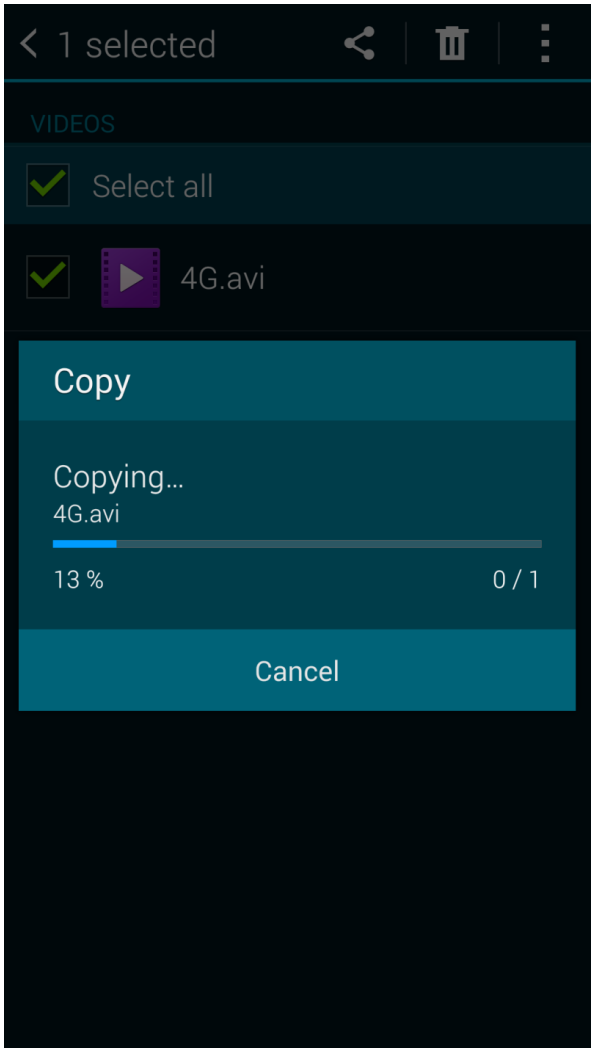
Boosting Quasi-Asynchronous I/Os (QASIOs)

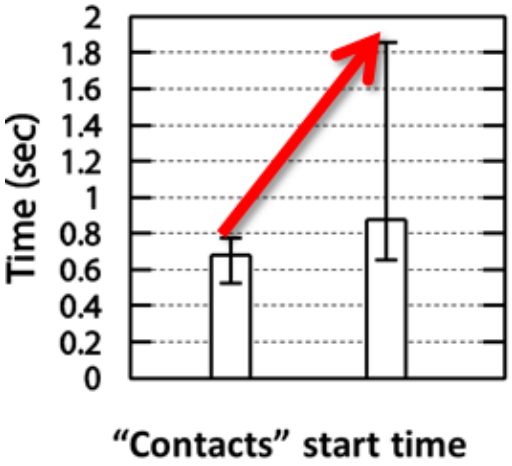
Joint work with
Daeho Jeong and Youngjae Lee

Jin-Soo Kim (*jinsookim@skku.edu*)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

The Problem


1 selected
VIDEOS
Select all
4G.avi
Copy
Copying...
4G.avi
13 %
0 / 1
Cancel


09:41
72%
09:41
Tue, February 10
매탄3동 23°C
08/08 08:38
Google
Email
Camera
Play Store
Google
Phone
Internet
Apps


Time (sec)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
"Contacts" start time

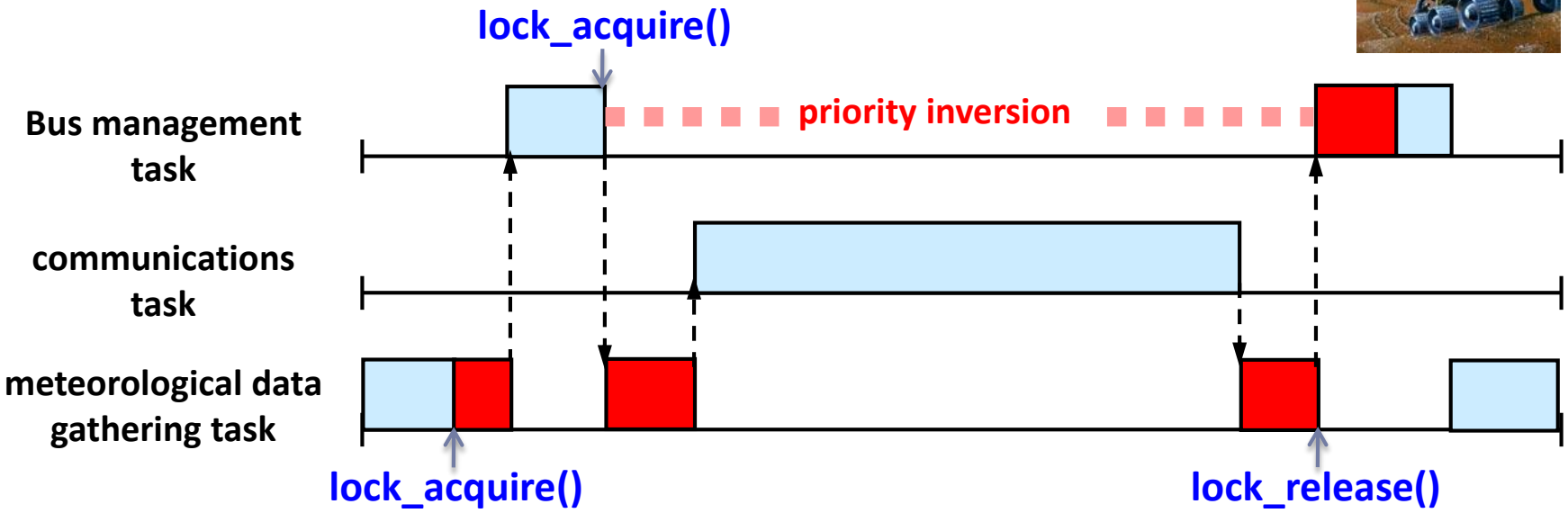
Why?

- CPU? Bus? Power management?
- Metadata contention?
- I/Os already dispatched to the storage?
- No room in memory or request queue?

- The foreground task is waiting for the completion of asynchronous I/Os queued in the I/O scheduler!
  - Over 1 second on an MLC eMMC
  - Over 4 seconds on a TLC eMMC
    (to complete a single file system call)

Priority Inversion

- A situation where a higher-priority job is unable to run because a lower-priority job is holding a resource it needs, such as a lock
- *What really happened on Mars?*



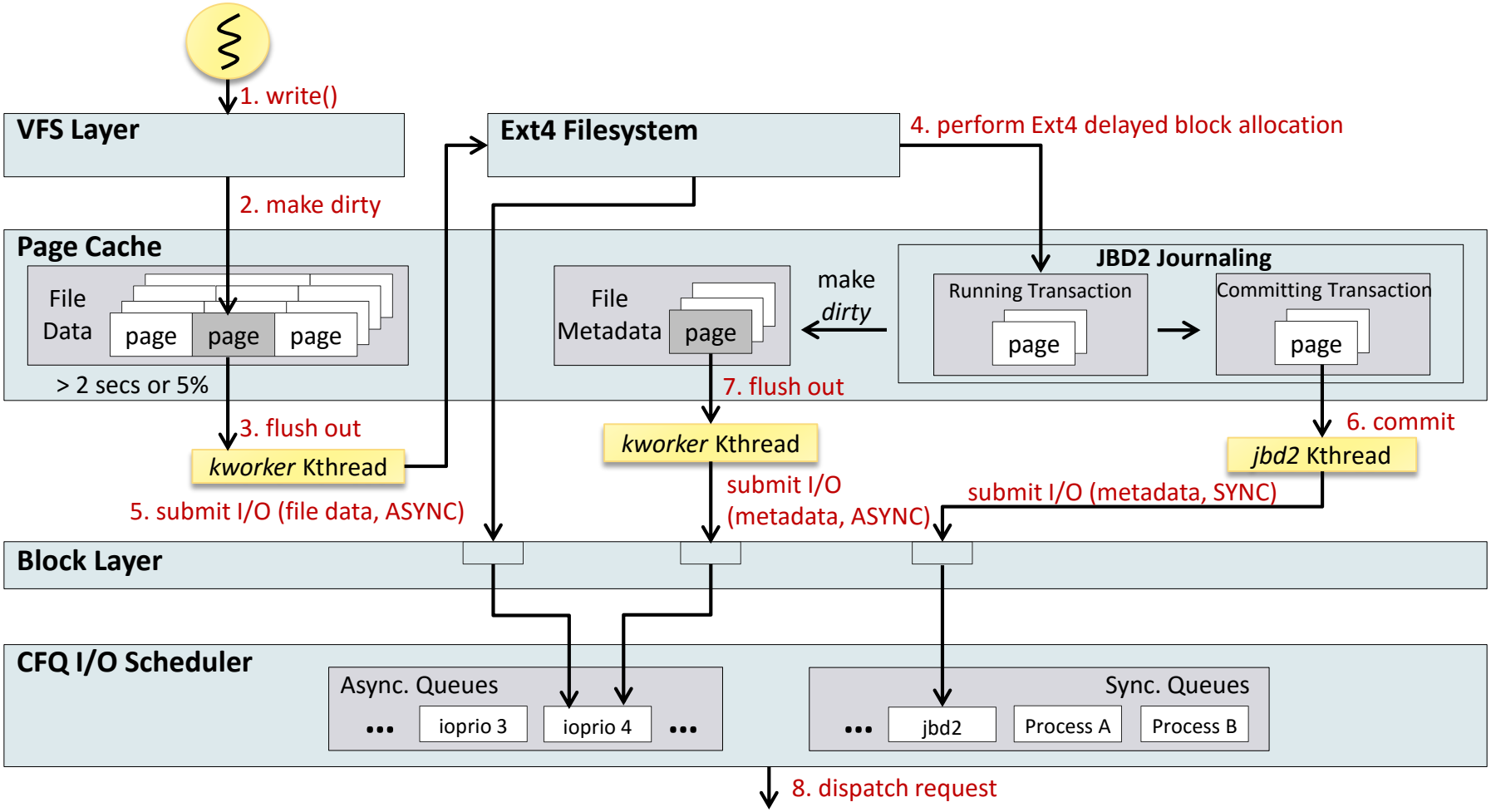
A Closer Look at I/O

- Synchronous I/O
  - read()
  - write() followed by fsync() or sync()
  - write() opened with O_SYNC
  - Journal writes by jbd2
- Asynchronous I/O
  - write()
  - Metadata writes (to their original locations)

CFQ I/O Scheduler

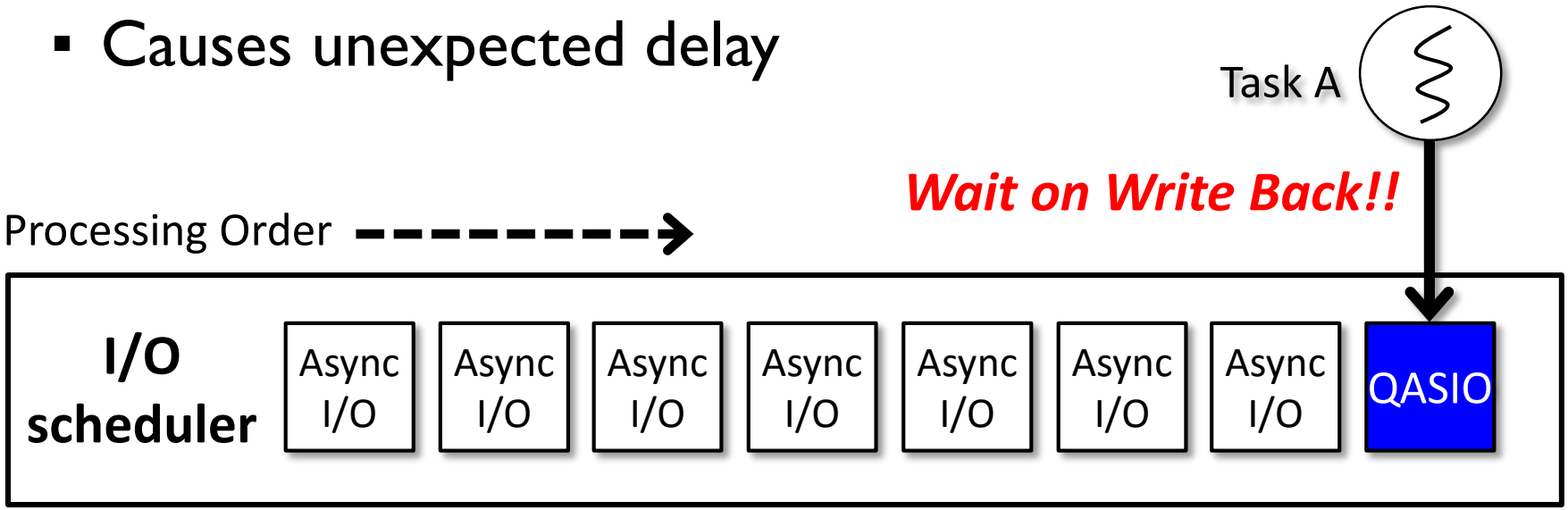
- A Sync queue for each process
- An Async queue is shared among tasks with the same I/O priority
- Sync queues have larger time slices than Async queues
- Sets a limit for Async requests that can be dispatched in a single time slice
- A new Sync request preempts other Async requests
- Time to complete an Async I/O can be prolonged indefinitely
  - Async I/Os are optimized for throughput not for latency

Linux Kernel I/O Path

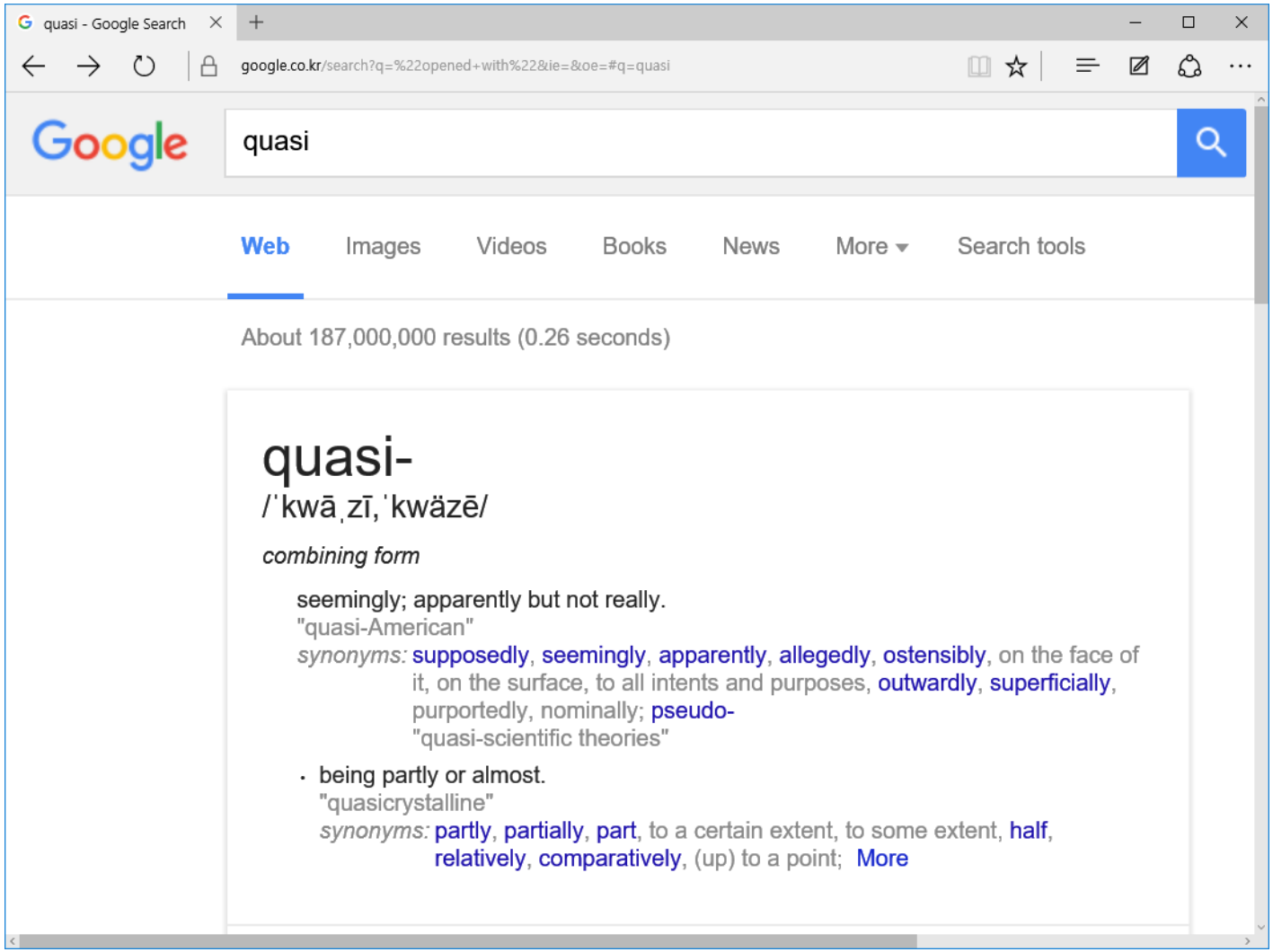


Quasi-Asynchronous I/O (QASIO)

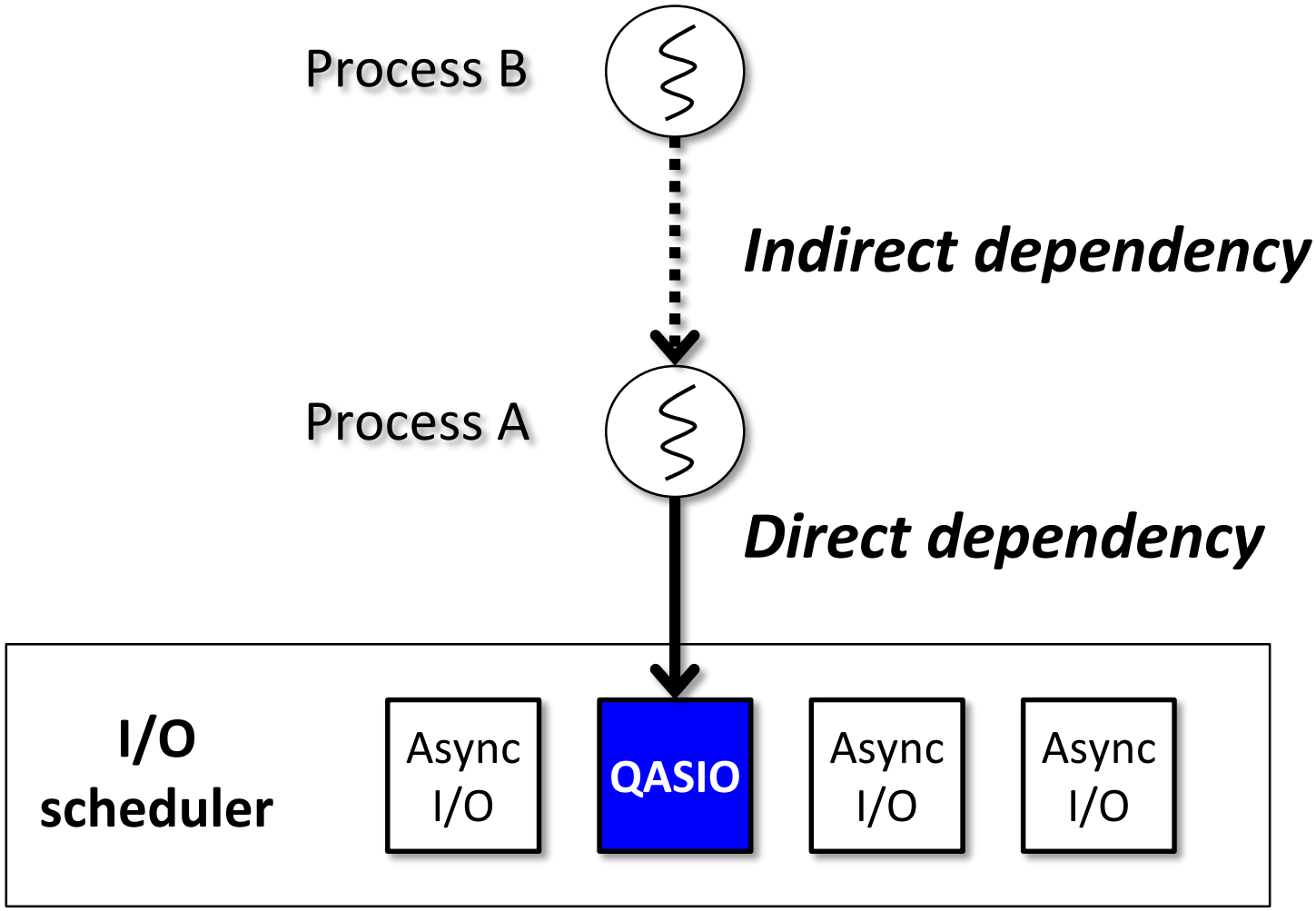
- An I/O which is issued asynchronously, but has the synchronous property because one or more tasks are waiting for its completion
- An asynchronous I/O is promoted to a QASIO at run time when a task is blocked on it
- Causes unexpected delay



Quasi-?


quasi - Google Search
google.co.kr/search?q=%22opened+with%22&ie=&oe=#q=quasi
Google
quasi
Web
Images
Videos
Books
News
More
Search tools
About 187,000,000 results (0.26 seconds)
quasi-
/ˈkwāˌzī,ˈkwäzē/
combining form
seemingly; apparently but not really.
"quasi-American"
synonyms: supposedly, seemingly, apparently, allegedly, ostensibly, on the face of it, on the surface, to all intents and purposes, outwardly, superficially, purportedly, nominally; pseudo-
"quasi-scientific theories"
being partly or almost.
"quasicrystalline"
synonyms: partly, partially, part, to a certain extent, to some extent, half, relatively, comparatively, (up) to a point; More

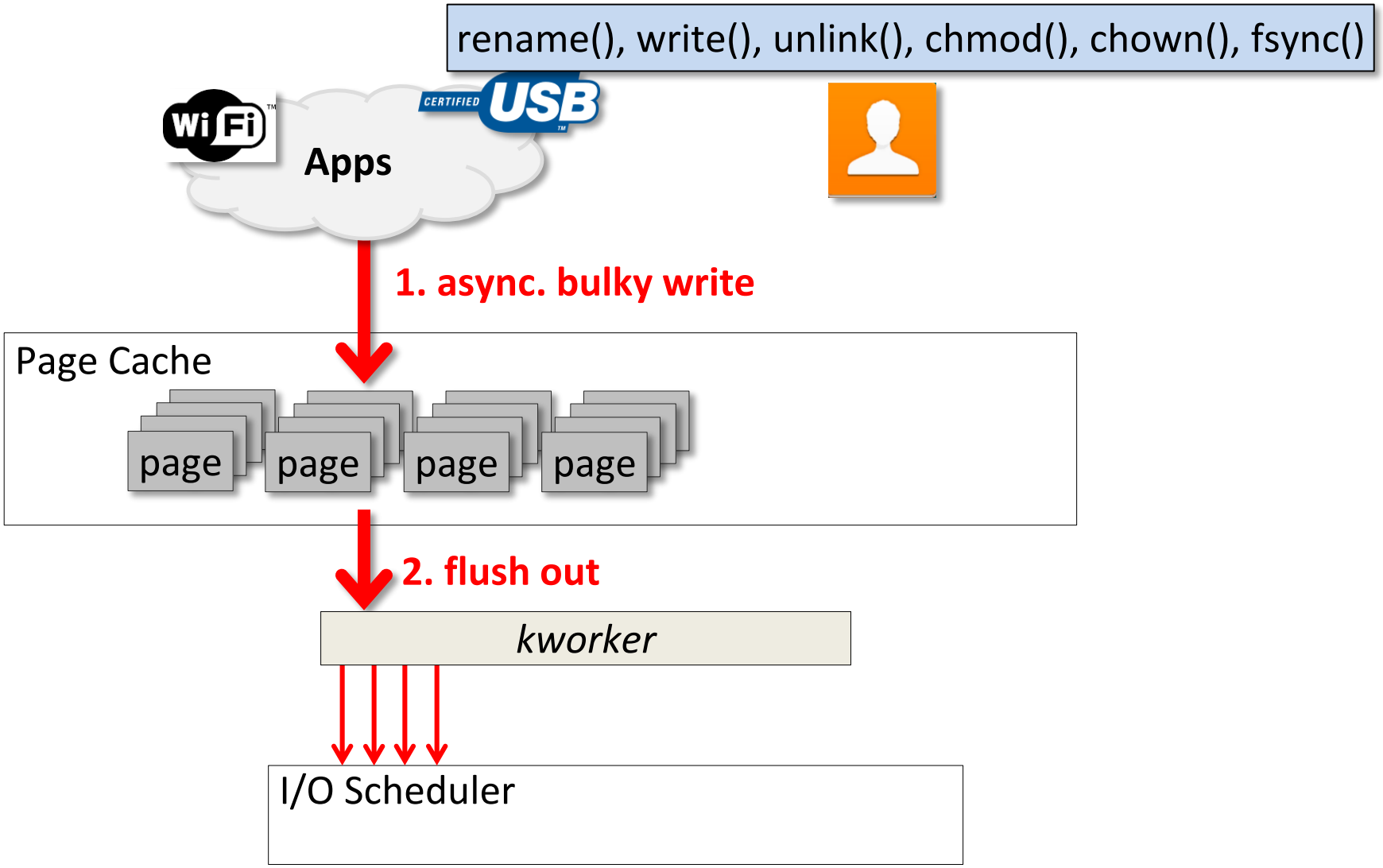
Dependencies on QASIO



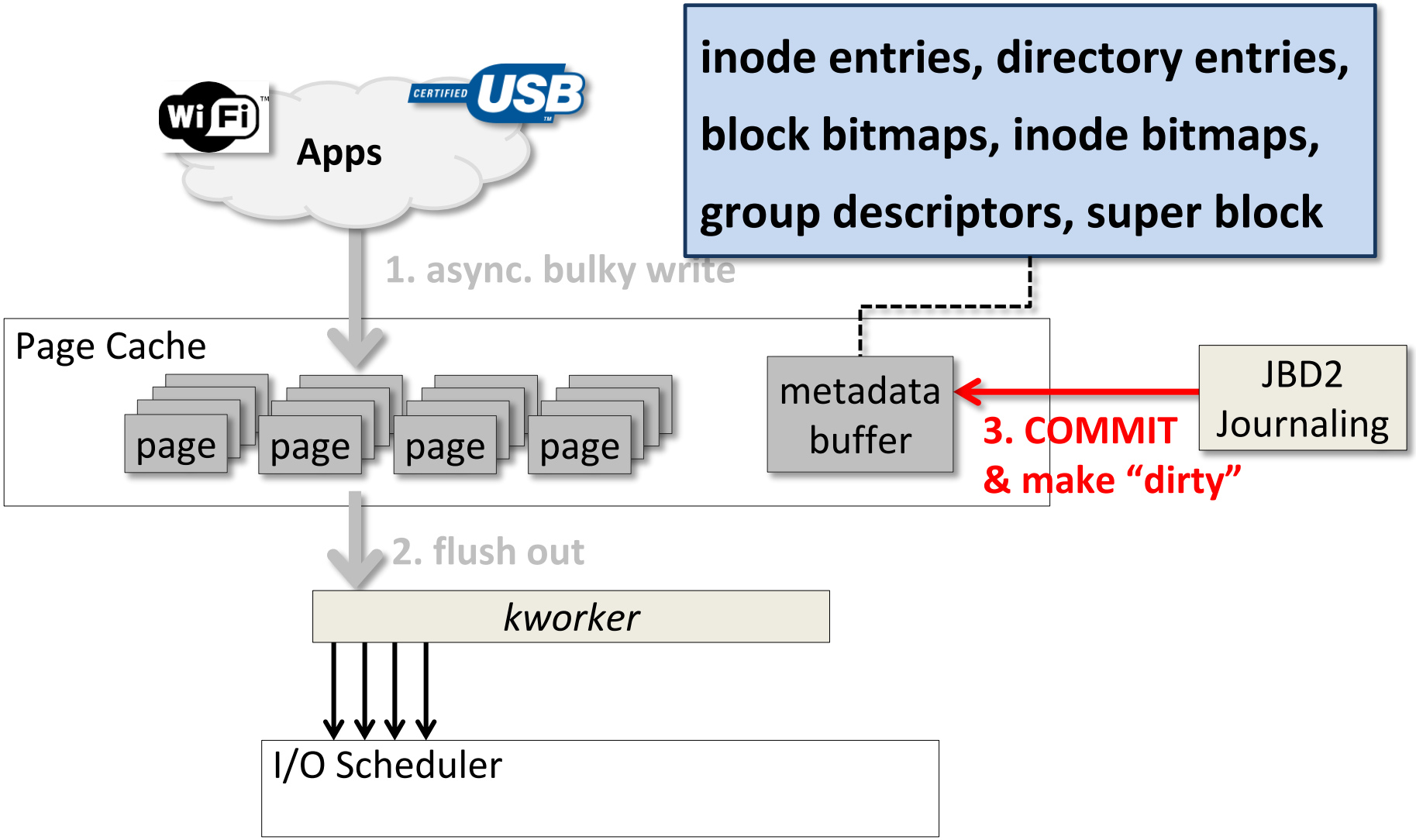
Types of Dependencies on QASIO

Direct Dependencies	
D_{meta}	When modifying a metadata page
D_{data}	When modifying a data page
D_{sync}	When guaranteeing data to be written back
$D_{discard}$	When completing discard commands
Indirect Dependencies	
$I_{jhandle}$	When unable to obtain a journal handle
$I_{jcommit}$	When unable to complete `fsync()`

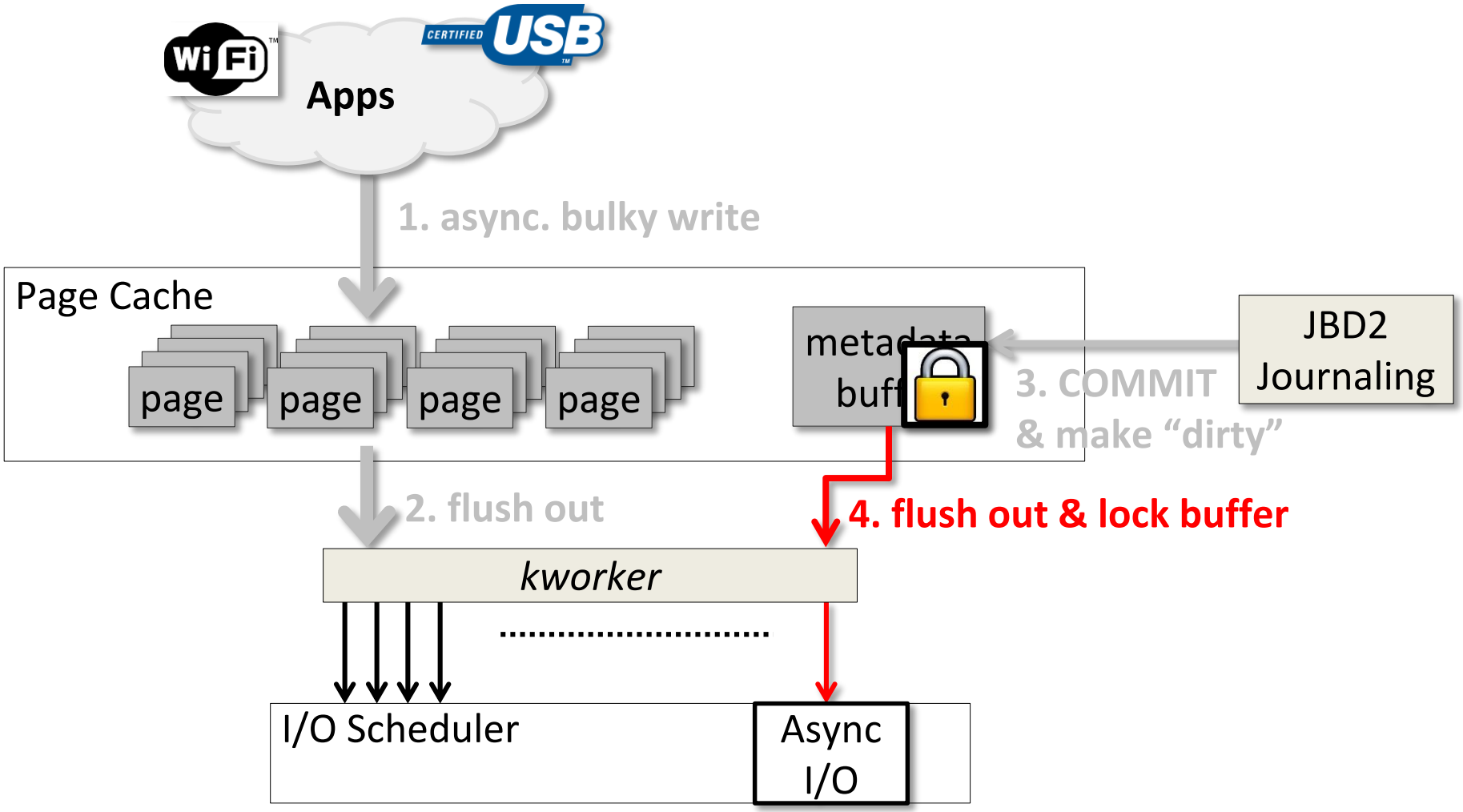
Case Study: *Dmeta*



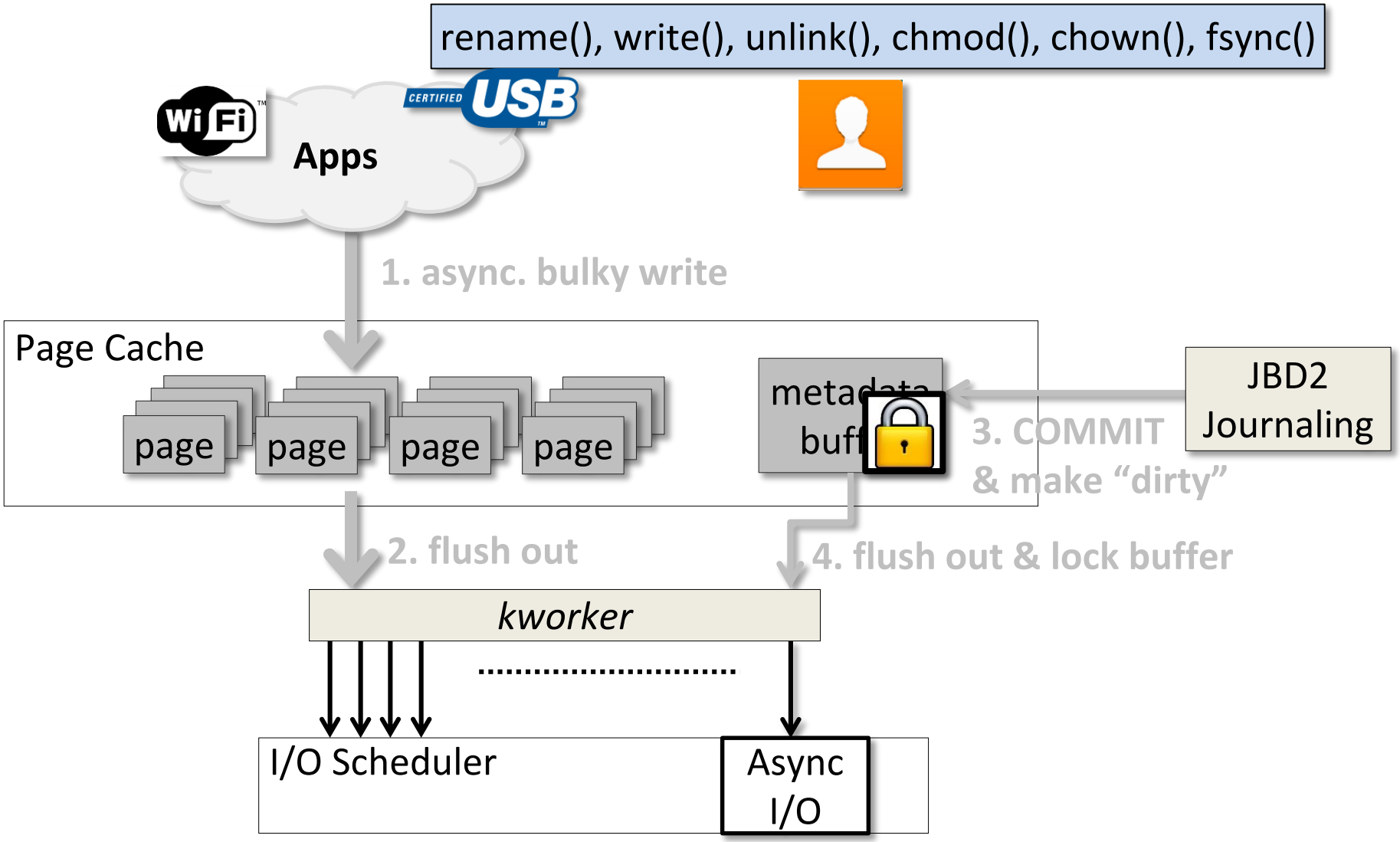
Case Study: *Dmeta*



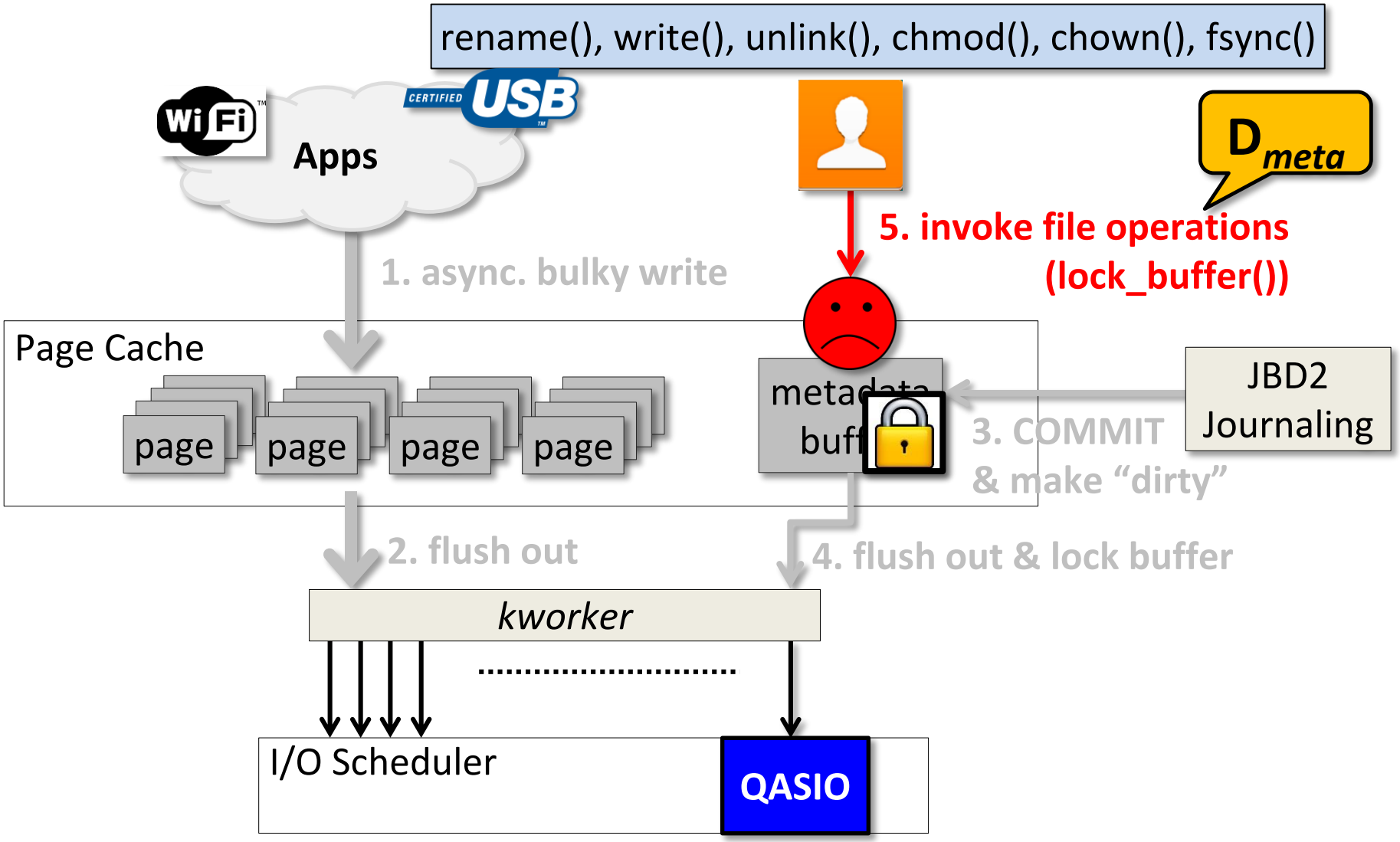
Case Study: *Dmeta*



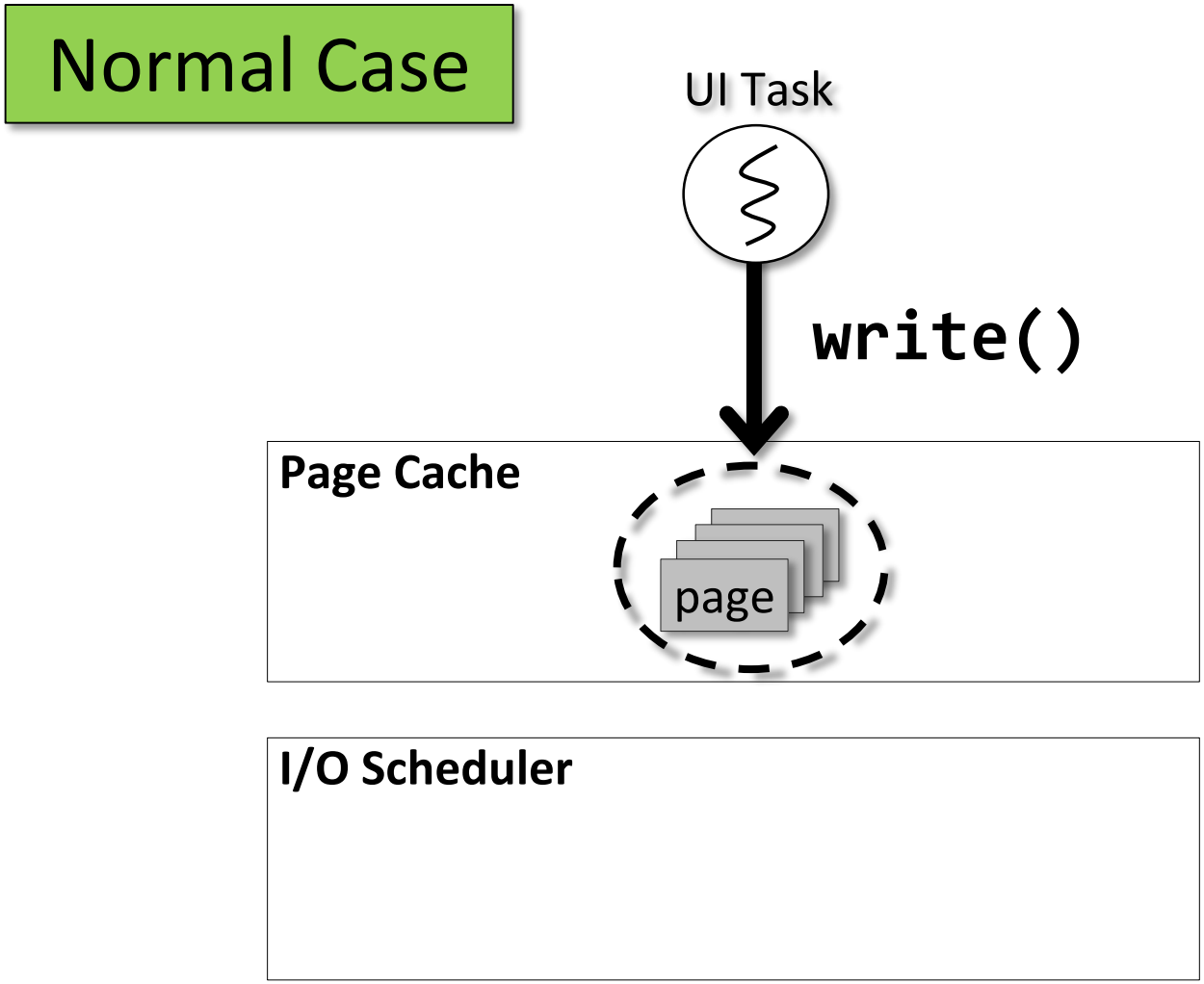
Case Study: *Dmeta*



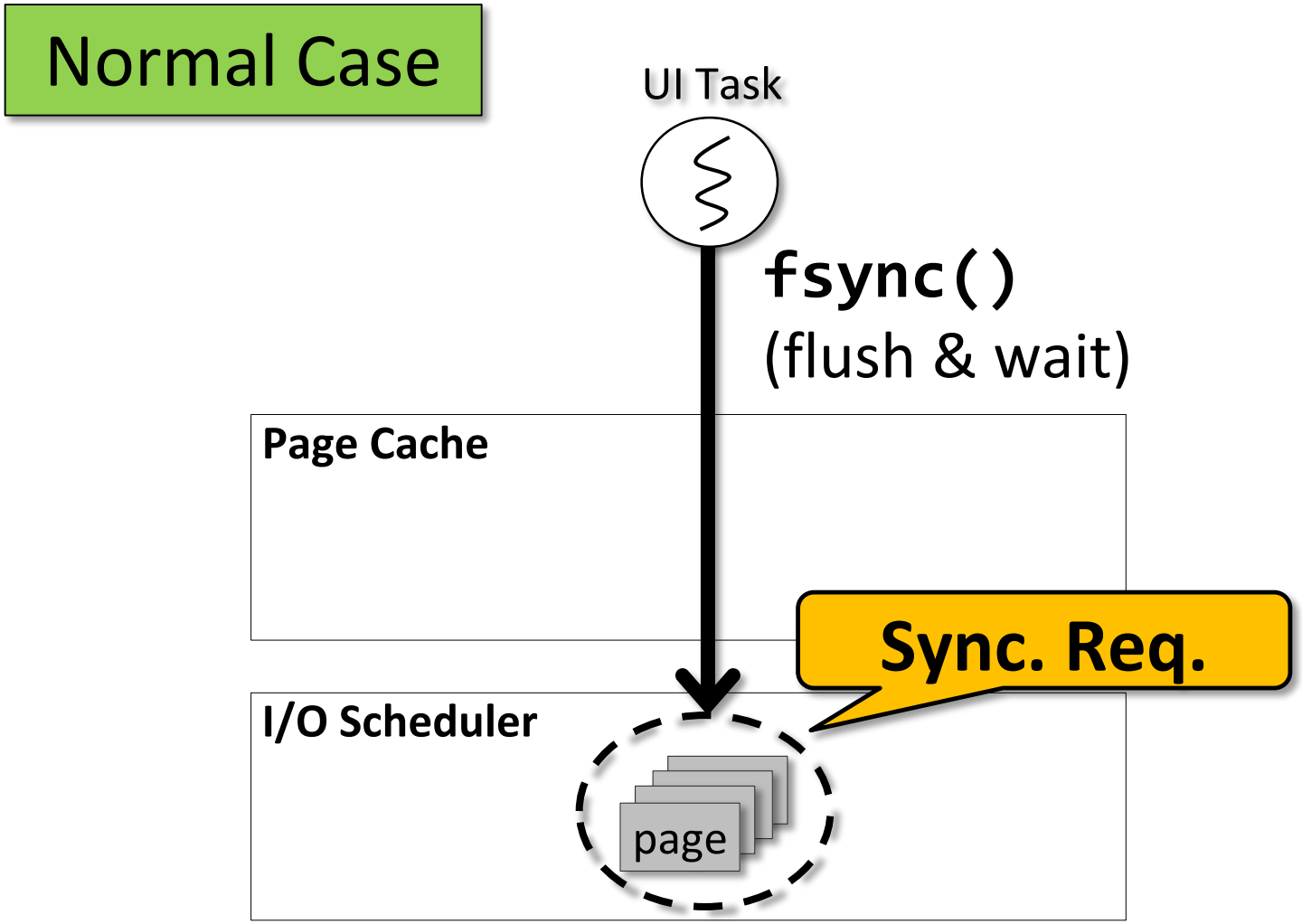
Case Study: *Dmeta*



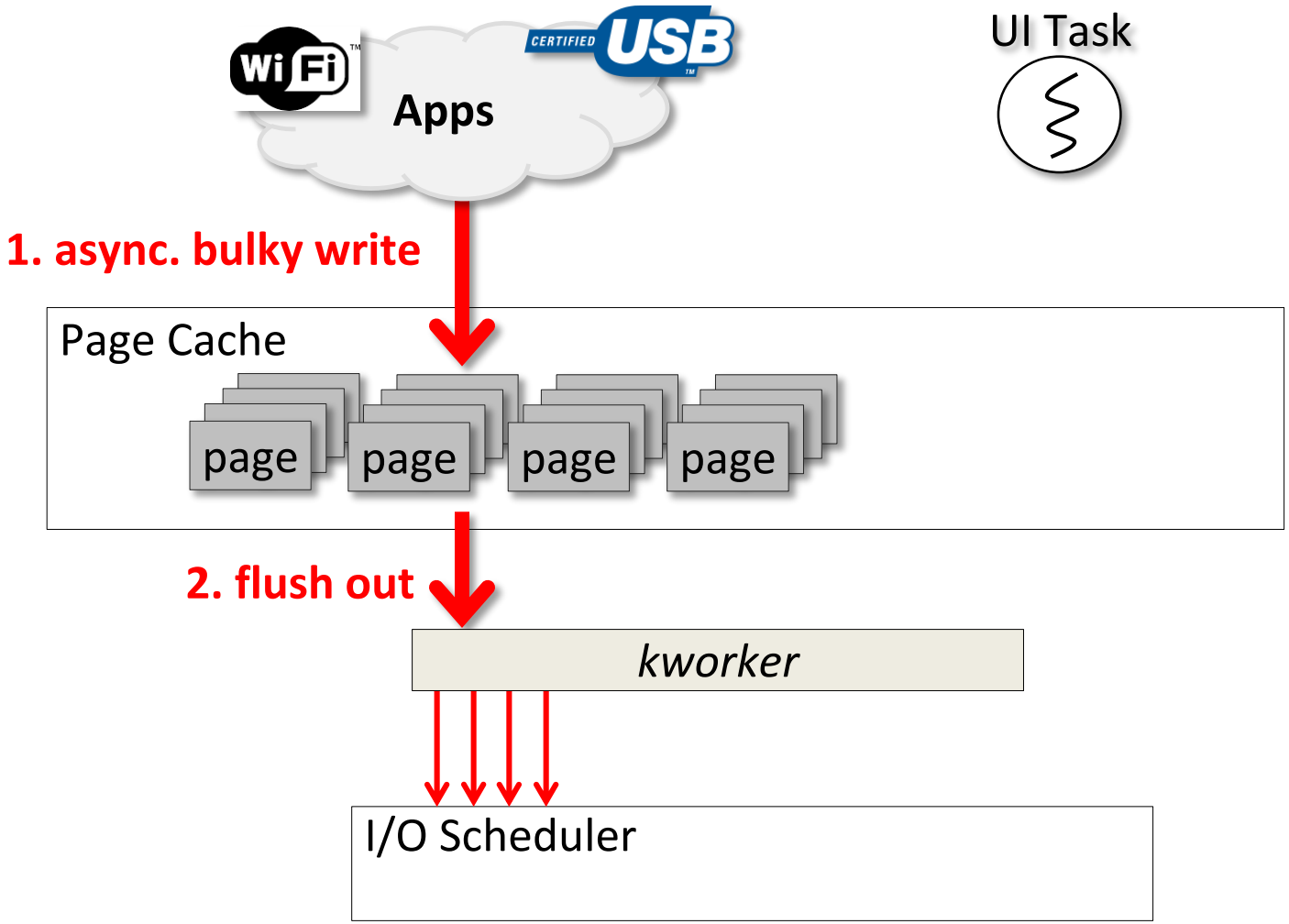
Case Study: *Dsync*



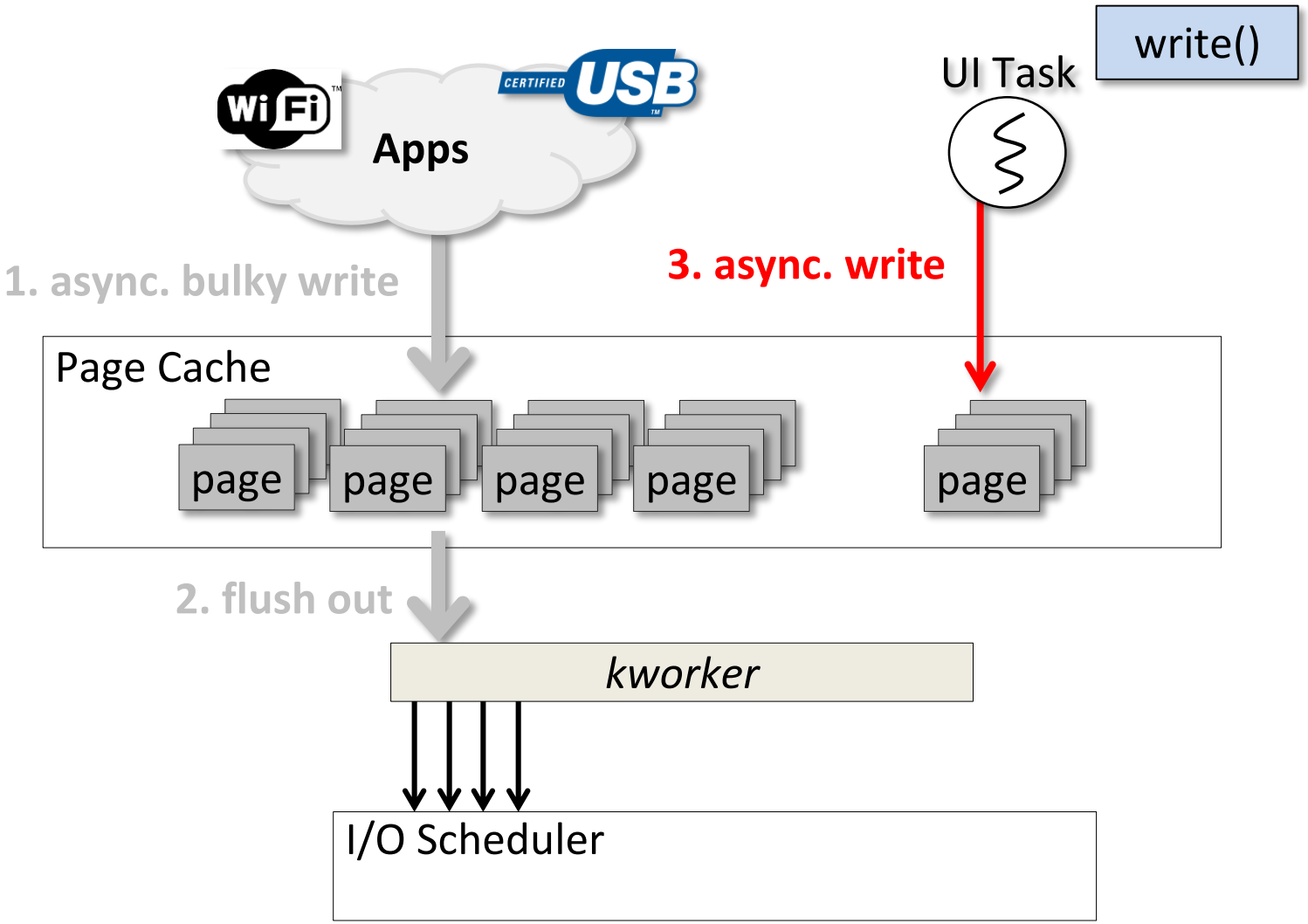
Case Study: *Dsync*



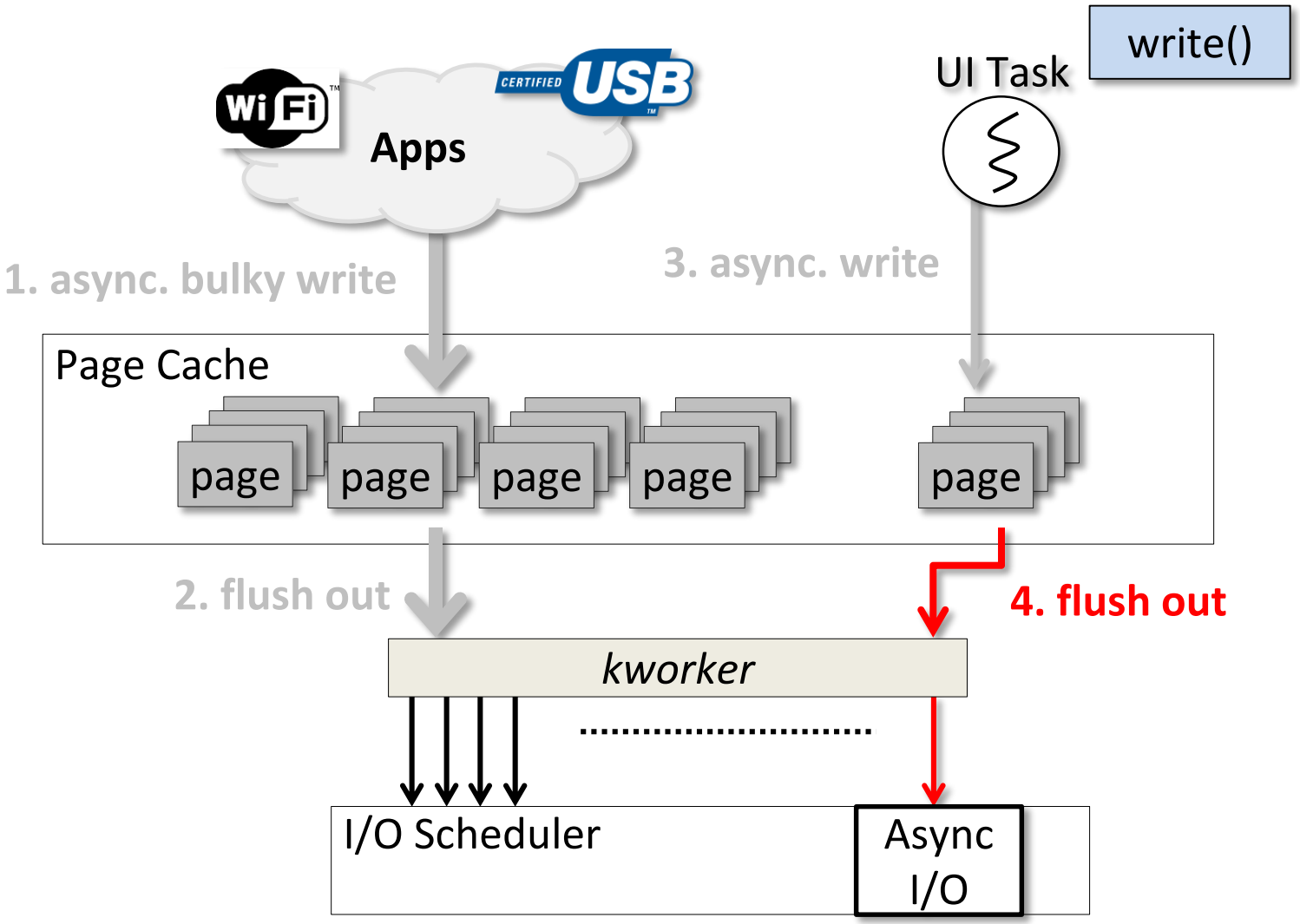
Case Study: *Dsync*



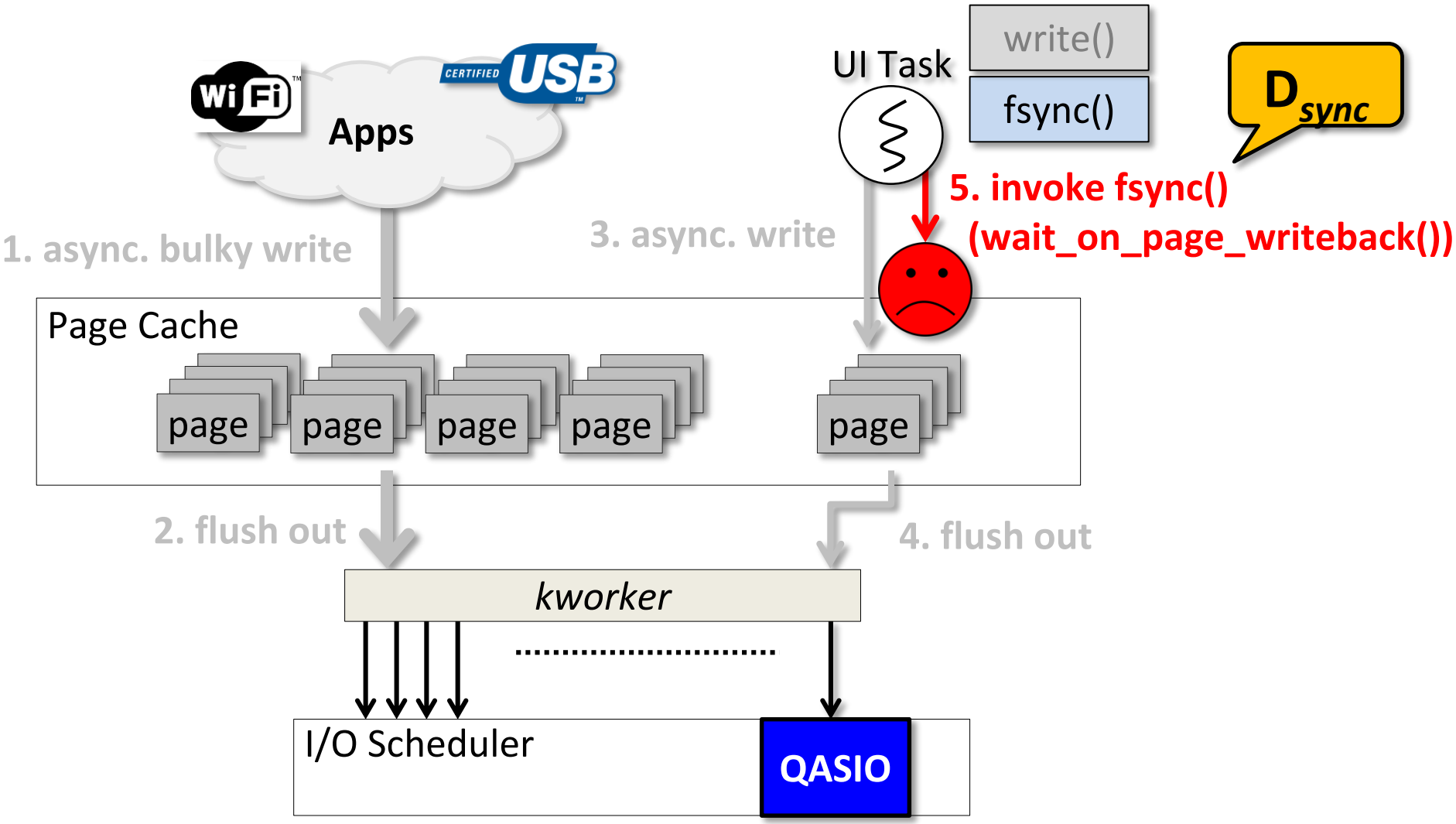
Case Study: *Dsync*



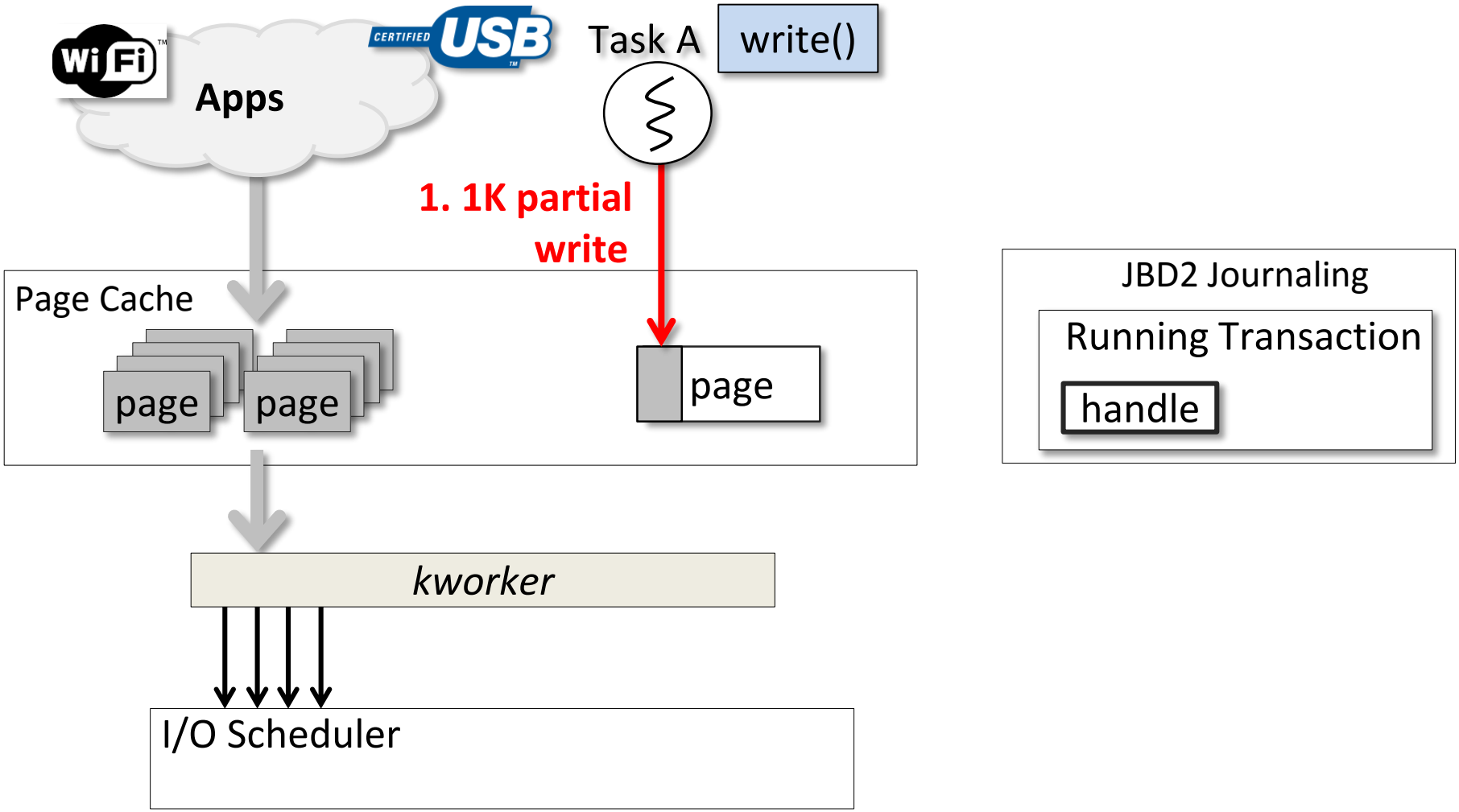
Case Study: *Dsync*



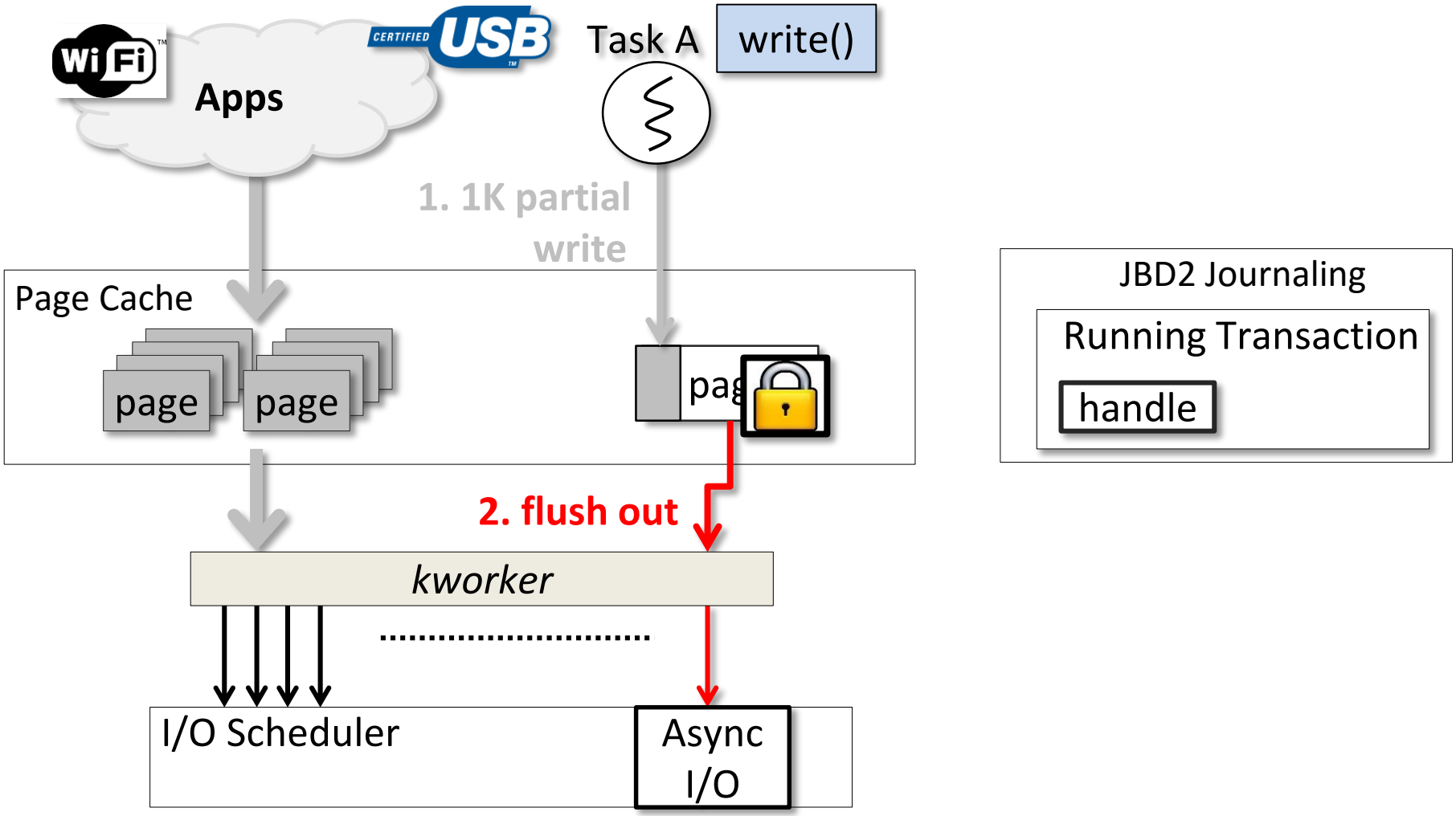
Case Study: *Dsync*



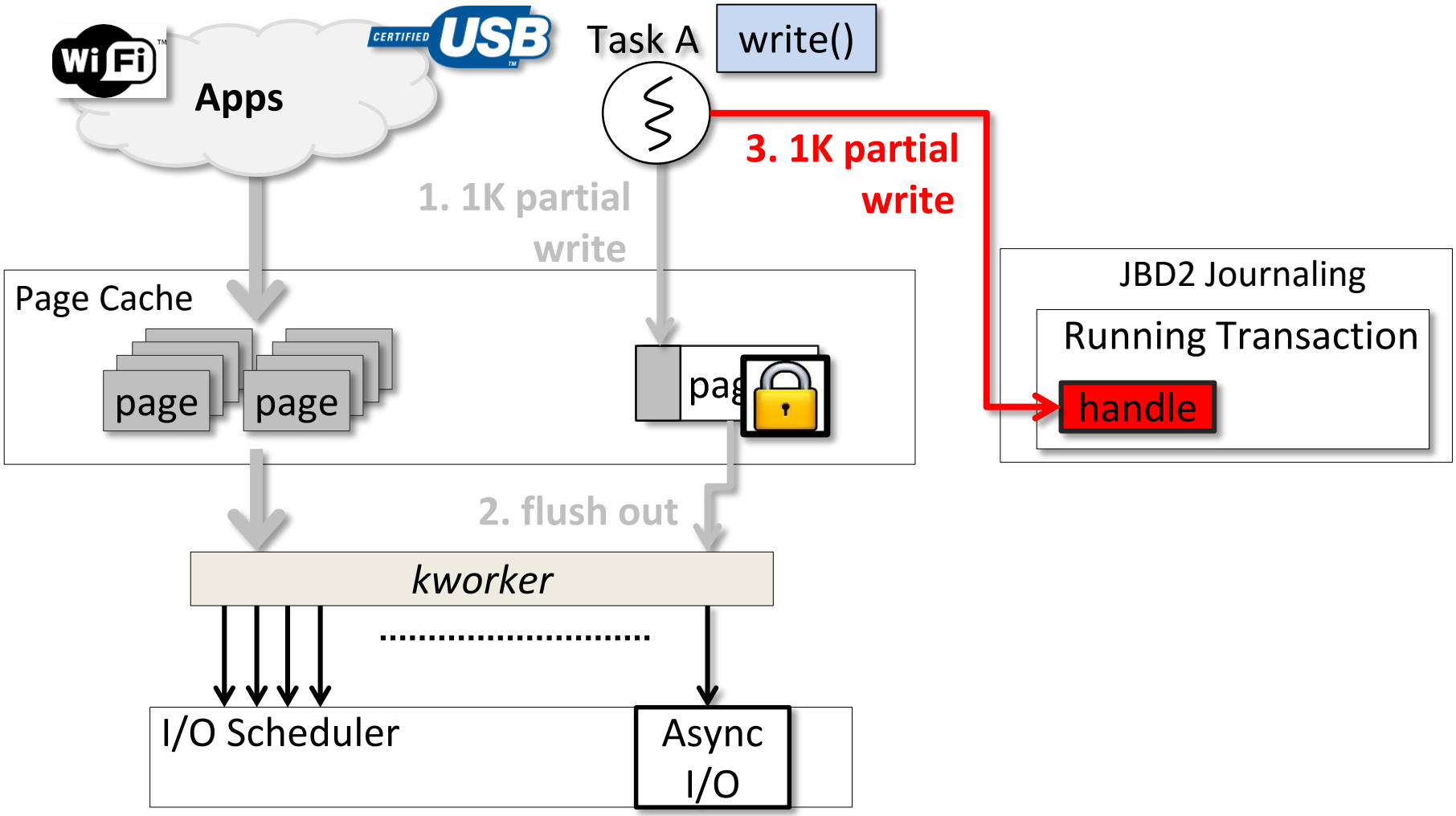
Case Study: *Ijhandle*



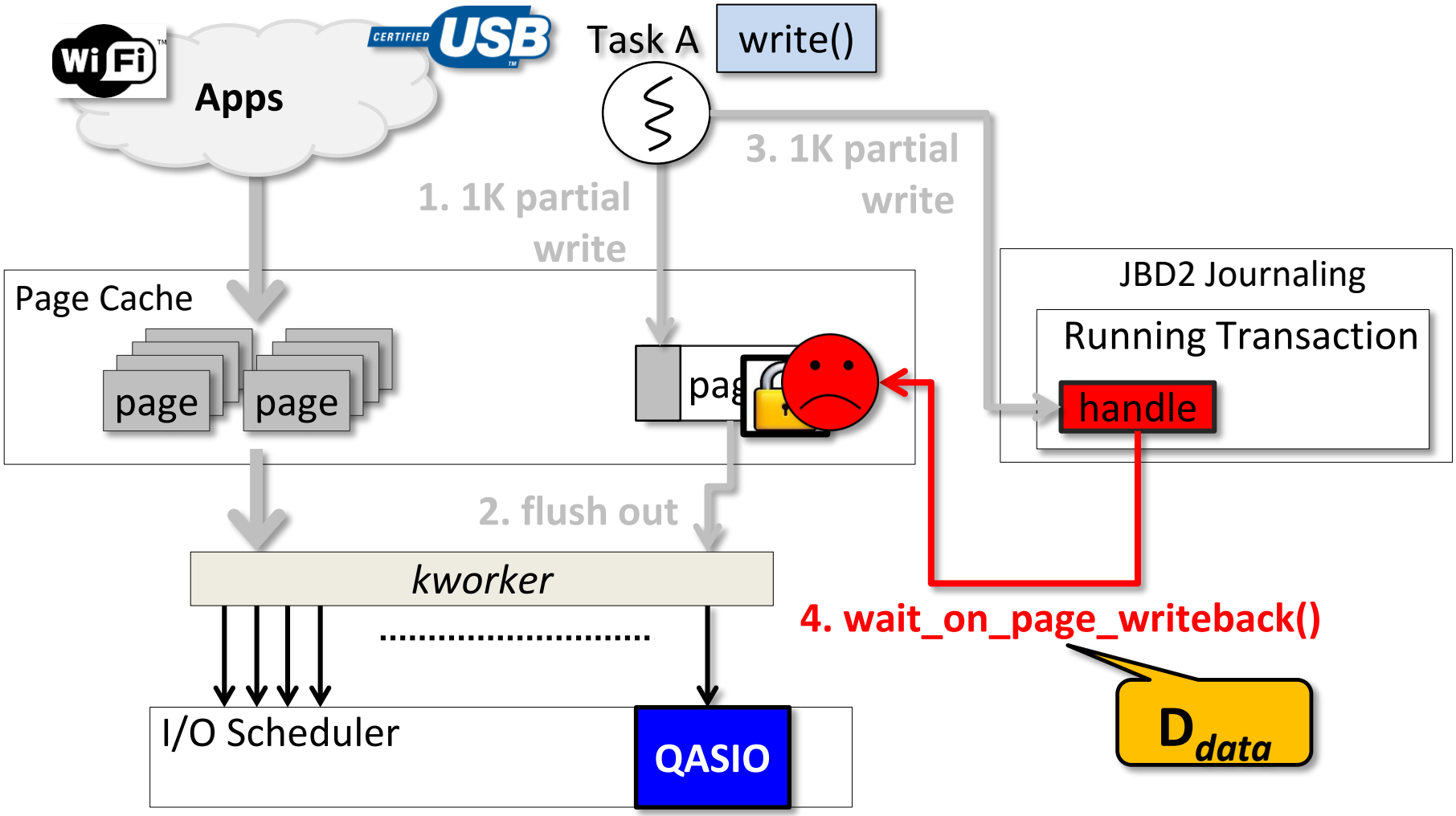
Case Study: *Ijhandle*



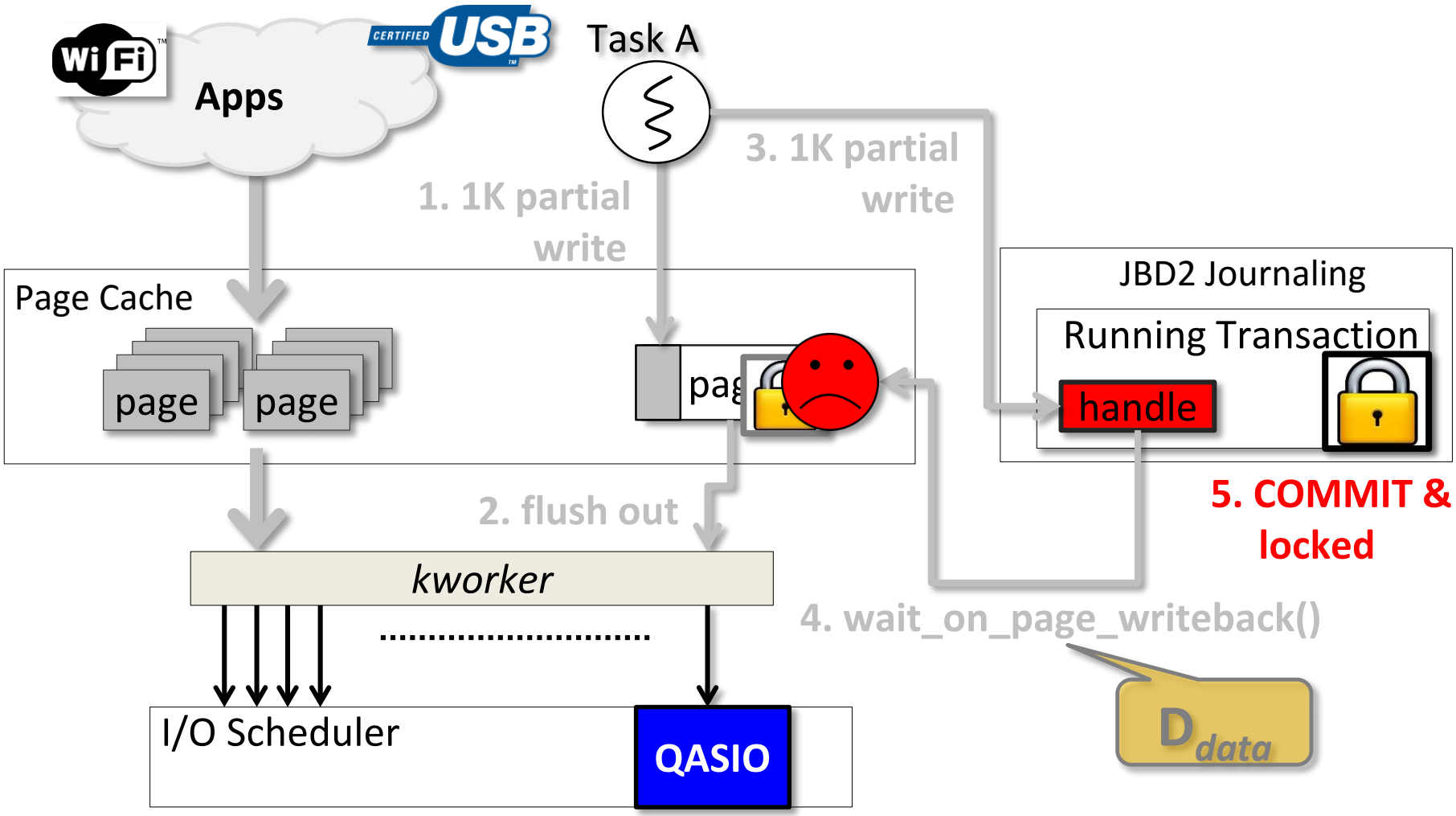
Case Study: *Ijhandle*



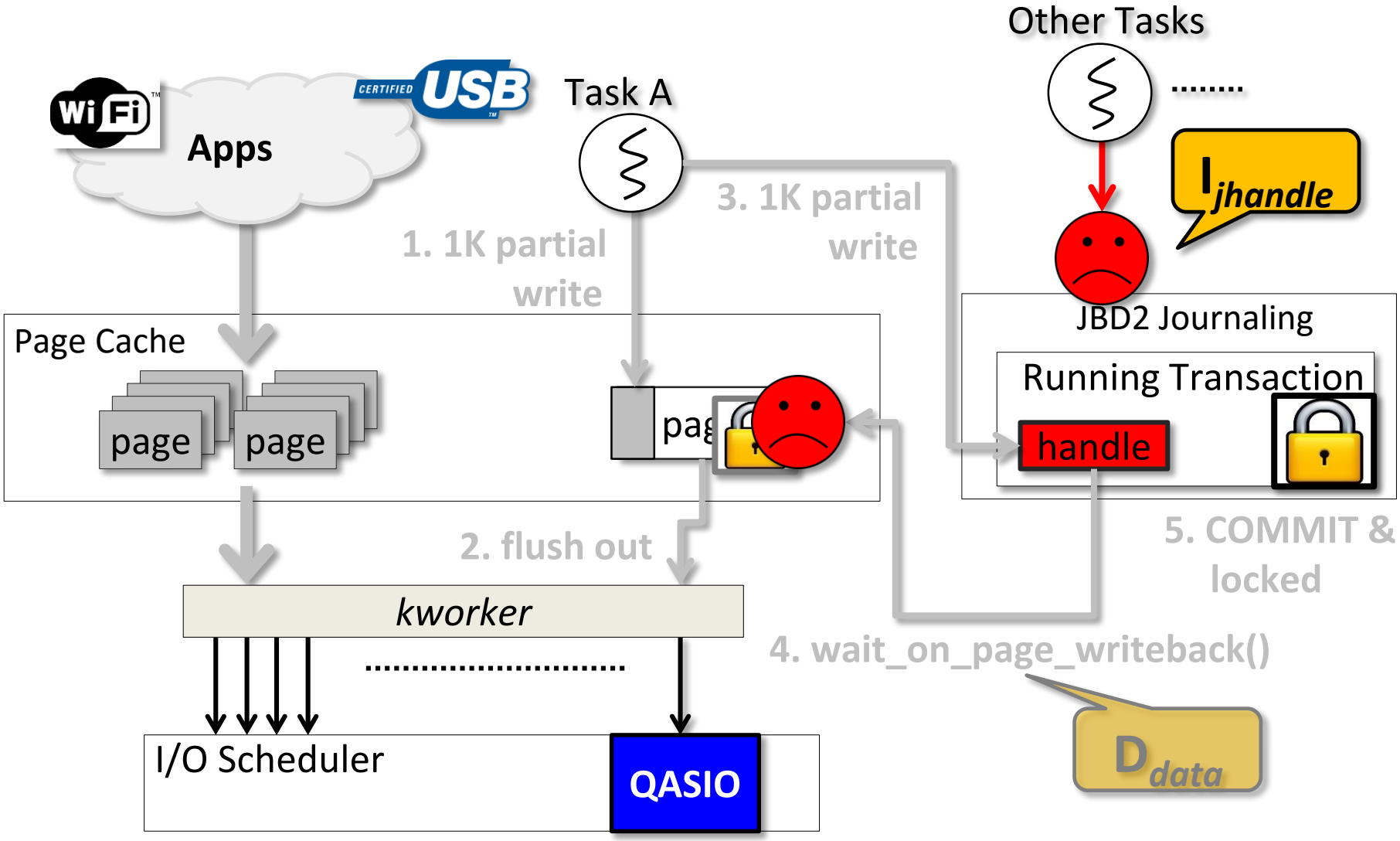
Case Study: *Ijhandle*



Case Study: *Ijhandle*



Case Study: *Ijhandle*



Boosting QASIOs

- Just focus on resolving direct dependencies

- Step 1: Detect a QASIO at run time
  - When a task is waiting for the completion of an async I/O
  - In the VFS, Page Cache, and Ext4 Filesystem layers
- Step 2: Notify the I/O scheduler
  - Requires a new interface to give the information on QASIO
  - In the block layer
- Step 3: Prioritize the QASIO
  - In the I/O scheduler

Detecting QASIOs

VFS Layer

```
lock_buffer();
```

Detect D_{meta}

Page Cache

```
wait_on_page_writeback();
```

Detect D_{data} & D_{sync}

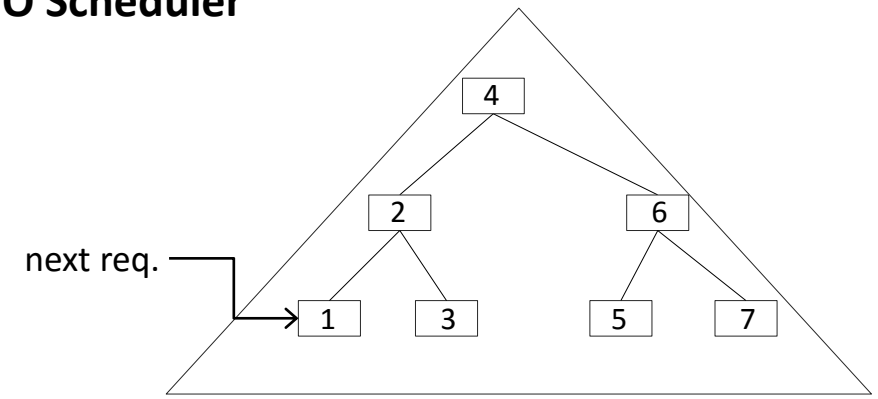
EXT4 Filesystem

```
ext4_free_data_callback();
```

Detect $D_{discard}$

Block Layer

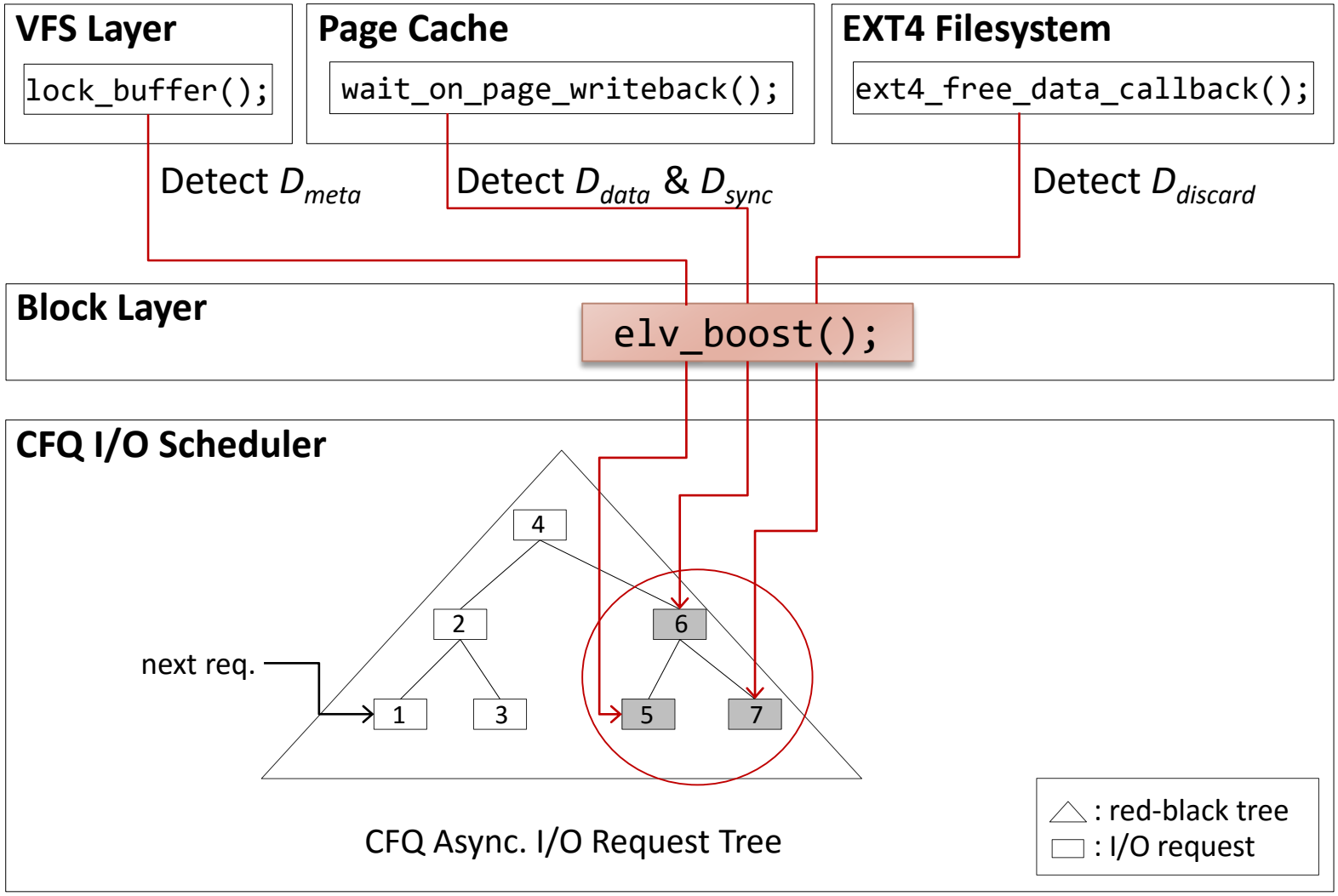
CFQ I/O Scheduler



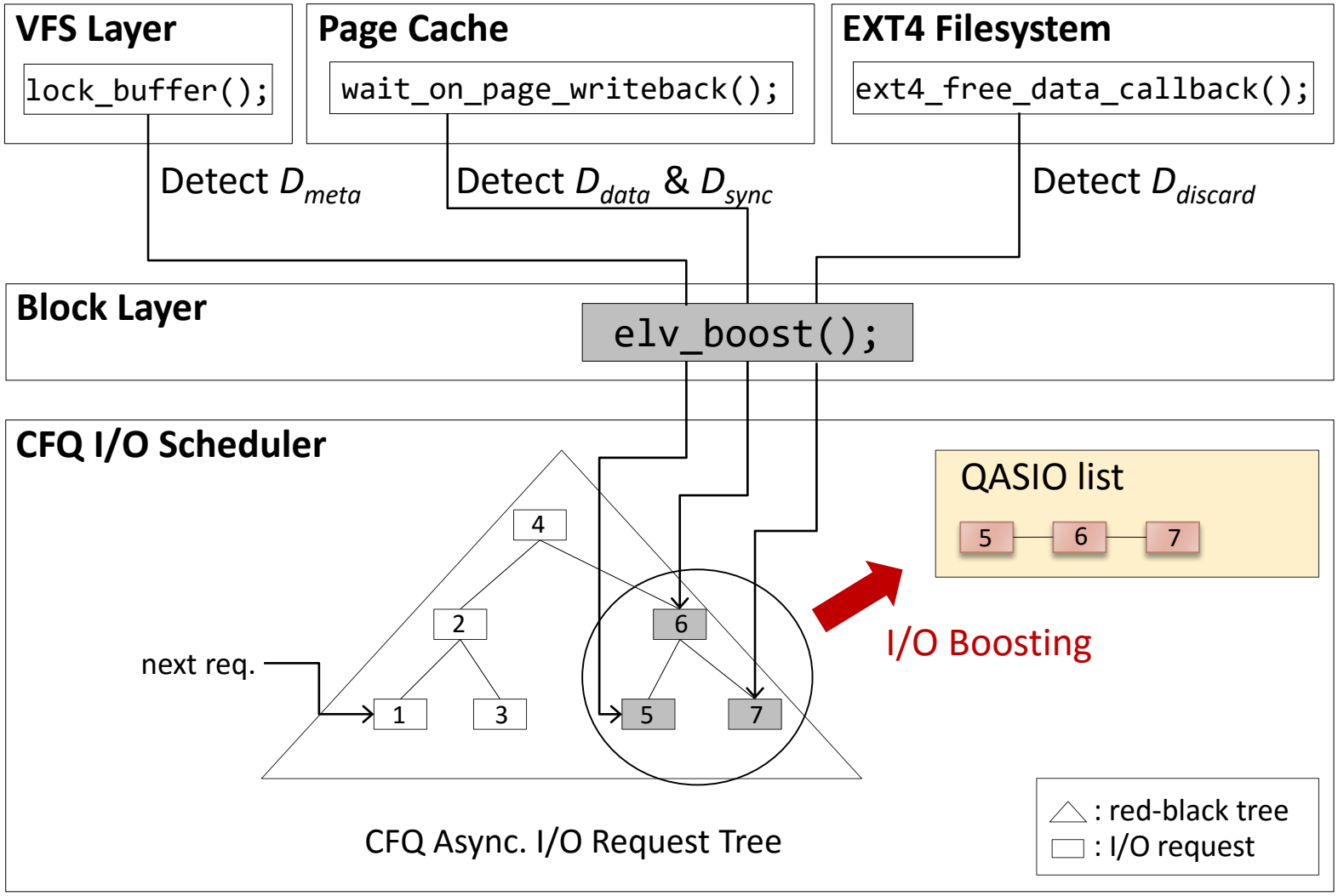
CFQ Async. I/O Request Tree

△ : red-black tree
▭ : I/O request

Notifying I/O Scheduler



Prioritizing QASIOs



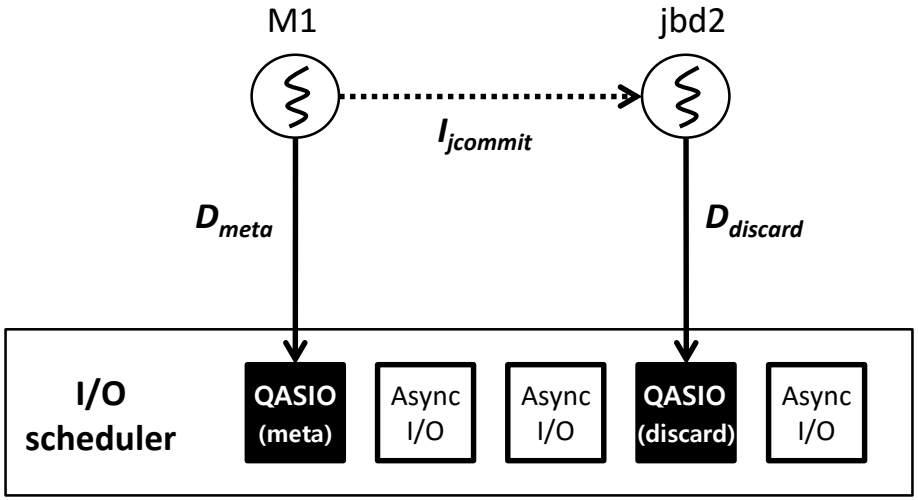
Evaluations

- Samsung Galaxy S5
  - Exynos 5422 (quad Cortex-A15 & quad Cortex-A7)
  - 2GB DRAM
  - 16GB eMMC storage (SW: 54.5MB/s, RW: 10.4MB/s)
  - Android platform version 4.4.2 (KitKat)
  - Linux kernel version 3.10.9

- Evaluation methodology
  - Microbenchmarks (M1 ~ M5)
  - Real-life scenarios (Contacts, Camera, Angry Birds)
  - Android I/O benchmarks

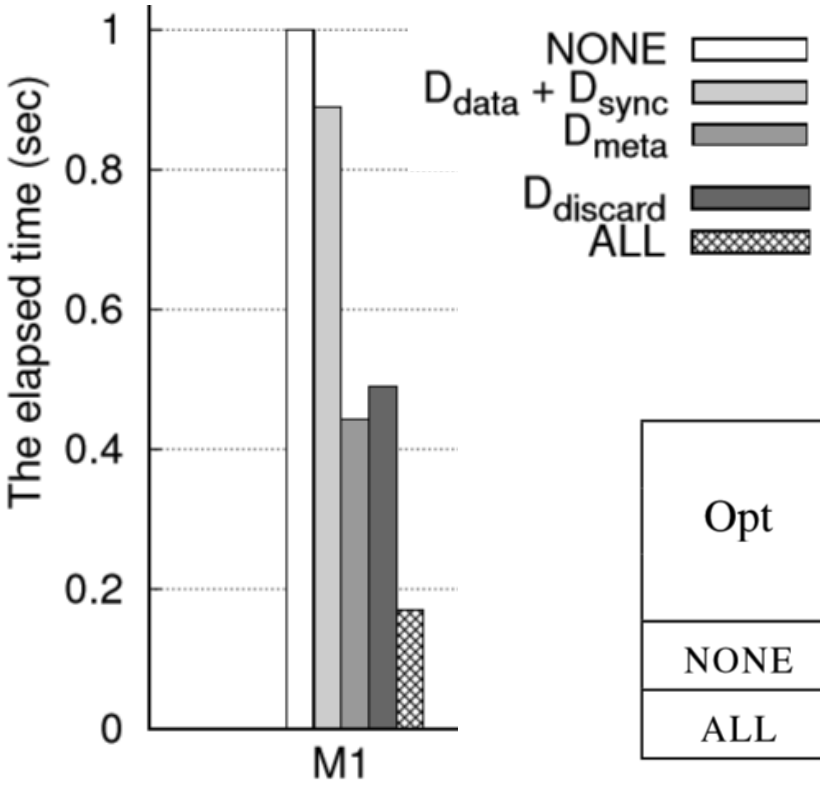
Microbenchmarks

- M1
  - Iterates the creation of a 4KB file, 500 times
  - `creat()` → `write(4KB)` → `fsync()` → `close()`
  - Mimics the I/O pattern of DBMS (e.g. SQLite)
  - A 8GB file is being created in the background



Microbenchmarks

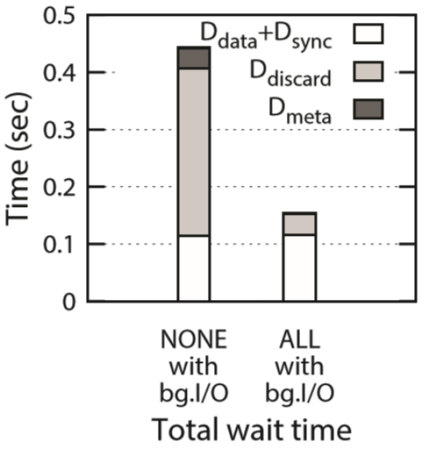
- M1 results

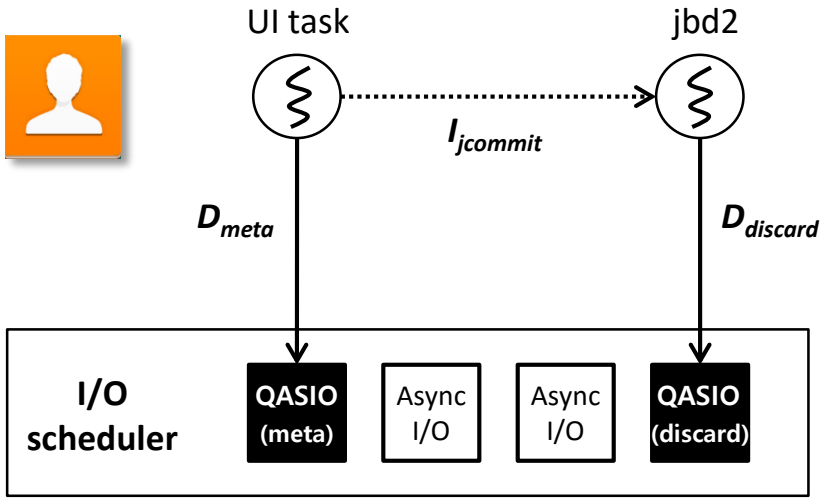


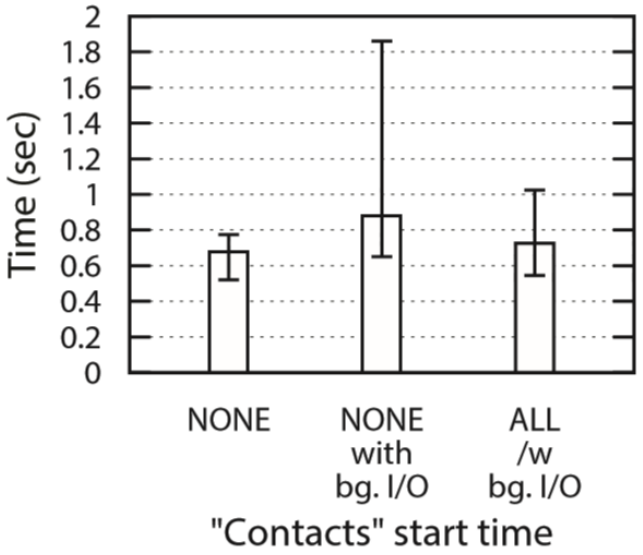
Opt	M1			
	`creat()`		`fsync()`	
	Avg	Max	Avg	Max
NONE	119.57	1435.54	98.24	1119.62
ALL	**1.02**	**39.15**	**35.64**	**144.69**

Launching "Contacts" App

- The UI task updates states into databases or files
  - `rename()`, `write()`, `fsync()`, `unlink()`, …
  - All modify metadata (*Dmeta*)
  - `fsync()` has *Ijcommit* to jbd2

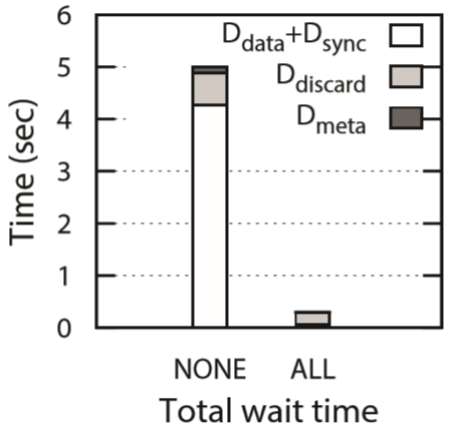


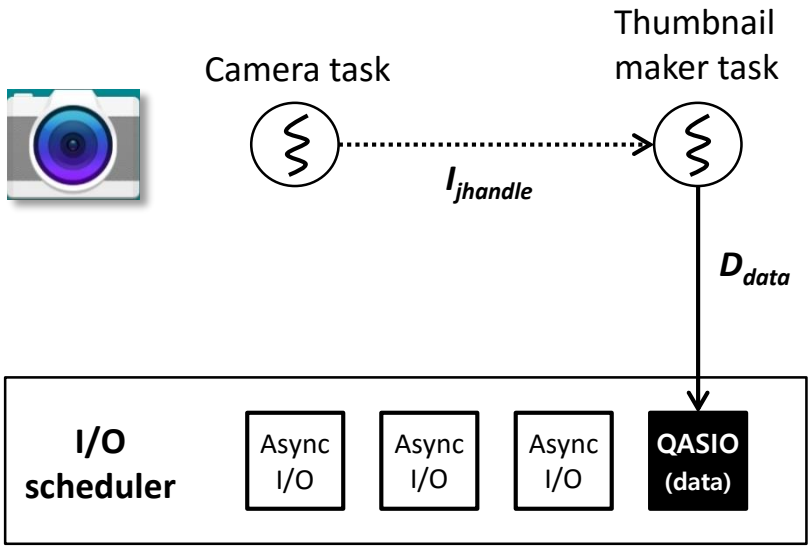


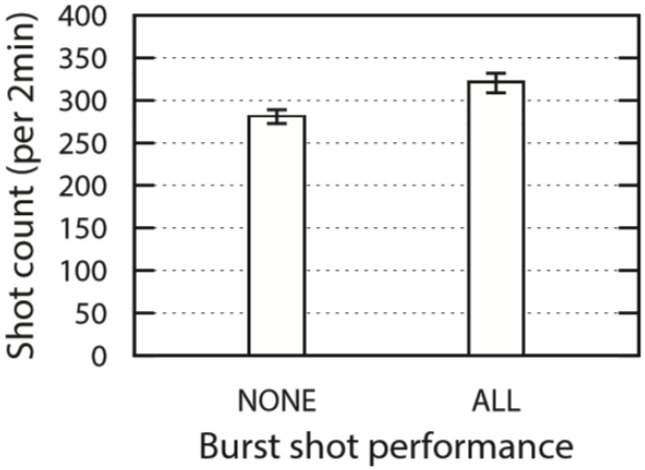


Burst Mode in "Camera" App

- The thumbnail maker creates thumbnails after burst shot
  - Writes data in a small size (*Ddata*)
  - The camera task has *Ijhandle* to the thumbnail maker

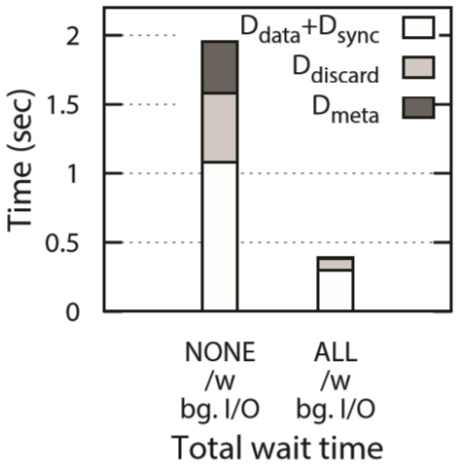


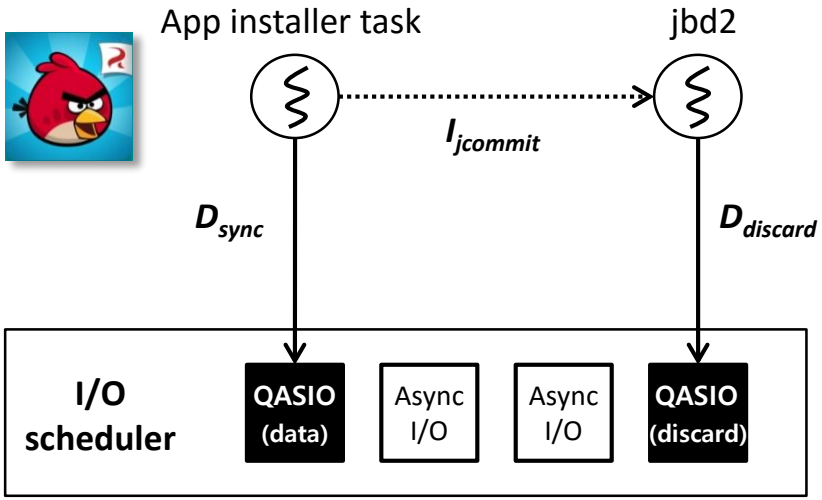


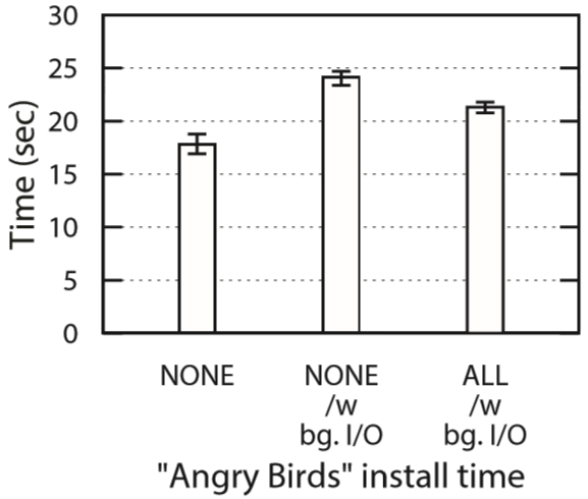


Installing "Angry Birds" App

- The App installer saves the extracted data to the app repository
  - Buffered `write()` + `fsync()` (*Dsync*)
  - `fsync()` has $I_{jcommit}$ to jbd2

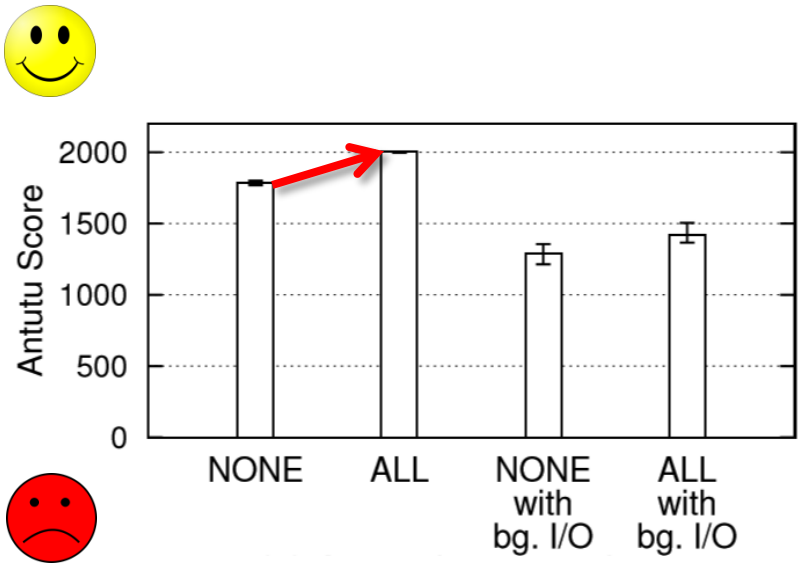




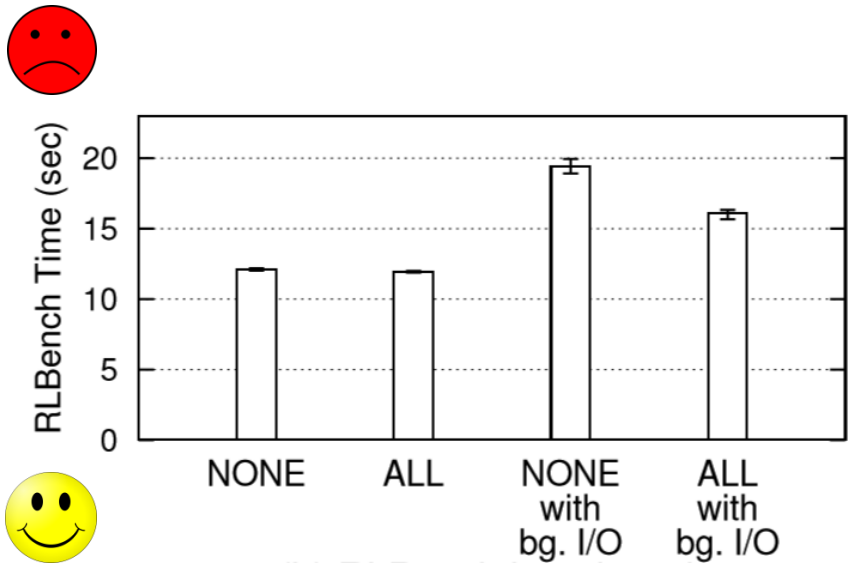


Android I/O Benchmarks

- Antutu & RLBench



Antutu benchmark



RLBench benchmark

2015 USENIX Research in Linux File and Storage Technologies Summit

February 19, 2015, Santa Clara, CA

Summarized by Rik Farrow

Don asked about the developers' perspective and the application developers' perspective. Ted countered by asking whether this is the application you really care about? What is the real world case where we are not optimized for this area? If he decided to fix this in ext4, but people who run into this are using xfs, why should he fix this? When thinking about writing papers, pick something that will be high impact—that is, something that will affect lots of users. That's why Ted liked the Quasi I/O paper [9]. But does three seconds instead of one second really matter to the handset user?

Conference Reports @ **;login:** *(June 2015)*

Thank You!

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu