An Architecture for Survivable Coordination in Large Distributed Systems

Presented By

Mohamed Ali
Assumptions

- **System**
  - Widespread distributed

- **Message**
  - May be delayed
  - May be lost

- **Clients**
  - may be transient, may need to take actions at different times and may not be available at others
  - corrupted by an attacker (Byzantine faults)

- **Servers**
  - May fail and be corrupted by attackers as well (Byzantine faults)
Contributions

- The Fleet system
  - An architecture for persistent data servers
  - Protocols for the clients to use
  - Emulate shared data abstraction
  - Facilitate consensus among clients
Properties of the Fleet system

- **Scalability**
  - scale to hundreds of servers
  - spread across a wide area
  - capable of serving thousands of clients at a time.

- **Survivability**
  - survive the arbitrarily malicious corruption of up to a threshold of its servers
  - any number of clients
Properties of the Fleet system (cont’d)

- Complete operations on shared data objects after interacting with only a small subset (quorum) of servers,
  - no centralized locking
  - no server-to-server interaction.
Properties of the Fleet system (cont’d)

- Timed append-only array (TAOAs):
  - Shared objects
  - Client irreversibly writes a value at the next available slot
  - The servers adds a timestamp with each appended value
  - Timestamp is returned together with the value
  - Timestamps capture the partial ordering
  - The value and timestamp are both impossible to modify or erase, even by the original writer
Properties of the Fleet system (cont’d)

- Using TAOAs, clients emulate a consensus object

- The result is a consensus value in an expected low-degree-polynomial
Example applications

- Public key infrastructures
  - certificate-generating services that create certificates
  - revocation services that enable a client to promptly invalidate her certificate
  - directory services that enable a client to locate the most up-to-date certificate for a name or key
Example applications (cont’d)

- Robust publishing and dissemination
  - Enable a client to publish a document
  - Anyone can retrieve it
  - Nobody, even the author or an adversary, could:
    - eliminate the document
    - deny access to it

- National voting systems
  - Vote only once
Comparison to previous systems

- Every server processes, every client request
- Does not scale well

- Each operation complete at only a fraction of the servers (quorum systems)
- scales well
Architecture overview

Persistent object servers

Server 2
- Object store
- Q-RPC

Server 4
- Object store
- Q-RPC

Server 1
- Object store
- Q-RPC

Server 3
- Object store
- Q-RPC

Server 5
- Object store
- Q-RPC

Q-RPC

Object stubs

Application

Client

Client
Quorum systems

- A quorum system Q is a set of subsets of servers with the property that, for any Q1,Q2 in Q,
  - $|Q_1 \cap Q_2| \geq 2b+1$ enables clients to infer correct replies from the contacted quorum.

- for any set B of servers where $|B|=b$, there is some Q’ in Q such that $B \cap Q’=\_\_\_\_$ ensures that a client can always contact a full quorum.

b is the maximum number of faulty servers.
Data Objects

- **Shared storage**
  - Shared variable abstractions support read and write operations
  - We can guarantee atomic updates

- **At-most-one mutual exclusion**
  - Supports a contend operation, which succeeds in at most one among all contending clients.
  - Voter identifier can be used to cast a vote only once

- **Consensus objects**
  - A consensus object is a shared object to which a client can propose a value
  - Receive a single value in return.
  - The consensus object returns the same value to each client
  - The returned value is one proposed by some client
Scalability

- Two factor for scalability
  - efficient protocol design (scale with number of clients)
  - the use of quorum systems (scale with number of servers)

Note: Consensus object does not scale quite well because the amount of work a client must do is a function of the number of clients.
Timed append-only array (TAOAs)  
The Read operation

- Client executes a Q-RPC to obtain the value/timestamp pair in $T_{u[i]}$ from each server $u$ in some quorum $Q$.
- The client gets the answered agreed upon by more than $b$ servers.
- Each client records the latest index it read from a server. This index is used to inform other servers during the append operation of how large the array is on other servers.
- The client stores the “signed” replies of the $b+1$ server. These signed replies are used to update other servers in the append operation.
Timed append-only array (TAOAs)
The Append operation (Step 1/3)

- The client contacts a quorum of servers Q
- The client sends the latest index “i” it read in the array to every server “u” in the quorum.
  - If entry “i” in server “u” is empty, the server acquires the \( b+1 \) signed value stored in the client. (updating the servers themselves)

- If “i” is less than the upper bound of the array on u, the client reads and store all the signed values of the array that he misses. (to update other servers in the future)
Timed append-only array (TAOAs)
The Append operation (Step 2/3)

- Client includes the responses (updating servers) to the first Q-RPC, along with the value “v” that it intends to append.
- Each server u prepares to update its array
- Each server echoes the “v” digitally signed to the client
Timed append-only array (TAOAs)
The Append operation (Step 3/3)

- Client now forwards the digitally signed replies back to the servers via a third Q-RPC

- Upon receiving this request, each server updates its array and acknowledges.
A consensus protocol

- Each client executes a sequence of logical rounds until it reaches a decision

- There is one TAOA per client

- Client communicate values to the system by appending them to its array

- Client attaches its round number to each value it appends
A consensus protocol (cont’d)

- In each round, a client starts by appending its currently preferred value.

- Then, client reads the latest values appended by all of the other processes.

- Among these values, the ones with the highest round number are called the leaders' values.
A consensus protocol (cont’d)

- If the leaders agree, it tries to adopt their value as its own preferred value and move to the next round (or decide);

- If the leaders disagree and it is a leader itself, it attempts to flip a (multivalued) coin, adopt the value of the coin as its preferred value, and then move to the next round.

- The coin returns one value of those who are initially proposed by clients.
A consensus protocol (cont’d)

- Decision is possible for a leader when all the processes who disagree with its preferred value are at least two rounds behind.

- Intuitively, this protocol converges at the latest when a round starts with all the leaders preferring the same value. (matter of probability)
Conclusions

- An architecture for the construction of survivable and scalable data repositories is presented.

- The TAOAs can be used to support communication among clients when Byzantine failures are a concern.

- Shared objects allow individual clients to obtain a consensus value.