Federating Consistency for Partition Prone Networks

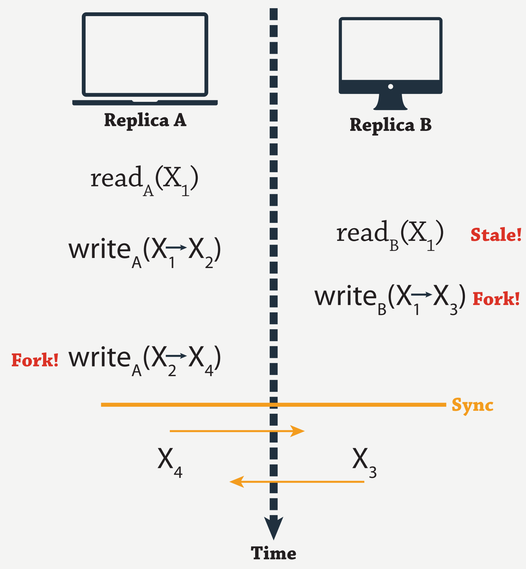
Benjamin Bengfort and Pete Keleher
University of Maryland • Department of Computer Science

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Introduction

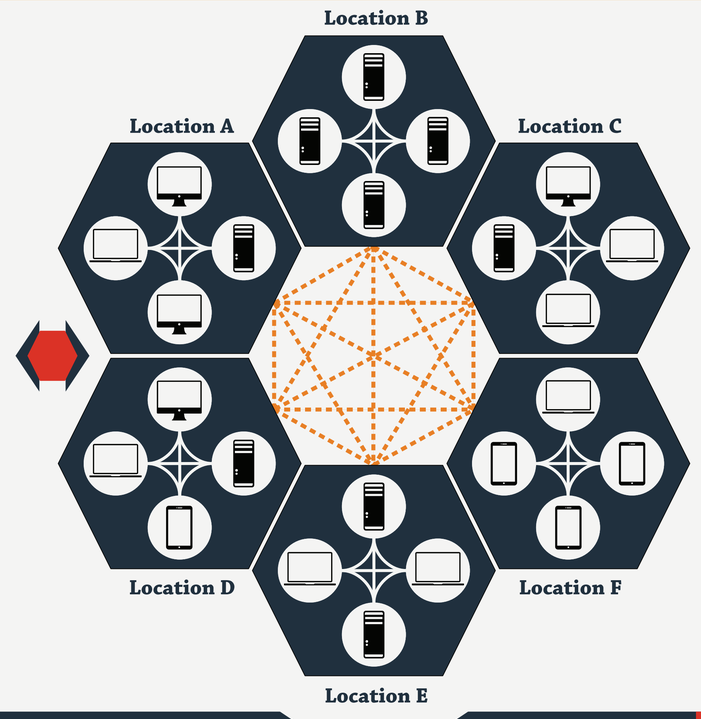
Groups of devices coordinated by consensus can efficiently order events under ideal conditions, but become less effective in dynamic and heterogenous environments. Weakly consistent devices efficiently tolerate faults and dynamic conditions but are slow to converge on a single ordering of system events.

Federated consistency combines the strengths of both approaches into a single protocol. Federated systems use a strongly consistent inner core to maintain a totally ordered sequence of accesses. A cloud of weakly-consistent devices disseminates orderings and enables progress despite varying connectivity.



A fork is two concurrent writes to the same object, which is inconsistent because there are two possible orderings of accesses.

Environment

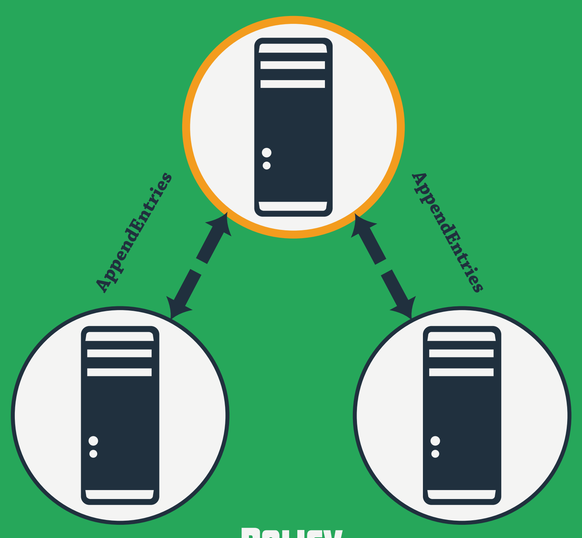


We target dynamic, geo-replicated distributed system models that co-locate replicas with clients rather than traditional cloud-service oriented approaches. Our model considers extremely variable latency, bandwidth, and availability across wide-area links that connect more stable local networks.

We posit a file system as the natural use case of local, client-oriented systems, though the model easily generalizes to any distributed storage system. Clients access objects locally, creating a metadata version history that is replicated to the entire system, though data may only be partially replicated.

Raft
Strong Consistency

Leader-oriented consensus algorithms nominate a dedicated proposer in a fault-tolerant manner to conduct consistent replication. All accesses go through the leader so linearizablility or sequential consistency is guaranteed. On a conflict, the leader simply drops the conflicting version.

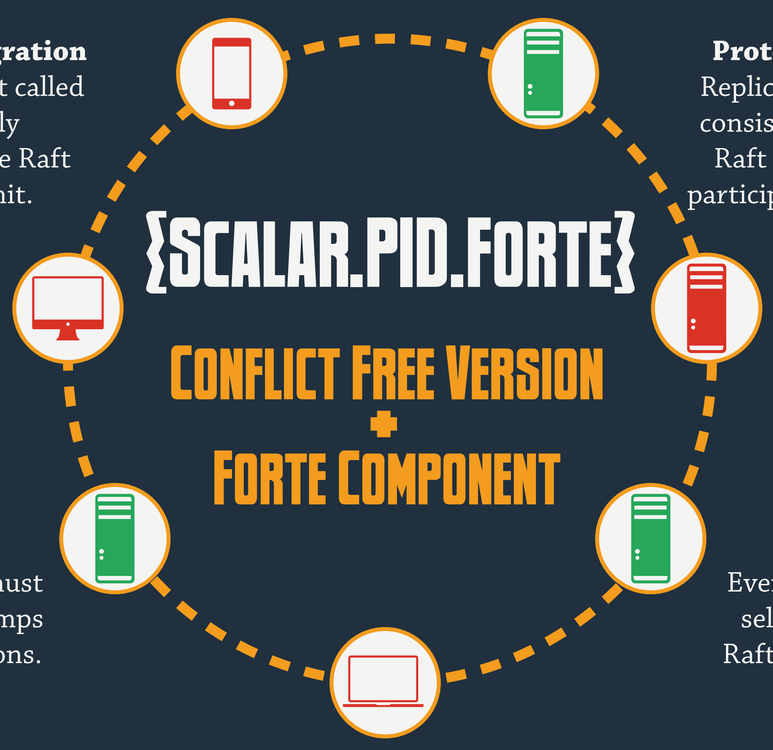


Policy
First Write to Leader Wins

Hybridization of Consistency by Replica Protocol

Consistency Integration
A vector component called the forte is only incremented by the Raft leader on commit.

Protocol Integration
Replicas select their local consistency policy but all Raft replicas must also participate in anti-entropy.



Eventual replicas must propagate forte bumps to all derived versions.

Eventual replicas can select frequency of Raft synchronization.

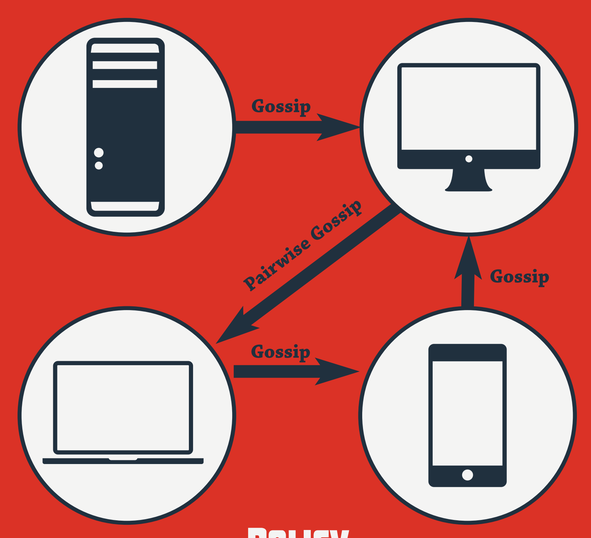
Adjust topology and pairwise selection likelihoods

Adjust timing parameters and adapt to environment

Federated
Tunable Consistency

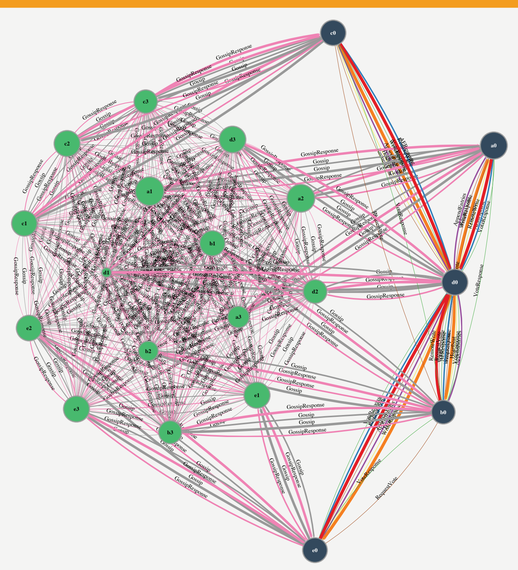
Anti-Entropy
Eventual Consistency

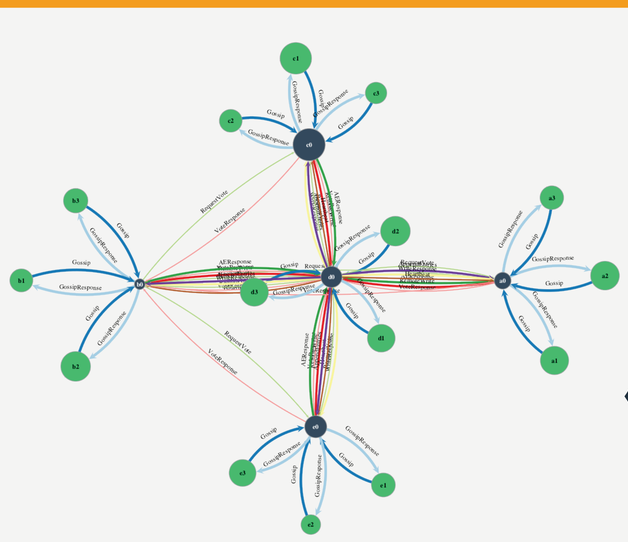
Eventually consistent replication protocols utilize periodic pairwise anti-entropy sessions to synchronize replica states. Because there is no central authority, accesses are highly available and localized. However, anti-entropy tolerates inconsistencies and may even "stomp" writes by not fully replicating them.



Policy
Latest Version Wins

Simulation





Our initial evaluation utilized discrete event simulations with asynchronous processes to investigate the effect of latency, conflict, and network outages on consistency. Every experiment includes multiple simulated runs of 20 processes in varying environments and configurations, accessing multiple objects concurrently.

The primary simulation parameter was variable latency across wide areas, described by the mean and standard deviation of one-way message latency. Other parameters include the likelihood of conflict & outages, timing parameters for consistency protocols & workload, and distributions for anti-entropy selection.

Results