hulu

Yet Another Raft Lib

@Tao @Xiaoke @Jing @Qianqian

Contents

- Raft Introduction
- Project Overview
- Architecture
- Raft Lib
- Thrift Interface
- KV Store
- Demo

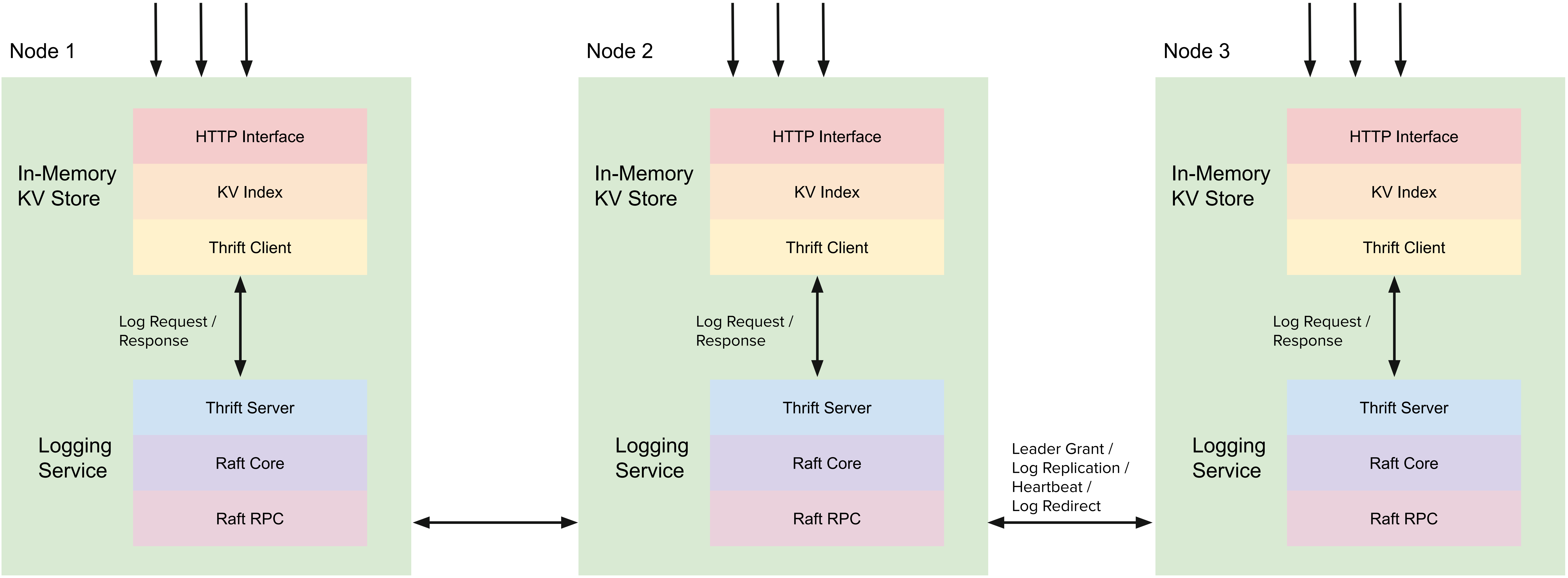
Raft Introduction

- **An understandable & efficient consensus algorithm**
  - Can be viewed as a distributed log store which guarantees
    - A committed log keep consistent with all nodes
    - System is available as long as a majority of nodes are alive
  - Main building block
    - Leader Election
    - Log Replication
    - Dynamic Configuration
    - Log Compaction

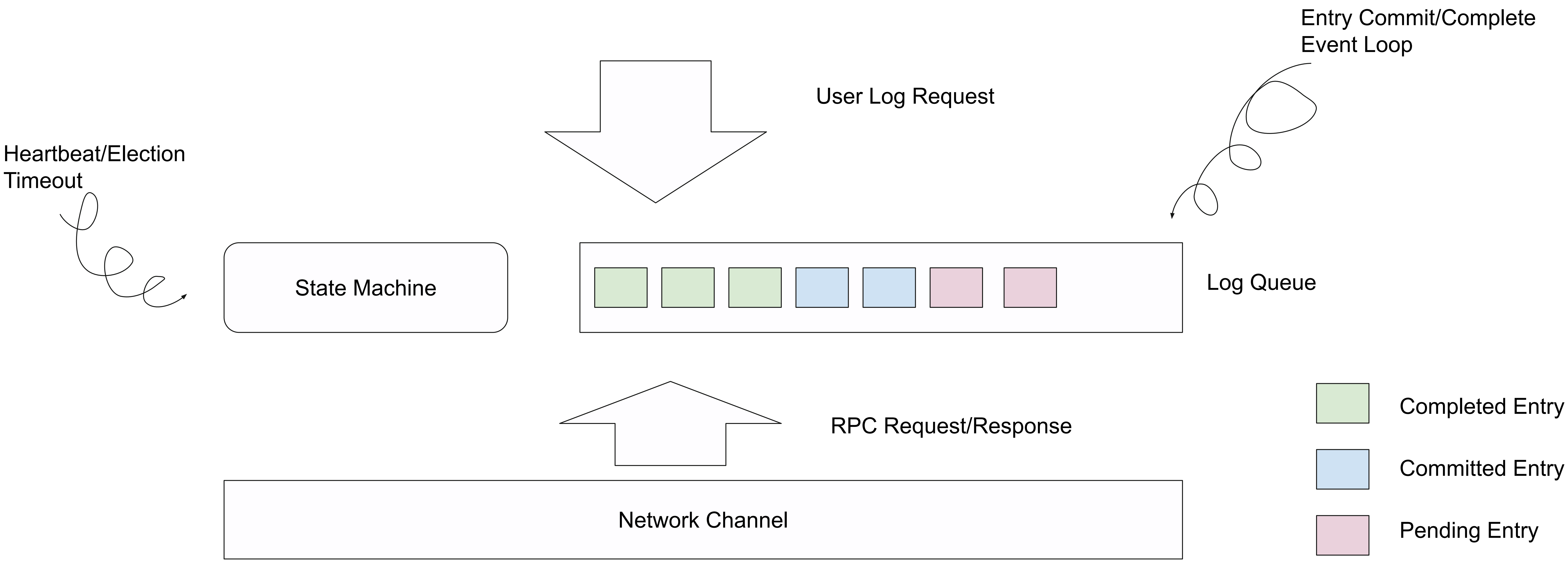
Project Overview

1. A raft lib with core logging functionality written in scala
2. A cross-language logging service written upon thrift
3. A high available KV store written in python

Architecture



Raft Lib - Overview



Entry Commit/Complete
Event Loop

User Log Request

Heartbeat/Election
Timeout

State Machine

Log Queue

RPC Request/Response

Network Channel

🟩 Completed Entry

🟦 Committed Entry

🟪 Pending Entry

Raft Lib - Interface

```
trait Raft {
    // `log` method logs a entry asynchronously, but no committence
guarantee
    def log(data: String)
    // listener is used for post process a committed entry
    def registerListener(func: String => Unit)
}
```

Raft Lib - Thread Model

- **RPC threads**
    - Handle other nodes' request / transfer state / append log
- **Timer thread**
    - For leader, send heartbeat to others periodically
    - For follower, start leader election after a time out
- **Log Event Loop**
    - Try commit(leader-only) / complete log entries

Raft Lib - RPC

- **Vote Request (issued by candidate)**
- **Log Append Request (issued by leader)**
  - Send new entries to follower
  - Also used for heartbeat as a side effect
- **Log Redirect Request (issued by follower)**
  - Redirect user log requests to leader

Thrift Interface

```
service RaftService {
    void log(1: required string data)
    // work around for single-direction rpc, each entry can only be
retrieved once
    list<string> getAvailableLogs()
}
```

KV Store

- **Use python `dict` for indexing**
  - For SET, write a "key=value" log to Logging Service
  - For GET, get available logs and update dict, then serve request

Demo

- Leader Election
- Log Replication
- Node Crash
- Node Recover

hulu

THANK YOU