#WWDC19

Optimizing Storage in Your App

Better performance and efficiency

Kai Kaahaaina, CoreOS
Alejandro Lucena, Developer Tools

Optimizing Storage in Your App

Better performance and efficiency

Optimizing Storage in Your App

Better performance and efficiency

Battery life

Optimizing Storage in Your App

Better performance and efficiency

Battery life

Performance

Optimizing Storage in Your App

Better performance and efficiency

Battery life

Performance

Reduce footprint

Optimizing Storage in Your App

Better performance and efficiency

Battery life

Performance

Reduce footprint

Device health

Efficient image assets

File system metadata

Syncing to disk

Serialized data files

Core Data

SQLite

Efficient Image Assets

Image Assets

Growing in size

Increasing screen resolutions

- iPhone 6s — 1334 x 750
- iPhone Xs — 2436 x 1125

Higher resolution cameras

Harder to manage

Image Assets

Growing in size

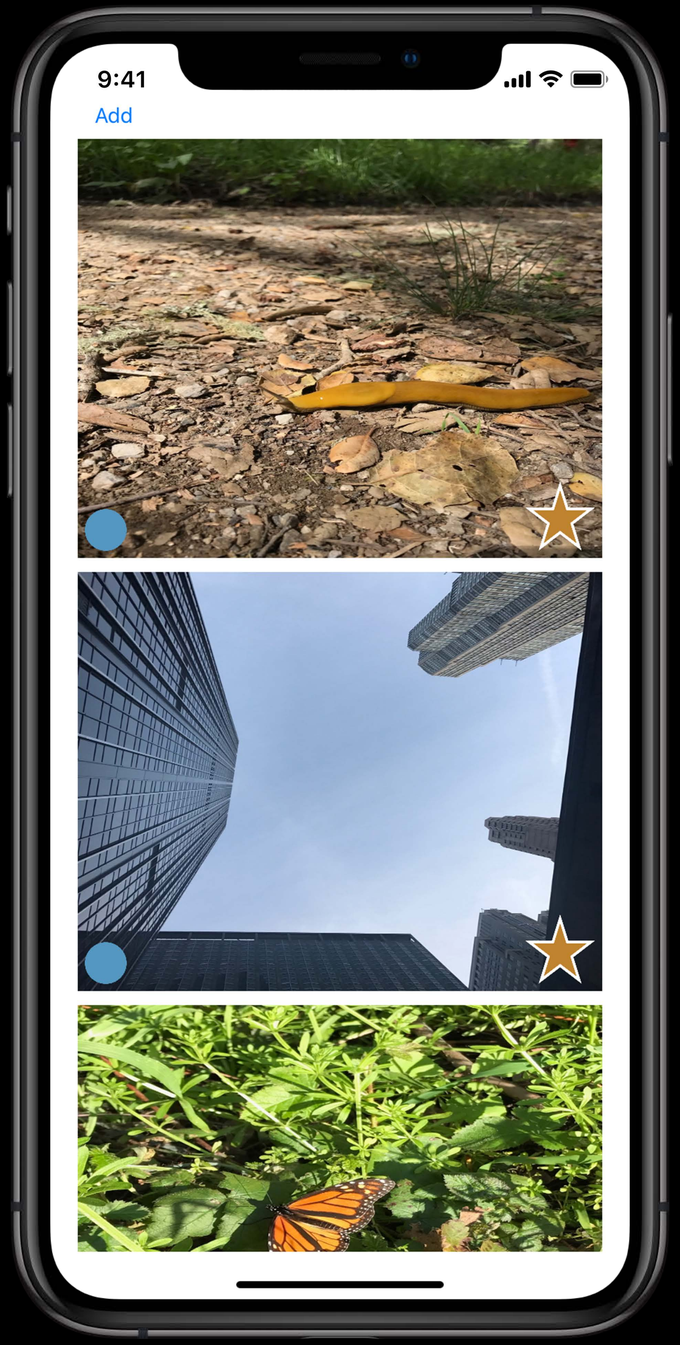
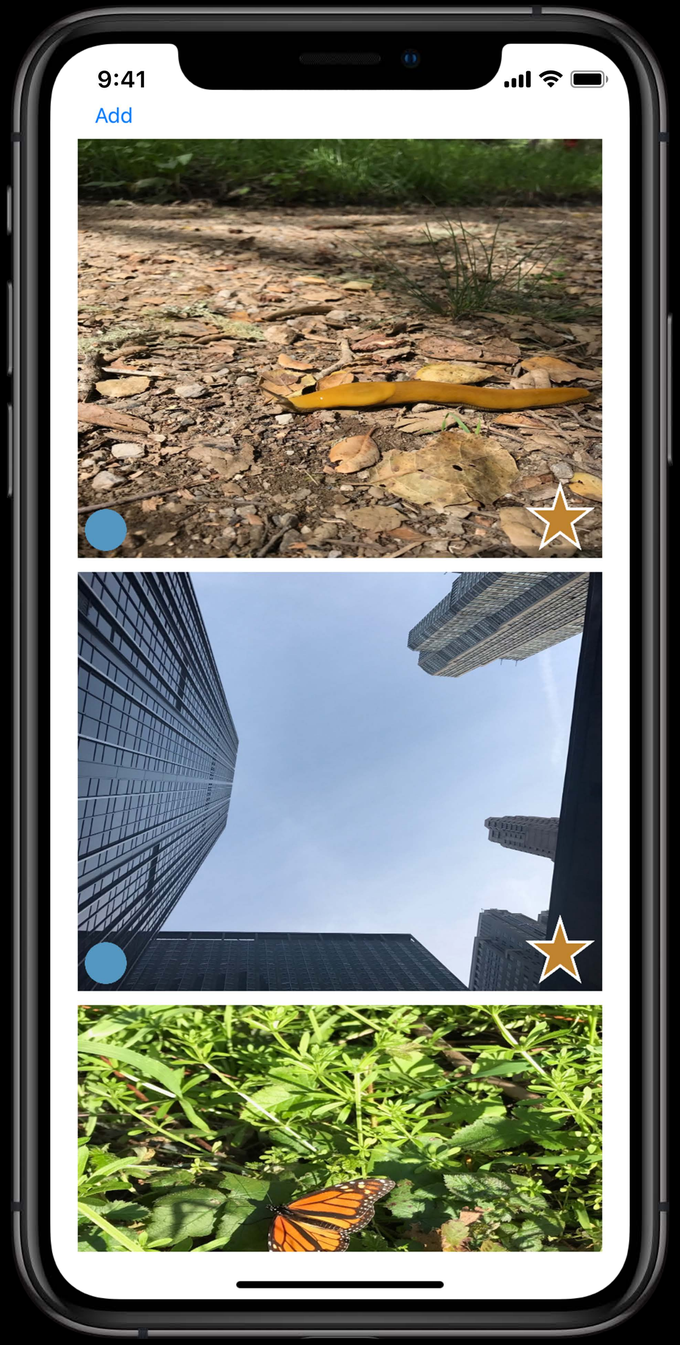


Image Assets

Growing in size

JPEG Size: 24.6MB



HEIC

A more efficient and capable alternative to JPEG

HEIC

A more efficient and capable alternative to JPEG

~50% smaller files than JPEG at comparable quality

HEIC

A more efficient and capable alternative to JPEG

~50% smaller files than JPEG at comparable quality

Smaller on disk footprint

HEIC

A more efficient and capable alternative to JPEG

~50% smaller files than JPEG at comparable quality

Smaller on disk footprint

Faster network downloads and uploads

HEIC

A more efficient and capable alternative to JPEG

~50% smaller files than JPEG at comparable quality

Smaller on disk footprint

Faster network downloads and uploads

Faster to load and save to disk

HEIC

A more efficient and capable alternative to JPEG

Store auxiliary images (depth, disparity, and so on)

HEIC

A more efficient and capable alternative to JPEG

Store auxiliary images (depth, disparity, and so on)

Supports alpha

HEIC

A more efficient and capable alternative to JPEG

Store auxiliary images (depth, disparity, and so on)

Supports alpha

Lossless compression

HEIC

A more efficient and capable alternative to JPEG

Store auxiliary images (depth, disparity, and so on)

Supports alpha

Lossless compression

Multiple images in a single container

HEIC

A more efficient and capable alternative to JPEG

Available starting with

- iOS 11
- macOS High Sierra

Image Assets

Growing in size

JPEG Size: 24.6MB

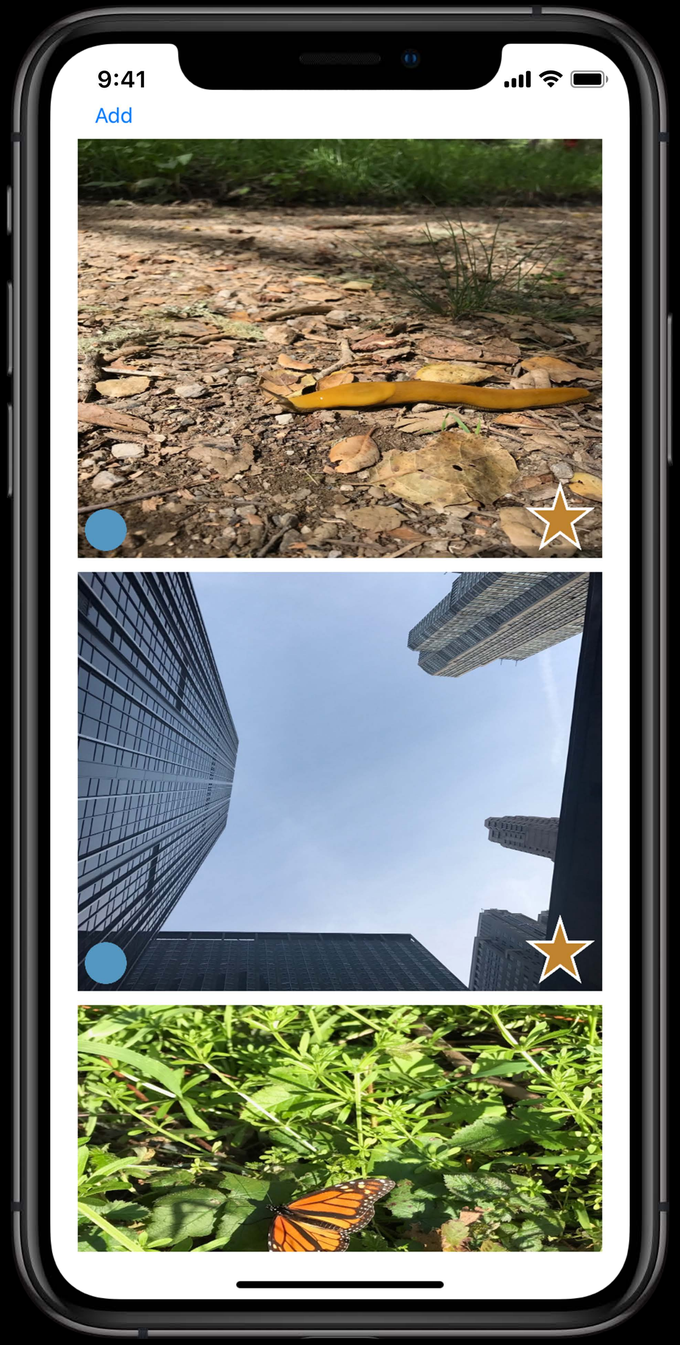


Image Assets

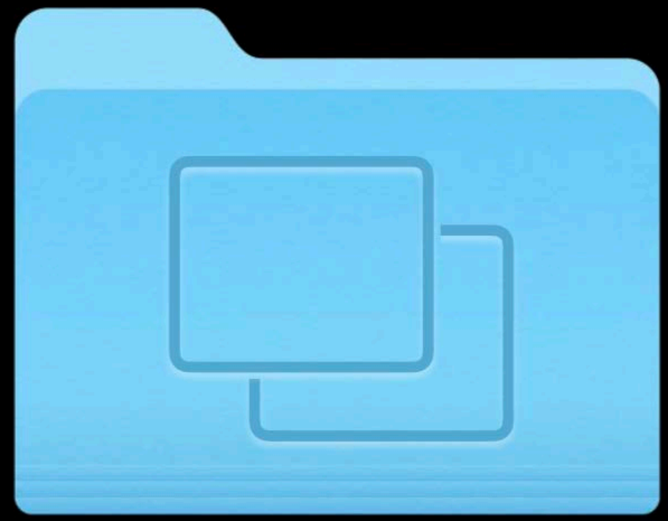
Growing in size

JPEG Size: 24.6MB

HEIC Size: 17.9MB

Asset Catalog

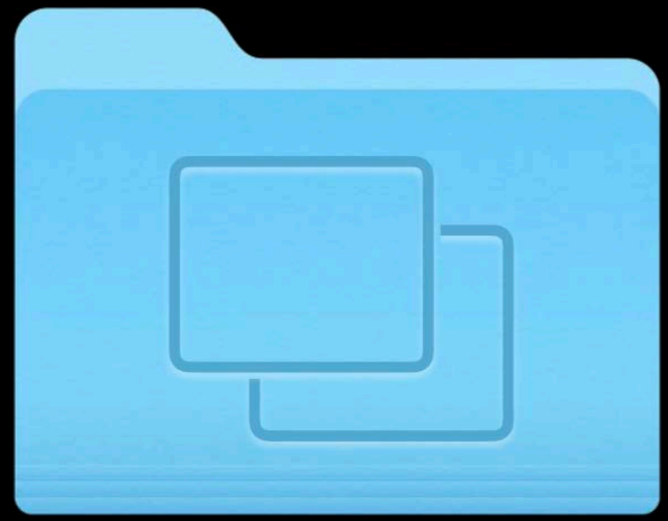
Simple app resource management



Asset Catalog

Simple app resource management

App icon

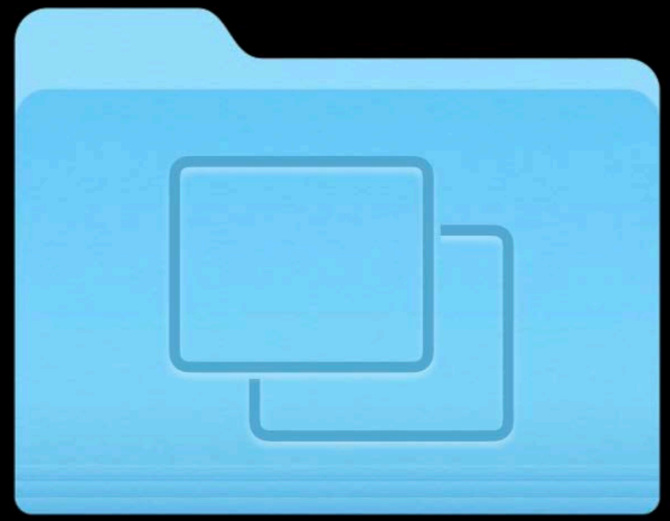


Asset Catalog

Simple app resource management

App icon

Device and scale variants



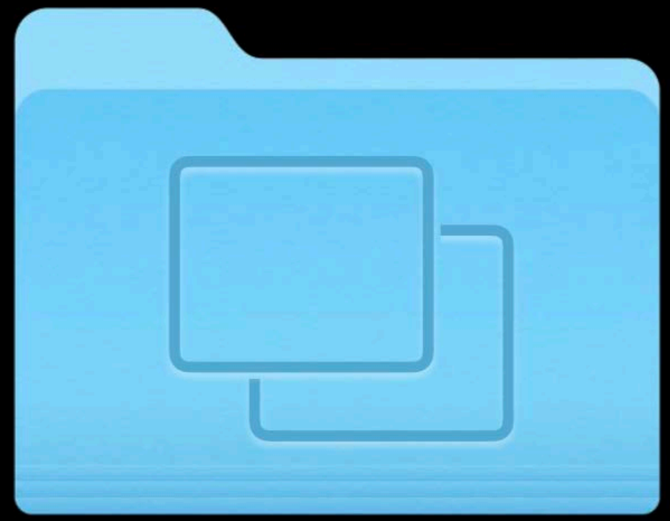
Asset Catalog

Simple app resource management

App icon

Device and scale variants

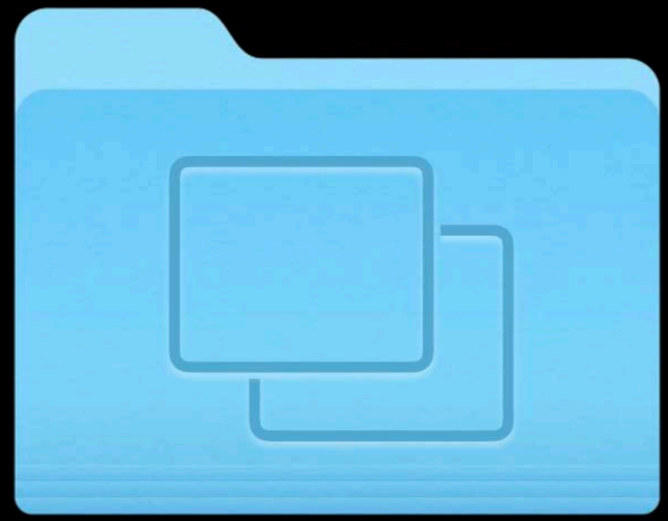
On demand resources



Asset Catalog

Storage and performance benefits

Storage efficiency

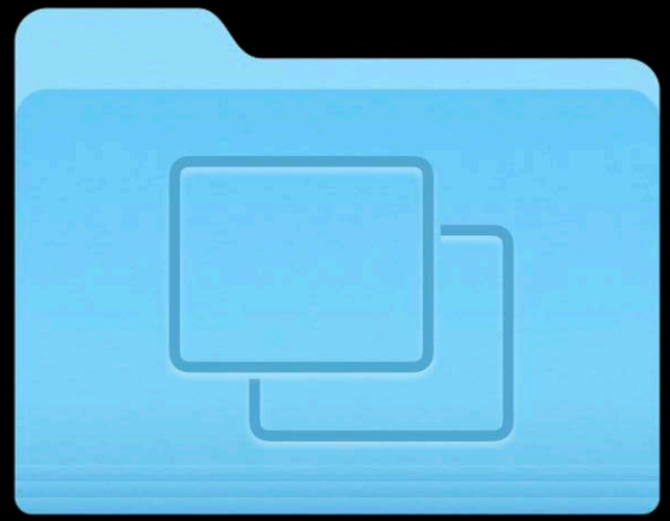


Asset Catalog

Storage and performance benefits

Storage efficiency

- On-disk foot print

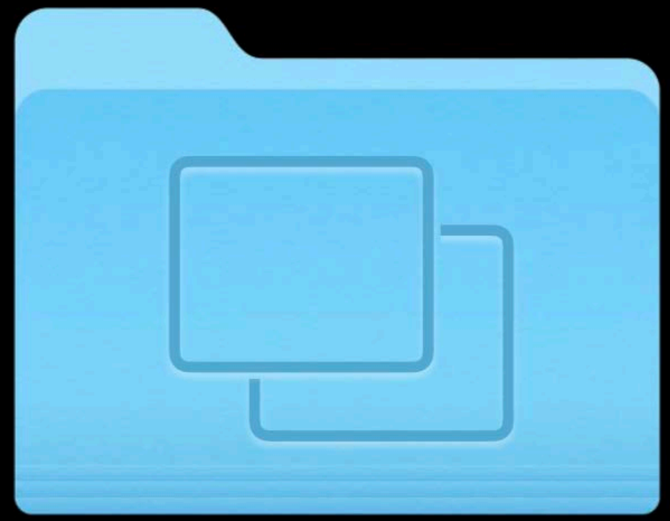


Asset Catalog

Storage and performance benefits

Storage efficiency

- On-disk foot print
- App slicing (iOS)



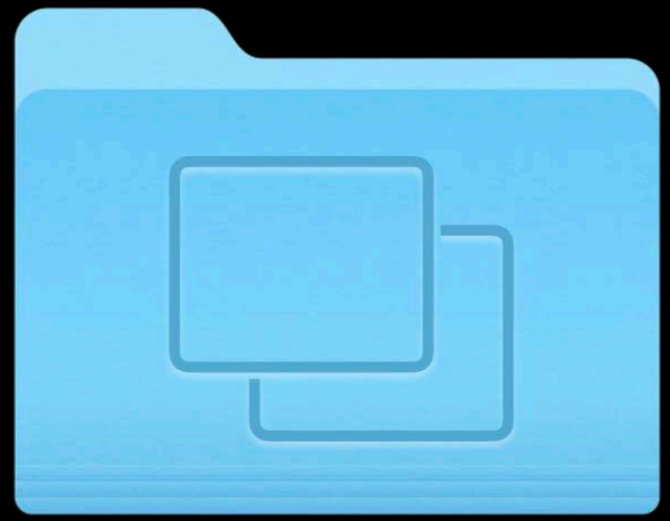
Asset Catalog

Storage and performance benefits

Storage efficiency

- On-disk foot print
- App slicing (iOS)

Performance



Asset Catalog

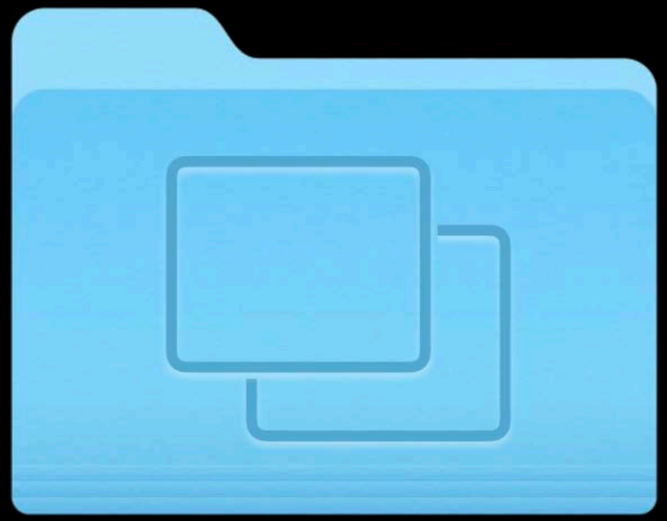
Storage and performance benefits

Storage efficiency

- On-disk foot print
- App slicing (iOS)

Performance

- Image loading



Asset Catalog

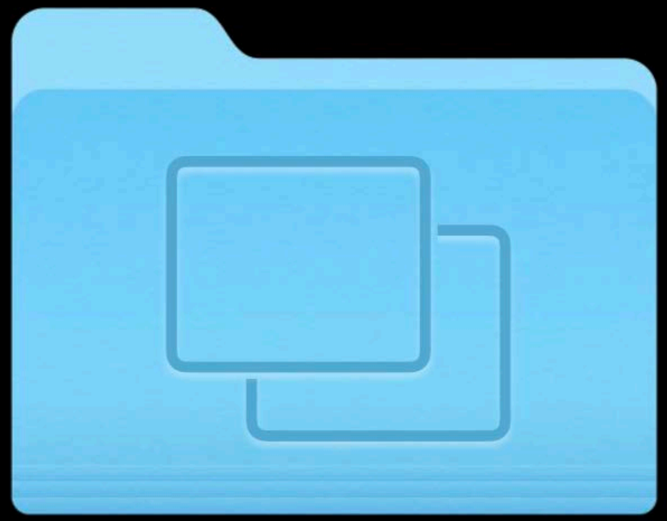
Storage and performance benefits

Storage efficiency

- On-disk foot print
- App slicing (iOS)

Performance

- Image loading
- App launch



Asset Catalog

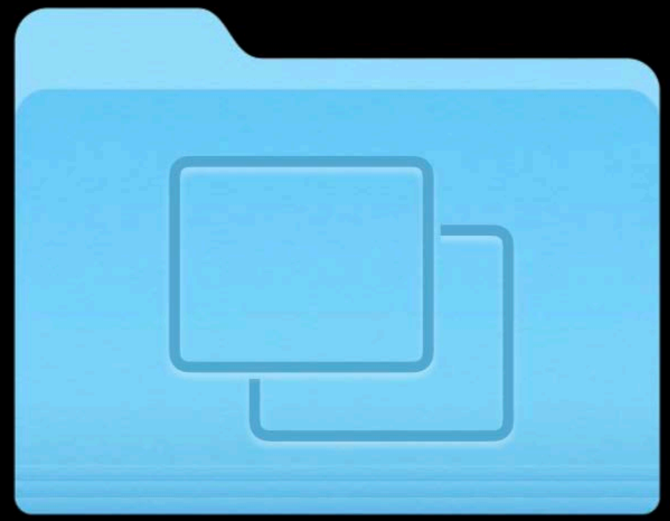
Storage and performance benefits

Storage efficiency

- On-disk foot print
- App slicing (iOS)

Performance

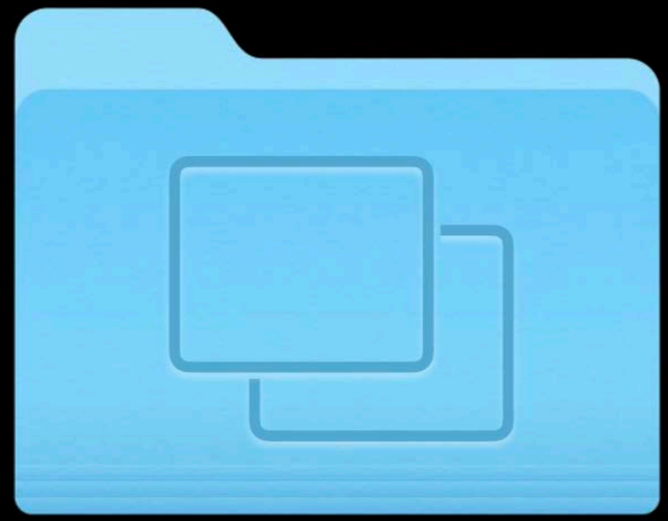
- Image loading
- App launch (up to 10% faster!)



Asset Catalog

Easy GPU based compression

Lossless by default

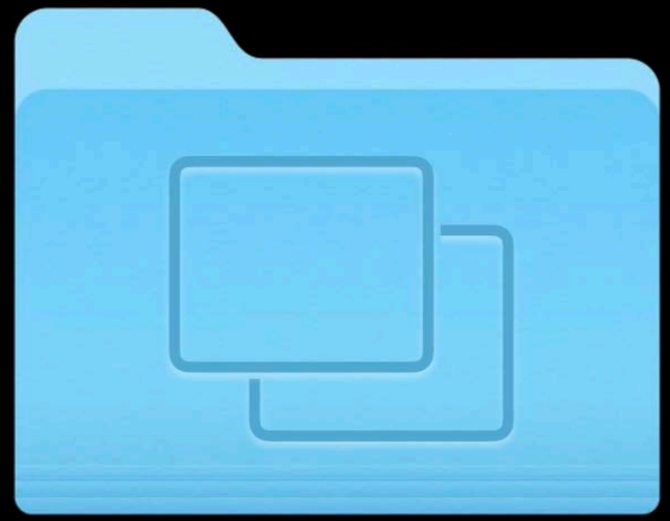


Asset Catalog

Easy GPU based compression

Lossless by default

Lossy image compression available



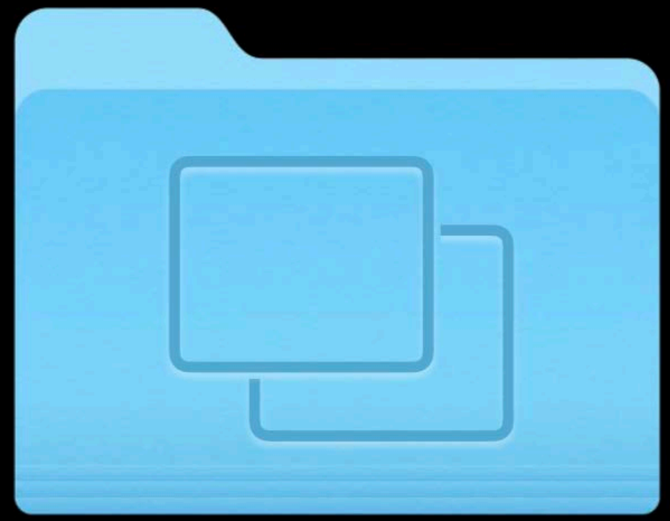
Asset Catalog

Easy GPU based compression

Lossless by default

Lossy image compression available

- Hardware accelerated decompression



Asset Catalog

Easy GPU based compression

Lossless by default

Lossy image compression available

- Hardware accelerated decompression
- Lower memory footprint

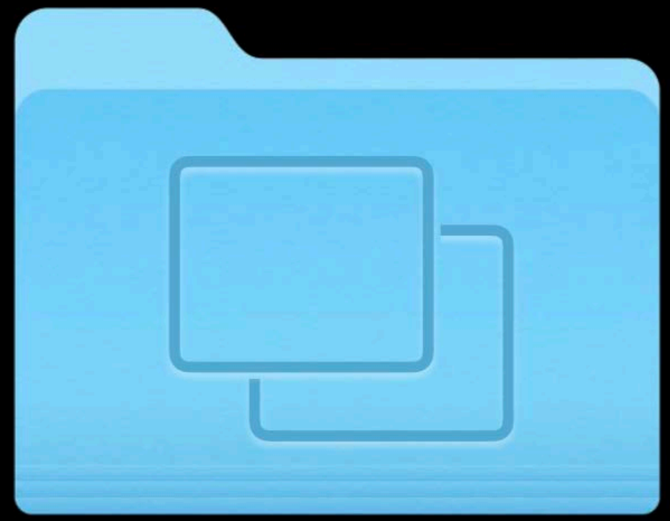


Image Assets

Growing in size

JPEG Size: 24.6MB

HEIC Size: 17.9MB

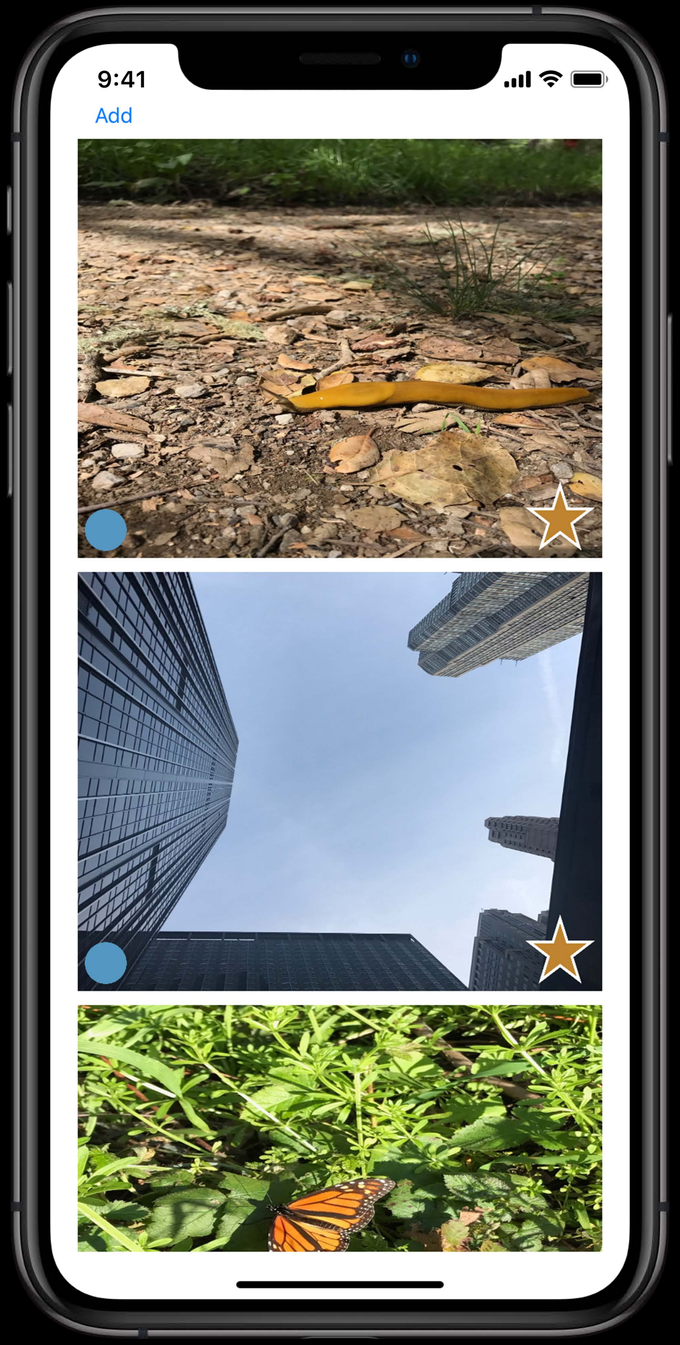
Image Assets

Growing in size

JPEG Size: 24.6MB

HEIC Size: 17.9MB

AssetCatalog: 14.9MB



Use HEIC and Asset Catalogs to reduce footprint

File System Metadata

File System Metadata

Modifying a plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>last_app_launch</key>
    <date>2019-06-07T07:26:49Z</date>
</dict>
</plist>
```

File System Metadata

Modifying a plist

On launch our demo app reads and updates this plist

File System Metadata

Modifying a plist

On launch our demo app reads and updates this plist

- One read

File System Metadata

Modifying a plist

On launch our demo app reads and updates this plist

- One read
- Three writes

File System Metadata

Modifying a plist

On launch our demo app reads and updates this plist

- One read
- Three writes
- `fsync`

File System Metadata

Why three writes instead of one?

File system metadata writes happen as a result of

- Creating a file
- Deleting a file
- Renaming a file
- Updating a file
- And other file system operations

File System Metadata

What is file system metadata?

Name

File System Metadata

What is file system metadata?

Name

Size

File System Metadata

What is file system metadata?

Name

Size

Location

File System Metadata

What is file system metadata?

Name

Size

Location

Dates

File System Metadata

What is file system metadata?

Name

Size

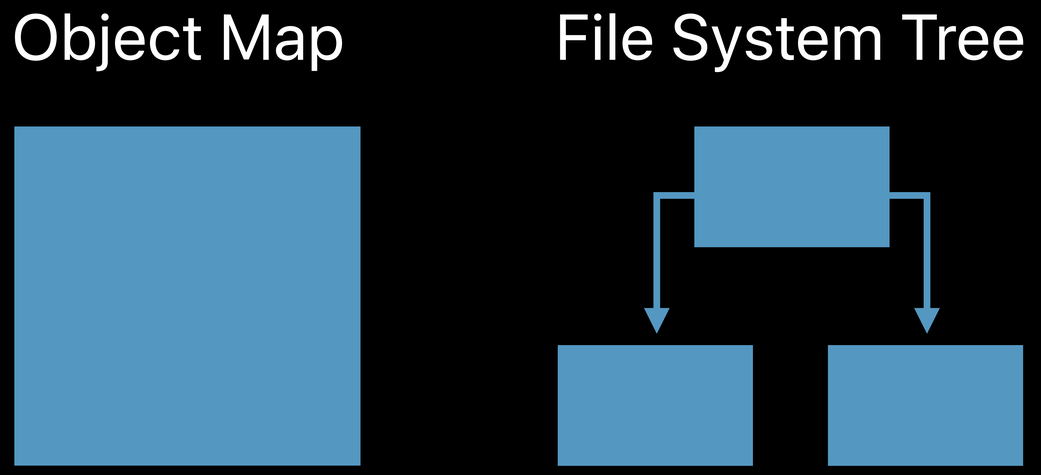
Location

Dates

And more

File System Metadata

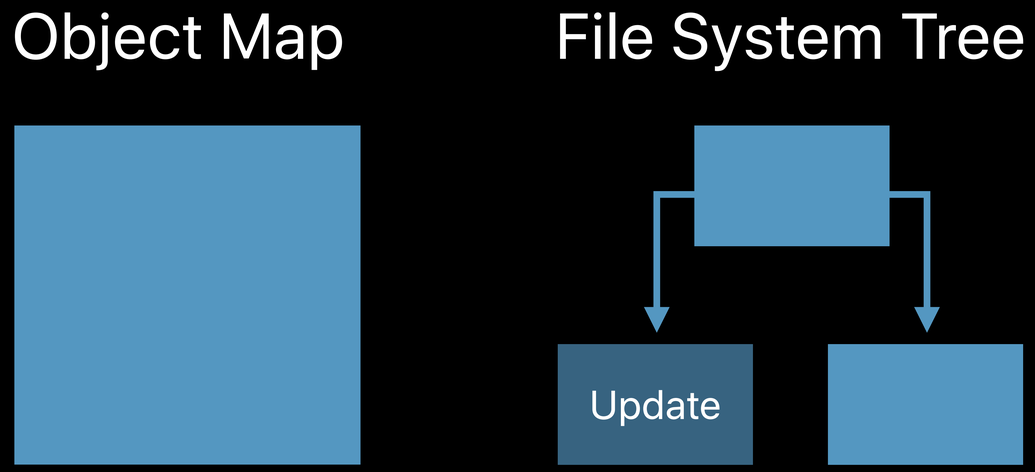
Writing a 240 byte NSDictionary to a file



File System Metadata

Writing a 240 byte NSDictionary to a file

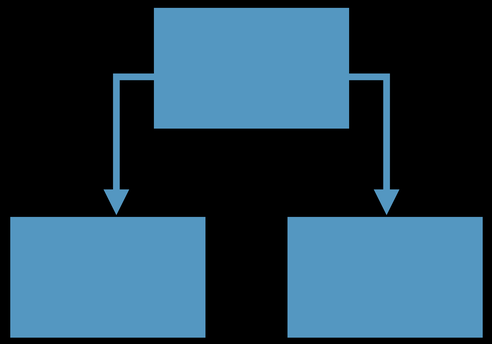
Update file system node



File System Metadata

Writing a 240 byte NSDictionary to a file

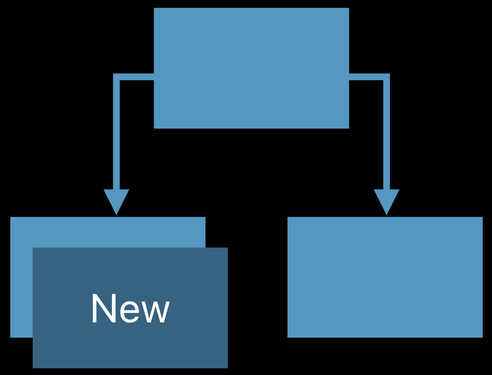
File System Tree



File System Metadata

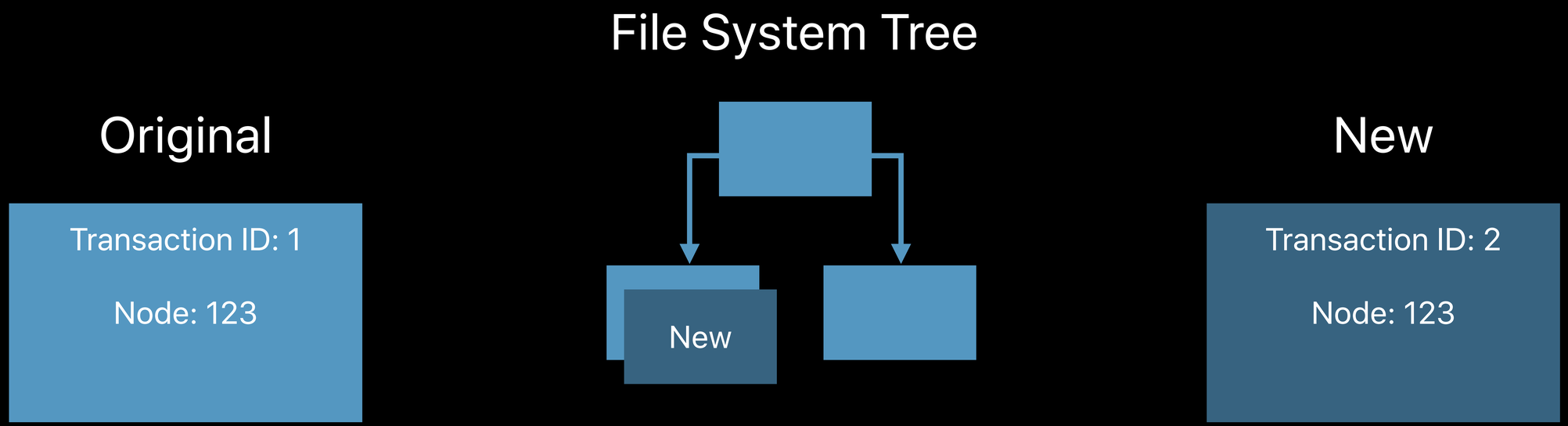
Writing a 240 byte NSDictionary to a file

File System Tree



File System Metadata

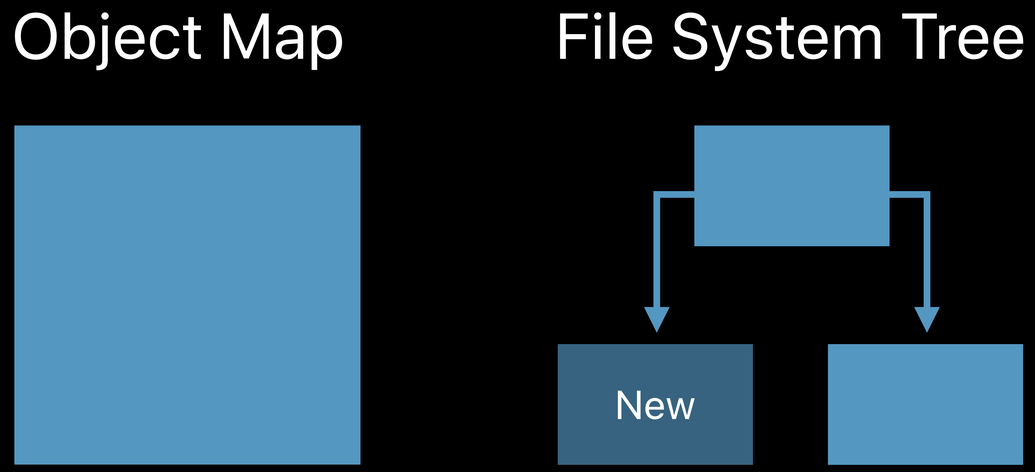
Writing a 240 byte NSDictionary to a file



File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node

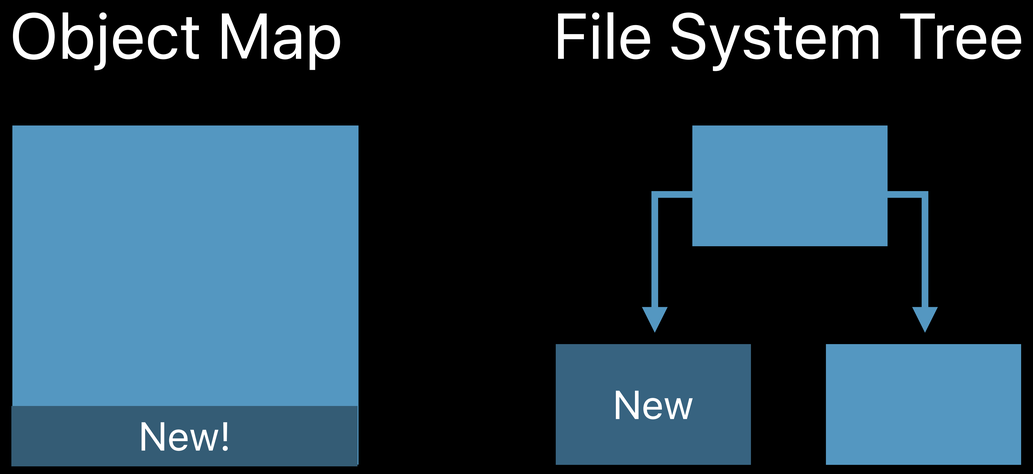


File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node

Update object map



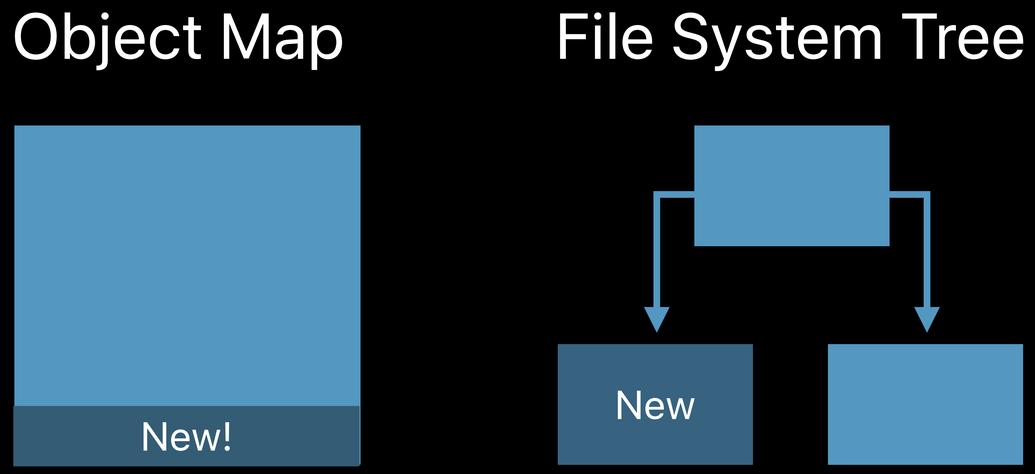
File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node

Update object map

Total Metadata: 8K



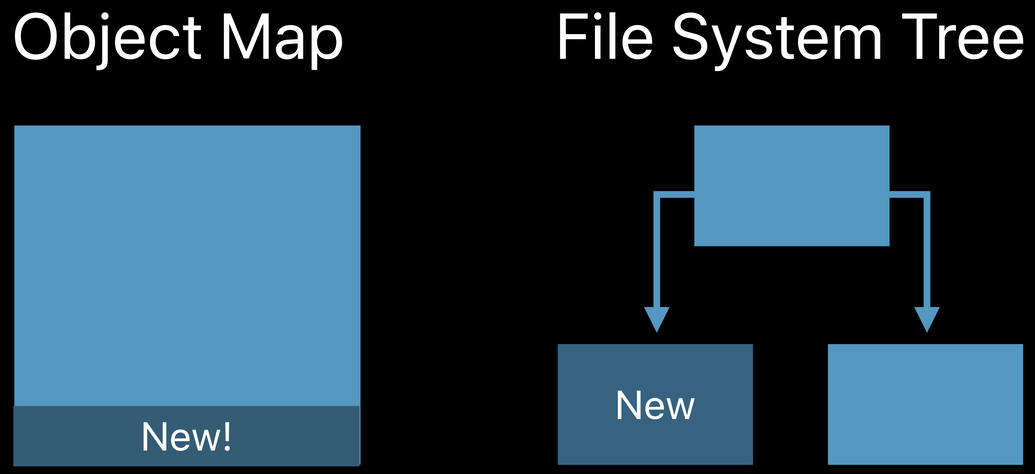
File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node (4K)

Update object map

Total Metadata: 8K



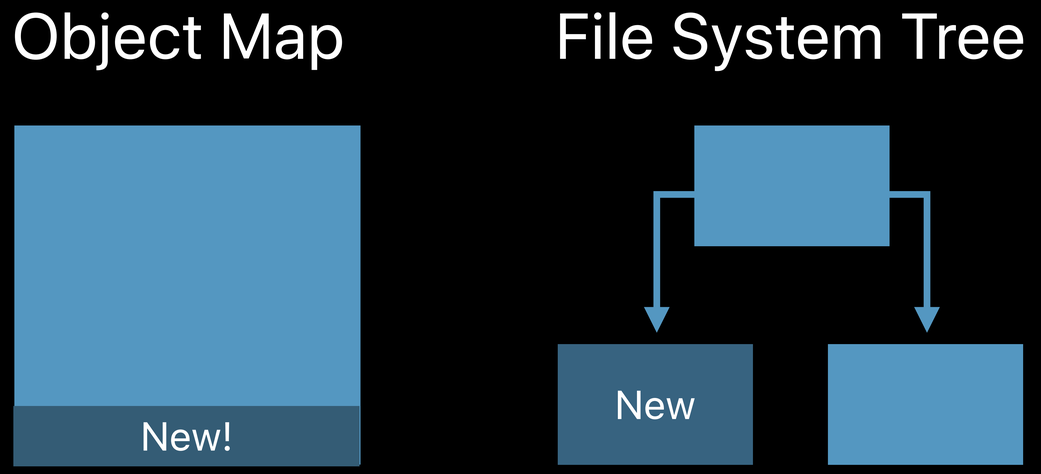
File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node (4K)

Update object map (4K)

Total Metadata: 8K



File System Metadata

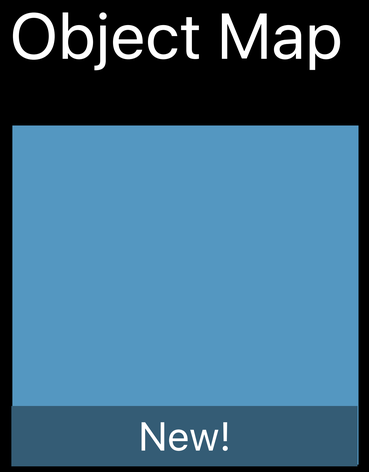
Writing a 240 byte NSDictionary to a file

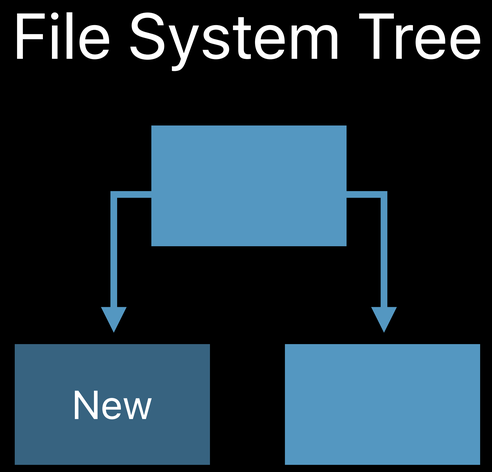
Update file system node (4K)

Update object map (4K)

Total Metadata: 8K

File itself





File System Metadata

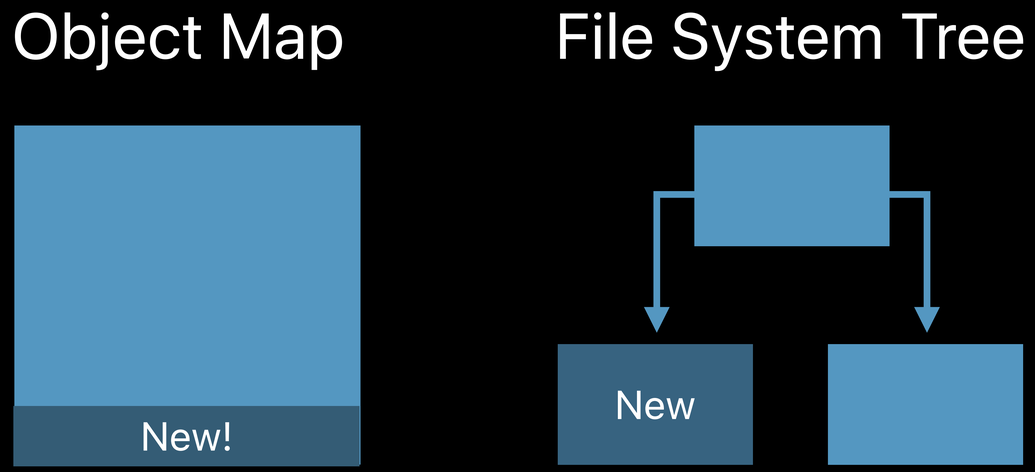
Writing a 240 byte NSDictionary to a file

Update file system node (4K)

Update object map (4K)

Total Metadata: 8K

File itself (4K)



File System Metadata

Writing a 240 byte NSDictionary to a file

Update file system node (4K)

Update object map (4K)

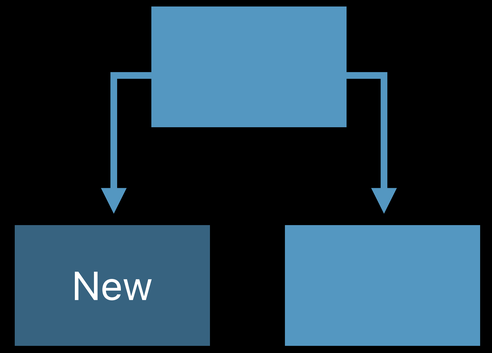
Total Metadata: 8K

File itself (4K)

Total data: 12K

Object Map

File System Tree



File System Metadata

Book keeping workload for common operations

Create a file: 8K

File System Metadata

Book keeping workload for common operations

Create a file: 8K

Delete a file: 8K

File System Metadata

Book keeping workload for common operations

Create a file: 8K

Delete a file: 8K

Rename a file: 16K

File System Metadata

Book keeping workload for common operations

Create a file: 8K

Delete a file: 8K

Rename a file: 16K

Modifying a file: 8K

File system metadata
updates are not free.

File System Metadata

Efficient non-persistent files

Create a file

File System Metadata

Efficient non-persistent files

Create a file

Keep it open and unlinked

File System Metadata

Efficient non-persistent files

Create a file

Keep it open and unlinked

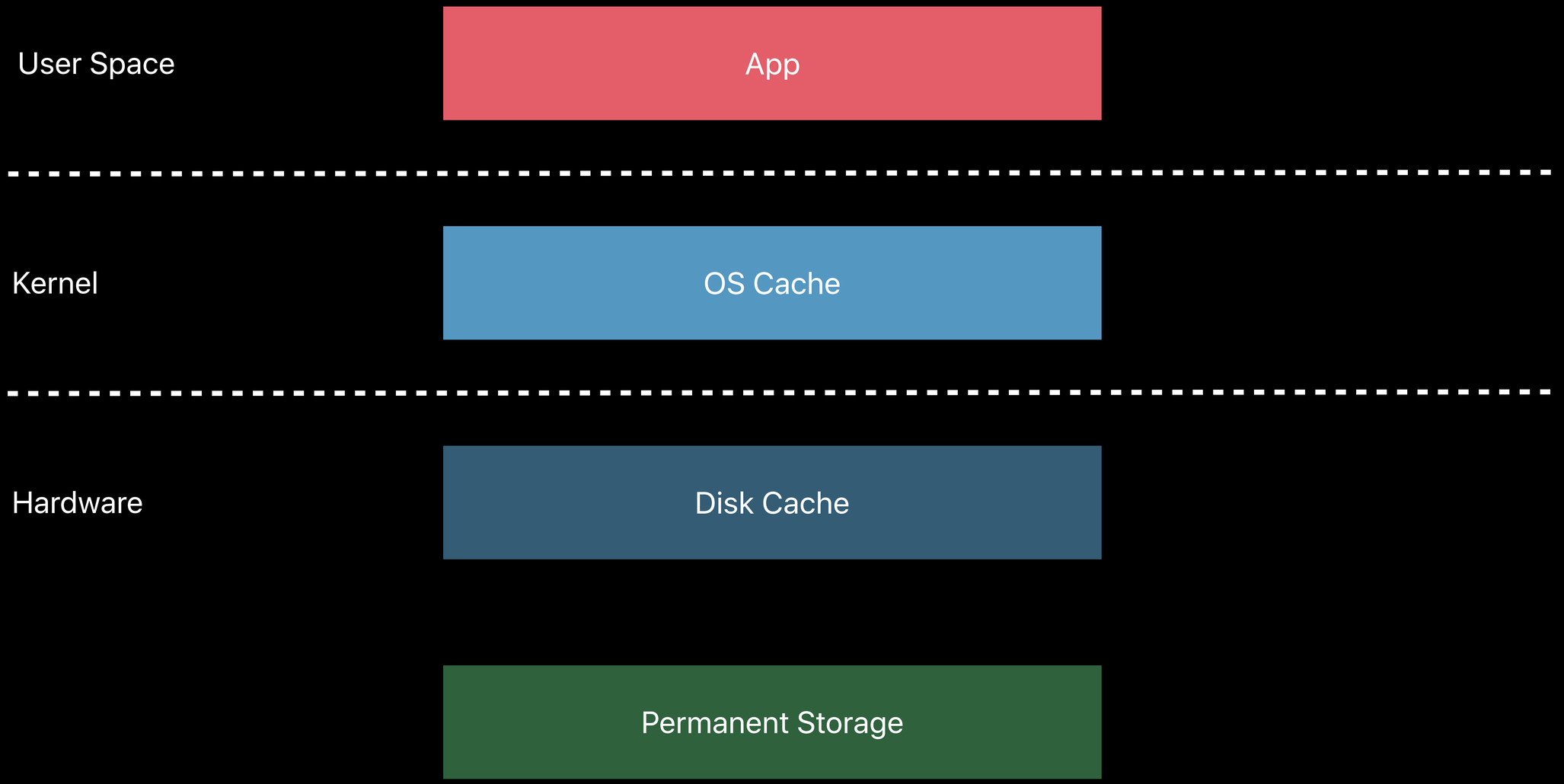
Do not call `fsync()`

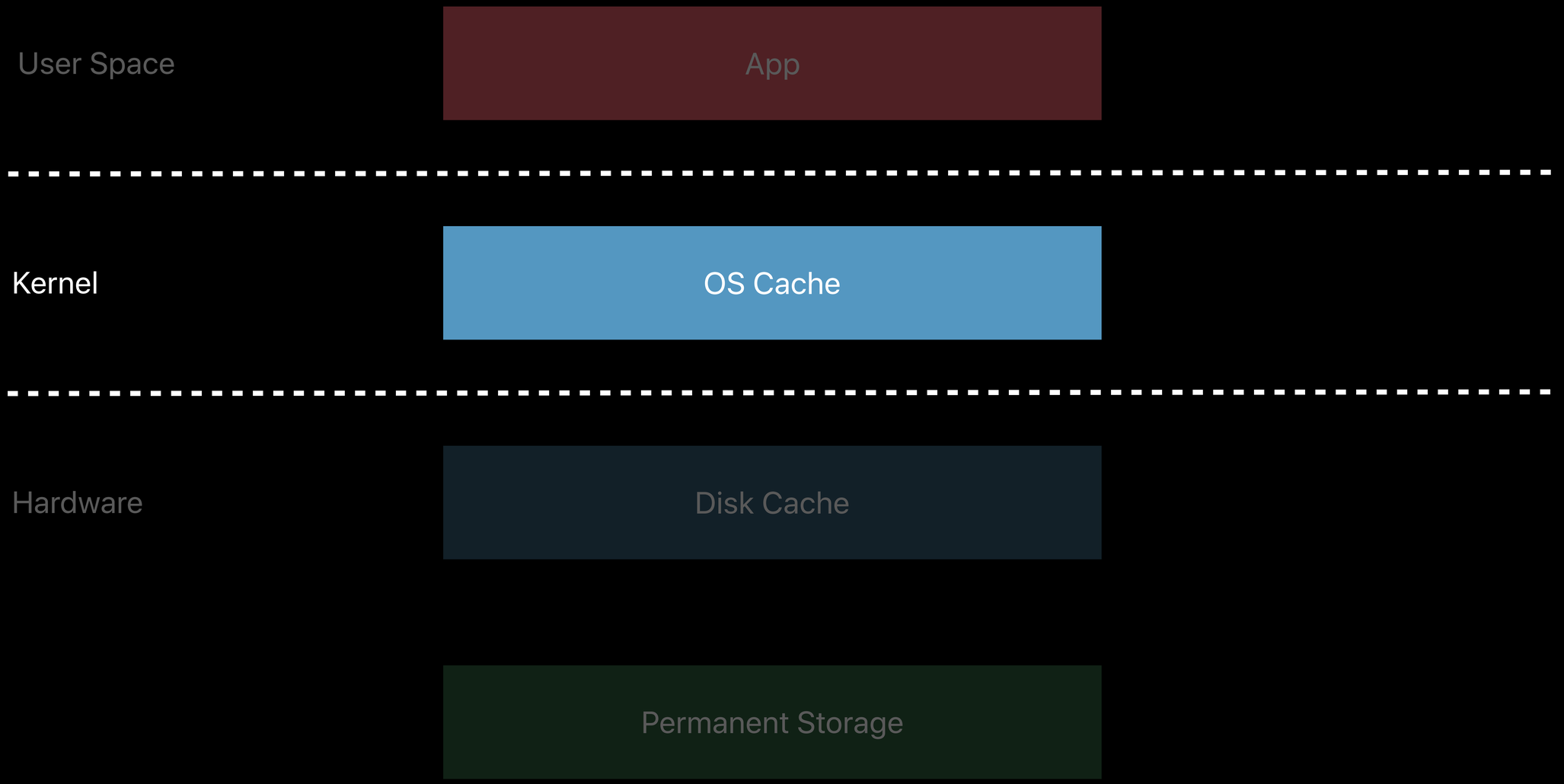
File System Metadata

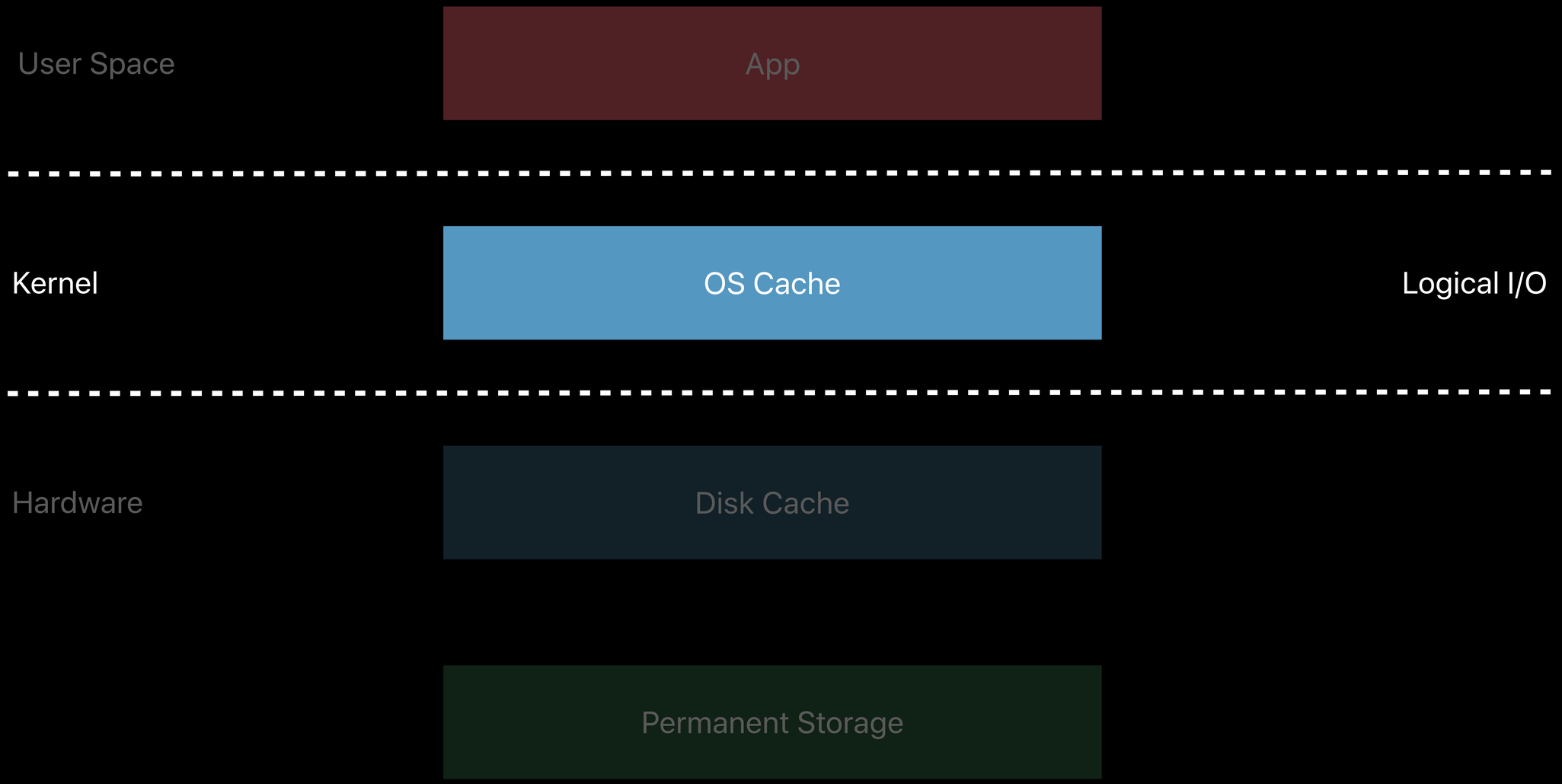
More information on APFS

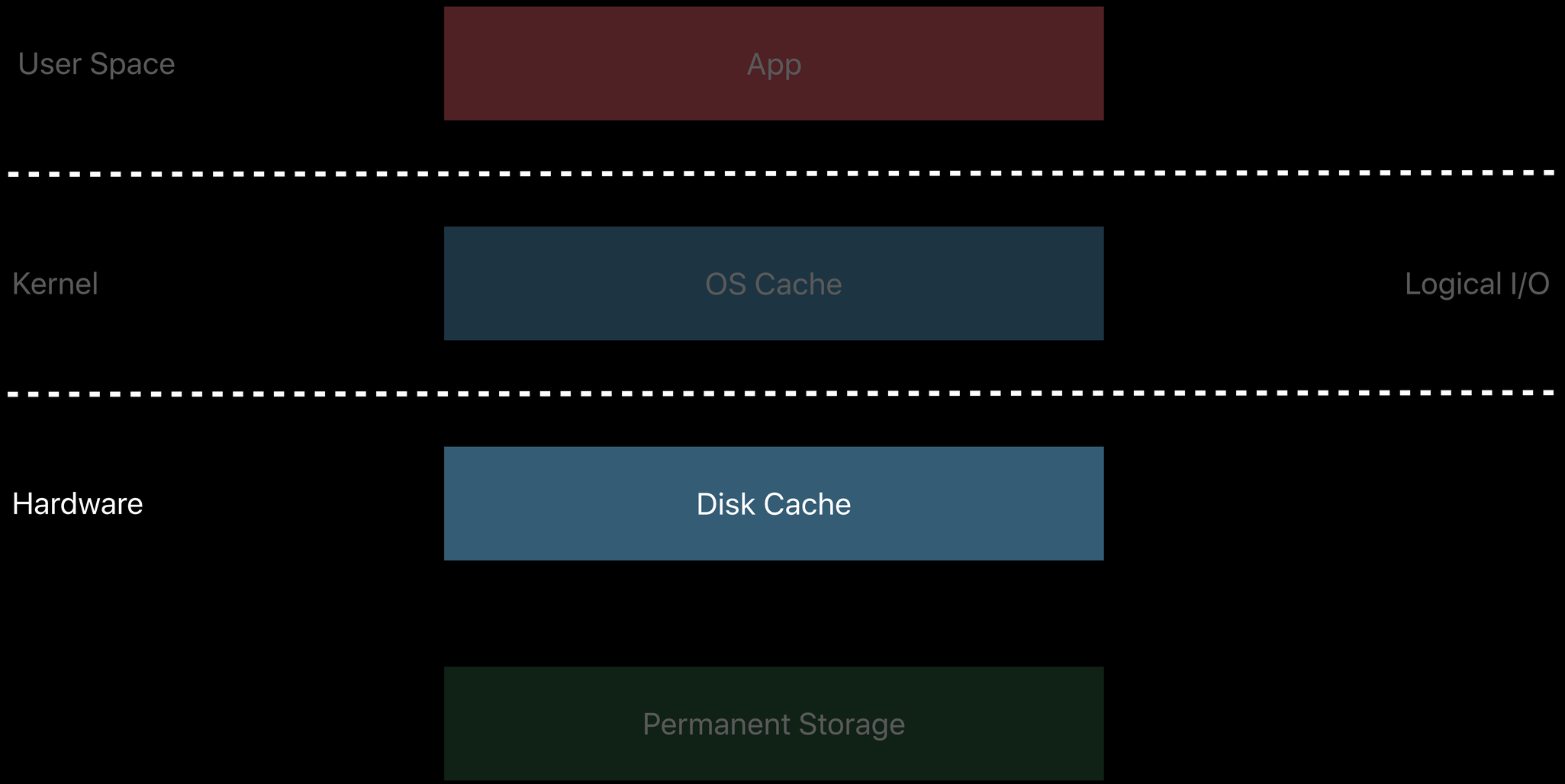
developer.apple.com/support/apple-file-system/Apple-File-System-Reference.pdf

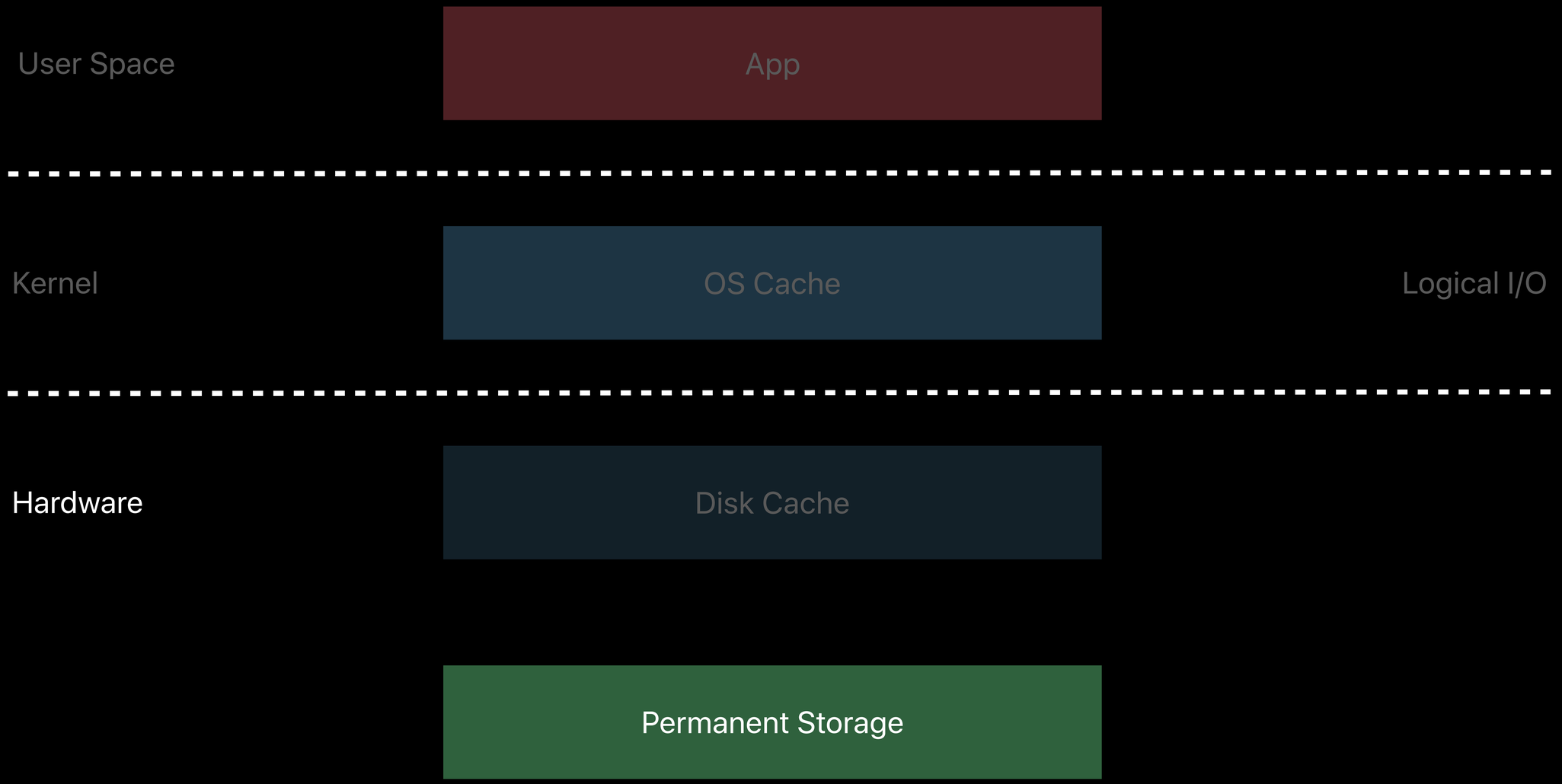
Syncing to Disk

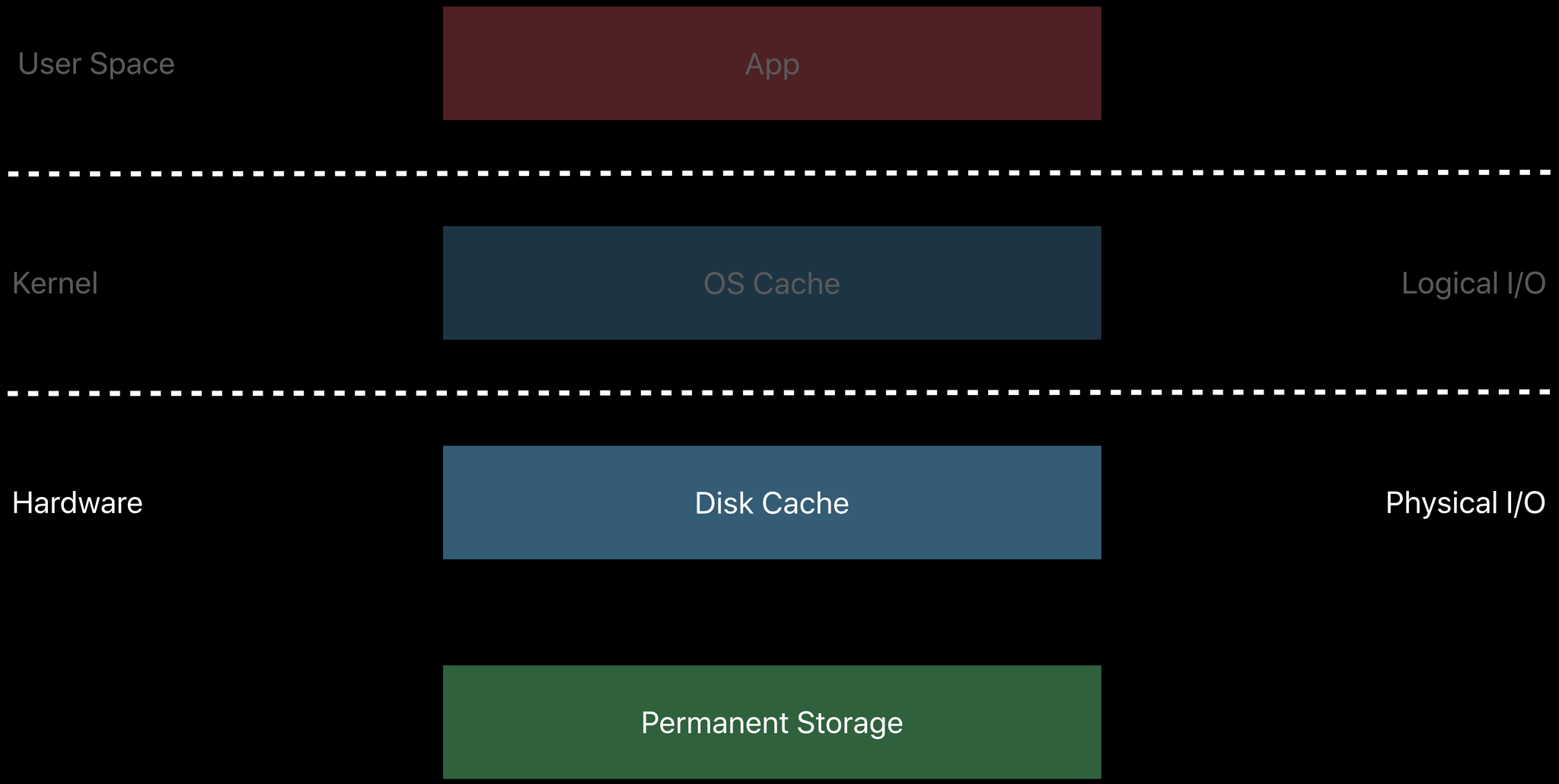

User Space
App
Kernel
OS Cache
Hardware
Disk Cache
Permanent Storage

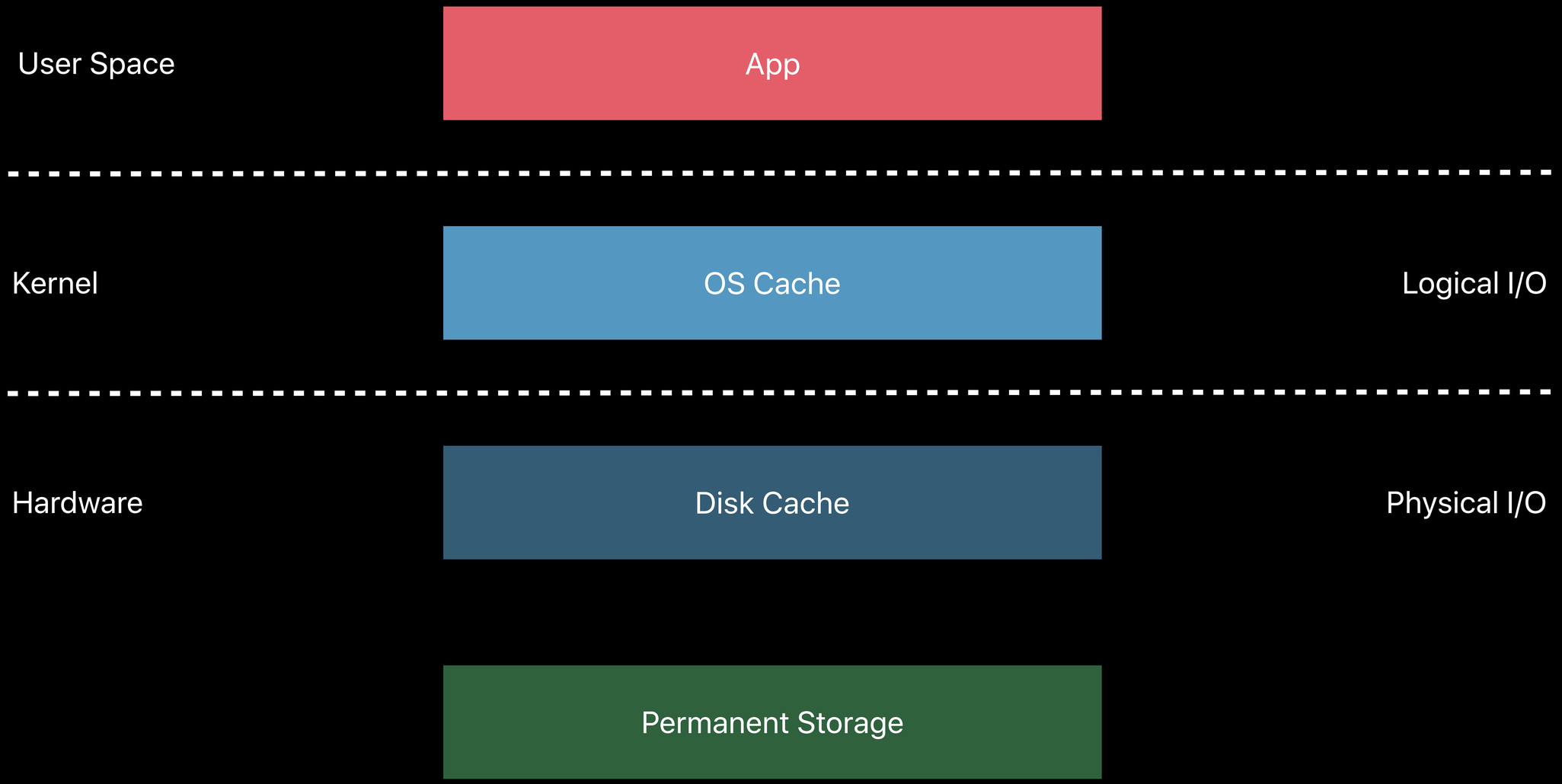

User Space
App
Kernel
OS Cache
Hardware
Disk Cache
Permanent Storage


User Space
App
Kernel
OS Cache
Logical I/O
Hardware
Disk Cache
Permanent Storage


User Space
App
Kernel
OS Cache
Logical I/O
Hardware
Disk Cache
Permanent Storage

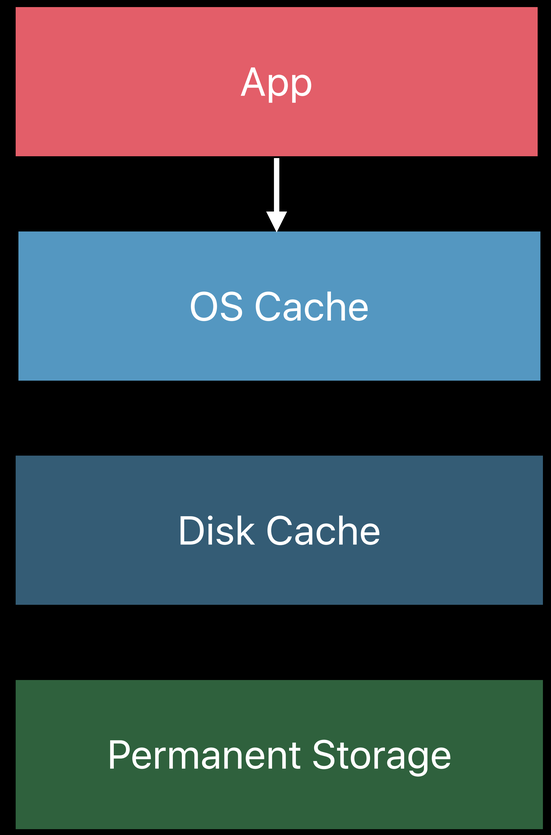

User Space
App
Kernel
OS Cache
Logical I/O
Hardware
Disk Cache
Permanent Storage


User Space
App
Kernel
OS Cache
Logical I/O
Hardware
Disk Cache
Physical I/O
Permanent Storage


User Space
App
Kernel
OS Cache
Logical I/O
Hardware
Disk Cache
Physical I/O
Permanent Storage

`fsync()`

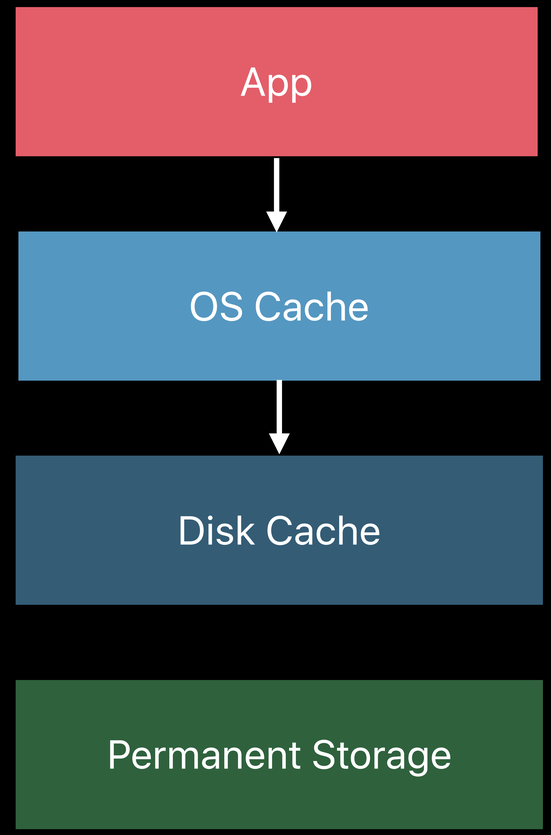
Flushes cached data to disk



`fsync()`

Flushes cached data to disk

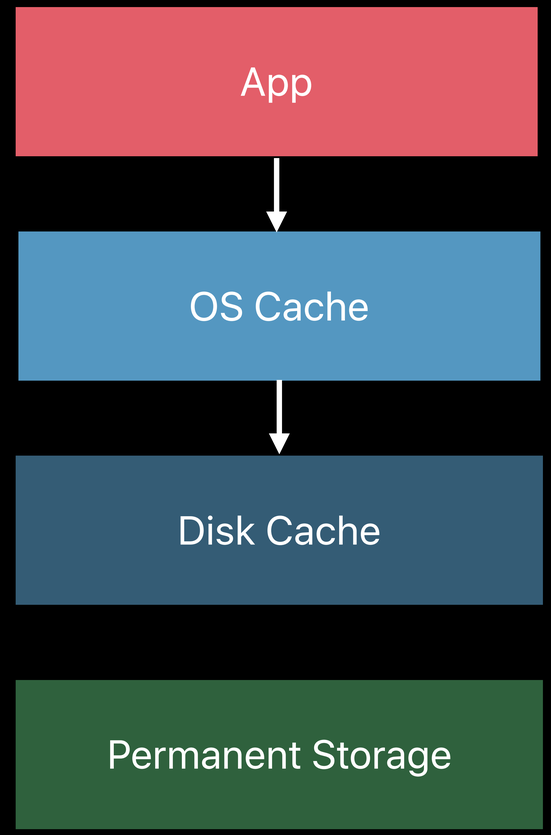
- Moves data from the OS cache to the disk cache



`fsync()`

Flushes cached data to disk

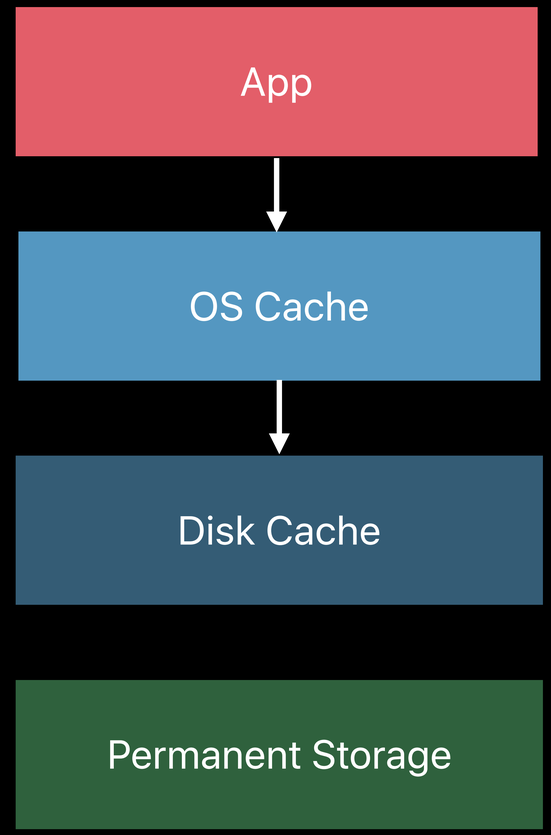
- Moves data from the OS cache to the disk cache
- Data may not be written to permanent storage immediately



`fsync()`

Flushes cached data to disk

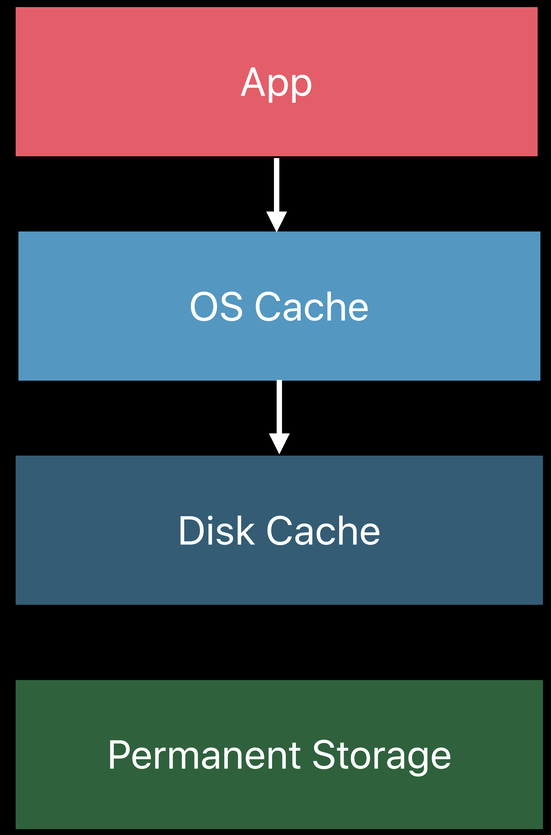
- Moves data from the OS cache to the disk cache
- Data may not be written to permanent storage immediately
- Does not guarantee write ordering



`fsync()`

Flushes cached data to disk

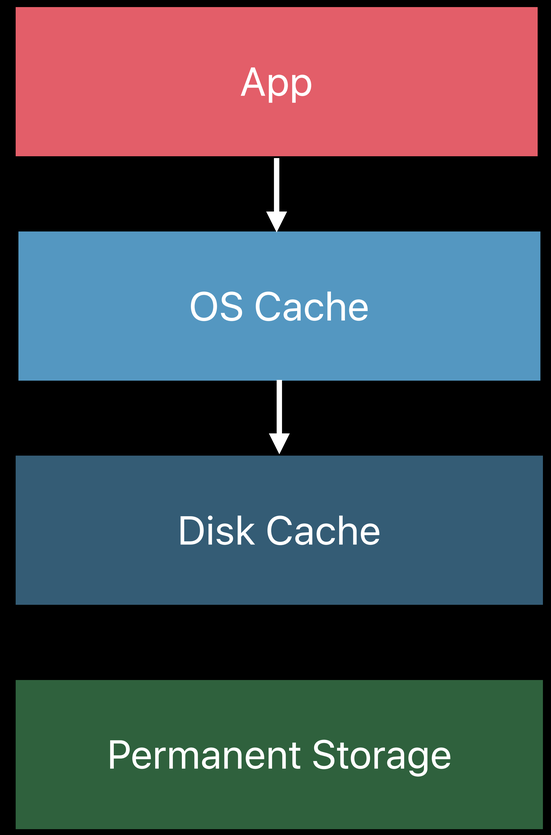
- Moves data from the OS cache to the disk cache
- Data may not be written to permanent storage immediately
- Does not guarantee write ordering
- Expensive



`fsync()`

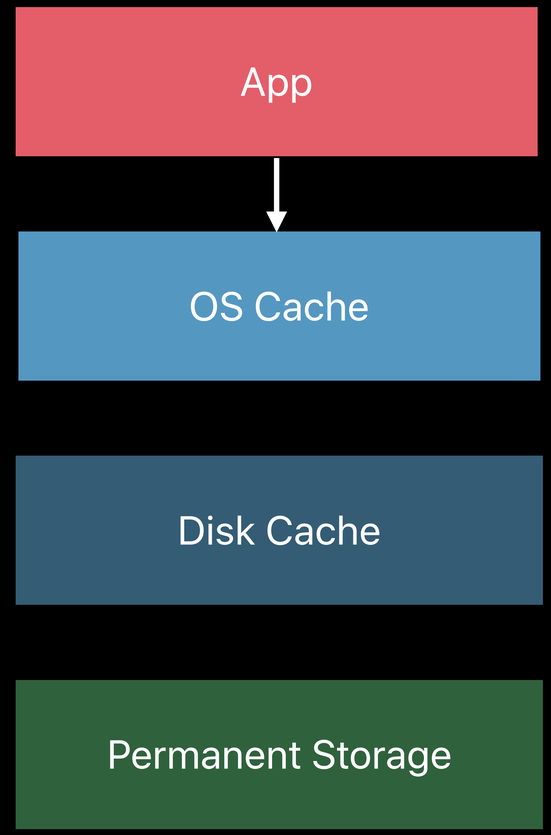
Flushes cached data to disk

- Moves data from the OS cache to the disk cache
- Data may not be written to permanent storage immediately
- Does not guarantee write ordering
- Expensive
- Done periodically by the OS



F_FULLFSYNC

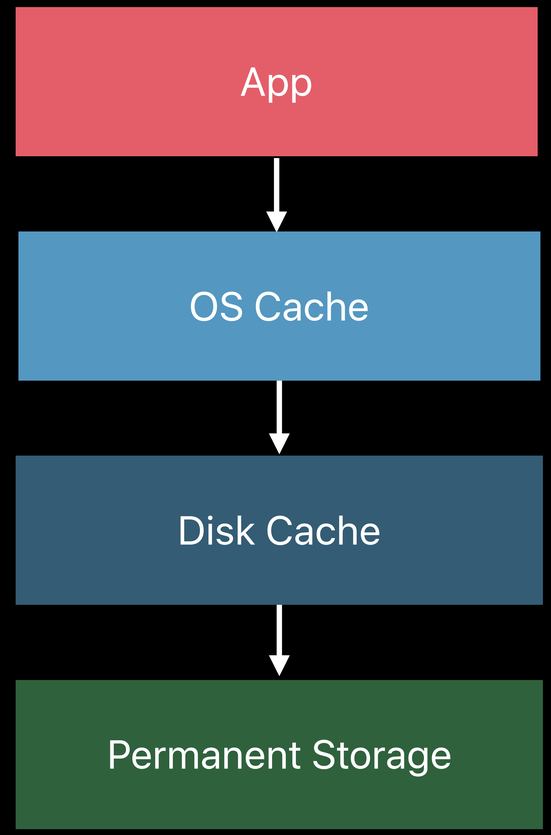
Drive flush its cache to disk



F_FULLFSYNC

Drive flush its cache to disk

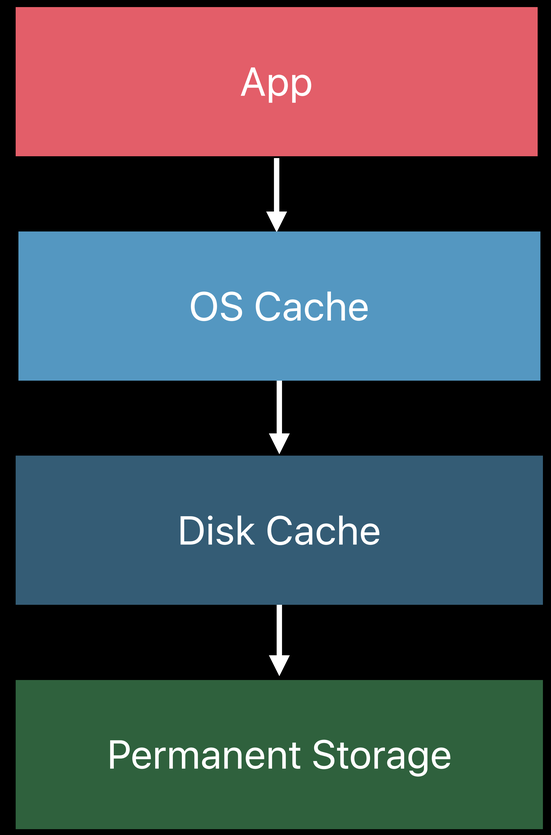
- Causes all data in disk cache to be flushed



F_FULLFSYNC

Drive flush its cache to disk

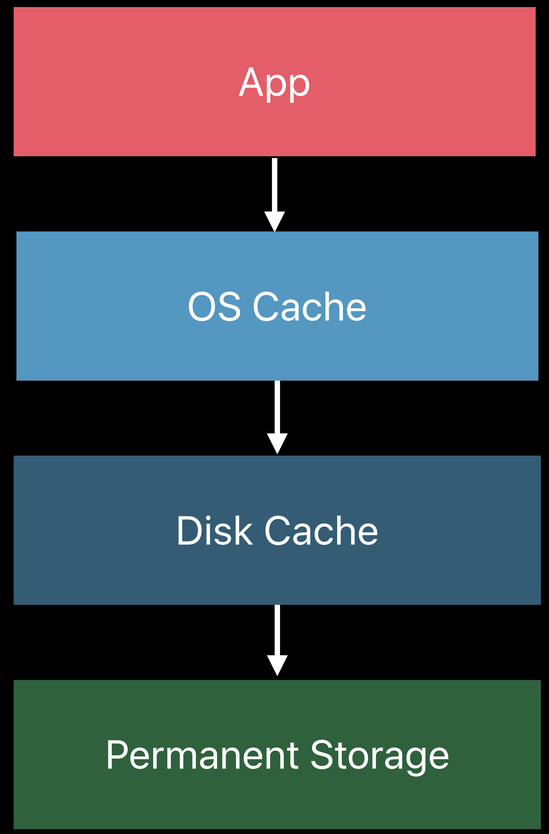
- Causes all data in disk cache to be flushed
- Expensive



F_FULLFSYNC

Drive flush its cache to disk

- Causes all data in disk cache to be flushed
- Expensive
- Done periodically by OS



F_BARRIERFSYNC

Enforces I/O ordering

`F_BARRIERFSYNC`

Enforces I/O ordering

- `fsync()` with a barrier

`F_BARRIERFSYNC`

Enforces I/O ordering

- `fsync()` with a barrier

Much less expensive than `F_FULLFSYNC`

Enforcing I/O Ordering

`F_BARRIERFSYNC`

Ensures I/O ordering for the specified data

`F_FULLFSYNC`

Indiscriminately commits all disk cache data to permanent storage

Serialized Data Files

Plists, XML, and JSON

The good

- Convenient

Plists, XML, and JSON

The good

- Convenient
- Excellent for infrequently written data

Plists, XML, and JSON

The good

- Convenient
- Excellent for infrequently written data
- Easy to parse

Plists, XML, and JSON

The trade off

Plists, XML, and JSON

The trade off

- Entire file must be rewritten for every change

Plists, XML, and JSON

The trade off

- Entire file must be rewritten for every change
- Scales poorly

Plists, XML, and JSON

The trade off

- Entire file must be rewritten for every change
- Scales poorly
- Easy to misuse

Plists, XML, and JSON

The trade off

- Entire file must be rewritten for every change
- Scales poorly
- Easy to misuse
- Metadata intensive

Plists, XML, and JSON

The trade off

- Entire file must be rewritten for every change
- Scales poorly
- Easy to misuse
- Metadata intensive

Not meant to be a database

NSDictionary

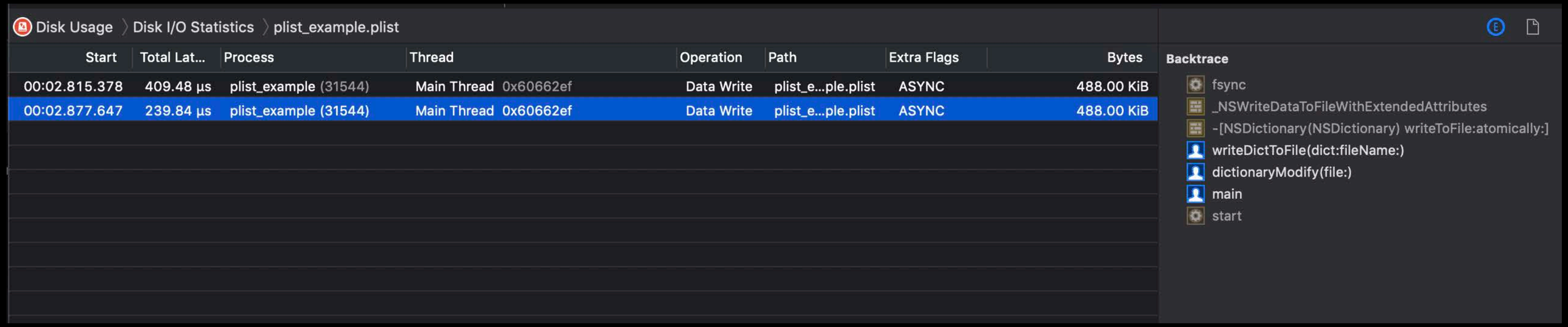
Create, read, and modify example

Filesystem Activity 〉 Filesystem Events

Start	Duration	Process	Thread	Narrative
00:02.814.992	166.06 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed open on path plist_example.plist
00:02.815.177	153.14 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) perfomed write on fd 3 with a requested size of 484.46 KiB. Amount of bytes actually processed: 484.46 KiB
00:02.815.332	474.19 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed fsync on fd 3 and path (plist_example.plist)
00:02.815.810	15.83 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed close on fd 3 (plist_example.plist)
00:02.816.509	25.57 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed open on path plist_example.plist
00:02.816.537	2.00 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed fstat64 on fd 3 and path (plist_example.plist)
00:02.816.561	296.53 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) perfomed read on fd 3 with a requested size of 484.46 KiB. Amount of bytes actually processed: 484.46 KiB
00:02.816.859	5.23 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed close on fd 3 (plist_example.plist)
00:02.877.305	164.45 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed open on path plist_example.plist
00:02.877.473	135.80 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) perfomed write on fd 3 with a requested size of 484.46 KiB. Amount of bytes actually processed: 484.46 KiB
00:02.877.611	284.92 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed fsync on fd 3 and path (plist_example.plist)
00:02.877.897	9.04 µs	plist_example (31544)	plist_example (pid: 31544, tid: 0x606...	plist_example (31544) performed close on fd 3 (plist_example.plist)

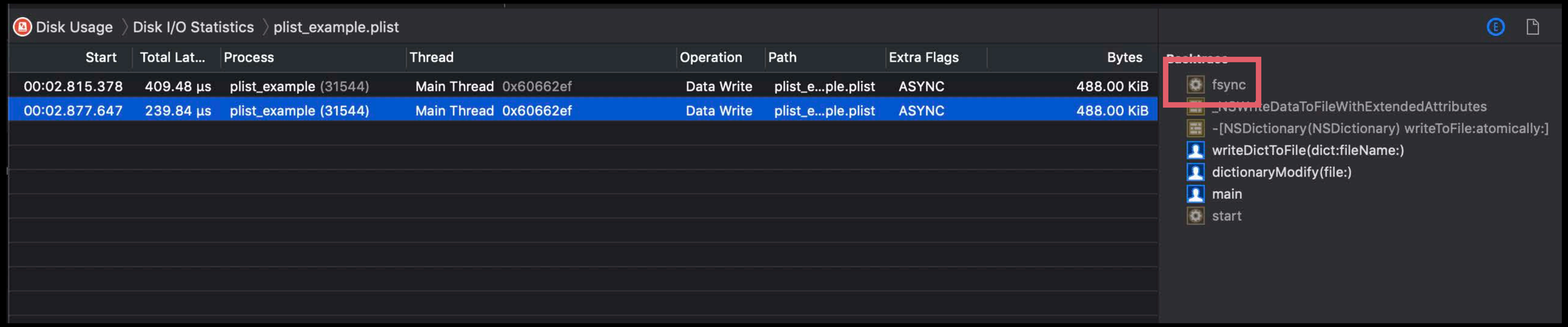
NSDictionary

Modify example



NSDictionary

Modify example



Core Data

Core Data

Core Data management

- Built on SQLite

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships
- Change tracking and notifications

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships
- Change tracking and notifications
- Automatic version tracking and multi-writer conflict resolution
- Automatic connection pooling

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships
- Change tracking and notifications
- Automatic version tracking and multi-writer conflict resolution
  - Automatic connection pooling
- CloudKit integration

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships
- Change tracking and notifications
- Automatic version tracking and multi-writer conflict resolution
  - Automatic connection pooling
- CloudKit integration (new with iOS 13)

Core Data

Core Data management

- Built on SQLite
- Manages object graphs and relationships
- Change tracking and notifications
- Automatic version tracking and multi-writer conflict resolution
  - Automatic connection pooling
- CloudKit integration (new with iOS 13)
- Live queries

Core Data

Core Data management

- Automatic memory management

Core Data

Core Data management

- Automatic memory management
- Statement aggregation in transactions

Core Data

Core Data management

- Automatic memory management
- Statement aggregation in transactions
- Schema migrations

Core Data

Core Data management

- Automatic memory management
- Statement aggregation in transactions
- Schema migrations
- Denormalization

Core Data

Core Data management

- Automatic memory management
- Statement aggregation in transactions
- Schema migrations
- Denormalization (new with iOS 13)

Core Data

Core Data management

- Automatic memory management
- Statement aggregation in transactions
- Schema migrations
- Denormalization (new with iOS 13)
- And so much more!

50–70%

Reduction in code to support the model layer

SQLite

Connections

Journaling

Transactions

File size and privacy

Partial indexes

Connections

Opening and Closing Connections

Can cause expensive operations!

- Consistency checking
- Journal recovery
- Journal checkpointing

Opening and Closing Connections

Can cause expensive operations!

- Consistency checking
- Journal recovery
- Journal checkpointing

Recommended usage model

- Keep connections open as long as possible
- Close only when necessary
- Pool connections on multi-threaded processes

Journaling

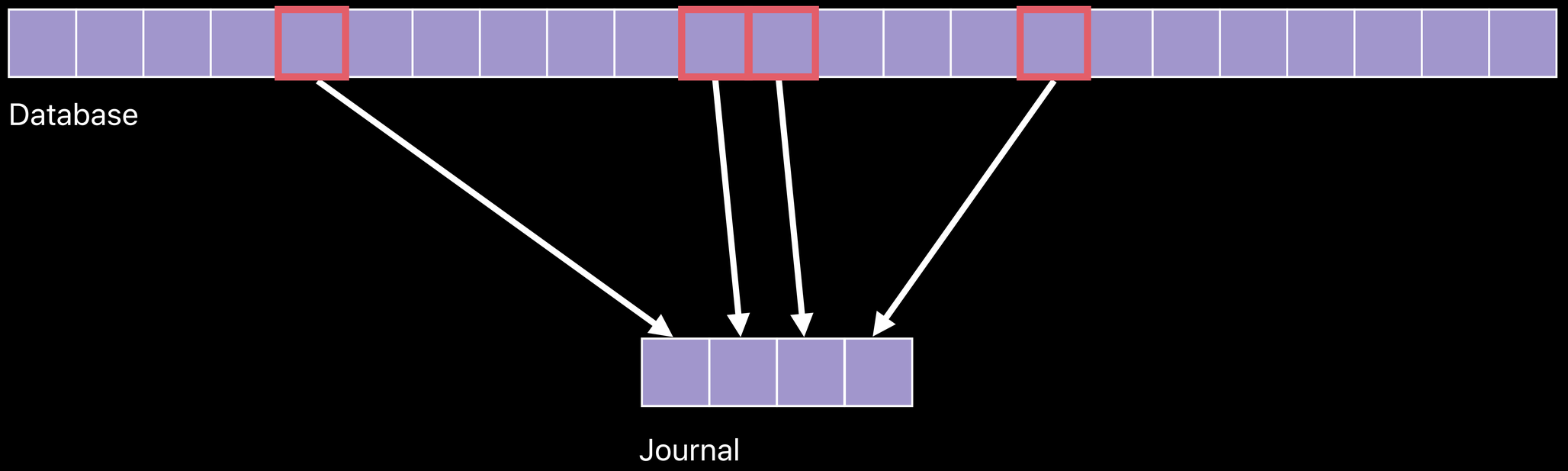
Delete Mode Journaling

Default SQLite journaling mode

How Does Delete Mode Journaling Work?

Database

How Does Delete Mode Journaling Work?



How Does Delete Mode Journaling Work?

Database

Journal

How Does Delete Mode Journaling Work?

Database

WAL Mode Journaling Reduces Writes

```
PRAGMA journal_mode=WAL;
```

Write Ahead Logging

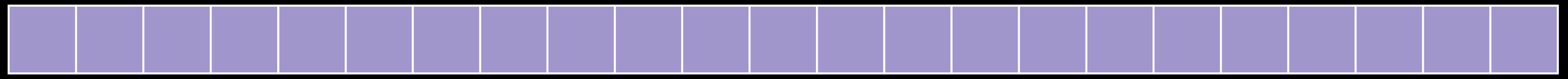
Combines multiple writes to the same page

Uses fewer barriers

Supports multiple readers at the same time as a writer

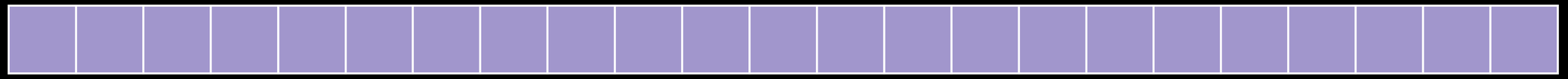
Supports snapshots

How Does WAL Mode Journaling Work?



Database

How Does WAL Mode Journaling Work?



Database

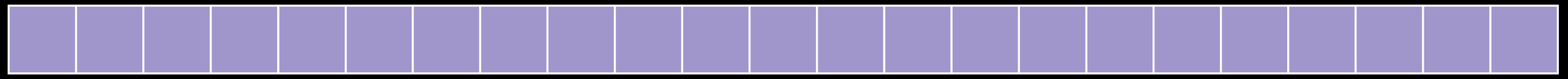
Write Ahead Log

How Does WAL Mode Journaling Work?

Database

Write Ahead Log

How Does WAL Mode Journaling Work?



Database

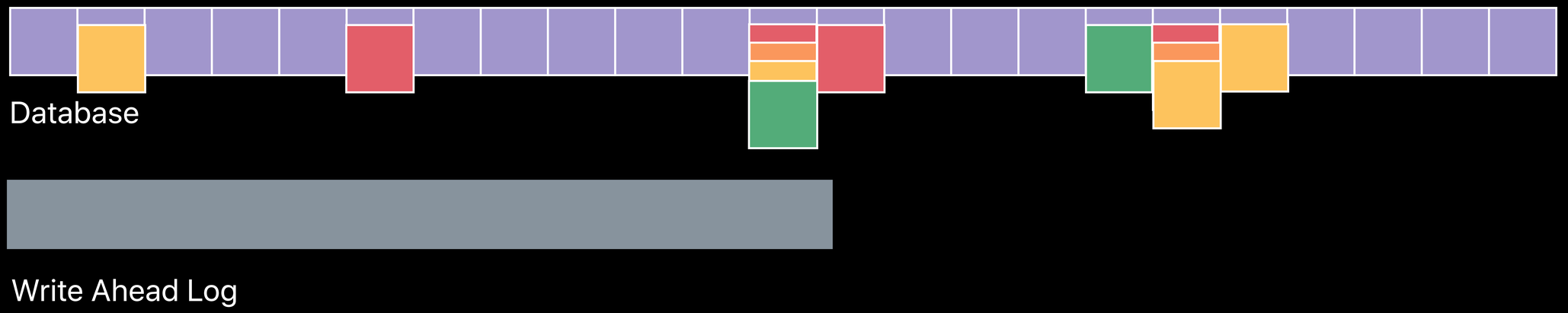
Write Ahead Log

How Does WAL Mode Journaling Work?

Database

Write Ahead Log

How Does WAL Mode Journaling Work?



How Does WAL Mode Journaling Work?

Database

Write Ahead Log

WAL mode is more
efficient for most use cases.

Transactions

Transactions Help Reduce Writes

Use for multiple `INSERT`, `UPDATE`, and `DELETE` statements

Transactions Help Reduce Writes

Use for multiple `INSERT`, `UPDATE`, and `DELETE` statements

Pages that are changed by multiple statements are only written once

Multiple Single Statement Transactions

Transaction #1

Transaction #2

Transaction #3

Multiple Statement Transaction

Transaction #1 with multiple statements

Transactions Help Reduce Writes

Use for multiple `INSERT`, `UPDATE`, and `DELETE` statements

Pages that are changed by multiple statements are only written once

Useful for aggregating changes over time!

File Size and Privacy

File Size and Privacy

What happens when we delete data from a database?

Space containing the deleted data is marked as free

While no longer part of the database the deleted data is still on disk

File Size and Privacy

How do we securely delete sensitive data?

File Size and Privacy

```
PRAGMA schema.secure_delete=FAST;
```

How do we securely delete sensitive data?

File Size and Privacy

```
PRAGMA schema.secure_delete=FAST;
```

How do we securely delete sensitive data?

- Automatically zeros deleted data
- No cost for data within the same page as the header
- Default behavior starting with iOS 13

File Size and Privacy

Don't use `VACUUM`

File Size and Privacy

Don't use `VACUUM`

- `VACUUM` is a slow I/O intensive operation

Why Is VACUUM Expensive?

Database

Why Is `VACUUM` Expensive?

Database

Journal

Why Is `VACUUM` Expensive?

Database

Journal

Why Is `VACUUM` Expensive?

Database

Journal

Why Is `VACUUM` Expensive?

Database

File Size and Privacy

Don't use `VACUUM`

- `VACUUM` is a slow I/O intensive operation
- All valid data gets written at least twice!

File Size and Privacy

```
PRAGMA schema.auto_vacuum=INCREMENTAL;
```

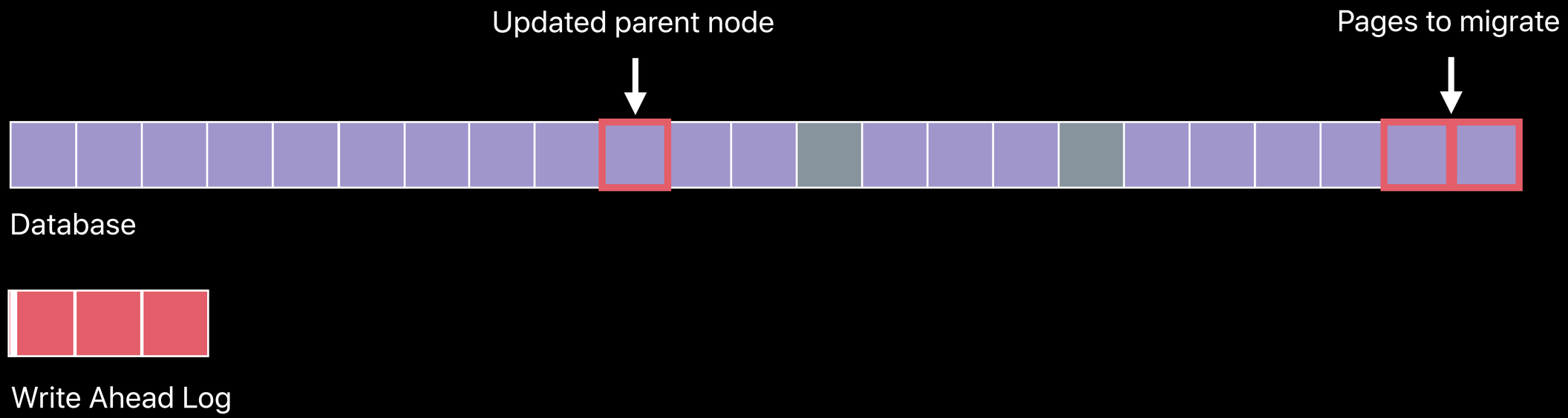
```
PRAGMA schema.incremental_vacuum=(N);
// 'N' is the number pages to be vacuumed
```

How Is Incremental Auto Vacuum More Efficient?

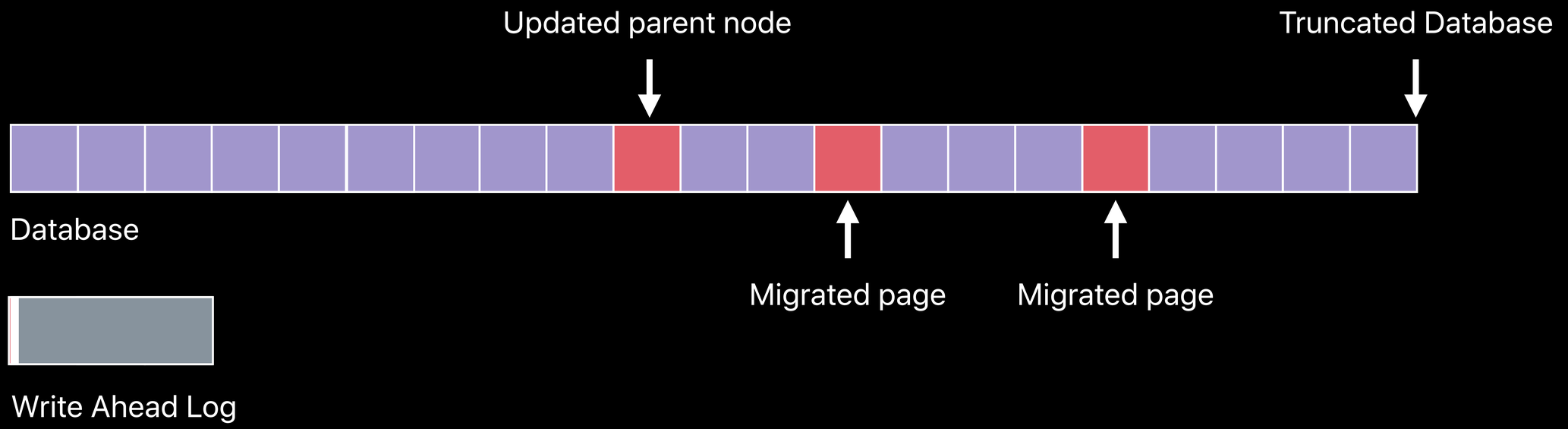
Database

Write Ahead Log

How Is Incremental Auto Vacuum More Efficient?



How Is Incremental Auto Vacuum More Efficient?



Use incremental auto vacuum and fast secure delete to manage both file size and privacy.

Partial Indexes

Indexes

Faster `ORDER BY`, `GROUP BY`, and `WHERE` clauses

Each index adds additional I/O when writing to the database

Partial Indexes

Indexes with a `WHERE` clause

Partial indexes only store data which matches the `WHERE` clause

Queries that do not match the `WHERE` clause cannot use the index

SQLite Summary

SQLite Summary

Best practices

Keep database connections open

SQLite Summary

Best practices

Keep database connections open

Use WAL mode

SQLite Summary

Best practices

Keep database connections open

Use WAL mode

Multiple statements per transaction

SQLite Summary

Best practices

Keep database connections open

Use WAL mode

Multiple statements per transaction

Fast secure delete and auto vacuum incremental

SQLite Summary

Best practices

Keep database connections open

Use WAL mode

Multiple statements per transaction

Fast secure delete and auto vacuum incremental

Partial indexes

File Activity Instrument

Alejandro Lucena, Developer Tools

File Activity Instrument

File Activity Instrument

Supports all Apple devices

File Activity Instrument

Supports all Apple devices

Support for tracing single processes and all processes

File Activity Instrument

Supports all Apple devices

Support for tracing single processes and all processes

Obtains both logical and physical I/O information

File Activity Instrument

Supports all Apple devices

Support for tracing single processes and all processes

Obtains both logical and physical I/O information

Offers automated reasoning

Automated Reasoning

Automated Reasoning

Excessive physical writes

Automated Reasoning

Excessive physical writes

Failed I/O related calls

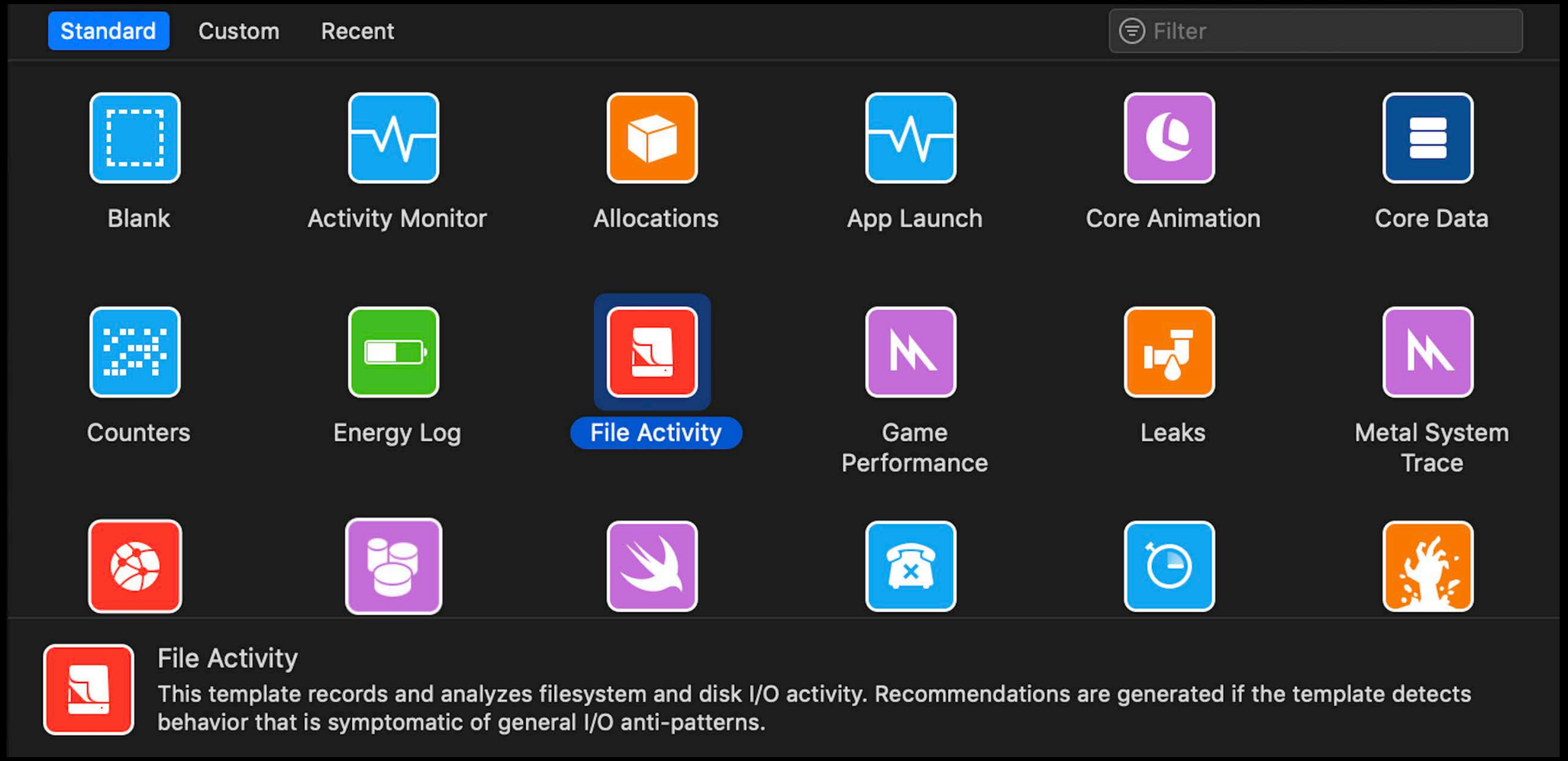
Automated Reasoning

Excessive physical writes

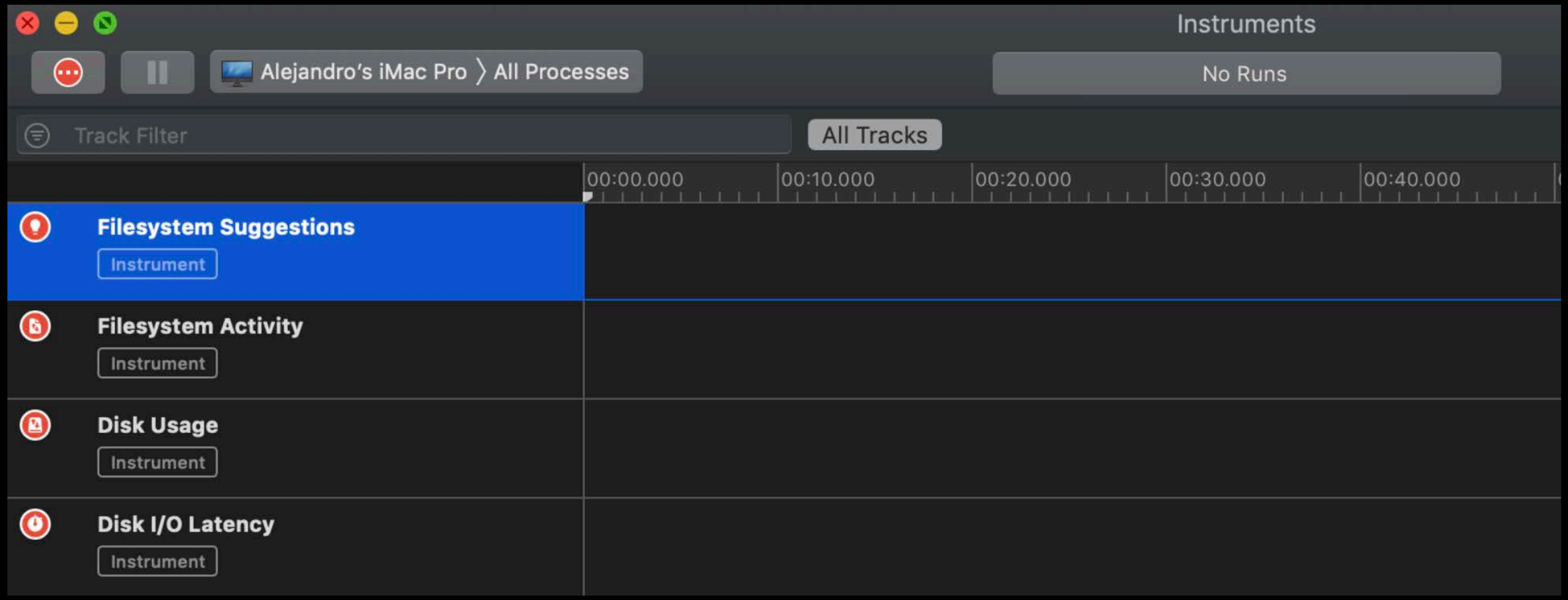
Failed I/O related calls

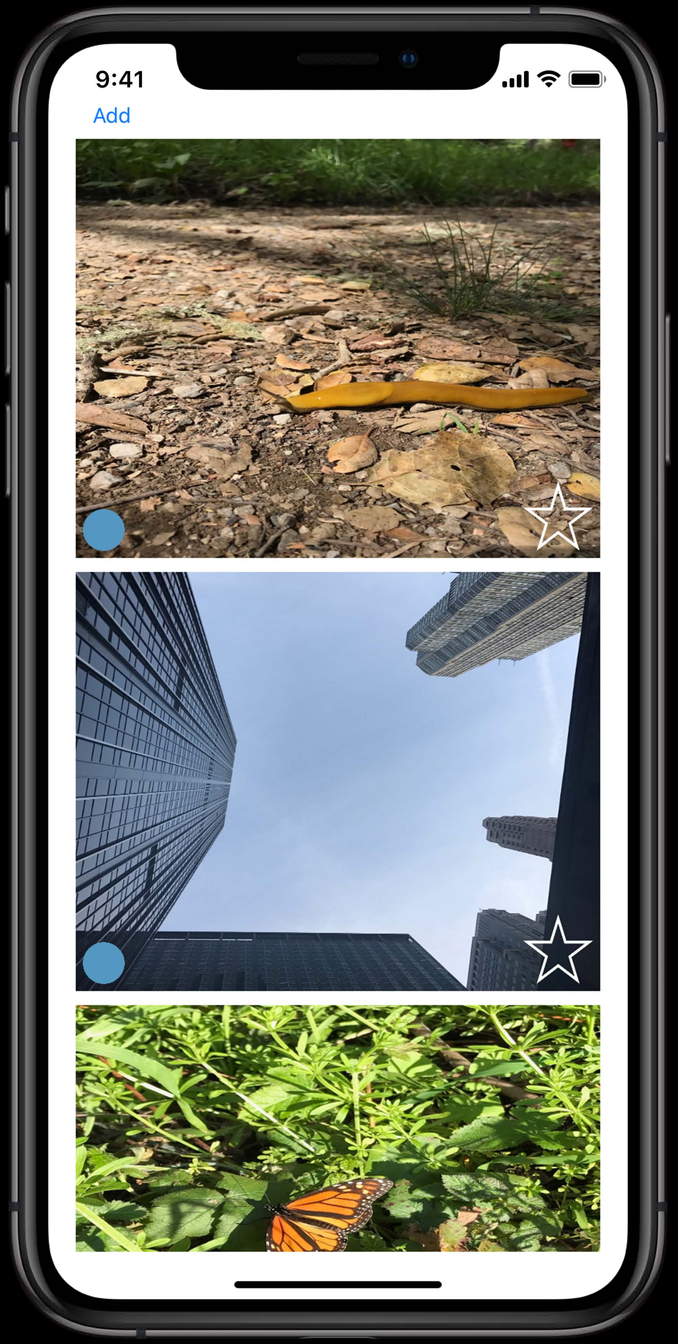
Suboptimal caching

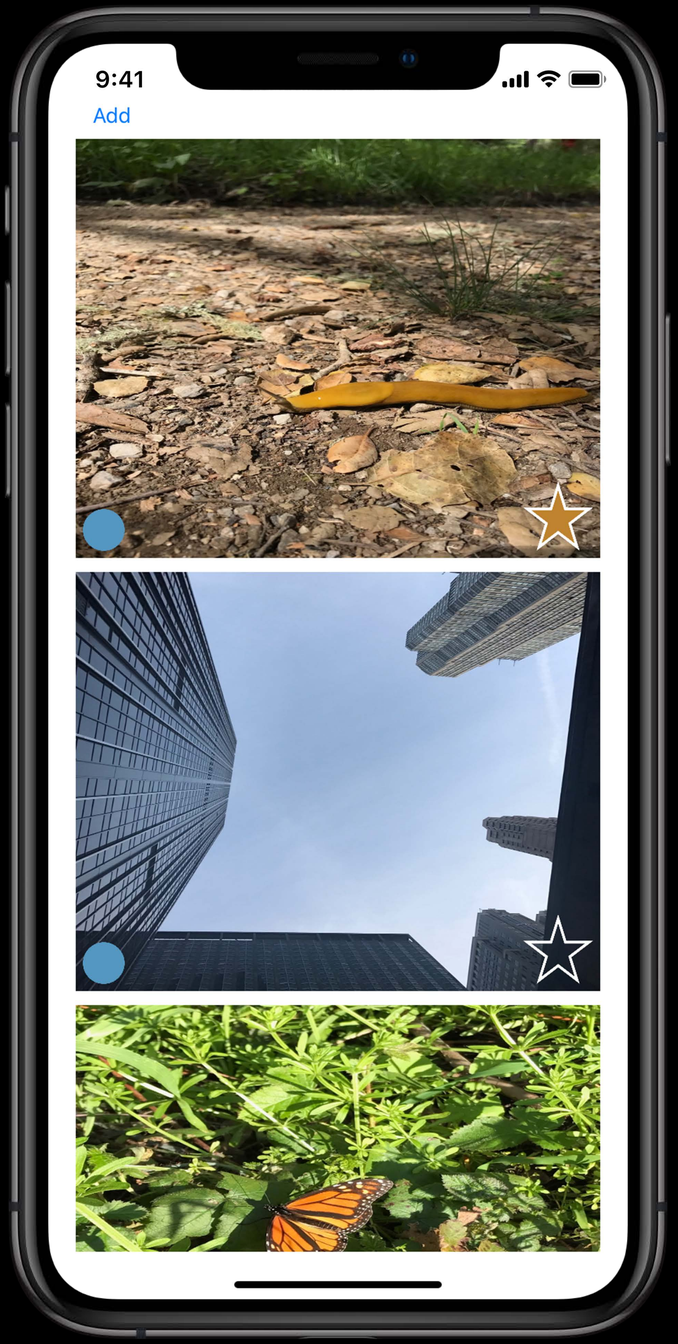
File Activity Instrument

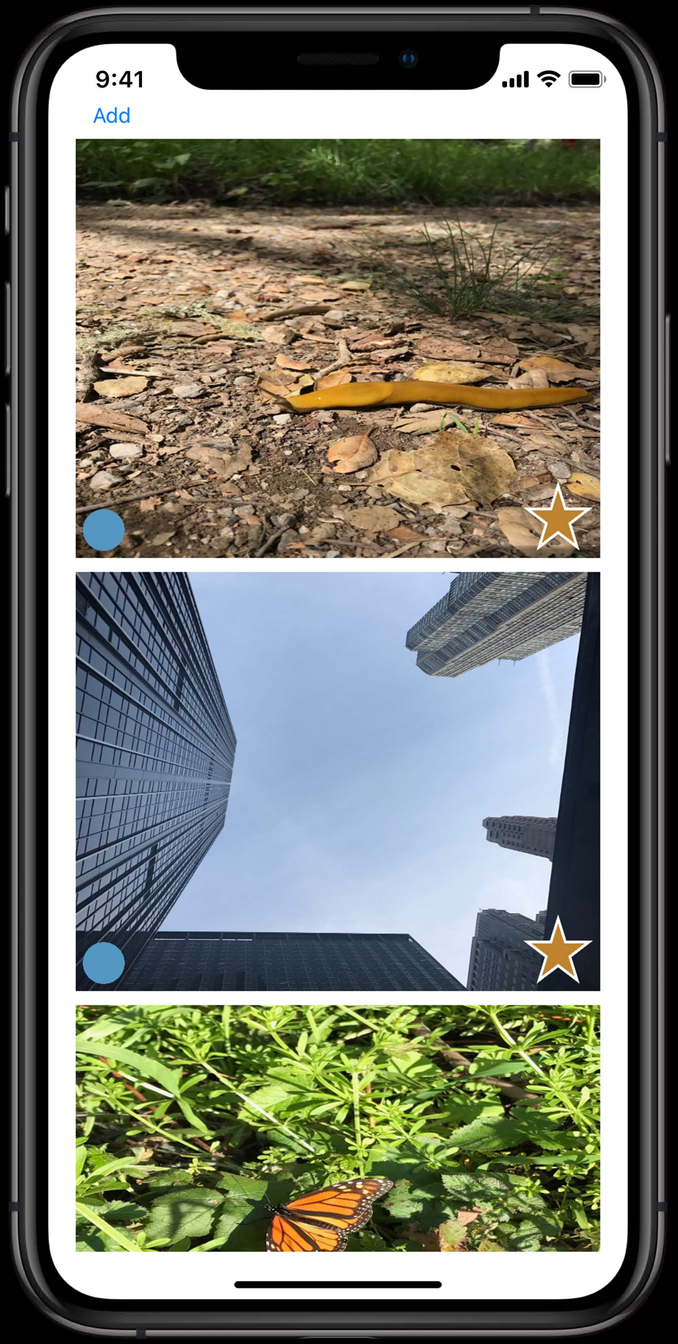


File Activity Instrument



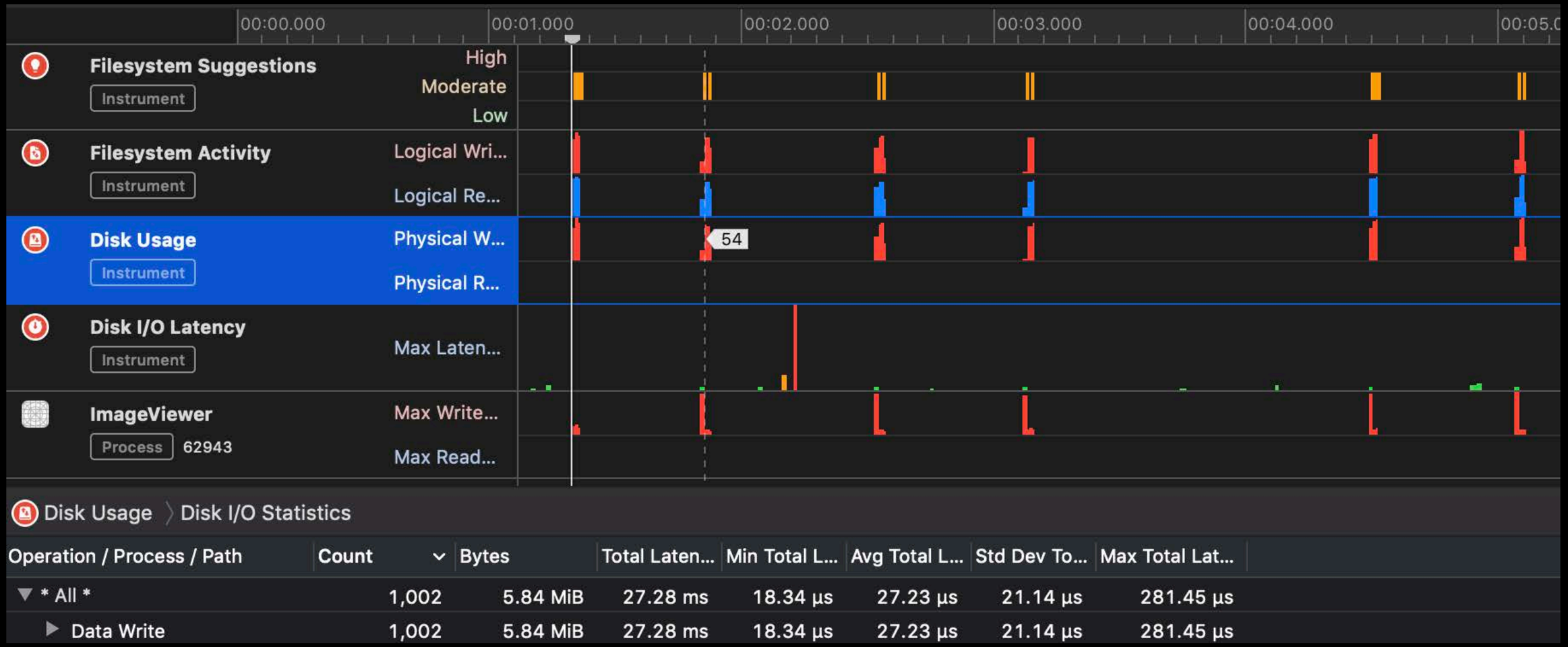

9:41
Add


9:41
Add


9:41
Add

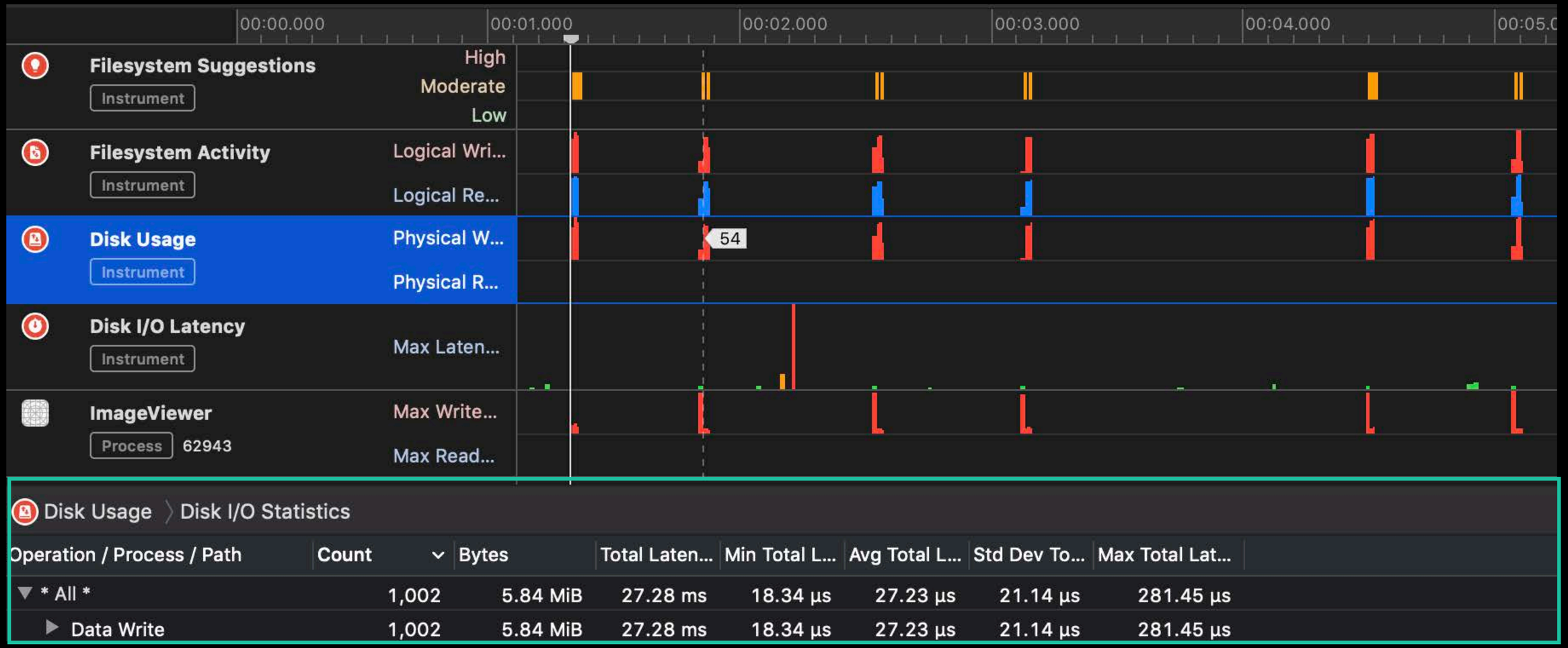
Open and Close per Operation

Disk Usage



Open and Close per Operation

Disk Usage



Open and Close per Operation

Disk Usage — Detail view

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten…	Min Total L…	Avg Total L…	Std Dev To…	Max Total L…
▼ * All *	1,003	5.84 MiB	25.71 ms	18.42 µs	25.63 µs	22.17 µs	302.64 µs
▼ Data Write	1,002	5.84 MiB	25.52 ms	18.42 µs	25.47 µs	21.59 µs	302.64 µs
▼ ImageViewer (61960)	1,002	5.84 MiB	25.52 ms	18.42 µs	25.47 µs	21.59 µs	302.64 µs
/ImageViewer/open_close_imageviewer.db-journal	747	3.89 MiB	19.71 ms	18.42 µs	26.39 µs	24.86 µs	302.64 µs
/ImageViewer/open_close_imageviewer.db	255	1.95 MiB	5.81 ms	19.26 µs	22.79 µs	3.50 µs	39.47 µs
▶ Data Read	1	4.00 KiB	186.51 µs	186.51 µs	186.51 µs	n/a	186.51 µs

Open and Close per Operation

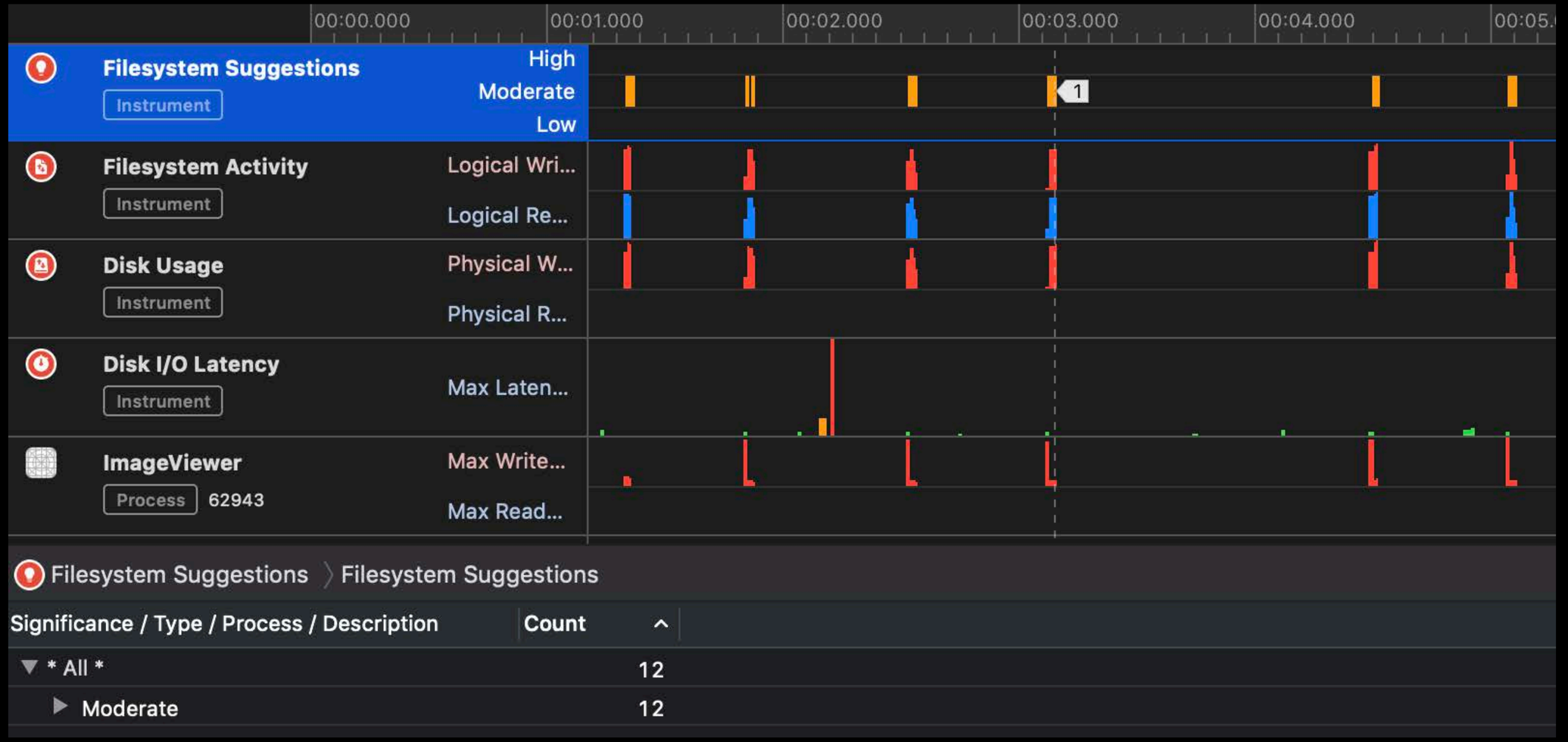
Disk Usage — Detail view

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	1,003	5.84 MiB	25.71 ms	18.42 µs	25.63 µs	22.17 µs	302.64 µs
▼ Data Write	1,002	5.84 MiB	25.52 ms	18.42 µs	25.47 µs	21.59 µs	302.64 µs
▼ ImageViewer (61960)	1,002	5.84 MiB	25.52 ms	18.42 µs	25.47 µs	21.59 µs	302.64 µs
/ImageViewer/open_close_imageviewer.db-journal	747	3.89 MiB	19.71 ms	18.42 µs	26.39 µs	24.86 µs	302.64 µs
/ImageViewer/open_close_imageviewer.db	255	1.95 MiB	5.81 ms	19.26 µs	22.79 µs	3.50 µs	39.47 µs
▶ Data Read	1	4.00 KiB	186.51 µs	186.51 µs	186.51 µs	n/a	186.51 µs

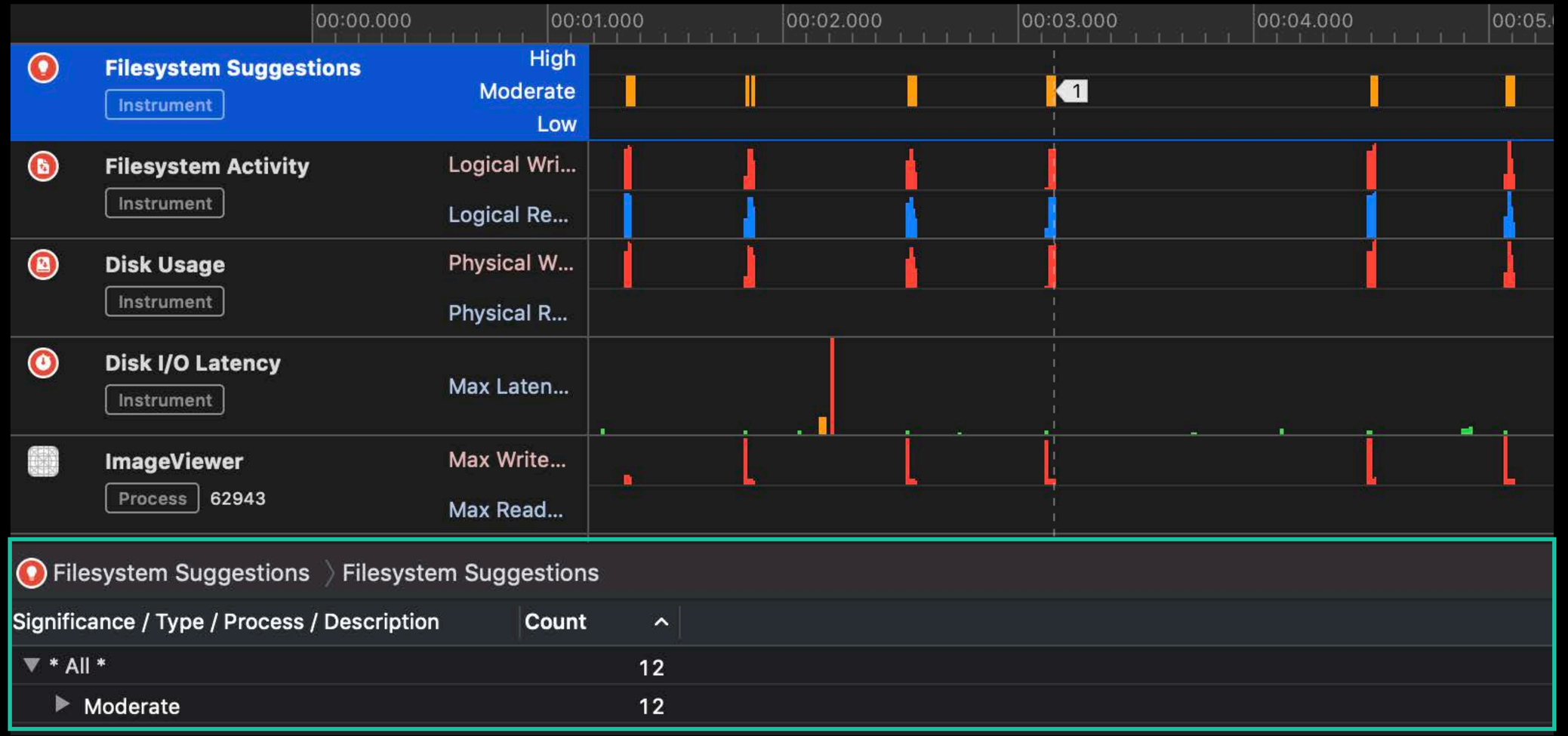
Open and Close per Operation

Filesystem Suggestions



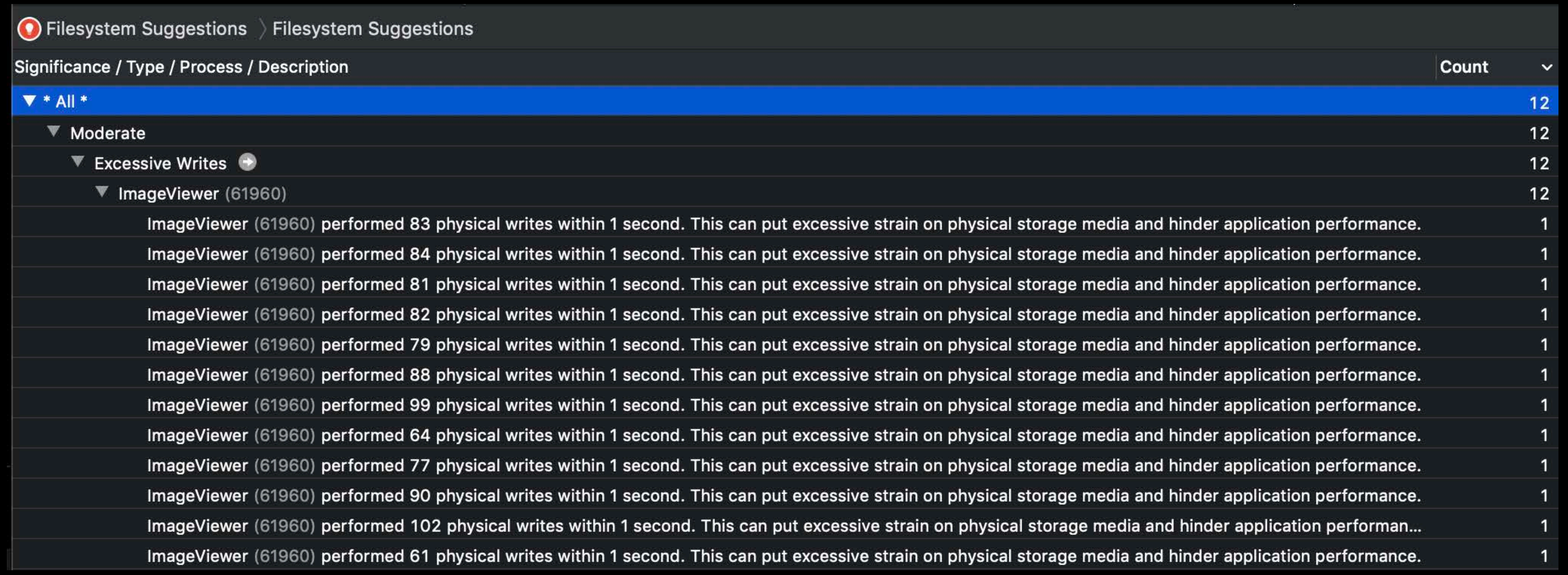
Open and Close per Operation

Filesystem Suggestions



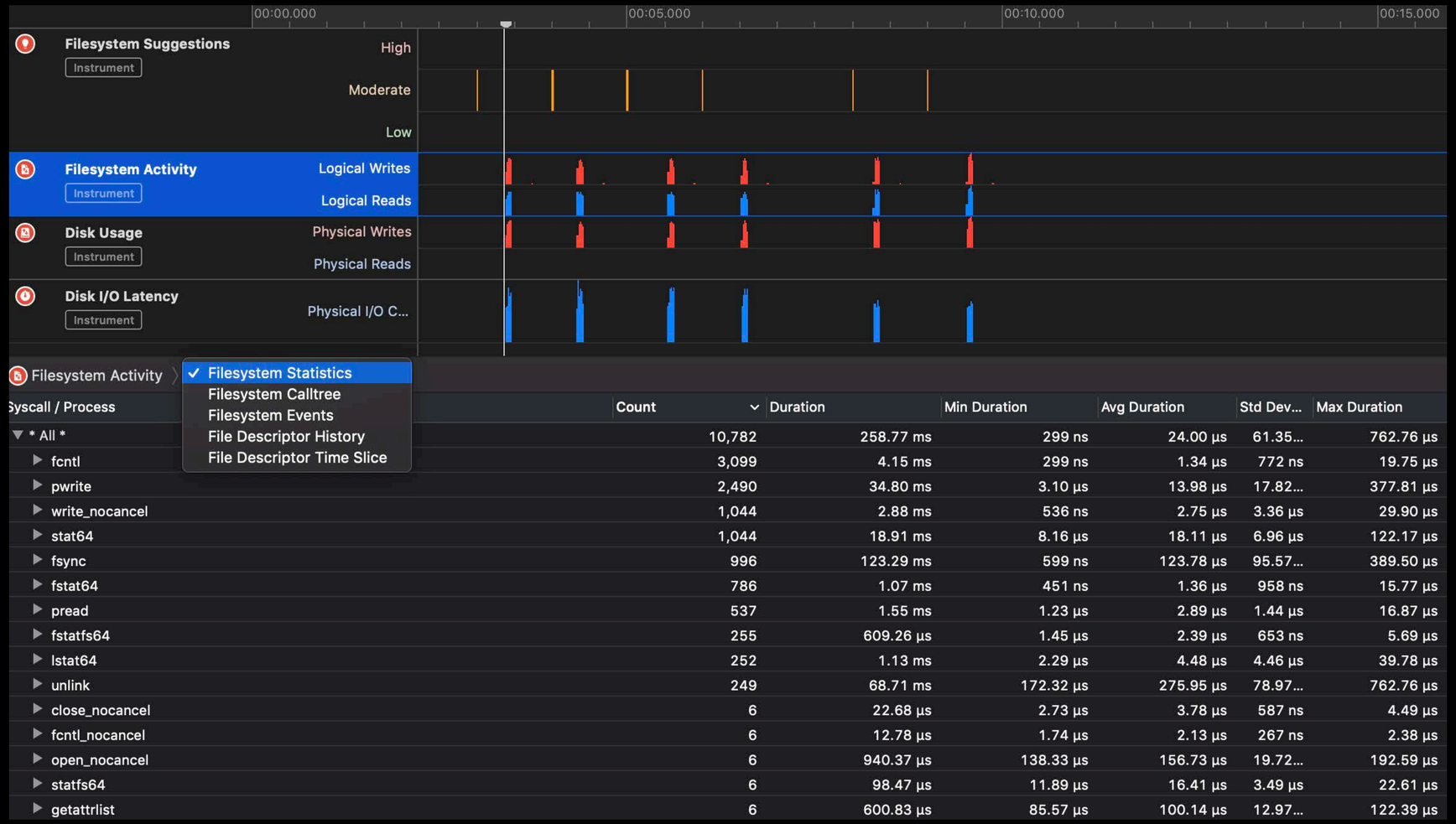
Open and Close per Operation

Filesystem Suggestions — Detail view



Open and Close per Operation

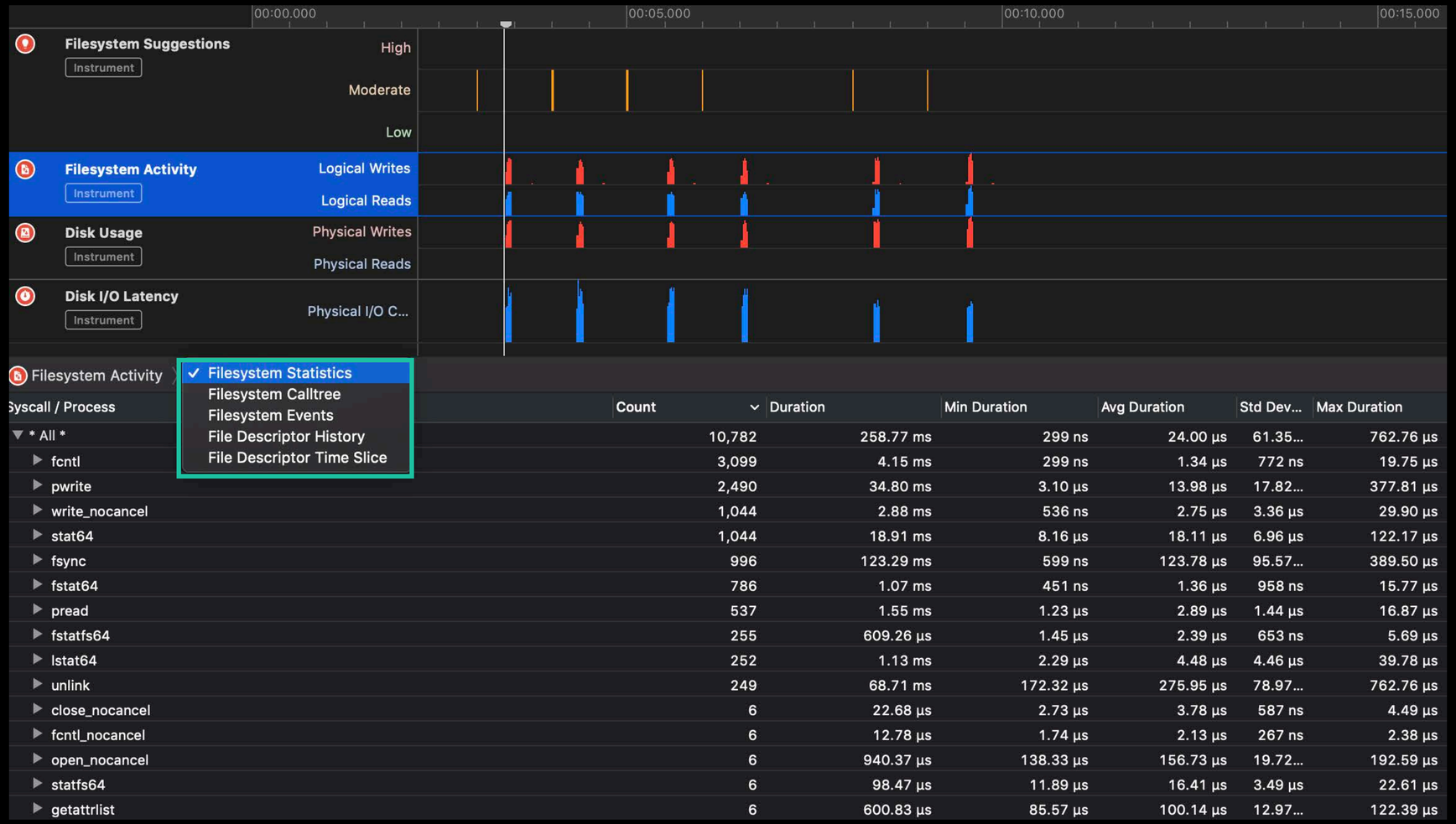
Filesystem Activity



Syscall / Process	Count	Duration	Min Duration	Avg Duration	Std Dev...	Max Duration
* All *	10,782	258.77 ms	299 ns	24.00 µs	61.35...	762.76 µs
fcntl	3,099	4.15 ms	299 ns	1.34 µs	772 ns	19.75 µs
pwrite	2,490	34.80 ms	3.10 µs	13.98 µs	17.82...	377.81 µs
write_nocancel	1,044	2.88 ms	536 ns	2.75 µs	3.36 µs	29.90 µs
stat64	1,044	18.91 ms	8.16 µs	18.11 µs	6.96 µs	122.17 µs
fsync	996	123.29 ms	599 ns	123.78 µs	95.57...	389.50 µs
fstat64	786	1.07 ms	451 ns	1.36 µs	958 ns	15.77 µs
pread	537	1.55 ms	1.23 µs	2.89 µs	1.44 µs	16.87 µs
fstatfs64	255	609.26 µs	1.45 µs	2.39 µs	653 ns	5.69 µs
lstat64	252	1.13 ms	2.29 µs	4.48 µs	4.46 µs	39.78 µs
unlink	249	68.71 ms	172.32 µs	275.95 µs	78.97...	762.76 µs
close_nocancel	6	22.68 µs	2.73 µs	3.78 µs	587 ns	4.49 µs
fcntl_nocancel	6	12.78 µs	1.74 µs	2.13 µs	267 ns	2.38 µs
open_nocancel	6	940.37 µs	138.33 µs	156.73 µs	19.72...	192.59 µs
statfs64	6	98.47 µs	11.89 µs	16.41 µs	3.49 µs	22.61 µs
getattrlist	6	600.83 µs	85.57 µs	100.14 µs	12.97...	122.39 µs

Open and Close per Operation

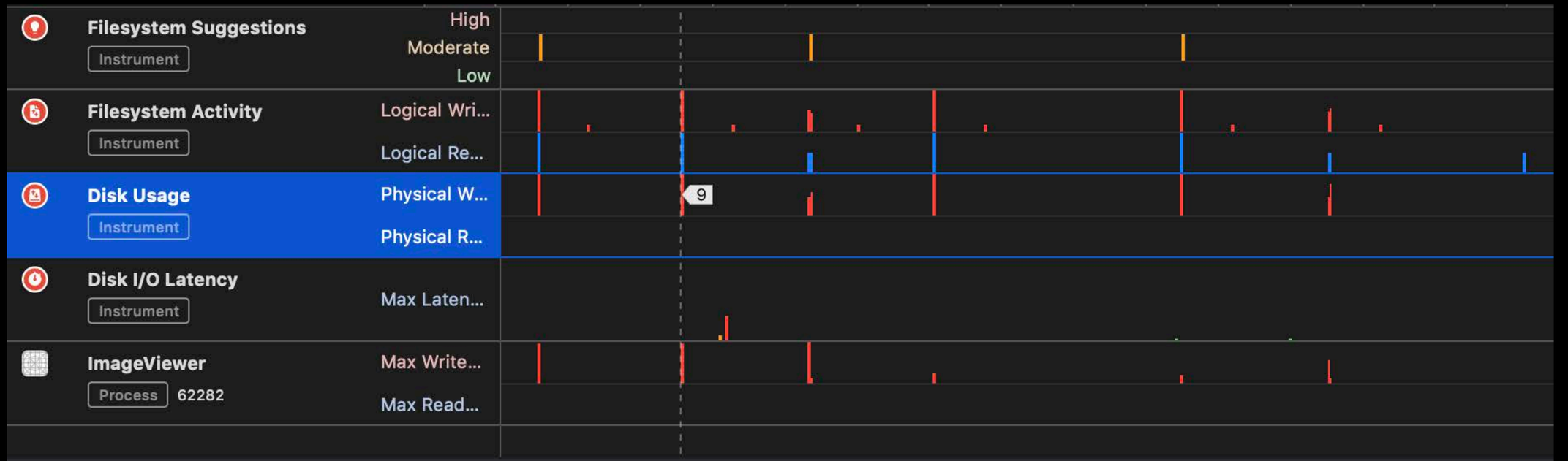
Filesystem Activity



Syscall / Process	Count	Duration	Min Duration	Avg Duration	Std Dev...	Max Duration
* All *	10,782	258.77 ms	299 ns	24.00 µs	61.35...	762.76 µs
fcntl	3,099	4.15 ms	299 ns	1.34 µs	772 ns	19.75 µs
pwrite	2,490	34.80 ms	3.10 µs	13.98 µs	17.82...	377.81 µs
write_nocancel	1,044	2.88 ms	536 ns	2.75 µs	3.36 µs	29.90 µs
stat64	1,044	18.91 ms	8.16 µs	18.11 µs	6.96 µs	122.17 µs
fsync	996	123.29 ms	599 ns	123.78 µs	95.57...	389.50 µs
fstat64	786	1.07 ms	451 ns	1.36 µs	958 ns	15.77 µs
pread	537	1.55 ms	1.23 µs	2.89 µs	1.44 µs	16.87 µs
fstatfs64	255	609.26 µs	1.45 µs	2.39 µs	653 ns	5.69 µs
lstat64	252	1.13 ms	2.29 µs	4.48 µs	4.46 µs	39.78 µs
unlink	249	68.71 ms	172.32 µs	275.95 µs	78.97...	762.76 µs
close_nocancel	6	22.68 µs	2.73 µs	3.78 µs	587 ns	4.49 µs
fcntl_nocancel	6	12.78 µs	1.74 µs	2.13 µs	267 ns	2.38 µs
open_nocancel	6	940.37 µs	138.33 µs	156.73 µs	19.72...	192.59 µs
statfs64	6	98.47 µs	11.89 µs	16.41 µs	3.49 µs	22.61 µs
getattrlist	6	600.83 µs	85.57 µs	100.14 µs	12.97...	122.39 µs

Open and Close as Needed

Disk Usage



Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ Data Write	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ ImageViewer (62282)	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db-journal	36	192.00 KiB	2.79 ms	18.71 µs	77.46 µs	88.50 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db	18	96.00 KiB	489.55 µs	19.98 µs	27.20 µs	5.02 µs	35.20 µs

Open and Close as Needed

Disk Usage — Detail view

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ Data Write	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ ImageViewer (62282)	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db-journal	36	192.00 KiB	2.79 ms	18.71 µs	77.46 µs	88.50 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db	18	96.00 KiB	489.55 µs	19.98 µs	27.20 µs	5.02 µs	35.20 µs

Open and Close as Needed

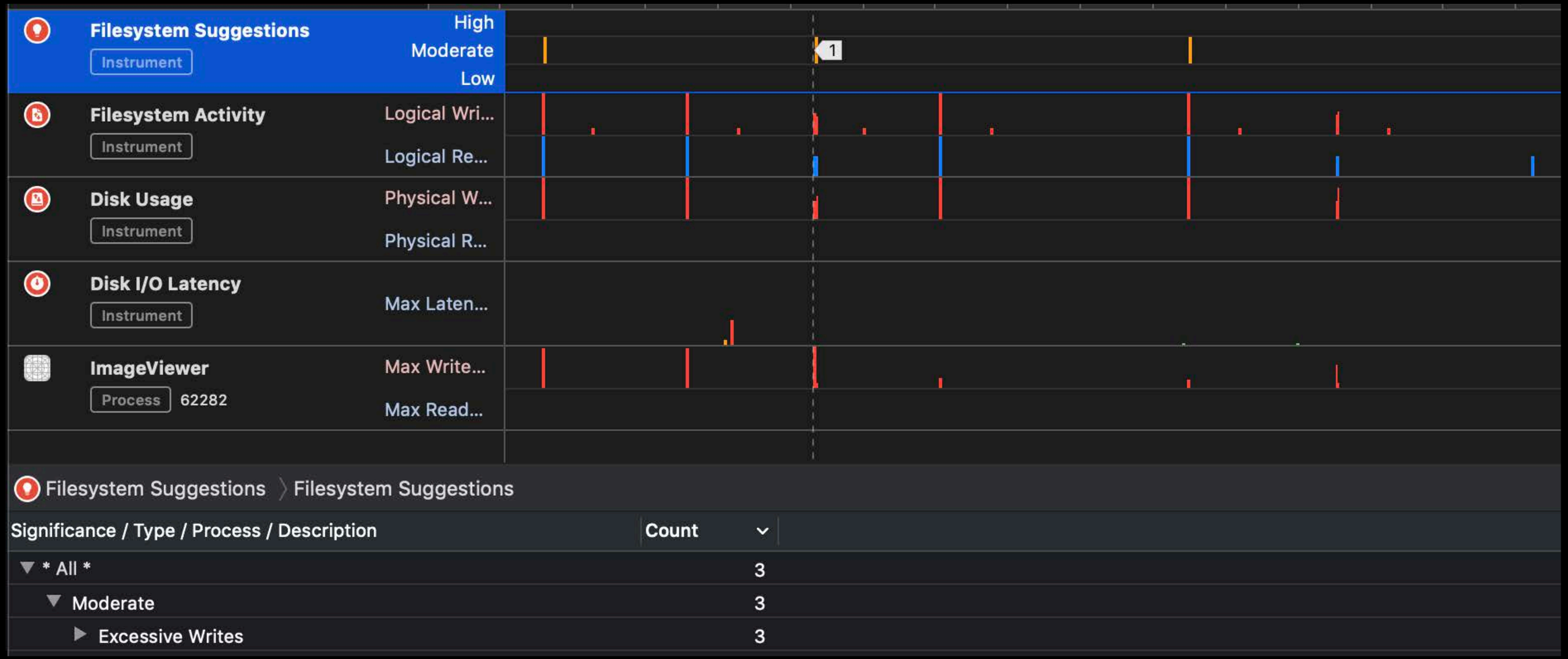
Disk Usage — Detail view

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ Data Write	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
▼ ImageViewer (62282)	54	288.00 KiB	3.28 ms	18.71 µs	60.71 µs	75.84 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db-journal	36	192.00 KiB	2.79 ms	18.71 µs	77.46 µs	88.50 µs	275.92 µs
/ImageViewer/open_close_as_needed_imageviewer.db	18	96.00 KiB	489.55 µs	19.98 µs	27.20 µs	5.02 µs	35.20 µs

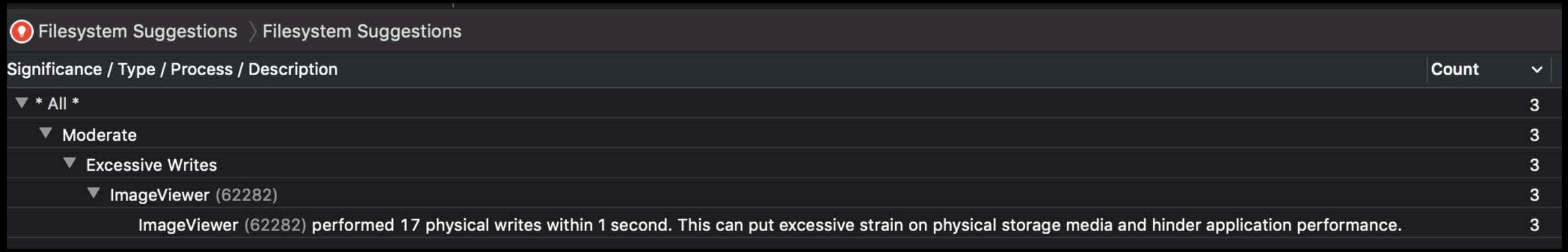
Open and Close as Needed

Filesystem Suggestions



Open and Close as Needed

Filesystem Suggestions — Detail view



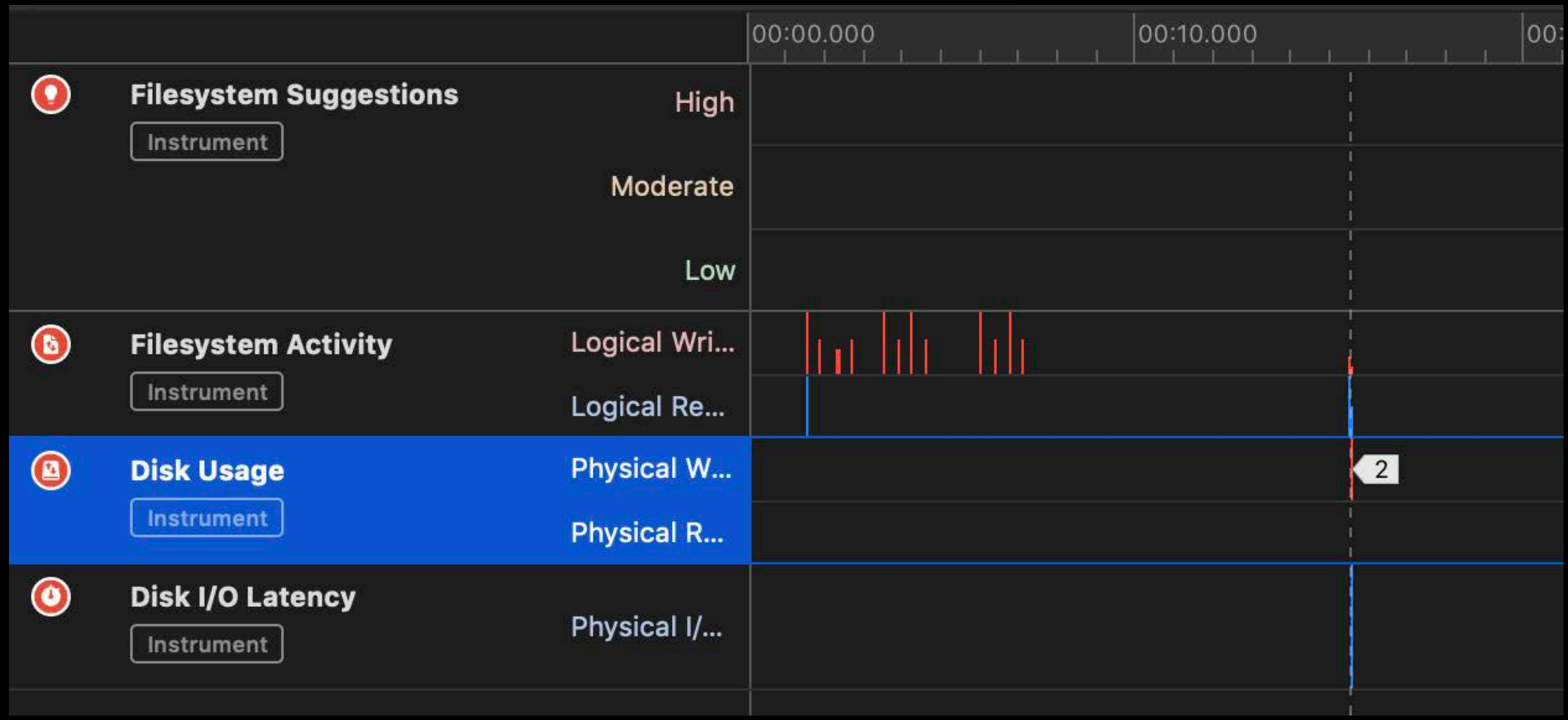
Delete Mode Journaling

Delete Mode Journaling

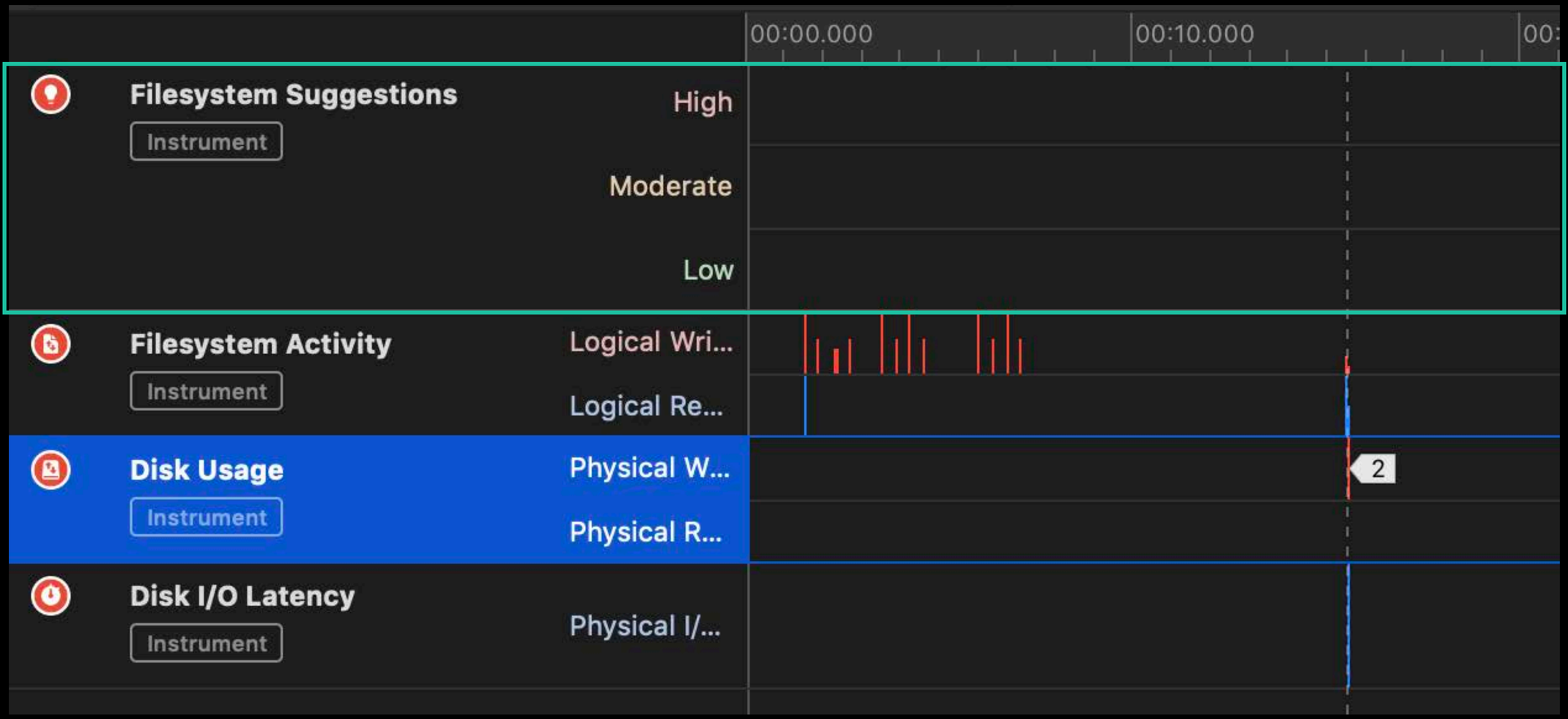
Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	54	288.00 KiB	2.40 ms	18.98 µs	44.42 µs	57.65 µs	275.53 µs
▼ Data Write	54	288.00 KiB	2.40 ms	18.98 µs	44.42 µs	57.65 µs	275.53 µs
▼ ImageViewer (63410)	54	288.00 KiB	2.40 ms	18.98 µs	44.42 µs	57.65 µs	275.53 µs
/ImageViewer/delete_mode.db-journal	36	192.00 KiB	1.96 ms	18.98 µs	54.33 µs	68.71 µs	275.53 µs
/ImageViewer/delete_mode.db	18	96.00 KiB	443.16 µs	19.35 µs	24.62 µs	4.24 µs	35.66 µs

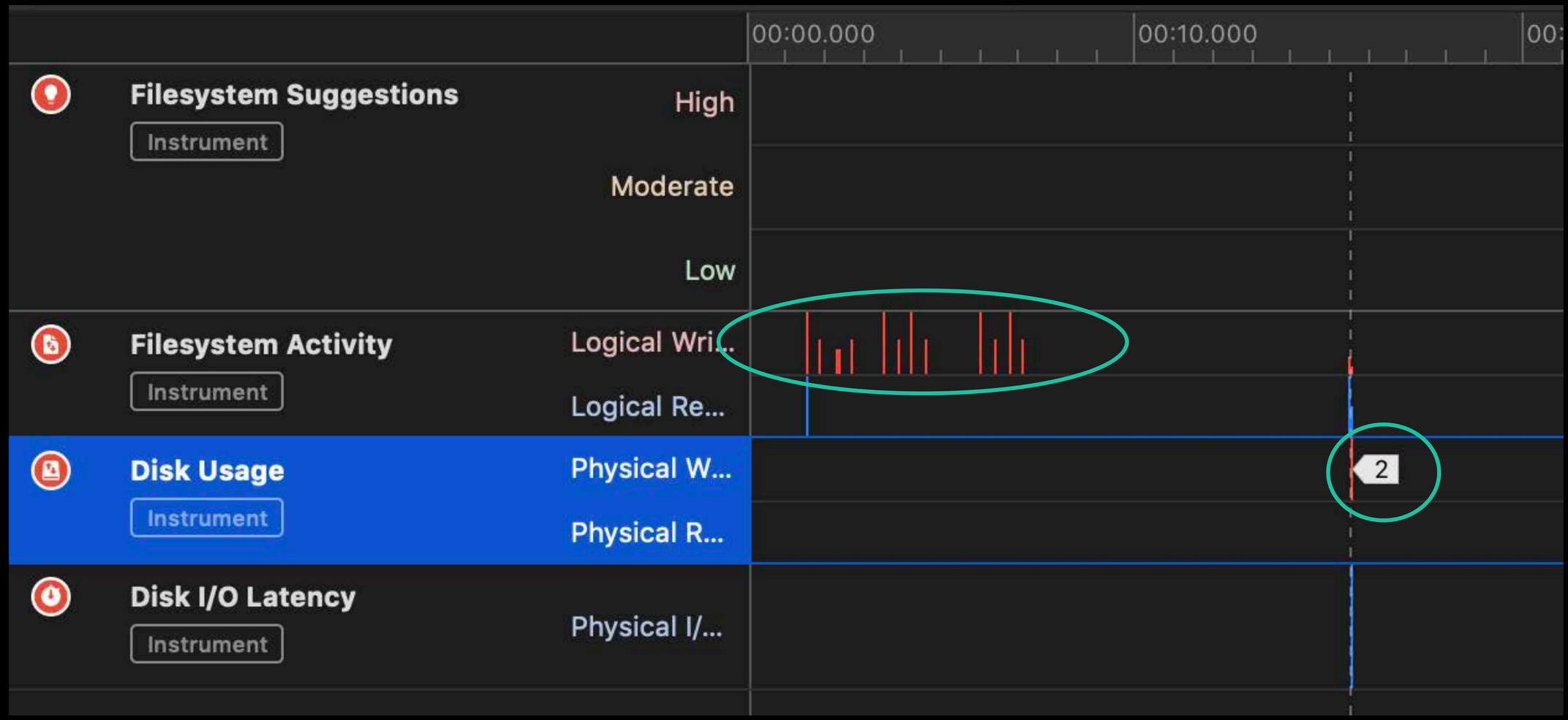
WAL Mode Journaling



WAL Mode Journaling

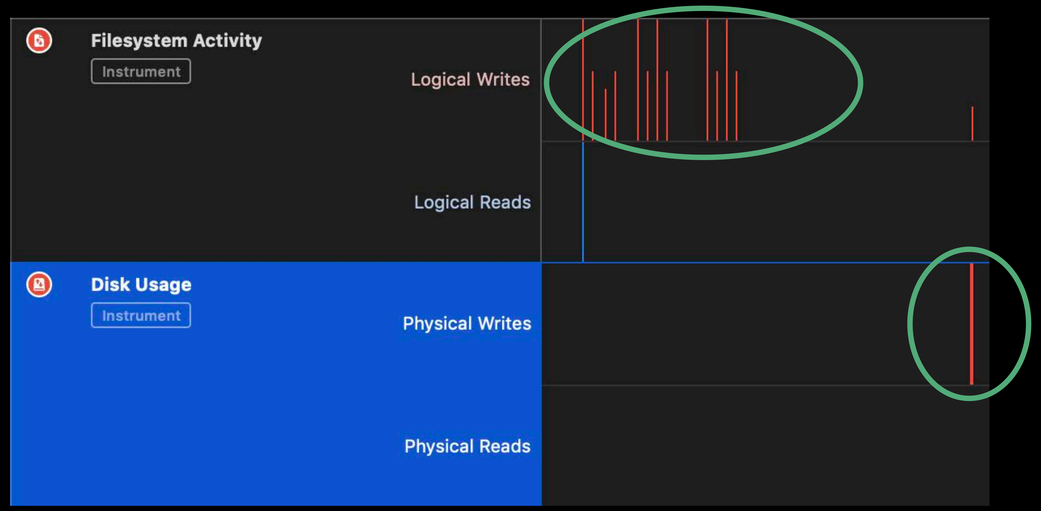


WAL Mode Journaling

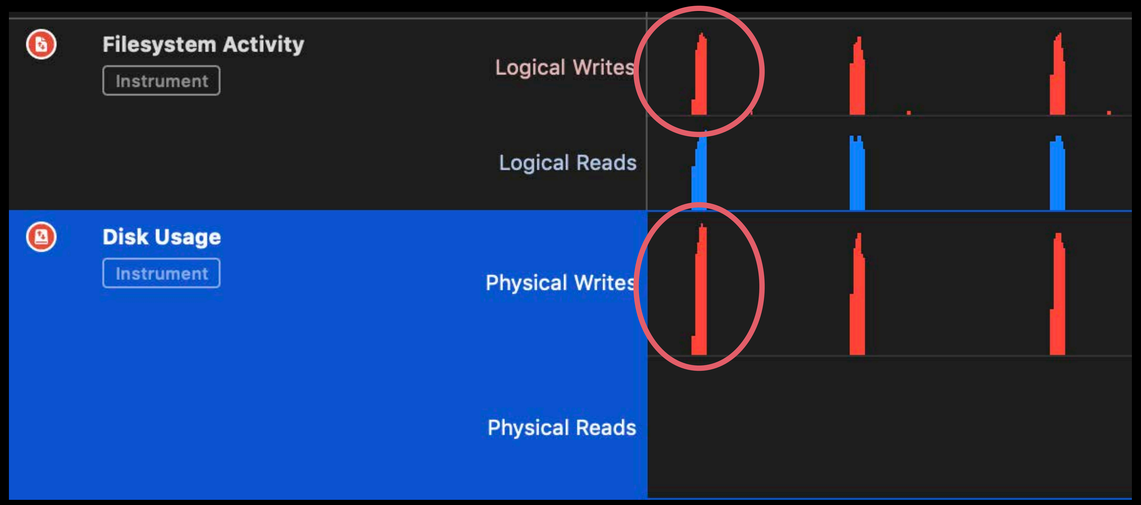


Journaling Comparison

WAL Journaling

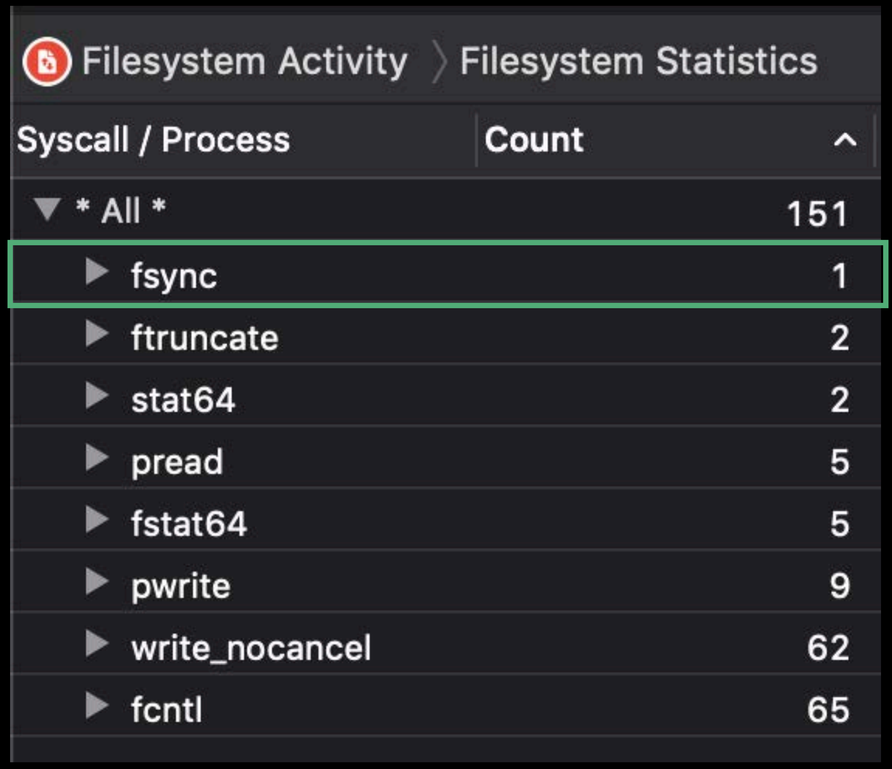


Delete Mode Journaling

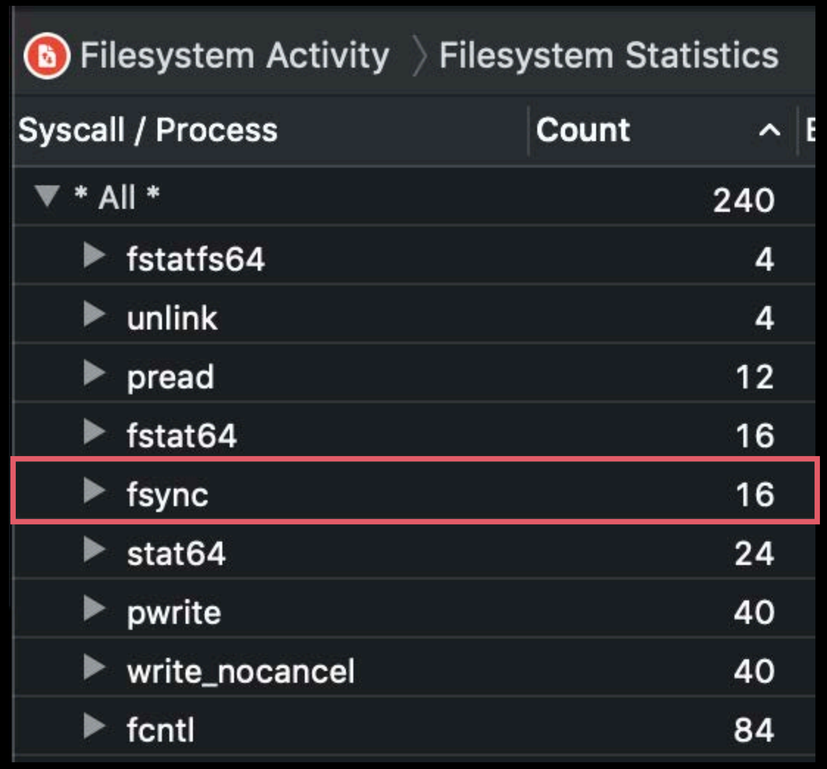


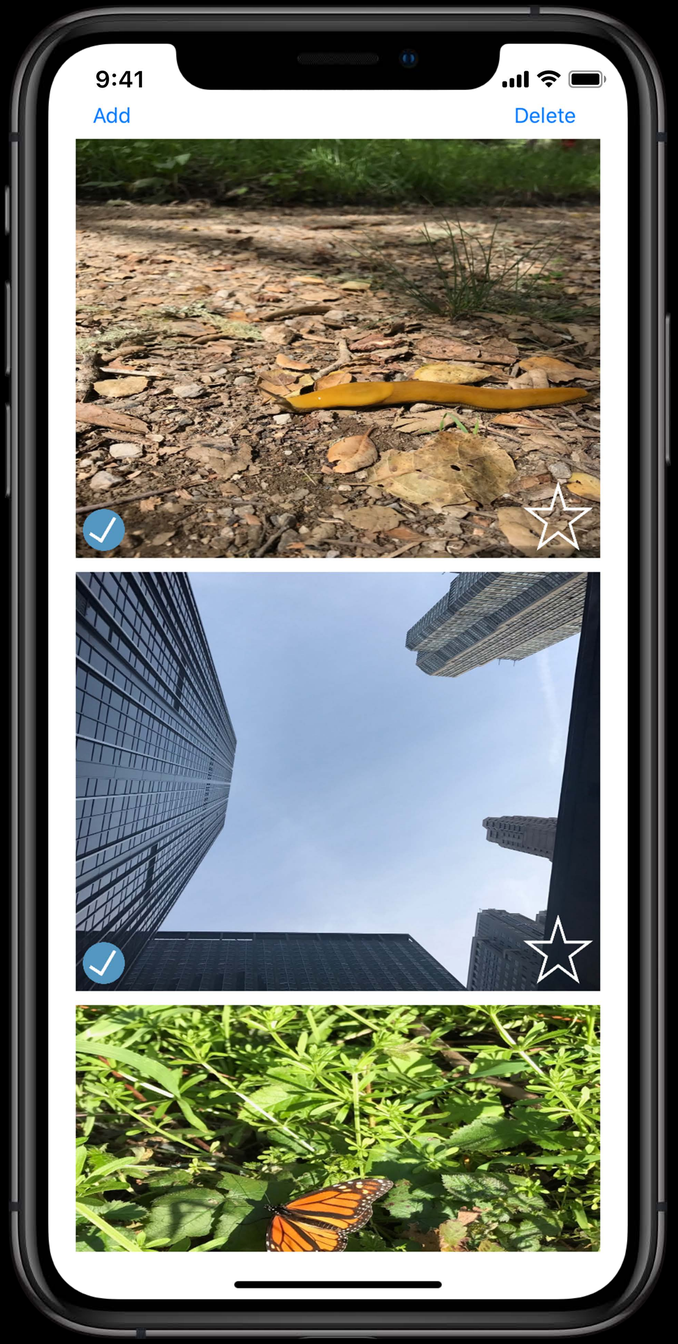
Journaling Mode Comparison

WAL Journaling

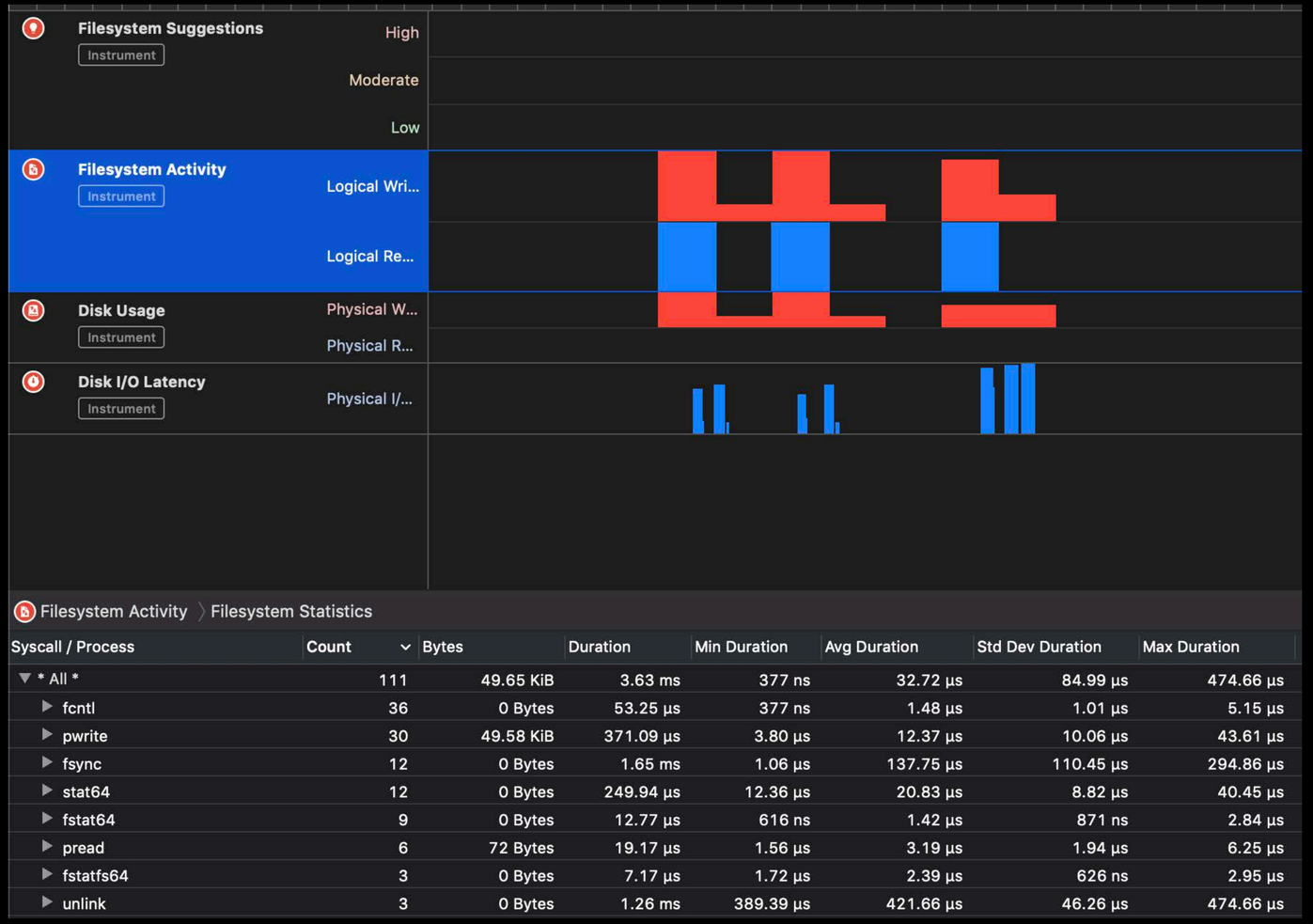


Delete Mode Journaling



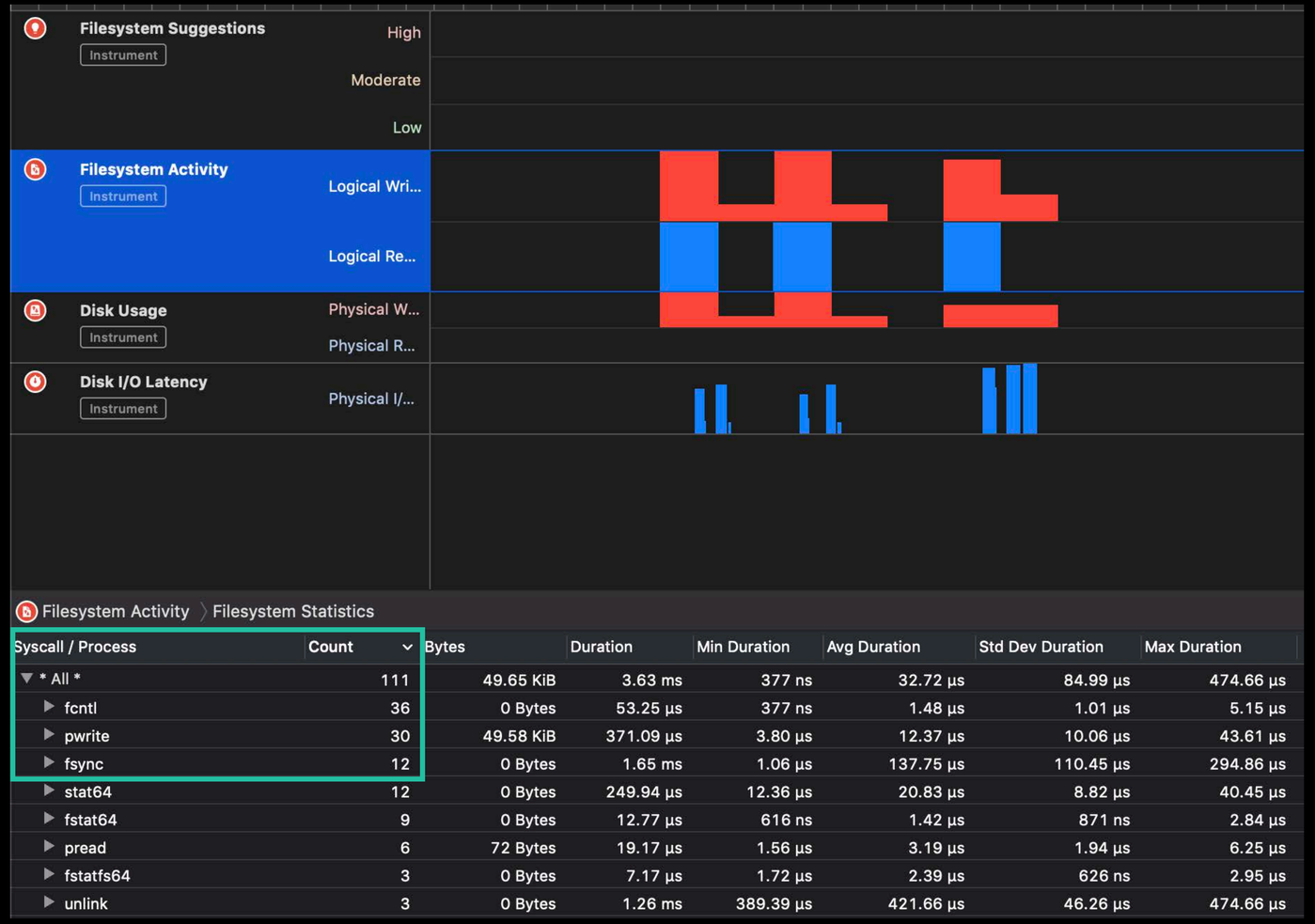

9:41
Add
Delete

Single Statement Transactions



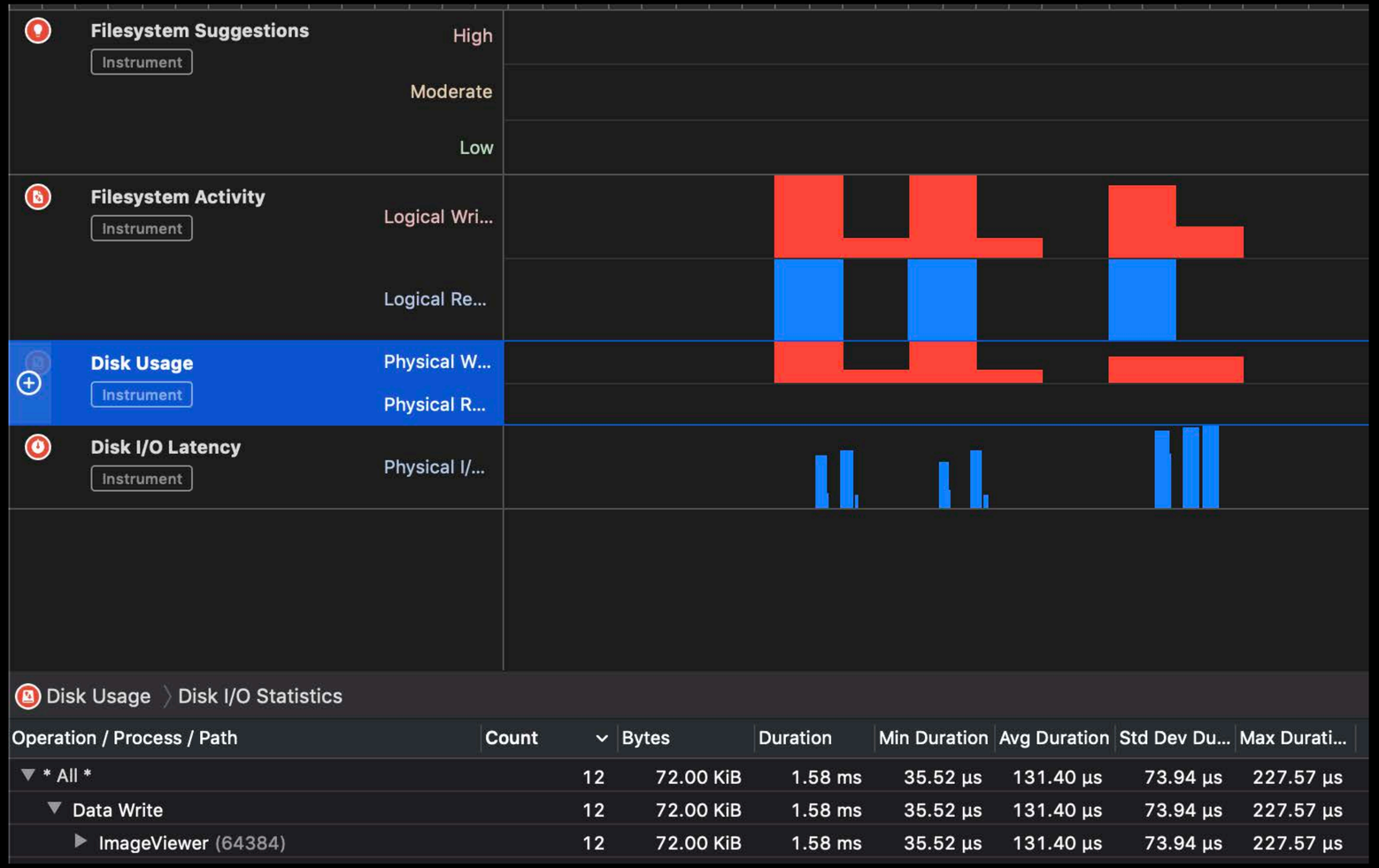
Syscall / Process	Count	Bytes	Duration	Min Duration	Avg Duration	Std Dev Duration	Max Duration
▼ * All *	111	49.65 KiB	3.63 ms	377 ns	32.72 µs	84.99 µs	474.66 µs
▶ fcntl	36	0 Bytes	53.25 µs	377 ns	1.48 µs	1.01 µs	5.15 µs
▶ pwrite	30	49.58 KiB	371.09 µs	3.80 µs	12.37 µs	10.06 µs	43.61 µs
▶ fsync	12	0 Bytes	1.65 ms	1.06 µs	137.75 µs	110.45 µs	294.86 µs
▶ stat64	12	0 Bytes	249.94 µs	12.36 µs	20.83 µs	8.82 µs	40.45 µs
▶ fstat64	9	0 Bytes	12.77 µs	616 ns	1.42 µs	871 ns	2.84 µs
▶ pread	6	72 Bytes	19.17 µs	1.56 µs	3.19 µs	1.94 µs	6.25 µs
▶ fstatfs64	3	0 Bytes	7.17 µs	1.72 µs	2.39 µs	626 ns	2.95 µs
▶ unlink	3	0 Bytes	1.26 ms	389.39 µs	421.66 µs	46.26 µs	474.66 µs

Single Statement Transactions

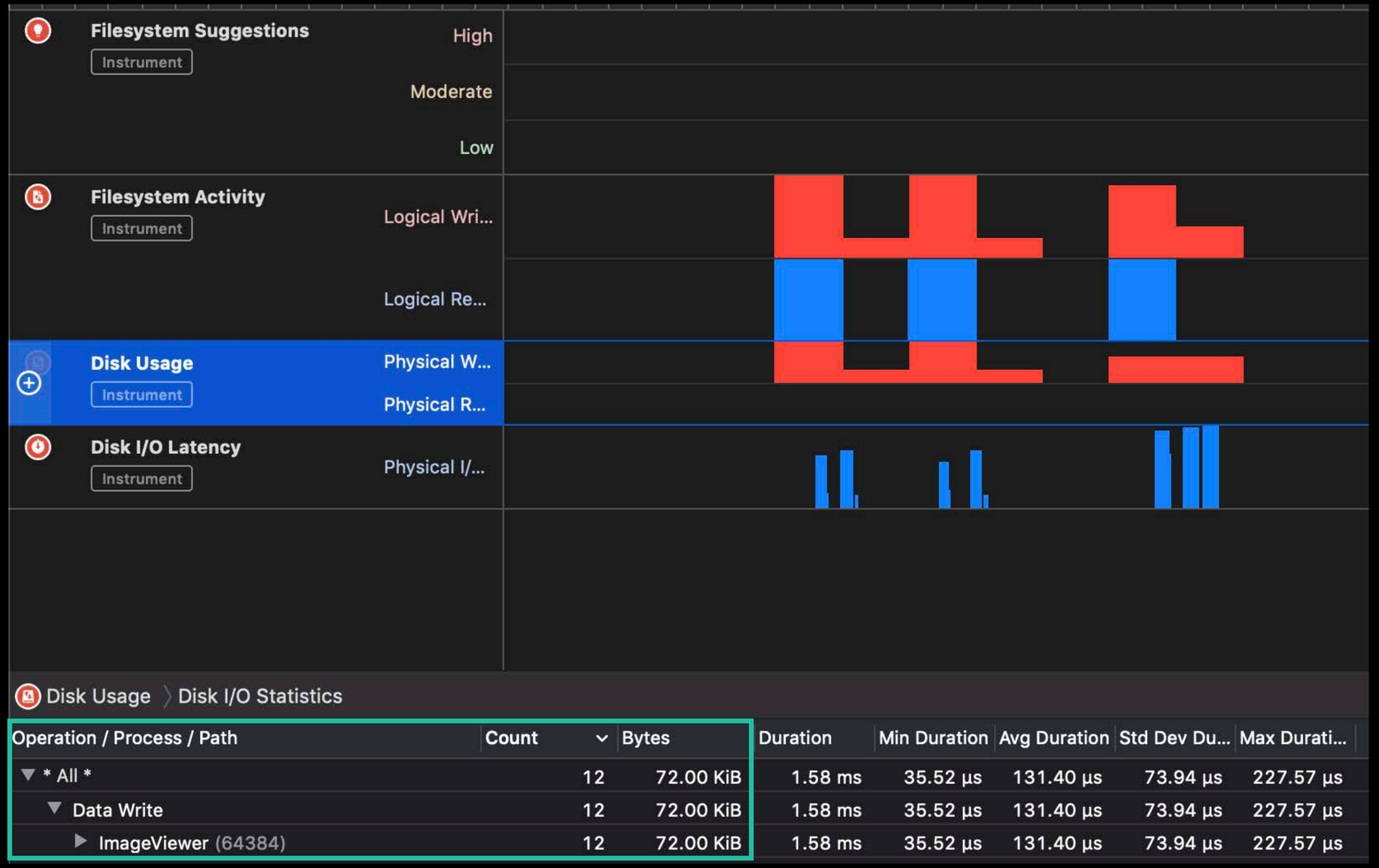


Syscall / Process	Count	Bytes	Duration	Min Duration	Avg Duration	Std Dev Duration	Max Duration
▼ * All *	111	49.65 KiB	3.63 ms	377 ns	32.72 µs	84.99 µs	474.66 µs
▶ fcntl	36	0 Bytes	53.25 µs	377 ns	1.48 µs	1.01 µs	5.15 µs
▶ pwrite	30	49.58 KiB	371.09 µs	3.80 µs	12.37 µs	10.06 µs	43.61 µs
▶ fsync	12	0 Bytes	1.65 ms	1.06 µs	137.75 µs	110.45 µs	294.86 µs
▶ stat64	12	0 Bytes	249.94 µs	12.36 µs	20.83 µs	8.82 µs	40.45 µs
▶ fstat64	9	0 Bytes	12.77 µs	616 ns	1.42 µs	871 ns	2.84 µs
▶ pread	6	72 Bytes	19.17 µs	1.56 µs	3.19 µs	1.94 µs	6.25 µs
▶ fstatfs64	3	0 Bytes	7.17 µs	1.72 µs	2.39 µs	626 ns	2.95 µs
▶ unlink	3	0 Bytes	1.26 ms	389.39 µs	421.66 µs	46.26 µs	474.66 µs

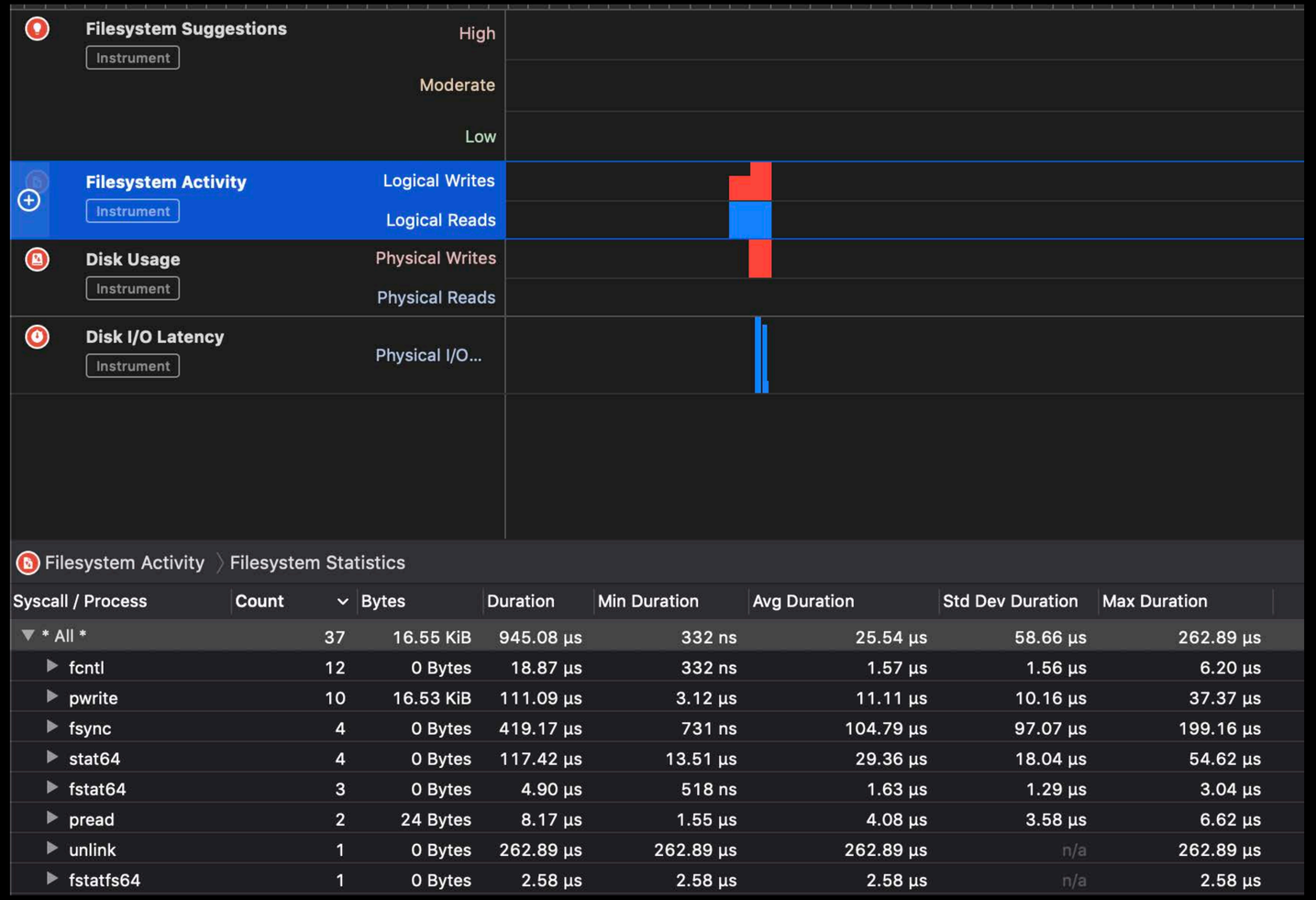
Single Statement Transactions



Single Statement Transactions

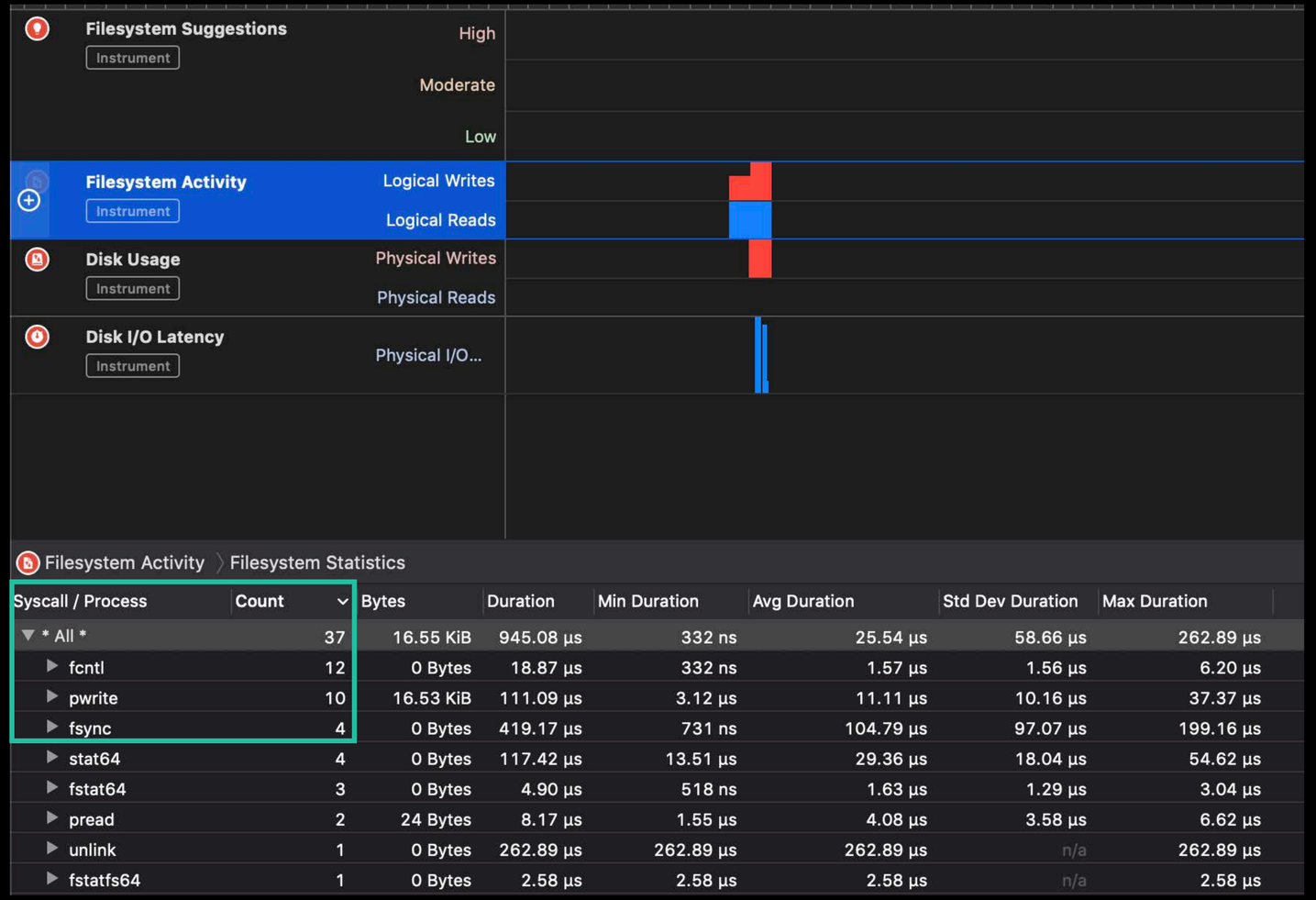


Multiple Statement Transaction



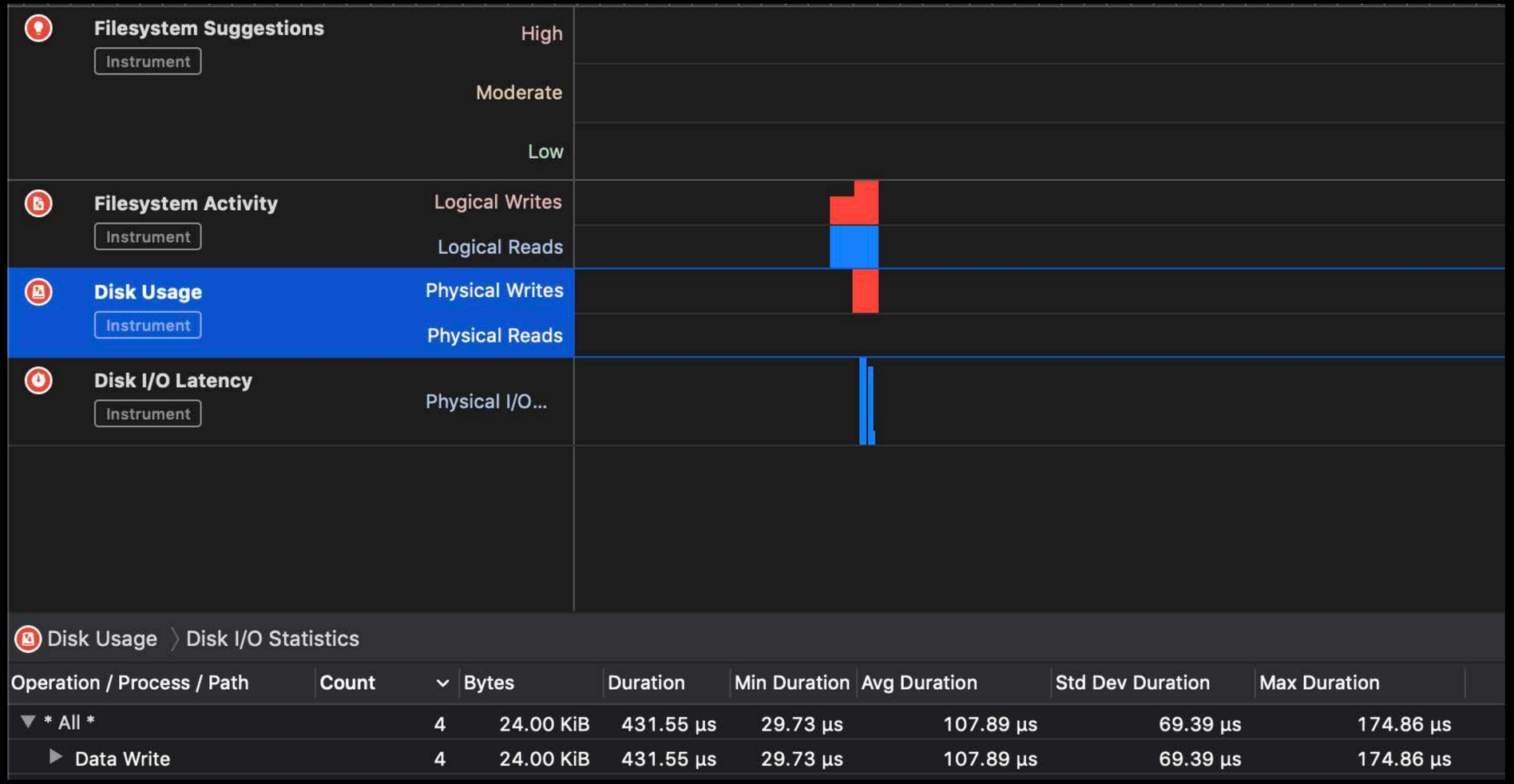
Syscall / Process	Count	Bytes	Duration	Min Duration	Avg Duration	Std Dev Duration	Max Duration
▼ * All *	37	16.55 KiB	945.08 µs	332 ns	25.54 µs	58.66 µs	262.89 µs
▶ fcntl	12	0 Bytes	18.87 µs	332 ns	1.57 µs	1.56 µs	6.20 µs
▶ pwrite	10	16.53 KiB	111.09 µs	3.12 µs	11.11 µs	10.16 µs	37.37 µs
▶ fsync	4	0 Bytes	419.17 µs	731 ns	104.79 µs	97.07 µs	199.16 µs
▶ stat64	4	0 Bytes	117.42 µs	13.51 µs	29.36 µs	18.04 µs	54.62 µs
▶ fstat64	3	0 Bytes	4.90 µs	518 ns	1.63 µs	1.29 µs	3.04 µs
▶ pread	2	24 Bytes	8.17 µs	1.55 µs	4.08 µs	3.58 µs	6.62 µs
▶ unlink	1	0 Bytes	262.89 µs	262.89 µs	262.89 µs	n/a	262.89 µs
▶ fstatfs64	1	0 Bytes	2.58 µs	2.58 µs	2.58 µs	n/a	2.58 µs

Multiple Statement Transaction



Syscall / Process	Count	Bytes	Duration	Min Duration	Avg Duration	Std Dev Duration	Max Duration
▼ * All *	37	16.55 KiB	945.08 µs	332 ns	25.54 µs	58.66 µs	262.89 µs
▶ fcntl	12	0 Bytes	18.87 µs	332 ns	1.57 µs	1.56 µs	6.20 µs
▶ pwrite	10	16.53 KiB	111.09 µs	3.12 µs	11.11 µs	10.16 µs	37.37 µs
▶ fsync	4	0 Bytes	419.17 µs	731 ns	104.79 µs	97.07 µs	199.16 µs
▶ stat64	4	0 Bytes	117.42 µs	13.51 µs	29.36 µs	18.04 µs	54.62 µs
▶ fstat64	3	0 Bytes	4.90 µs	518 ns	1.63 µs	1.29 µs	3.04 µs
▶ pread	2	24 Bytes	8.17 µs	1.55 µs	4.08 µs	3.58 µs	6.62 µs
▶ unlink	1	0 Bytes	262.89 µs	262.89 µs	262.89 µs	n/a	262.89 µs
▶ fstatfs64	1	0 Bytes	2.58 µs	2.58 µs	2.58 µs	n/a	2.58 µs

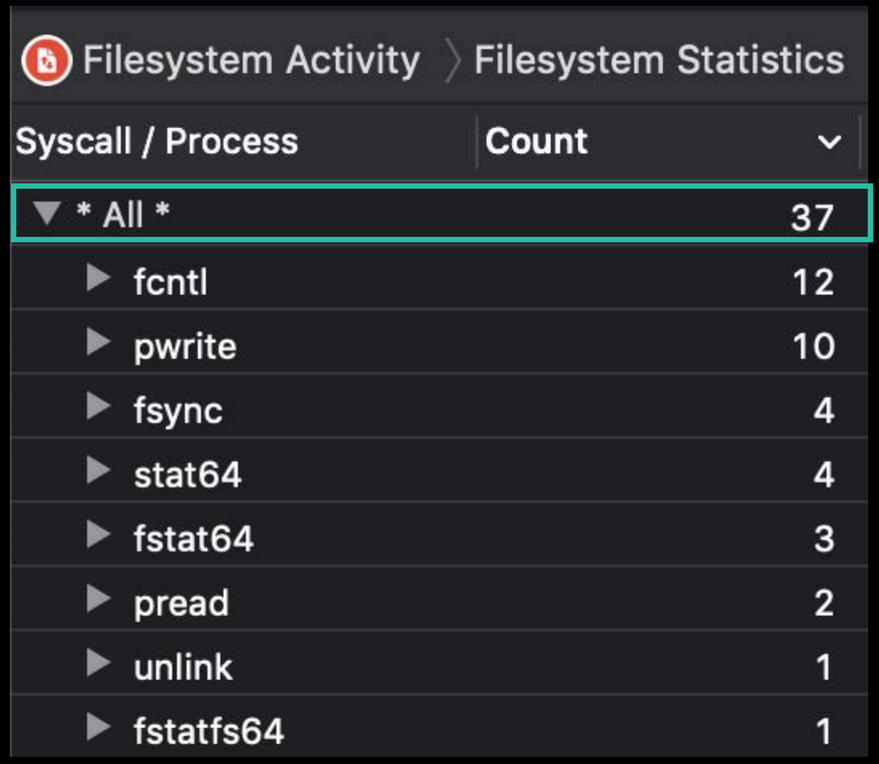
Multiple Statement Transaction



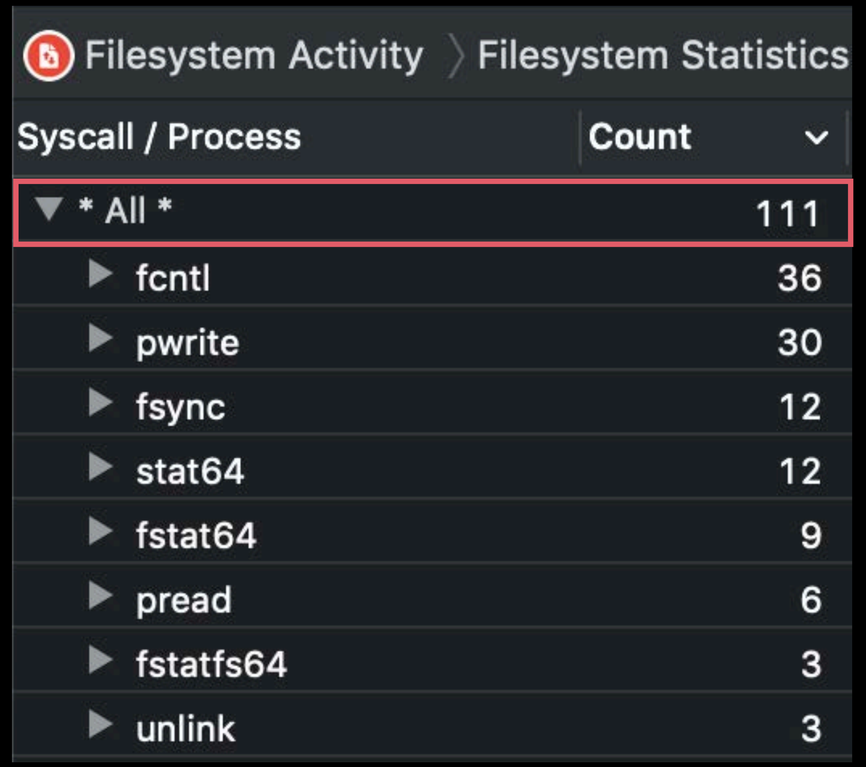
Multiple Statement Transaction

Transactions Comparison

Multiple Statement

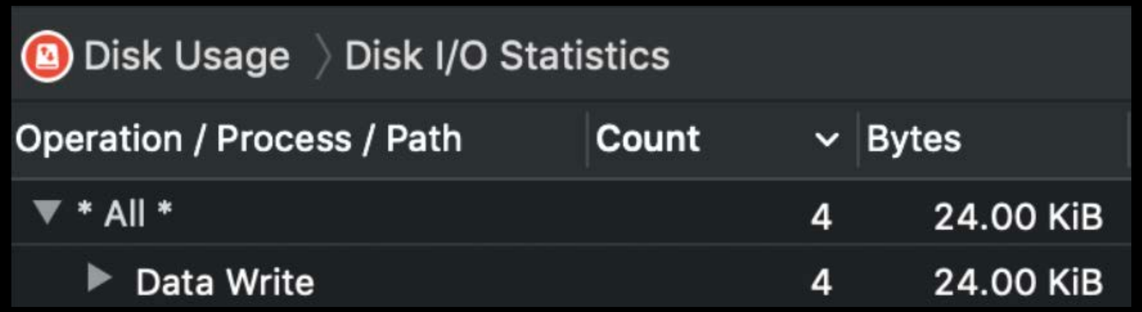


Single Statement

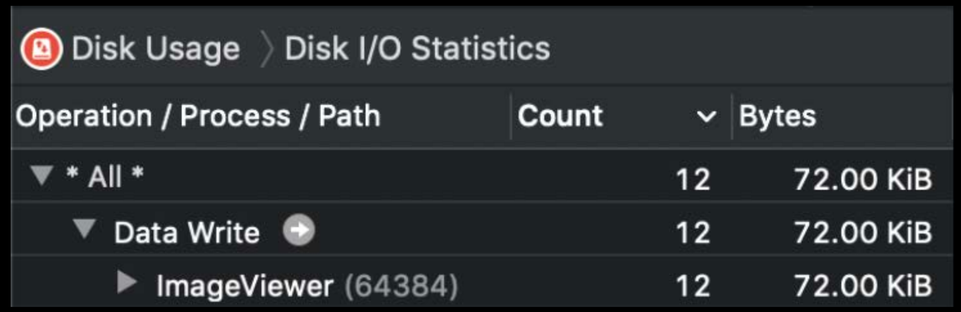


Transactions Comparison

Multiple Statement



Single Statement



Full Vacuum

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
▼ Data Write	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
▼ ImageViewer (65819)	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
/ImageViewer/full_vacuum.db-journal	18	108.00 KiB	932.60 µs	18.93 µs	51.81 µs	68.99 µs	276.55 µs
/ImageViewer/full_vacuum.db	9	60.00 KiB	231.75 µs	18.91 µs	25.75 µs	5.52 µs	35.35 µs

Full Vacuum

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
▼ Data Write	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
▼ ImageViewer (65819)	27	168.00 KiB	1.16 ms	18.91 µs	43.12 µs	57.25 µs	276.55 µs
/ImageViewer/full_vacuum.db-journal	18	108.00 KiB	932.60 µs	18.93 µs	51.81 µs	68.99 µs	276.55 µs
/ImageViewer/full_vacuum.db	9	60.00 KiB	231.75 µs	18.91 µs	25.75 µs	5.52 µs	35.35 µs

Incremental Vacuum

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
▼ Data Write	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
▼ ImageViewer (66075)	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
/ImageViewer/incremental_vacuum.db-journal	9	48.00 KiB	785.72 µs	23.22 µs	87.30 µs	99.60 µs	301.52 µs
/ImageViewer/incremental_vacuum.db	3	24.00 KiB	96.88 µs	31.62 µs	32.29 µs	805 ns	33.18 µs

Incremental Vacuum

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes	Total Laten...	Min Total L...	Avg Total L...	Std Dev To...	Max Total L...
▼ * All *	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
▼ Data Write	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
▼ ImageViewer (66075)	12	72.00 KiB	882.60 µs	23.22 µs	73.55 µs	88.51 µs	301.52 µs
/ImageViewer/incremental_vacuum.db-journal	9	48.00 KiB	785.72 µs	23.22 µs	87.30 µs	99.60 µs	301.52 µs
/ImageViewer/incremental_vacuum.db	3	24.00 KiB	96.88 µs	31.62 µs	32.29 µs	805 ns	33.18 µs

Vacuuming Comparison

Incremental

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes
▼ * All *	12	72.00 KiB
▼ Data Write	12	72.00 KiB
▼ ImageViewer (66075)	12	72.00 KiB
/ImageViewer/incremental_vacuum.db-journal	9	48.00 KiB
/ImageViewer/incremental_vacuum.db	3	24.00 KiB

Full

Disk Usage 〉 Disk I/O Statistics

Operation / Process / Path	Count	Bytes
▼ * All *	27	168.00 KiB
▼ Data Write	27	168.00 KiB
▼ ImageViewer (65819)	27	168.00 KiB
/ImageViewer/full_vacuum.db-journal	18	108.00 KiB
/ImageViewer/full_vacuum.db	9	60.00 KiB

Summary

Summary

Apply these lessons

Summary

Apply these lessons

Profile with File Activity Instrument

Summary

Apply these lessons

Profile with File Activity Instrument

Continue optimizing storage!

More Information

developer.apple.com/wwdc19/419

Making Apps with Core Data	WWDC 2019
Performance, Power, Crashes, and Debugging Lab	Friday, 3:00


WWDC19