A Container Deployment Optimization Method for Edge Computing

Jianfeng Zhang¹,*, Bin Liu² and Xiaochuan Wang¹
¹College of Computer, National University of Defense Technology, Hunan, China
²China Science Edge Intelligent Information Technology (Suzhou) Co, Ltd, Suzhou, China

*Corresponding author email: jfzhang@nudt.edu.cn

Abstract. In recent years, with the rapid development of Cloud Computing, Big Data and Internet of Things, the amount of data generated by edge nodes grows profusely, which results in high network bandwidth requirements. In order to improve the QoS of Cloud Computing, edge computing has received wide-spread attention in both industry and academia. As a resource isolation technology for edge nodes, container virtualization has been widely used. However, under the condition of unstable network connections in edge environments, there are some shortcomings on native docker container, such as large amount of image data transmission, network interruption and retransmission cost, which affect the quality of edge services. Therefore, this paper proposes an Images Distribution Method based on Deduplication (IDM-Dedup), which improves the transmission efficiency of container image and reduces the time to deploy and migrate edge services. The experimental results show that IDM-Dedup is more effective than native docker container method.

1. Introduction

In recent years, with the rapid development of Cloud Computing, Big Data and Internet of Things, traditional computing models have been changed. In particular, the growth rate of data is far exceeds the growth rate of network bandwidth. However, new applications such as the Internet of Things have put forward higher requirements for response time. Therefore, edge computing, as a new computing model, has received wide-spread attention in both industry and academia [1, 2].

Edge computing is a distributed computing model in which computing occurs near the location where data is being collected and analyzed. By placing computing services closer to these locations than cloud computing, users benefit from faster, more reliable services. Compared with cloud computing, edge computing can better support end-user’s applications [3-5]:

- Saving bandwidth. The amount of data being generated by edge devices is huge. Transferring all these data to cloud center will consume lots of bandwidth. So we should process these data locally and only transfer the significant data to cloud center, while discarding the rest.
- Reducing latency. In cloud computing, data is sending to cloud center to be processed, which result in a brief delay especially under unstable network condition. In some situation, it is far less acceptable. With edge computing, critical data can be processed on the edge, which has the quicker response time.
- Improving security and privacy. Edge computing avoid user’s private data to travel over a whole network, which is an obvious bonus from a security perspective.
At present, edge computing has been used in many fields, such as smart cities, smart transportation, smart manufacturing and other fields. The existing edge computing platforms are basically based on resource management and scheduling platforms such as Openstack and Kubernetes. Based on the limited resources of edge nodes, docker container offers a light weight and high performance alternative to virtualization. However, under the condition of unstable network connections in edge environments, the existing docker deployment methods need data transmission and retransmission. Therefore, under the condition of unstable network connections in edge environments, we propose an Images Distribution Method based on Deduplication (IDM-Dedup) which improves the transmission efficiency of container images and reduces the cost of edge service deployment and migration [6].

The remainder of this paper is organized as follows. Related works are discussed in section 2. In Section 3, we propose our method to optimizing the transmission efficiency of the container image. In Section 4, an extensive empirical evaluation is reported. Our work of this paper is summarized in Section 5.

2. Related Works
Edge Computing Platform (ECP) enables customers to meet growing business demands by effortlessly deploying and scaling up container-based applications. ECP places high-performance compute, storage and network resources as close as possible to end users. Doing so lowers the cost of data transport, decreases latency, and increases locality. At present, edge computing platforms mainly include ParaDrop [7], Cloudlet [8, 9], PCloud [10], StrlingX [11], Akraino [12] and other projects.

ParaDrop is an open source edge computing platform proposed by WiNGS Lab. The project extends the wireless gateway to an edge computing platform and uses the gateway resources to deploy cloud applications. The overall structure of ParaDrop is shown in the figure 1. Container is used to isolate applications in Operating System, so multiple applications can run on a same gateway.

Cloudlet is an edge computing project proposed by Carnegie Mellon University, which turns the original "mobile device-cloud" into "mobile device-Cloudlet-cloud". The architecture of Cloudlet is shown in figure 2. In cloudlet, container is also used to provide an isolated operating environment for applications.

PCloud is the research result of the Korvo Research Group of Georgia Institute of Technology in the field of edge computing. It provides services for mobile devices by integrating computing, storage and cloud computing resources. StrlingX is an open source project under the OpenStack Foundation that provides services for distributed edge clouds. Akraino Edge Stack is a Linux Foundation project that supports high-availability cloud services for edge computing, which develops and runs carrier-grade computing applications in virtual machines (VMs) and containers. At the same time, AWS also launches its own edge computing platform Greengrass [13] in 2016, and Microsoft officially launched the technology preview version of its edge computing platform-Azure IoT Edge at the Build 2017 conference.

In summary, most Edge Computing platforms and projects use docker to deploy edge services. Compared with other virtual machines such as KVM and Xen [14], the size of docker images are
much smaller than Virtual Machine (VM) images, and can be started and stopped in milliseconds. So docker is suitable to launch applications at the edge than VM [15].

3. Optimization Methods
This section mainly introduces the container deployment optimization methods for edge computing. Firstly the overall architecture is given, secondly the hierarchical model optimization method to reduce image size is given, and thirdly transmission optimization methods in unstable connection are given, finally, the efficient distribution and dynamic synthesis mechanism of the container image in the mobile edge environment is given.

3.1. Optimization Overall Architecture
Compared with data center nodes, the architecture of edge devices is heterogeneous (e.g., AMD, Intel, ARM, PowerPC). Meanwhile edge computing has different software environments, insufficient computing resources, and unstable network connections.

In order to provide best services for end users, this paper proposes an optimization methods for deploying and migrating edge applications based on docker environment. The architecture is shown in figure 3.

![Diagram of Optimization Overall Architecture](image)

**Figure 3.** Optimization overall architecture.

Edge Applications Infrastructure Layer: This layer uses docker based on lightweight virtualization technology, which uses namespaces and cgroups mechanisms to provide isolation and security guarantees. Edge Applications Infrastructure Layer includes:

- **Edge Application Virtualization.** In order to rapidly deploy Edge Applications on edge nodes, the edge applications need to be virtualized as docker images. Especially, accelerators on edge nodes based on GPU and FPGA should be considered.
- **Layer Model Optimization.** Each docker image has a list of read-only layers which represent filesystem differences. Layers are stacked on each other to generate a base for a container’s root filesystem. In edge environments, traditional method has large amount of transmission data. So Layer Model Optimization is used to reduce the size of docker images.
- **Private Docker Registry.** The existing container management platforms mainly use Docker Hub which is the world’s largest library of container images as their Docker Registry. Since the public docker registry exists weak security and low bandwidth, it is necessary to build a private docker registry for the edge application.

Edge Applications Pre-install Layer: This layer optimizes the deployment efficiency of the container image in the edge environments. Through image pre-distribution, CDN, and peer-to-peer mechanism, the response times of service are improved. Edge Applications Pre-install Layer includes:

- **Images Distribution.** Images Distribution is used to download the basic layer of images to edge nodes, which reduces the amount of dynamic data transmission during service deployment.
- **Image Cache.** CDN is used to cache docker images in edge environments.
3.2. Hierarchical Model Optimization

In data center environments, applications are deployed to clouds by converting from a physical computer image (its operating system, data and application) into a virtual machine (VM) through P2V technology. Generally speaking, the virtual machines converted by P2V methods have a big size. So in edge environments, it is difficult to adopt P2V technology, and container-based virtualization is most suitable. The container-based virtualization adopts a hierarchical way to organize container images, which improves the transmission efficiency of images. However, in this model, the bottom layer usually uses a standard Operating System, then the docker image is constructed hierarchically by dockerfile. This method greatly reduces disk usage by sharing the same image layers. However, under the condition of unstable network connections in edge environments, there is still a shortage of large dynamic data transmission. Therefore, this article improves the efficiency of container deployment by optimizing the image layer model for edge application. The details are as follows:

- Choose lightweight base image. Traditional docker images are mainly based on standard Ubuntu and RHEL images as base images, which makes the docker image huge for edge environments. So, most lightweight operation systems such as busybox and alpine are used, which can result in significant resource and cost savings.

- Maximize the cache layer. In order to improve the first boot up time of the application service, the base image layers are pushed to the edge node before running a docker container. So it is necessary to reduce the layers of container as much as possible. The new base image is generated by combining these common layers which exist in all the application image layers.

- Redundancy optimization. During the image generation process, a large number of redundant files will be generated due to the installation of software packages. In order to optimize the image size, no useful software packages in this image should be deleted. So in this paper, inotifyait tool can be used to optimize the container image.

3.3. Transmission Mechanism Optimization

Generally, docker uses a client-server architecture. Docker images are stored in a docker registry, and required images are pulled from specific registry. During this process, the Docker Registry HTTP API V2 is used to facilitate distribution of images to the docker engine. However, under the condition of unstable network connections in edge environments, images need to be retransmitted due to network jitter when using Docker Registry HTTP API V2. The retransmissions not only cause great transmission overhead, but also affect the service response time. So in this paper, BitTorrent communication protocol is used to distribute images in the edge environments. BitTorrent communication protocol enables edge nodes to share images to its neighbors, which accelerates the pulling efficiency of the image, and improves the efficiency to docker deployment and migration.

3.4. Image precaching mechanism

According to the previous descriptions in this paper, when the container starts, edge nodes need to download the required image from Docker Registry or there neighbours. Under the condition of unstable network connections in edge environments, the downloading process may be very time-consuming. So, in edge environments, if an edge node is detected to be alive, base images will be pushed to this node. By precaching mechanism, the images are downloaded ahead of time in anticipation of their use, which speeds up docker boot time.

4. Experiment Evaluation

In this Section we experimentally evaluate the effectiveness and the efficiency of our methods described in this paper. In Section 4.1, we illustrate the experiment environments. In section 4.2, we
verify the efficiency of hierarchical model optimization. In section 4.3, the ability to resume the transfer at a breakpoint is checked. Finally, we test the concurrent transmission capability.

4.1. Experiment Environments
We run all the experiments on four servers (S1, S2, S3, S4) with Linux Operating System, ARM-64 processor, and 128G memory, 2T hard disk and 1000 Mbps Ethernet. And in our experiments, the docker version is 18.03, the kubernetes version is 1.15, and a private Docker Registry is deployed on server S1.

4.2. Hierarchical Model Optimization Verification
The effectiveness of the hierarchical model optimization method is verified by comparing the size of images between native method (DockerNative) and proposed method (IDM-Dedup) in our paper. Firstly, we pull several commonly application images from docker hub, such as Ubuntu, latex, Hadoop, Redis, MySQL and MariDB. Then we use the method proposed in this paper to optimize these images. Finally, by comparing these two methods, the results are shown in figure 4, we can see that the size of images optimized by IDM-Dedup is approximately half of the image in docker hub.

4.3. Verify the Ability to Resume the Transfer at A Breakpoint
We design an unstable network connection to verify the ability to resume the transfer at a breakpoint. Firstly, we push hadoop image to the private Docker Registry on server S1. Secondly, we pull Hadoop image on server S2. After the image is successfully downloaded, we can verify that the image is pulled from Docker Registry on server S1. Thirdly, we use linux traffic control tool (TC) to simulate the unstable network connection on server S1, bandwidth is set to 1Mbps, and delay is set to 1 seconds. Fourthly, we pull image on server S3 and S4 for several times. After the image is completely downloaded, we can verify that some images are pulled from Docker Registry on server S1, and some images are pulled from server S2.
Through the above experiments, it is verified that under unstable connection conditions, edge nodes can pull images from their neighbour nodes through the peer-to-peer transmission mechanism, which eliminates a single point of failure in Docker Registry.

4.4. Concurrent Transmission Capability
In order to test concurrent transmission capability, we deploy three virtual machines on each server S2, S3, and S4 to simulate edge nodes through VirtualBox. Each virtual machine will run a Hadoop in docker. Then we test the concurrent transmission capability of our method in this paper by running Hadoop on these nine edge nodes. The results are shown in figure 5, we can see that the times of traditional images distribution method (DockerNative) have a linear growth trend when the number of concurrent nodes increases, because the bottleneck is the single Docker Registry. We also see that the
IDM-Dedup method has a better performance, because images can be pulled from not only Docker Registry, but also edge nodes.

5. Conclusion
In our paper, we have discussed the advantages of edge computing, and shortcomings of existing docker technologies under the condition of unstable network connections in edge environments. To solve the shortcomings on native docker container, such as large amount of image data transmission, network interruption and retransmission cost, this paper proposes an Images Distribution Method based on Deduplication (IDM-Dedup), which improves the transmission efficiency of container images and reduces the time to deploy and migrate edge services. The experimental results show that IDM-Dedup is more effective than native docker container method. As a future work, we will extend our work to the real environments in cellular network.

Acknowledgments
We gratefully acknowledge the efforts of everyone in our project. Also we would like to thank the editors and reviewers for their insightful suggestions.

References
[1] Liu, Fang, Guoming Tang, Y. Li, Zhiping Cai, Xingzhou Zhang and T. Zhou. A Survey on Edge Computing Systems and Tools. Proceedings of the IEEE, 107 (2019): 1537-1562.
[2] J. Pan and J. McElhannon. Future Edge Cloud and Edge Computing for Internet of Things Applications. IEEE Internet of Things Journal, vol. 5, no. 1, 439-449, 2018.
[3] Zhao Ziming, Liu Fang, Cai Zhiping, Xiao Nong. Edge Computing: Platforms, Applications and Challenges. Journal of Computer Research and Development, 2018, 55(2): 327-337.
[4] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE ACM Transactions on Networking, 24(5):2795–2808, 2016.
[5] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys and Tutorials, 19(3):1628–1656, 2017.
[6] Kai Peng, Victor C. M. Leung, Xiaolong Xu, Lixin Zheng, Jiabin Wang, Qingjia Huang. A Survey on Mobile Edge Computing: Focusing on Service Adoption and Provision. Wireless Communications and Mobile Computing, vol. 2018.
[7] Peng Liu, Dale Willis, and Suman Banerjee. Paradrop: Enabling lightweight multi-tenancy at the networks extreme edge. 1–13, 2016.
[8] Alsaleh, A. Can cloudlet coordination support cloud computing infrastructure? Journal of Cloud Computing, volume 7, 8 (2018).
[9] Ha, Kiryong, Yoshihisa Abe, Thomas Eiszler, Zhuo Chen, Wenlu Hu, Brandon Amos, Rohit Upadhyaya, P. Pillai and M. Satyanarayanan. You can teach elephants to dance: agile VM handoff for edge computing. ACM/IEEE Symposium on Edge Computing, 2017.
[10] Jang, Minsung, K. Schwan, Ketan Bhawdaj, Ada Gavrilovska and A. Avasthi. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. IEEE Conference on Computer Communications (2014): 2220-2228.