OUTLINE

• What is CORFU
• Why CORFU is good
• How does CORFU work
• Evaluation
WHAT IS CORFU
WHAT IS CORFU

• Island in Greece
• Next to Paxos
WHAT IS CORFU

- CORFU: Clusters of Raw Flash Units
- A Distributed Shared Log
- Compare with Paxos:
  - Paxos: distributed shared same log
  - CORFU: partition log into different flash drive
WHAT IS CORFU

• Simple Example:
  Control Plane

Sequencer
Layout
User
Log
WHAT IS CORFU

• Simple Example:
  Control Plane
  Sequencer
  Layout
  User
  A = 2
  User wants to write
  Log

Log
WHAT IS CORFU

• Simple Example:
  Control Plane

Sequencer
Layout

User
A = 2

Log

1. give me server address
WHAT IS CORFU

• Simple Example:
  Control Plane

Sequencer
Layout

User
A = 2

2. Servers are at ...

Log
WHAT IS CORFU

- Simple Example:
  Control Plane

  Sequencer

  Layout

  3. give me next available address

  User
  A = 2

  Log
WHAT IS CORFU

- Simple Example:

  Control Plane
  4. next address = 0

  Sequencer

  Layout

  User
  A = 2

  Log
WHAT IS CORFU

• Simple Example:

  Control Plane
  4. next address = 0

Sequencer

Layout

User

A = 2

Log

A=2

4. next address = 0
WHAT IS CORFU

• Simple Example:
  Control Plane

Sequencer
Layout

User
A = 2

Log

User wants to read

A=2
A=2
A=2
A=2
A=2
WHAT IS CORFU

- Simple Example:

  Control Plane

  1. Next address = 1

  User: A = 6

  User writes new value

  Sequencer

  Layout

A = 2
WHAT IS CORFU

• Simple Example:

Control Plane
4. next address = 1

Sequencer

User
A = 6

Layout

User writes new value

A=2  A=6
WHAT IS CORFU

- Simple Example:
  Control Plane

Sequencer

User
A = 6

User wants to read

A=2
A=6

A?
WHAT IS CORFU

• Simple Example:
  Control Plane

  Sequencer
  Layout
  User
  A = 6

<table>
<thead>
<tr>
<th>Last address = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=2</td>
</tr>
</tbody>
</table>

A?
WHAT IS CORFU

• Simple Example:
  Control Plane

Sequencer

Layout

User

A = 6

Last address = 1

A = 6
WHAT IS CORFU

• Simple Example:

Sequencer
Layout

User A = 6
User B

A=2
A=6
WHAT IS CORFU

- Simple Example: \(A@1, B@2\)

Sequencer

User
\[ A = 6 \]

Layout

User
\[ B = 5 \]
WHAT IS CORFU

• Simple Example:

\[ \text{A@1, B@2} \]

Sequencer

Layout

User A = 6

User B = 5

Transaction

A = 1;
B = 9;

A = 2

A = 6

B = 5
WHAT IS CORFU

• Simple Example:

\[ A@3, \ B@3 \]

Sequencer

Layout

User

A = 6

User

B = 5

Transaction

A = 1;

B = 9;
WHY CORFU IS GOOD?

• What’s special?

Sequencer is fast
200K tokens/s;
assuming 4KB entries and two-way replication
Used to be traditional I/O bottleneck
WHY CORFU IS GOOD?

• What’s special?

Client centric:
Fast datapath: directly writing to flash drive
As fast as the sequencer can assign them 64-bit tokens
WHY CORFU IS GOOD?

• What’s special?

Client centric:
Reduces the complexity, cost, latency and power consumption of the flash units
WHY CORFU IS GOOD?

• Flash storage is an ideal medium for shared log designs

• flash provides fast, contention free random reads, hundreds of clients can concurrently access a shared log
WHY CORFU IS GOOD?

• Flash storage has problems

• Flash is **read** and **written** in increments of pages (typically of size 4KB)

• Before a page can be **overwritten**, it must be erased in size of multi-page blocks (of size 256KB)

• Flash wears out as a page is erased and overwritten
WHY CORFU IS GOOD?

• Flash storage solve problem by Flash Translation Layer (FTL) within an SSD

• SSDs implement a logical address space over raw flash

• Wear leveling
WHY CORFU IS GOOD?

• The best data structure for a single flash device is a log
  
  • It is always best to write sequentially to flash
  
  • Faster than random access
  
  • Even worn out
WHY CORFU IS GOOD?

• Other features
  • scalable: add more using module
  • fault tolerant: via chain replication, F+1 replicas
  • strong consistency
WHY CORFU IS GOOD?

• CORFU organizes an entire cluster of flash drives as a single log. This log is distributed across multiple drives and shared by multiple clients.
WHY CORFU IS GOOD?

• Shared log

• Failure atomicity and node recovery

• Recovery from multicast packet loss

• Distributed storage systems for consistent remote mirroring
WHY CORFU IS GOOD?

- Distributed log
  - Not limited to bandwidth of primary server
  - Even age distribution
HOW DOES CORFU WORK

• CORFU presents applications running on clients
• With the abstraction of a shared log
• Implemented over a cluster of flash units
• By a client-side library
HOW DOES CORFU WORK

- 4 main APIs

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$append(b)$</td>
<td>Append an entry $b$ and return the log position $\ell$ it occupies</td>
</tr>
<tr>
<td>$read(\ell)$</td>
<td>Return entry at log position $\ell$</td>
</tr>
<tr>
<td>$trim(\ell)$</td>
<td>Indicate that no valid data exists at log position $\ell$</td>
</tr>
<tr>
<td>$fill(\ell)$</td>
<td>Fill log position $\ell$ with junk</td>
</tr>
</tbody>
</table>

Figure 2: API exposed by CORFU to applications
HOW DOES CORFU WORK

• Three functions
  
  • A mapping function from logical positions in the log to flash pages on the cluster of flash units.
  
  • A tail-finding mechanism for finding the next available logical position on the log for new data.
  
  • A replication protocol to write a log entry consistently on multiple flash pages.
HOW DOES CORFU WORK

• To read data at a specific log position

• client-side library uses the mapping function to find the appropriate flash page

• directly issues a read to the device where the flash page is located.
HOW DOES CORFU WORK

• Challenge:

• single-copy semantics for the shared log

• flash units could fail and clients could crash
HOW DOES CORFU WORK

• Assumptions and Flash Unit Requirements
  • reads and writes on an address space of fixed-size pages
  • ‘write-once’ semantics on the flash unit’s address space
HOW DOES CORFU WORK

• Write-once

  • **Reads** on pages that have **not** yet been **written** should return an error code (error unwritten)

  • **Writes** on pages that have **already** been **written** should also return an error code (error overwritten)
HOW DOES CORFU WORK

• Should support Trim

• allows clients to indicate that the flash page is not in use anymore
HOW DOES CORFU WORK

• Should support seal

  • Each incoming message to a flash unit is tagged with an epoch number.

  • When a particular epoch number is sealed at a flash unit,

    • Reject all subsequent messages sent with an epoch equal or lower to the sealed epoch.
HOW DOES CORFU WORK

• Mapping

• **local**, read-only replica of a **data structure**
  called a projection that carves the log into disjoint ranges

• Within each range in the log, positions are mapped to flash pages in the corresponding list of extents via a simple, **deterministic** function.
HOW DOES CORFU WORK

- Mapping
  - A local, read-only replica of a data structure called a projection that carves the log into disjoint ranges.
  - Within each range in the log, positions are mapped to flash pages in the corresponding list of extents via a simple, deterministic function.

Example Projection →
Range [0 – 40K) is mapped to F0 and F1. Range [40K – 80K) is mapped to F2 and F3.

Figure 3: Example projection that maps different ranges of the shared log onto flash unit extents.
HOW DOES CORFU WORK

• for replication, each extent is associated with a replica set of flash units rather than just one unit
HOW DOES CORFU WORK

- Find position 45K in the log maps
- Extents on units F2 to F3
- With the round-robin function, the resulting page would be F2: 2500.

Figure 3: Example projection that maps different ranges of the shared log onto flash unit extents.
HOW DOES CORFU WORK

• Change of mapping - reconfiguration

• each client observes a totally ordered sequence of projections as the position-to-page mapping evolves over time; we call a projection’s position in this sequence its **epoch**.

• all the messages are tagged with the projection’s epoch.
HOW DOES CORFU WORK

• reconfiguration

• Use auxiliary

• is a durably stored sequence of projections

• the position of the projection in the sequence is equivalent to its epoch.
HOW DOES CORFU WORK

• reconfiguration - two steps

• 1. Sealing the current projection

  • send seal to flash drives that maps differently

  • flash units will reject inflight messages – writes as well as reads – sent to them in the context of the sealed projection

  • When clients receive these rejections, they realize that the current projection has been sealed, and wait for a new projection to be installed
HOW DOES CORFU WORK

• reconfiguration - two steps

• 2. Writing the new projection at the auxiliary
  
  • attempts to write the new projection $\Pi_{i+1}$ at the $(i + 1)\text{th}$ position in the auxiliary

  • If already exist, aborts its own reconfiguration, reads the existing projection at position $(i + 1)$ in the auxiliary, and uses it as its new current projection.
HOW DOES CORFU WORK

• reconfiguration - two steps

  1. Writing the new projection at the auxiliary
  2. (i + 1)th position

• attempts to write the new projection $P_{i+1}$ at the (i + 1)th position

• If already exist, aborts its own reconfiguration, reads the existing projection at position (i + 1) in the auxiliary, and uses it as its new current projection.

Figure 4: Sequence of projections: When $F_6$ fails in (A) with the log tail at 50K, clients move to (B) in order to replace $F_6$ with $F_8$ for new appends. Once old data on $F_6$ is rebuilt, $F_8$ is used in (C) for reads on old data as well. When the log tail goes past 80K, clients add capacity by moving to (D).
HOW DOES CORFU WORK

• Finding the tail in CORFU

• safety-under-contention

• uses a dedicated sequencer that assigns clients ‘tokens’, corresponding to empty log positions.
HOW DOES CORFU WORK

• But

• ‘holes’ can appear in the log when clients obtain tokens and then fail to use them immediately due to crashes or slowdowns.

• A simple solution is to have other clients fill holes aggressively with a reserved ‘junk’ value.

• If the slow client come back again, simply fails
HOW DOES CORFU WORK

• Replication in CORFU

• chaining protocol

• \( f \) failures with just \( f + 1 \) replicas

• data must be visible to reads only after it reaches all replicas
HOW DOES CORFU WORK

• chaining protocol

• write to a replica set of flash pages

• it updates them in a deterministic order, waiting for each flash unit to respond before moving to the next one.
HOW DOES CORFU WORK

• chaining protocol

• read from the replica set, two options.

• If they are unaware of whether the log position was successfully written to or not, go to the last unit of the chain. If the last unit has not yet been updated, it will return an error unwritten.

• if they know already that the log position has been successfully written to, it can go to any replica in the chain for better read performance.
HOW DOES CORFU WORK

• Garbage Collection

• abstraction of an infinitely growing log to applications

• Use trim to mark unused data in SSD

• small amount of proactive data movement across flash units in order to merge consecutive ranges in the projection.
HOW DOES CORFU WORK

- Flash unit - seal

- When a seal command arrives with a new epoch to seal, the flash unit first flushes all ongoing operations and then updates its cur sealed epoch. It then responds to the seal command with cur_highest_offset, the highest address written in the address space it exposes;
APPLICATIONS

• CORFU-Store
  • This protocol ensures linearizability for single-key puts and gets, as well as atomicity for multi-key puts and gets.

• CORFU-SMR
  • Each SMR server simply plays the log forward to receive the next command to execute.
EVALUATION

• a cluster of 32 Intel X25V drives

• Input/output operations per second (IOPS, pronounced eye-ops)
EVALUATION

• end-to-end latency

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

- Throughput

Figure 7: Throughput for random reads and appends.
EVALUATION

- Reconfiguration

Figure 8: Reconfiguration performance on 32-drive cluster. Left: Appending client waits on failed drive for 1 second before reconfiguring, while reading client continues to read from alive replica. Middle: Distribution of sealing and total reconfiguration latency for 32 drives. Right: Scalability of sealing with number of drives.
EVALUATION

• Application - CORFU-Store

Figure 9: Example CORFU Application: CORFU-Store supports atomic multi-gets and multi-puts at cluster scale.
EVALUATION

• Application - CORFU-SMR

Figure 10: Example CORFU Application: CORFU-SMR supports high-speed state machine replication.
SUMMARY

• shared log abstraction over a flash cluster.

• first distributed, shared log design where maximum append throughput is not a function of any single node’s I/O bandwidth.

• low-latency fault-tolerance mechanisms for recovering from flash unit crashes and holes in the log.

• strongly consistent key-value store and a state machine replication library that use CORFU
REFERENCES


• CORFU: A Shared Log Design for Flash Clusters: Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobber, Michael Wei, John D. Davis