

FlowOS-RM: Disaggregated Resource Management System

Ryousei Takano, Kuniyasu Suzuki, Hidetaka Koie
Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST), Japan

Motivation

- 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 flexibility, low resource utilization, etc.
- Task-specific accelerators such as Google TPU 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.

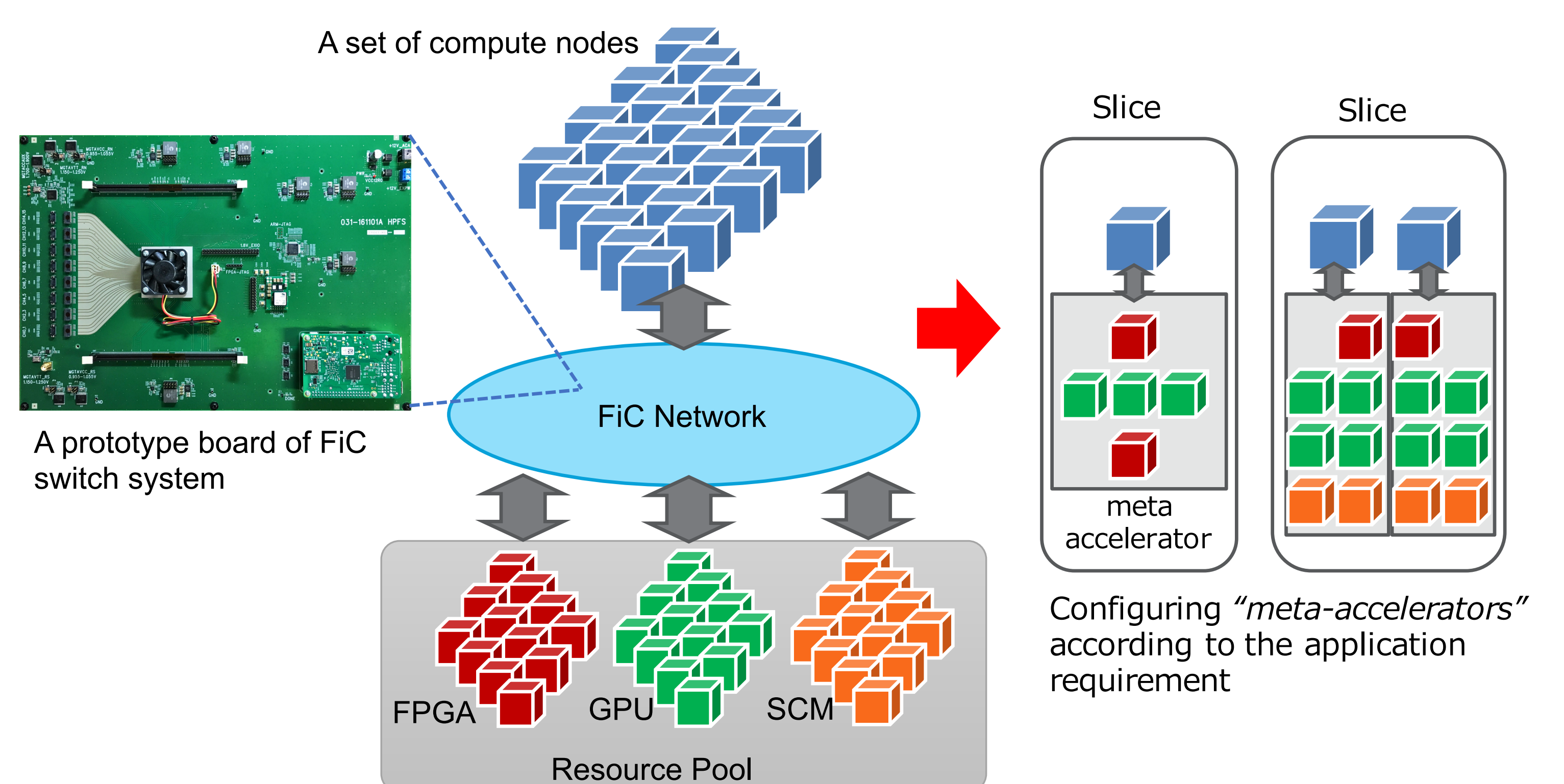
Goals

- Resource disaggregation is a promising solution to address the limitations of traditional data centers.
- 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.
- We demonstrate the feasibility of the prototype system using a distributed deep learning application.

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's request.
- **FlowOS** manages the entire FiC resources, and supports execution of a users' program on provided slices.
- FiC switch system has been developed. Note: In this poster, we use ExpEther [4] instead of FiC network to disaggregate accelerators from servers.

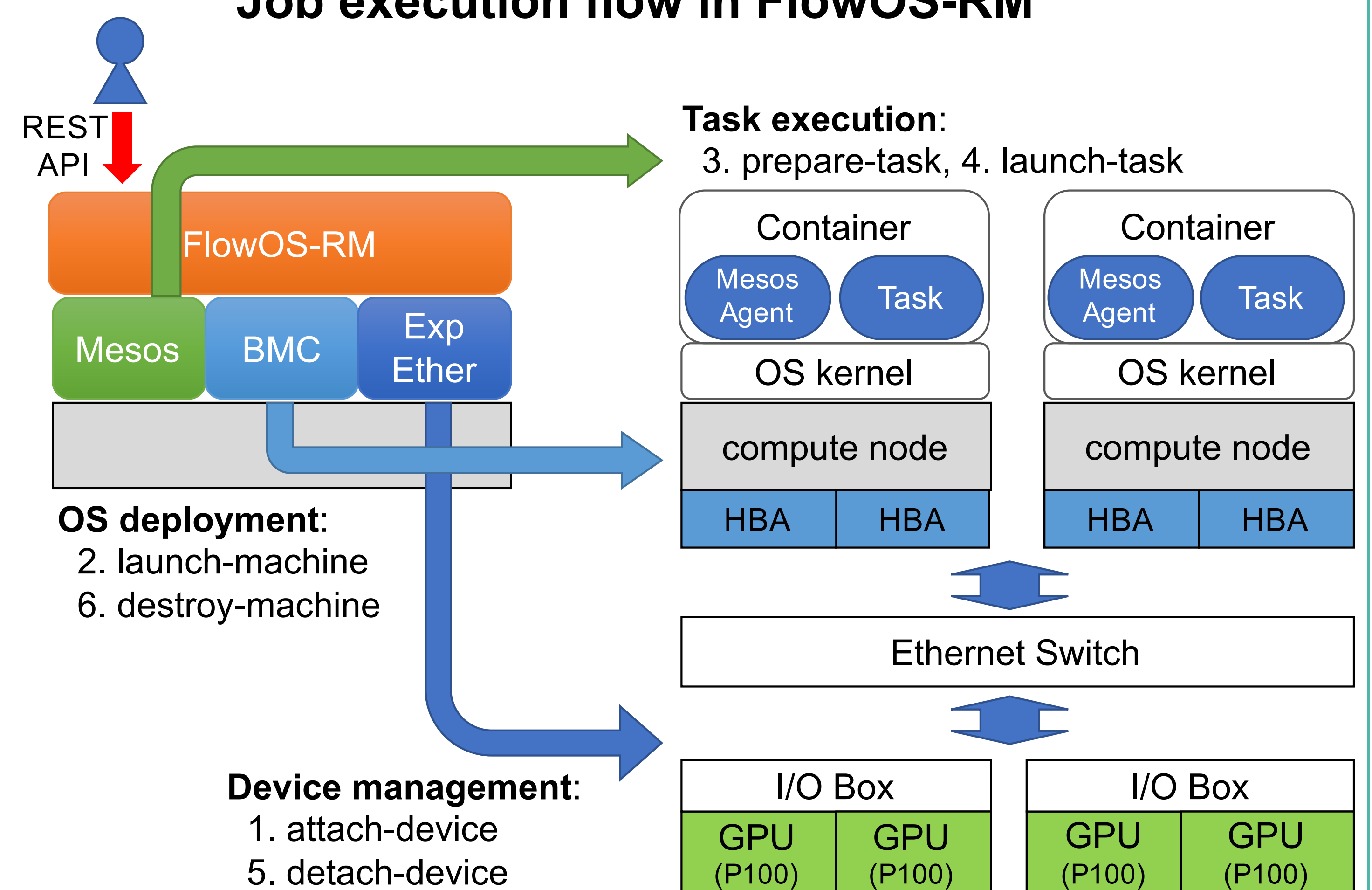
The overview of Flow-in-Cloud (FiC)



FlowOS-Resource Manager (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 [4] 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) [1][2] 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 user requirements and launches a task on each node in the manner of Mesos.

Job execution flow in FlowOS-RM



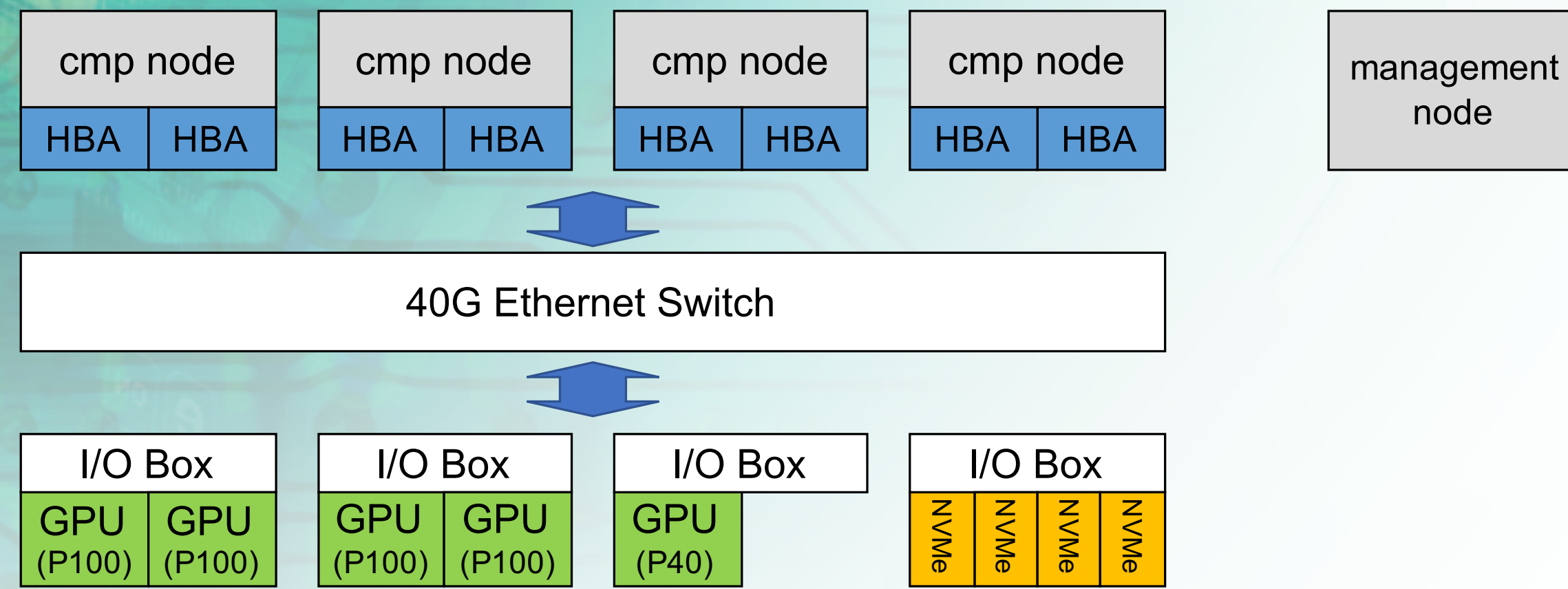
attach-device	Attach devices to a node
launch-machine	Boot a node with a specific OS kernel and container and join active nodes under the Apache Mesos
prepare-task	Do housekeeping for launching a task, including submitting a task to the corresponding node through the Apache Mesos
launch-task	Launch a task in a node (running state of a task)
detach-device	Detach devices from a node
destroy-machine	Shutdown a node and leave active nodes

Experiment

Experimental Setting

- In order to demonstrate the feasibility of FlowOS-RM, we have conducted distributed deep learning experiments on a four-node cluster environment. 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.
- FlowOS-RM built three slice configurations: 4node-1gpu, 2node-2gpu, and 1node-4gpu.

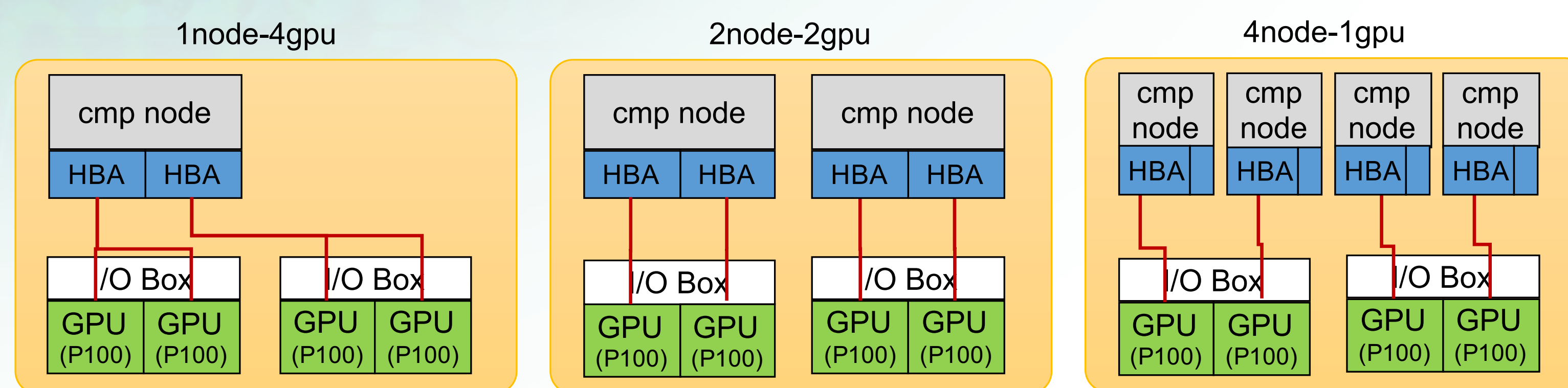
Physical Cluster Configuration



Compute Node Configuration

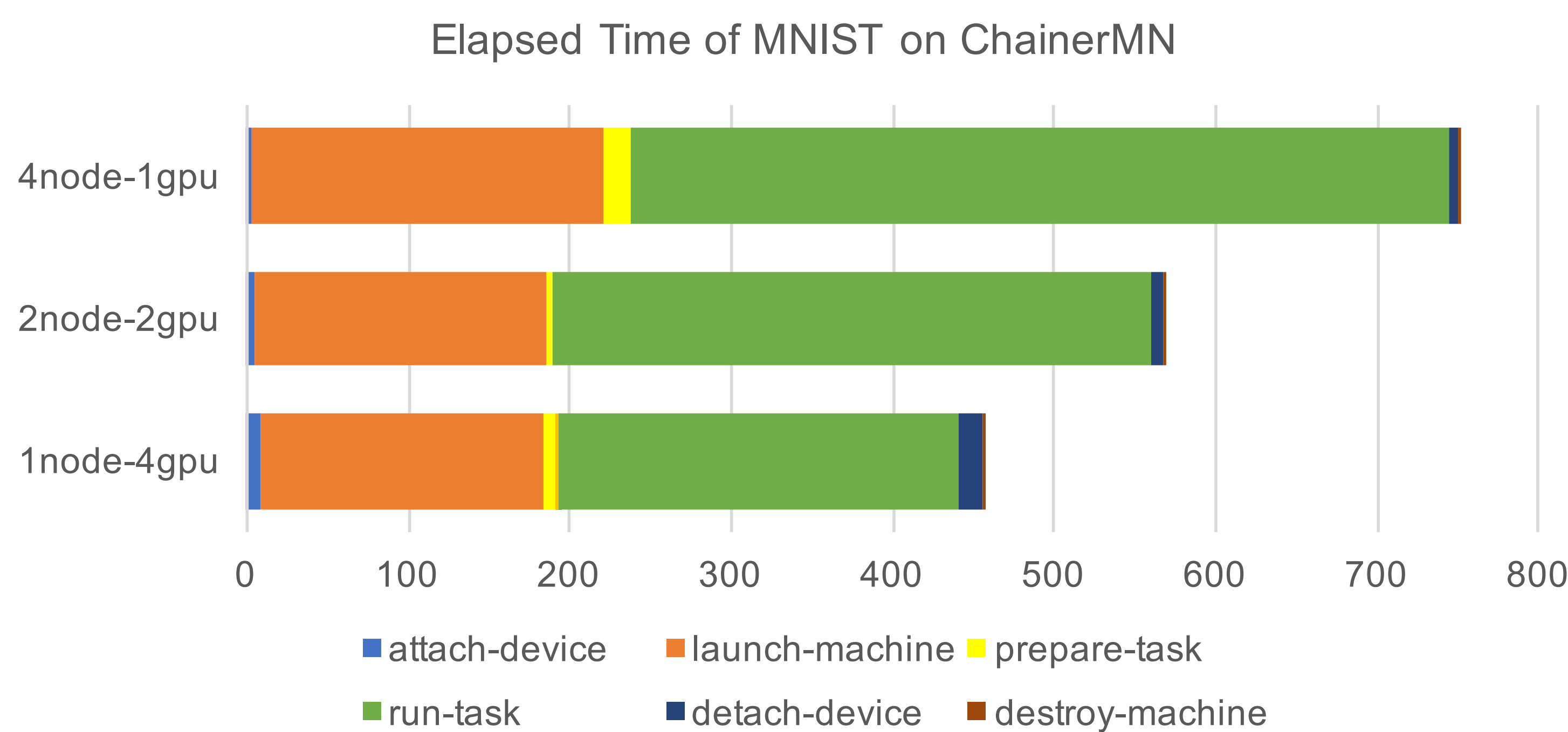
CPU	Intel Xeon E5-2630v4/2.2GHz
M/B	Supermicro X10SRG-F
Memory	128 GB DDR4-2133
NIC	Intel I350 (Gigabit Ethernet)
Disaggregated Resources (PCIe device)	
GPU	NVIDIA Tesla P100 x4, P40 x1
NVMe	Intel SSD 750 x4
Software Configuration	
OS	CentOS 7.4
	Mesos 1.4.1, ChainerMN, OpenMPI 3.1.0, CUDA 8.0.61

Slice Configurations



Experimental Results

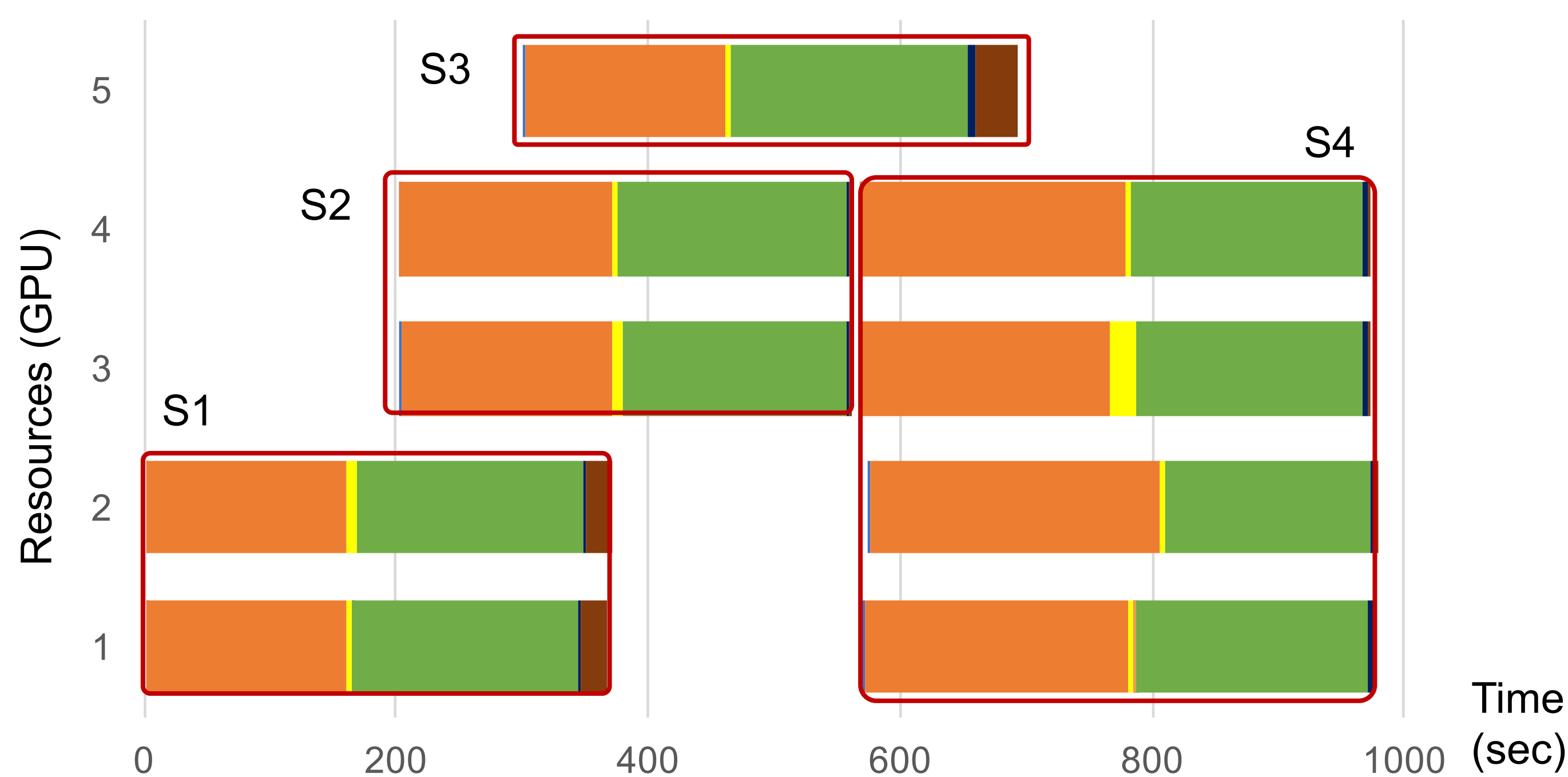
Slice construction/destruction overhead



- 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.
- The MNIST training run faster as the number of GPUs per node increases as below:

Slice configuration	Elapsed Time (sec)
4node-1gpu	366.36
2node-2gpu	237.31
1node-4gpu	104.57

Resource Sharing



- We have confirmed disaggregated resources are sharing among several slices according to 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.
- S1 and S2: 2node-2gpu (P100)
- S3: 1node-1gpu (P40)
- S4: 4node-4gpu (P100)

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 [2] on this system.

Reference

- [1] K. Suzuki, et al, "Bare-Metal Container", IEEE HPCC 2016. (<https://github.com/baremetalcontainer/bmc>)
- [2] K. Suzuki, et al, "Profile Guided Kernel Optimization for Individual Container Execution on Bare-Metal Container", ACM/IEEE SC 2017 (poster)
- [3] Apache Mesos, <http://mesos.apache.org/>
- [4] ExpEther, <http://www.expether.org/>
- [5] ChainerMN, <https://github.com/chainer/chainermn>

Acknowledgement: This work is partially based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), Japan.