Motivation – How to Agree on Total Order?
Motivation – How to Agree on Total Order?

What’s slot 1?
It’s ‘E’

What’s slot 5?
It’s ‘9’
Motivation – How to Agree on Total Order?

- What's slot 1?
  - It's 'E'

- What's slot 5?
  - It's '9'

Shared Log:

E E C S 5 9 1 !
Motivation – How to Build a Shared Log?

Server + ssd

- Performance limited by server’s bandwidth
Motivation - How to Build a Shared Log?

- Performance limited by server’s bandwidth

Server + ssd

![X Mark]
Motivation – How to Build a Shared Log?

- Client communicates directly with flash units
- Increased throughput

CORFU

slot 0

slot 5
Motivation – How to Build a Shared Log?

- Client communicates directly with flash units
  - Increased throughput
Design – Client API

- `append(entry b)`: get the position `l`
- `read(log position l)`: get the entry
- `trim(log position l)`: garbage collection
- `fill(log position l)`: indicate hole
Design – Client API

- append(entry b): get the position l
- read(log position l): get the entry
- trim(log position l): garbage collection
- fill(log position l): indicate hole
Design – Client API

- append(entry b): get the position l
- read(log position l): get the entry
- trim(log position l): garbage collection
- fill(log position l): indicate hole
Each log position is **mapped** to storage pages in cluster.
Design – Architecture

Each log position is mapped to storage pages in cluster

Controller (for every flash unit)
Each log position is **mapped** to storage pages in cluster.
Design – Architecture

Tail-finding mechanism

Each log position is **mapped** to storage pages in cluster
Design – Architecture

Each log position is mapped to storage pages in cluster.

Replication (single pos map to multiple flash units)
Design – Architecture

Each log position is mapped to storage pages in cluster.

Tail-finding mechanism

Map
log pos→ flash page
(maintained by clients)

Replication
(single pos map to multiple flash units)

Controller
(for every flash unit)
Detail – Controller for Flash Unit

- Write-once semantics
  - not trimmed => each slot can only be written once
Detail – Controller for Flash Unit

- Write-once semantics
  - not trimmed => each slot can only be written once
Write-once semantics
- not trimmed => each slot can only be written once

Seal => used for map change
- set epoch number
- reject requests with smaller epoch

**epoch #1**

<table>
<thead>
<tr>
<th>flash</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td>02</td>
</tr>
<tr>
<td></td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>04</td>
</tr>
</tbody>
</table>
Detail – Map

- Map log position to flash pages
- Map is maintained by clients
  - need to agree on a single map
- Change of map
  - consensus algorithm => same map among clients
    - infrequently (failure/need more log position)
  - epoch + seal => old map get rejected
Detail – Tail-Finding Mechanism

- Solution 1: Let the client find the tail
  - utilize the write-once semantics
  - contention + congestion => bad performance
- Solution 2: Sequencer to assign log position
  - hole => fill command
  - only optimization, cannot rely on the sequencer
Solution 1: Let the client find the tail
  - utilize the write-once semantics
  - contention + congestion => bad performance
Solution 2: Sequencer to assign log position
  - hole => fill command
  - only optimization, cannot rely on the sequencer
Map will map a log position to multiple flash pages (in different flash units)

- $f+1$ replicas, data be visible only after it reaches all replicas

How to write?

- Chain replication (write in deterministic order)

- pos 1
  - map to
  - page00 on A
  - page11 on B
  - page23 on C
Evaluation

Figure 5: Latency for CORFU operations on different flash unit configurations.

Figure 7: Throughput for random reads and appends.