CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

REPLICATION CONTROL

Lecture A

REPLICATION

Server-side Focus

- Concurrency Control = how to coordinate multiple concurrent clients executing operations (or transactions) with a server

Next:

- Replication Control = how to handle operations (or transactions) when there are multiple servers, with data perhaps replicated across servers

Replication: What and Why

- Replication = An object has identical copies, each maintained by a separate server
  - Copies are called “replicas”
- Why replication?
  - Fault-tolerance: With *k* replicas of each object, can tolerate failure of any (*k-1*) servers
  - Load balancing: Spread read/write operations out over the *k* replicas => load lowered by a factor of *k* compared to a single replica
  - Replication => Higher Availability

Availability

- If each server is down a fraction f of the time
  - Server's failure probability
- With no replication, availability of object =

  = Probability that single copy is up

  = $(1 - f)$
- With k replicas, availability of object =

  Probability that at least one replica is up

  = 1 – Probability that all replicas are down

  = $(1 - f^k)$

Nines Availability

- With no replication, availability of object =
  $= (1 - f)$
- With k replicas, availability of object =
  $= (1 - f^k)$

Availability Table

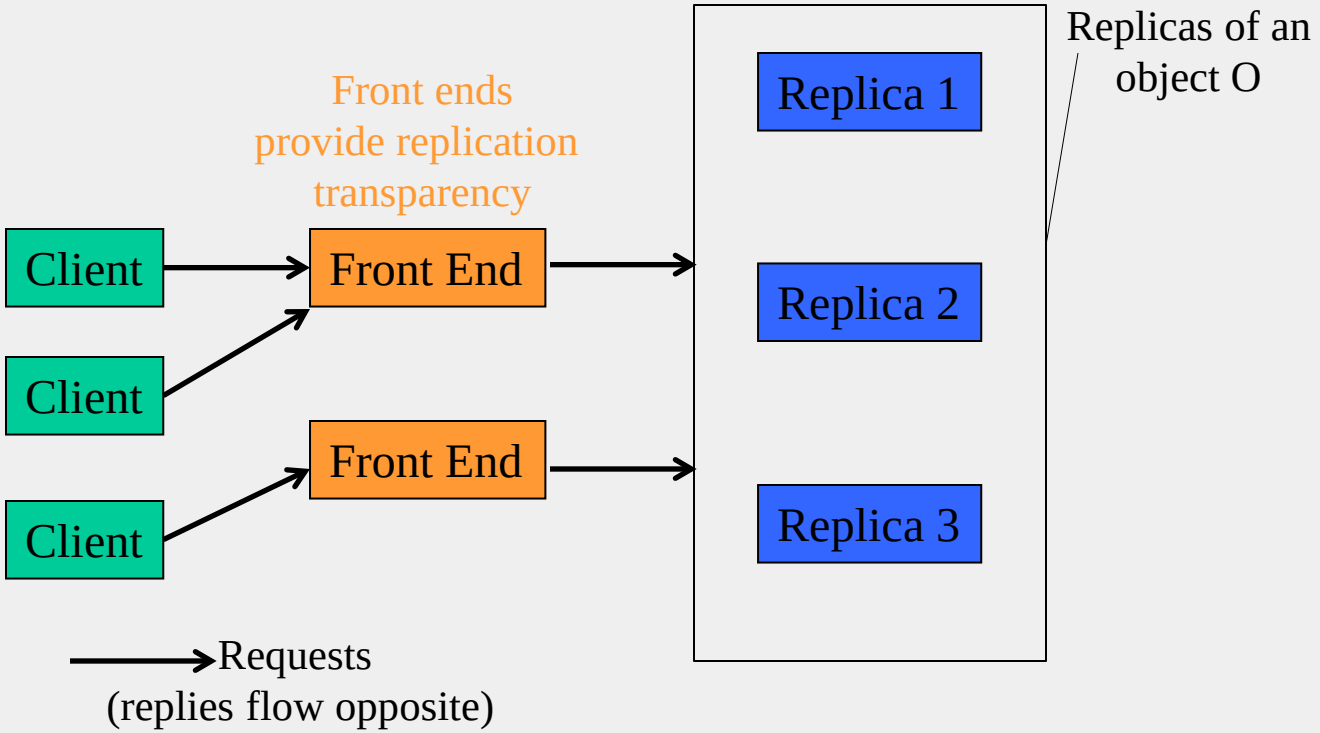
f=failure probability	No replication	k=3 replicas	k=5 replicas
0.1	90%	99.9%	99.999%
0.05	95%	99.9875%	6 Nines
0.01	99%	99.9999%	10 Nines

What's the Catch?

- Challenge is to maintain two properties

1. Replication Transparency
   - A client ought not to be aware of multiple copies of objects existing on the server side
2. Replication Consistency
   - Ensure that clients see single consistent copy of data, in spite of replication
   - For transactions, guarantee ACID

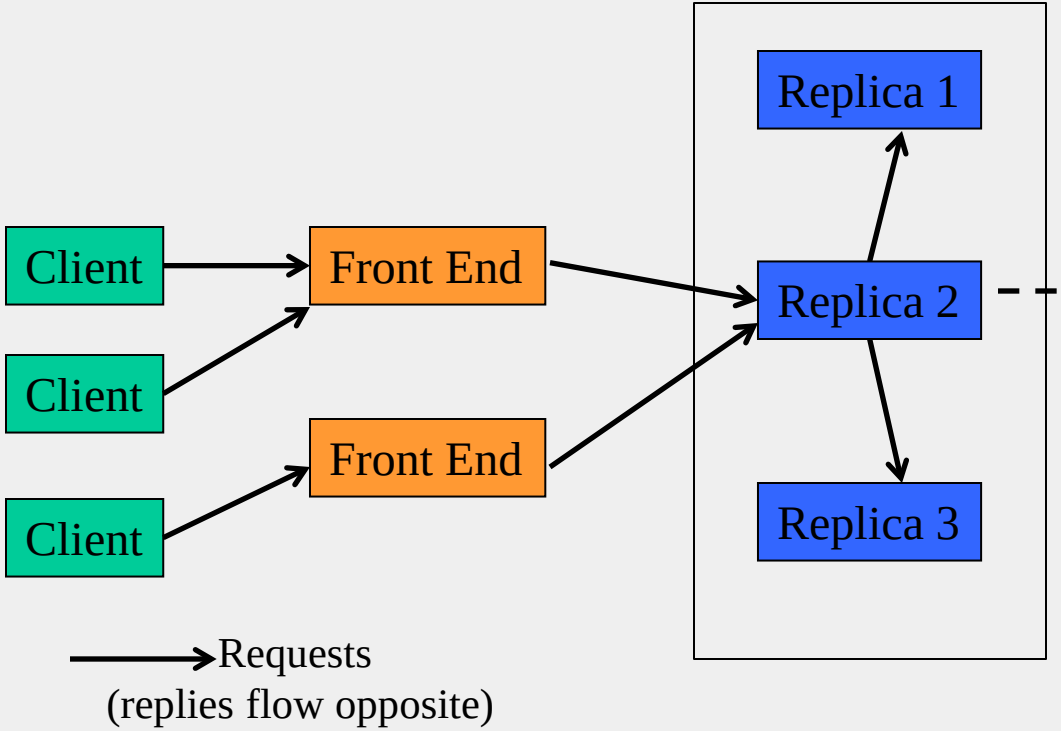
Replication Transparency



Replication Consistency

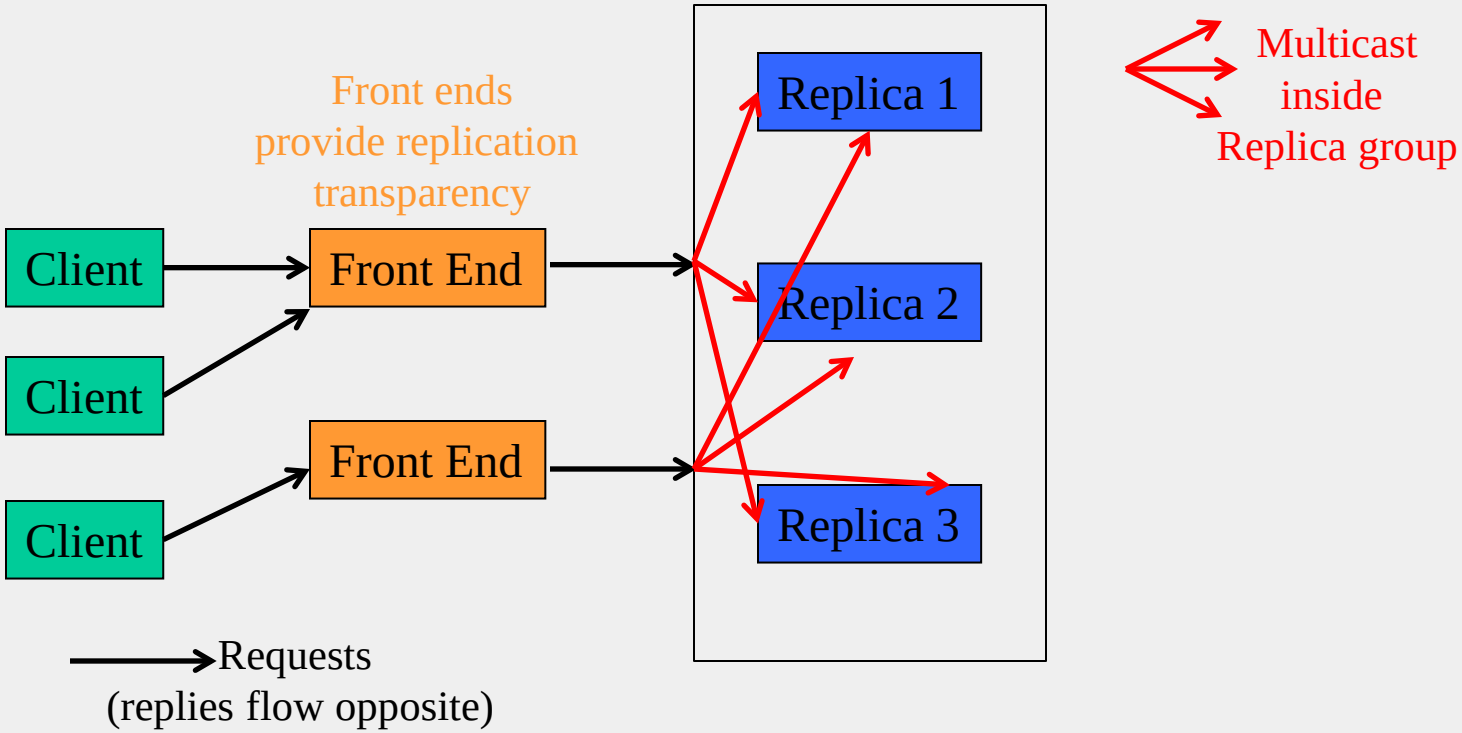
- Two ways to forward updates from front-ends (FEs) to replica group
  - Passive Replication: uses a primary replica (master)
  - Active Replication: treats all replicas identically
- Both approaches use the concept of “Replicated State Machines”
- Each replica’s code runs the same state machine
  - *Multiple copies of the same State Machine begun in the Start state, and receiving the same Inputs in the same order will arrive at the same State having generated the same Outputs.* [Schneider 1990]

Passive Replication



- Master => total ordering of all updates
- On master failure, run election

Active Replication



Active Replication Using Concepts You've Learnt earlier

- Can use any flavor of multicast ordering, depending on application
  - FIFO ordering
  - Causal ordering
  - Total ordering
  - Hybrid ordering
- Total or Hybrid (*-Total) ordering + Replicated State machines approach
  - => all replicas reflect the same sequence of updates to the object

Active Replication Using Concepts You've Learnt earlier (2)

- What about failures?
  - Use virtual synchrony (i.e., view synchrony)
- Virtual synchrony with total ordering for multicasts =>
  - All replicas see all failures/joins/leaves and all multicasts in the same order
  - Could also use causal (or even FIFO) ordering if application can tolerate it

Transactions and Replication

- One-copy serializability
  - *A concurrent execution of transactions in a replicated database is one-copy-serializable if it is equivalent to a serial execution of these transactions over a single logical copy of the database.*
  - (Or) The effect of transactions performed by clients on replicated objects should be the same as if they had been performed one at a time on a single set of objects (i.e., 1 replica per object).
- In a non-replicated system, transactions appear to be performed one at a time in some order.
  - Correctness means serial equivalence of transactions
- When objects are replicated, transaction systems for correctness need
  - Serial equivalence + One-copy serializability

Next

- More on transactions with distributed servers