Cluster Consensus
When Aeron Met Raft

Martin Thompson - @mjpt777
Do you love or hate ORMs?
Message: 1
Message: 2
Message: 3
Message: 4
Message: 5
Message: 6
What does “Consensus” mean?
consensus
noun \kən-ˈsen(t)-səs\
: general agreement : unanimity

Source: http://www.merriam-webster.com/


con•sen•sus
noun \ kən-ˈsen(t)-səs \\

: general agreement : unanimity

: the judgment arrived at by most of those concerned

Source: http://www.merriam-webster.com/
In Search of an Understandable Consensus Algorithm
(Extended Version)
Diego Ongaro and John Ousterhout
Stanford University

Abstract
Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems. In order to enhance understandability, Raft separates the key elements of consensus, such as leader election, log replication, and safety, and it enforces a stronger degree of coherency to reduce the number of states that must be considered. Results from a user study demonstrate that Raft is easier for students to learn than state space reduction (relative to Paxos, Raft reduces the degree of nondeterminism and the ways servers can be inconsistent with each other). A user study with 43 students at two universities shows that Raft is significantly easier to understand than Paxos: after learning both algorithms, 33 of these students were able to answer questions about Raft better than questions about Paxos.

Raft is similar in many ways to existing consensus algorithms (most notably, Oki and Liskov’s Viewstamped Replication [29, 22]), but it has several novel features:

- **Strong leader**: Raft uses a stronger form of leadership than other consensus algorithms. For example,
Raft Refloated: Do We Have Consensus?

Heidi Howard    Malte Schwarzkopf    Anil Madhavapeddy    Jon Crowcroft

University of Cambridge Computer Laboratory
first.last@cl.cam.ac.uk

ABSTRACT

The Paxos algorithm is famously difficult to reason about and even more so to implement, despite having been synonymous with distributed consensus for over a decade. The recently proposed Raft protocol lays claim to being a new, understandable consensus algorithm, improving on Paxos without making compromises in performance or correctness.

Our study in this paper evaluates the claims about Raft made by its designers. Is it indeed easily understandable, and can the encouraging performance and correctness results presented by Ongaro and Ousterhout be independently confirmed?

In the endeavour to answer this question, we re-implemented Raft in a functional programming language (OCaml) and repeat the experiment. We present the results of our study in this paper.
Raft in a Nutshell
1. **RequestVote RPC**
   Invoked by candidates to gather votes
RPCs

1. **RequestVote RPC**
   Invoked by candidates to gather votes

2. **AppendEntries RPC**
   Invoked by leader to replicate and heartbeat
Safety Guarantees

• Election Safety
• Leader Append-Only
• Log Matching
• Leader Completeness
• State Machine Safety
Version all the things!
Clustering Aeron
Is it Guaranteed Delivery™ ???
What is the “Architect” really looking for?
“Guaranteed Processing™”
Let’s talk about design
Diagram showing clients connected to a service.
NIO Pain!
Do servers crash?
FileChannel channel = null;
try {
    channel = FileChannel.open(directory.toPath());
}
catch (final IOException ignore) {
}

if (null != channel) {
    channel.force(true);
}
FileChannel channel = null;
try {
    channel = FileChannel.open(directory.toPath());
} catch (final IOException ignore) {
}
if (null != channel) {
    channel.force(true);
}
FileChannel channel = null;
try {
    channel = FileChannel.open(directory.toPath());
} catch (final IOException ignore) {
}
if (null != channel) {
    channel.force(true);
}
Files.force(directory.toPath(), true);
Performance
Let’s consider an RPC design approach
Client

Consensus Module

Service

Consensus Module

Service

Consensus Module

Service
Why are CPUs so fast?
Instruction Pipelining

Fetch
Instruction Pipelining

Fetch
Decode
Instruction Pipelining

Fetch  Decode  Execute

Time
Instruction Pipelining

Fetch  Decode  Execute  Retire

Time
Instruction Pipelining
Instruction Pipelining

Fetch | Decode | Execute | Retire
---|---|---|---
Fetch | Decode | Execute | Retire
Fetch | Decode | Execute | Retire
Instruction Pipelining
Consensus Pipeline
Consensus Pipeline

Time

Order | Log | Transmit
Consensus Pipeline

Order → Log → Transmit → Commit
Consensus Pipeline
Consensus Pipeline
NIO Pain!
ByteBuffer byteBuffers = ByteBuffer.allocate(64 * 1024);

byteBuffer.putInt(index, value);
```
ByteBuffer byteBuffer = ByteBuffer.allocate(64 * 1024);
byteBuffer.putBytes(index, srcBytes, srcIndex, length);
```
ByteBuffer byteBuffer = ByteBuffer.allocate(64 * 1024);
byteBuffer.putBytes(srcBytes, srcIndex, length);

https://bugs.openjdk.java.net/browse/JDK-5029431
How can Aeron help?
Message Index => Byte Index
Don’t Copy!

⇒

Encapsulate, Multicast, Spy
Counters

=>

Bounded Consumption
Batching – Amortising Costs

Average overhead per item or operation in batch
Batching – Amortising Costs

- System calls
- Network round trips
- Disk writes
- Expensive computations
Interesting Requirements
Timers
All state must enter the system as a message!
public void foo()
{
    // Decide to schedule a timer
    cluster.scheduleTimer(correlationId, cluster.timeMs() + TimeUnit.SECONDS.toMillis(5));
}

public void onTimerEvent(final long correlationId, final long timestampMs)
{
    // Look up the correlationId associated with the timer
}
public void foo()
{
    // Decide to schedule a timer
    cluster.scheduleTimer(correlationId, cluster.timeMs() + TimeUnit.SECONDS.toMillis(5));
}

public void onTimerEvent(final long correlationId, final long timestampMs)
{
    // Look up the correlationId associated with the timer
}
Timers

```java
public void foo()
{
    // Decide to schedule a timer
    cluster.scheduleTimer(correlationId, cluster.timeMs() + TimeUnit.SECONDS.toMillis(5));
}

public void onTimerEvent(final long correlationId, final long timestampMs)
{
    // Look up the correlationId associated with the timer
}
```
Storage
Log Replay and Snapshots

Distributed File System?
Log Replay and Snapshots

Distributed File System?

Aeron Archive
Recorded Streams
Multiple Services on the same stream
NIO Pain!
1. MappedByteBuffer
   DirectByteBuffer

2. DirectByteBuffer
   MappedByteBuffer
ByteBuffer?

I know
In Closing
Do epic shit, or die trying.
Questions?

https://github.com/real-logic/aeron
Twitter: @mjpt777

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”

- Leslie Lamport