Lab 2: Replication

CƠ CHẾ REPLICATION VÀ BULLY ELECTION

1. Cơ chế Replication (Sao chép dữ liệu)

1.1. Mô tả cơ chế

Replication giúp **đồng bộ dữ liệu** giữa leader và các node follower, đảm bảo dữ liệu **không bị mất** khi leader gặp sự cố.

Cơ chế hoạt động như sau:

1. **Leader xử lý yêu cầu từ client (ghi/xóa dữ liệu).**
2. **Leader gửi lệnh replication đến các node follower.**
3. **Follower nhận dữ liệu và cập nhật database của mình.**

1.2. Cấu trúc dữ liệu

Hệ thống sử dụng **RPC** để truyền tải thông tin replication. Các cấu trúc dữ liệu liên quan đến replication bao gồm:

Request gửi từ Leader đến Follower

```
type ReplicateDataReq struct {
    Action     string // "SET" hoặc "DELETE"
    BucketName string
    Key        int
    Value      string
}
```

Response từ Follower gửi lại Leader

```
type ReplicateDataRes struct {
    IsSuccess bool
```

```
}
```

1.3. Quy trình Replication

(1) Khi Leader nhận request từ Client

Khi client gửi yêu cầu `SetKey`, leader sẽ:

- Ghi dữ liệu vào database.
- Gửi lệnh replication đến tất cả follower.

Code trong `node.go`

```
func (server *LeaderRPC) SetKey(args *SetKeyArgs, reply *Response) error {
    database := server.node.database
    database.mutex.Lock()
    defer database.mutex.Unlock()

    // Lưu dữ liệu vào database của leader
    database.db.Set(args.BucketName, args.Key, []byte(args.Value))
    reply.Message = "OK"

    // Gửi lệnh replication đến các node follower
    server.doReplicate(ReplicateDataReq{
        BucketName: args.BucketName,
        Key:        args.Key,
        Value:      args.Value,
        Action:     "SET",
    })
    return nil
}
```

---

(2) Leader gửi replication đến các node follower

Code trong `node.go`

```
func (leaderRPC *LeaderRPC) doReplicate(args ReplicateDataReq) {
    peerIds := leaderRPC.node.peerIds
    for _, peerId := range peerIds {
        var replicateRes ReplicateDataRes
        err := leaderRPC.node.callToFollower(
            peerId,
            "InternalRPC.ReplicateAction",
            args,
            &replicateRes,
        )
        if err != nil {
            log.Println(err)
        }
    }
}
```

Leader gửi lệnh `ReplicateAction` đến từng follower qua RPC.

(3) Follower nhận và cập nhật dữ liệu

Code trong `node.go`

```
func (nodeRPC *InternalRPC) ReplicateAction(args ReplicateDataReq, reply *R
eplicateDataRes) error {
    log.Printf("[Node %d] Nhận replication: %s - key: %d", nodeRPC.node.id, ar
gs.Action, args.Key)
    database := nodeRPC.node.database
    database.mutex.Lock()
    defer database.mutex.Unlock()

    if args.Action == "DELETE" {
        _, err := database.db.Del(args.BucketName, args.Key)
        if err != nil {
            return err
        }
```

```
    } else if args.Action == "SET" {
        err := database.db.Set(args.BucketName, args.Key, []byte(args.Value))
        if err != nil {
            return err
        }
    }
    reply.IsSuccess = true
    return nil
}
```

Follower cập nhật dữ liệu vào database của mình.

---

2. Cơ chế Bully Election

2.1. Mô tả cơ chế

Bully Election đảm bảo rằng **một leader duy nhất được bầu chọn** khi hệ thống khởi động hoặc khi leader hiện tại gặp lỗi.

💡 **Quy trình election:**

1. Một node **phát hiện leader chết**.
2. Nó gửi yêu cầu **"Election"** đến các node có ID cao hơn.
3. Nếu không có node nào phản hồi, **node đó trở thành leader**.
4. Nếu có node cao hơn phản hồi, **nó tiếp tục gửi "Election"** lên trên.

2.2. Quy trình Election

(1) Một node phát hiện leader đã chết

Code trong `node.go`

```
func (node *Node) startHeartbeatRoutine() {
    go func() {
        for {
            time.Sleep(3 * time.Second)
```

```
            node.electionMutex.Lock()
            leader := node.leaderId
            node.electionMutex.Unlock()

            if leader == -1 {
                node.startElection()
                continue
            }
            if leader == node.id {
                continue
            }
            var heartbeatRes HeartbeatRes

            err := node.callToLeader(
                leader,
                "LeaderRPC.Heartbeat",
                HeartbeatReq{node.id},
                &heartbeatRes,
            )
            if err != nil || !heartbeatRes.Alive {
                log.Printf("[Node %d] Không thấy leader, bắt đầu bầu cử!", node.id)
                node.startElection()
            }
        }
    }()
}
```

Cứ mỗi 3 giây, một node kiểm tra xem leader có còn sống không.

Nếu leader chết, node này sẽ kích hoạt election.

---

(2) Gửi yêu cầu bầu cử

Code trong `node.go`

```
func (node *Node) startElection() {
    node.electionMutex.Lock()
    if node.isElectionRunning {
        node.electionMutex.Unlock()
        return
    }
    node.isElectionRunning = true
    node.electionMutex.Unlock()

    higherIds := []int{}
    for _, peerId := range node.peerIds {
        if peerId > node.id {
            higherIds = append(higherIds, peerId)
        }
    }
    var isFoundHigherActiveNode = false
    for _, nextId := range higherIds {
        log.Printf("[Node %d] Gửi yêu cầu Election đến node %d", node.id, nextI
d)
        var electionReply ElectionRes
        err := node.callToFollower(
            nextId,
            "InternalRPC.Election",
            ElectionReq{node.id},
            &electionReply,
        )
        if err == nil && electionReply.Ack {
            isFoundHigherActiveNode = true
            break
        }
    }

    if !isFoundHigherActiveNode {
        node.isElectionRunning = false
        node.becomeLeader()
```

```
    }
}
```

Node gửi yêu cầu election lên các node có ID cao hơn.

Nếu **không có node nào phản hồi**, **node này trở thành leader**.

---

(3) Node cao nhất trở thành Leader

Code trong `node.go`

```
func (node *Node) becomeLeader() {
    node.electionMutex.Lock()
    defer node.electionMutex.Unlock()
    log.Printf("[Node %d] Tôi là leader mới!", node.id)

    node.leaderId = node.id
    node.startLeaderServer()

    for _, nextId := range node.peerIds {
        var notifyLeaderRes NotifyLeaderRes
        node.callToFollower(
            nextId,
            "InternalRPC.NotifyLeader",
            NotifyLeaderReq{node.id},
            &notifyLeaderRes,
        )
    }
}
```

Node tự đặt mình làm leader và thông báo cho các node khác.

---

3. Chạy thử

3.1. Server:

Chạy theo thứ tự id 1, 3, 2

```
go run serverMain.go node.go -id=1 -peers=2,3
2025/03/26 15:56:17 [Node 1] Internal server listening on :8001
2025/03/26 15:56:22 [Node 1] Start Heartbeat Rountine
2025/03/26 15:56:25 [Node 1] Call Election to node 2
2025/03/26 15:56:25 [Node 1] Calling method InternalRPC.Election
2025/03/26 15:56:25 [Node 1] Could not connect to peer 2 at 8002: dial tcp 127.0
2025/03/26 15:56:25 Has error while trying to get key dial tcp 127.0.0.1:8002: cor
2025/03/26 15:56:25 [Node 1] Call Election to node 3
2025/03/26 15:56:25 [Node 1] Calling method InternalRPC.Election
2025/03/26 15:56:25 [Node 1] Could not connect to peer 3 at 8003: dial tcp 127.0
2025/03/26 15:56:25 Has error while trying to get key dial tcp 127.0.0.1:8003: cor
2025/03/26 15:56:25 [Node 1] I am now the leader.
2025/03/26 15:56:25 [Node 1] Notify to node 2 that leader is 1
2025/03/26 15:56:25 [Node 1] Calling method LeaderRPC.StepDown
2025/03/26 15:56:25 [Node 1] Could not connect to peer 2 at 8000: dial tcp 127.0
2025/03/26 15:56:25 dial tcp 127.0.0.1:8000: connect: connection refused
2025/03/26 15:56:25 [Node 1] Notify to node 3 that leader is 1
2025/03/26 15:56:25 [Node 1] Calling method LeaderRPC.StepDown
2025/03/26 15:56:25 [Node 1] Could not connect to peer 3 at 8000: dial tcp 127.0
2025/03/26 15:56:25 dial tcp 127.0.0.1:8000: connect: connection refused
2025/03/26 15:56:25 [Node 1] Leader server listening on 8000
2025/03/26 15:56:25 [Node 1] Notify to node 2 that leader is 1
2025/03/26 15:56:25 [Node 1] Calling method InternalRPC.NotifyLeader
2025/03/26 15:56:25 [Node 1] Could not connect to peer 2 at 8002: dial tcp 127.0
2025/03/26 15:56:25 dial tcp 127.0.0.1:8002: connect: connection refused
2025/03/26 15:56:25 [Node 1] Notify to node 3 that leader is 1
2025/03/26 15:56:25 [Node 1] Calling method InternalRPC.NotifyLeader
2025/03/26 15:56:25 [Node 1] Could not connect to peer 3 at 8003: dial tcp 127.0
2025/03/26 15:56:25 dial tcp 127.0.0.1:8003: connect: connection refused
2025/03/26 15:56:52 [Node 1] Leader listener closed: accept tcp [::]:8000: use o
2025/03/26 15:56:52 [Node 1] Receive NotifyLeader 3
2025/03/26 15:56:52 [Node 1] Acknowledge that leader is now 3
2025/03/26 15:56:52 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:56:55 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:56:58 [Node 1] Calling method LeaderRPC.Heartbeat
```

```
2025/03/26 15:57:01 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:04 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:07 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:10 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:13 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:15 [Node 1] Receive NotifyLeader 3
2025/03/26 15:57:15 [Node 1] Acknowledge that leader is now 3
2025/03/26 15:57:16 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:19 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:22 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:25 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:28 [Node 1] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:31 [Node 1] Calling method LeaderRPC.Heartbeat
...
```

```
go run serverMain.go node.go -id=3 -peers=1,2
2025/03/26 15:56:44 [Node 3] Internal server listening on :8003
2025/03/26 15:56:49 [Node 3] Start Heartbeat Rountine
2025/03/26 15:56:52 [Node 3] I am now the leader.
2025/03/26 15:56:52 [Node 3] Notify to node 1 that leader is 3
2025/03/26 15:56:52 [Node 3] Calling method LeaderRPC.StepDown
2025/03/26 15:56:52 [Node 3] Notify to node 2 that leader is 3
2025/03/26 15:56:52 [Node 3] Calling method LeaderRPC.StepDown
2025/03/26 15:56:52 [Node 3] Could not connect to peer 2 at 8000: dial tcp 127.
2025/03/26 15:56:52 dial tcp 127.0.0.1:8000: connect: connection refused
2025/03/26 15:56:52 [Node 3] Leader server listening on 8000
2025/03/26 15:56:52 [Node 3] Notify to node 1 that leader is 3
2025/03/26 15:56:52 [Node 3] Calling method InternalRPC.NotifyLeader
2025/03/26 15:56:52 [Node 3] Notify to node 2 that leader is 3
2025/03/26 15:56:52 [Node 3] Calling method InternalRPC.NotifyLeader
2025/03/26 15:56:52 [Node 3] Could not connect to peer 2 at 8002: dial tcp 127.
2025/03/26 15:56:52 dial tcp 127.0.0.1:8002: connect: connection refused
2025/03/26 15:57:15 [Node 3] I am now the leader.
2025/03/26 15:57:15 [Node 3] Notify to node 1 that leader is 3
```

```
2025/03/26 15:57:15 [Node 3] Calling method InternalRPC.NotifyLeader
2025/03/26 15:57:15 [Node 3] Notify to node 2 that leader is 3
2025/03/26 15:57:15 [Node 3] Calling method InternalRPC.NotifyLeader
```

```
go run serverMain.go node.go -id=2 -peers=1,3
2025/03/26 15:57:07 [Node 2] Internal server listening on :8002
2025/03/26 15:57:12 [Node 2] Start Heartbeat Rountine
2025/03/26 15:57:15 [Node 2] Call Election to node 3
2025/03/26 15:57:15 [Node 2] Calling method InternalRPC.Election
2025/03/26 15:57:15 [Node 2] Receive NotifyLeader 3
2025/03/26 15:57:15 [Node 2] Acknowledge that leader is now 3
2025/03/26 15:57:18 [Node 2] Calling method LeaderRPC.Heartbeat
2025/03/26 15:57:21 [Node 2] Calling method LeaderRPC.Heartbeat
...
```