Journal of the Korea Industrial Information Systems Research Vol.26 No.3, Jun. 2021 :1-7
http://dx.doi.org/10.9723/jksiis.2021.26.3.001 ISSN:1229-3741

Fog Computing 환경에서의 최적화된 컨테이너 배포 정책+

(An Optimal Container Deployment Policy in Fog Computing Environments)

진 성 근[1)*], 전 인 걸[2)]
(Sunggeun Jin and In-Geol Chun)

요 약 Fog Computing (FC) 호스트의 새로운 요청 도착에 대처하기 위해 적절한 컨테이너가 배포된다. 이 경우 두 가지 시나리오를 고려할 수 있다. (1) 컨테이너 배포를 위한 충분한 자원이 준비될 때까지 요청이 대기열에 추가될 수 있다. (2) FC 호스트는 자원이 제한되거나 부족하여 새 컨테이너 배포를 수용할 수 없는 경우 도착한 서비스 요청을 근처 FC 호스트로 전송할 수 있다. 여기서, 더 많은 인접 FC 호스트를 사용할수록 각 FC 호스트의 컨테이너 배포 시간이 더 짧아 진다. 반면, FC 호스트 수가 증가할수록 더 많은 FC 호스트를 거쳐가야 하므로 서비스 요청이 전달되는데 더 긴 시간이 걸릴 수 있다. 이러한 이유로 활용되는 FC 호스트의 수에 따라 컨테이너 배포 시간에 따른 트레이드오프 관계가 성립한다. 결과적으로, 우리는 최적의 인접 FC 호스트 수를 사용하기 위해 트레이드오프 관계를 분석한다.

핵심주제어: 포그 컴퓨팅, 컨테이너 배포

Abstract Appropriate containers are deployed to cope with new request arrivals at Fog Computing (FC) hosts. In the case, we can consider two scenarios: (1) the requests may be queued until sufficient resources are prepared for the container deployments; (2) FC hosts may transfer arrived service requests to nearby FC hosts when they cannot accommodate new container deployments due to their limited or insufficient resources. Herein, for more employed neighboring FC hosts, arrived service requests may experience shorter waiting time in container deployment queue of each FC host. In contrast, they may take longer transfer time to pass through increased number of FC hosts. For this reason, there exists a trade-off relationship in the container deployment time depending on the number of employed FC hosts accommodating service request arrivals. Consequently, we numerically analyze the trade-off relationship to employ optimal number of neighboring FC hosts.

Keywords: Fog computing, container deployment.

* Corresponding Author: sgjin@daegu.ac.kr
+ 이 성과는 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2016R1D1A1B04932067)
Manuscript received May 19, 2021 / revised June 17, 2021 / accepted June 20, 2021
1) 대구대학교 컴퓨터소프트웨어학과, 제1저자, 교신저자
2) 한국전자통신연구원 고신뢰CPS연구그룹 책임연구원, 제2저자

1. INTRODUCTION

In the past few years, we have witnessed fast evolution of Fifth-Generation (5G) network technologies providing extremely low latency for various services having stringent time requirements such as connected vehicles, smart factories, augmented reality services and streaming games. Accordingly, it is expected to control drones and vehicles accurately in remote places and enjoy streaming games giving rich user experiences in near future [3], [5].

Meanwhile, cloud computing services have proliferated on a worldwide scale, thus making it possible to enjoy complicated applications, high performance servers, and virtual desktop services online without advanced personal computers, cumbersome software installations, and annoying system managements. In spite of the benefits, cloud computing servers are generally located far from the places where users are trying to access. Accordingly, it may take long time to reach long-distance servers so that long round-trip times may incur service disruptions. In addition, tremendous number of user devices generate overwhelming data so that cloud computing servers suffer from manipulating those data. For this reason, Fog Computing (FC) as an edge computing technology was introduced to provide prompt cloud computing services and/or reduce redundant data processing overhead on cloud computing servers by bringing a part of functions provided by cloud computing servers to the places near to the 5G users.

The FC hosts may expedite Internet-of-Things (IoT) services by dramatically reducing operational overhead with FC applications running on the edge of 5G networks. For the purpose, it is necessary to launch FC applications suitable for the services on selected FC hosts. It is realized by deploying proper containers providing light virtual operational environments for the FC applications runs [4].

Basically, a proper FC application begins at a container on the FC host where a user triggers a new service. Occasionally, the FC host may not accommodate the FC application since typical FC hosts may have limited resources due to their innate feature for embedded systems [2]. Then, it may transfer the requests to the other neighbouring FC hosts in order to deploy containers and launch proper FC applications. This transfer operation is allowed through edge computing server interfaces defined in European Telecommunications Standards Institute (ETSI) standard [1]. However, we may experience additional latency for the container deployment since the requests may travel through several FC hosts until arriving at affordable FC hosts. In other words, farther hosts are employed, longer latency it takes for the container deployments.

On the contrary, thanks to the request transfers at the requested FC host, the requested FC host has reduced number of requests. Therefore, the waiting times are shortened in the container deployment queue. Consequently, it is true that forwarding requests may be accumulated in the container deployment queues in the steady state as detailed later. Nevertheless, the request transfers may incur decrease of the waiting times in container deployment queues and increase of forwarding latency over multiple FC hosts. For this reason, we can recognize that there exists trade-off relationship in the view of overall container deployment times depending on how to manage the request transfers over FC hosts. In the paper, we show the trade-off relationship by conducting numerical analysis. Then, we explain an optimal con-

tainer deployment strategy.

The paper is organized as follows: In Section II, we explain our FC system model and its operations for our numerical analysis. In Section III, we numerically analyse the FC system model with our assumptions. In Section IV, we discuss an optimal container deployment strategy with evaluations of numerical equations. Section V concludes this paper.

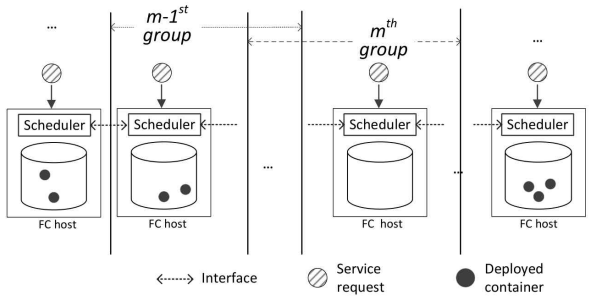

Fig. 1 Container deployments over FC hosts

2. SYSTEM MODEL

Each FC host has a scheduler in charge of container placements. When a new service is requested to an FC host, it usually deploys a new container in order to launch the service. However, it is possible that the resources in an FC host are occasionally exhausted for various reasons including overloaded traffic and/or computations.

In this case, the scheduler in an FC needs to determine whether to deploy a new container in spite of the depleted resources or transfer the request to a neighbouring FC host depending on its resource utilization status. The transferred requests are treated as a normal request in its neighbouring FC hosts. Accordingly, their schedulers should determine to accommodate or forward the transferred requests depending on its resource availability. In any events, it takes time to manage the requests for container deployment as well as request forwarding.

In our system, K FC hosts are grouped and each FC host is capable of deploying N containers. The hosts in each group are selected by the order of request forwarding latency from an originated FC host. Fig. 1 shows a logical topology for FC hosts. From this figure, we elaborate on how to manage container deployment requests below: (1) Once a request arrives, an FC host deploys a container suitable for the request in case when there a room for the deployment. (2) When FC host fails to deploy a container due to its insufficient resources, it transfers the request to its nearest FC host in terms of forwarding latency. (3) In case of the deployment failure in its nearest FC host, the request is repeatedly forwarded to the next FC hosts in turn until successful container deployment. (4) The request forwarding is stopped when none of available FC hosts are found in a group having K FC hosts. (5) Instead, the request is queued in the originated FC host and stays waiting for successful container deployment.

Fig. 2 shows a Markov process representing the container deployment operation for our system model. In the figure, $Q(N)$ indicates the state that N service requests are queued an FC host while waiting for container deployment. In the steady state, $Q(N)$ of all FC hosts are identical since we consider infinite number of hosts and their groups. For each group, index k is labeled in order of forwarding latency from the FC host where a service request arrives. Obviously, the container deployment requests are serviced when $n \leqq N$ so that containers can be deployed up to N. Otherwise, the requests are transferred

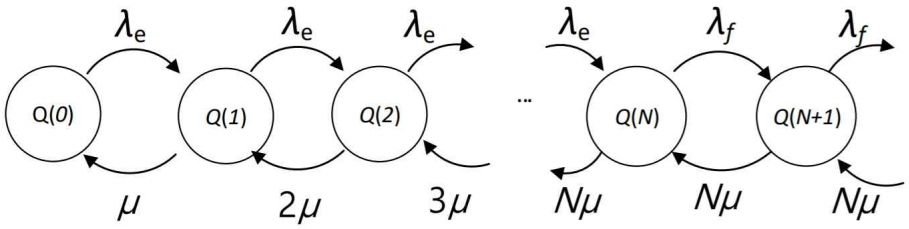

Fig. 2 A Markov process for the container deployment operation of FC hosts in the steady state

to a nearest available FC host while they are queued for none of available neighbouring FC hosts.

3. Numerical Analysis

For the numerical analysis, we have four assumptions summarized by: (1) service request arrivals follow Poisson distribution with mean rate λ; (2) container lifetimes are exponentially distributed with the mean time of $1/\mu$; (3) each FC host can accommodate N containers at maximum; (4) container deployment requests may experience additional forwarding latency given by its expectation $E[t_s]$ between two neighbouring FC hosts.

For our analysis, we should consider the operations in the steady state. As explained previously, an available FC host receives container deployment requests from its fully occupied neighbouring FC hosts belonging to the same group with the recipient FC host. In the steady state, due to the fact that there are infinite number of groups and each FC host shares the same operations as other FC hosts, all FC hosts have homogeneous queue. Therefore, we can analyze it with a single queue while considering the request transfer rate and incoming request rate due to neighbors' request transfer operations.

Now, we consider stationary probability pi_n for the stationary state that there are n requests in a queue of an FC host. Herein, we can derive the probability P_f representing the case when an FC host is fully occupied by:

$$P_f = \sum_{n=N}^{\infty} \pi_n . \quad (1)$$

Assuming neighbouring K FC hosts are reserved for container deployments, we can derive the probability P_t for container deployment request transfer to a nearest available FC host by:

$$P_t = (1-P_f)\sum_{k=1}^{K-1}(P_f)^k = P_f(1-P_f^{K-1}). \quad (2)$$

Note that the probability $(1-P_t)$ indicates a request is queued for none of available FC hosts. Therefore, we have $\lambda_e = \lambda(1+P_t)$ and $\lambda_f = \lambda(1-P_t)$. From the equations, we can derive stationary probability π_n with FC host utilization ratio $\rho_e (= \lambda_e/\mu)$ in case when containers of an FC host are not fully utilized by:

$$\pi_n = (\frac{\lambda_e}{n\mu})\pi_{n-1} = \frac{1}{n!}(\rho_e)^n\pi_0, \quad (3)$$

where $1 \le n \le N$. We continue to have π_n for $N < n$ by:

$$\pi_n = (\frac{\lambda_f}{N\mu})^{n-N}\pi_N = \frac{1}{N!}(\frac{\lambda_e}{\mu})^N(\frac{\lambda_f}{N\mu})^{n-N}\pi_0 = \frac{1}{N!}(\frac{N\rho_e}{\rho_f})^N(\rho_f)^n\pi_0. \quad (4)$$

Note that $\sum_{n=0}^{\infty}\pi_n = 1$. Therefore, we can derive π_0 by:

$$\pi_0 = \frac{1}{\sum_{n=1}^{N}\frac{(\rho_e)^n}{n!}+\frac{1}{N!}(\frac{N\rho_e}{\rho_f})^N\sum_{n=N+1}^{\infty}(\rho_f)^n} = \frac{1}{1+\sum_{n=1}^{N}\frac{(\rho_e)^n}{n!}+\frac{1}{N!}(N\rho_e)^N\frac{\rho_f}{1-\rho_f}}. \quad (5)$$

In fact, the time required for container deployment consists of two factors, i.e., average request waiting time in a queue and average request forwarding latency. From Eqs. (4) and (5), we first derive the average waiting time with Little's Law by:

$$\frac{1}{\lambda_f}\sum_{n=N+1}^{\infty}(n-N)\pi_n. \quad (6)$$

From this equation, we finally derive the average container deployment time by:

$$\frac{1}{\lambda_f}\sum_{n=N+1}^{\infty}(n-N)\pi_n + E[t_S]\sum_{k=1}^{K-1}k(P_f)^k(1-P_{f)}. \quad (7)$$

As discussed already, we consider two scenarios in terms of analytical concern. Actually, two scenarios are extreme cases for $K=1$ and $K \to \infty$, respectively.

First, in case when each FC host is utilized to deploy requested containers without neighbouring FC hosts, arriving requests are accumulated in infinite queue in each FC host. From this scenario, we can easily derive average waiting time with $K=1$ by:

$$\frac{1}{\lambda}\sum_{n=N+1}^{\infty}(n-N)\pi_n. \quad (8)$$

In this equation, we can recognize the deployment time varies depending on the request arrival rate and containers lifetimes.

Second, infinite number of neighbouring FC hosts are employed for container deployments. In this case, fully occupied FC host can immediately send newly arrived requests to an FC host among infinite number of FC hosts. It implies that every waiting queue is empty in FC hosts. Keeping the scenario in mind, we can have the average latency time for request forwarding with $K \to \infty$ by:

$$E[t_S]\sum_{k=1}^{\infty}k(P_f)^k(1-P_f) = E[t_S]\frac{P_f}{1-P_f}. \quad (9)$$

From Eqs. (7), (8), and (9), we realize that we can control container deployment time between two extreme cases by adjusting the range(=K) for request forwarding.

4. Evaluations

Typical FC hosts have limited resources since they are usually used for IoT devices including sensors, wearable devices, health monitoring devices, etc. Meanwhile, the devices may be consistently utilized since monitoring data should be gathered continuously. For this reason, we adopt parameter values by $N=5$, $\lambda=0.1/\mathrm{s}$, and $E[t_S]=0.1\mathrm{s}$. However, we do not specify μ since it is replaced by ρ. Since $0 \le \rho < 1$ and K is natural number, we can apply brute force search for the evaluations.

We first observe how probability K, that

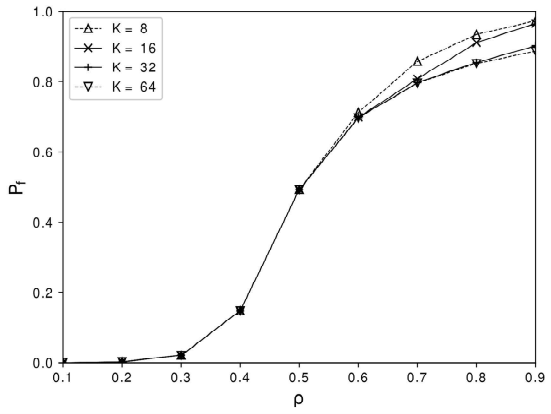

Fig. 3 The probability P_f that FC hosts are fully occupied in the steady state

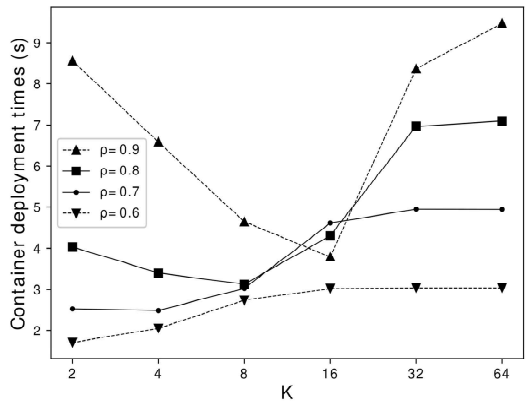

Fig. 4 Container deployment times

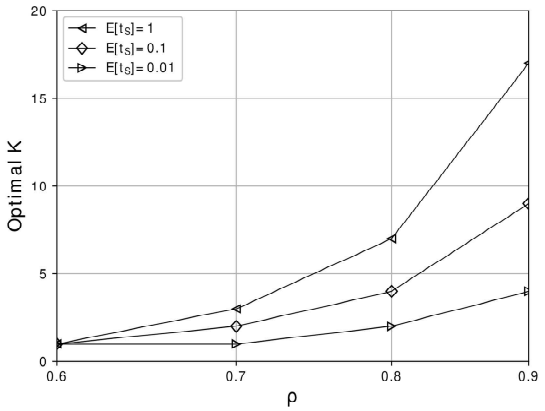

Fig. 5 Optimal values of K

FC hosts are fully occupied in the steady state, depending on ρ shown in Fig. 3. In this figure, we can observe meaningful differences when $0.6 \leq \rho \leq 0.9$. Therefore, we focus on the ranges while ignoring underutilized condition for $\rho < 0.6$ in the subsequent evaluations.

Fig. 4 shows the container deployment times depending on K for each value of ρ. In this figure, we can observe container deployment time is high for $\rho = 0.9$ since FC hosts stay very busy. Similarly, container deployment times increase in proportion to ρ. In addition, we can find there exist minimum points by adjusting the values of K. It implies that there is trade-off relationship between request waiting time and request forwarding time. For example, K. container deployment time decreases while K increases from 2 up to 16. After that, it increases for $K \geqq 16$ from the minimum container deployment time. Particularly, we can see the highest point for $\rho = 0.7$ compared with the deployment times for $\rho = 0.8$ and 0.9 when $K = 16$. However, it is not surprising finding since the lowest point for $\rho = 0.7$ is less than the lowest points for $\rho = 0.8$ and 0.9. It implies that the way how to manage K influences the container deployment times greatly.

Finally, we find optimal values of K minimizing container deployment times depending on ρ in a brutal force manner from Eqs. (7)-(9) as shown in Fig. 5. From this figure, we can realize that more FC hosts should be employed for heavier workload and longer request forwarding latency. Consequently, in order to minimize the container deployment times, we need to dynamically adjust the number of employed FC hosts by monitoring workload of each FC hosts carefully.

5. Conclusion

We find there exist a trade-off relationship between waiting times in container deployment queue and container request forwarding times over multiple FC hosts in an analytical manner. The analytical results show that we need to deal with the number of employed FC hosts carefully for minimum container deployment times while monitoring system utilizations of FC hosts. In future, we further study practical container deployment strategy by implementing monitoring system to derive precise service arrival model and container deployment times while the proposed scheduling policy is provided with the dynamic adjustment policy discussed in the paper.

References

ETSI. Multi-access Edge Computing (MEC); Framework and Reference Architecture. GS MEC 003 version 3.2.0, January 2019.

H. Shah-Mansouri and V. W. S. Wong, "Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 3246 - 3257, August 2018.

M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-Enabled Tactile Internet," IEEE Journal on Selected Areas in Communications, vol. 34, no. 3, pp. 460 - 473, March 2016.

S. Hoque, M. S. d. Brito, A. Willner, O. Keil, and T. Magedanz, "Towards Container Orchestration in Fog Computing Infrastructures," in Proc. COMPSAC'17, 2017.

Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, Mobile Edge Computing A Key Technology towards 5G. ETSI, September 2015.

진 성 근 (Sunggeun Jin)

- 정회원
- 경북대학교 전자공학 공학사
- 경북대학교 전자공학 공학석사
- 서울대학교 무선통신 공학박사
- 한국전자통신연구원 선임연구원
- (현재) 대구대학교 컴퓨터 소프트웨어전공 부교수
- 관심분야: 클라우드 컴퓨팅, 5G/6G 무선 네트워크

전 인 걸 (In-Geol Chun)

- 성균관대학교 정보공학 공학석사
- 성균관대학교 컴퓨터공학 공학박사
- (현재) 한국전자통신연구원 차세대시스템SW연구실 PL (프로젝트리더, 책임연구원)
- (현재) 한국과학기술연합대학원대학교(UST) 컴퓨터소프트웨어학과 부교수
- (현재) TTA PG609 CPS표준화그룹 의장
- (현재) 대한임베디드공학회 상임이사