Evolution of Financial Exchange Architectures

Martin Thompson - @mjpt777
10 years ago…
Evolution

Design
Resilience
Performance
Deployment
Design
State Machines

Input $\times$ State $\rightarrow$ State
State Machines

\[ \text{Input} \times \text{State} \rightarrow \text{State} \]

\[ \text{Input} \times \text{State} \rightarrow \text{Output} \]
Replicated State Machines

Ordered Inputs + Deterministic Execution => Same State & Outputs
Distributed Event Log
Rich Domain Models (DDD)
Rich Domain Models (DDD)
Data Structures (CS)
Time & Timers
Fairness
Migration by Asset Class

OTC => Exchange Traded
Resilience
Fault Tolerance
Fault Tolerance

*Primary + Secondary*  
*vs*  
*Consensus*
Leslie Lamport - Paxos

Barbara Liskov - Viewstamp Replication

Ken Birman - Virtual Synchrony
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 Safety Guarantees

• Election Safety
• Leader Append-Only
• Log Matching
• Leader Completeness
• State Machine Safety
Consensus Module

Service

Consensus Module

Boom!

Consensus Module

Service

Boom!

Consensus Module

Service
Importance of Code Quality & Model Fidelity
Robustness
How well does your application handle errors?
Performance
Latency distribution awakening
Systemic & queueing events
Garbage Collectors
Memory Access Patterns & Data Structures
Binary Codecs
Spectre & Meltdown
Greatly increased cost for system calls, page faults, and context switching
Advances in Hardware
New IO APIs
Mechanical Sympathy
Does programming language choice matter?
Deployment
Continuous Delivery
24 * 7 Operations
Flexible Scaling
Wrapping up...
What will the next 10 years hold?
@mjpt777

https://github.com/real-logic/aeron

“The future is already here – it’s just not evenly distributed”

- William Gibson