Impact of remote memory and network performance on execution performance of disaggregated micro data centers

Akishige IKOMA[†a)], ***Nonmember***, **Yuichi OHSITA**[†b)], ***Member***, ***and*** **Masayuki MURATA**[†c)], ***Fellow***

SUMMARY Micro data centers are small data centers located close to users that have a more limited amount of resources compared with traditional data centers. Therefore, the resources in micro data center need to be used efficiently to accommodate more applications. One approach for achieving efficient resource usage is to disaggregate the resources. In a disaggregated micro data center, stand-alone resources are connected via a network. In this data center, we can flexibly construct a virtual computer by connecting the required resources. However, performance degradation occurs due to communication delays between resources. In this paper, we evaluate the impact of communication performance on the performance of an application in a disaggregated micro data center. To evaluate the impact, we emulate an application running in a micro data center by adding a communication delay between the CPU and remote memory. As a result, we show that the latency has a larger impact on the performance of the application than does the bandwidth.

key words: *disaggregation, micro data center, remote memory*

1. Introduction

Many services are now provided through the cloud. These services utilize large-scale computational resources, memory, and storage in large-scale data centers, and handle a large amount of data that cannot be handled by end user devices. However, not all services can be provided through the cloud. Time-sensitive applications are difficult to provide through the cloud due to the large latency between cloud data center and end device. The increasing amount of data from IoT devices also becomes a problem if all data are sent to the cloud data center.

Edge computing is one technology for addressing this kind of problems [1]. In this approach, many small data centers (which we refer to as *micro data centers (μDCs)*) are deployed near users. Each μDC is close to the end devices and has more resources than do the end devices, which allows time-sensitive services to be provided by utilizing μDCs. However, since μDCs have a more limited amount of resources compared with large data centers, efficient resource utilization is necessary.

One approach to achieving efficient resource utilization is by disaggregating resources [2]. In this approach, each μDC is constructed of resources connected by a network as shown in Figure 1. We call these μDCs *disaggregated micro data centers (μDDCs)*. Unlike traditional data centers where each server has its own CPUs, RAM, and storage, we can flexibly use the resources in μDDCs by allocating the needed amount of resources to each task, giving optimal resource utilization [3]. In addition, the resources in μDDCs can be easily upgrade. If more memory is needed, we can add just memory modules, and if more computation resources are required, we can add just CPU modules.

However, the network used in a μDDC has an impact on the performance of applications running on μDDCs. In particular, the communication delay between the CPU and remote memory may degrade the performance [4]. Several methods for minimizing this performance degradation have been investigated. For example, Gao et al. used RDMA to speed up remote memory accesses [4] and Zervas et al. utilized high-speed communication technologies such as optical interconnects and optical switches [5]. However, these studies targeted large-scale parallel computing, and did not consider artificial intelligence (AI) applications such as image recognition that are expected to run on μDCs. Disaggregated architectures for AI applications [6], [7] have also been proposed. However, the impact of network performance on AI applications has not been investigated. In this paper, we evaluate the impact of network bandwidth and latency on the execution performance of AI applications.

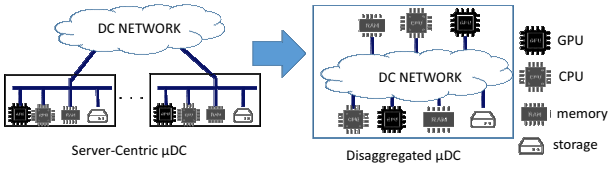

Fig. 1 Differences between a server-centric data center and a disaggregated micro data center

The remainder of this paper is organized as follows. Section 2 describes the evaluation environment and methodology, and Section 3 presents the evaluation results. In Section 4, we discuss the impact of the network performance on the performance of AI applications. Finally, Section 5 summarizes this paper and suggests topics for future work.

2. Evaluation Settings

2.1 μDDC

A μDDC is a μDC that consists of resource units such as CPU, GPU, and memory that are connected to each other

[†]The author is with Osaka University, Suita, Osaka 565-0871 Japan.
a) E-mail: E-mail: a-ikoma@ist.osaka-u.ac.jp
b) E-mail: E-mail: y-ohsita@ist.osaka-u.ac.jp
c) E-mail: E-mail: murata@ist.osaka-u.ac.jp

by a network and can communicate with each other. In this paper, each CPU in the μDDC has a small cache and runs applications using the data in the cache. When the required data don't exist in the cache, the CPU obtains the data from the remote memory. That is, the CPUs and the remote memories communicate with each other to run the applications. In this paper, we assume that the μDDC uses a paging technique. That is, if required data cannot be found in the cache, the CPU obtains the page that contains the required data from the remote memory. When the cache becomes full, some pages are moved to the remote memory.

In a μDDC, each application is run after allocating required resources and deploying required data in CPU caches and/or remote memories. In this paper, we focus on the performance of applications after completing allocation and deployment.

2.2 Evaluation environment

In this evaluation, we implement an evaluation environment for emulating a μDDC. This environment contains a CPU with a cache and a remote memory. Although they are implemented in a single computer, we add communication delays when accessing remote memory to emulate applications on a μDDC where communication delays occur when accessing remote memory. This evaluation environment is constructed by creating a swap device that acts as a remote memory. This swap device makes a process wait for the communication delay and then handles the request. In this evaluation, this swap device is created by using the memory installed in the computer. We allocate 200 MB of memory as the cache attached to the CPU and the rest of the memory is used as a swap device.

A computer with an Intel(R) Xeon(R) CPU E5-2687W and 64 GB DDR3 SDRAM is used to run the evaluation environment and the page size is set to 4 KB.

2.3 Evaluation method

In this evaluation, we use image classification as an example of services that run in μDDC. We use ResNet [8] and Inception-v3 [9] provided by Keras [10] as the models used for the image classification, and run their inference step by using TensorFlow. We run each of these processes 10 times, and measure the time from inputting one image to obtaining the output as the process time.

We define the performance degradation rate R as a metric for evaluating the impact of network performance by

$$R = \frac{T^{\text{disaggregated}} - T^{\text{traditional}}}{T^{\text{traditional}}}$$

where $T^{\text{disaggregated}}$ is the process time of μDDC and $T^{\text{traditional}}$ is the process time of the traditional computer. We obtain $T^{\text{disaggregated}}$ by running the process in the evaluation environment and $T^{\text{traditional}}$ by running the process on the same computer without adding communication delay.

We first investigate the impact of bandwidth by setting the bandwidth to 40 Gbps and 100 Gbps with the latency set to 8 μs. We next investigate the impact of the latency. We set the latency to 0.2 μs, 2 μs, and 8 μs with the bandwidth set to 100 Gbps.

3. Results

3.1 Impact of bandwidth on performance

Figure 2 shows the impact of the network bandwidth on the performance of μDDC. This figure shows all performance degradation rates monitored in our experiment. The median values of the performance degradation rates are also shown as a bar graph.

In our evaluation, the system processes in the OS may also affect the process time of each model since the application process may have to wait when a system process is using the CPU. Especially when the number of page faults is small, the impact of the communication delay is relatively small compared with that of system processes. In our evaluation, ResNet-50 is a relatively small model and the number of page faults is small. As a result, the impact of the communication delay is small. However, the communication delay has a large impact on the other models. The median of the performance degradation rates are over 10 % for all models except ResNet-50. We can discuss the impact of the bandwidth by comparing the performance degradation rates in the cases of 40 Gbps and 100 Gbps. Figure 2 shows increasing the bandwidth from 40 Gbps to 100 Gbps does not reduce the performance degradation rate. That is, the bandwidth has only a small impact on the performance of the application.

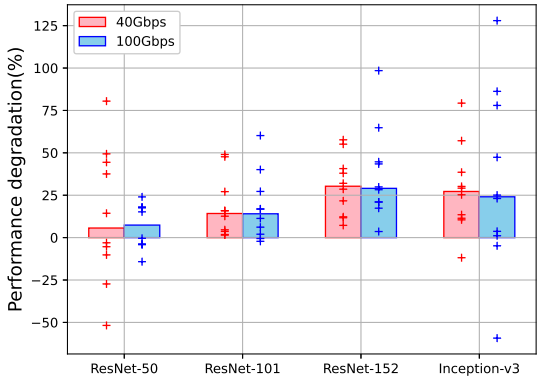

Fig. 2 Impact of network bandwidth on performance

3.2 Impact of latency on performance

Figure 3 shows the impact of the latency on the performance of the μDDC. Like Figure 2, all performance degradation rates monitored in our experiment are shown in this figure.

The median values of the performance degradation rates are also shown as a bar graph.

Figure 3 shows that the performance degradation rate decreases as the latency decreases. In all models, we can decrease the performance degradation rate by half by reducing the latency from 8 μs to 0.2 μs. That is, unlike the bandwidth, the latency has a large impact on the performance. Figure 3 also indicates that the performance degradation rate depends on the model. The performance degradation rate of a small model like ResNet-50 is small, while the performance degradation rate of a relatively large model like ResNet-152 is large. To investigate the impact of the communication delay, we focus on the time required to obtain the data from the remote memory.

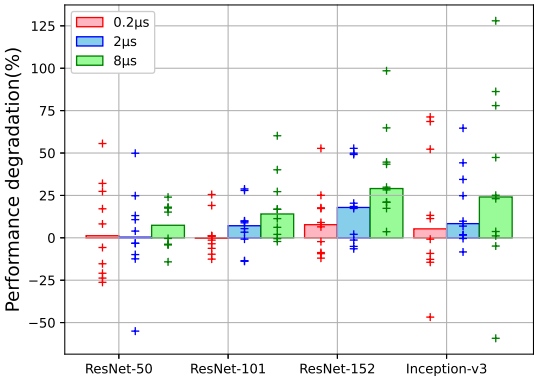

Fig. 3 Impact of latency on performance

Figures 4 and 5 show the timelines of memory accesses, where the time slots used by the CPU are shown in red and the time slots used to obtain data from the remote memory are shown in green. When the latency is small, the CPU can obtain the required data from the remote memory immediately. However, if the latency is large, it takes long time to obtain the required data. As a result, the green areas become larger in the case of 8 μs, and it takes more time to complete the application task.

Table 1 compares the ratio of the total time used to obtain the data from the remote memory. This shows that more than double the time is required to obtain the data when the latency is 8 μs, compared with the case of 0.2 μs. This also shows that the ratio for a small model like ResNet-50 is smaller than for the other models. This comparison correlates well with the comparison of the performance degradation rates. That is, the time to obtain the required data is the cause of the performance degradation.

4. Discussion

The above results indicate that the latency has a large impact on the performance of the application in μDDCs whereas the bandwidth has only a small impact. This difference is caused

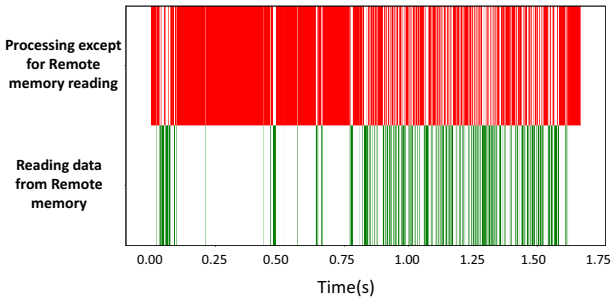

Fig. 4 Timeline of remote memory reading process and other processes in ResNet-152 with 8 μs latency

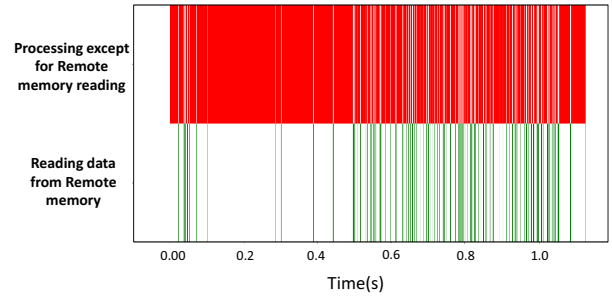

Fig. 5 Timeline of remote memory reading process and other processes in ResNet-152 with 0.2 μs latency

Table 1 Ratio of total time spent reading from remote memory to execution time for each model

	0.2μs	2μs	8μs
ResNet-50	7.94%	10.15%	18.44%
ResNet-101	10.96%	14.94%	27.73%
ResNet-152	12.50%	16.16%	27.31%
Inception-v3	9.93%	12.31%	22.24%

by the communication patterns in μDDCs. In our evaluation, we assume that the μDDC is using a paging technique. Thus, the CPU obtains only 4 KB of data every time a page fault occurs. The large bandwidth cannot significantly reduce the time required to obtain the data because the data size is not large. However, since the latency affects the time required to obtain the data, the latency has a large impact on the performance.

However, our results indicate that the computational resources are also important. Even in the worst case, 3/4 of the process time is consumed by the CPU while the remote memory access only consumes 1/4 of the process time. That is, we need to allocate sufficient computational resources to the applications.

5. Conclusion

In this paper, we evaluated the impact of communication performance on the performance of applications in μDDCs. To evaluate the performance of μDDCs, we emulated an application running in a μDDC by adding communication delays between the CPU and remote memory. As a result, we showed that the latency has a larger impact on the per-

formance of the application than bandwidth. In addition, our results indicate that computational resources are also important for the performance of μDDCs.

As future work, we intend to propose a method for allocating resources in μDDCs to multiple applications based on the results of this paper. In addition, we intend to investigate the network architectures suitable for μDDCs based on the results.

Acknowledgement

This work is partially supported by NICT.

References

[1] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, pp. 94–120, Jan. 2018.

[2] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, "Network support for resource disaggregation in next-generation datacenters," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pp. 1–7, Nov. 2013.

[3] Y. Cheng, R. Lin, M. De Andrade, L. Wosinska, and J. Chen, "Disaggregated data centers: Challenges and tradeoffs," *IEEE Communications Magazine*, Mar. 2019.

[4] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 249–264, Nov. 2016.

[5] G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, "Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, pp. A270–A285, Feb. 2018.

[6] Y. Kwon and M. Rhu, "A disaggregated memory system for deep learning," *IEEE Micro*, vol. 39, no. 5, pp. 82–90, 2019.

[7] Y. Kwon, Y. Lee, and M. Rhu, "Tensordimm: A practical near-memory processing architecture for embeddings and tensor operations in deep learning," pp. 740–753, 10 2019.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, June 2016.

[10] "Keras: Deep learning for humans." `https://github.com/keras-team/keras`, last accessed on 2021-8-29.