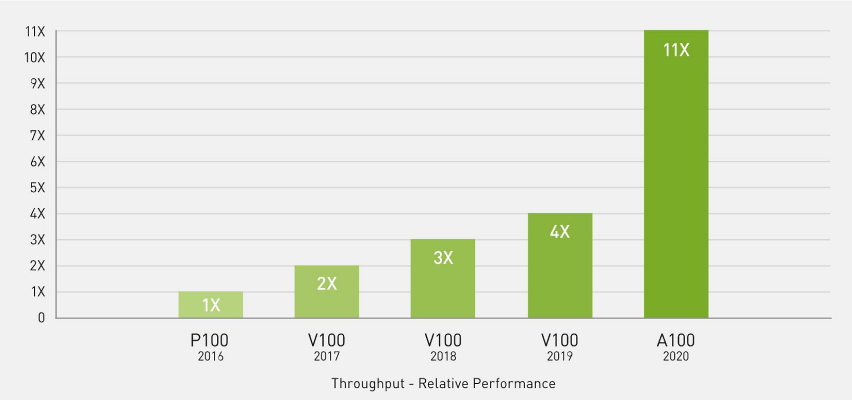
Samsung Research

FastFlow: Accelerating Deep Learning Model Training with Smart Offloading of Input Data Pipeline

Taegeon Um, *Byungsoo Oh, Byeongchan Seo,*
Minhyeok Kweun, Goeun Kim, Woo-Yeon Lee

Samsung Research, Data Intelligence Team

GPUs are the Most Important Resources!



https://www.nvidia.com/en-us/data-center/a100/

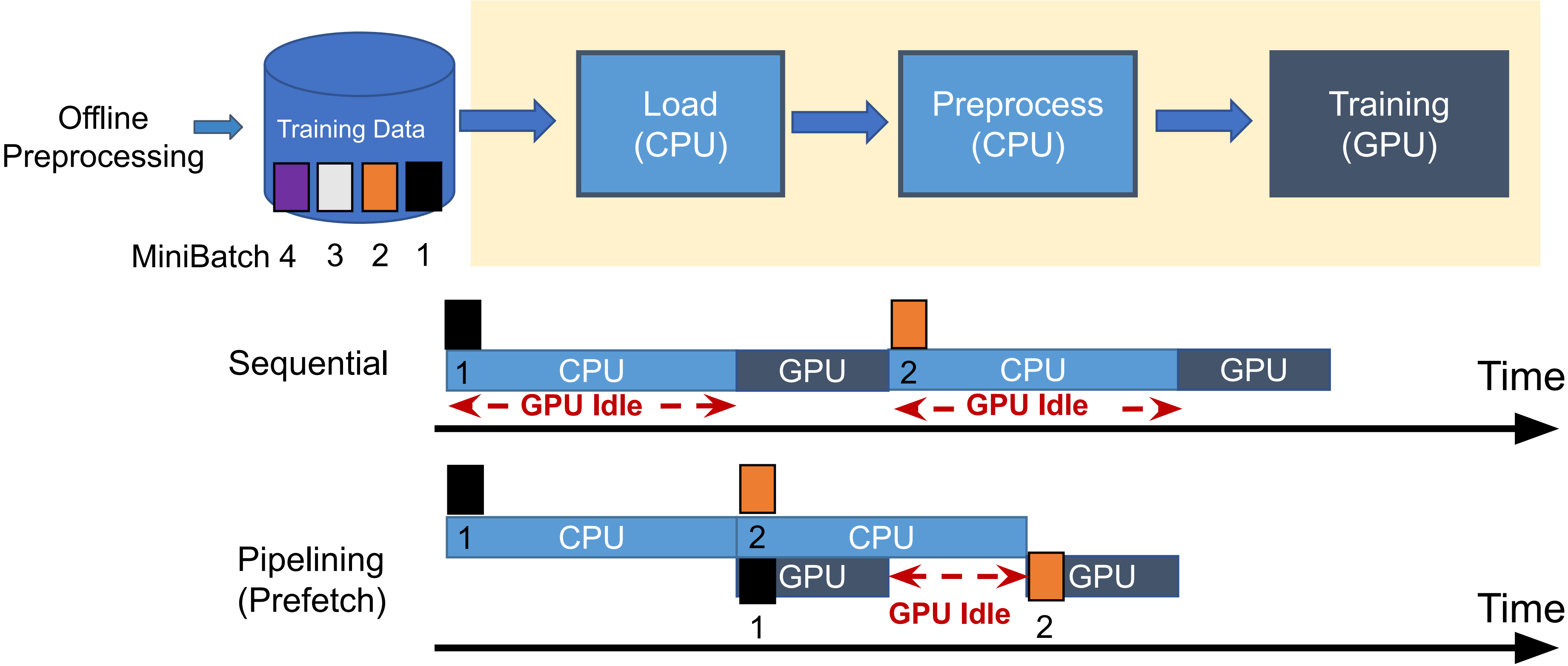
How to Reduce DL Training Time?

- **High-End, Expensive GPUs**
  - V100 << A100 << H100
- **More GPUs**
  - Training on 1 GPU << Training on 10 GPUs

Does this always guarantee training speed up?

Problem: CPU Bottlenecks and Prep Stalls

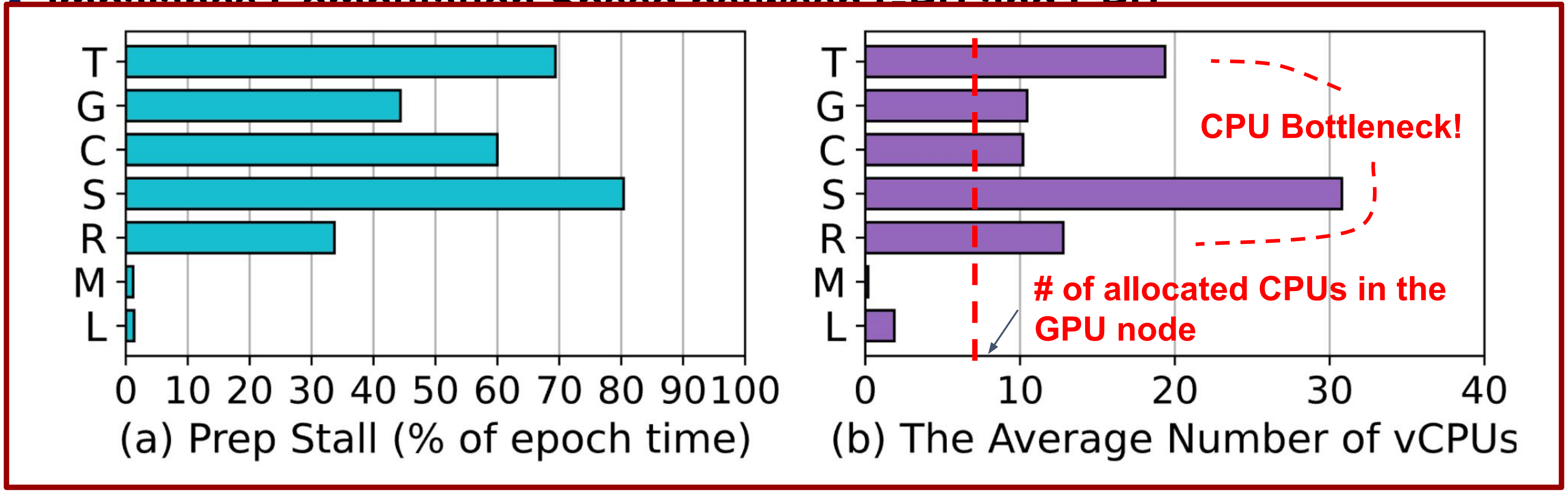
Samsung Research



Preprocessing Stalls ⇒ GPU Util. ↓ Training Speed ↓

Various Reasons of Prep Stalls

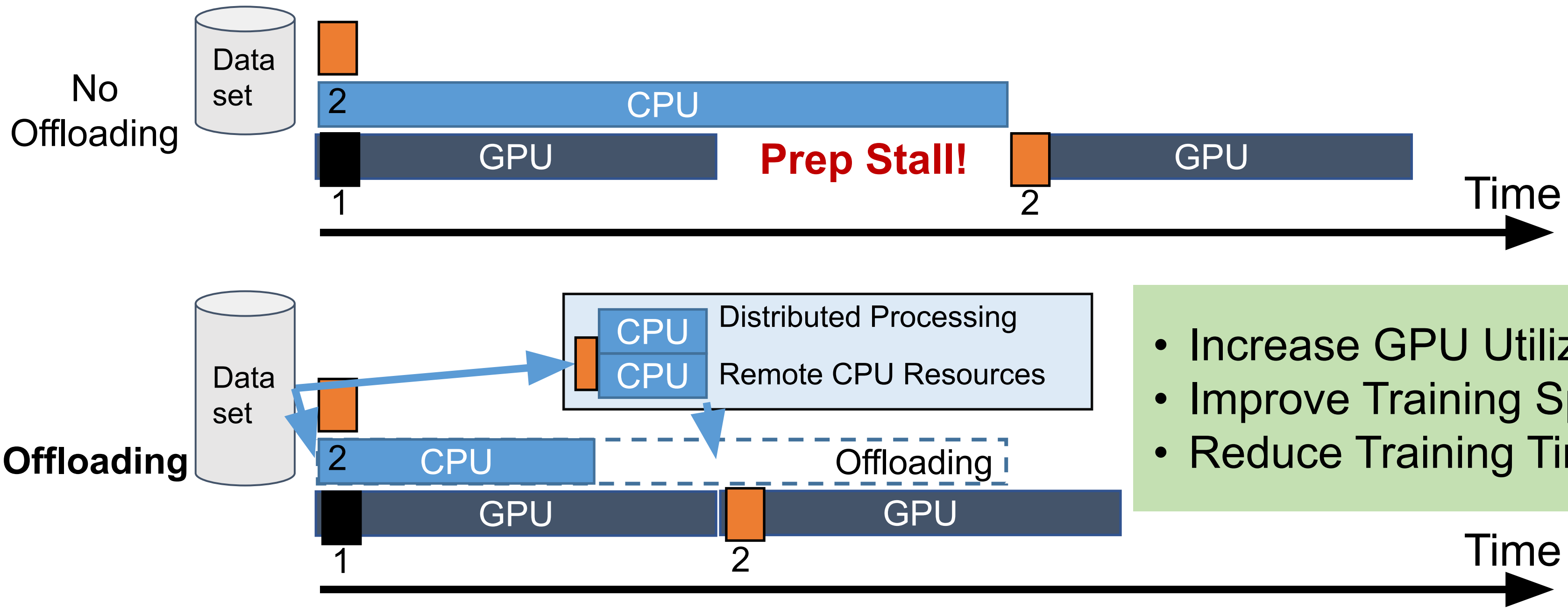
- **Imbalance Computation Speed between GPU and CPU**



- **Fixed CPU:GPU Ratio, but Various Workloads Require Diverse # of CPUs**
  - 8 vCPU cores : 1 V100 GPU (e.g., AWS P3.2xlarge)

Our Approach

- **Automatically Offloading Preprocessing to (Remote) CPU Resources**
  - Automatic Decisions of **<u>When</u>** to Offload, **<u>Which</u>** Operator to Offload, **<u>How</u>** Much Data to Offload



- Increase GPU Utilization
- Improve Training Speed
- Reduce Training Time

Our Approach

- **Automatically Offloading Preprocessing to (Remote) CPU Resources**
  - Automatic Decisions of **When** to Offload, **Which** Operator to Offload, **How** Much Data to Offload

Existing Work

- tf.data.service: Manual offloading by users
- Cachew(ATC '22): Automatically decides the number of (CPU) workers and offloads all data and preprocessing operations to remote CPUs

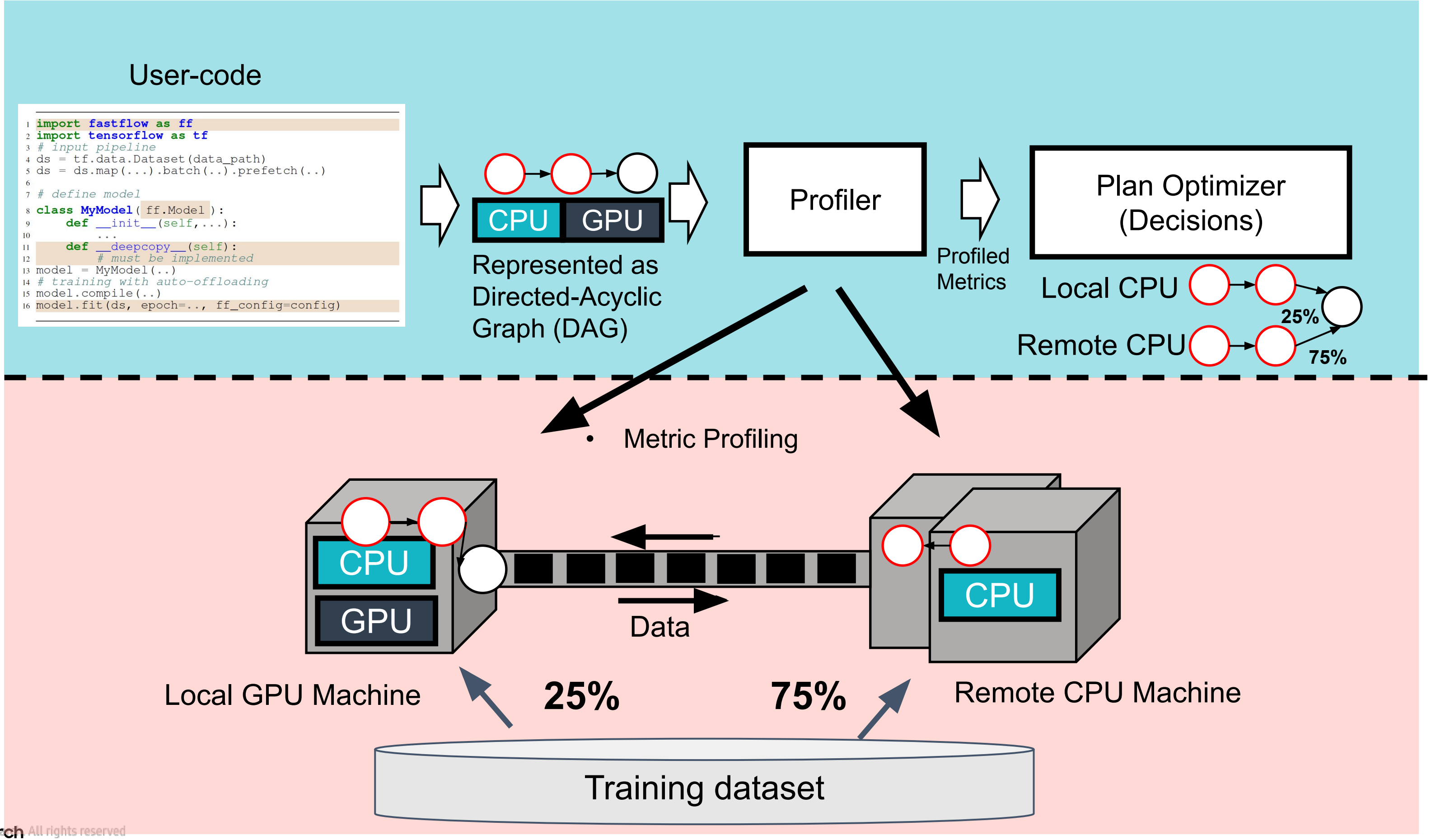
Our Work

- Automatically decides **when** to offload, **which** operators to offload, **how much data** to offload by considering diverse remote CPU environments (e.g., limited network bandwidth and CPU computations)

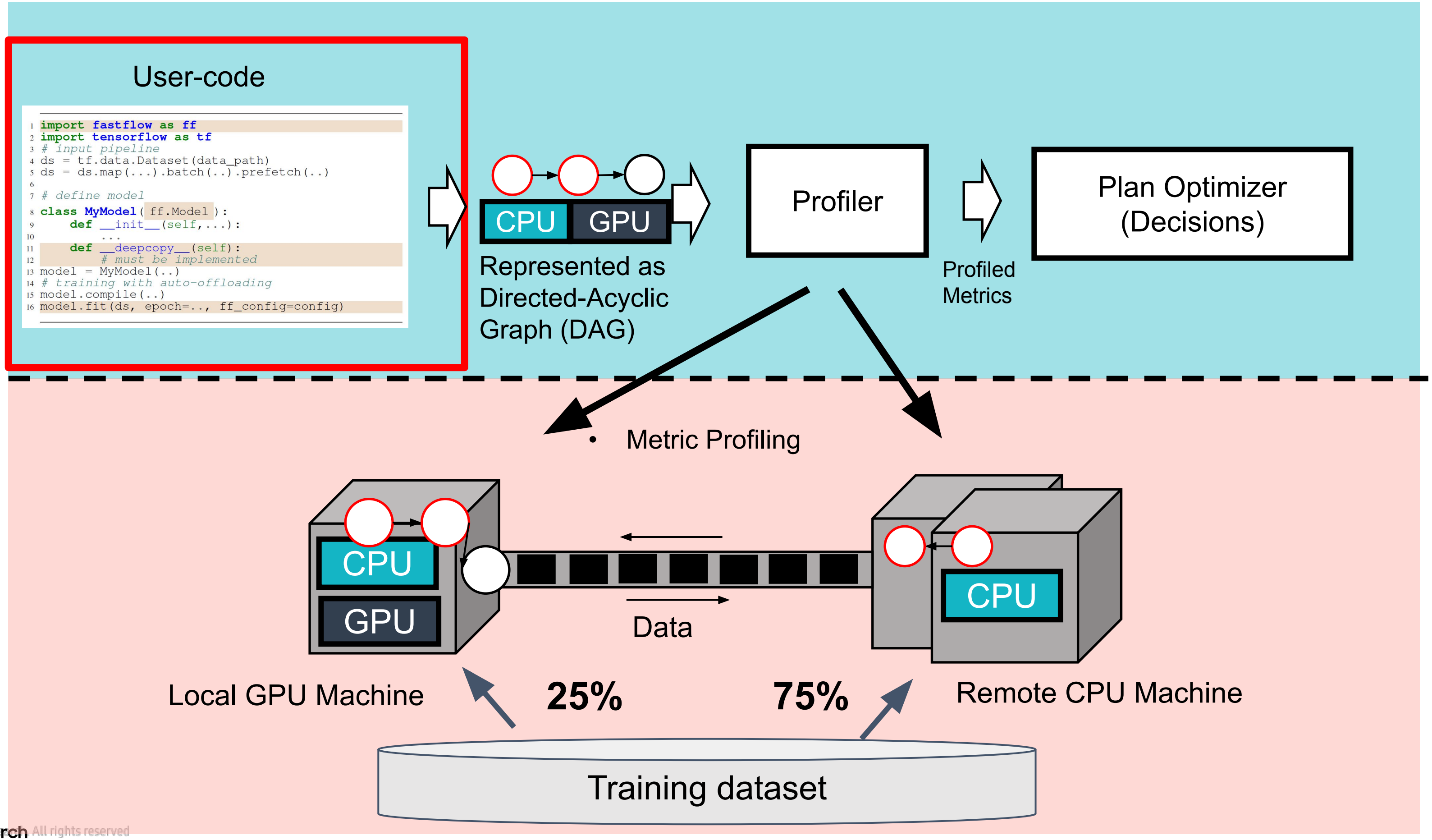
Outline

- Problem & Motivation
- Limitations of Existing Work and Our Approach
- **FastFlow Design**
- Demo
- Evaluation
- Conclusion

FastFlow: A DLT System with Smart Offloading



FastFlow: A DLT System with Smart Offloading



FastFlow: User-API

- Goal: Minimize the Modification of Existing TensorFlow Code for Users to Easily Adopt FastFlow

TensorFlow Code

```
import tensorflow as tf
# Input pipeline
ds = tf.data.Dataset(data_path)
# Manual configuration for offloading
ds = ds.map(...)
       .batch(..).prefetch(..)
# define model
class MyModel(tf.keras.Model):
    def __init__(self,...):
        ...
model = MyModel(..)
# training
model.compile(..)
model.fit(ds, epoch=..,)
```

FastFlow Code

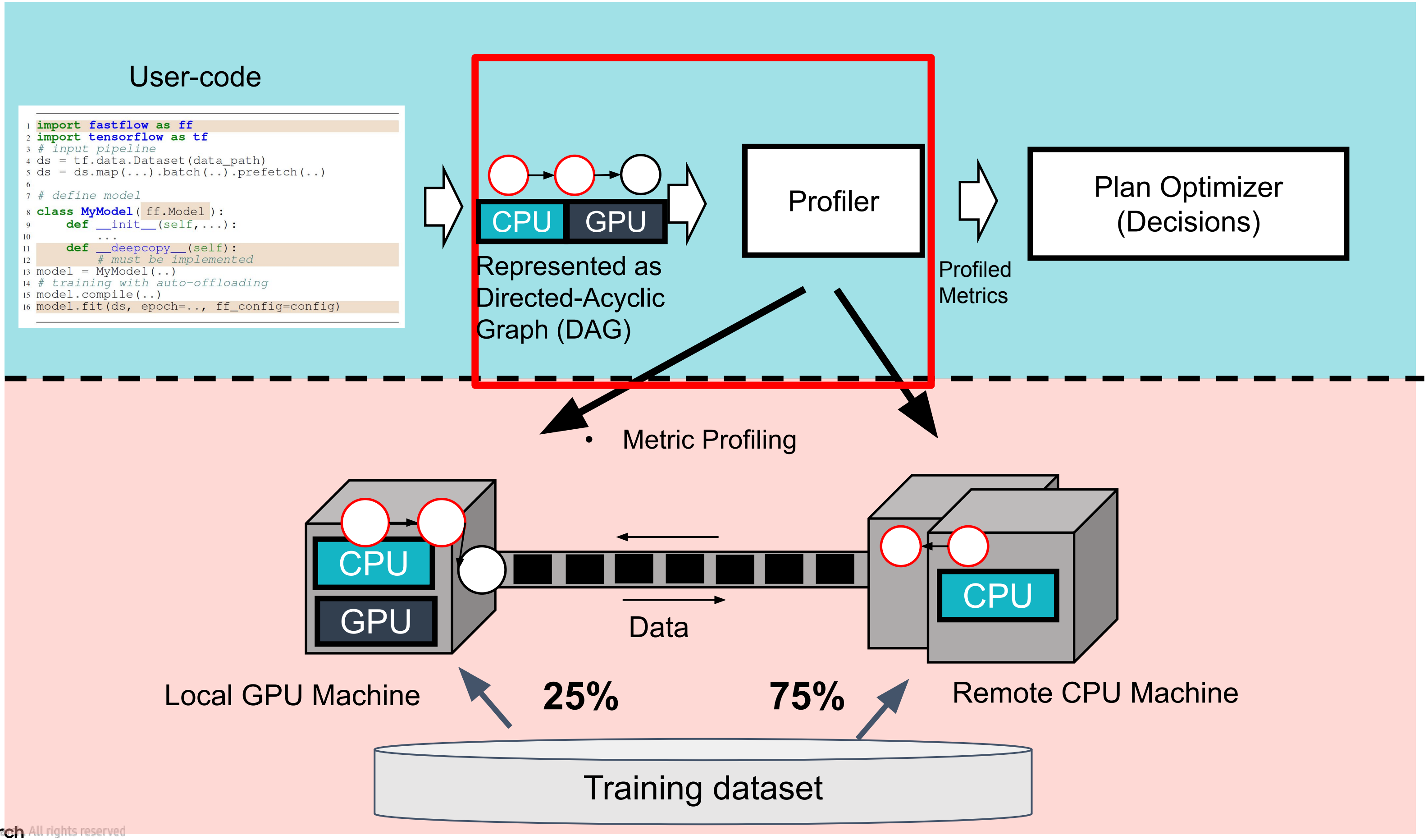
```
import fastflow as ff
import tensorflow as tf
# input pipeline
ds = tf.data.Dataset(data_path)
ds = ds.map(...).batch(..).prefetch(..)

# define model
class MyModel( ff.Model ):
    def __init__(self,...):
        ...


model = MyModel(..)
# training with auto-offloading
model.compile(..)
model.fit(ds, epoch=.., ff_config=config)
```

- No Modification of Main TensorFlow Logic (Model Generation, Preprocessing)
- Only Need to Set Some Configuration Parameters for Smart Offloading

FastFlow: A DLT System with Smart Offloading



FastFlow: Profiling

- Goal: Lightweight Profiling Overheads For Optimal Offloading Decisions
  - Characteristics of ML/DL Training; Iterative per Minibatch
  - Profiling A Few Steps (e.g., ~100)
    - Negligible Profiling Overheads (time), but High Accuracy of Profiled Metrics
  - Profile Metrics Related to Applications and Resource Environments
    - Ex) Training throughput with local or remote CPUs (Lthp/Rthp), GPU throughput (Gthp), Offloading overhead (PCycle, OCycle)

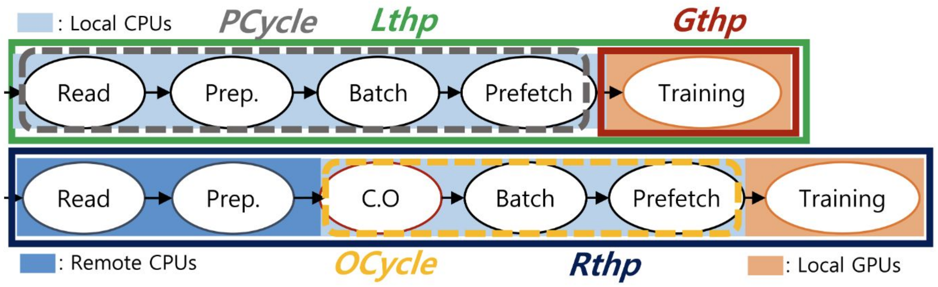
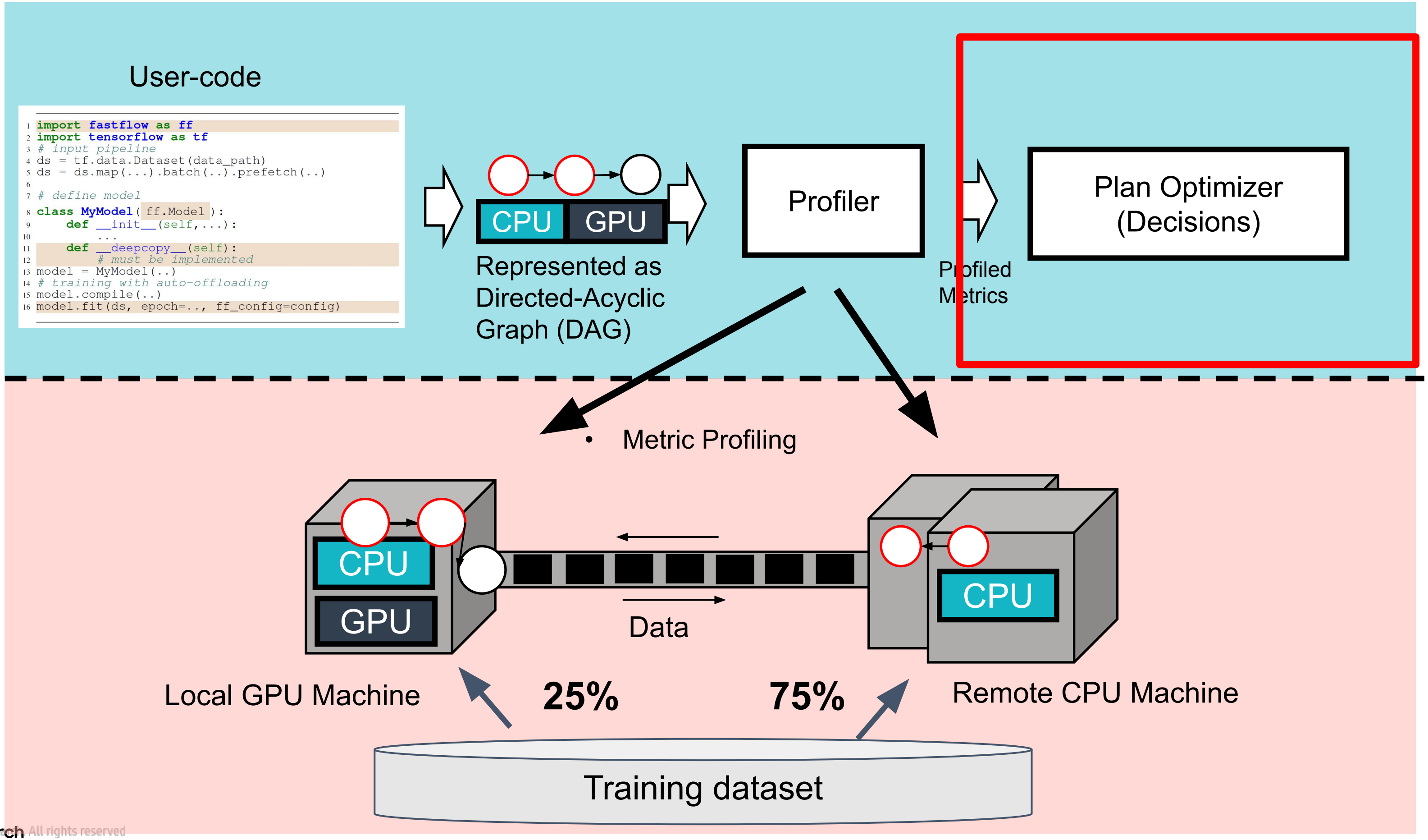


Figure 4: An illustration of measuring metrics for offloading decisions.

FastFlow: A DLT System with Smart Offloading

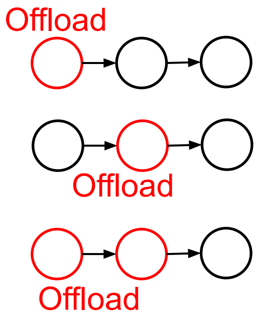


FastFlow: Offloading Decisions

- Goal: Optimal <u>Automatic</u> Decisions for Offloading

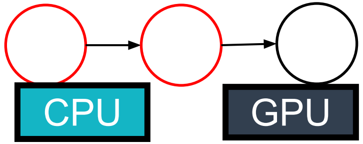
1. <u>When to Offload? (Do We Have to Offload?)</u>
   - If Preprocessing Speed on CPUs < Computation Speed on GPUs ⇒ Try to Offload! (CPU is the Bottleneck)
     Otherwise, Do Not Offload
2. <u>Which Operations to Offload?</u>
   - Compare several candidate pipelines for offloading
     and choose the one that leads to maximum training throughput

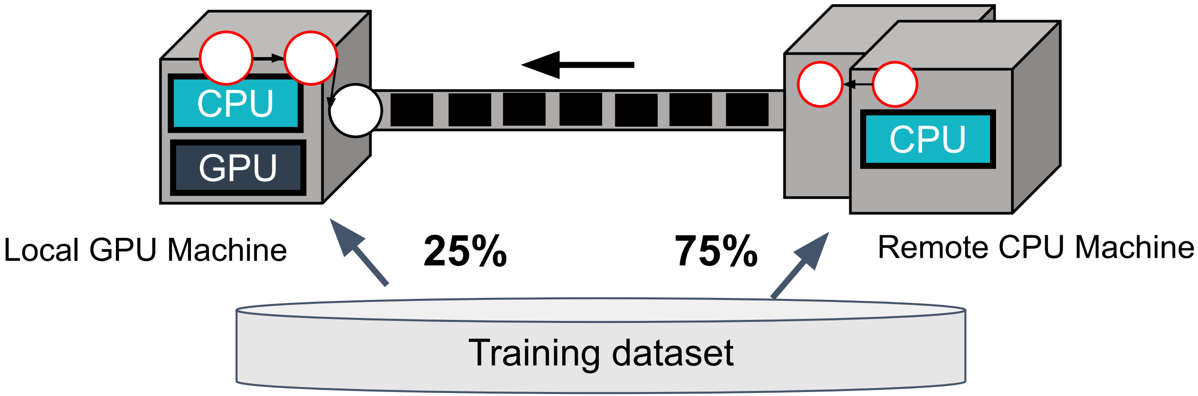
Best!



FastFlow: Offloading Decisions

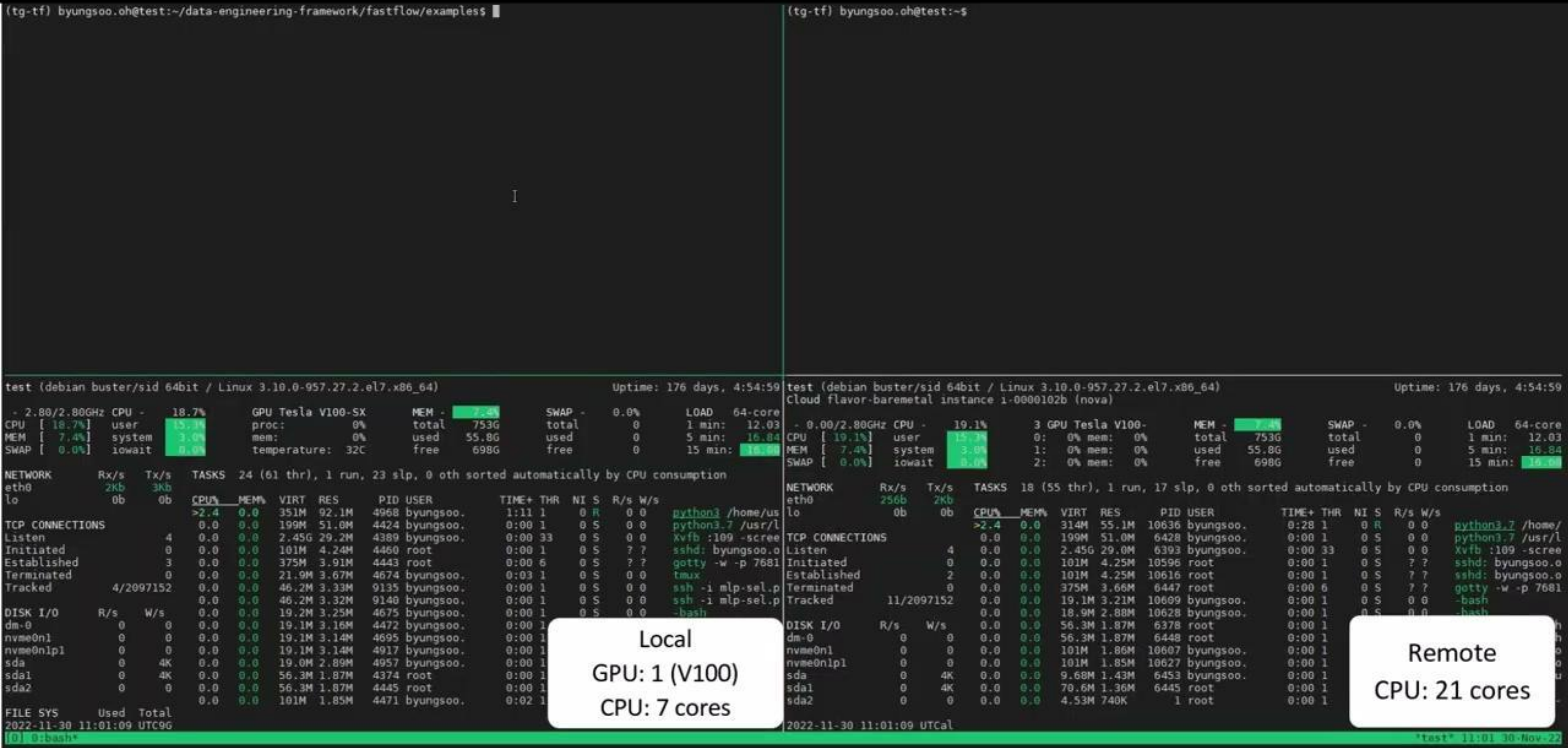
- Goal: Optimal Automatic Decisions for Offloading
  3. How Much Data to Offload?
     - Considers Computing Capacity and Network Bandwidth of Remote Machines
     - Ex) If Remote CPU Throughput = 3x Local CPU Throughput, then 75% in Remote 25% in Local





Outline

- Problem & Motivation
- Limitations of Existing Work and Our Approach
- FastFlow Design
- Demo
- Evaluation
- Conclusion

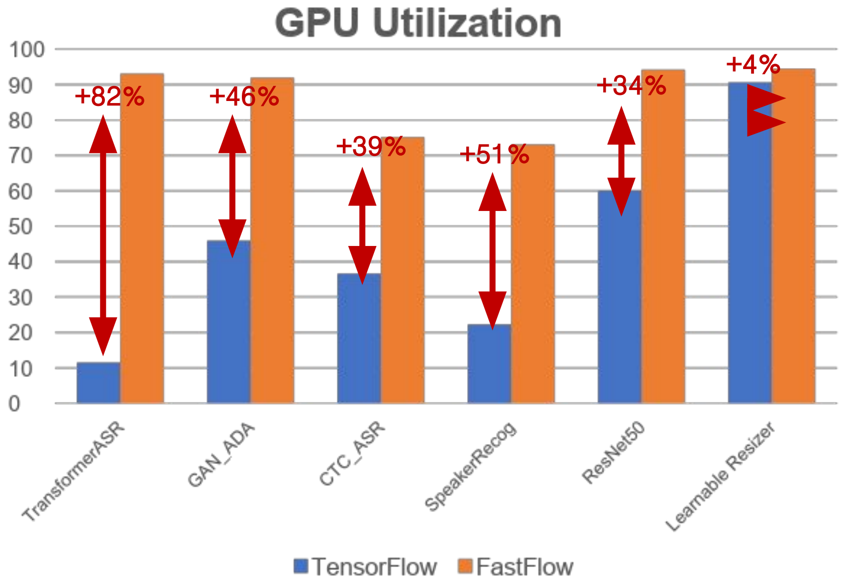

(tg-tf) byungsoo.oh@test:~/data-engineering-framework/fastflow/examples$
(tg-tf) byungsoo.oh@test:~$
test (debian buster/sid 64bit / Linux 3.10.0-957.27.2.el7.x86_64)
Uptime: 176 days, 4:54:59
GPU Tesla V100-SX
Cloud flavor-baremetal instance i-0000102b (nova)
3 GPU Tesla V100-
NETWORK
TASKS 24 (61 thr), 1 run, 23 slp, 0 oth sorted automatically by CPU consumption
TASKS 18 (55 thr), 1 run, 17 slp, 0 oth sorted automatically by CPU consumption
TCP CONNECTIONS
DISK I/O
FILE SYS
2022-11-30 11:01:09 UTC
Local
GPU: 1 (V100)
CPU: 7 cores
Remote
CPU: 21 cores

Outline

- Problem & Motivation
- Limitations of Existing Work and Our Approach
- FastFlow Design
- Demo
- **Evaluation**
- Conclusion

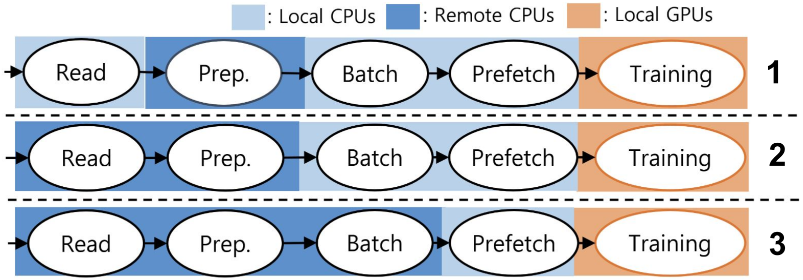
FastFlow: Evaluation

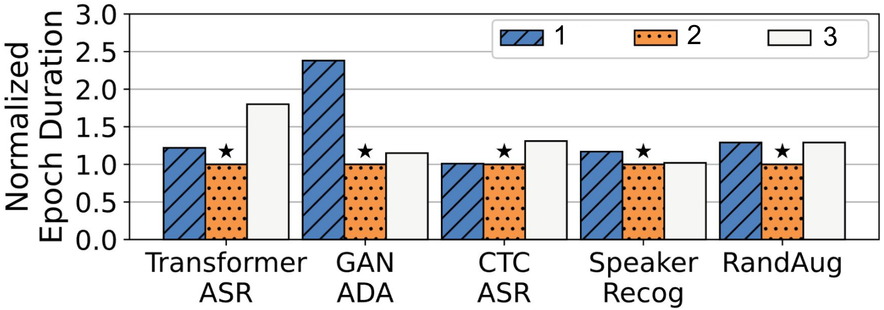
- Q) How Much Does FastFlow Improve Performance And GPU Util. Compared to TensorFlow w/o Offloading? (21 remote CPUs)



- FastFlow Accelerates Training Up to 3.8x Compared to TensorFlow
- FastFlow Increases GPU Utilization from 4%~82% Points Compared to TensorFlow

- Q) Does FastFlow Make Optimal Decisions for Offloading?
  - **<u>Decision of Which Operations to Offload</u>**





<Evaluation on remote CPUs with high disk and network bandwidth>

Figure 8: Impact of the offloading operator selection of FastFlow. * mark shows the selection of FastFlow.

- FastFlow chooses **2** when remote nodes have high disk and network bandwidth
- If remote CPU node has low disk I/O, FastFlow chooses **1** to prevent disk bottleneck.

- Q) Does FastFlow Make Optimal Decisions for Offloading?
  - **Decision of How Much Data to Offload**

x (# of local CPUs) :y (# of remote CPUs)

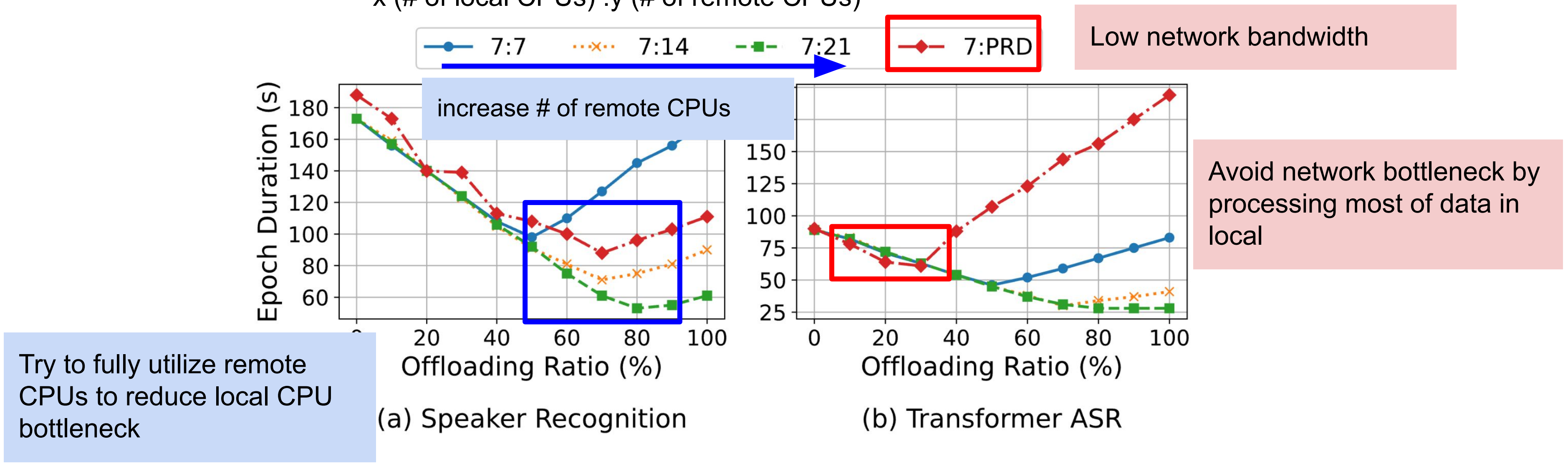


Figure 9: Epoch duration in various offloading ratios on different remote workers. (a) is compute-intensive, and (b) is data-intensive preprocessing workload.

FastFlow: Evaluation

Samsung Research

- Q) How Long Does the Profiling Take Time ?

Job	Profile time (min)	Train time (min)	Fraction (%)
T	1.91 / 0.08*	68.37	**2.80 / 0.11***
G	0.61 / 0.08*	162.22	**0.37 / 0.05***
C	9.27 / 0.20*	299.32	**3.10 / 0.07***
S	1.28 / 0.02*	150.58	**0.85 / 0.01***
R	7.01 / 0.24*	11396.50	**0.06 / 0.002***
M	0.78 / 0.53*	933.35	**0.08 / 0.06***
L	0.65 / 0.26*	49.23	**1.31 / 0.53***

- Profiling Overhead is Negligible (Only ~3% of Total Training Time)

: Profiling Without Metric Store/Reload

: Profiling With Metric Store/Reload.
FastFlow Further Reduces Profiling Time By Reusing Metrics Measured Before

Outline

- Problem & Motivation
- Limitations of Existing Work and Our Approach
- FastFlow Design
- Demo
- Evaluation
- **Conclusion**

Conclusion

- Although GPUs are the most important and expensive resource for DL training, it is under-utilized because of CPU bottlenecks
- To address the CPU bottlenecks, we designed and implemented FastFlow. We evaluated that FastFlow can significantly improve training speed with automatic decisions of offloading
- FastFlow can be applied to diverse applications on various resource environments without the modification of main training logic

Samsung Research

Thank you