Chain Replication

Karthik Nagabhushan
PES1UG19CS208

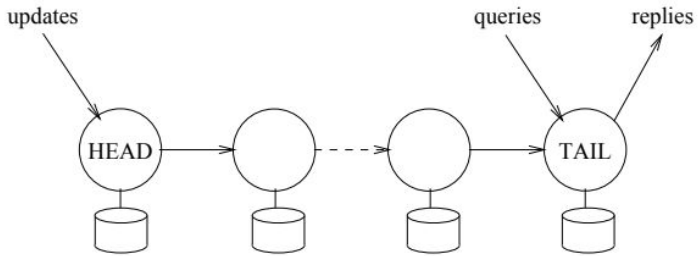
Likhith
PES1UG19CS242

Shardul Pande
01FB16ECS353

PROBLEM STATEMENT

Implementation of Chain Replication

- Chain Replication is an approach of **coordinating clusters** of fail-stop storage servers.
- Data is replicated across multiple nodes, which are **arranged as a chain**.



PROBLEM STATEMENT

Write Operations :

- All the write operations from clients go to the **head** of the chain.
- When a node receives a write operation it executes the write and **passes it down the next node** in the chain until it reaches the tail.
- A write is marked as **committed** when it **reaches the tail**.

Read Operations :

- The **tail node** handles all the read operations.
- Read will only return the committed values.

Fault tolerance:

- System should be able to work properly, even if upto N-1 nodes out of N nodes of the chain fails.

Context

- Chain Replication supports high throughput and availability **without sacrificing strong consistency**.(The write operations are performed in the same sequential order and the updates are necessarily reflected in the results returned by subsequent queries).
- There is **no performance bottleneck** as compared to master-worker architectures, as each node has to pass the updates only to its successor in the chain.
- It supports indefinite **scaling**.
- Achieves **high availability** => At Most N-1 Servers can fail concurrently.

Highlights and contributions

This implementation has **3 main components**:

- Master:
  - Lookafter the whole system.
  - **Detect failed nodes**
  - **Reconfigure chain** when nodes fail.
  - Informs Clients
- Node:
  - Systems part of chain itself.
  - Process and forward requests within the chain.
  - **Head**: Handle update requests. **Tail**: Handle read requests.
- Client:
  - Provide user the **interface** to interact with the chain (as storage service).
  - **Issues requests** for query and update operations

Libraries used:

- RPyC
- Threading
- Pickle

Likhith: Failure-handling (master.py), node.py

Karthik: Nodes & clients registration (master.py)

Shardul: User-interface (client.py)

Demo

- Starting programs and creating a chain network of few nodes.
- Adding key-value pairs.
- Reading key-value pairs.
- Handling:
  - In-between node failure.
  - Head node failure.
  - Tail node failure.
- Discussing different scenarios in which a node can fail and how our implementation handles it.

Thank you