEE Trans. on Cloud Computing*, 2021.
[13] Y. Han, D. Niyato, C. Leung, C. Miao, and D. I. Kim, "Dynamics in coded edge computing for iot: A fractional evolutionary game approach," *IEEE Internet of Things Journal*, 2022.
[14] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2020.
[15] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4854–4866, 2018.
[16] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *Sensors*, vol. 21, no. 3, p. 779, 2021.
[17] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM transactions on networking*, vol. 26, no. 4, pp. 1619–1632, 2018.
[18] P. Peng, E. Soljanin, and P. Whiting, "Diversity vs. parallelism in distributed computing with redundancy," in *IEEE International Symposium on Information Theory, ISIT 2020, Los Angeles, CA, USA, June 21-26*, 2020, pp. 257–262.
[19] M. F. Aktas and E. Soljanin, "Straggler mitigation at scale," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2266–2279, 2019.