Persistent storage tailored for containers

Quentin “mefyl” Hocquet
mefyl@infinit.sh

CTO @ Infinit

Version 1.2-26-gbcb3c69

Plan

Containers and persistent storage

Infinit storage platform

Dive-in

Demo

Q&A

Containers and persistent storage

Containers are fast, scalable and flexible.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

However containers tend to be ***stateless***, which can be quite limiting. We need ***persistent storage*** for containers.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

However containers tend to be ***stateless***, which can be quite limiting. We need ***persistent storage*** for containers.

- It should be created and started as easily as a container.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

However containers tend to be ***stateless***, which can be quite limiting. We need ***persistent storage*** for containers.

- It should be created and started as easily as a container.
- It should be able to scale with your container pool.

Containers and persistent storage

Containers are fast, scalable and flexible.

- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

However containers tend to be ***stateless***, which can be quite limiting. We need ***persistent storage*** for containers.

- It should be created and started as easily as a container.
- It should be able to scale with your container pool.
- It should work the same way for development, tests, production, …

Containers and persistent storage

Containers are fast, scalable and flexible.

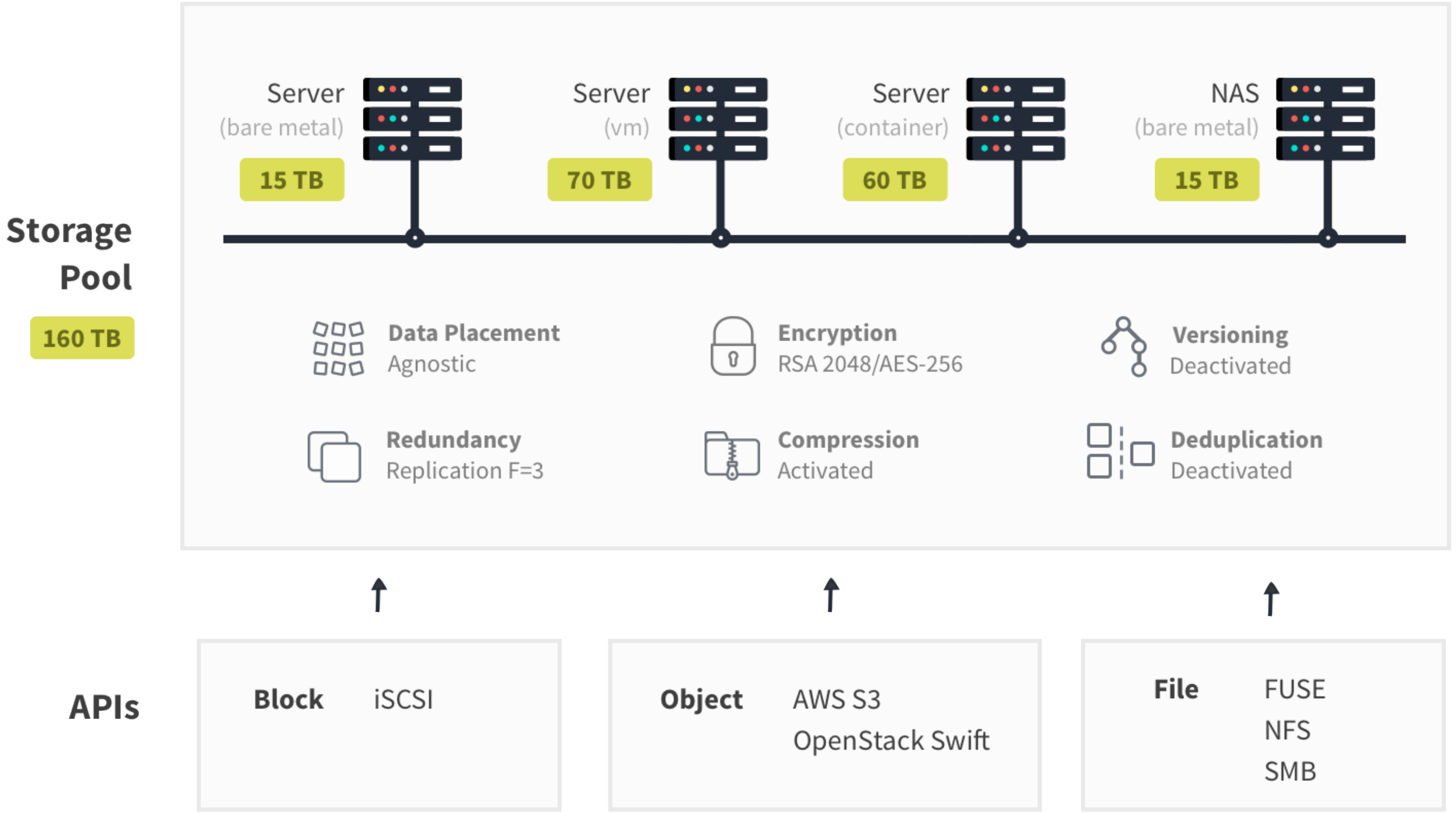
- Fast and easy to start and stop.
- Fast and easy to scale.
- Unified from development to production.
- Yet customizable for every situation.

However containers tend to be ***stateless***, which can be quite limiting. We need ***persistent storage*** for containers.

- It should be created and started as easily as a container.
- It should be able to scale with your container pool.
- It should work the same way for development, tests, production, …
- It should adapt to all situations.

Infinit storage platform

Infinit is a storage platform designed with containers in mind. It ***aggregates*** nodes local storage into a single virtual pool and provides ***several APIs*** on top of it.



Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.
- Nodes can come and go at will.

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.
- Nodes can come and go at will.

Infinit follows the container philosophy:

- Can be ***created and run*** as seamlessly as a container.

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.
- Nodes can come and go at will.

Infinit follows the container philosophy:

- Can be ***created and run*** as seamlessly as a container.
- Can ***scale with you container pool***.

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.
- Nodes can come and go at will.

Infinit follows the container philosophy:

- Can be ***created and run*** as seamlessly as a container.
- Can ***scale with you container pool***.
- Is the ***same in all situations***: development, unit tests, production …

Infinit storage platform

The Infinit platform is ***truly distributed***: all nodes are equal.

- Works the same with 1 or 10k nodes.
- Nodes can come and go at will.

Infinit follows the container philosophy:

- Can be ***created and run*** as seamlessly as a container.
- Can ***scale with you container pool***.
- Is the ***same in all situations***: development, unit tests, production …
- Can be ***configured*** for each situation: encryption, redundancy, compression, …

Infinit design

Infinit fundamental principles:

Infinit design

Infinit fundamental principles:

- Federate all nodes in an ***overlay network*** for lookup and routing.

Infinit design

Infinit fundamental principles:

- Federate all nodes in an ***overlay network*** for lookup and routing.
- Store data as blocks in a ***distributed hashtable*** (key-value store) with a ***per-block consensus***.

Infinit design

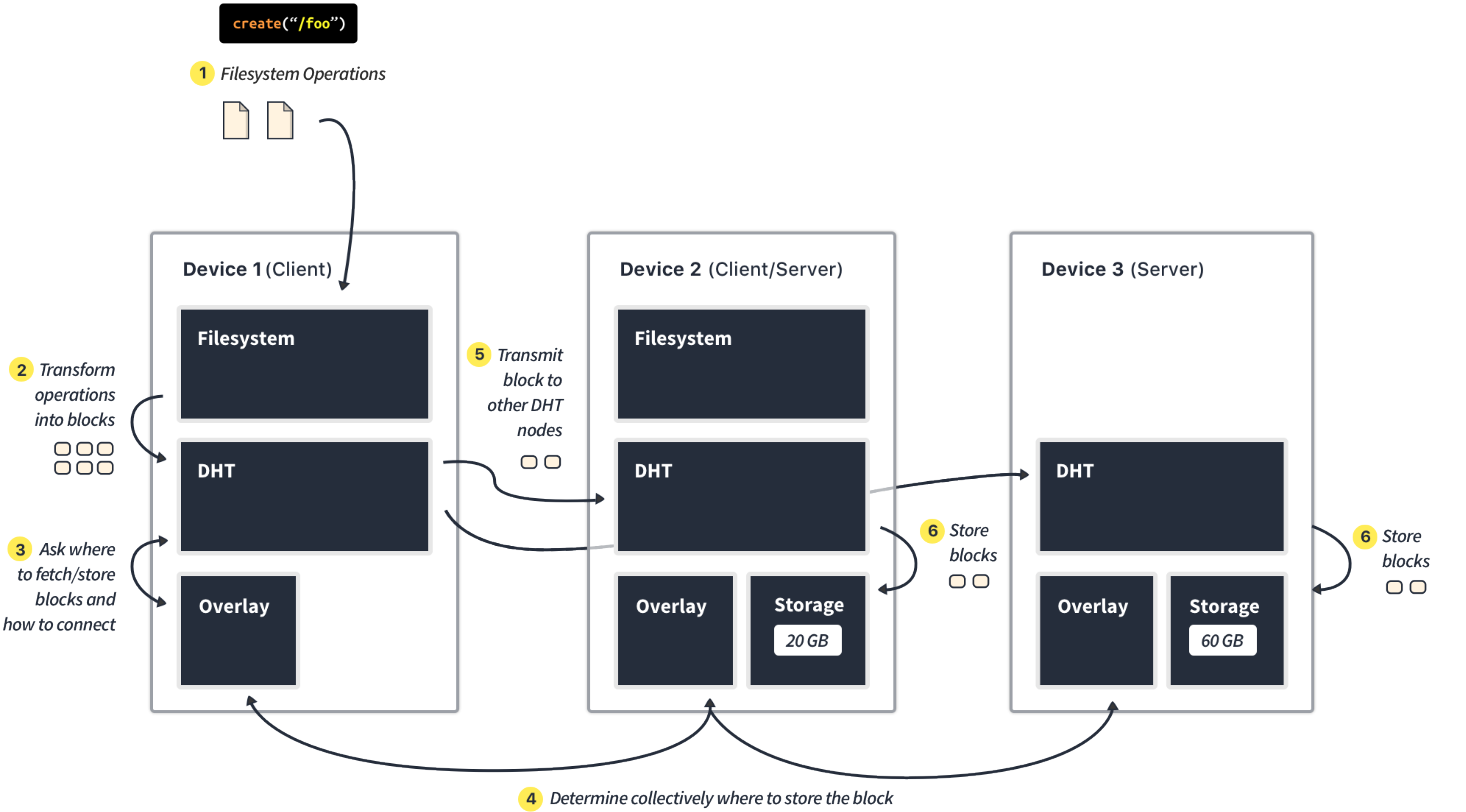
Infinit fundamental principles:

- Federate all nodes in an ***overlay network*** for lookup and routing.
- Store data as blocks in a ***distributed hashtable*** (key-value store) with a ***per-block consensus***.
- Use ***cryptographic access control*** to dispense from any leader.

Infinit design

Infinit fundamental principles:

- Federate all nodes in an ***overlay network*** for lookup and routing.
- Store data as blocks in a ***distributed hashtable*** (key-value store) with a ***per-block consensus***.
- Use ***cryptographic access control*** to dispense from any leader.
- Use ***symmetrical operations*** to ensure resilience and flexibility.



```
create("/foo")
```

1 Filesystem Operations

2 Transform operations into blocks

3 Ask where to fetch/store blocks and how to connect

Device 1 (Client)

Filesystem

DHT

Overlay

5 Transmit block to other DHT nodes

Device 2 (Client/Server)

Filesystem

DHT

Overlay

Storage

20 GB

6 Store blocks

Device 3 (Server)

DHT

Overlay

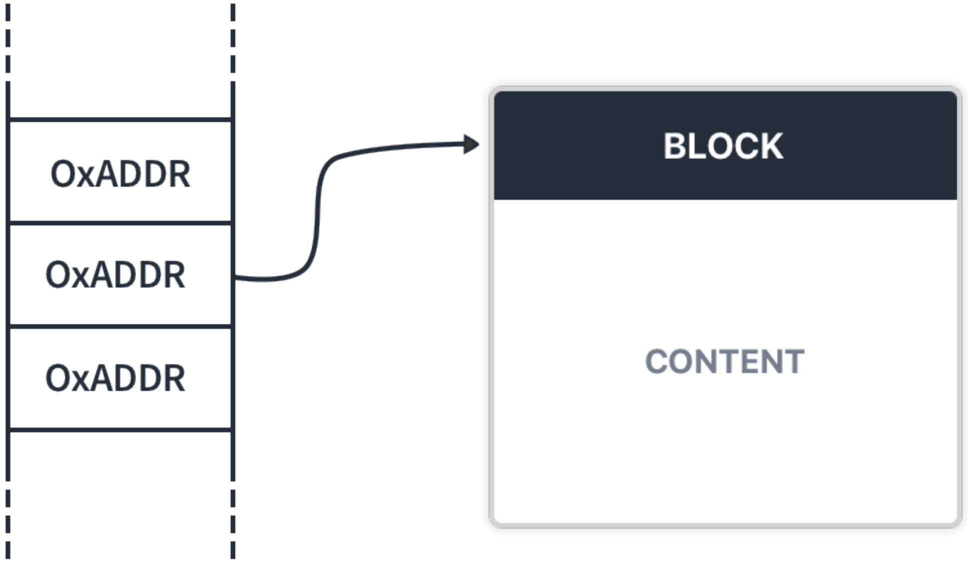
Storage

60 GB

6 Store blocks

4 Determine collectively where to store the block

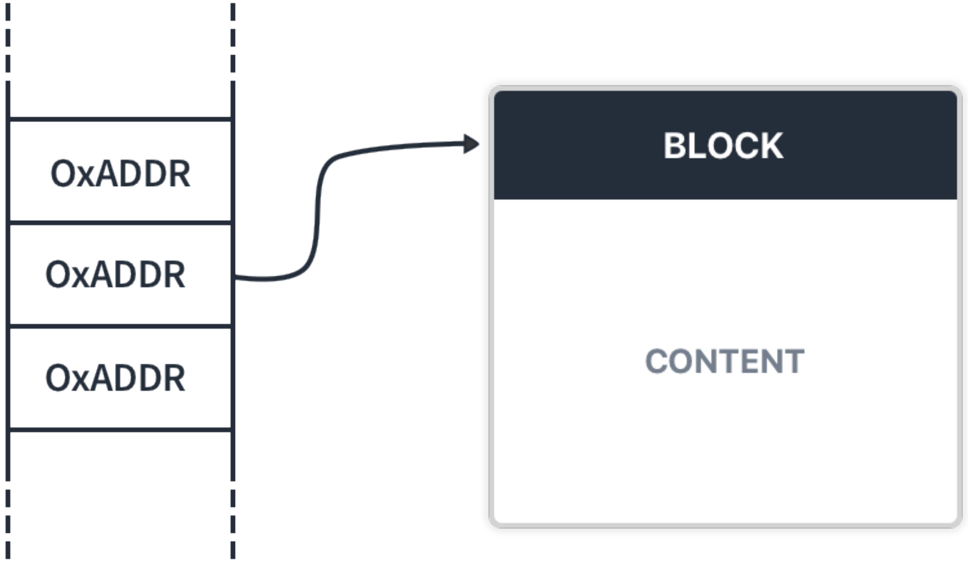
Dive-in: DHT blocks



Dive-in: DHT blocks

Mutable blocks

- Subject to conflicts.
- Subject to invalidation.
- Hard to certify and cipher.



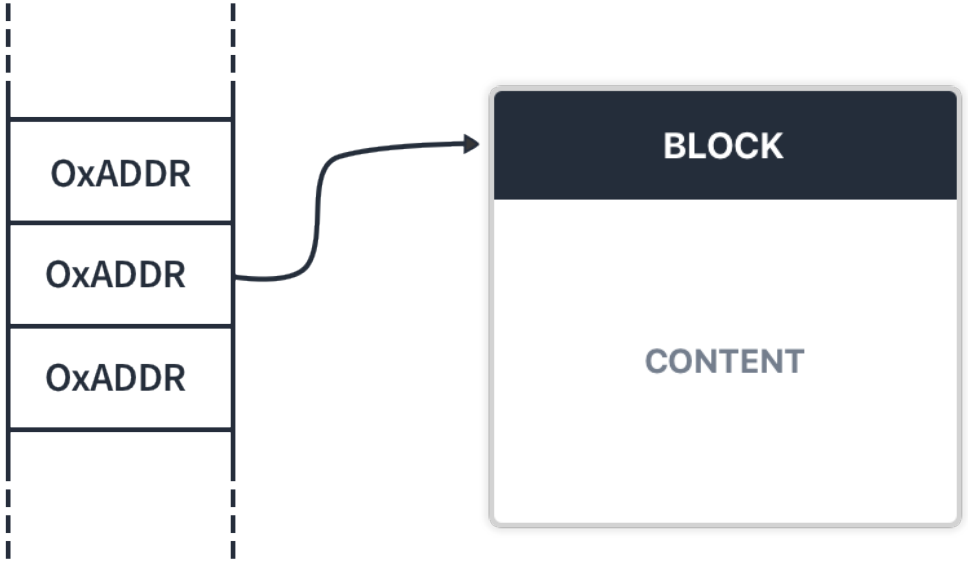
Dive-in: DHT blocks

Mutable blocks

- Subject to conflicts.
- Subject to invalidation.
- Hard to certify and cipher.

Immutable blocks

- No conflicts.
- No invalidation: cachable forever.
- Easy to certify since content addressable: `address = hash(contents)`.



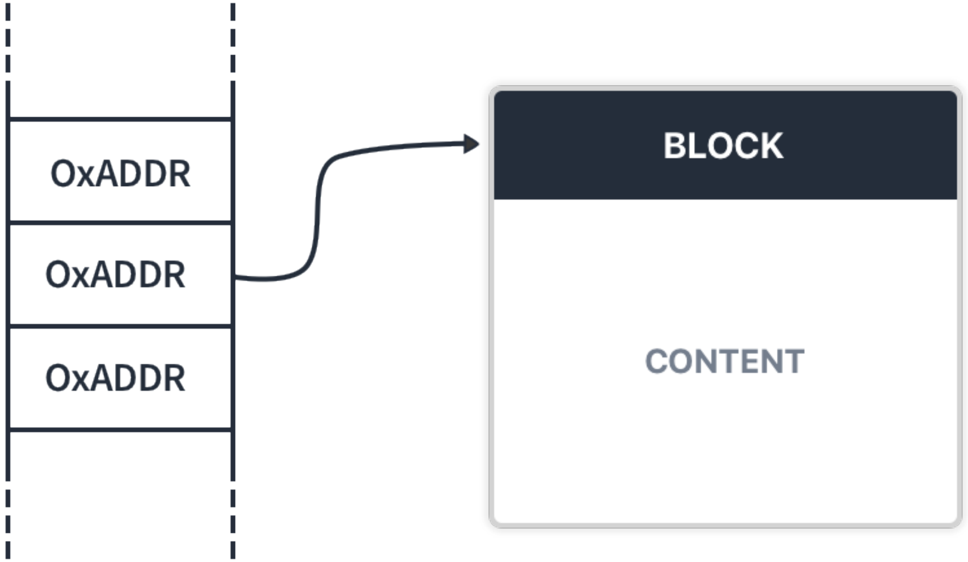
Dive-in: DHT blocks

Mutable blocks

- Subject to conflicts.
- Subject to invalidation.
- Hard to certify and cipher.

Immutable blocks

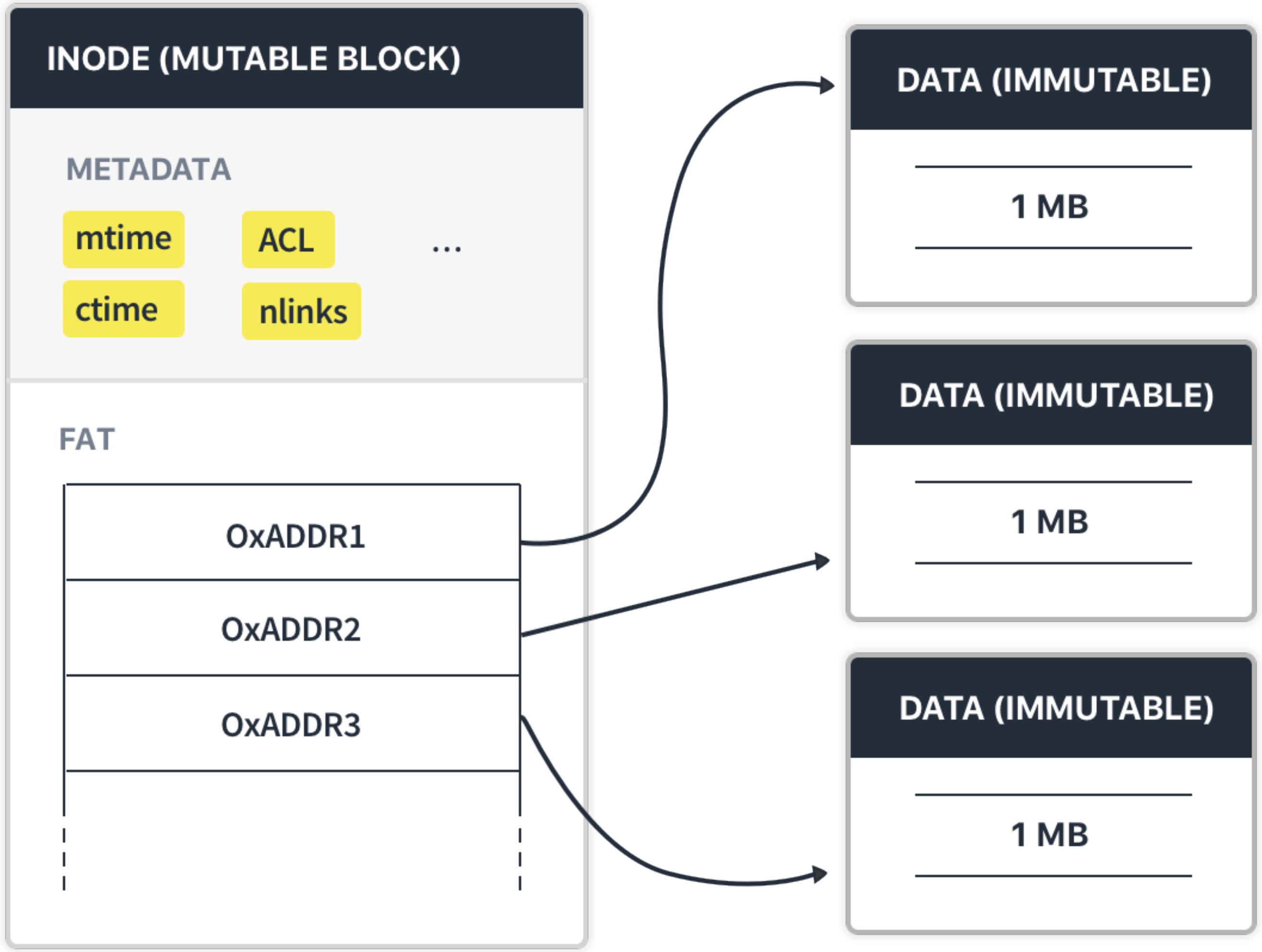
- No conflicts.
- No invalidation: cachable forever.
- Easy to certify since content addressable: `address = hash(contents)`.



Immutable block are ***cheap*** to write and read, fetchable from ***any source*** and ***cachable*** permanently on-disk.

Dive-in: filesystem layer

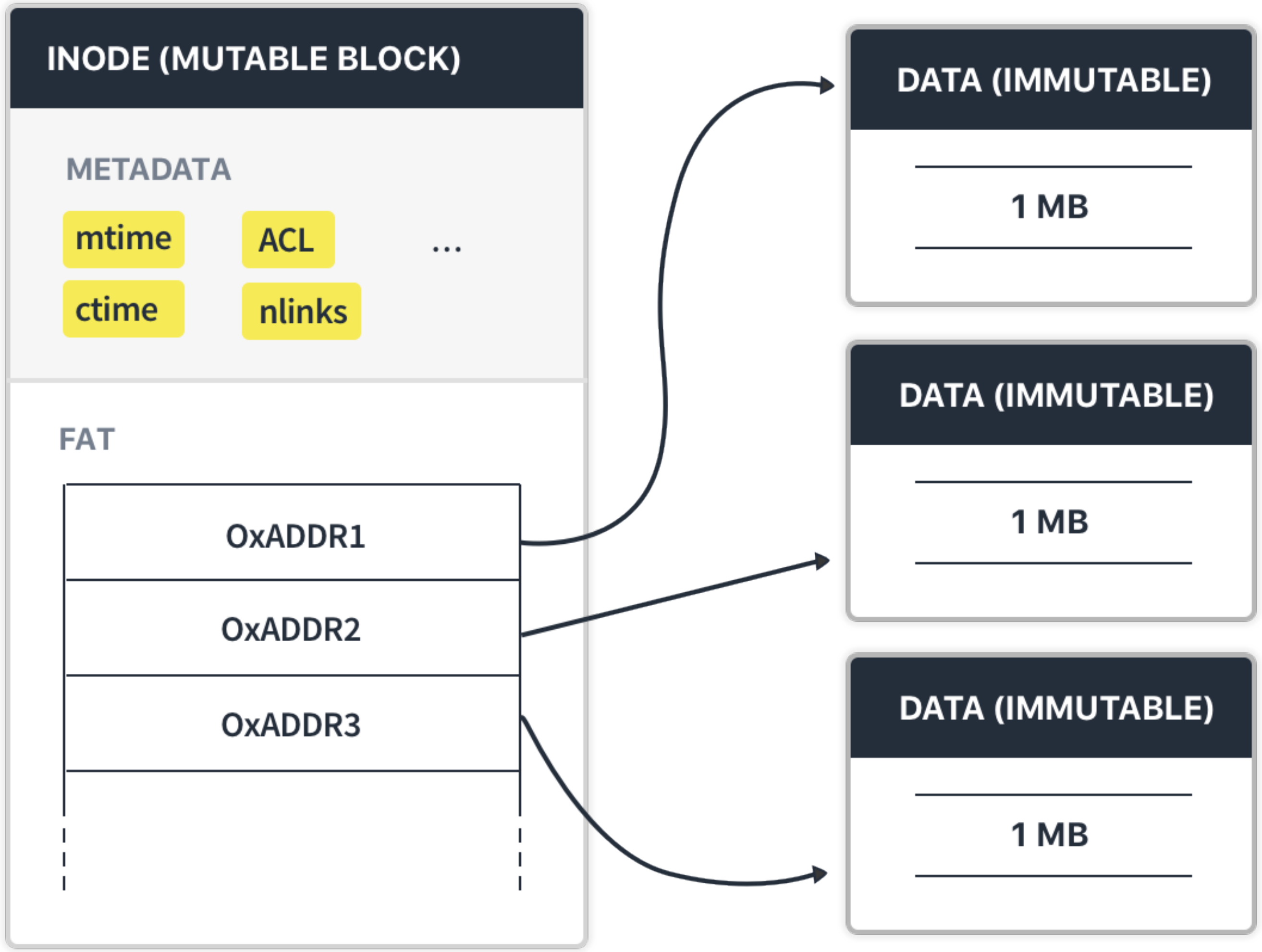
A file is mostly a mutable block with metadata and a FAT of immutable block.



Dive-in: filesystem layer

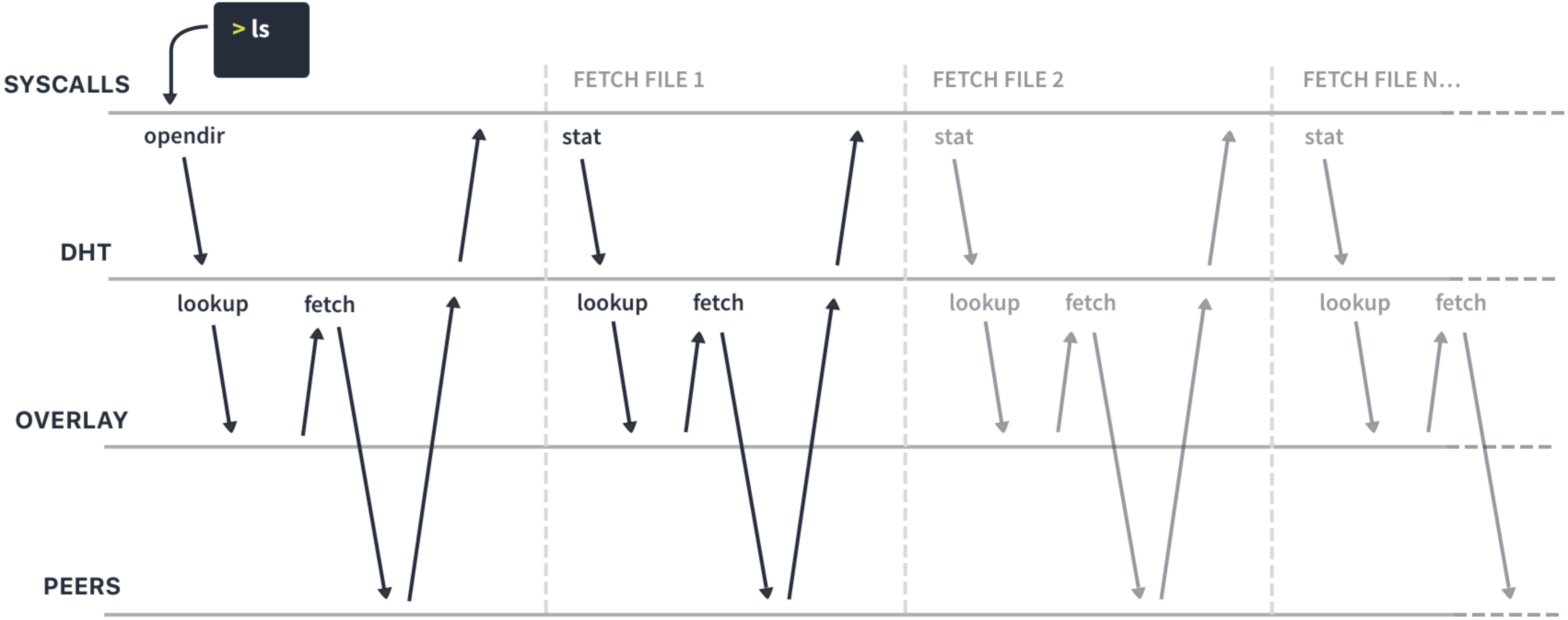
A file is mostly a mutable block with metadata and a FAT of immutable block.

File contents is ***cachable*** at will, ***cheap*** and ***atomic*** writes.



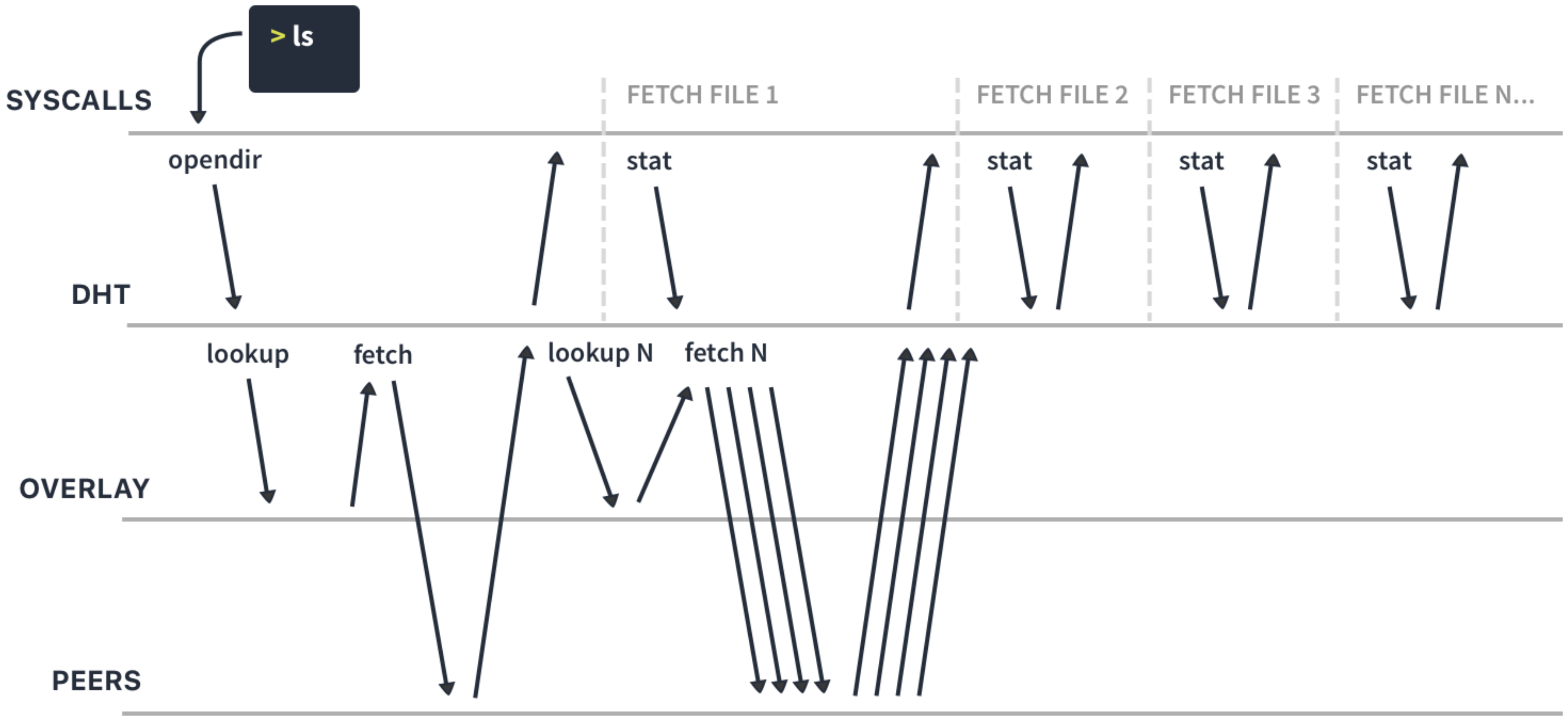
Dive-in: filesystem layer

The POSIX API is inherently ***sequential***. We are highly ***parallel***.



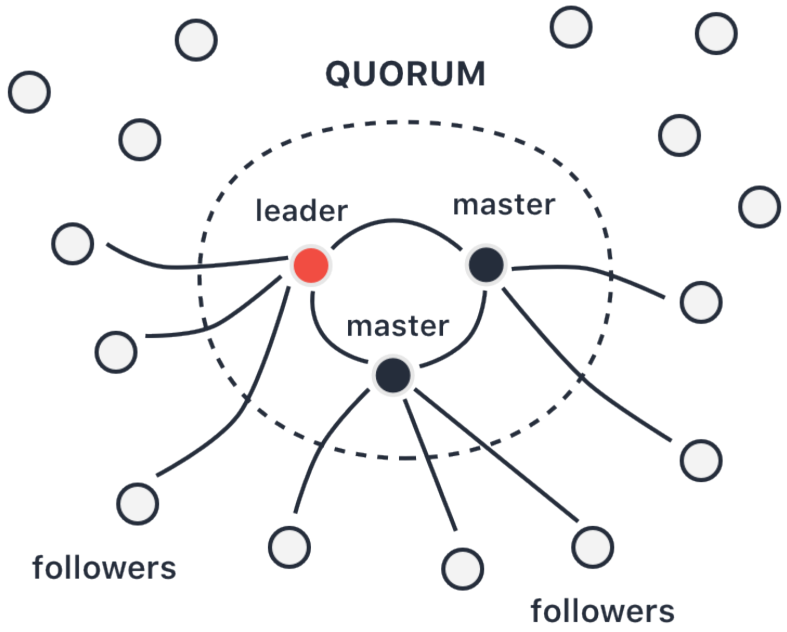
Dive-in: filesystem layer

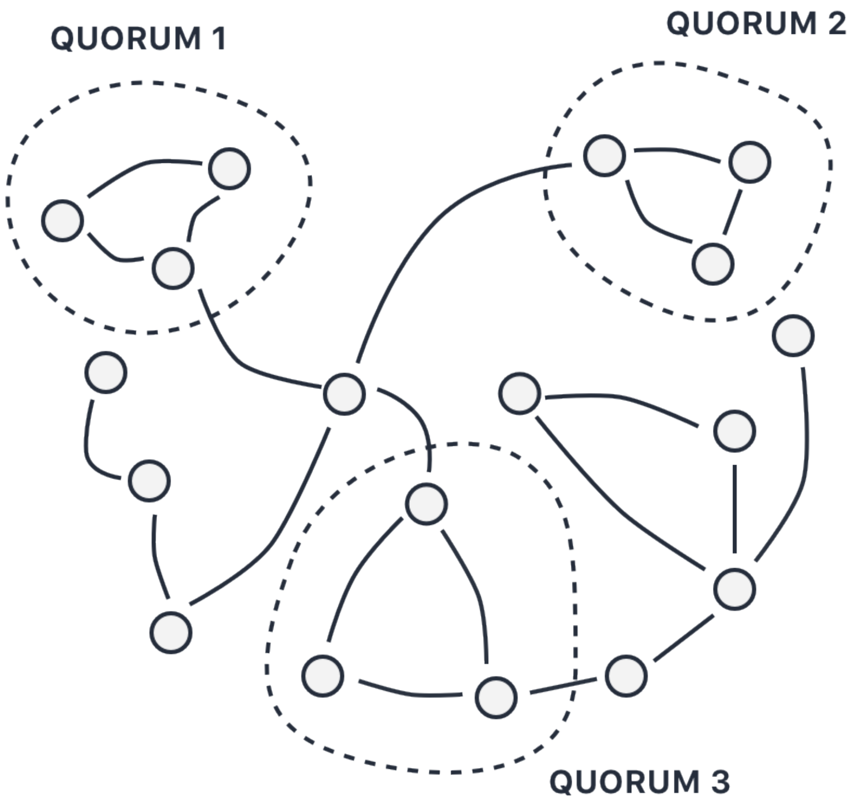
Directories prefetching and ***files look-ahead*** enables batching and pipelining.



Dive-in: consensus

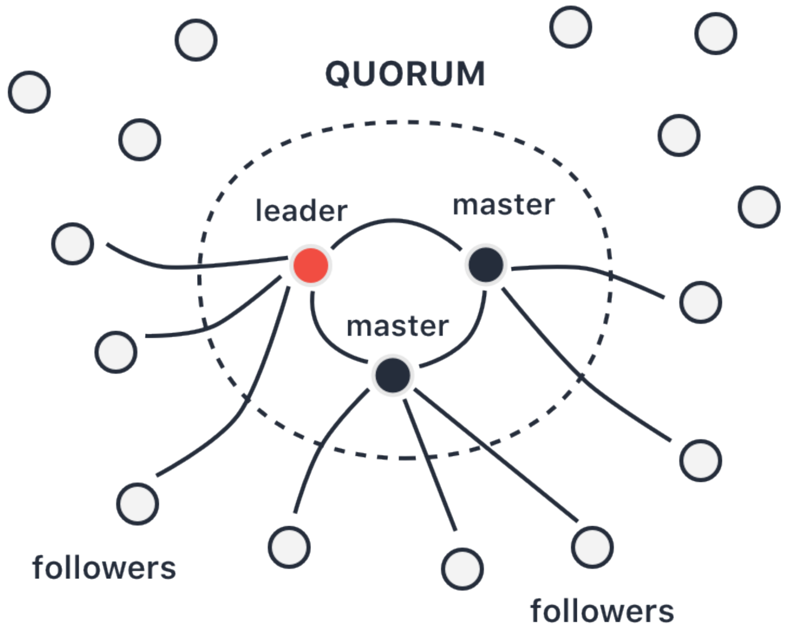
Each block is managed by a specific quorum of node with a variable composition, running multipaxos.

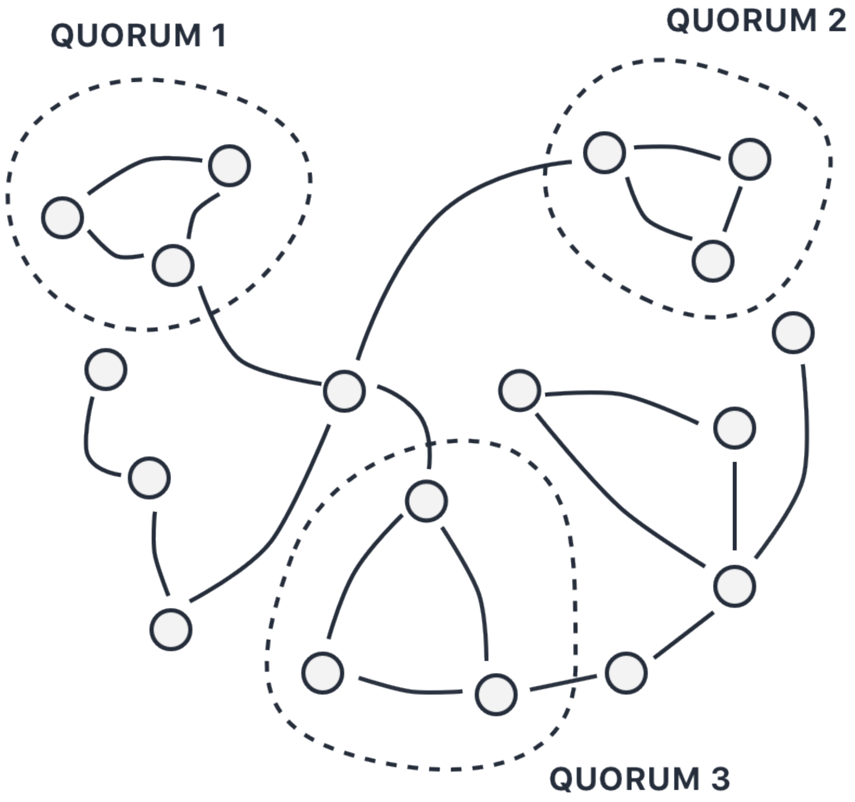




Dive-in: consensus

Each block is managed by a specific quorum of node with a variable composition, running multipaxos.

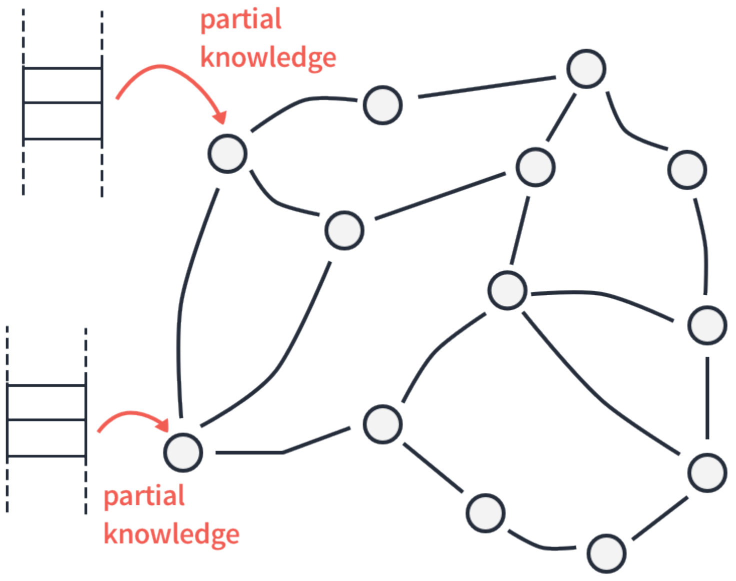




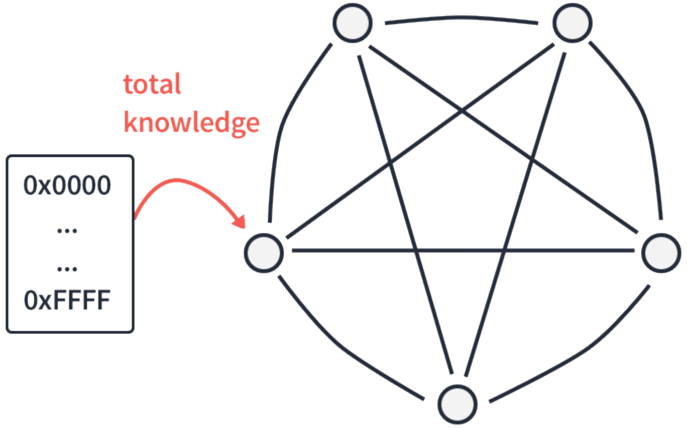
No failure point or bottleneck, strong read after write consistency.

Dive-in: overlay

The overlay algorithm is one major customization point of the platform.



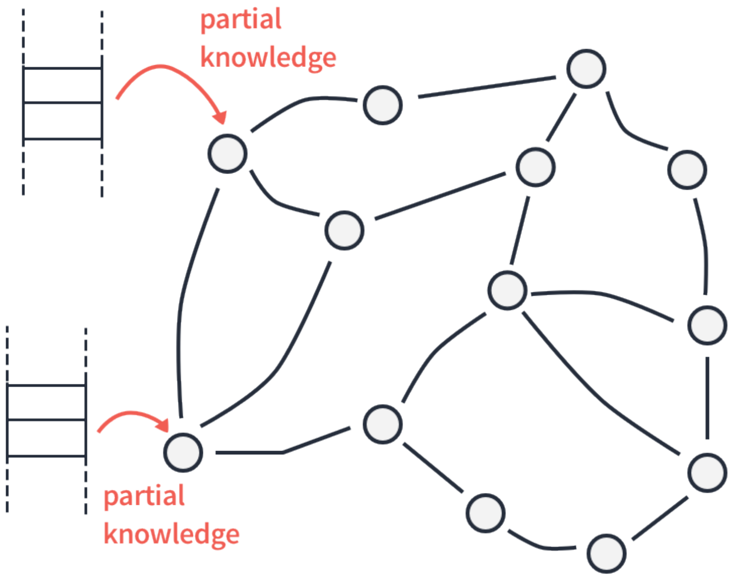
Kelips



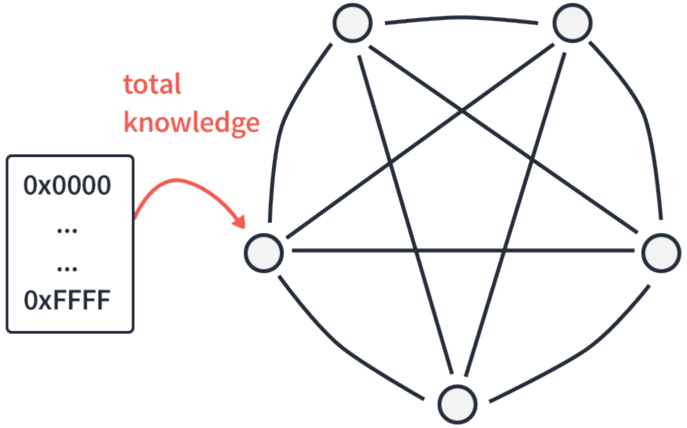
Kouncil

Dive-in: overlay

The overlay algorithm is one major customization point of the platform.



Kelips



Kouncil

Data placement: rack-aware, zone-aware, reliability-aware, ensure local copies, ...

Demo!

Let's persist that storage!

Questions ?

infinit