FIOS: A Fair, Efficient Flash I/O Scheduler

Stan Park Kai Shen

University of Rochester

Background

- Flash is widely available as mass storage, e.g. SSD
- $/GB still dropping, affordable high-performance I/O
- Deployed in data centers as well low-power platforms
- Adoption continues to grow but very little work in robust I/O scheduling for Flash I/O
- Synchronous writes are still a major factor in I/O bottlenecks

Flash: Characteristics & Challenges

- No seek latency, low latency variance
- I/O granularity: Flash page, 2–8KB
- Large erase granularity: Flash block, 64–256 pages
- Architecture parallelism

- Erase-before-write limitation
- I/O asymmetry
- Wide variation in performance across vendors!

Motivation

- Disk is slow $\rightarrow$ scheduling has largely been performance-oriented
- Flash scheduling for high performance ALONE is easy (just use noop)
- Now fairness can be a first-class concern
- Fairness must account for unique Flash characteristics

Motivation: Prior Schedulers

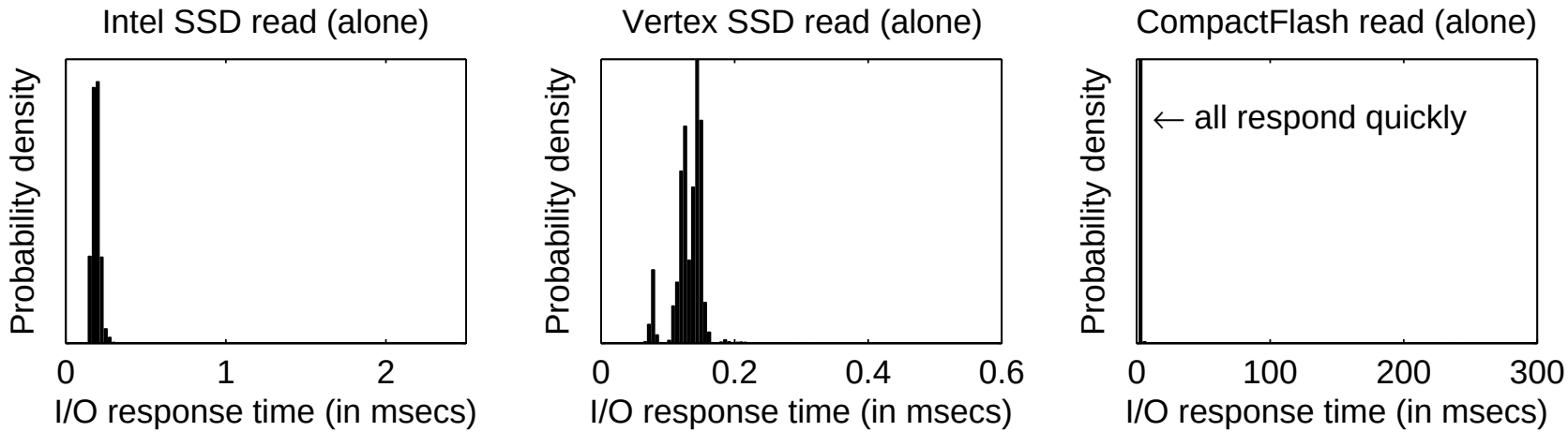
Fairness-oriented Schedulers: Linux CFQ, SFQ(D), Argon

- Lack Flash-awareness and appropriate anticipation support
  - Linux CFQ, SFQ(D): fail to recognize the need for anticipation
  - Argon: overly aggressive anticipation support

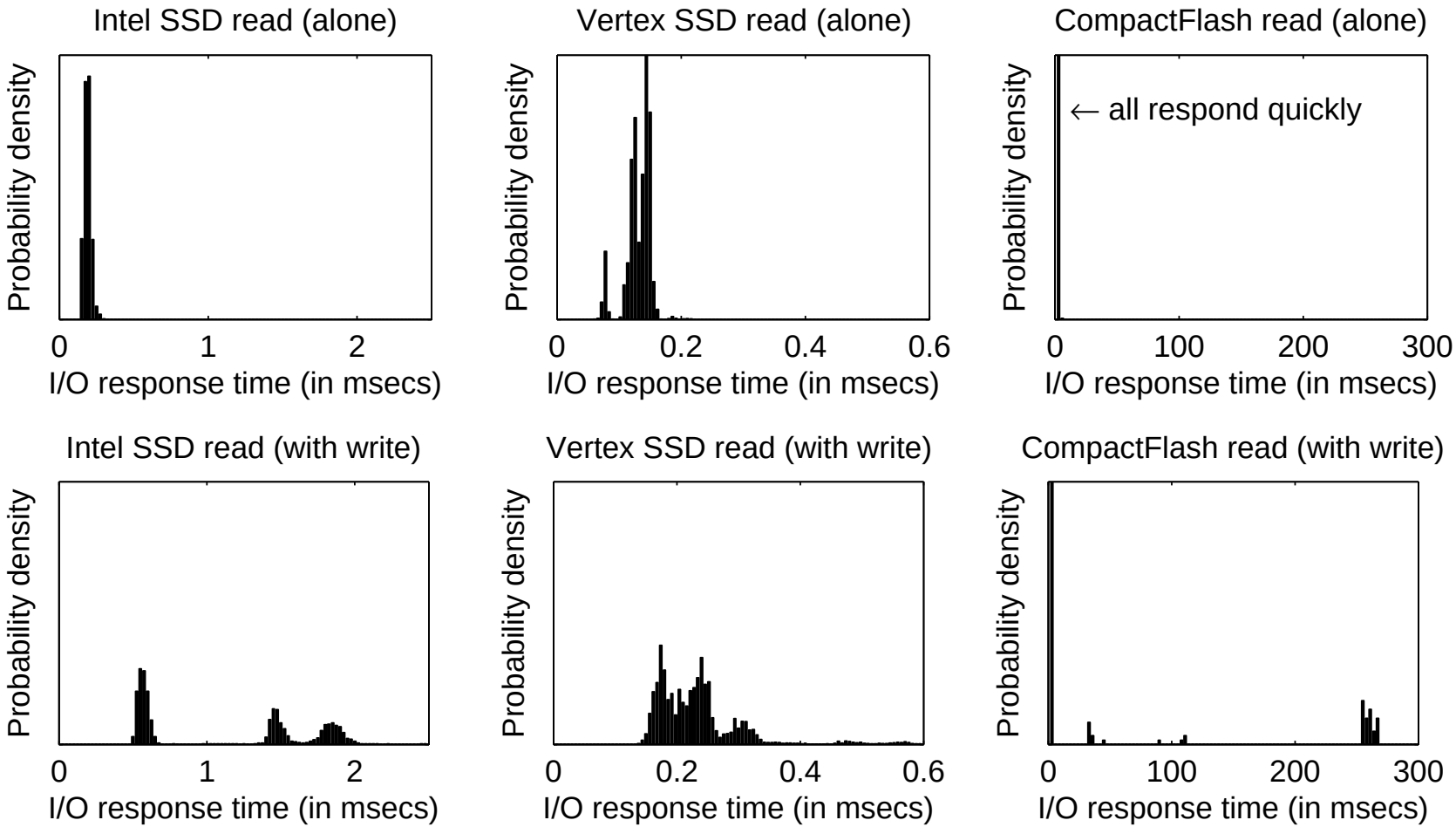
Flash I/O Scheduling: write bundling, write block preferential, and page-aligned request merging/splitting

- Limited applicability to modern SSDs, performance-oriented

Motivation: Read-Write Interference

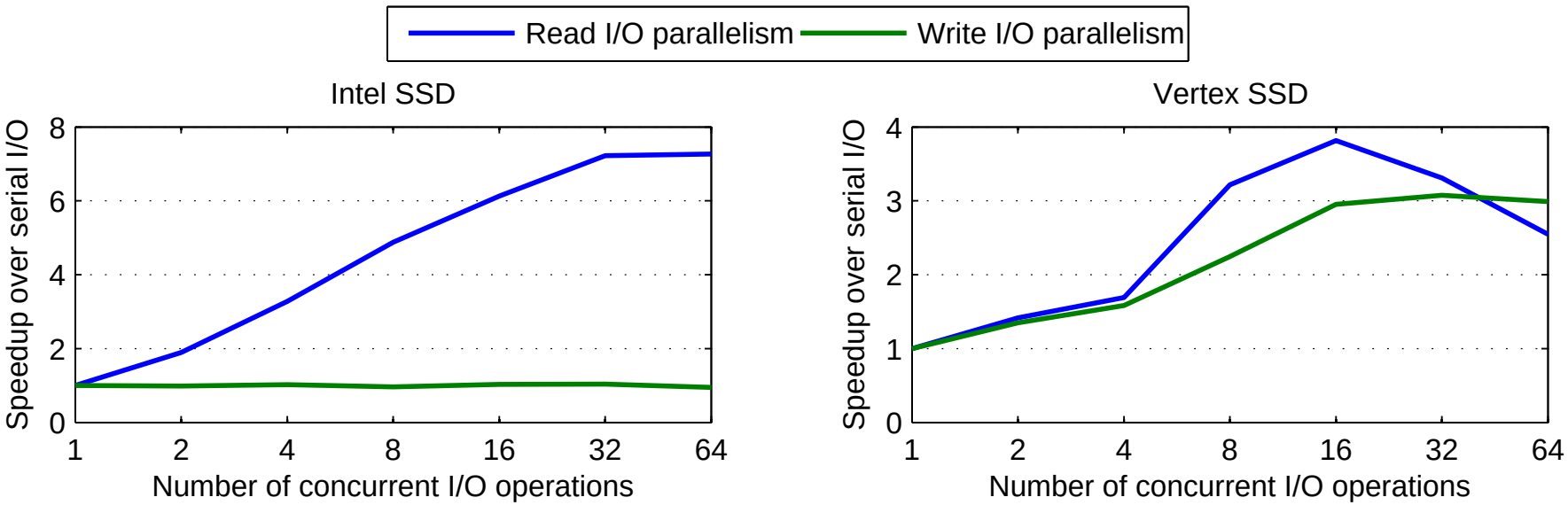


Motivation: Read-Write Interference



Fast read response is disrupted by interfering writes.

Motivation: Parallelism

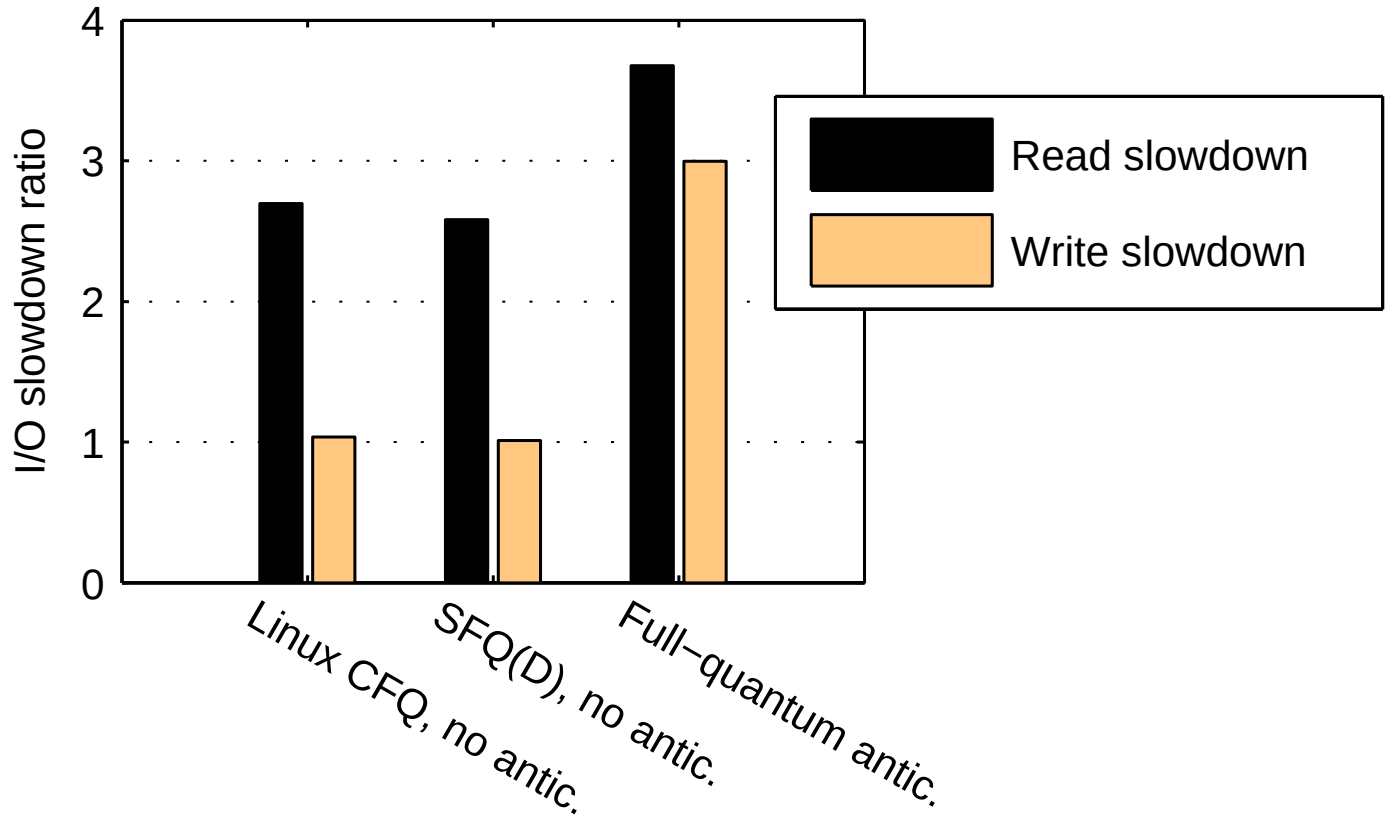


SSDs can support varying levels of read and write parallelism.

Motivation: I/O Anticipation Support

- Reduces potential seek cost for mechanical disks
- ...but largely negative performance effect on Flash
- Flash has no seek latency: no need for anticipation?
- No anticipation can result in unfairness: premature service change, read-write interference

Motivation: I/O Anticipation Support



Lack of anticipation can lead to unfairness; aggressive anticipation makes fairness costly.

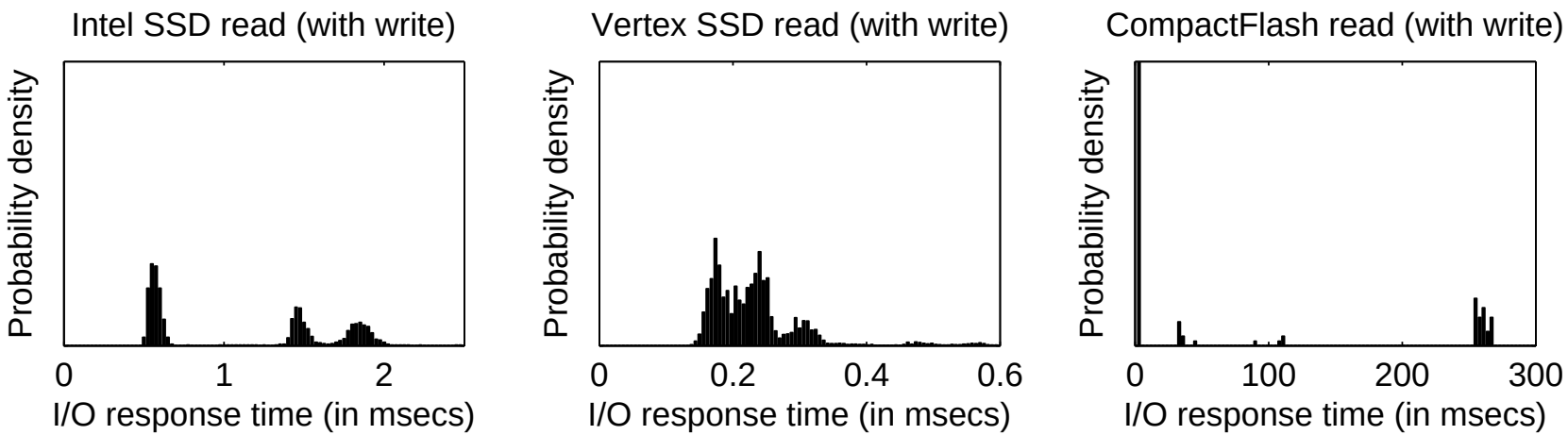
FIOS: Policy

- Fair timeslice management: Basis of fairness
- Read-write interference management: Account for Flash I/O asymmetry
- I/O parallelism: Recognize and exploit SSD internal parallelism while maintaining fairness
- I/O anticipation: Prevent disruption to fairness mechanisms

FIOS: Timeslice Management

- Equal timeslices: amount of time to access device
  - Non-contiguous usage
  - Multiple tasks can be serviced simultaneously
- Collection of timeslices = epoch; Epoch ends when:
  - No task with a remaining timeslice issues a request, or
  - No task has a remaining timeslice

FIOS: Interference Management



- Reads are faster than writes $\rightarrow$ interference penalizes reads more
- Preference for servicing reads
- Delay writes until reads complete

FIOS: I/O Parallelism

- SSDs utilize multiple independent channels
- Exploit internal parallelism when possible, minding timeslice and interference management
- Parallel cost accounting: New problem in Flash scheduling
  - Linear cost model, using time to service a given request size
  - Probabilistic fair sharing: Share perceived device time usage among concurrent users/tasks

    $\text{Cost} = \frac{T_{elapsed}}{P_{issuance}}$

    $T_{elapsed}$ is the requests elapsed time from its issuance to its completion, and $P_{issuance}$ is the number of outstanding requests (including the new request) at the issuance time

FIOS: I/O Anticipation - When to anticipate?

Anticipatory I/O originally used for improving performance on disk to handle deceptive idleness: wait for a desirable request.
Anticipatory I/O on Flash used to preserve fairness.

Deceptive idleness may break:

- timeslice management
- interference management

FIOS: I/O Anticipation - How long to anticipate?

- Must be much shorter than the typical idle period for disks
- Relative anticipation cost is bounded by α, where idle period is $T_{service} * \frac{\alpha}{1-\alpha}$ where $T_{service}$ is per-task exponentially-weighted moving average of per-request service time

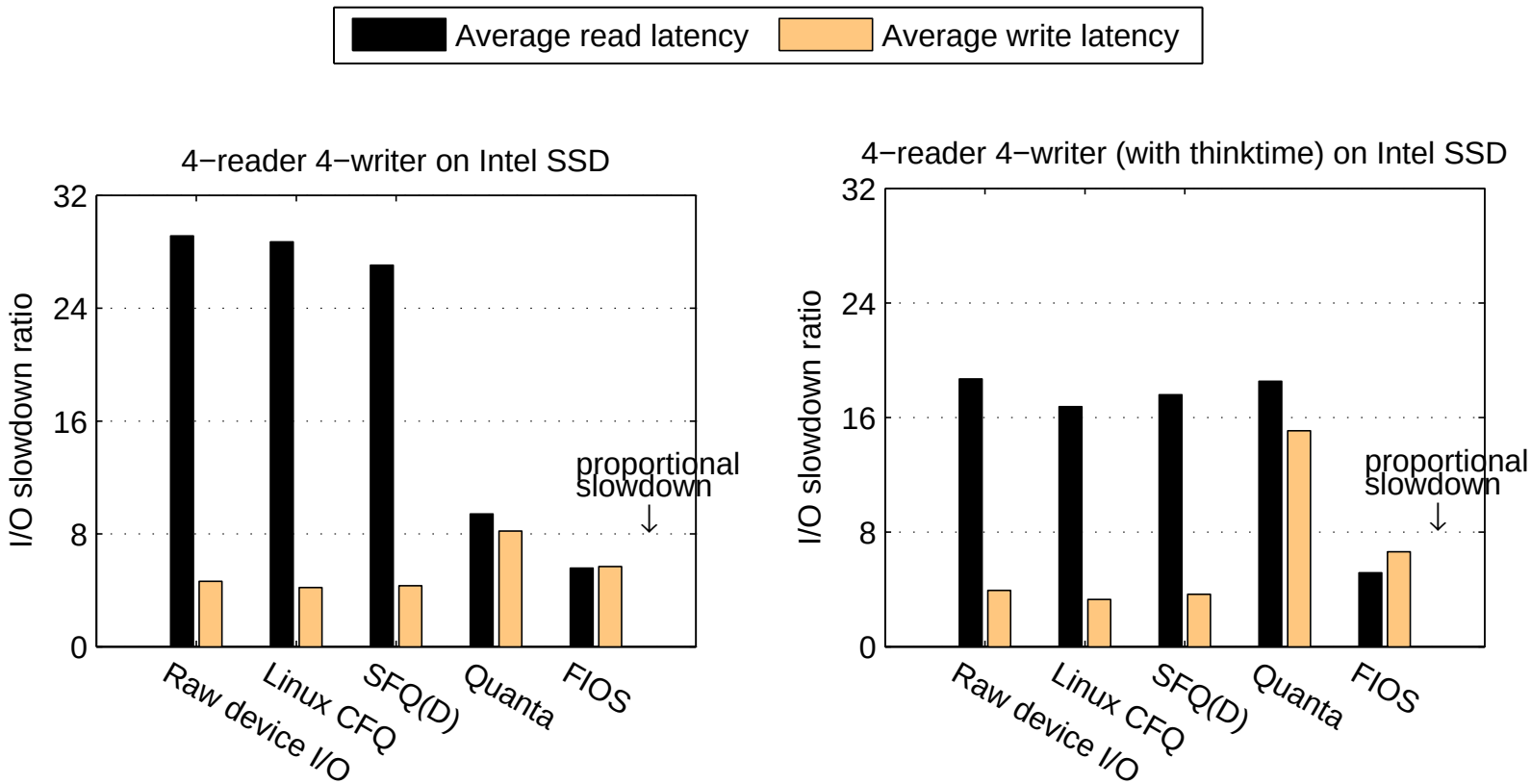
  (Default $\alpha = 0.5$)

  ex. I/O $\rightarrow$ anticipation $\rightarrow$ I/O $\rightarrow$ anticipation $\rightarrow$ I/O $\rightarrow \cdots$

Implementation Issues

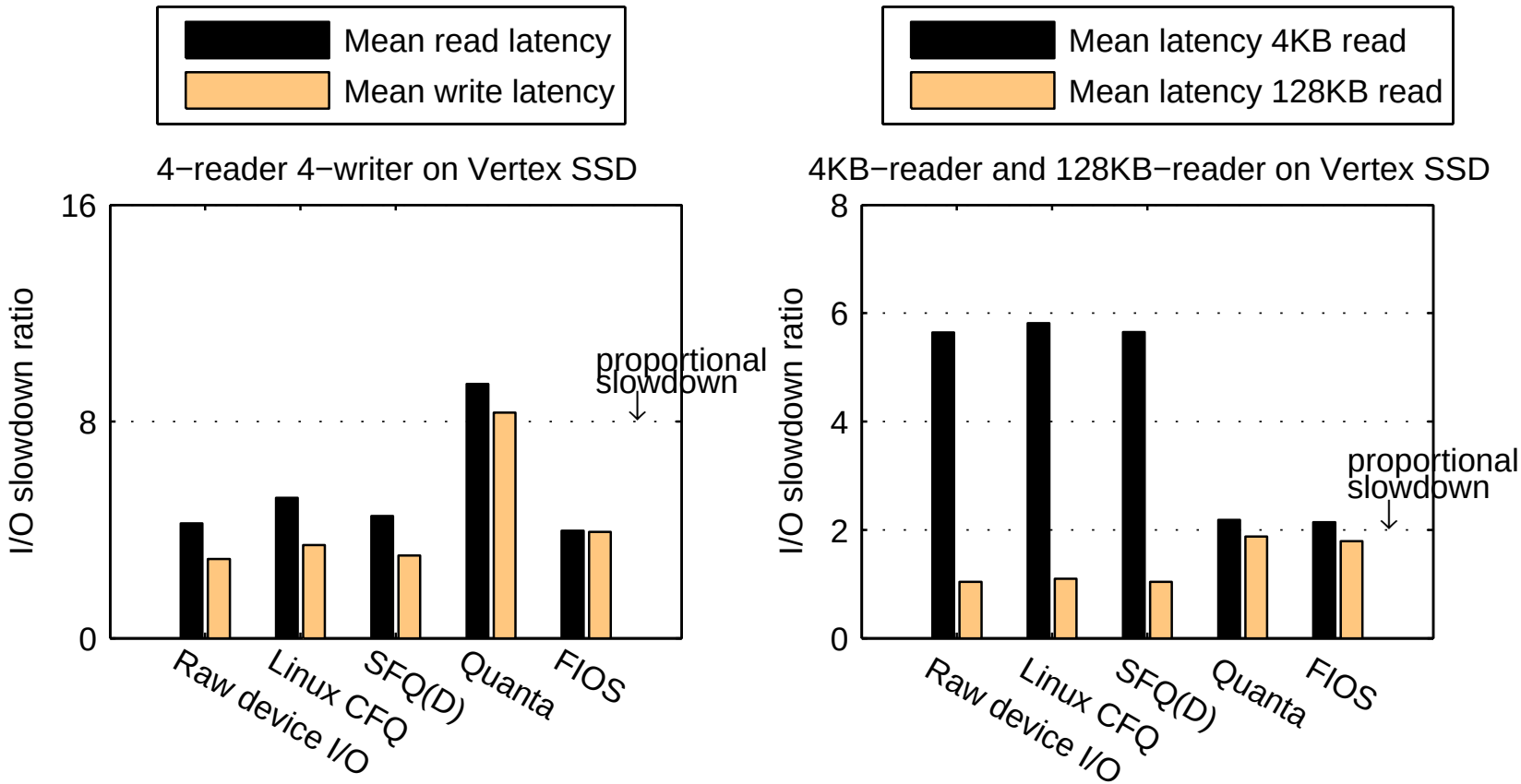
- Linux coarse tick timer $\rightarrow$ High resolution timer for I/O anticipation
- Inconsistent synchronous write handling across file system and I/O layers
- ext4 nanosecond timestamps lead to excessive metadata updates for write-intensive applications

Results: Read-Write Fairness



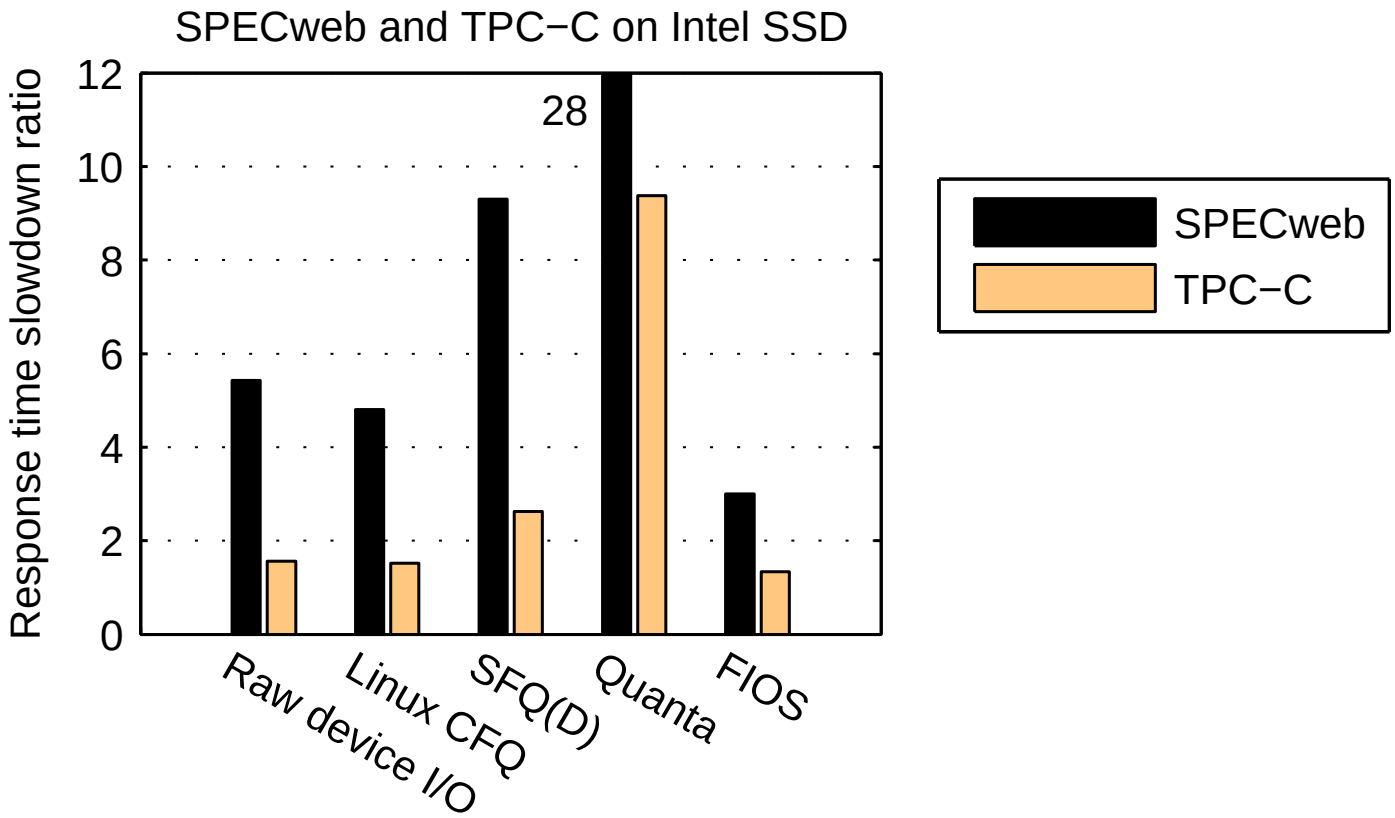
Only FIOS provides fairness with good efficiency under differing I/O load conditions.

Results: Beyond Read-Write Fairness



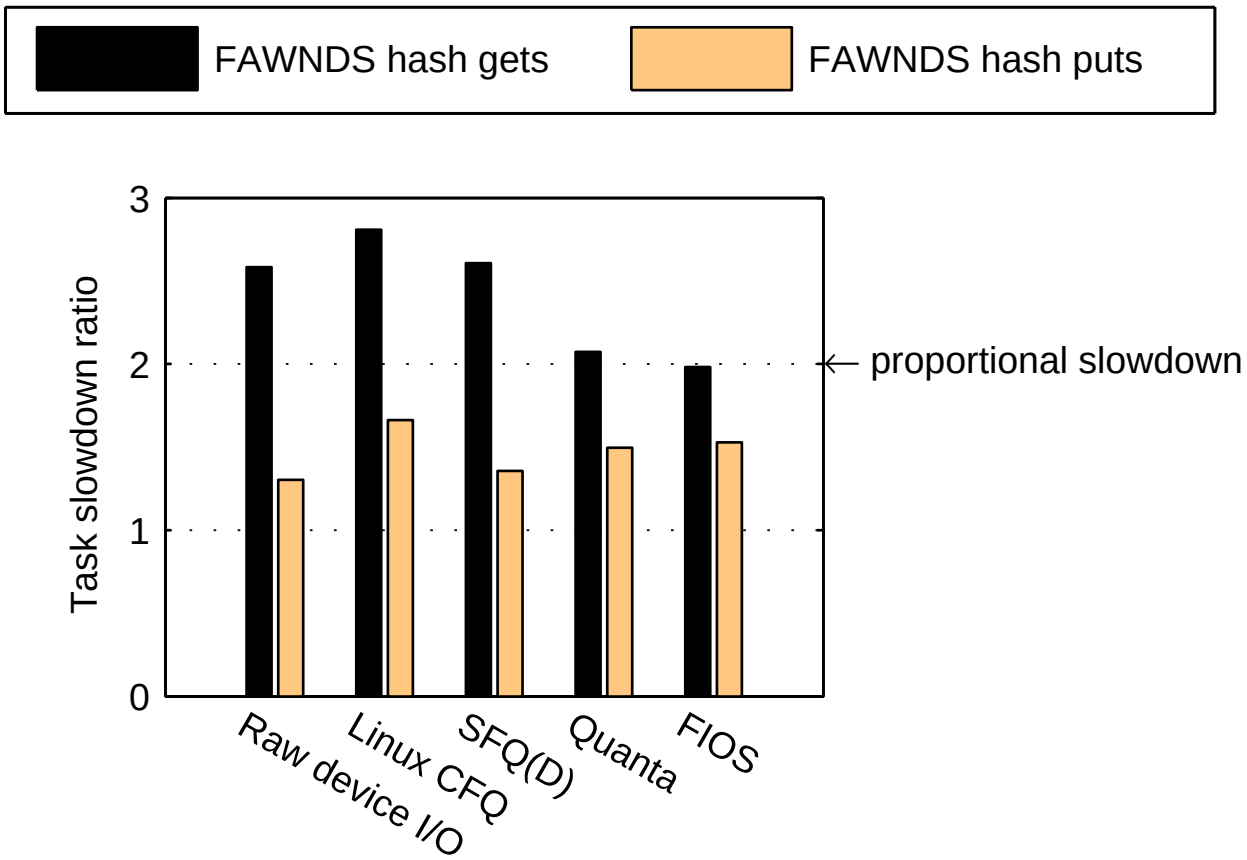
FIOS achieves fairness not only with read-write asymmetry but also requests of varying cost.

Results: SPECweb co-run TPC-C



FIOS exhibits the best fairness compared to the alternatives.

Results: FAWNDS (CMU, SOSP'09) on CompactFlash



FIOS also applies to low-power Flash and provides efficient fairness.

Conclusion

- Fairness and efficiency in Flash I/O scheduling
  - Fairness is a primary concern
  - New challenge for fairness AND high efficiency (parallelism)
- I/O anticipation is ALSO important for fairness
- I/O scheduler support must be robust in the face of varied performance and evolving hardware
  - Read/write fairness and BEYOND
- May support other resource principals (VMs in cloud).