The TinyKV Course

This is a series of projects on a key-value storage system built with the Raft consensus algorithm. These projects are inspired by the famous MIT 6.824 course, but aim to be closer to industry implementations. The whole course is pruned from TiKV and re-written in Go. After completing this course, you will have the knowledge to implement a horizontal scalable, high available, key-value storage service with distributed transaction support and a better understanding of TiKV implementation.

The whole project is a skeleton code for a kv server and a scheduler server at initial, and you need to finish the core logic step by step:

- Lab 1
    - Build a standalone key-value server
    - Build a high available key-value server with Raft
- Lab 2: Support distributed transaction on top of Project3

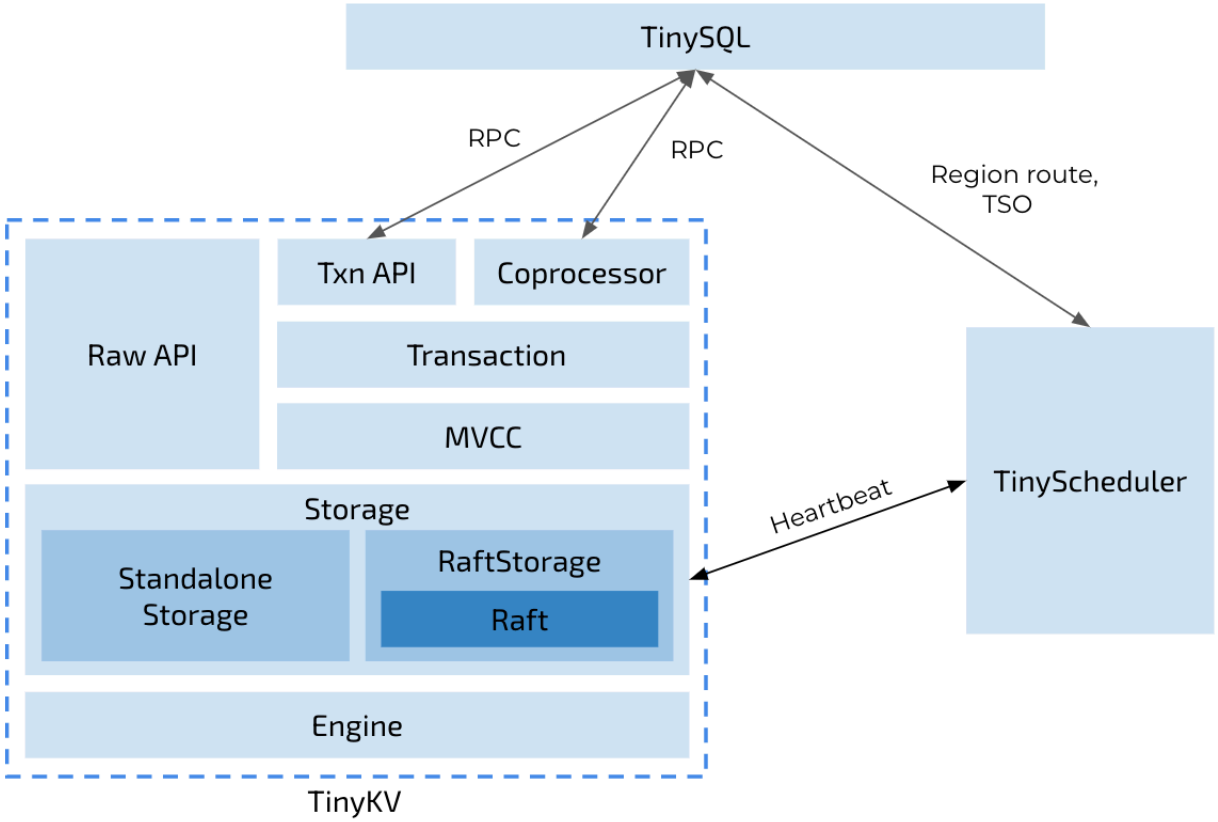
Course

Here is a reading list for the knowledge of distributed storage system. Though not all of them are highly related with this course, it can help you construct the knowledge system in this field.

Also, you'd better read the overview design of TiKV and PD to get a general impression on what you will build:

- TiKV
    - https://pingcap.com/blog-cn/tidb-internal-1/ (Chinese Version)
    - https://pingcap.com/blog/2017-07-11-tidbinternal1/ (English Version)
- PD
    - https://pingcap.com/blog-cn/tidb-internal-3/ (Chinese Version)
    - https://pingcap.com/blog/2017-07-20-tidbinternal3/ (English Version)

Overview of the code



Same as the architecture of TiDB + TiKV + PD that separates the storage and computation, TinyKV only focuses on the storage layer of a distributed database system. If you are also interested in SQL layer, see TinySQL. Besides that, there is a component called TinyScheduler as a center control of the whole TinyKV cluster, which collects information from the heartbeats of TinyKV. After that, the TinyScheduler can generate some scheduling tasks and distribute them to the TinyKV instances. All of them are communicated by RPC.

The whole project is organized into the following directories:

- `kv`: implementation of the TinyKV key/value store.
- `proto`: all communication between nodes and processes uses Protocol Buffers over gRPC. This package contains the protocol definitions used by TinyKV, and generated Go code for using them.
- `raft`: implementation of the Raft distributed consensus algorithm, used in TinyKV.
- `scheduler`: implementation of the TinyScheduler which is responsible for managing TinyKV nodes and for generating timestamps.

- `log`: log utility to output log base on level.

Course material

Please follow the course material to learn the background knowledge and finish code step by step.

- lab1 - log and storage engine
- lab2 - transaction engine the participant