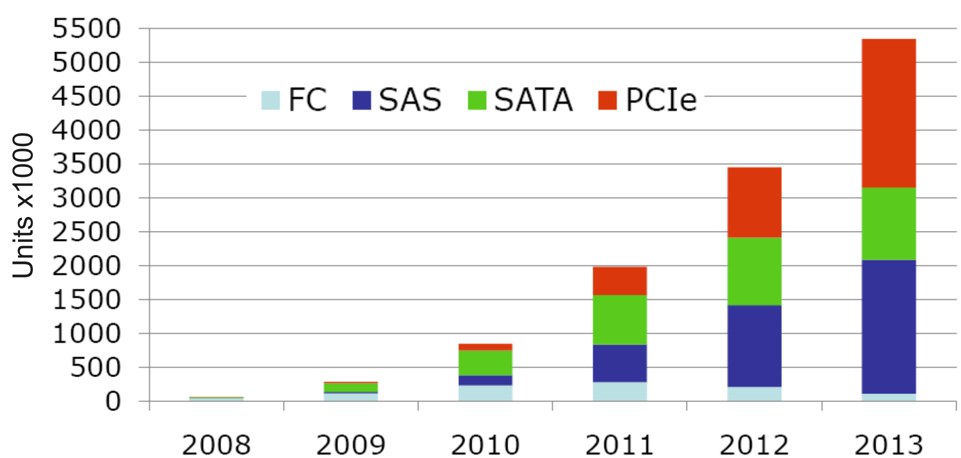
High Performance Solid State Storage Under Linux

Eric Seppanen, Matthew T. O'Keefe, David J. Lilja
Electrical and Computer Engineering
University of Minnesota

April 20, 2010

Motivation

- SSDs breaking through into new applications
- Why care about another SSD?
  - Faster! 1M IOPS, GBps+



Source: Preliminary Gartner estimates, 12/09

Contents

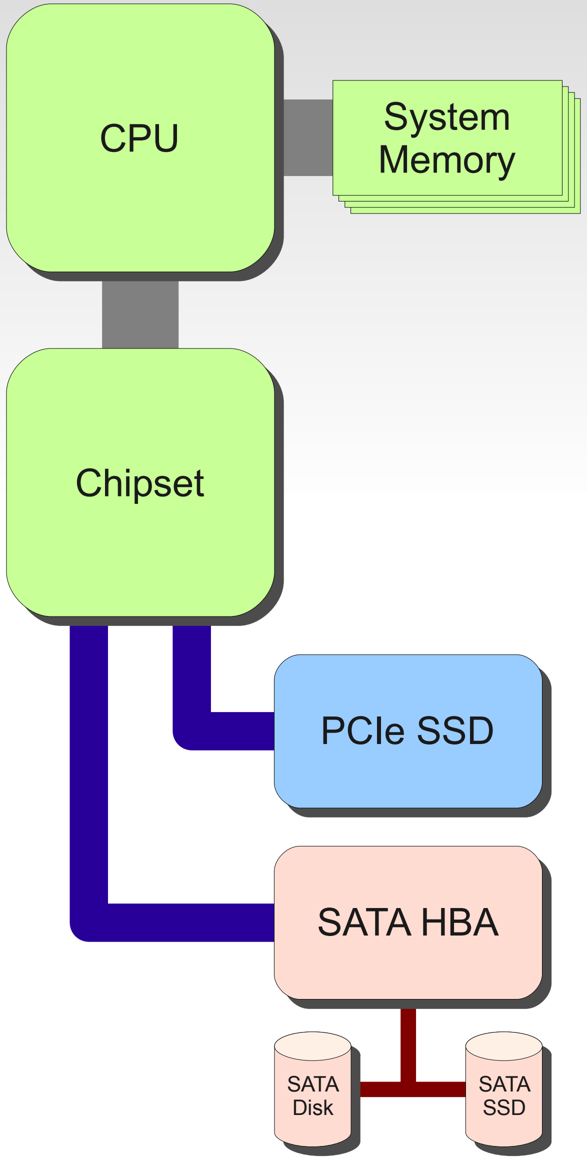
What's New:
PCIe SSD.
Performance.

SSD Benchmarking:
Goals, observations,
and pitfalls.

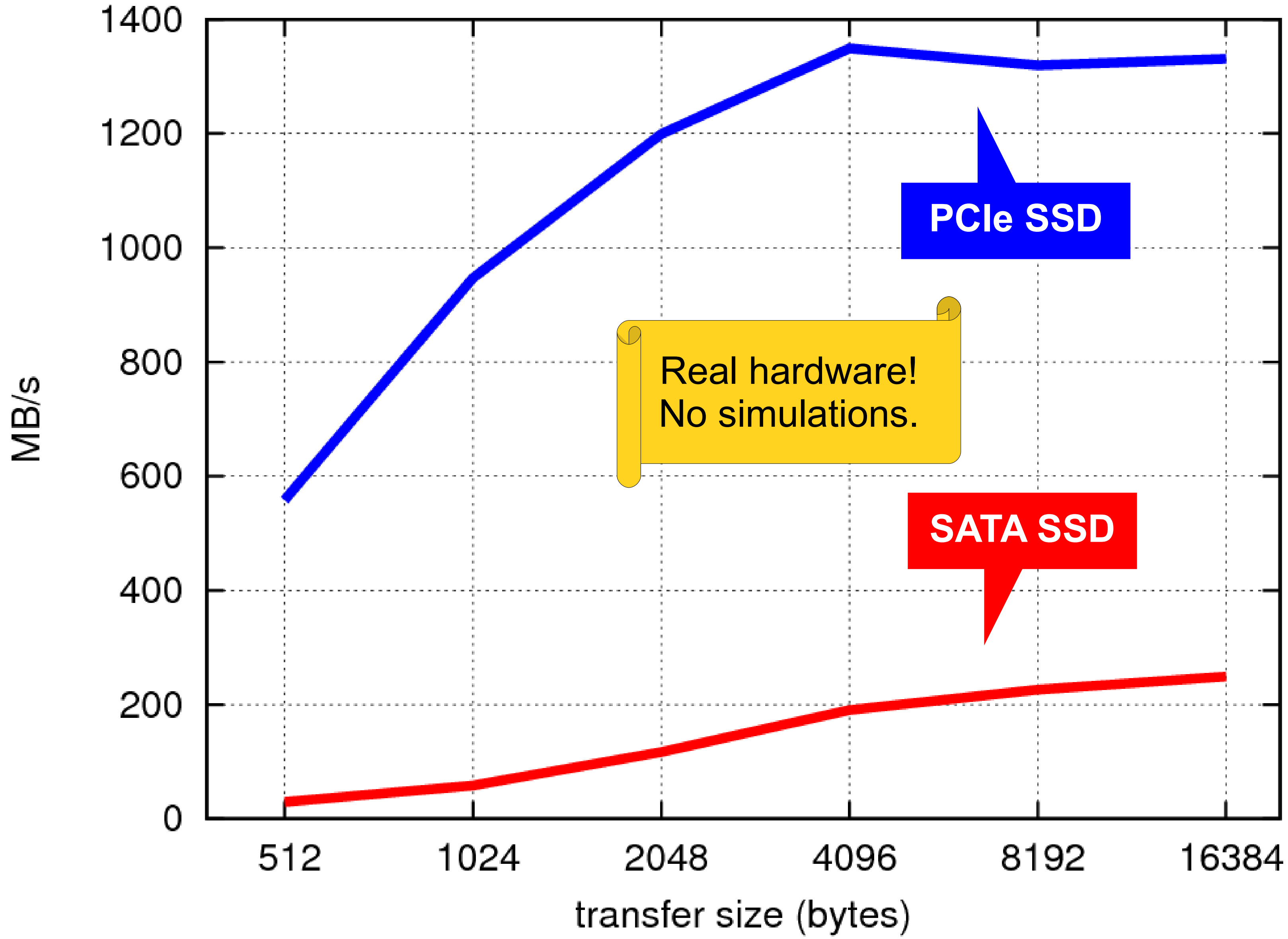
Fast SSDs and Linux:
How to go fast?
What needs changes.

Solid State Drive Hardware

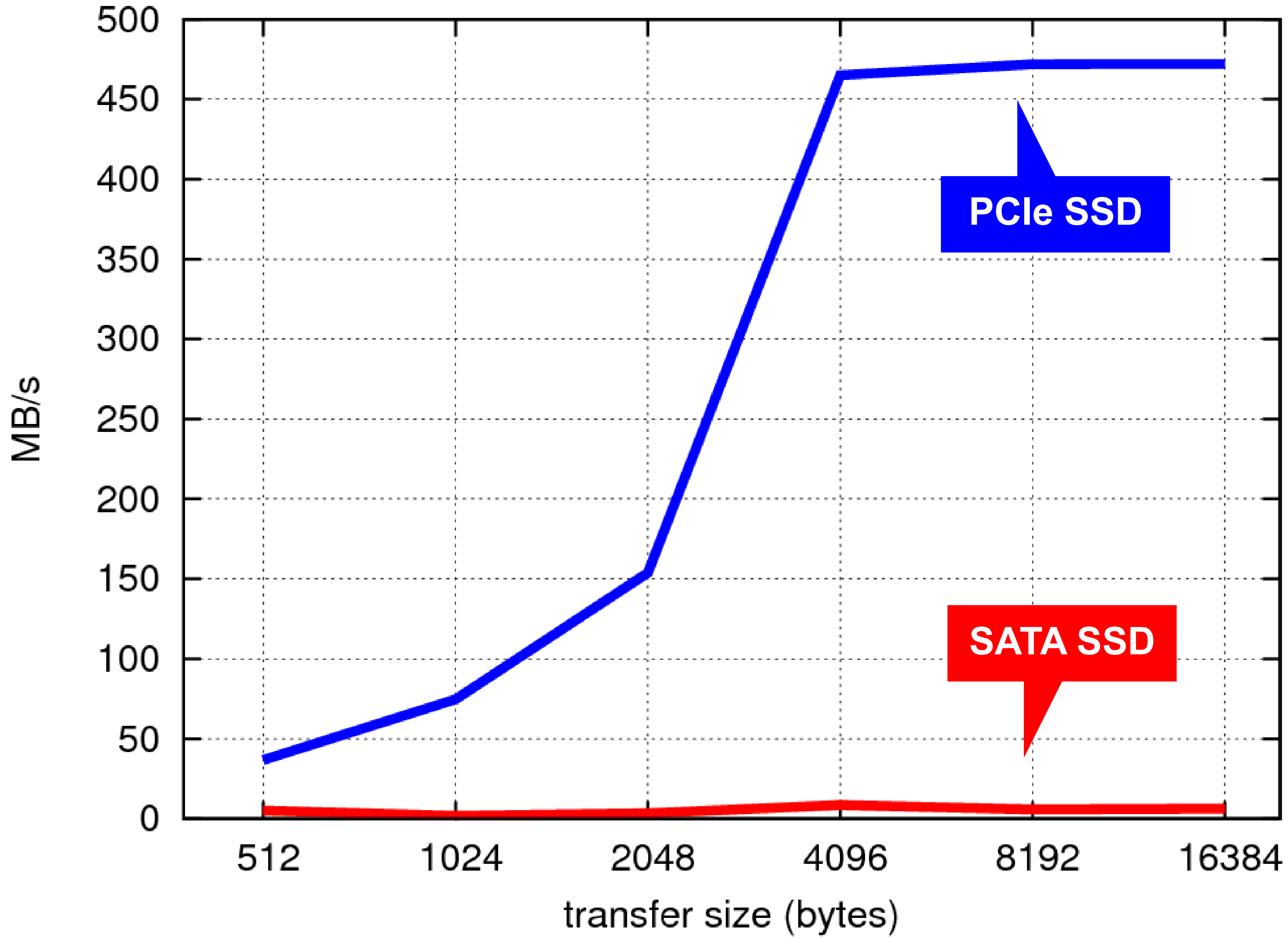
- Micron PCIe SSD prototype
  - NAND flash storage
  - PCI Express 1.0 x8
  - Onboard flash management
  - AHCI compatible +
  - Deep command queue



Random Read Throughput

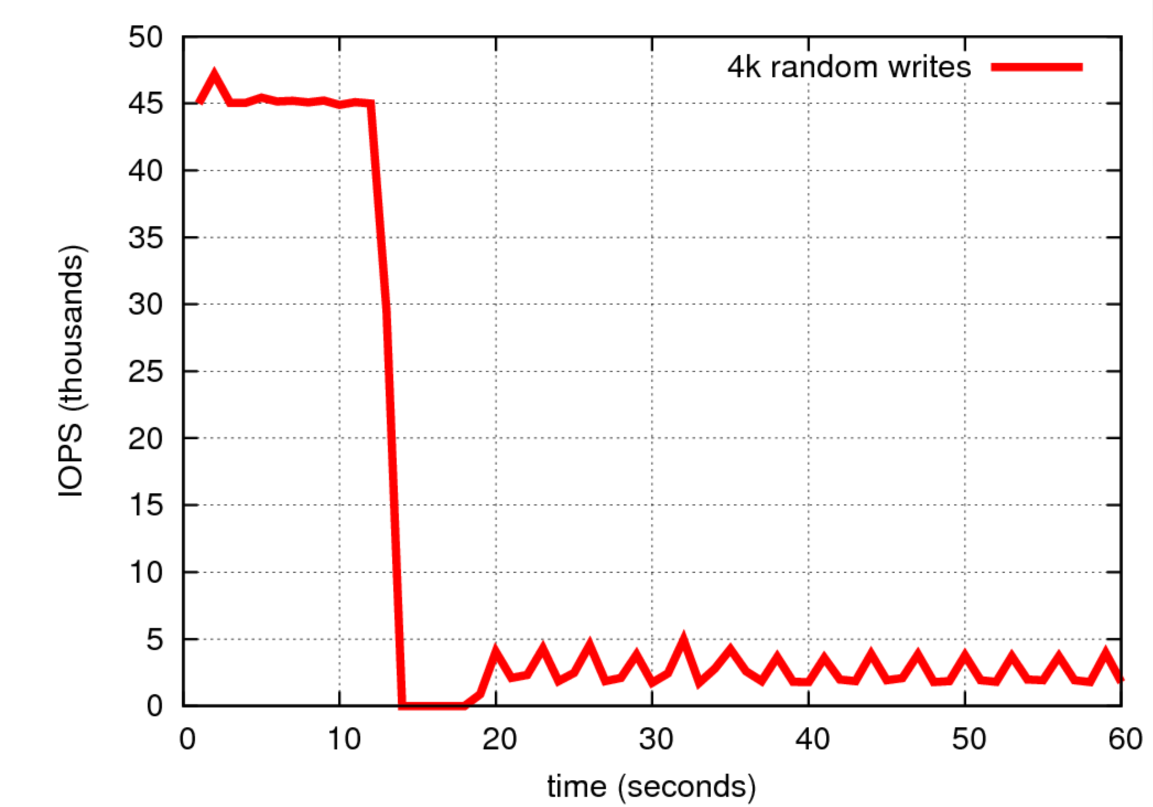


Random Write Throughput



Part 2: SSD Benchmarking

or, How Not to Fall Off a Cliff:



Measuring SSD Performance

- Be skeptical: use realistic but difficult workload.
  - Test areas where SSDs perform poorly
- Don't measure an SSD like a disk.
  - Account for new SSD performance factors
- Include parallelism in benchmark

(beware of Bonnie, IOZone, traces)

SSDs and Parallelism

- Disk hardware only capable of reading/writing one location at a time; SSDs can be reading/writing many places at once.
- Disks hold multiple requests in a queue: i.e. SATA disks have a 32-command queue.
- SSDs can process requests in parallel; we'll still call it a queue, but it's not used as one.

SSD vs Disk

- Disks are well-understood: seek time + rotational latency.
- SSDs have many more factors:
  - Native block size
  - Overprovisioning / empty space
  - FTL tasks can be nondeterministic
- The difference between maximum and minimum performance can be huge.

SSD performance over time

	Disk	SSD
Earlier I/O displaces drive head (ms)	✓	
Rotating platter causes latency (<1ms)	✓	
Earlier I/O ties up buses and flash planes (<1ms)		✓
Earlier I/O causes Garbage Collection tasks to run (~15 sec?)		✓
Earlier I/O patterns caused data fragmentation (weeks?)		✓
Earlier I/O usage consumed empty space (months?)		✓

Simplified Pessimistic Benchmark

- Test under difficult conditions:
    - Use lots of parallel I/O
    - Use random I/O
    - Perform small transfers
    - Fill the drive
    - Measure steady-state performance

Part 3: The Linux Kernel

or, How to Reach a Million IOPS

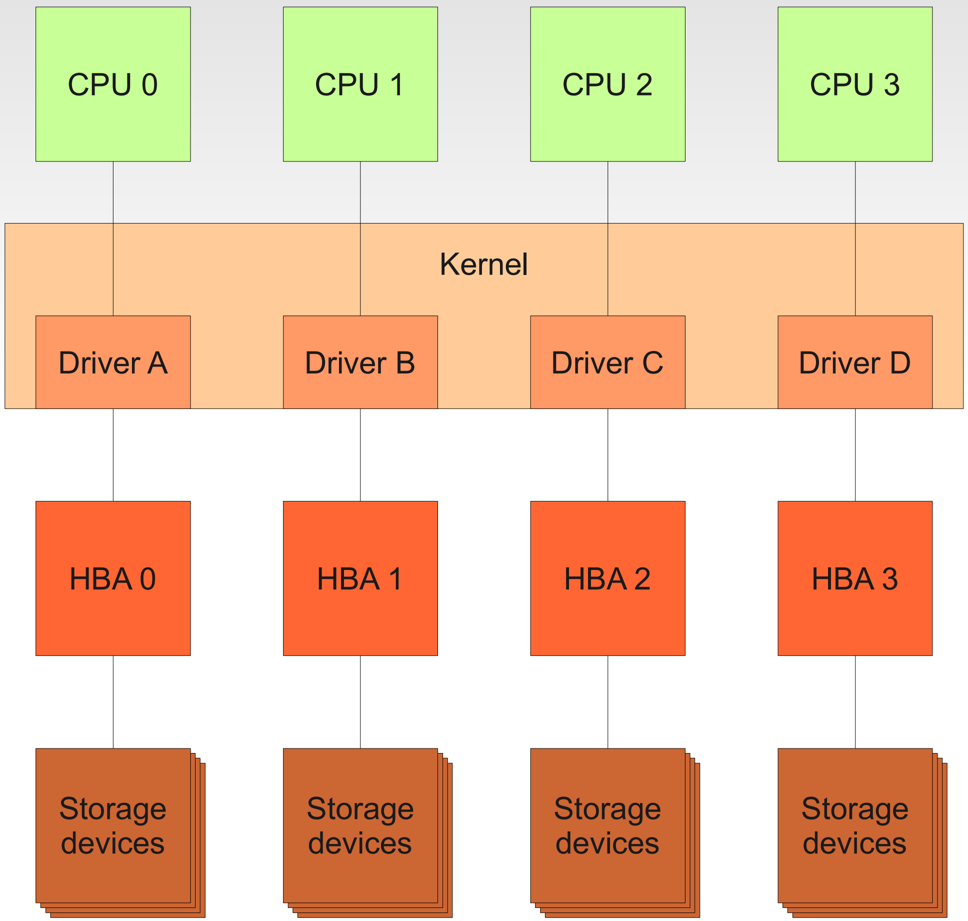
Extracting Best Performance

- CPU to device relationship has changed.
- Allow everything in parallel.
- Kernel I/O layers add significant overhead.
- Interrupt management becomes very important.

CPUs and devices

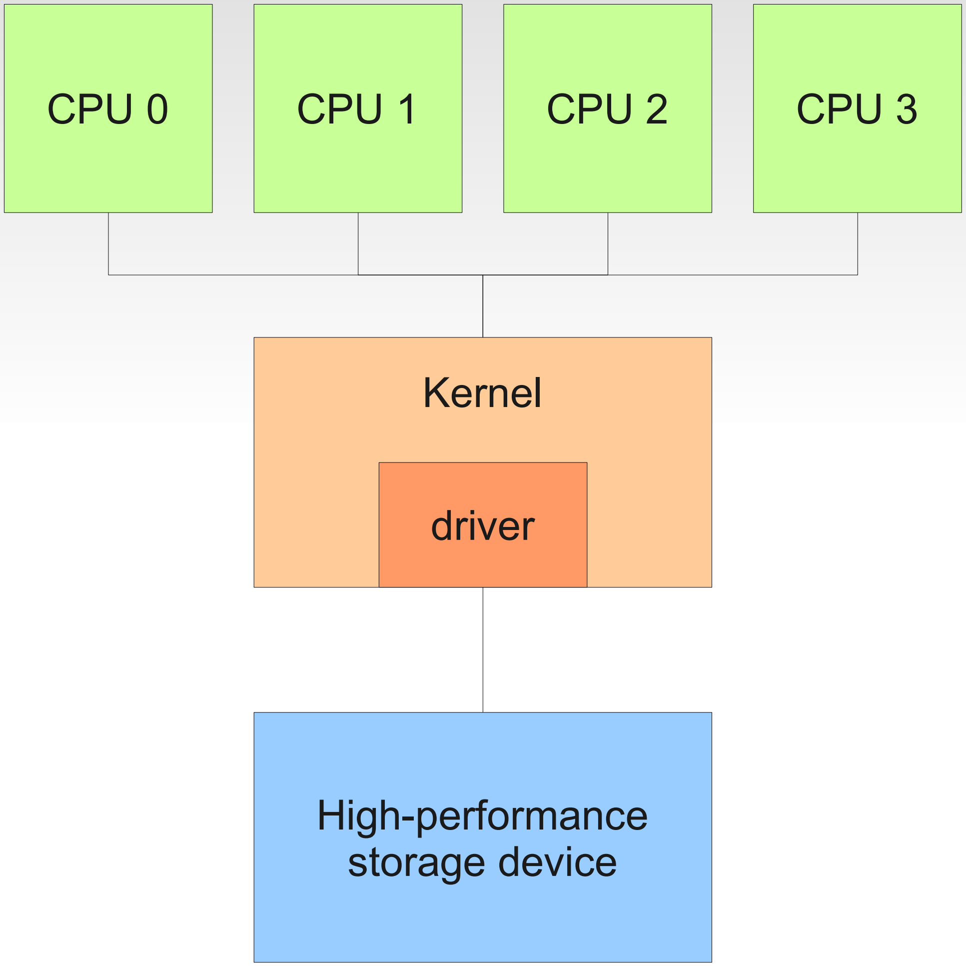
- New problem: the number of operations handled by one device is so high, it can't be managed by a single CPU core.
- If multiple CPU cores are needed, this affects the architecture of the device and interface software.

Aggregation of Slow Devices



- Achieve high throughput and parallelism by adding more devices.
- It's possible to manage I/O submit/retire with a single CPU.

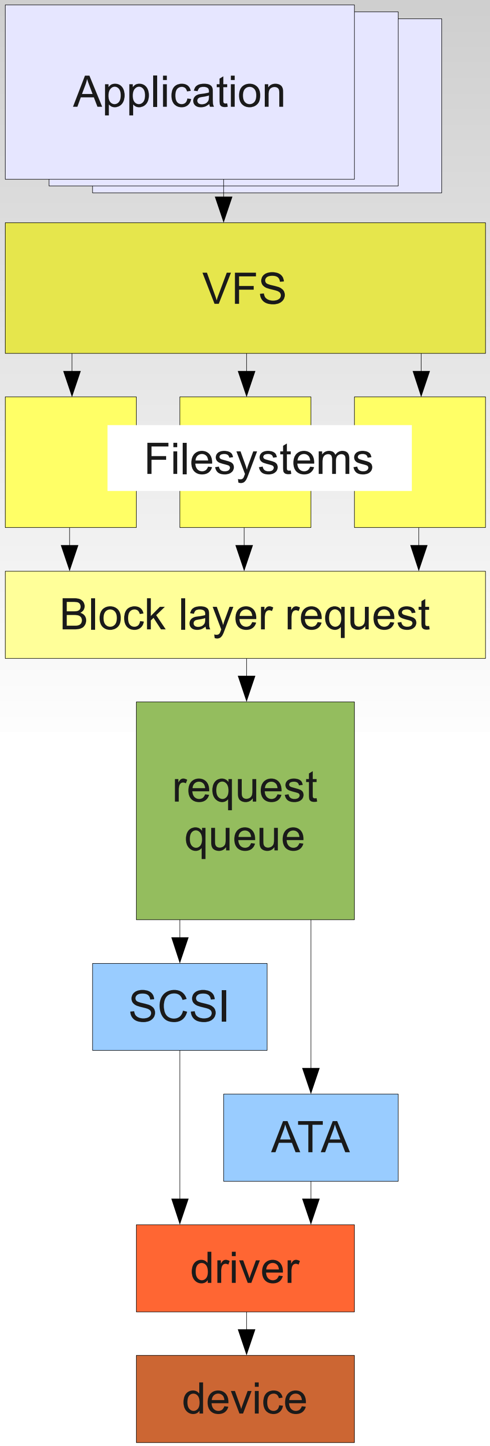
Consolidation Onto Fast Device



- Kernel, driver and device must perform parallel operations efficiently.
- Must be designed to interface with many CPUs.

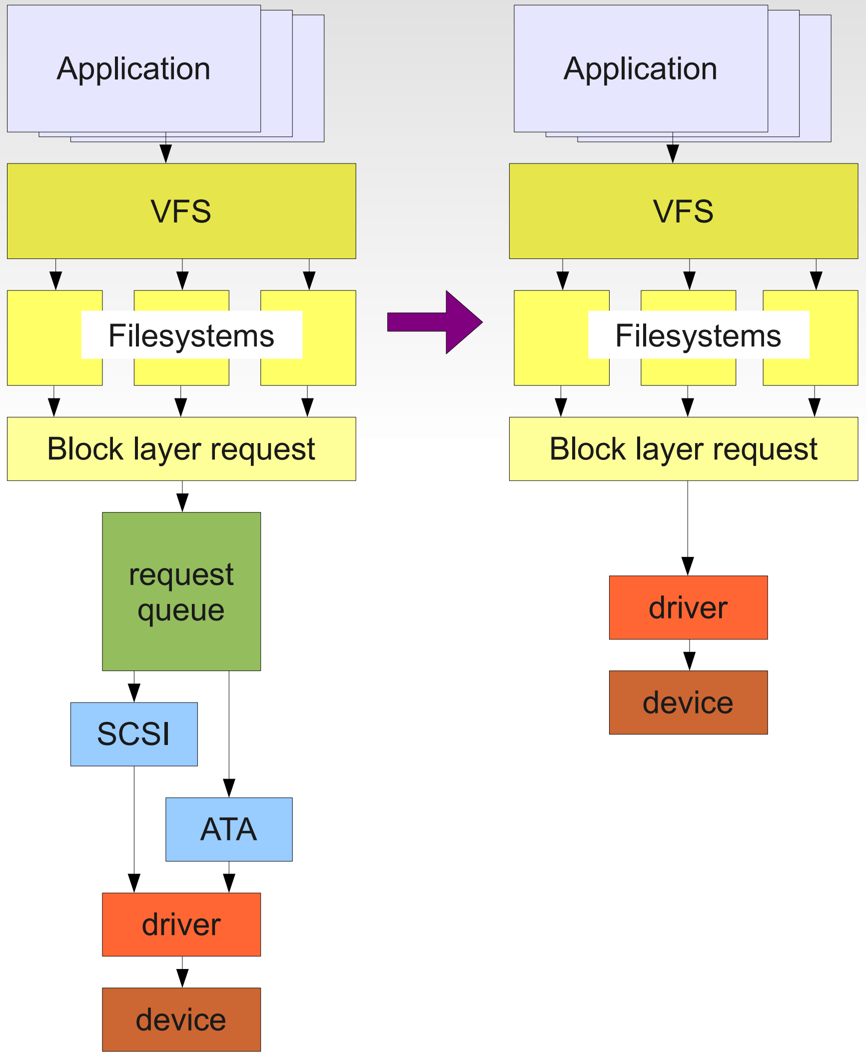
Linux I/O Architecture

- Linux I/O subsystem has layers that add latency and limits parallelism.
- Try bypassing layers to find performance bottlenecks.



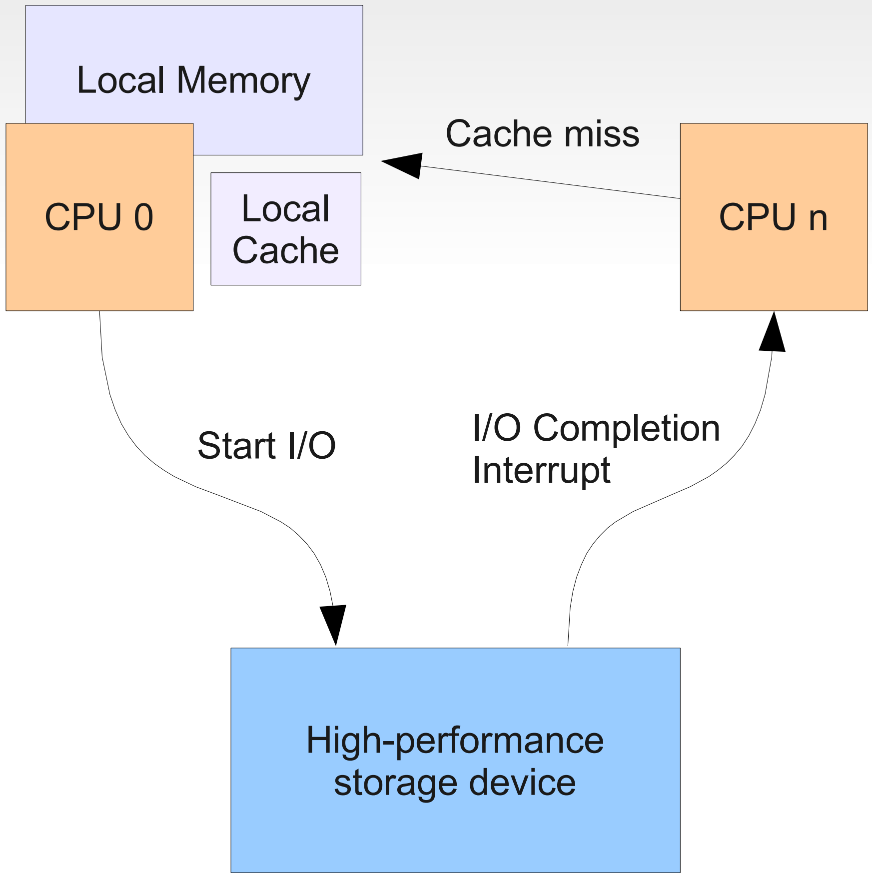
Linux I/O Architecture, continued

- Bypass SCSI, ATA layers to reduce CPU overhead
- Bypass request queue layers
  - Reduce CPU overhead
  - Get rid of disk-oriented optimizations
  - Skip locking that hurts parallelism



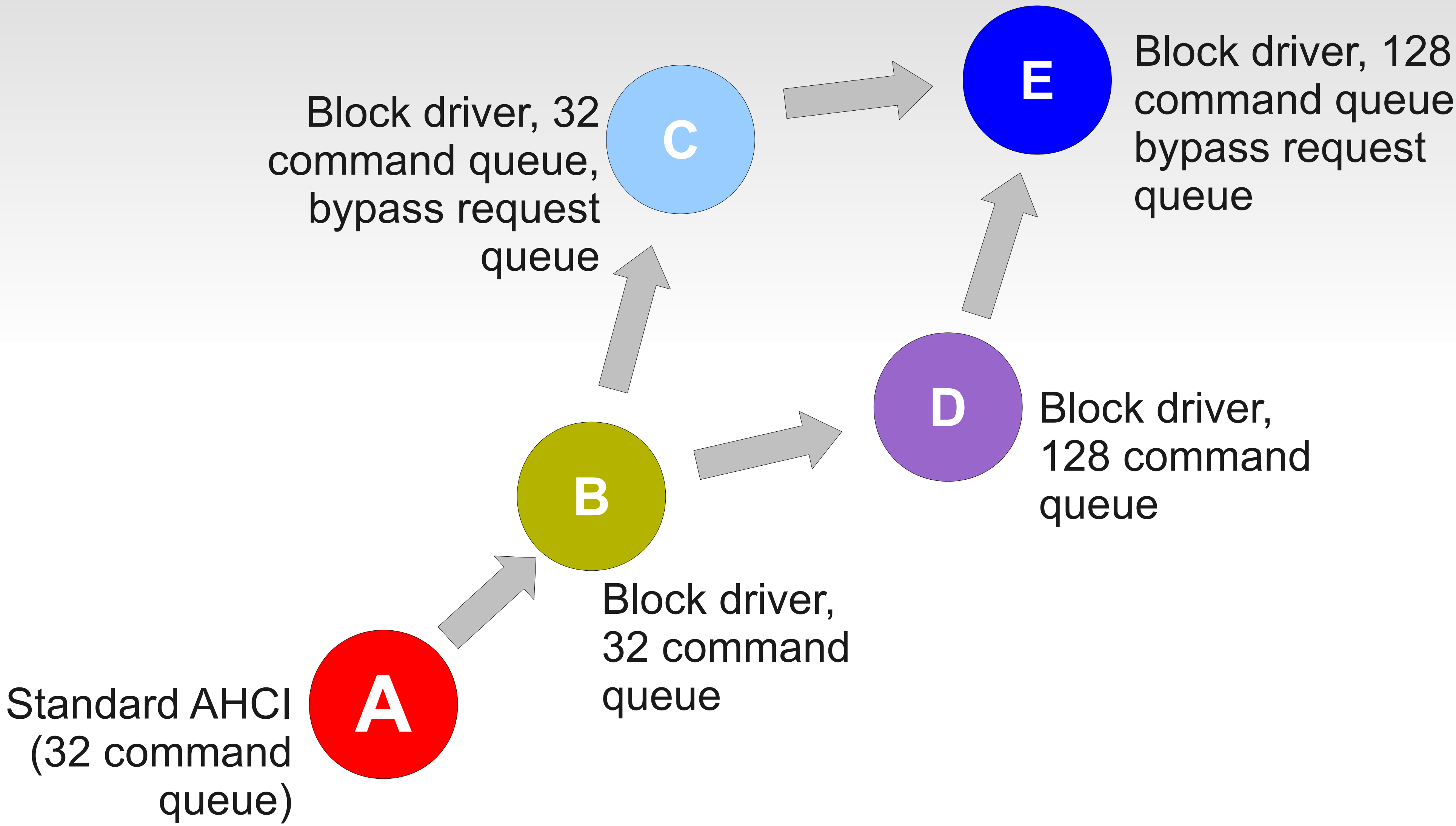
Interrupt Management

- Problem 1: Interrupt load can overwhelm a single CPU.
- Solution: spread interrupt load over multiple CPUs.

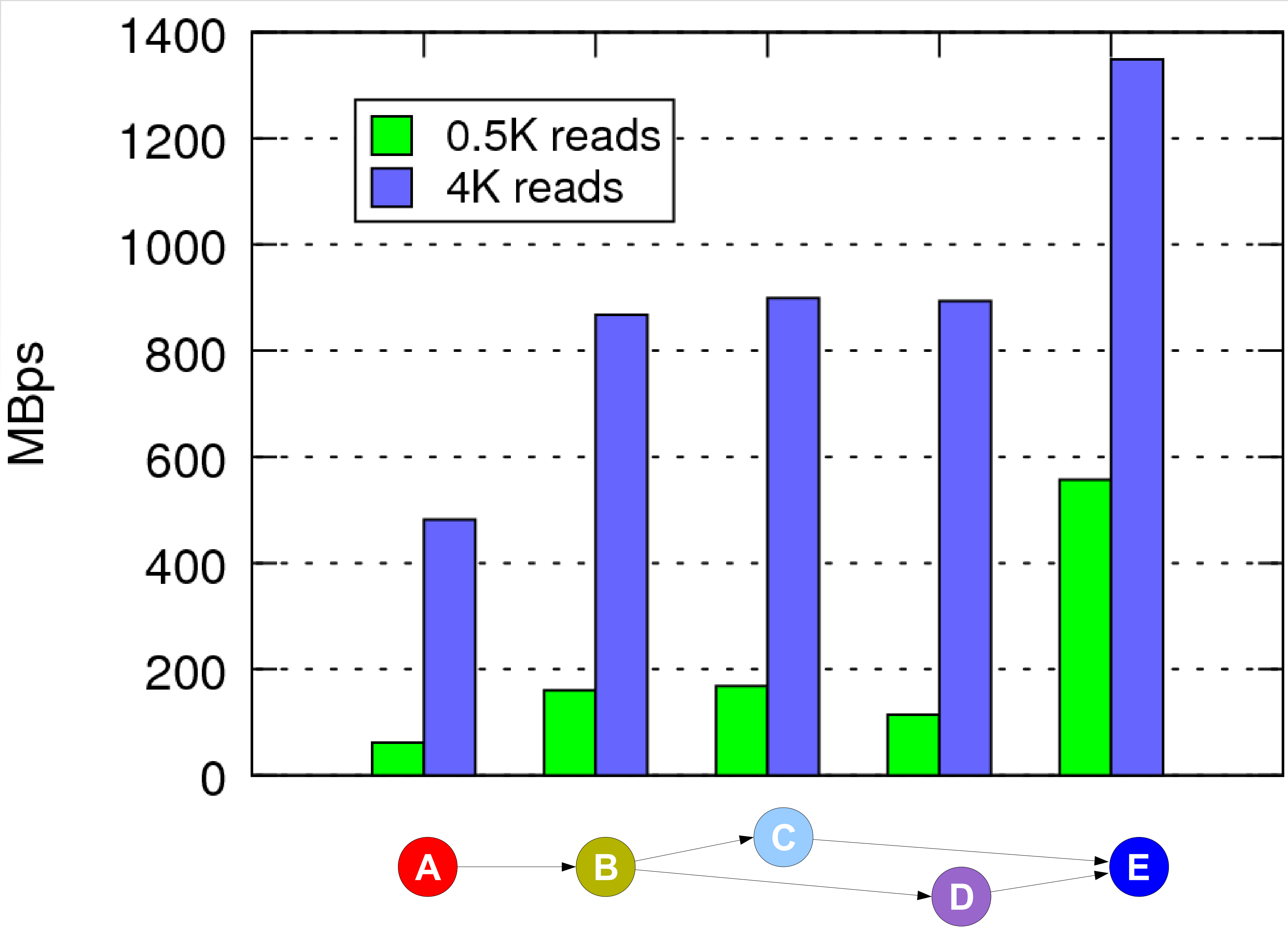


- Problem 2: Interrupts sent to a distant CPU can cause cache miss slowdowns.
- Solution: Redirect interrupt to nearby CPU if possible.

Driver Evolution



Driver Evolution



Conclusions

- High-performance SSDs can deliver significantly higher performance than commodity SSDs.
- Careful benchmarking is important to reveal worst-case performance.
- SSDs use parallelism to reach maximum performance.
- Linux kernel and driver improvements may be necessary to get best results.

End of Presentation

Thank you