

FlowOS-RM: Disaggregated Resource Management System

Ryousei Takano*, Kuniyasu Suzuki*, Hidetaka Koie*

* *Information Technology Research Institute*

National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan

{takano-ryousei, k.suzaki, koie-hidetaka}@aist.go.jp

I. MOTIVATION AND GOALS

A traditional data center consists of monolithic-servers and the software stack is built on top of them. When used for various AI and Big Data workloads, such architecture faces limitations including lack of operational flexibility, low resource utilization, low maintainability, etc.

Moreover, task-specific accelerators such as Google TPU, Fujitsu DLU, Microsoft BrainWave, and D-Wave Quantum Annealer are emerging. Utilization of such heterogeneous accelerators is the key to achieve sustainable performance improvement in the post-Moore era. From the view point of system, holistic resource management is essential, that is, a user is easily able to combine components including a generic processor and an accelerator for the sequence of a job.

Resource disaggregation is a promising solution to address the limitations of traditional data centers. In a disaggregated data center, CPU, accelerator, memory, and storage are separated and they are interconnected via a high-speed network. We propose a concept of disaggregated data center architecture, **Flow-in-Cloud**, that enables an existing cluster to expand an accelerator pool through a high-speed network. This poster demonstrates the feasibility of the prototype system using a distributed deep learning application.

II. FLOW-IN-CLOUD: DISAGGREGATED DATA CENTER ARCHITECTURE

Flow-in-Cloud (FiC) is a shared pool of heterogeneous accelerators such as GPU and FPGA, which are directly connected by a circuit-switched network. From the pool of accelerators, a slice is dynamically configured and provided according to a user request, as shown in Figure 1. FiC network comprises a set of FiC switch boards that have FPGA (Xilinx Kintex Ultrascale XCKU095), 32-10Gbps serial connections, DRAM, and Raspberry Pi 3 (controller). A circuit-switching logic and a user-defined logic written in a high-level synthesis language are running on the FPGA, and the latter logic is partially reconfigurable in advance of an application deployment. Note that this poster uses ExpEther [1] [2] instead of FiC network to disaggregate accelerators from servers.

FlowOS manages the entire FiC resources, and supports execution of a user job on provided slices. FlowOS employs a

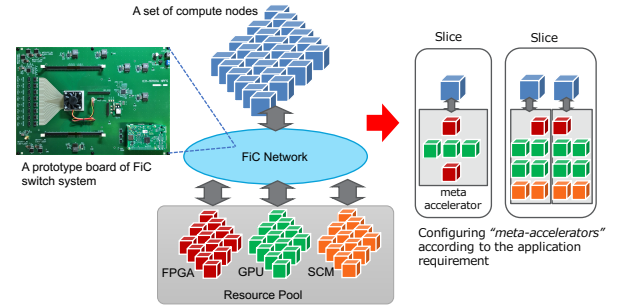


Fig. 1. The overview of Flow-in-Cloud Architecture

layered architecture including FlowOS-API, FlowOS-RM, and FlowOS-drivers. This poster focuses on FlowOS-RM.

III. FLOWOS-RM

FlowOS-RM seamlessly works in cooperation with a cluster resource manager such as Apache Mesos [3], Kubernetes, SLURM, and so on. FlowOS-RM provides users with the REST API to configure a slice and execute a job on it. FlowOS-RM supports a single-node task and an MPI type multi-node task.

To implement such a mechanism, FlowOS-RM combines the following components: (1) Disaggregate device management: ExpEther is a PCIe-over-Ethernet technology and it allows us to dynamically attach and detach remote PCIe devices through Ethernet. (2) OS deployment: Bare-Metal Container (BMC) [4] constructs an execution environment to run a Docker image with an application optimized OS kernel on a node. (3) Task scheduling and execution: FlowOS-RM is implemented on top of a Mesos framework, and it co-allocates nodes to meet a user requirement and launches a task on each node in the manner of Mesos.

Figure 2 presents a job execution flow in FlowOS-RM, where a job is a set of tasks and each task runs on a node belonging to a slice. Firstly, constructing a slice is divided into two steps. (1) *attach-device* attaches disaggregated devices to a node. (2) *launch-machine* turns on the power of a node with a specific OS kernel and a Docker container. Secondly, launching a job is divided into two steps. (3) *prepare-task* does housekeeping for launching a task in the manner of Mesos. (4) *launch-task* runs a task in a node. Finally, destructing a slice

is divided into two steps. (5) *detach-device* detaches devices from a node. (6) *destroy-machine* turns down a node.

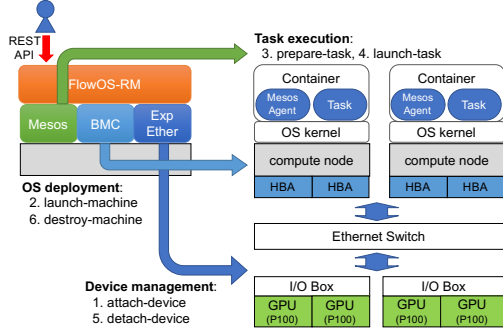


Fig. 2. Job execution flow in FlowOS-RM

IV. EXPERIMENT

A. Experimental Setting

In order to demonstrate the feasibility of FlowOS-RM, we have conducted distributed deep learning experiments on a four-node cluster environment as shown in Figure 3a. An MNIST application on ChainerMN [5] is used as a benchmark program. Each compute nodes have two ExpEther HBAs to connect PCIe devices on I/O Boxes through a 40 GbE switch.

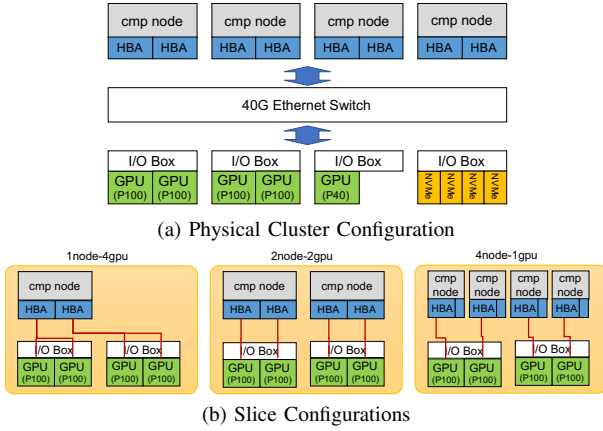


Fig. 3. Experimental Configuration

B. Experimental Results

1) *Slice construction/destruction overhead*: We ran an MNIST application on three slice configurations as shown in Figure 3b, and the breakdown in the execution time for each slice is shown in Figure 4a. The MNIST training run faster as the number of GPUs per node increases. Here we focus on the overhead of slice construction and destruction operations. A *launch-machine* operation takes longer as the number of nodes increases, because downloading a container image the size is about 3GB through GbE becomes the bottleneck. Some operations including *attach/detach-device* and *launch-task* take longer as the number of GPUs per node increases, because these operations are not parallelized. We

plan to reduce the above overhead by using a faster NIC and parallelizing operations.

2) *Resource sharing*: We have confirmed disaggregated resources are sharing among several slices according to a user requirement. In this experiment, a user submitted four jobs and FlowOS-RM allocated resources into each slice in the FIFO manner. The slice configurations of each job are as follows: Slice1 and 2 consist of 2node-2gpu (P100), Slice3 consists of 1node-1gpu (P40), and Slice4 consists of 4node-4gpu (P100). Figure 4b shows that resource sharing among slices works as expected.

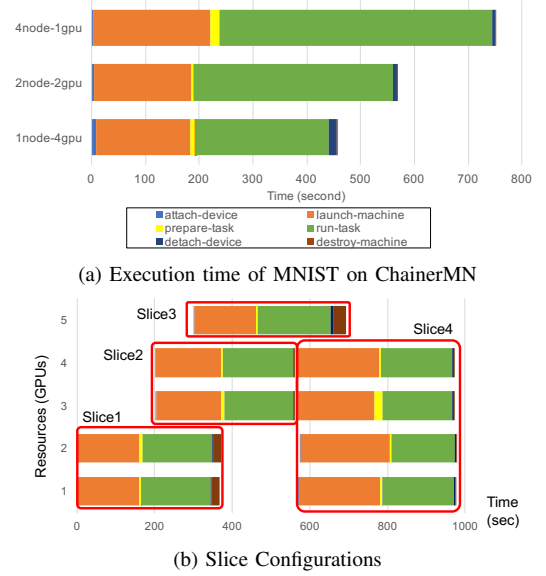


Fig. 4. Experimental Results

V. CONCLUSION AND FUTURE WORK

We have demonstrated effective resource sharing on the proposed disaggregated resource management system for AI and Big Data applications. We found some performance issues but the impact is limited for long hours-running applications. We plan to evaluate various applications with applying performance optimization techniques such as PGO [6] on this system.

REFERENCES

- [1] J. Suzuki, Y. Hidaka, J. Higuchi, T. Yoshikawa, and A. Iwata, "Expressether - ethernet-based virtualization technology for reconfigurable hardware platform," in *14th IEEE Symposium on High-Performance Interconnects (HOTI)*, 2006, pp. 45–51.
- [2] ExpEther Consortium, <http://www.expether.org>, 2018.
- [3] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *NSDI*, vol. 11, 2011, pp. 22–22.
- [4] K. Suzaki, H. Koie, and R. Takano, "Bare-metal container — direct execution of a container image on a remote machine with an optimized kernel —," in *18th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2016, pp. 25–36.
- [5] Preferred Networks, "ChainerMN," <https://github.com/chainer/chainermn>, 2018.
- [6] K. Suzaki, H. Koie, and R. Takano, "Profile guided kernel optimization for individual container execution on bare-metal container," in *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (Poster)*, 2017.