Implementation of RingBFT: Resilient Consensus over Sharded Ring Topology

Authors: Sajjad Rahnama, Suyash Gupta, Rohan Sogani, Dhruv Krishnan, Mohammad Sadoghi
Presenter: Jiangnan Chen, Fuming Fu, Haochen Yang, Weijia Wang, Xiaoxi Yu, Zaoyi Zheng
Date: Nov 29, 2021
Roadmap

1. Introduction
   PBFT - GeoBFT - RingBFT

2. Message Redesign in RingBFT

3. Message Execution in RingBFT

4. Global Sharing in RingBFT
Background

PBFT
- First protocol that survives Byzantine faults in asynchronous network
- Time complexity: exponential to polynomial
- Optimizations to reduce the cost of communication
- Avoid sending most large replies
- Reduce the number of message delays
- Improve the performance of read-only operations not modifying the service state

GeoBFT
- Introduced parallelization of consensus at the local level
- Minimized communication between clusters

RingBFT
- A novel meta-BFT protocol
- Sharded-replicated permissioned blockchain system adhering to ring order
- Linear communication between shards
- A scalable consensus for cross-shard transactions
Introduction

RingBFT

A solution to federated database application

Single-shard Consensus

Cross-shard Consensus

(1) Ring order

(2) Data lock

(3) Forward process

GeoBFT ➔ RingBFT

C1

Shard 1

Shard 2

Shard 3

C2

C3

Shard 1

Shard 2

Shard 3

why?
Message Redesign in RingBFT

class RingBFTForwardMessage : public Message
{
    public:
        void copy_from_buf(char *buf);
        void copy_to_buf(char *buf);
        void copy_from_txn(TxnManager *txn);
        void copy_to_txn(TxnManager *txn);
        uint64_t get_size();
        void init();
        void release();

        void sign(uint64_t dest_node_id);
        bool validate();
        string toString();

        uint64_t view;
        Array<uint64_t> index;
        uint64_t hasSize;
        string hash;

        Array<uint64_t> signSize;
        Array<uint64_t> signOwner;
        vector<string> signatures;

        deque<uint64_t> executeOrder; // indexes of shards in execution order
        deque<uint64_t> forwardOrder; // indexes of shards in forwarding order

        // node_id of next shard to forward
        uint64_t get_next_node_id(deque<uint64_t> ringOrder, deque<uint64_t> executeOrder);
};
Message Redesign in RingBFT

class RingBFTExecuteMessage : public Message {
    public:
    void copy_from_buf(char *buf);
    void copy_to_buf(char *buf);
    void copy_from_txn(TxnManager *txn);
    void copy_to_txn(TxnManager *txn);
    uint64_t get_size();
    void init();
    void release();
    void sign(uint64_t dest_node_id);
    bool validate();
    string toString();
    uint64_t view;
    Array<uint64_t> index;
    uint64_t hashSize;
    string hash;
    Array<uint64_t> signSize;
    Array<uint64_t> signOwner;
    vector<string> signatures;
    RingBFTExecuteMessage(deque<uint64_t> eo) {
        executeOrder = eo;
    }
    // get next node id in execution order
    uint64_t get_next_node_id(deque<uint64_t> executeOrder) {
        uint64_t next_execution_node_id = executeOrder.front();
        executeOrder.pop_front();
        return next_execution_node_id;
    }
Message Execution in RingBFT

Design dependent transaction:
Modified SpinLockMap to enable every node of shard to lock and unlock data fragments.

Use the array of SpinLockMap to represent the datafragment needed for every execution.
Message Execution in RingBFT

**WorkThread:**
1. Get message from workQueue
2. Process the message with regard to its type
3. Process first round execution and last round execution separately
Message Execution in RingBFT

First Round Execution:
1. Lock the data fragment
2. Check if the last one

Last Round Execution:
1. Perform the transaction
2. Unlock corresponding data fragment
3. Reply to client
Message Execution in RingBFT

Execution:
Message cannot be displayed
Global Sharing in RingBFT

Difference:
GeoBFT: Primary -> Replica of other shard, send f+1 messages
RingBFT: Both replica and primary send one message to next shard which has the same id
Global Sharing in RingBFT

Difference:
GeoBFT: Primary -> Replica of other shard, send f+1 messages
RingBFT: Both replica and primary send one message to next shard which has the same id

For cross-shard, number of nodes should be more than that in a cluster
Create message and coerce it
Register in transaction manager
Traverse all nodes:
1. Not in the same cluster
2. Have the same id
3. In adjacent shards (Ring order)
Add to message queue then clear the destination vector
The End