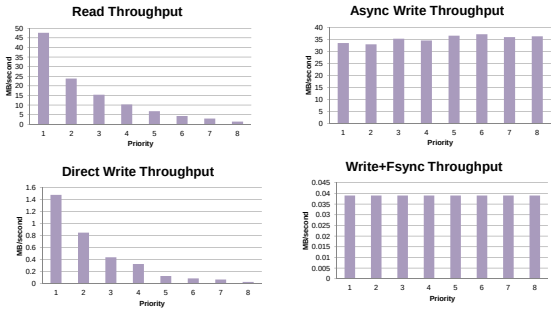
Split-Level I/O Scheduler

Suli Yang, Tyler Harter, Anand Krishnamurphy, Salini Kowsalya, Samer Al-Kiswany, Andrea C. Aarpaci-Dusseau, Remzi H. Arparci-Dusseau

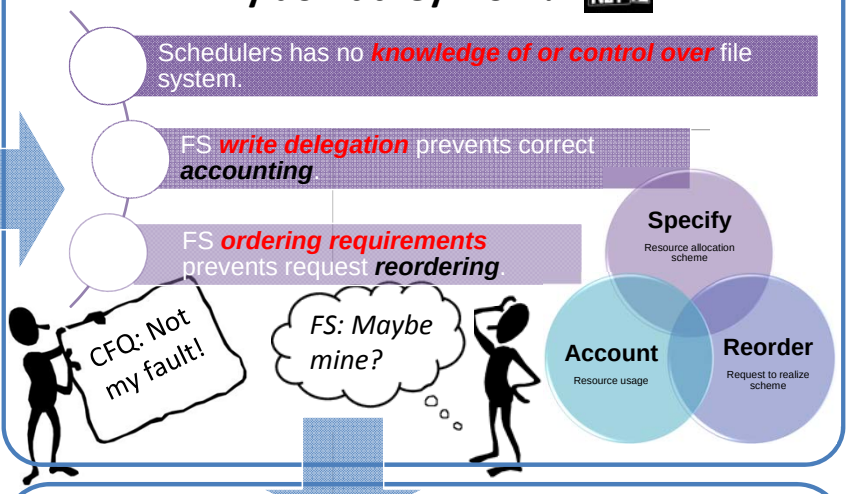
Why Not Block-level Scheduler?



Because they don't work!
(At least for writes…)

Why don't they work?

- Schedulers has no ***knowledge of or control over*** file system.
- FS ***write delegation*** prevents correct ***accounting***.
- FS ***ordering requirements*** prevents request ***reordering***

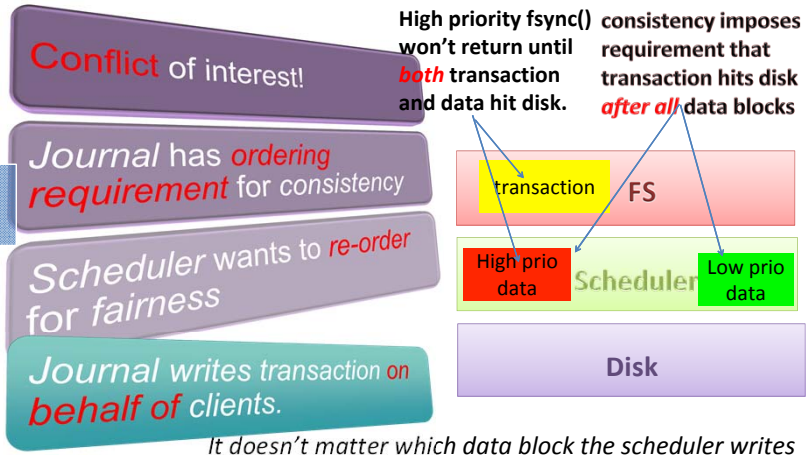


Other Problematic File System Features (for ext4 and others…)

	Accounting	Ordering
Journaling	bad	bad
Shared Metadata	bad	bad
Write Buffering	bad	neutral
Delayed Allocation	bad	good

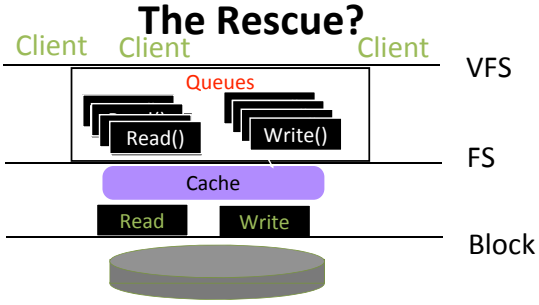
- Almost all file systems use ***ordering requirements*** to ensure crash consistency.
- ***Write delegation*** everywhere: delaying work makes it necessary.
- Write delegation and ordering requirements are ***universal*** file system properties.
- Make block level write scheduling **inherently hard** (if not impossible).

Journal: An Example of FS Being A Pain…



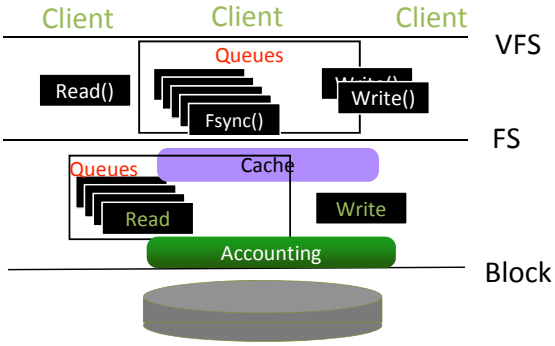
It doesn't matter which data block the scheduler writes to disk first. Priority inversion happens because high-prio fsync depends on low-prio data hits disk.

System Call Scheduling Comes to The Rescue?



- Simple; no file system complexities.
- What if reads/writes can be absorbed by cache?
- No block level info for seek time optimization.
- Not all system calls has the same cost.

Split-Level I/O Scheduler!



- Schedule reads and writes low and fsyncs high.
- Track I/O causes with many-to-many bipartite graph between clients and block requests.
- Low level accounting and optimization based on disk head time for all I/O.