FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives

ISCA 2018

Arash Tavakkol, Mohammad Sadrosadati, Saugata Ghose,
Jeremie S. Kim, Yixin Luo, Yaohua Wang, Nika Mansouri Ghiasi,
Lois Orosa, Juan Gómez-Luna, Onur Mutlu

ETH zürich
Carnegie Mellon University
UNICAMP
SAFARI
SAFARI Research Group

Executive Summary

- Modern solid-state drives (SSDs) use new storage protocols (e.g., NVMe) that eliminate the OS software stack
  - I/O requests are now scheduled inside the SSD
  - Enables high throughput: millions of IOPS
- OS software stack elimination removes existing fairness mechanisms
  - We experimentally characterize fairness on four real state-of-the-art SSDs
  - Highly unfair slowdowns: large difference across concurrently-running applications
- We find and analyse four sources of inter-application interference that lead to slowdowns in state-of-the-art SSDs
- FLIN: a new I/O request scheduler for modern SSDs designed to provide both fairness and high performance
  - Mitigates all four sources of inter-application interference
  - Implemented fully in the SSD controller firmware, uses < 0.06% of DRAM space

Table of Contents

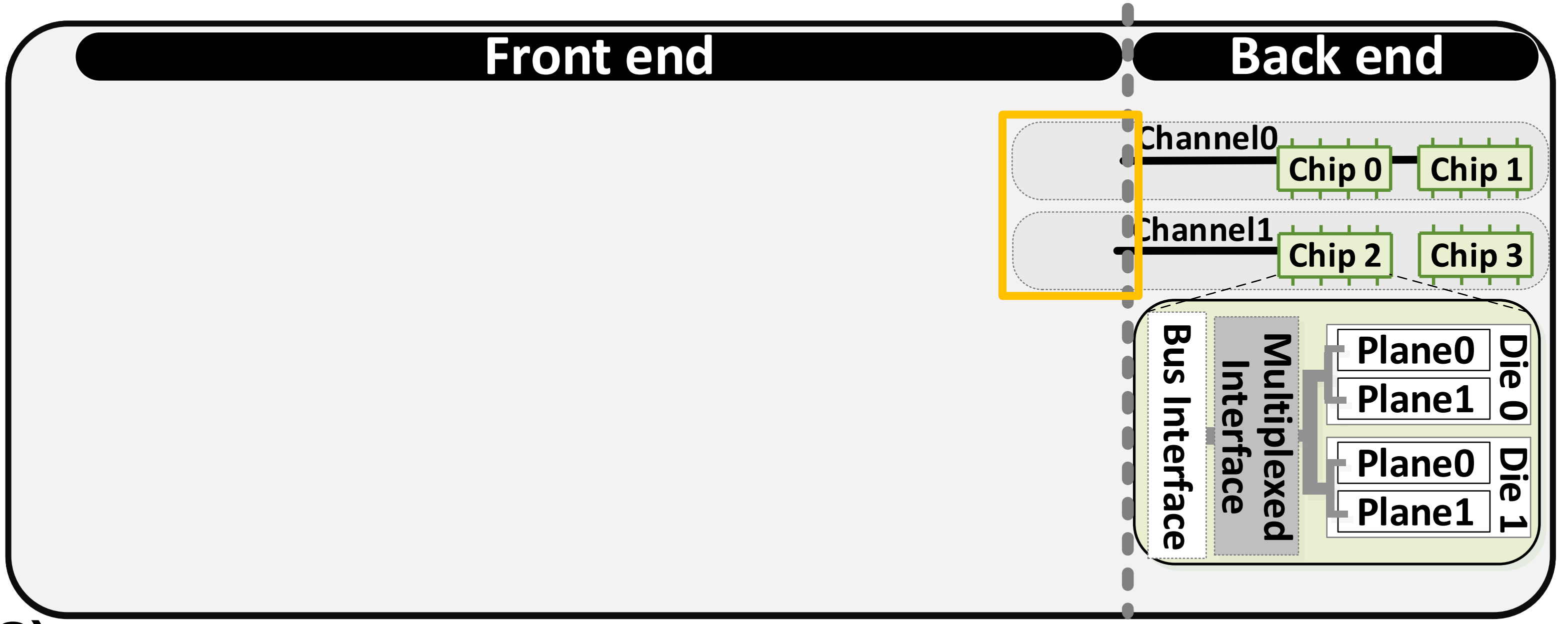
Table of Contents

- Background: Modern SSD design
- Sources of unfairness in modern solid state drives
- FLIN: Flash Level Interference-aware scheduler
- Experimental Evaluation

- Strengths and Weaknesses
- Related work
- Open discussion

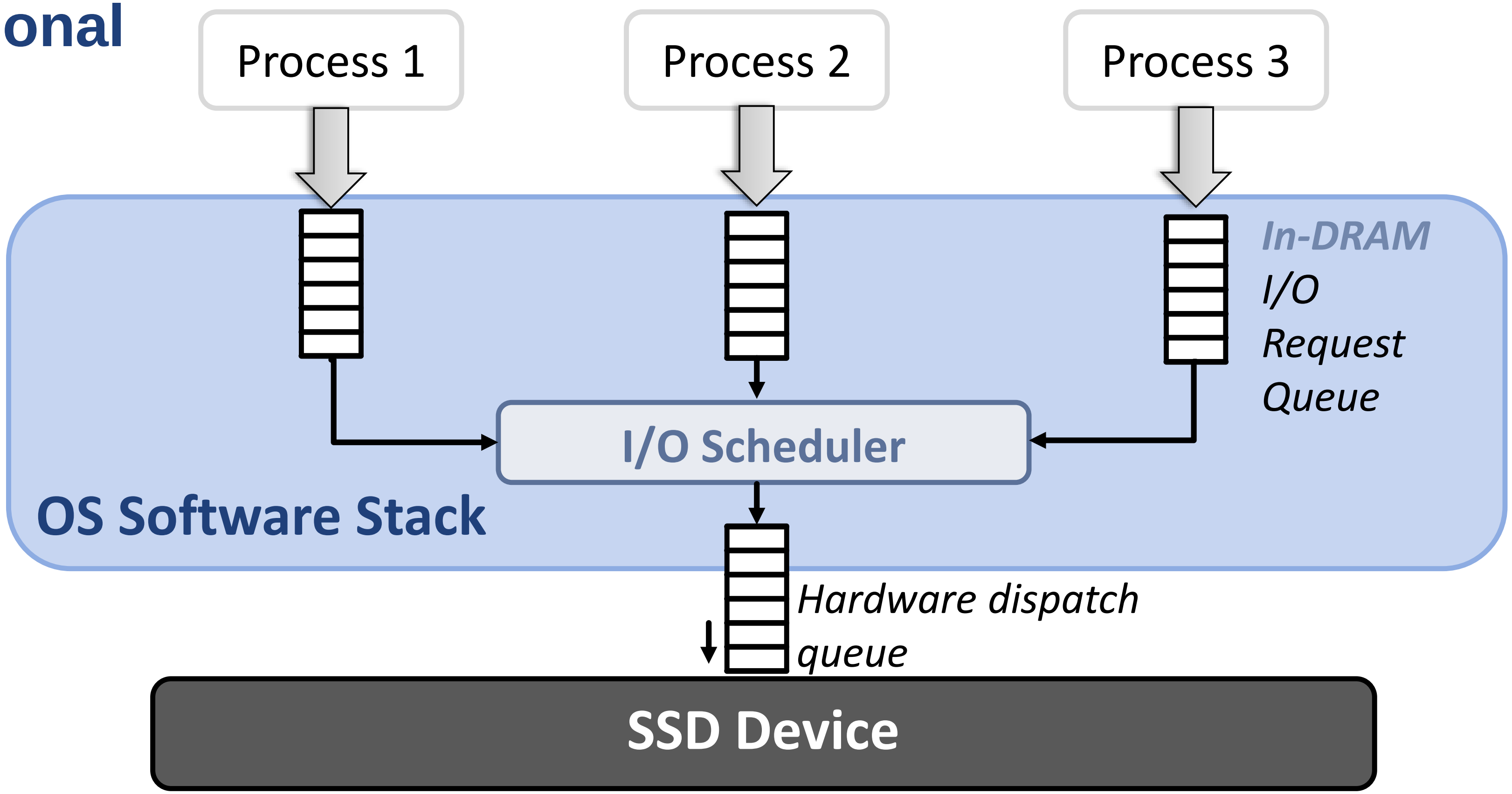
Internal Components of a Modern SSD

- Back end: Storage
  - Flash chips
- Front end: Control
  - Host Interface Logic **(HIL)**
    - Communicates with host
  - Flash Translation Layer **(FTL)**
    - Manages resources
    - Processes I/O
  - Flash Channel Controllers **(FCC)**
    - Direct access to back end



Conventional Host Interface Protocols

- SSDs adopted **conventional** host interface protocols
  - Designed for magnetic drives
- OS Software Stack handles requests
- Limited to **thousands** of I/O requests



Host Interface Protocols in Modern SSDs

- Modern SSDs use **high performance** host interface protocols
- Bypasses OS, **SSDs handle requests directly**

- Very **high throughput**
- Fairness implemented through software stack is lost

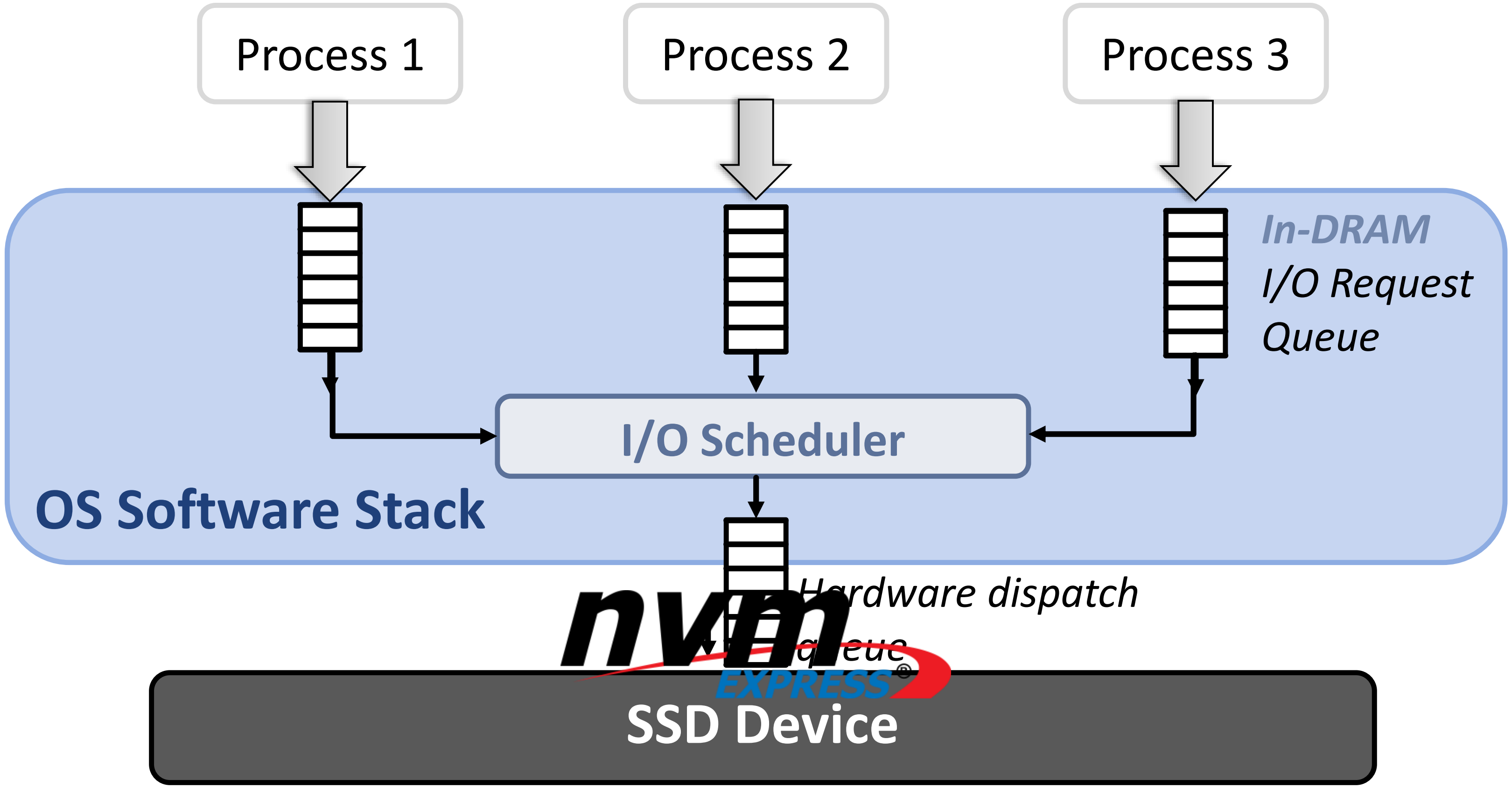
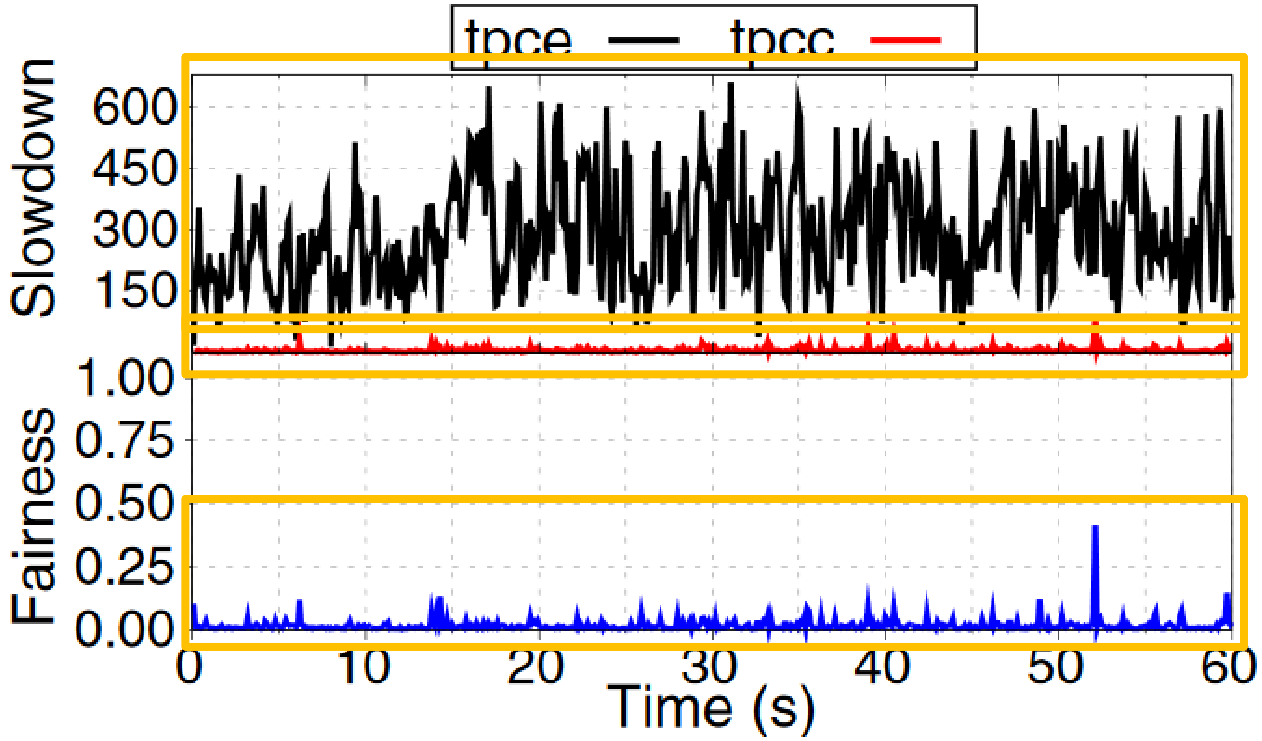


Table of Contents

Measuring (Un)fairness

- **Flow**:
  - A series of I/O requests generated by an application
- **Slowdown**:
  - $Slowdown = \frac{Shared\ response\ time}{Non\text{-}shared\ response\ time}$
- **Unfairness**:
  - $Unfairness = \frac{Max\ Slowdown}{Min\ Slowdown}$
- **Fairness**
  - $Fairness = \frac{1}{Unfairness}$

Representative Example



Causes of Unfairness

- **Interference** among concurrently running flows
- Detailed study of a simulation with MQSim [1]
- Four different sources of interference are uncovered

[1] MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs
https://github.com/CMU-SAFARI/MQSim

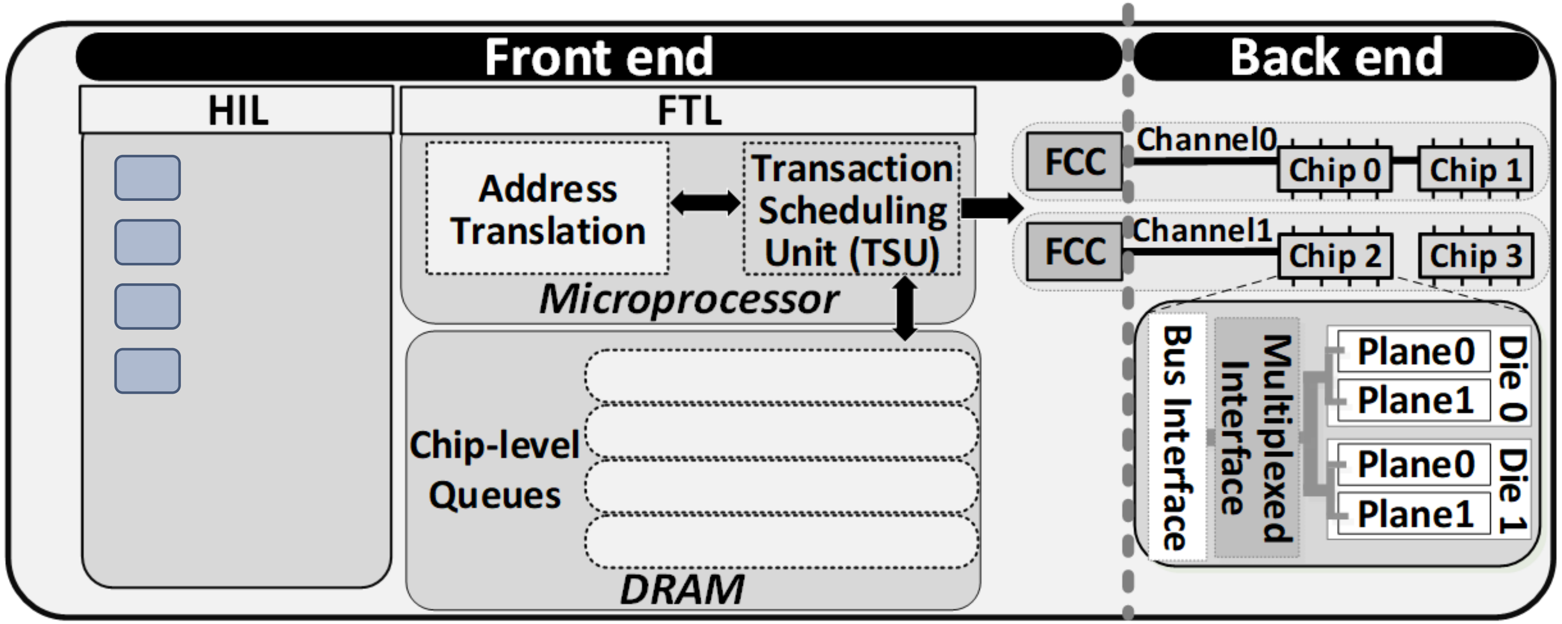
Source 1: Flows With Different I/O Intensities

- The **I/O intensity** of a flow affects the average queue wait time of flash transactions

The average response time of a low-intensity flow **substantially increases** due to interference from a high-intensity flow

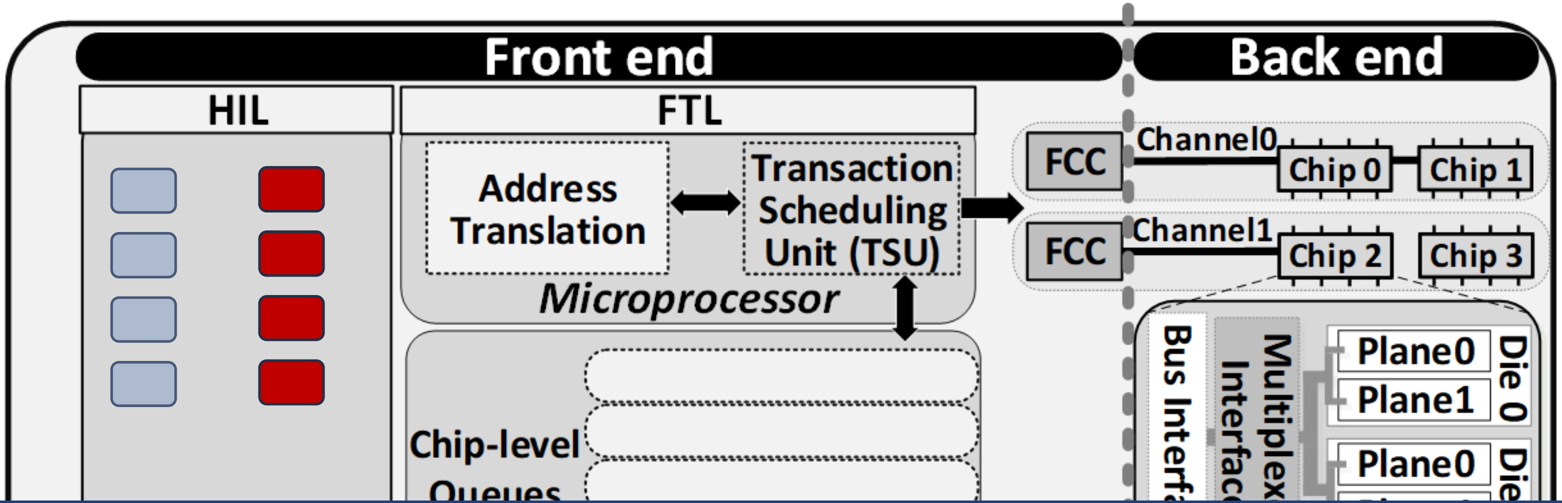
Source 2: Different Request Access Patterns

- Some flows take advantage of **chip level parallelism** in back end
- Leads to low queue time



Source 2: Different Request Access Patterns

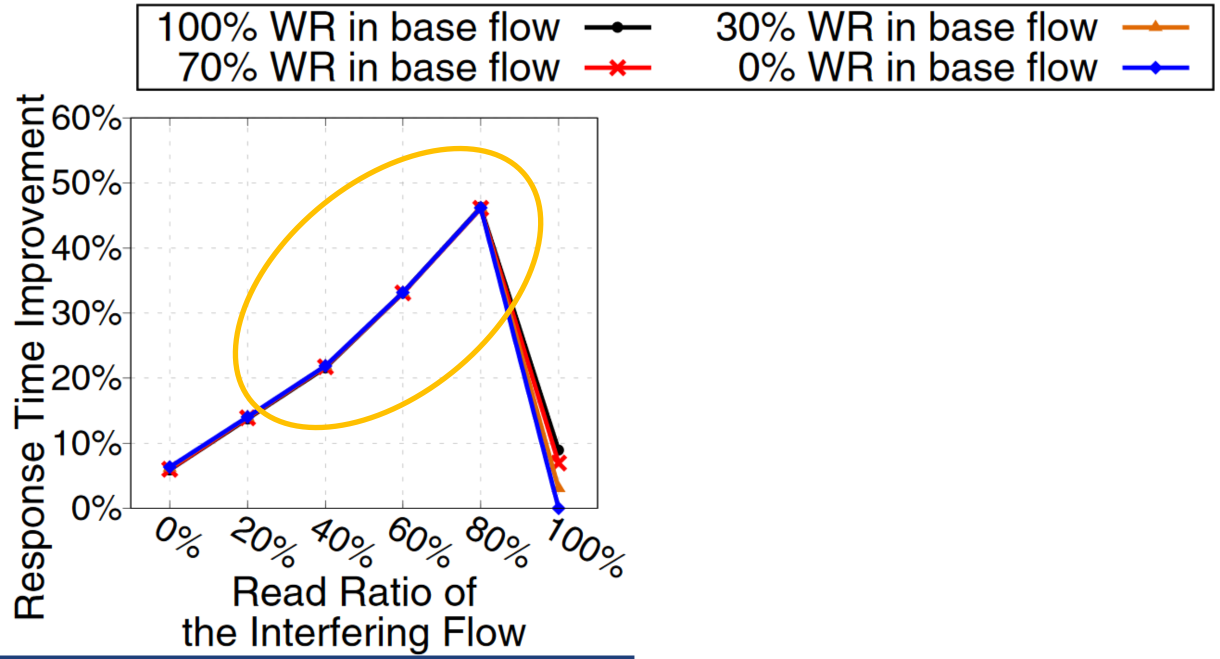
- Other flows have access patterns that **do not exploit patterns**



Flows with **parallelism-friendly access patterns** are **susceptible to interference** from flows whose access patterns do not exploit parallelism

Source 3: Flows With Different R/W Ratios

- Common schedulers prioritize Read operations
- Write transactions have increased wait times



When flows have **different read/write r**
existing schedulers do not effectively provid

Source 4: Different Garbage Collection Demands

- NAND flash memory performs writes out of place
  - To be rewritten, memory needs to be erased first
  - Erases can only happen on an entire flash block (hundreds of flash pages)
  - Pages marked invalid during write
- Garbage collection (GC) selects mostly empty blocks, moves remaining data and frees block
- High-GC flow: flows with a higher write intensity induce more garbage collection activities

The GC activities of a **high-GC** flow can unfairly block flash transactions of a **low-GC** flow

Summary

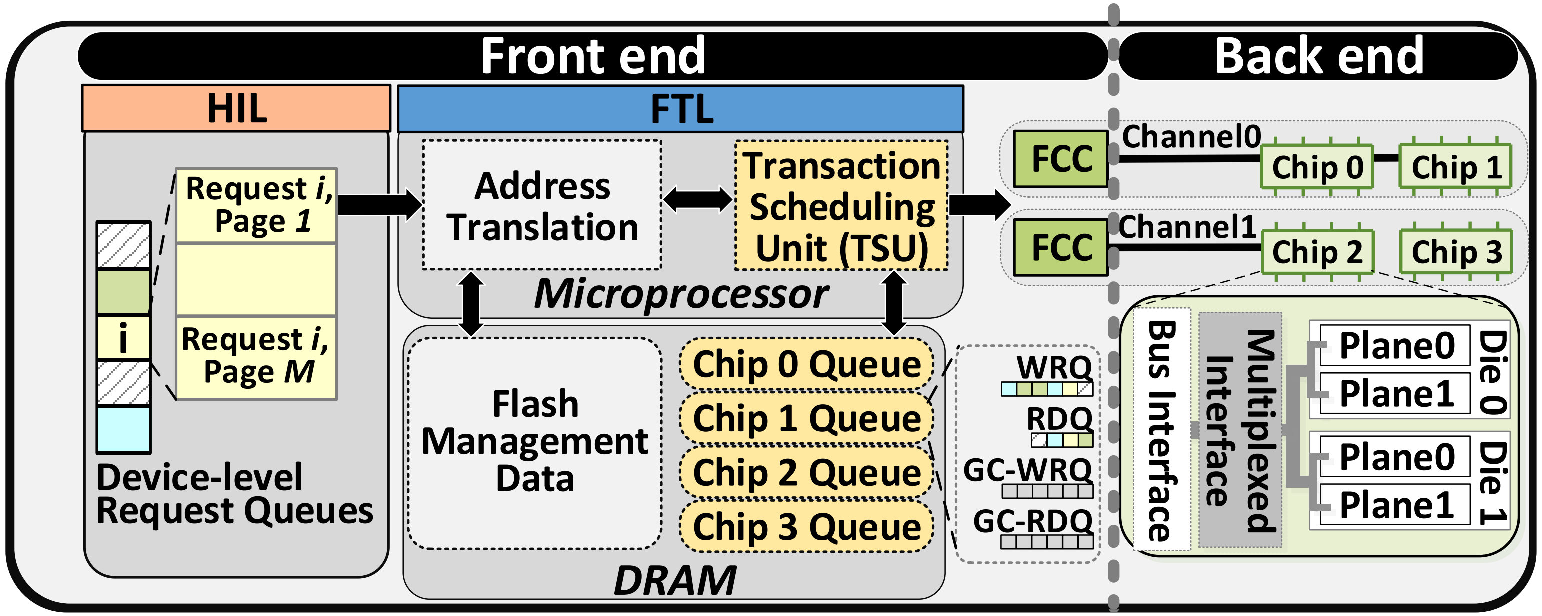
- Four sources of unfairness
  - Differing intensities
  - Differing request access patterns
  - Differing read/ write ratios
  - Differing GC demands

The goal is to design a new I/O scheduler that provides **fairness**, **maximum performance** and **throughput**

Table of Contents

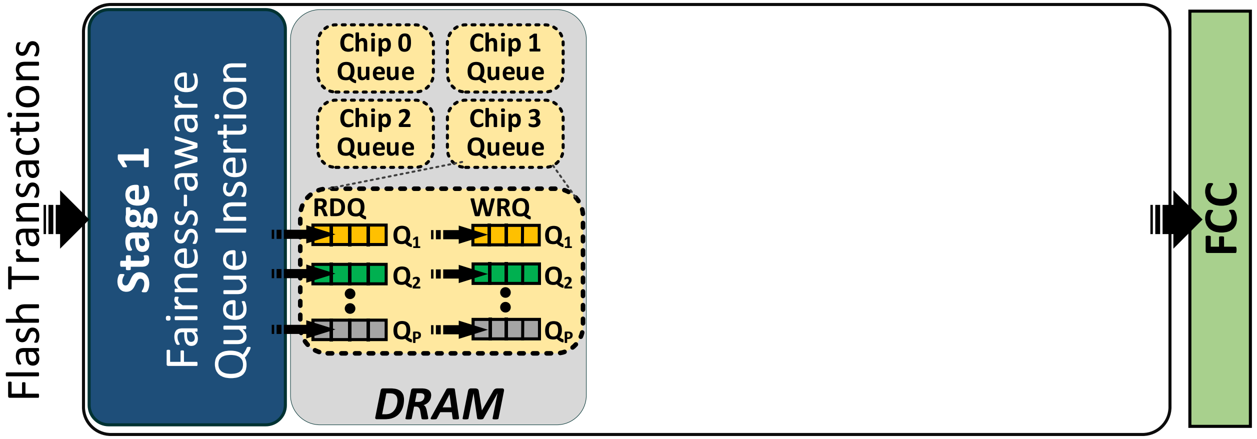
FLIN: Flash Level Interference Aware Scheduler

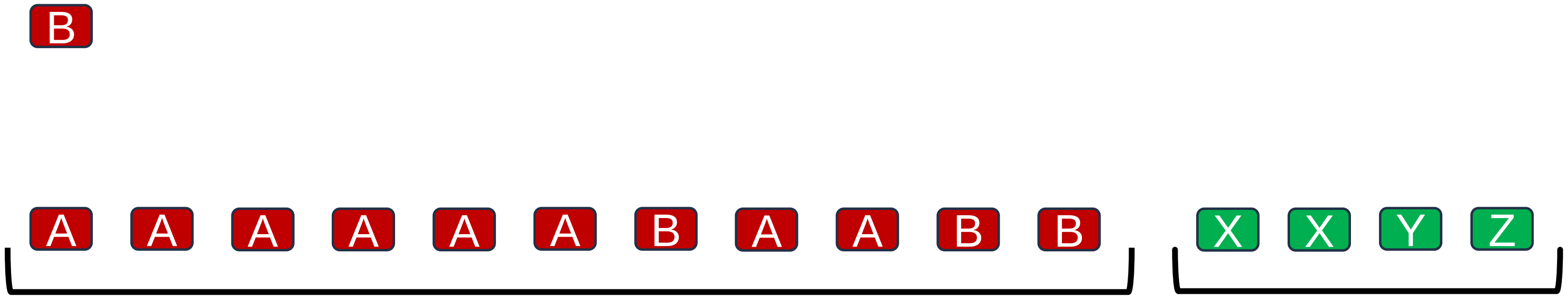
- Improved I/O request scheduler
- Replaces the transaction scheduling unit
- **Improves fairness** while **keeping throughput**
- Implemented in the SSD **firmware**, no hardware modification needed



FLIN: Stage 1
Fairness-aware Queue Insertion

- Separate, per chip read and write queues
- Low intensity flows have priority over high intensity flows
- Requests get reordered to guarantee fairness

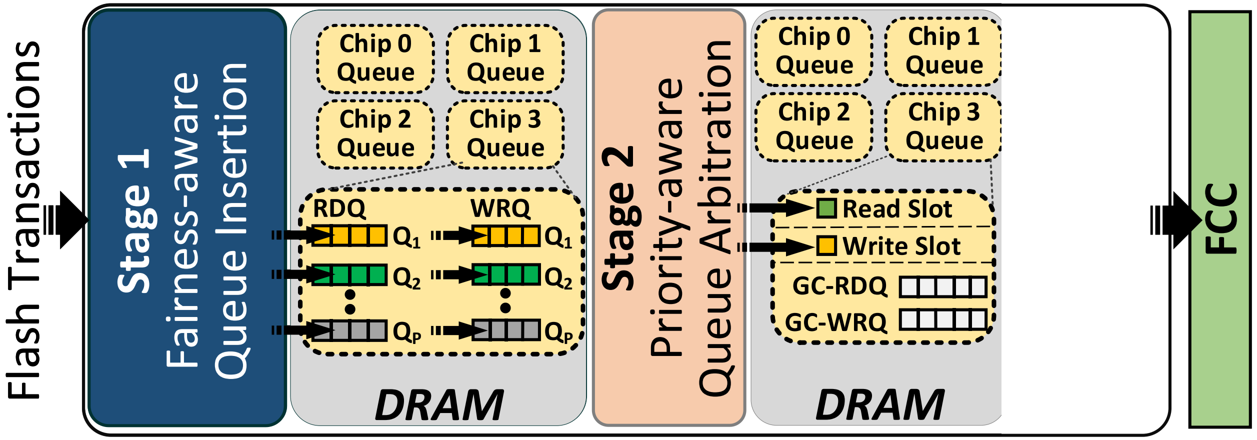


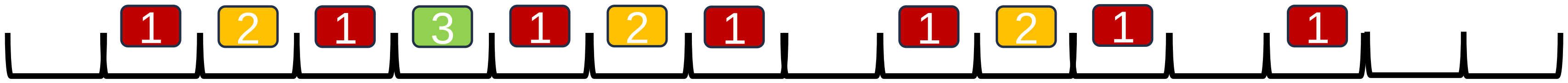


I/O Requests from high intensity flows I/O Requests from low intensity flows

FLIN: Stage 2
Priority-aware Queue Arbitration

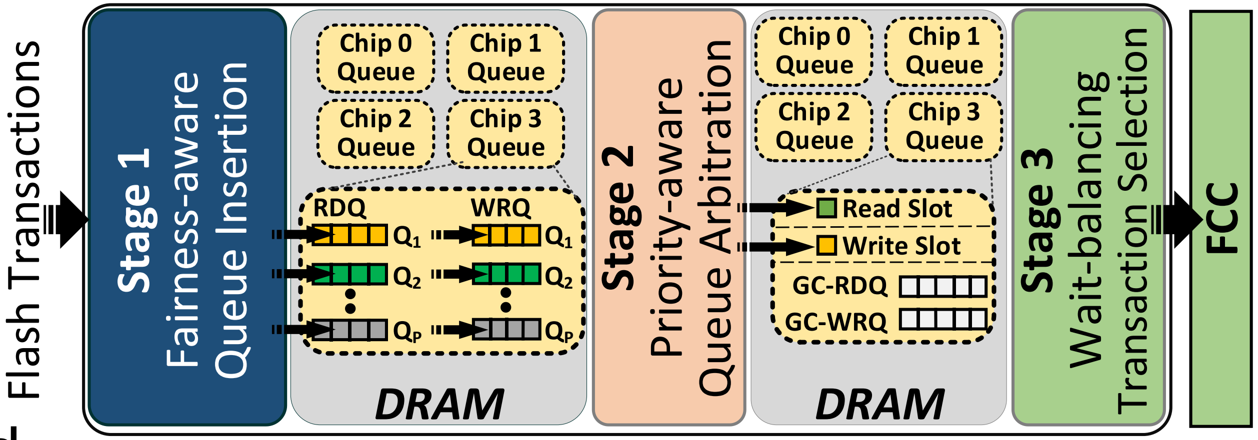
- Host can assign **priority level**
- Select one read and one write transaction and deliver to Stage 3
  - **Weighted round-robin** algorithm
  - Higher priority means more transactions
  - **No starvation**





FLIN: Stage 3
Wait-balancing Transaction Selection

- Minimizes interference of differing read/ write ratios and GC demands
- Chooses which transaction to dispatch to the FCC
- Instead of prioritizing reads, it prioritizes the one with less estimated proportional wait time ($t_{pw} = \frac{t_{wait}}{t_{process}}$)



- If write is selected, perform GC instead if available free space is smaller than some pre-defined threshold

Table of Contents

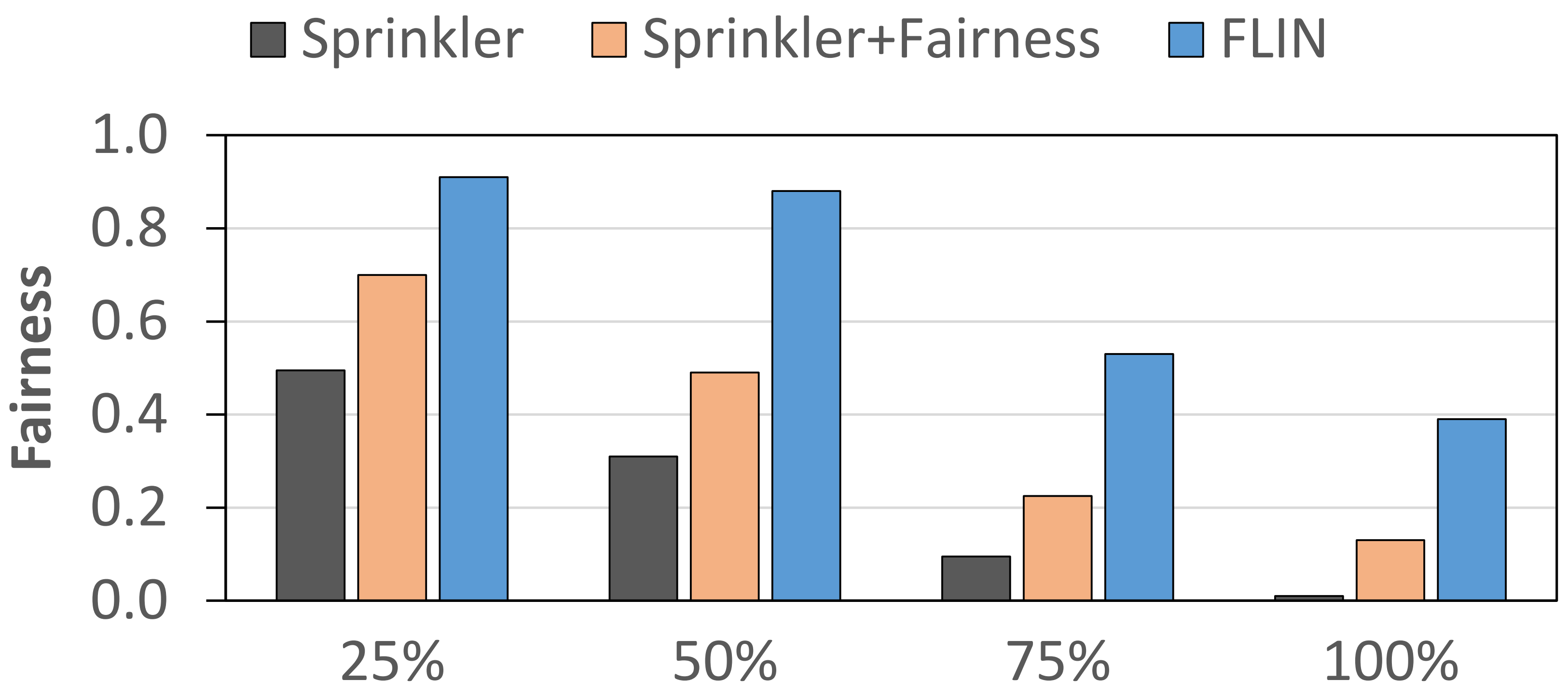
Evaluation Methodology

- **Simulation** based on MQSim
  - Protocol: NVMe 1.2 over PCIe 3.0
  - Model SSD: 480 GB size
  - Organization: 8 channels, 2 planes per die, 4096 blocks per plane, 256 pages per block, 8kB page size
- 40 Different model workloads
  - Classified as high or low interference
- 4 Metrics
  - Fairness, maximum slowdown, standard deviation of slowdowns and weighted speedup

Evaluation Baseline

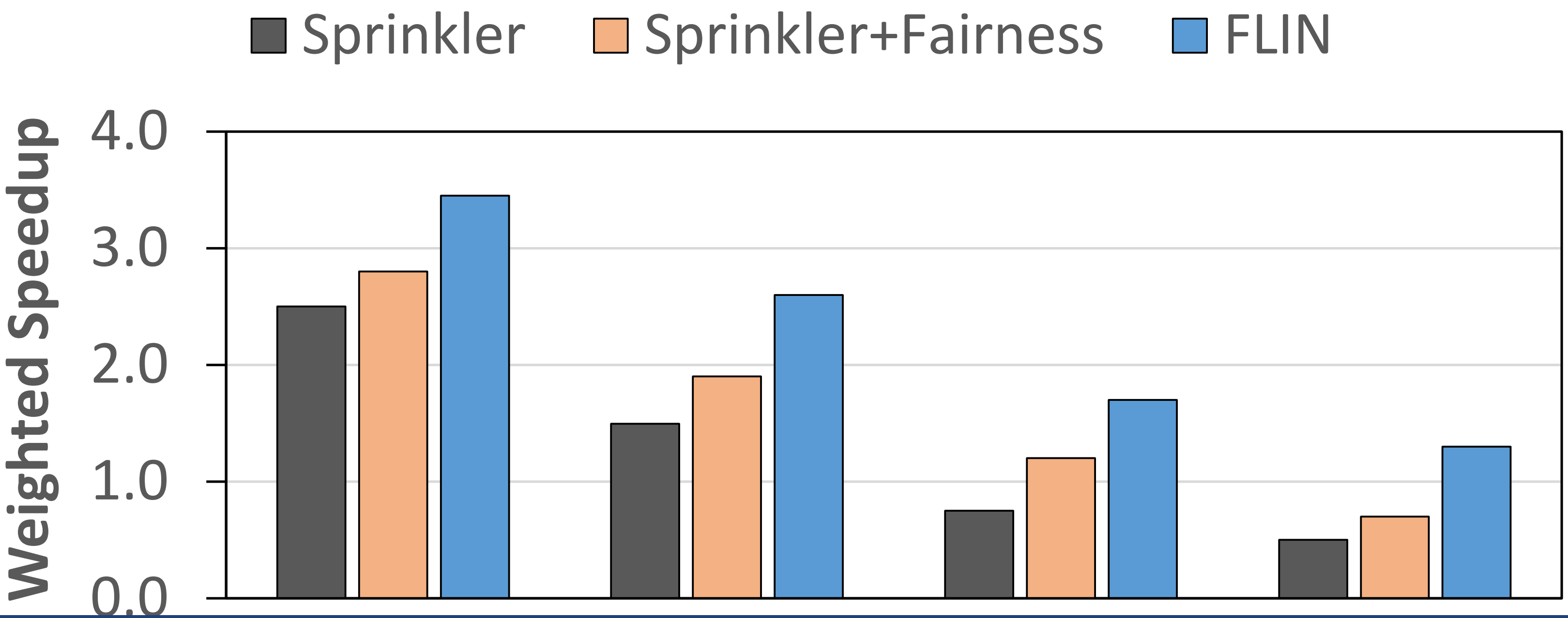
- Sprinkler [Jung et al. HPCA 2014]
  - State-of-the-art high-performance scheduler
- Sprinkler + Fairness [Jung et al. HPCA 2014, Jun et al NVMSA 2015]
  - Sprinkler scheduling algorithm with improved fairness
  - Does not mitigate all sources of interference

Fairness Results



FLIN improves fairness by an **average of 70%,**
by mitigating all four major sources of interference

Speedup Results



FLIN improves performance by an **average of 47%,**
by making use of idle resources in the SSD and improving the performance of **low-interference flows**

Conclusions

- Modern solid-state drives (SSDs) use new storage protocols(e.g., NVMe) that eliminate the OS software stack
  - I/O requests are now scheduled inside the SSD
  - Enables high throughput: millions of IOPS
- OS software stack elimination removes existing fairness mechanisms
  - We experimentally characterize fairness on four real state-of-the-art SSDs
  - Highly unfair slowdowns: large difference across concurrently-running applications
- We find and analyse four sources of inter-application interference that lead to slowdowns in state-of-the-art SSDs
- FLIN: a new I/O request scheduler for modern SSDs designed to provide both fairness and high performance
  - Mitigates all four sources of inter-application interference
  - Implemented fully in the SSD controller firmware, uses < 0.06% of DRAM space

Table of Contents

- Background: Modern SSD design
- Sources of unfairness in modern solid state drives
- FLIN: Flash Level Interference-aware scheduler
- Experimental Evaluation

- Strengths and Weaknesses
- Related work
- Open discussion

Strengths

- Solution is fully firmware based
  - Only software of one device needs modification
  - Manufacturers have an incentive to implement FLIN
- Very high fairness and some performance improvement
- Well written paper
  - Good background

Weaknesses

- Only a simulation
  - No actual implementation measured
- Model workloads might not be representative of real world scenarios
  - Designed for testing HDDs

Related Work

- Content Popularity-Based Selective Replication for Read Redirection in SSDs
  - Elyasi et al., 2018, MASCOTS
  - Improves performance and fairness by copying stored data

- CARS: A Multi-layer Conflict-Aware Request Scheduler for NVMe SSDs
  - Yang et al., 2019, DATE
  - Similar approach, but focusses on performance rather than fairness

Related Work

- NCQ-Aware I/O Scheduling for Conventional Solid State Drives
  - Fan et al., 2019, IPDPS
  - Native Command Queuing scheduling that is aware of latencies on the host rather than on the device

- An Efficient Hybrid I/O Caching Architecture Using Heterogeneous SSDs
  - Salkhordeh et al., 2019, TPDS
  - Improves throughput and energy efficiency by caching requests more efficiently, using three different layers

Open Discussion

- Can you think of any further improvements?
- Do you think fairness is a good metric?
- Do you think the host should take over more responsibility again?
- Do you think FLIN will be implemented by hardware manufacturers?