LinnOS : Predictability on Unpredictable Flash Storage with a Light Neural Network

Mingzhe Hao, Levent Toksoz, Nanqinqin Li, Edward Edberg Halim,

In 2020 USENIX Symposium on Operating Systems Design and Implementation

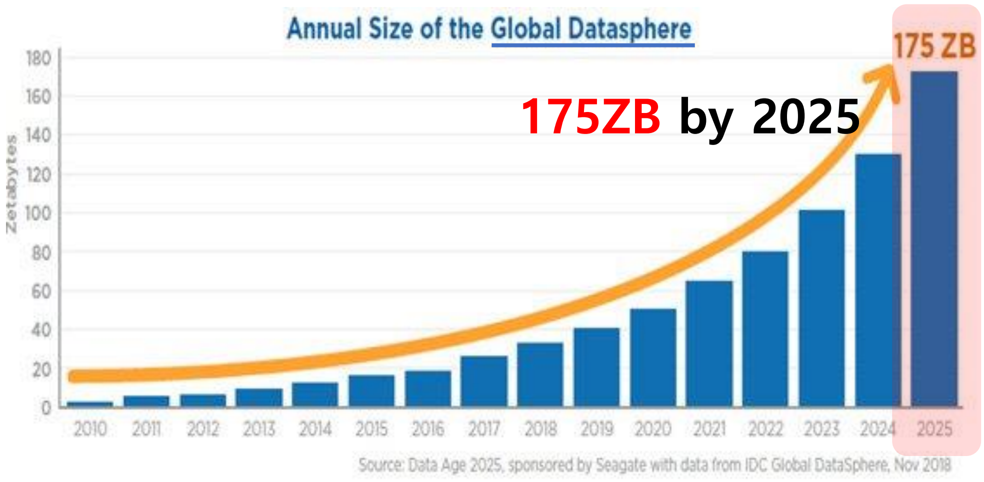
2021. 01. 26

Presentation by Han, Yejin

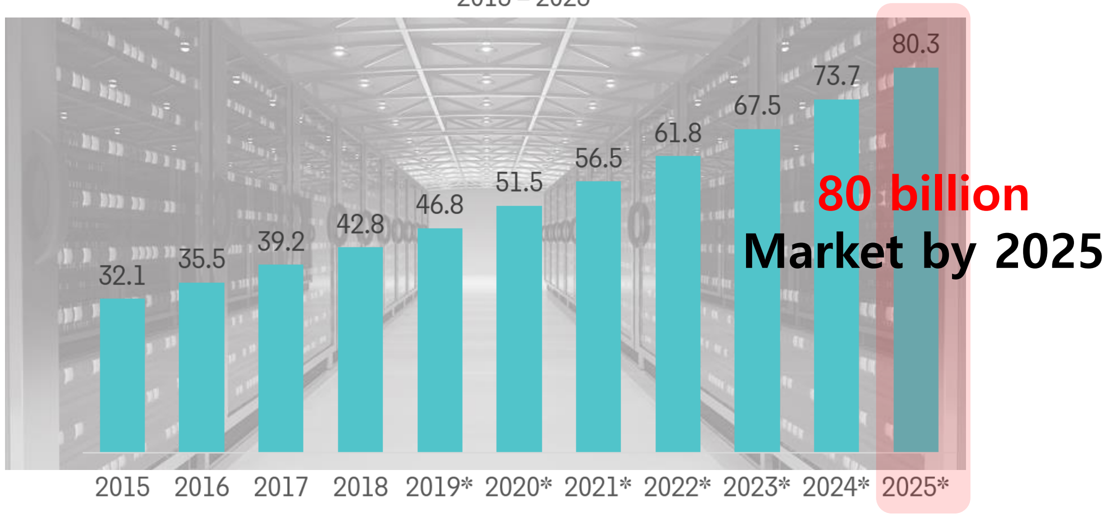
hyj0225@dankook.ac.kr

Contents

1. Introduction
2. Background
3. Motivation
4. LinnOS
5. Evaluation
6. Conclusion

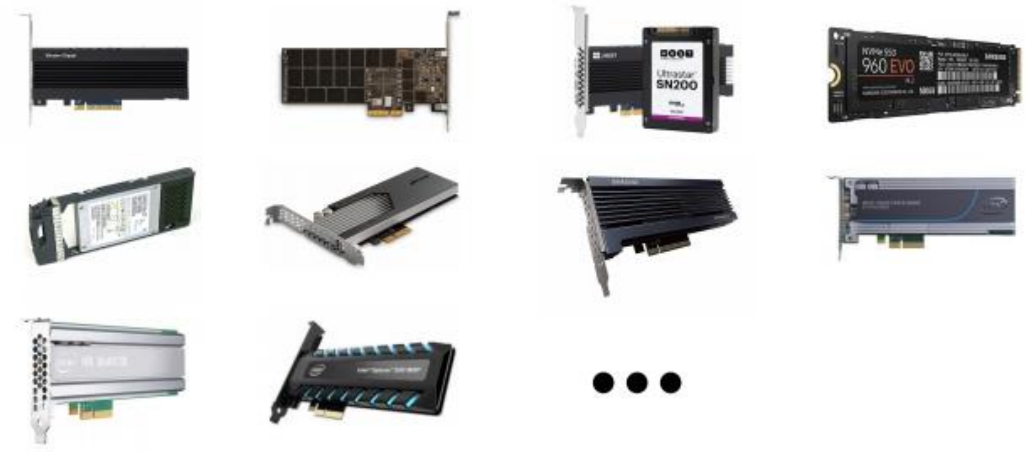


Size of 2D & 3D NAND Flash Market, in USD billion, Global, 2015 – 2025



*Forecast

Source: Semiconductor Equipment and Materials international (SEMI)

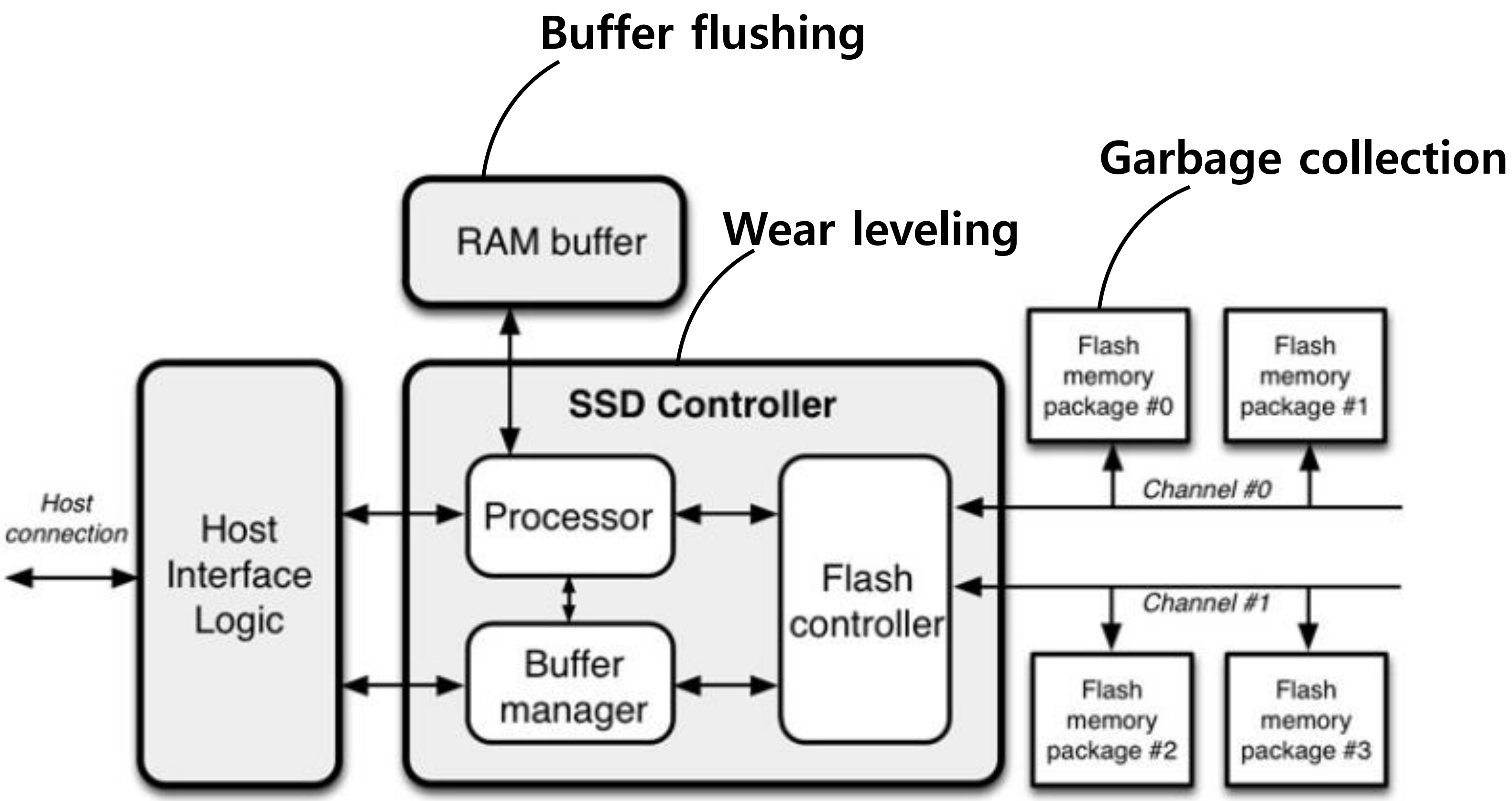


- Faster and faster SSDs are available
- SSD internal complexity grows

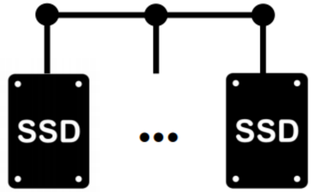
Unpredictable latency

Unpredictable latency

- SSD internal complexities threat to performance stability



Unpredictable latency at individual-I/O level



1st I/O	100µs	120µs
2nd I/O	80µs	150µs
3rd I/O	110µs	800µs
4th I/O	1.5ms	200µs
⋮	⋮	⋮

Conventional Approaches for predictability

White/gray-box

- Re-architect device internals/interface

KAML: A Flexible, High-Performance Key-Value SSD

AutoSSD: an Autonomic SSD Architecture

Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs

...

Powerful

Need to modify hardware

Black-box

- SSD-aware filesystems and applications

F2FS: A New File System for Flash Storage

Strata: A Cross Media File System

NOVA-Fortis: A Fault-Tolerant Non-Volatile Main Memory File System

...

Not modifying hardware

Considerable re-design in software stack

Conventional Approaches for predictability

Black-box: fs/storage applications, **speculative execution**

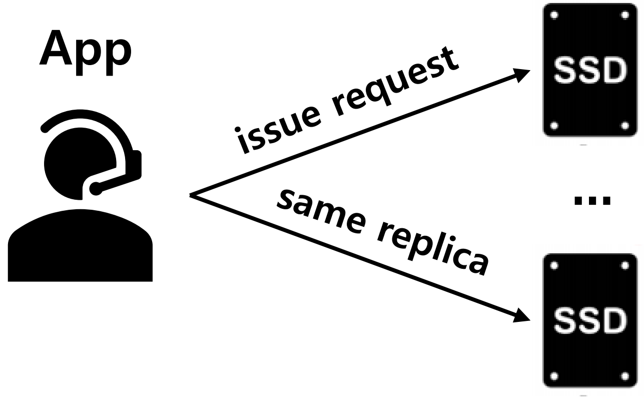
mongoDB cassandra **Hadoop** …

Conventional Approaches for predictability

Black-box: fs/storage applications, **speculative execution**

mongoDB cassandra Hadoop …

Hedged requests

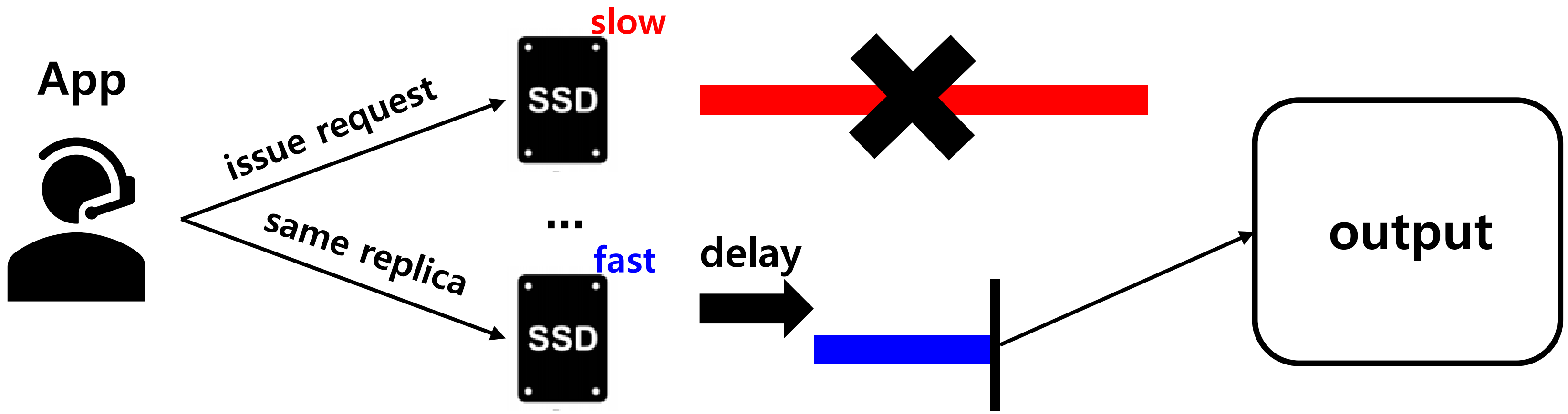


Conventional Approaches for predictability

Black-box: fs/storage applications, **speculative execution**

mongoDB cassandra Hadoop …

Hedged requests

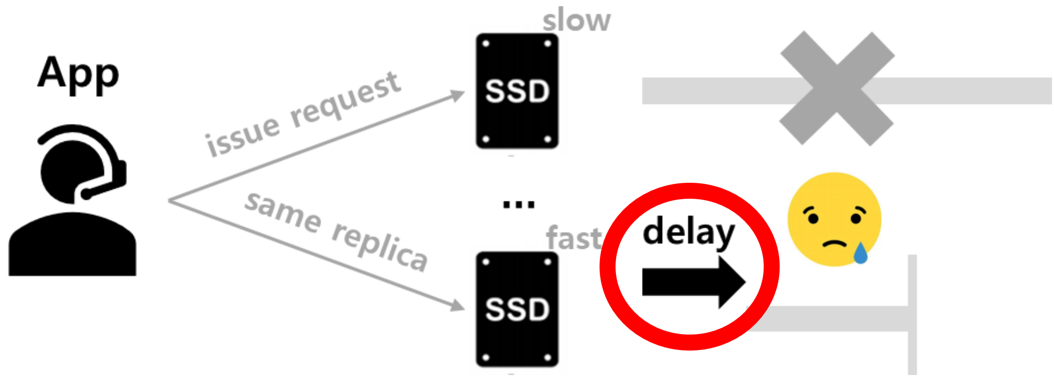


Mitigate every slow I/O!!

Device-Agnostic

speculative execution

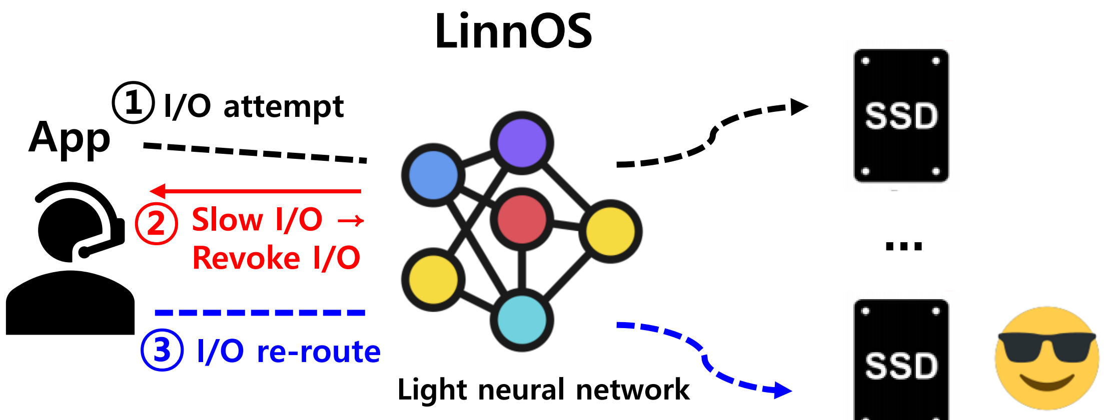
- Passively wait due to black-box



Learn the device behavior

LinnOS

- Proactively infer the black-box
- applications can know in advance whether performance expectations can be fulfilled



Fast & No-wait

LinnOS

- Machine learning for OS to learn black-box devices
- Admission control for clustered storage applications
- Per-I/O SSD performance prediction

Embedded System Lab.

Usage scenario

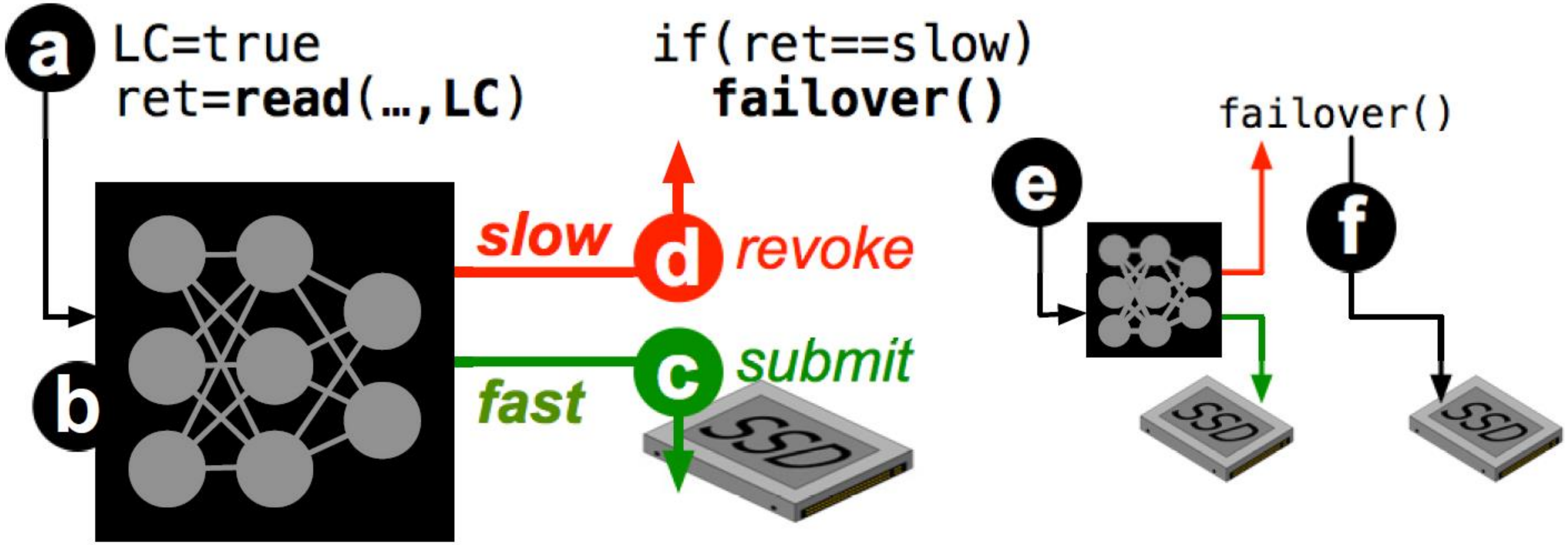


Figure 2: **Usage scenario.** *This usage scenario is explained in Section 3.1. "LC" implies latency critical.*

Overall architecture

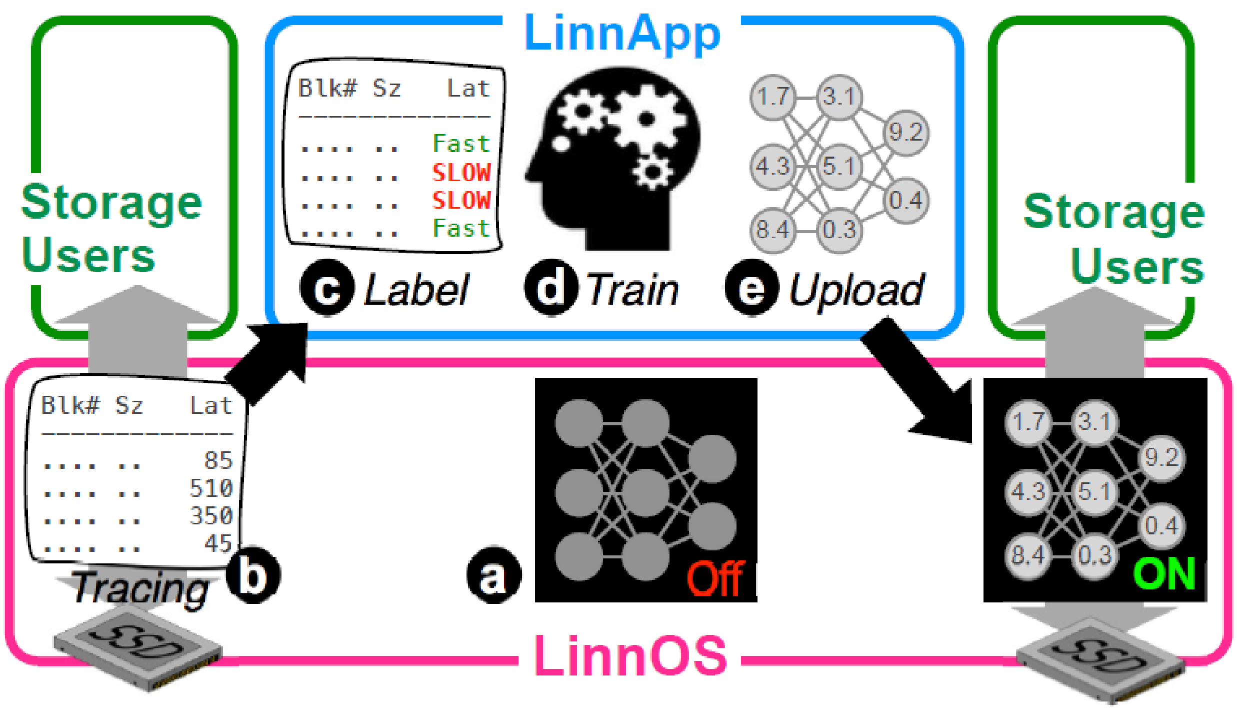


Figure 3: **LinnOS architecture.** *The figure displays LinnOS architecture including LinnApp, as summarized in Section 3.2. The two SSD pictures represent the same SSD instance; the left one depicts tracing/training and the right one live inference on the SSD.*

Design Challenges

Input features

Use what to predict?
(accuracy and inference overhead)

Output labeling

What to predict?
(accuracy and effectiveness)

Linn OS

How to minimize the impact?

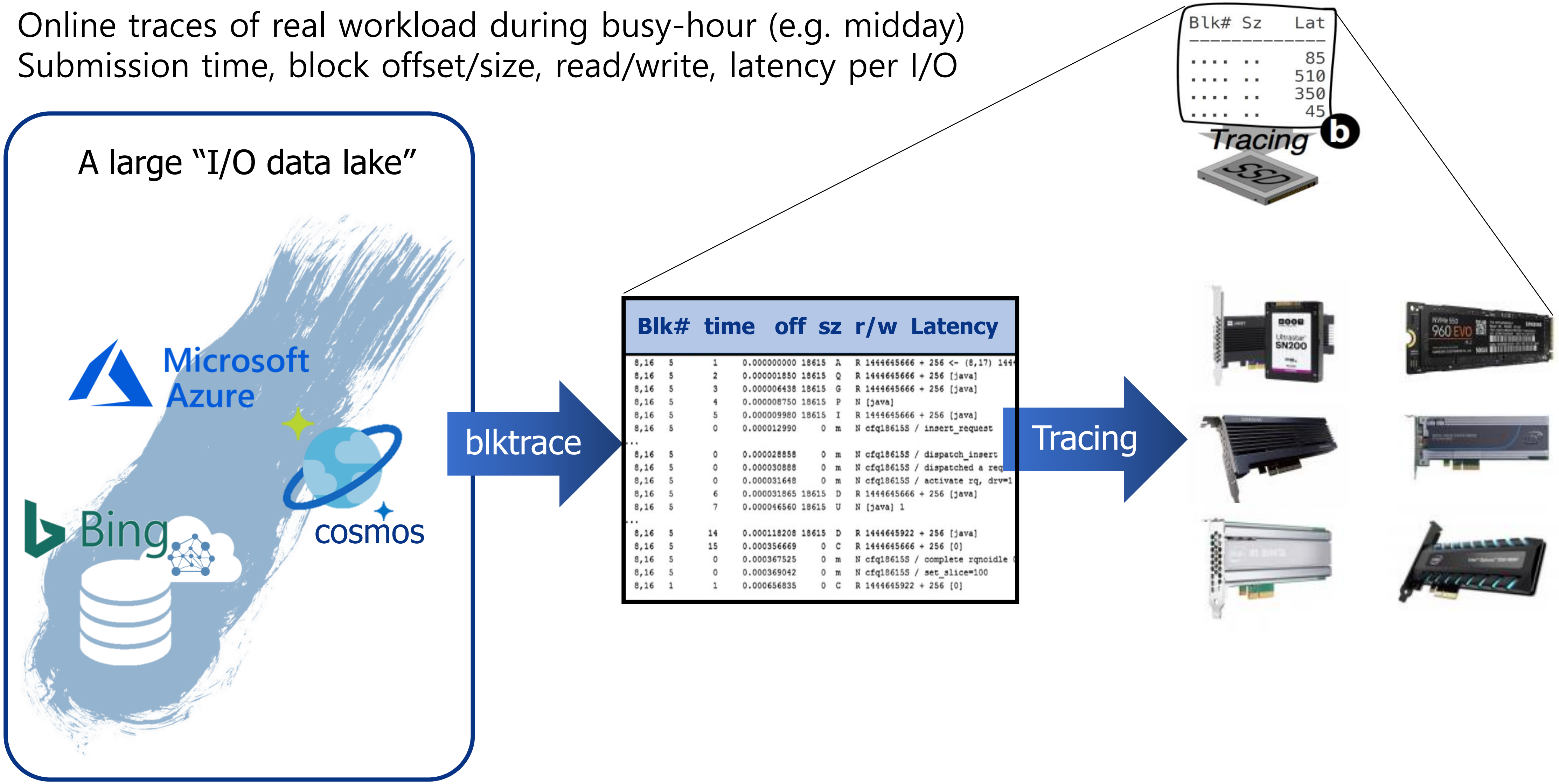
Handling inaccuracy

Design Solutions

1) Training Data Collection
2) Labeling (with Inflection Point)
3) Light Nerual Network Model
4) Improving Accuracy
5) Improving Inference Time

1) Training Data Collection

- Online traces of real workload during busy-hour (e.g. midday)
- Submission time, block offset/size, read/write, latency per I/O



Design Solutions

1) Training Data Collection
2) Output Labeling
3) Light Nerual Network Model
4) Improving Accuracy
5) Improving Inference Time

2) Output Labeling

Ideal Labeling

Exact µs-level latency
(e.g. 120µs, 70µs…)

Linear labeling
(e.g. 0-10µs, 10-20µs…)

exponantial labeling
(e.g. 2-4µs, 4-8µs…)

Flexible

Too many labels

Hard to make accurate and fast

Only 60-70% accuracy

64-128µs

128-256µs ← Mis-inferred

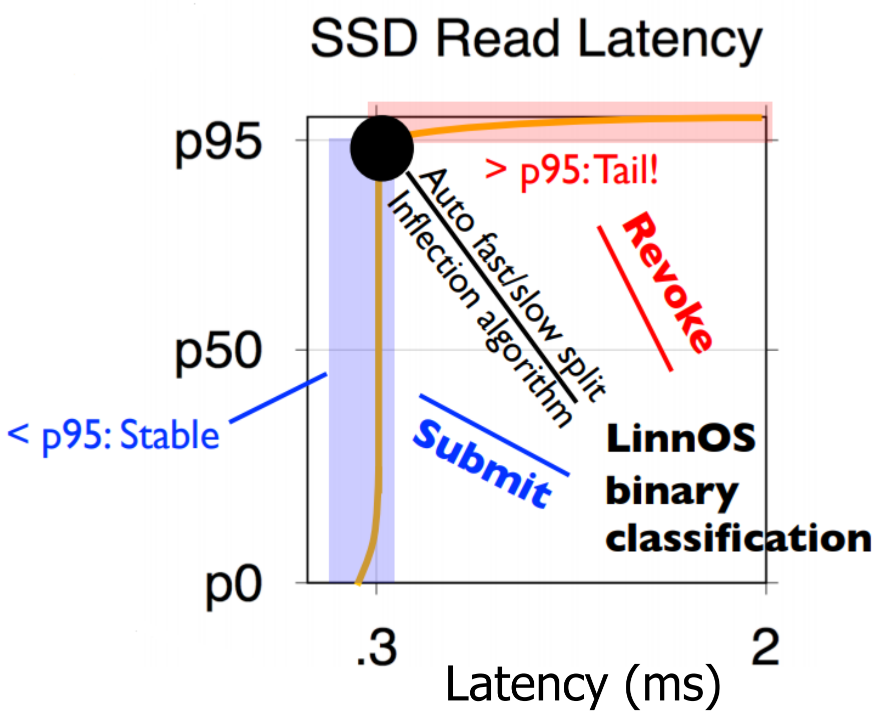
256-512µs ← Truth

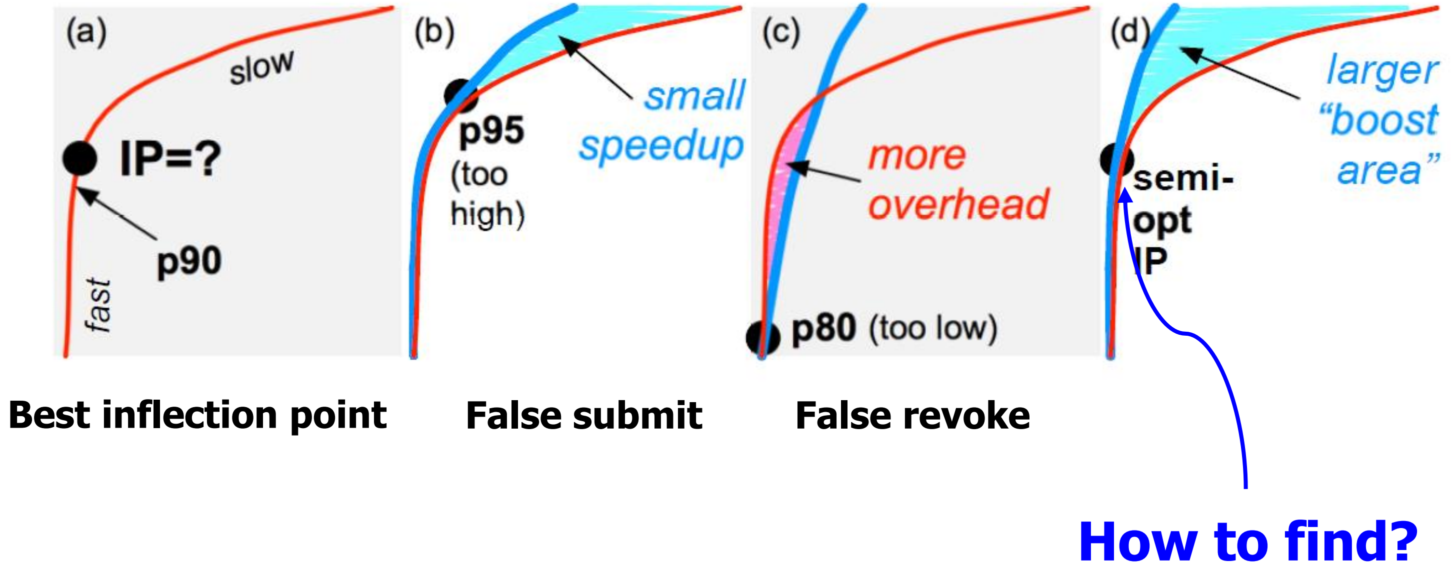
Alternatives?

2) Output Labeling

- Latencies often form a Pareto distribution with a high alpha number
- Users only worry about the tail behavior, not the precise latency

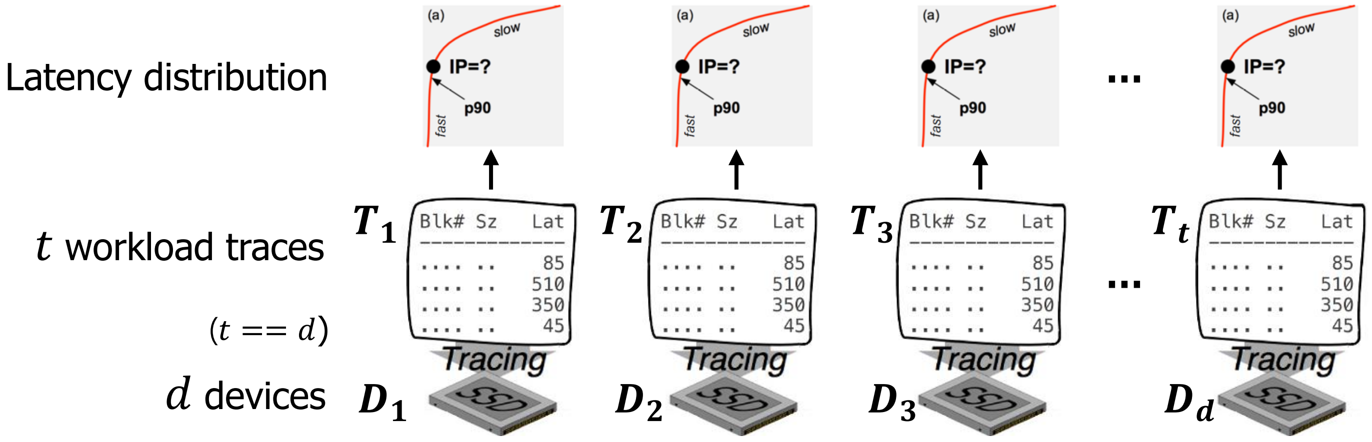
Labeling with **inflection point**





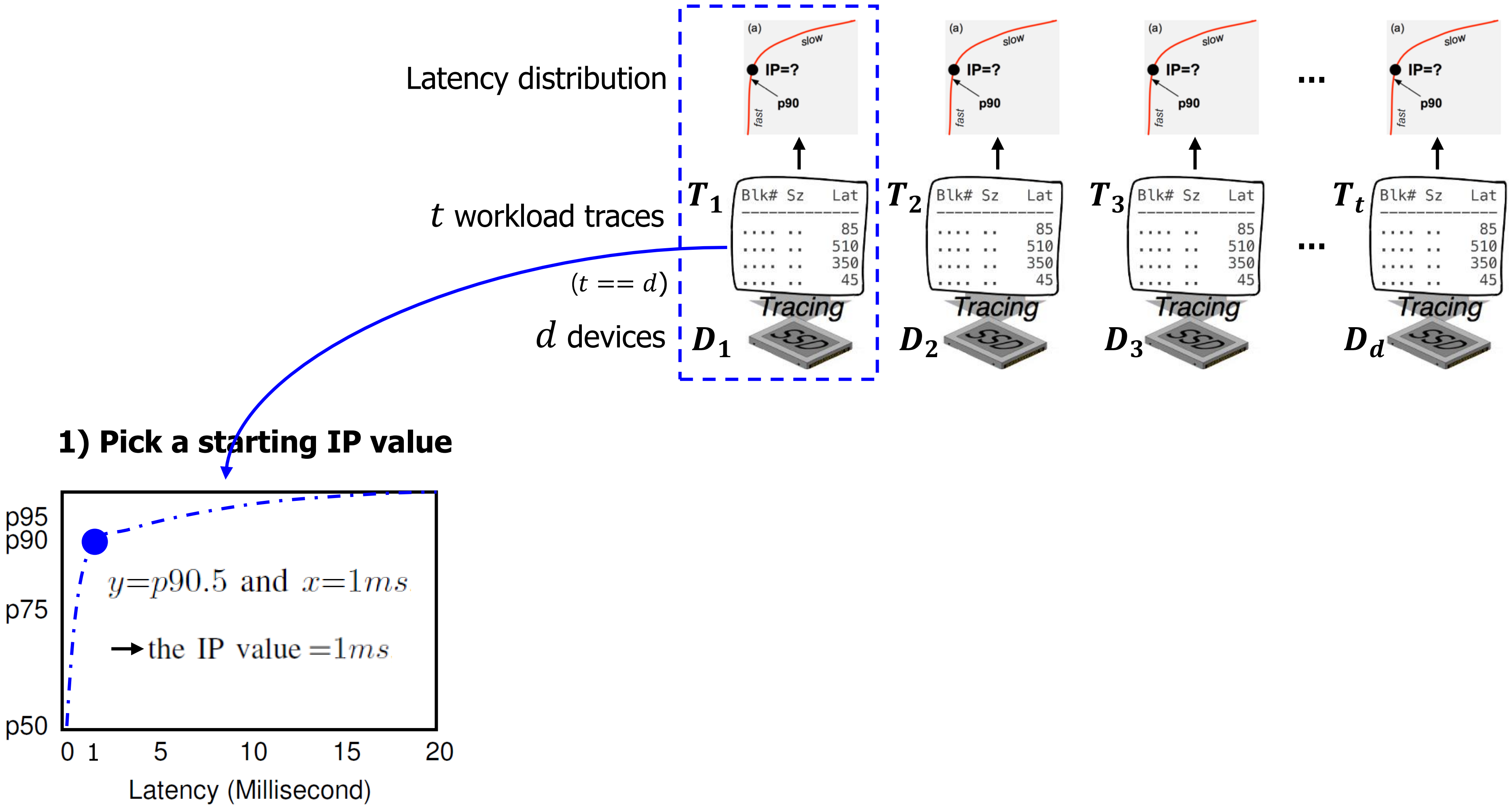
2) Output Labeling

- Inflection Point Algorithm



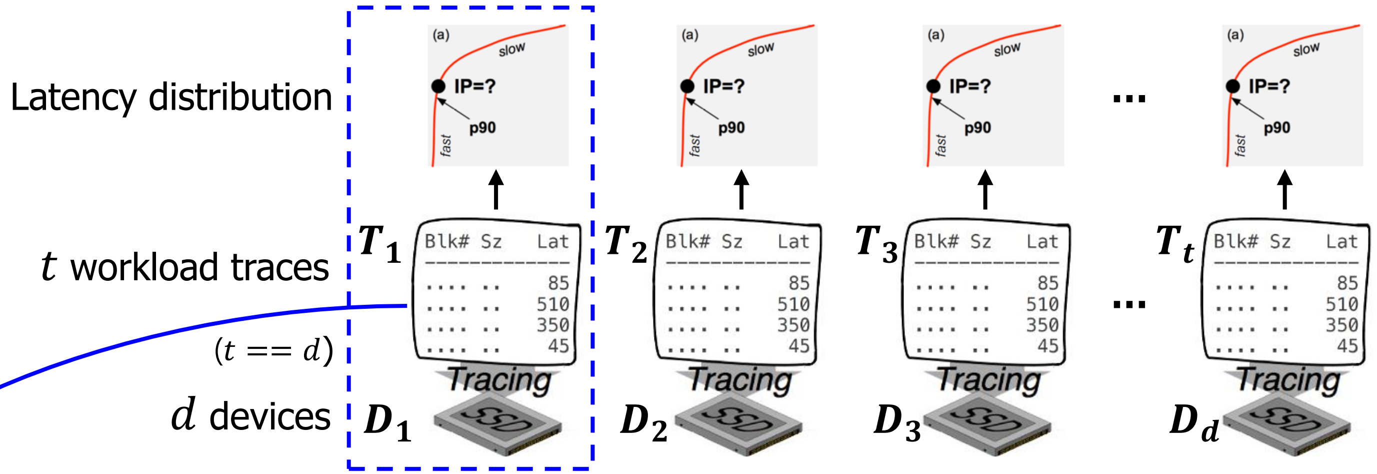
2) Output Labeling

- Inflection Point Algorithm

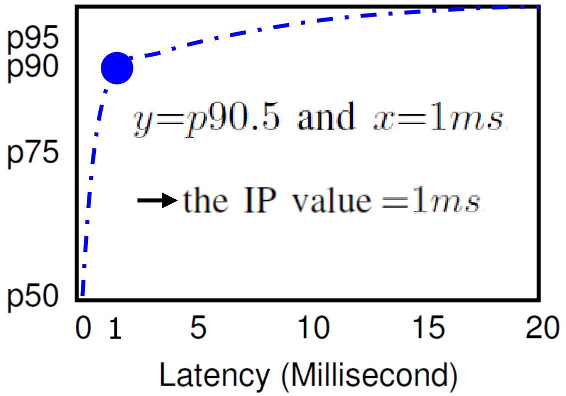


2) Output Labeling

- Inflection Point Algorithm



1) Pick a starting IP value



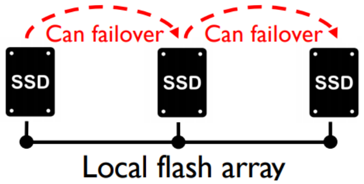
2) Simulate/repeat admission control

I/O request r_i's latency value $\leq 1ms$

→ the r_i's new latency is set to be the same;

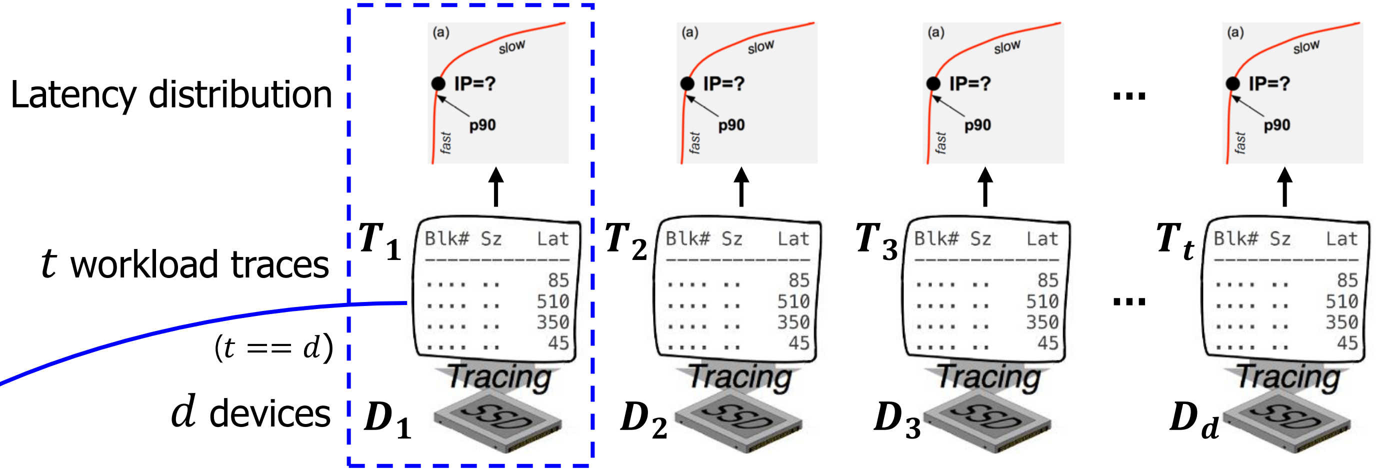
I/O request r_i's latency value $> 1ms$

→ revoked and failed over to randomly selected node selected node (*e.g.*, D_4)

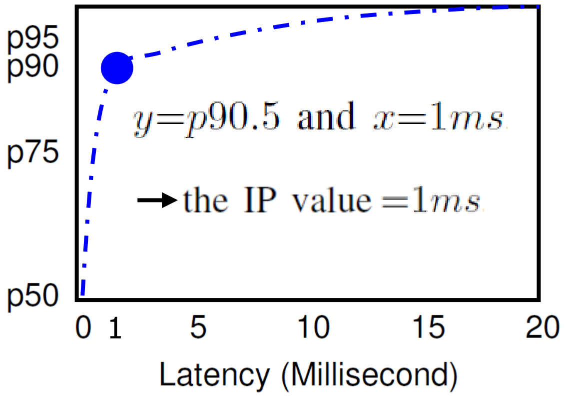


2) Output Labeling

- Inflection Point Algorithm



1) Pick a starting IP value



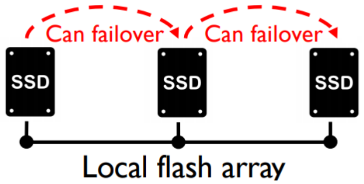
2) Simulate/repeat admission control

I/O request r_i's latency value $\leq 1ms$

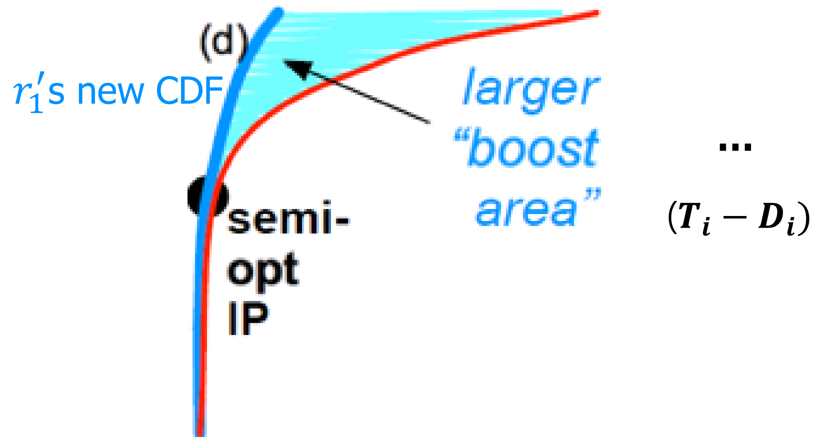
→ the r_i's new latency == $1ms$

I/O request r_i's latency value $> 1ms$

→ revoked and failed over to randomly selected node selected node (e.g., D_4)



3) New CDFs and pick the IP^{max}



Design Solutions

1) Training Data Collection
2) Labeling (with Inflection Point)
3) Input features
4) Improving Accuracy
5) Improving Inference Time

3) Input features

- The number of pending I/Os (4KB pages) when an incoming I/O arrives
- The latency of the 4 most-recently completed I/Os
- The number of pending I/Os when each of the 4 completed I/Os arrived

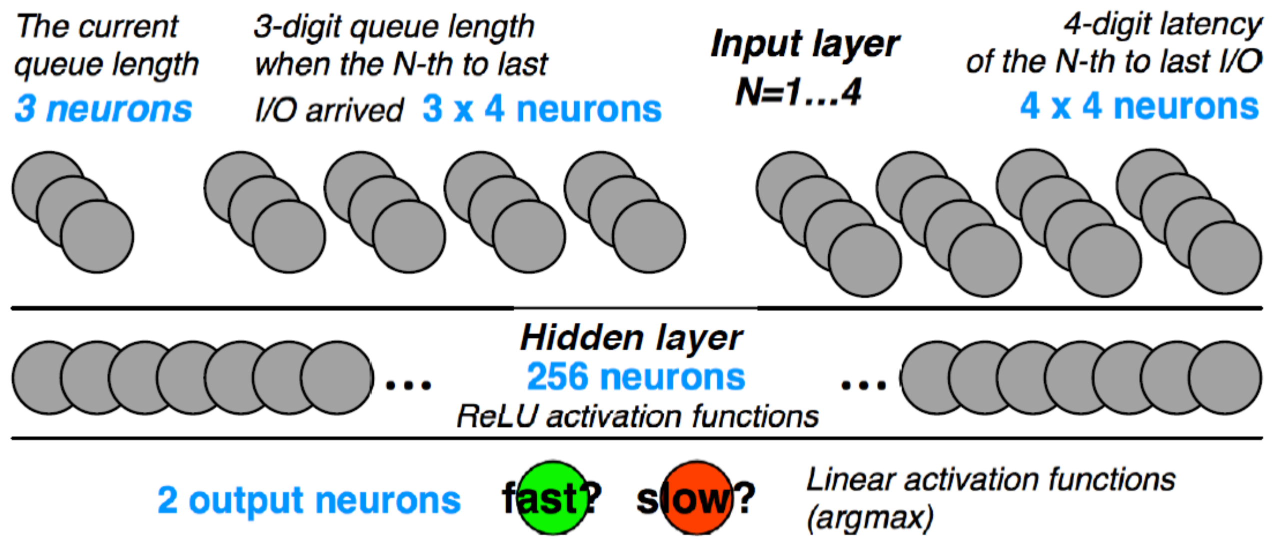


Figure 6: **Light neural network.** *The figure depicts LinnOS 3-layer neural network explained in Section 4.3.*

3) Input features

- The number of pending I/Os (4KB pages) when an incoming I/O arrives
- The latency of the 4 most-recently completed I/Os
- The number of pending I/Os when each of the 4 completed I/Os arrived

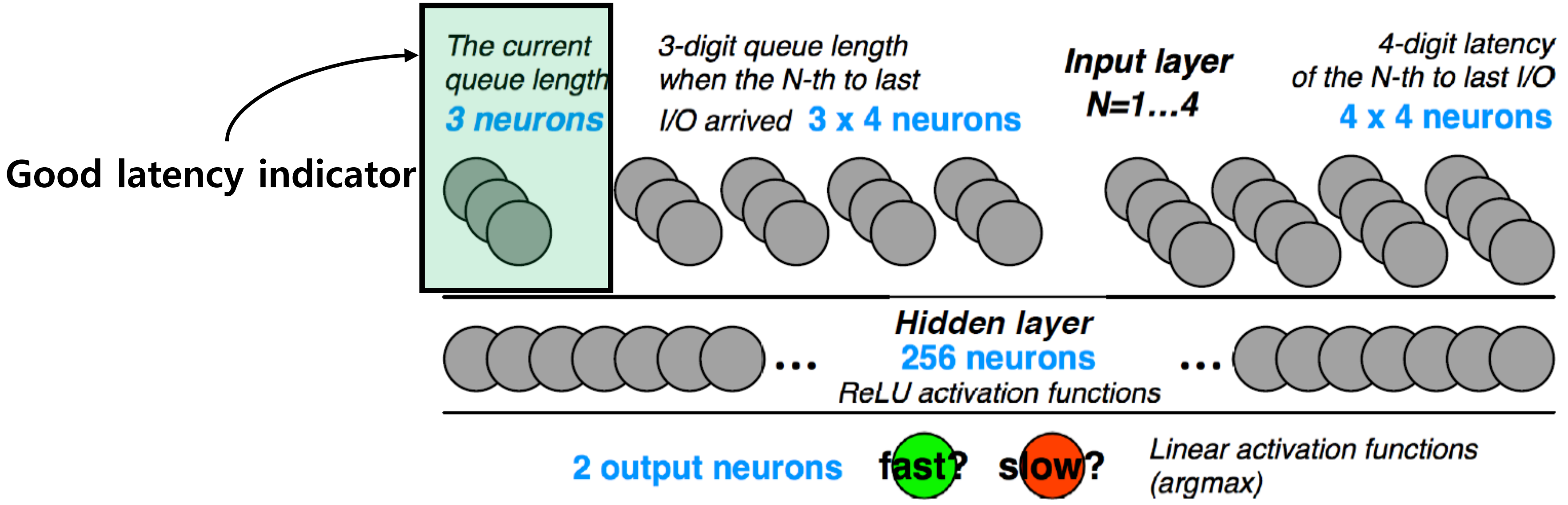
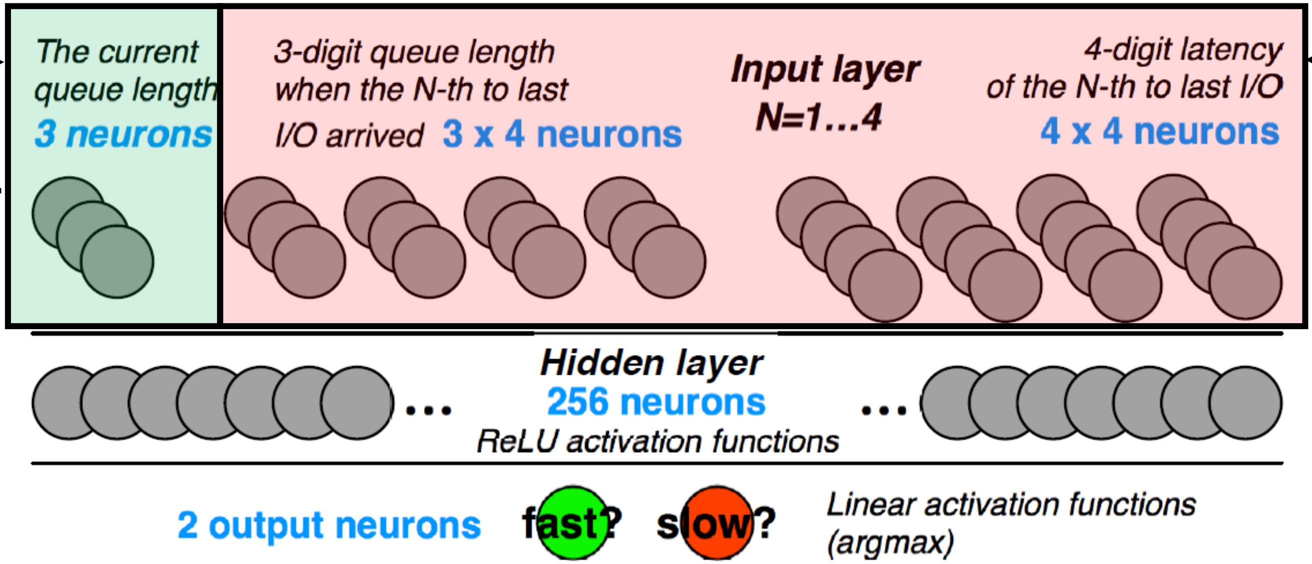


Figure 6: **Light neural network.** *The figure depicts LinnOS 3-layer neural network explained in Section 4.3.*

3) Input features

- The number of pending I/Os (4KB pages) when an incoming I/O arrives
- The latency of the 4 most-recently completed I/Os
- The number of pending I/Os when each of the 4 completed I/Os arrived

Good latency indicator



Low history queue length + high history latency = internal disruption

Figure 6: **Light neural network.** *The figure depicts LinnOS 3-layer neural network explained in Section 4.3.*

3) Input features

- The number of pending I/Os (4KB pages) when an incoming I/O arrives
- The latency of the 4 most-recently completed I/Os
- The number of pending I/Os when each of the 4 completed I/Os arrived

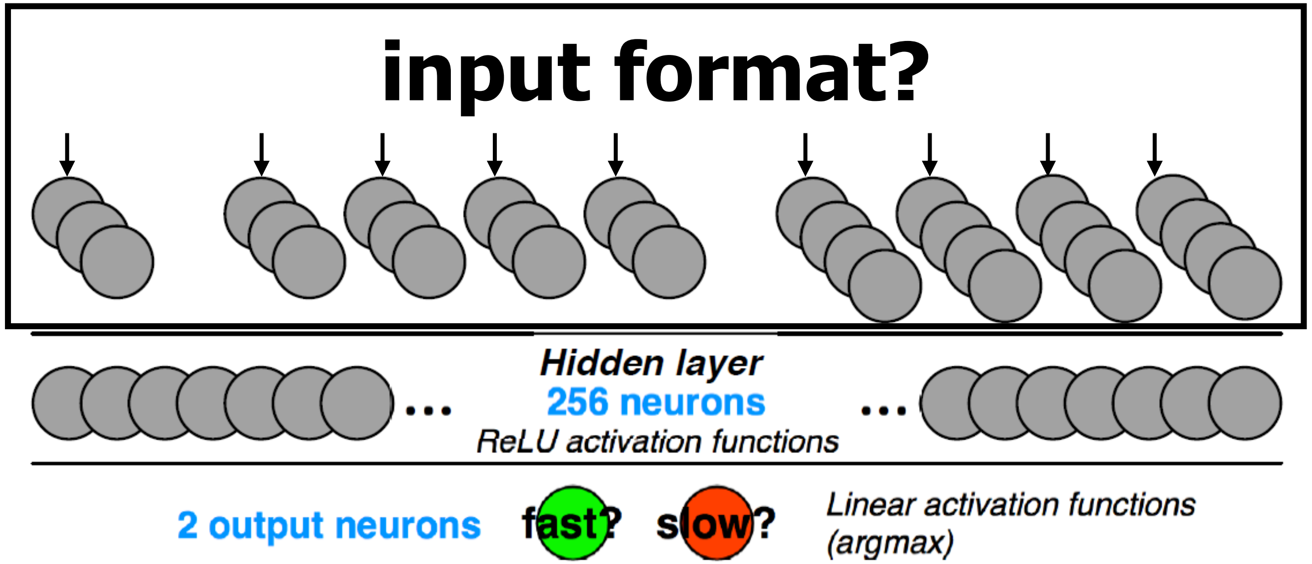
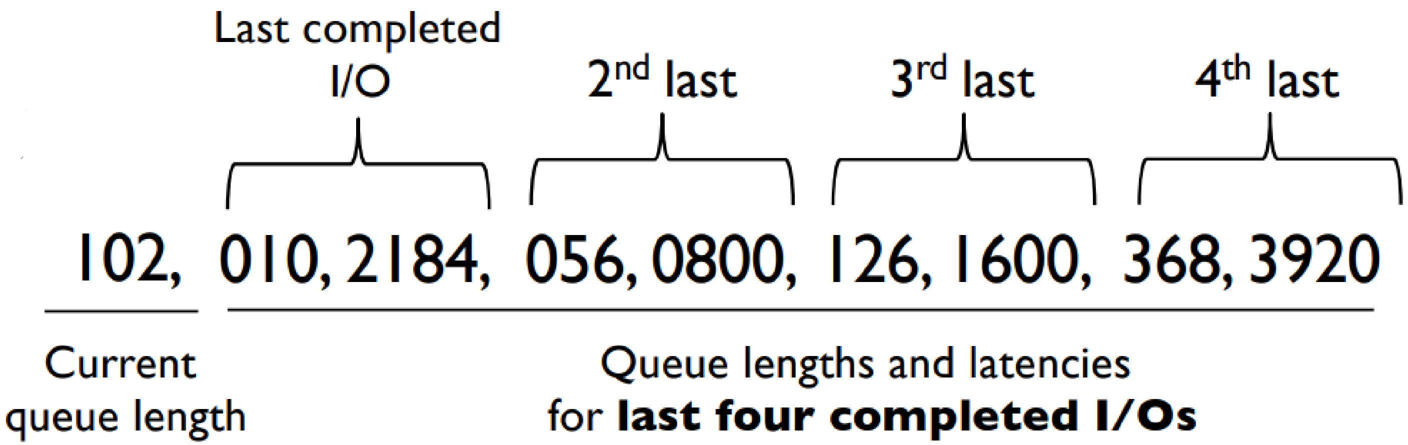


Figure 6: **Light neural network.** *The figure depicts LinnOS 3-layer neural network explained in Section 4.3.*

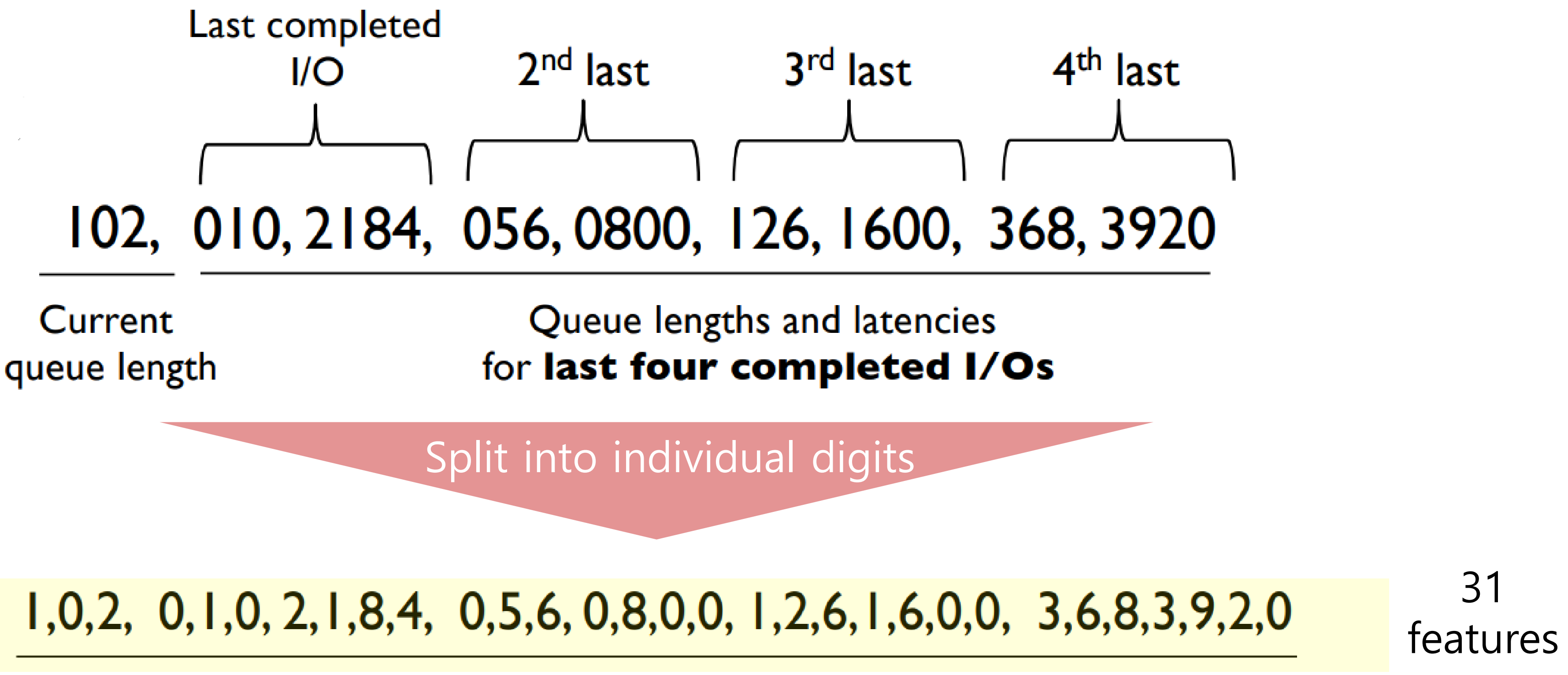
3) Input features

- Format the number of pending I/Os into three decimal digits
- Format μs latency value into four digits
- LinnOS model takes 31 input features, each a one-digit decimal number



3) Input features

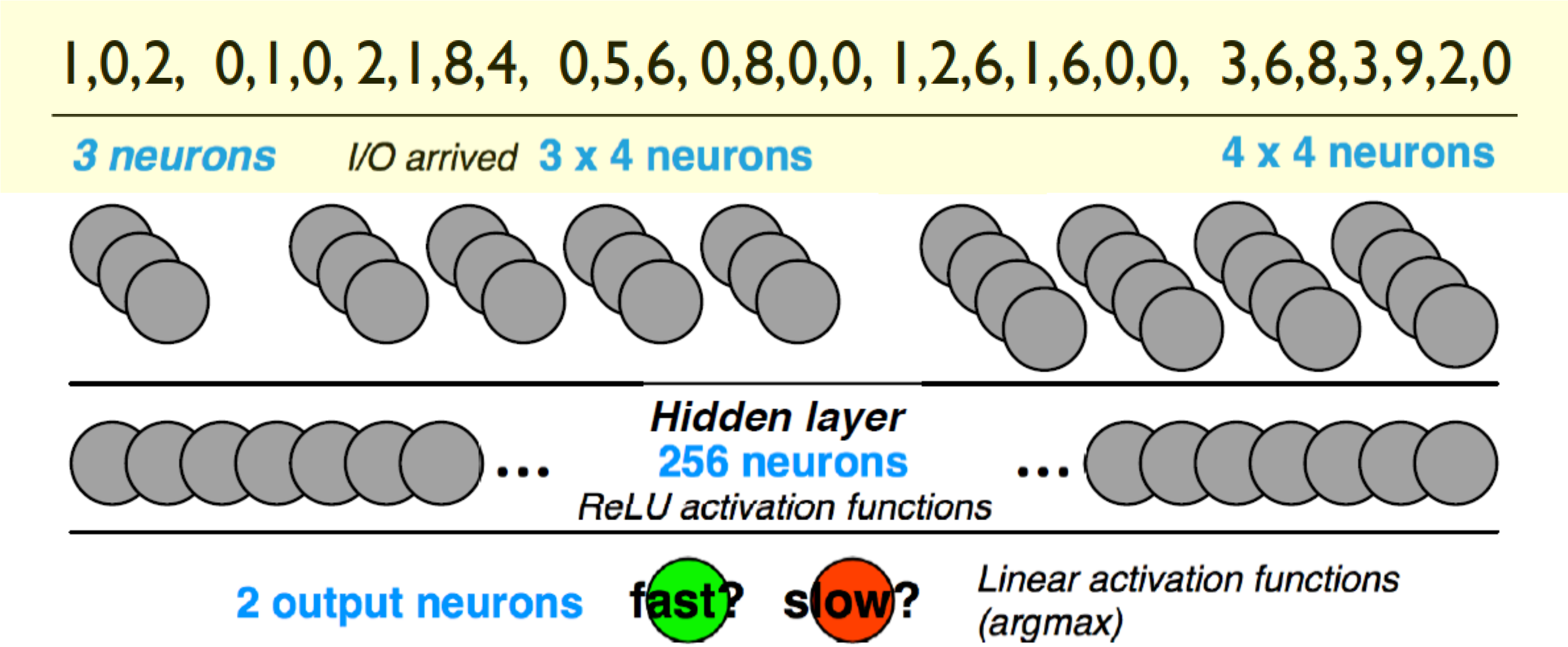
- Format the number of pending I/Os into three decimal digits
- Format μs latency value into four digits
- LinnOS model takes 31 input features, each a one-digit decimal number



3) Input features

- Format the number of pending I/Os into three decimal digits
- Format μs latency value into four digits
- LinnOS model takes 31 input features, each a one-digit decimal number

3 fully-connected layers (31 - 256 – 2)

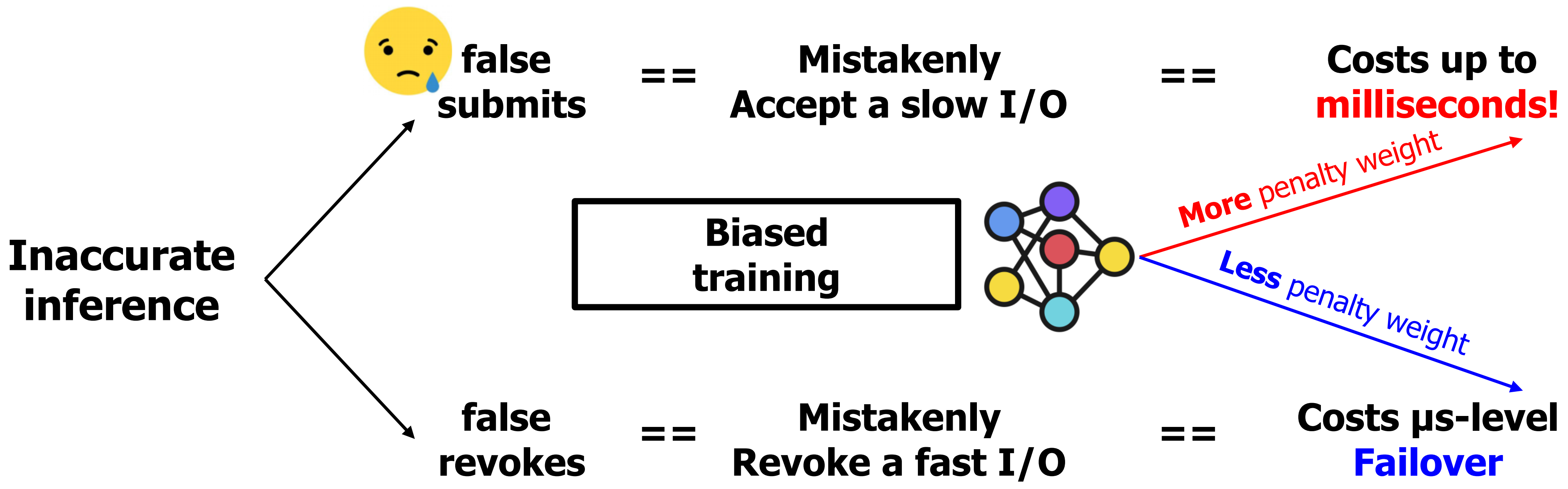


Design Solutions

1) Training Data Collection
2) Labeling (with Inflection Point)
3) Light Nerual Network Model
4) Improving Accuracy
5) Improving Inference Time

4) Improving Accuracy

- Wrong inference penalty is small for false revokes but high for false submits
- Use customized loss function: *biased training*
- Reduce false submits by allowing more false revokes



Design Solutions

1) Training Data Collection
2) Labeling (with Inflection Point)
3) Light Nerual Network Model
4) Improving Accuracy
5) Improving Inference Time

5) Improving Inference Time

Quantization

- Storage functionalities are striping and partitioning using mod operations over integers
- Floting point calculations are expensive and hard to manage inside the OS
- Adopt DNN quantization by maintaining precision of three decimal points

Co-processors

- Can Utilize one additional CPU core (if available)
- reduce the inference time from 6 to 4μs with 2-threaded optimized matrix multiplication

Summary

Input features

- Current queue length
- Queue lengths and latencies of history I/Os

Linn OS

Output labeling

- Binary classification (fast/slow)
- Inflection point Algorithm

Handling inaccuracy

- Biased training
- Different Penalty weights for false submit/revoke

Setup

1) workload

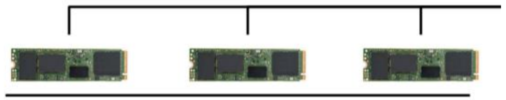
real production SSD-level traces
- Microsoft Azure server
- Bing Index server
- Bing Select server
- Cosmos server

Azure Bing

Cosmos •••

2) devices

- 3 Samsung SM951 M.2 NVMe



Three **homogeneous consumer-level** SSDs

- Intel P4600
- Samsung PM1725a NVMe
- WD ultrastar DC SN200 NVMe

enterprise-level SSDs

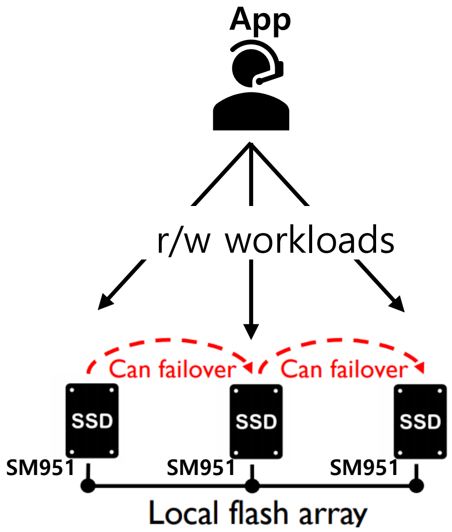
OR

Three **heterogenous enterprise-level** SSDs

- I8 CPU core (36-thread)
- 128GB DRAM
- + 6 SSDs for accuracy evaluation

3) The experiments

Methods compared:
- Baseline
- Cloning
- Hedging at p95
- Hedging at IP
- Simple heuristic
- Advanced heuristic
- LinnOS without hedging
- LinnOS



- **Inflection Point (IP) Stability**

	Consumer	Enterprise
Azure	p73.3, p77.0, p91.4	p91.0, p93.2, p97.8
BingIndex	p80.0, p94.5, p98.5	p80.1, p83.3, p97.0
BingSelect	p72.0, p76.9, p87.2	p75.3, p83.7, p86.8
Cosmos	p73.4, p82.5, p84.1	p83.2, p84.8, p95.1

Table 2: **Inflection point (IP) settings.** *This table, as explained in Section 5.2, shows the IP values that our algorithm in Section 4.2.1 computed for every workload-device pair.*

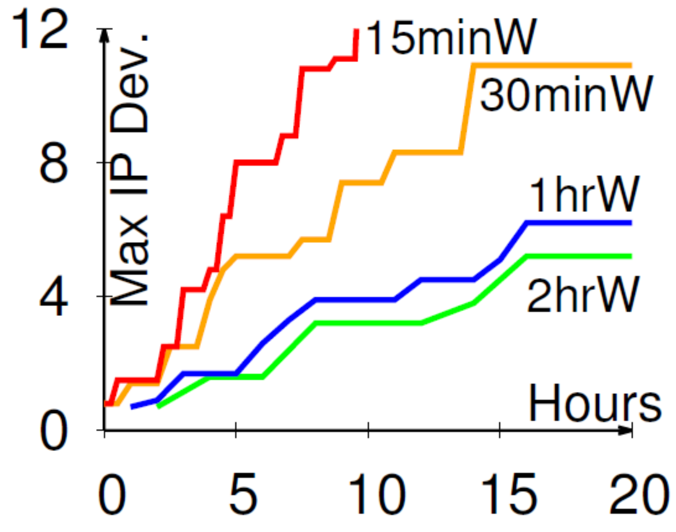


Figure 7: **IP stability.**

- **reduces average latency by 9.6-79.6%**

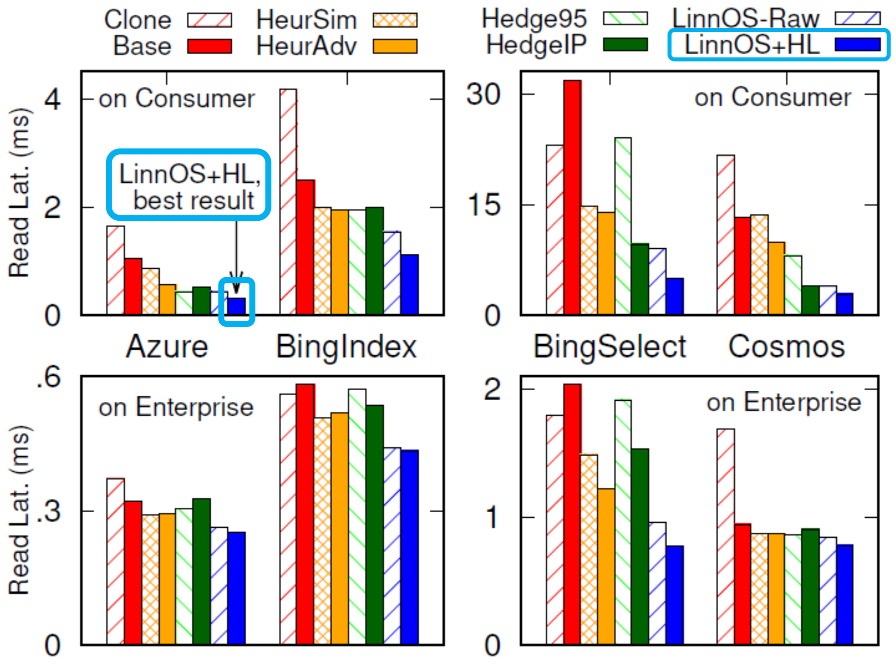
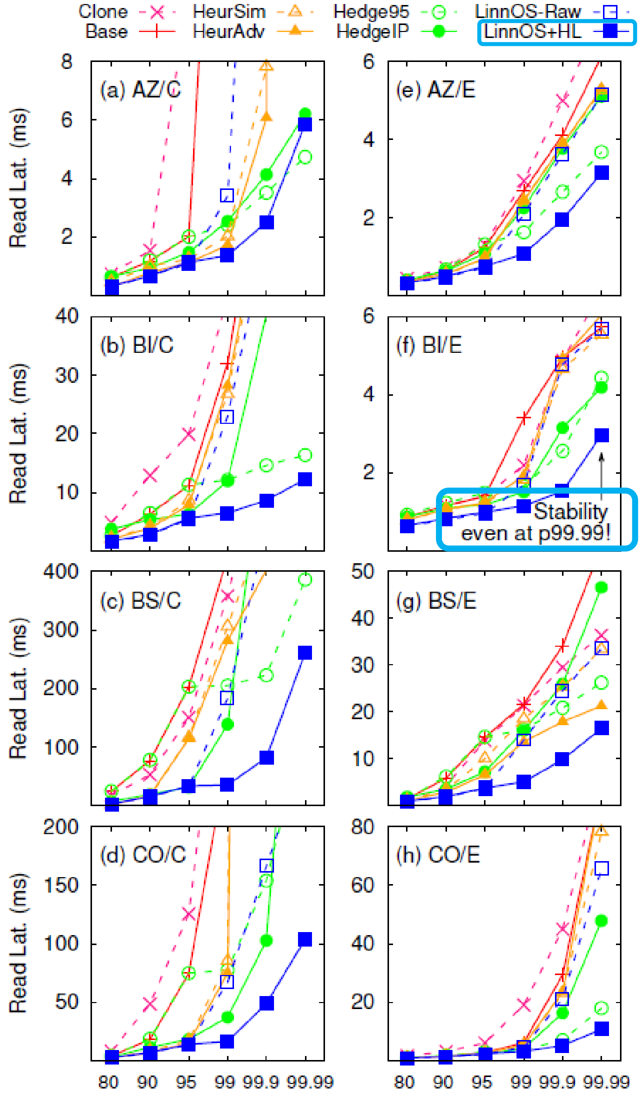


Figure 8: **Average latencies.** *The figures show that LinnOS consistently outperforms all other methods, as explained in Section 5.3. The top and bottom graphs represent experiments on the consumer and enterprise arrays, respectively.*

- **Brings stable latencies**



- high accuracy with low false submit/revoke

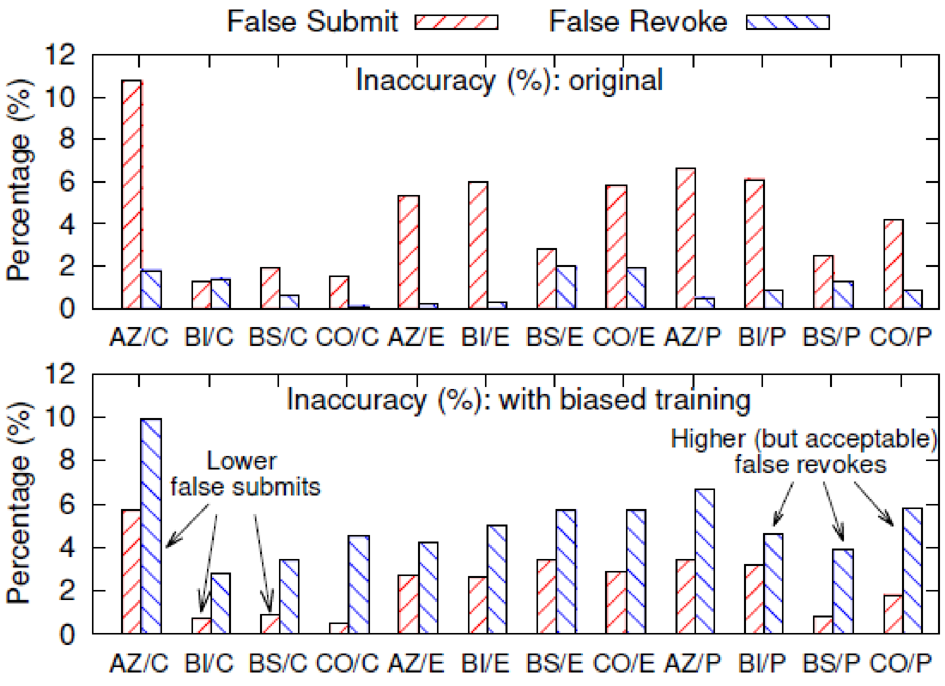
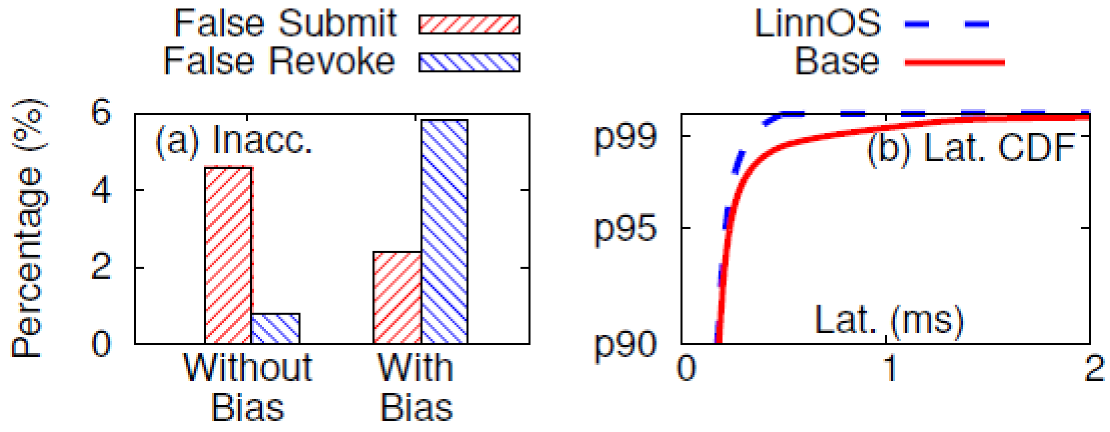


Figure 10: **Low inaccuracy.** *The figure shows the percentage of false submits and false revokes. Note that only false submits really matter (see Section 5.4). Additionally, "P" represents other device models that we can access from a public cloud. For graph readability, here for "P" we only show the results for one device model, while the observations stand across the rest. In total, the accuracy evaluation covers 10 device models (1C+3E+6P).*

- Works on public traces



- Helps MongoDB and FS

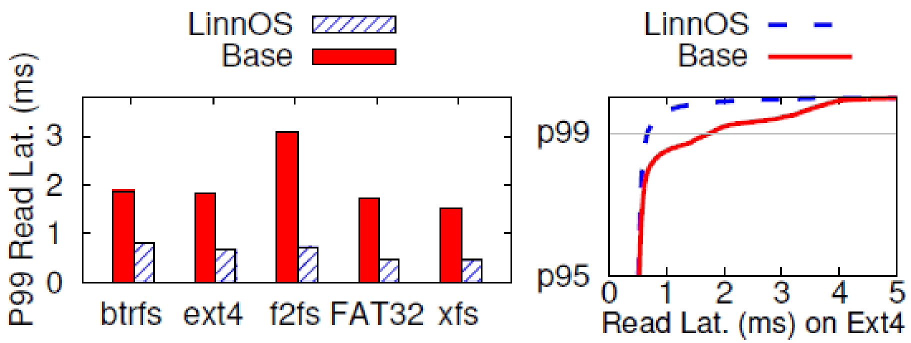


Figure 13: **MongoDB on different filesystems.** *This figure shows that LinnOS can easily help data applications achieve more predictable latency (§5.6.3).*

LinnOS

- Demonstrate that it is possible to incorporate machine learning into OS
- Outperforms many other popular methods
- Successfully predict the speed of every I/O to flash storage

LinnOS : Predictability on Unpredictable Flash Storage with a Light Neural Network

Mingzhe Hao, Levent Toksoz, Nanqinqin Li, Edward Edberg Halim,

In 2020 USENIX Symposium on Operating Systems Design and Implementation

Thank You!

2021. 01. 26

Presentation by Han, Yejin

hyj0225@dankook.ac.kr