Deep Learning Workload Analysis for Efficient Resource Allocation

Sayaka Takayama†
sayaka-t@ogl.is.ocha.ac.jp
Takashi Shiraishi†† Shigeto Suzuki††
Masao Yamamoto†† Yukihiro Watanabe†† Masato Oguchi†

†Ochanomizu University, Japan
††FUJITSU LABORATORIES LTD., Japan

About Me

- Name：Sayaka Takayama
- Country：Japan
- Affiliation：Master course of Ochanomizu University
- Major：Computer Science

Table of Contents

- Background
- Related Work
- Purpose
- Experiments
- Results
- GPU Priority Assignment
- Conclusions
- Future Work

Background

With the prosperity of deep learning, GPUs became popular as specialized hardware accelerators

* Compared with CPUs, GPUs are still scarce and valuable computing resources

* Deep learning does not fit well with current configuration practices and deployment models, which assume a static allocation of GPUs

Related Work - Existing Products

Ochanomizu University

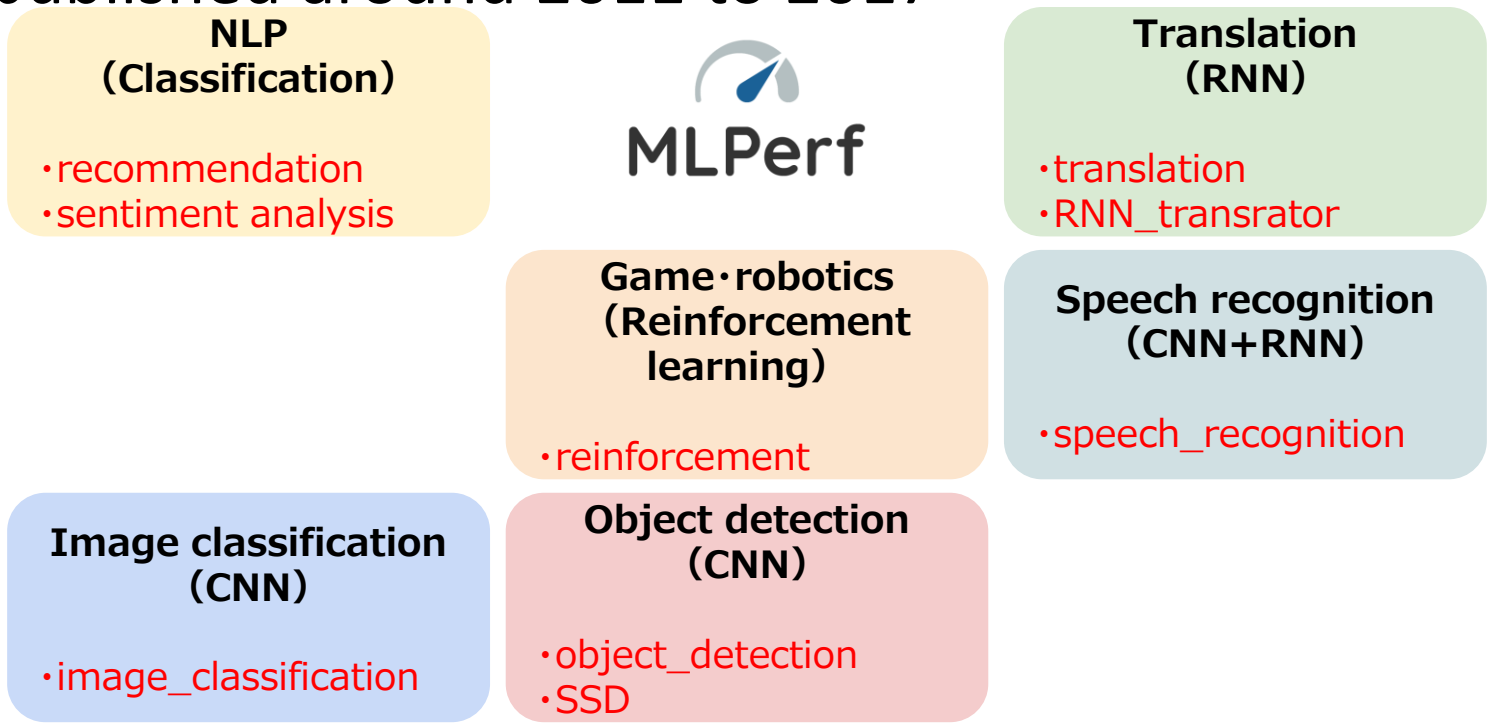
- FBLearner (Facebook)
  - Toolkit to simplify the tasks of leveraging machine learning for Facebook products
  - Schedules jobs and allocate resources on shared pool of GPUs and CPUs by using internal job scheduler
- Haswell (Intel)
  - CPU architecture which execute Dynamic Voltage and Frequency Scaling
  - Switches the voltage and operate frequency according to the load for each CPU core or cluster
- HGX-2 (NVIDIA)
  - Cloud server platform which provides high-precision calculation functions that can handle both HPC and AI workloads

Purpose

- GPU allocation control based on the difference in application performance when using different GPUs
  - Evaluate the performance of benchmarks executed on a framework that uses GPUs and compared the performance on machines of different generations
  - Estimate whether GPU Priority Assignment would lead to improvement using the data we collected

Experiments - MLPerf

- AI benchmarks supported by companies, such as Google, Intel, Baidu, and NVIDIA https://mlperf.org/
- Includes 9 benchmarks and based on papers published around 2011 to 2017



Experiments - MLPerf

Benchmark	Detail	Model	Base
image classification (IC)	Image classification by Resnet of TensorFlow	Resnet-50 v1	CNN
single stage detector (SSD)	Object detection by Single Shot MultiBox	Single Shot MultiBox	CNN
object detection (OD)	Object detection by mask and box mAP for metrix	Mask R-CNN	CNN
recommendation (RM)	Recommendation by learning binary information of user's interaction with item	Neural Collaborative Filtering	Collaborative Filtering (CF)
sentiment analysis (SA)	Binary classification with positive and negative from movie review	Seq2-BowN-CNN	CNN
rnn translator (RT)	Translate WMT16(English-German) into RNN	GNMT v2	RNN
translation (TL)	Translate WMT17(English-German) into attention mechanism	Transformer	RNN
speech recognition (SR)	Convert audio samples to text	Deep Speech2	CNN+RNN
reinforcement (RF)	Reinforcement learning of minigo	MiniGo(based on Alpha Go)	Monte Carlo(MC)

Experiments - Job Execution Conditions

Benchmark	epoch (step, iteration)	SEED	Job execution time (h:min:s)
image classification (IC)	53200(step)	1	3:01:13
single stage detector (SSD)	11	1	3:09:12
object detection (OD)	25000(iteration)	3	2:23:26
recommendation (RM)	6	1	1:09:34
sentiment analysis (SA)	100	1	1:22:50
rnn translator (RT)	1	1	2:48:18
translation (TL)	17200(step)	1	2:59:55
speech recognition (SR)	1	1	10:53:32
reinforcement (RF)	4	1	5:46:00

Experiments - Zabbix

- System monitoring software for centrally monitoring applications such as server and network
- Acquisition information
  - Obtain information at 1-minute intervals for about 3 hours
  - IPMI：Infrastructure information such as power and humidity
  - Perf：Information that can be acquired by the OS (CPU usage rate, memory usage rate, disk access speed)
  - nvidia-smi：Information about GPU
  - PMU：Obtains the number of instructions from the CPU event information and the number of memory accesses from the cache miss rate
  - Focus on Processor utilization and Memory utilization
  - Used PHPZabbixAPI for data collection https://github.com/confirm/PhpZabbixApi

Experiments - Agent Server

- OS：ubuntsu 16.04
- Server：FUJITSU Primergy RX2540 M4
- CPU：Intel Xeon Skylake 2sockets 20cores
  2.4GHz Gold 6148 150W
- GPU：NVIDIA Tesla V100 16GB
- Storage M2.SSD：290GB read 0.87GB/s write 1.75GB/s
- Memory：192GB DDR4 2666MHz
- Python：3.50
- CUDA：9.2

Experiments - Server for comparison・GPU

- OS：CentOS Linux release 7.5.1804 (Core)
- Server：FUJITSU Primergy CX400 M1
- CPU： Intel Xeon Haswell 2sockets 14cores
  2.6GHz E5-2697 145W
- GPU：NVIDIA Tesla P100 16GB
- Storage：HDD 270GB read 0.21GB/s write 1.07GB/s
- Memory：256GB DDR4 2133MHz
- Python：3.50
- CUDA：9.2

GPU	Cores	MHz	FP16 (TFLOPS)	FP32 (TFLOPS)	FP64 (TFLOPS)	Band width (GB/s)	Release
P100	3584	1300	18.636	9.318	4.659	720	2016/4
V100	5120	1455	119.19	14.90	7.45	900	2017/5

Experiments - Environments

Monitoring server

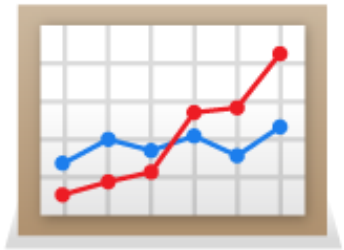
IPMI, Perf,
nvidia-smi, PMU

Agent server

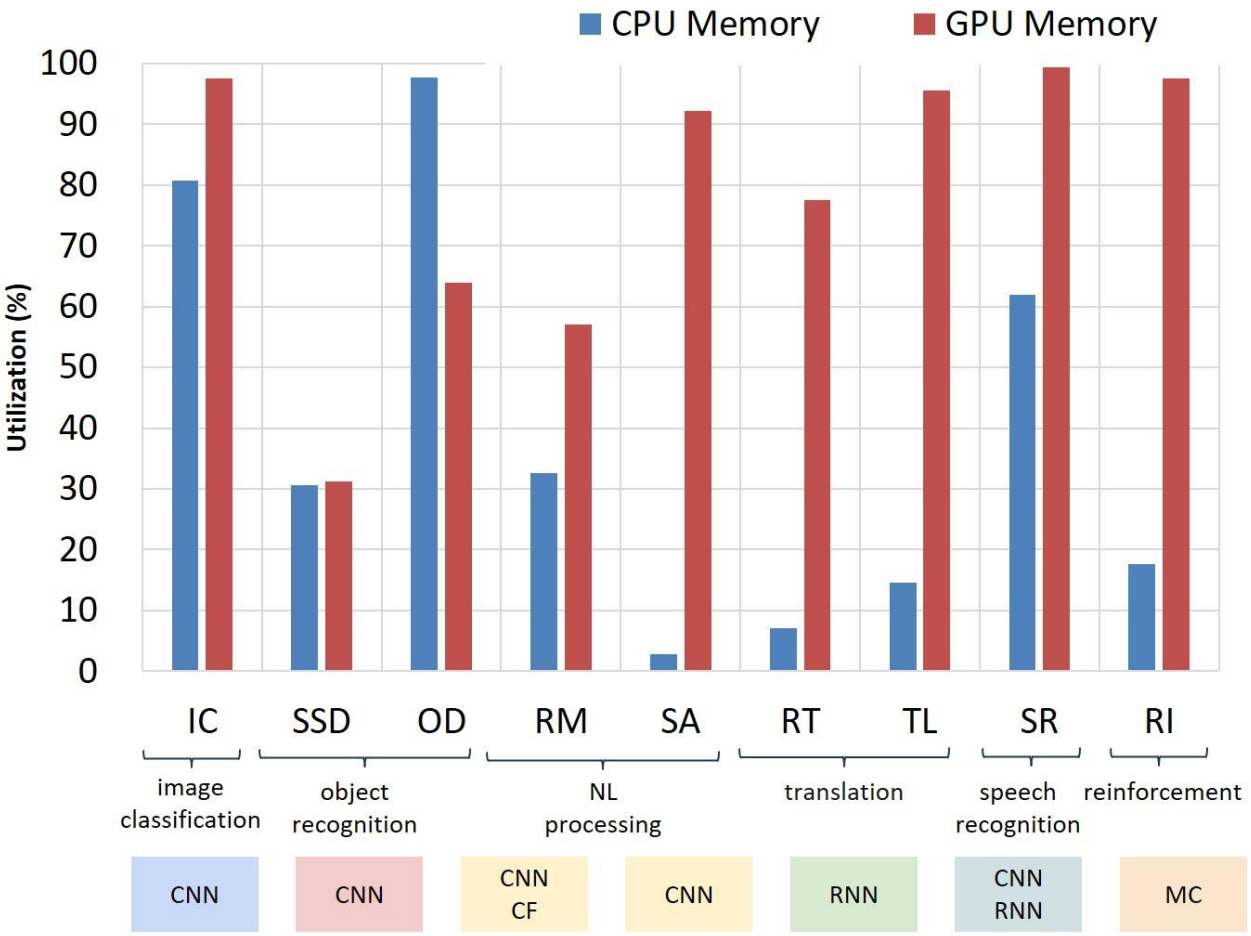
Get hardware information

AI Application

Analyze results when running each benchmark



Results – Memory utilization



- Compared with the memory utilization of a CPU, that of a GPU is large
  - capacity of GPU memory
- In this conditions, the total memory is sufficient, and the effect on job performance is considered to be small

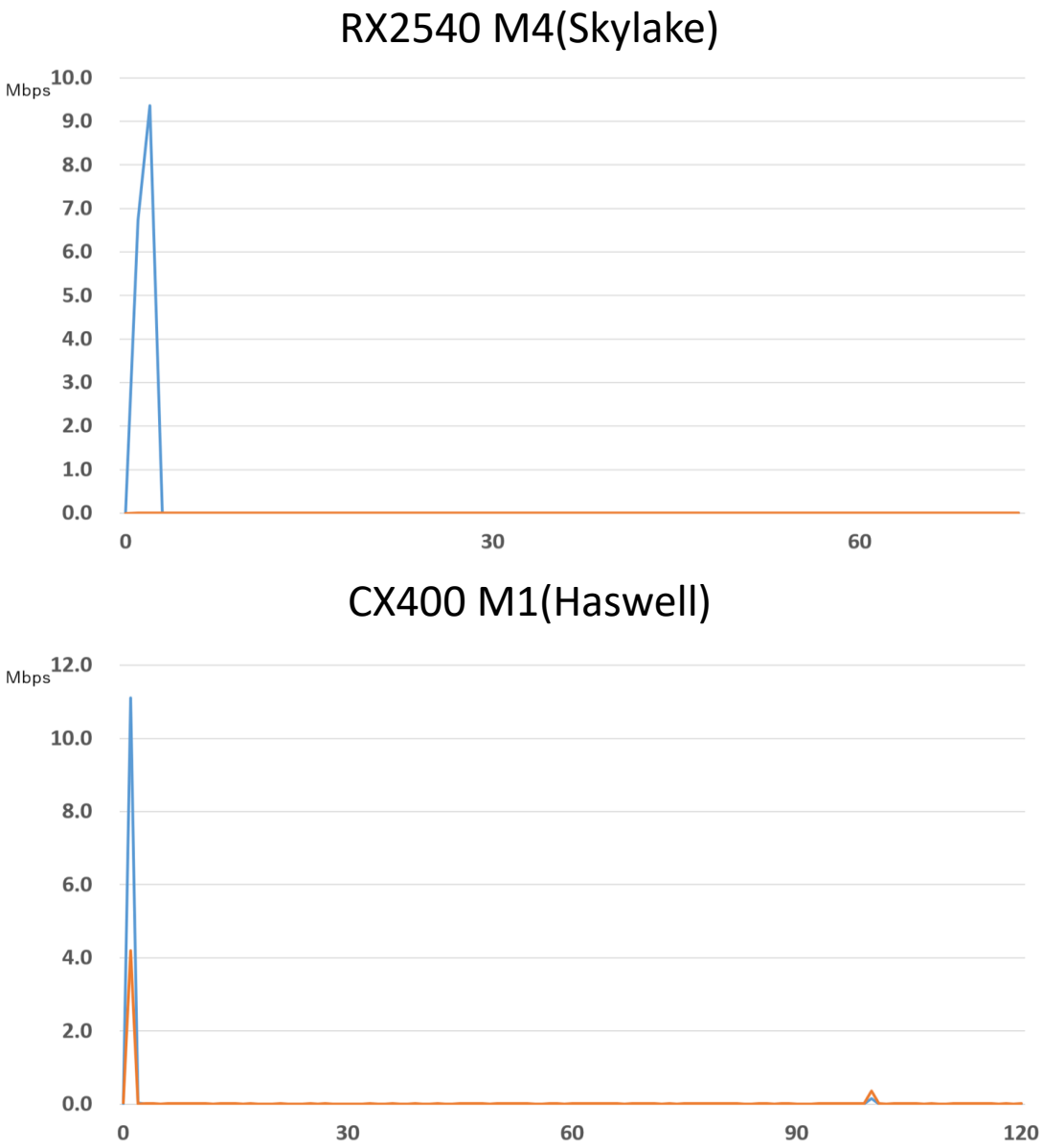
Results – Disk Access Speed

Benchmark	Read (**MB/s**)	Write (**MB/s**)
IC	73.92	111.32
SSD	6.75	10.96
OD	4.66	1.25
RM	2.67	29.26
SA	9.82	0.01
RT	29.50	0.06
TL	26.14	93.90
SR	7.10	8.05
RF	22.13	0.75

Maximum disk access speed for each benchmark on the Primergy RX2540 M4

- There is a margin for disk performance
  - The maximum disk access is 0.87 GB/s for reading and 1.75 GB/s for writing

Results – Disk Access Speed



Disk access speed of Sentiment Analysis

—: DISK IO read

—: DISK IO write

- The disk performance difference has a small impact on the benchmark
  - The maximum value of the disk access speed is sufficient for the server disk performance
  - The time required for disk access is extremely short

Results - Comparison of CPU

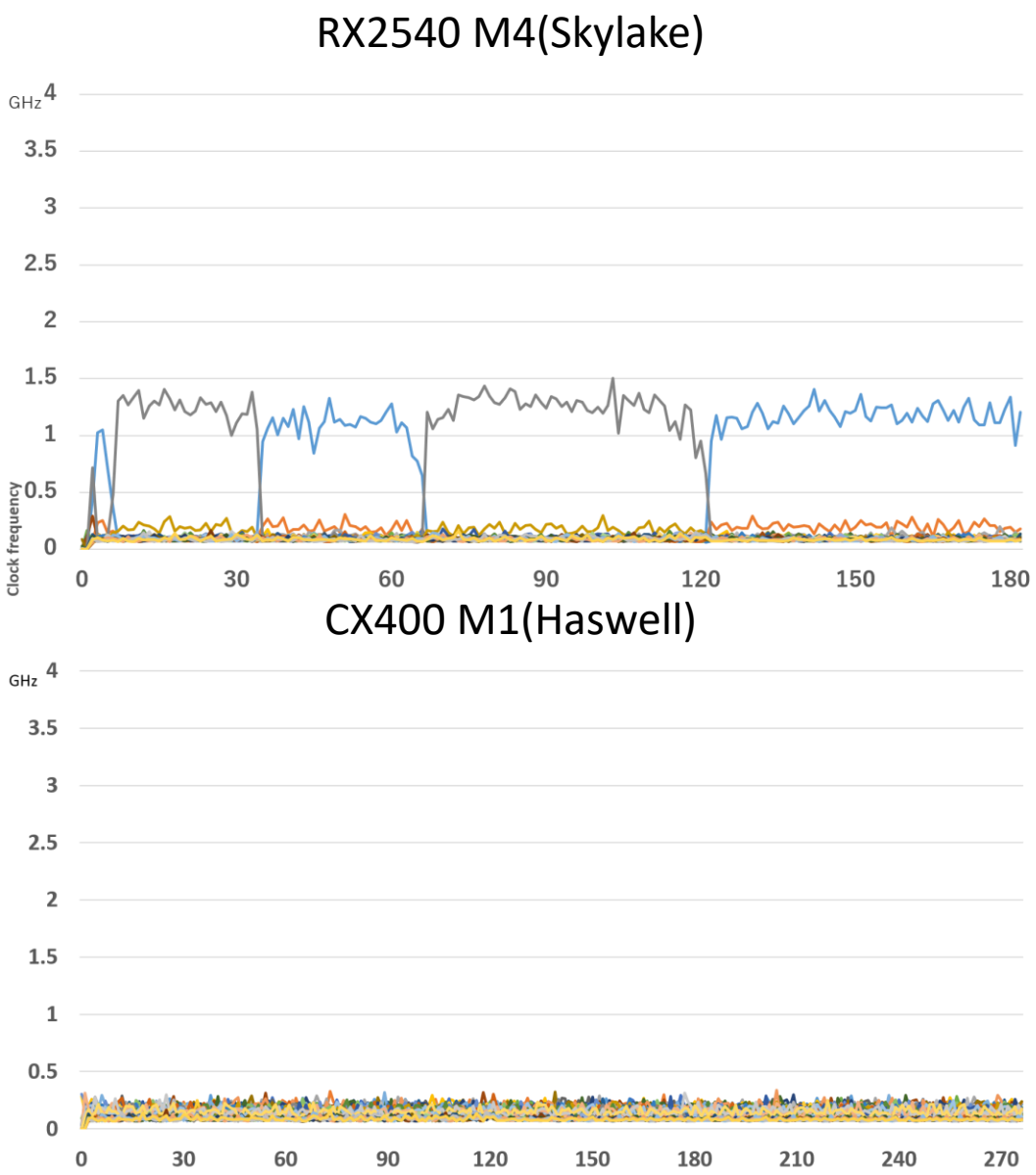
Benchmark	Haswell (h:min:s)	Skylake (h:min:s)	Skylake / Haswell
IC	4:36:10	4:33:04	0.99
SSD	3:56:42	3:12:18	0.81
OD	2:59:34	2:57:51	0.99
RM	1:12:00	1:09:07	0.96
SA	2:00:12	1:48:14	0.90
RT	4:06:41	4:05:28	1.00
TL	4:37:47	4:29:21	0.97
SR	-	15:07:34	-
RF	6:28:00	6:08:37	0.95

Comparison of job execution time when CPU is changed

- Server changed from RX2540 M4 (CPU: Skylake) to CX400 M1 (CPU: Haswell)
- GPU：P100

- Unlike the GPU comparison, no difference in job time characteristics
- The difference in the job performance due to the changes in the CPU performance is small

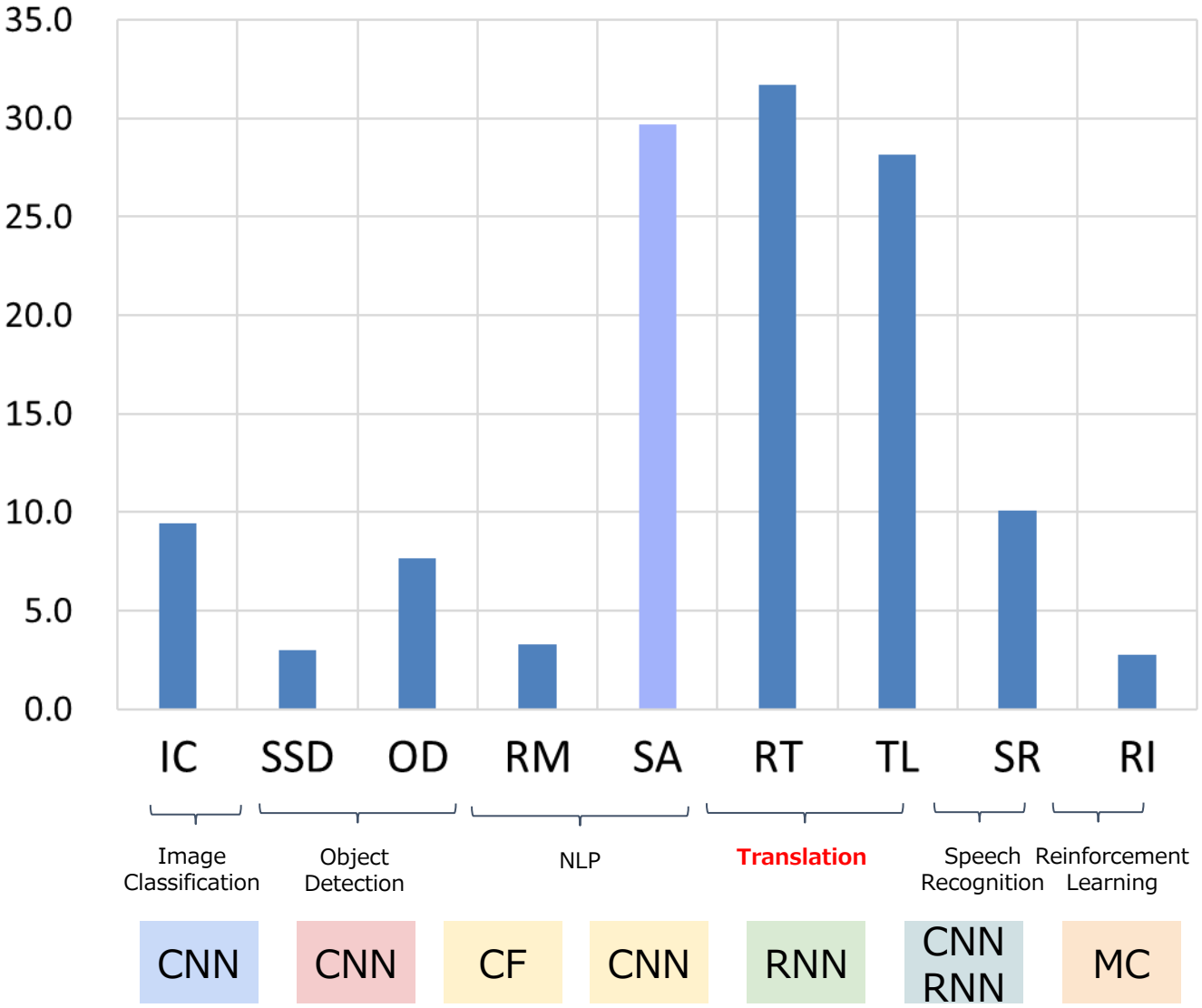
Results – Clock Frequency



Clock frequency per thread of Sentiment Analysis

- In Skylake, only certain threads have high clock frequency
  - Specification to use only some threads when the frequency drops

Results - GPU/CPU utilization



- The overall application tends to be GPU-necked
- Translation-based applications require a particularly large amount of GPUs
- SA has a larger value than other CNN bases
  - Simple binary classification

Results - Average GPU･CPU utilization

Benchmark	CPU Utilization (%)	GPU Utilization (%)
IC	5.1	95.4
SSD	9.9	59.8
OD	4.9	74.5
RM	6.7	44.3
SA	1.4	82.0
RT	1.5	95.5
TL	1.5	83.9
SR	3.2	65.1
RF	11.4	62.7

- Overall low CPU usage and high GPU usage
- There are also applications in which CPU performance is considered to be important, such as RM and SSD

Results - Comparison of GPU

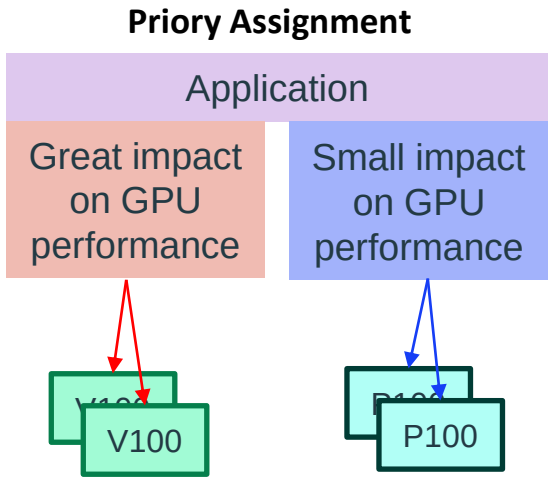
Benchmark	P100 (h:min:s)	V100 (h:min:s)	V100 / P100
IC	4:33:04	3:01:13	0.66
SSD	3:12:18	3:09:12	0.98
OD	2:57:51	2:23:26	0.81
RM	1:09:07	1:09:34	1.01
SA	1:48:14	1:22:50	0.76
RT	4:05:28	2:48:18	0.68
TL	4:29:21	2:59:55	0.67
SR	15:07:34	10:53:32	0.72
RF	6:08:37	5:46:00	0.93

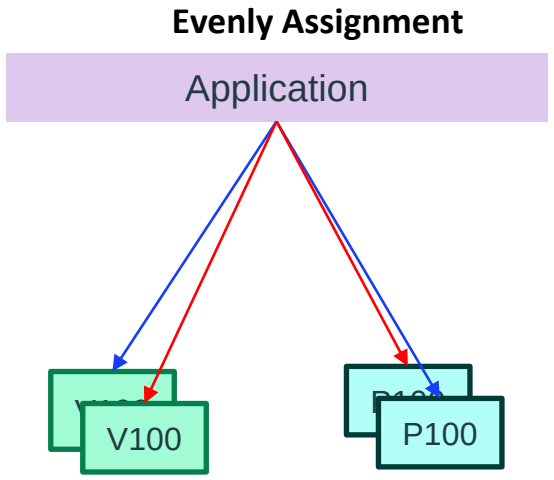
- Compared to the average GPU utilization, the change in the job time when changing the GPU is larger for the benchmarks with higher average GPU utilization
- A job with a high GPU utilization shows a high job performance improvement due to changes in the GPU performance

GPU Priority Assignment

- Assumed one application fully uses a physical machine in a Docker environment
- P100 or V100 is allocated to each machine which handles MLPerf

- **Priority Assignment**：Prioritize new GPUs for applications that have a large impact on GPU performance
- Evenly Assignment：Allocate GPUs evenly regardless of the influence of GPU performance





GPU Priority Assignment

Number of Jobs	Priority Assignment (h:min:s)	Evenly Assignment (h:m:s)	Reduce (%)
200	74:30:00	81:11:27	8.241
400	149:00:00	162:22:54	8.241
600	223:30:00	243:34:21	8.241
800	298:00:00	324:45:48	8.241
1000	372:30:00	405:57:15	8.241
10000	3724:42:00	4509:32:33	8.248

- Priority assignment is expected to reduce the total execution time by 8.24%
- from the viewpoint of fairness of allocation among applications, the order of job processing should not necessarily be determined only by the type of benchmark

Conclusions

- Evaluated the characteristics of AI benchmarks “MLPerf” with GPUs
  - Acquired Information of hardware at the time of benchmark execution for feature analysis for each benchmark
    - Compared performance on machines of different generations such as servers and GPUs
  - Results
    - differences in servers and CPUs are considered to have little impact on AI application performance
    - there is a big difference in job performance caused by the difference of GPU performance for each benchmark
- Priority Assignment
  - Estimated and compared the execution time assuming a Docker environment where old GPUs and new GPUs coexist
    - The case a new GPU is preferentially assigned to a benchmark which the improvement in job execution time due to GPU performance was significant
    - The case GPU is evenly assigned
  - Estimation showed Priority Assignment would lead to efficient operation of limited GPU resources

Future Work

- Utilizing AI workload feature analysis for operation control of next-generation computers
  - Construct a system that actually controls the operation of the GPU