计算机系统能力培养任务书

TinyKV 分布式键值存储系统

院　　系	计算机科学与技术
专业班级	计算机 XXXX
姓　　名	XXX
学　　号	UXXXXXXX
指导教师	万继光

2024 年月日

目 录

1 课程背景

1.1 PingCAP

PingCAP 是一个致力于构建全球领先开源数据库的企业。其产品 TiDB，是一个一站式的实时 HTAP（混合事务/分析处理）数据库。TiDB 是强一致事务性的分布式数据库，提供故障自恢复的高可用性和在线弹性水平扩展，可跨数据中心部署。这款开源项目在 GitHub 上收获了超过三万颗星。

企业级的高性能、易扩展的特性，让 PingCAP 的 TiDB 数据库得到了广泛的应用和认可。同时，PingCAP 在开源社区也持续积极贡献并与全球开发者深度合作，共同推动全球数据库技术的发展。

1.2 Talent Plan

PingCAP 致力于扩大开源数据库社区，为此专门推出了 PingCAP Talent Plan 系列课程（https://tidb.net/talent-plan）。TiDB 社区以 Talent Plan 开源数据库开发课程为依托，联合优秀高校和企业，建设成对全国各高校数据库开发人才培养的最佳实践平台。既能帮助学习者掌握数据库开发的理论知识，进行实际数据库开发殿炼，又能给与学习者使用开源资源，开发开源软件的培养。

在 Talent Plan 中，开发者可以选择以下路径：

1) tinysql：实现一个 Mini 版本的分布式关系型数据库
2) tinykv：实现一个 Mini 版本的分布式 Key-Value 数据库
3) 进阶：参与 TiDB 工业级开源项目开发
4) 进阶：参与 TiKV 的工业级开源项目开发
5) Rust 编程原理与实践

本课程基于路径二 tinykv 展开，在学习过程中，参与者不仅可以增加一条高含金量项目实践经验，在简历上也会更具吸引力，同时可以加深对业界顶尖的分布式数据库 TiKV 的理解。

2 内容简介

2.1 TinyKV 系统整体架构

TinyKV（https://github.com/tidb-incubator/tinykv）是一个基于 Raft 一致性算法的分布式键值存储系统。与存算分离的 TiDB + TiKV + PD 的架构类似，TinyKV 仅关注分布式数据库系统的存储层。

TinyKV 系统整体架构如下图 2-1。其中，SQL 层为 TinySQL（详见 https://github.com/tidb-incubator/tinysql）；TinyScheduler 为整个 TinyKV 集群的控制中心，该组件收集来自 TinyKV 的心跳信息，生成调度任务并将任务分配给 TinyKV 实例。所有实例间通过 RPC 进行通信。

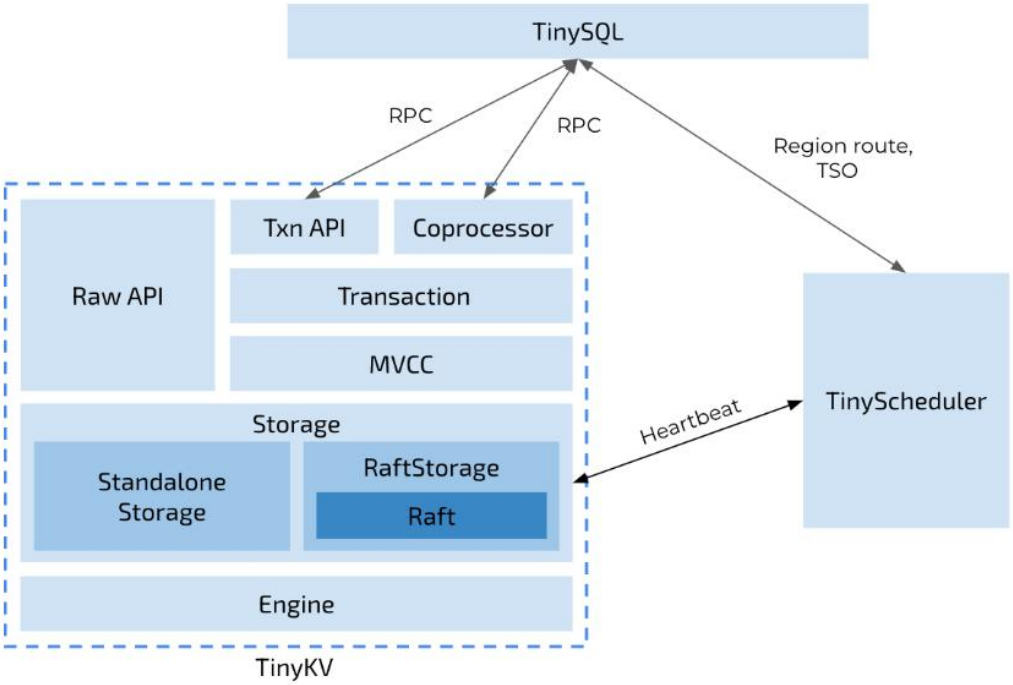

图 2-1 TinyKV 系统整体架构

TinyKV 的实现包括四个子项目：Standalone KV，Raft KV，Multi-raft KV 和 Transaction。

2.2 Project 1 Standalone KV

在项目一中，我们需要实现一个基本的键值操作服务。Standalone 意味着单节点而非分布式系统。我们将熟悉 Go 语言编程以及键值操作和 Column Family

等基础概念。

具体而言，需要通过调用底层的 API 实现一个裸的键值操作服务，该服务支持四个基本操作：put/delete/get/scan。该项目的实现可以分为 2 个步骤，包括：

1) 实现独立的存储引擎

2) 实现裸的键值处理服务

该项目相对简单，对后续项目没有影响，旨在让我们快速入门和熟悉整个环境。

2.3 Project 2 Raft KV

项目二要求我们实现一个具有容灾能力的键值服务，其结构为图 2-2 中 RaftStorage 部分。

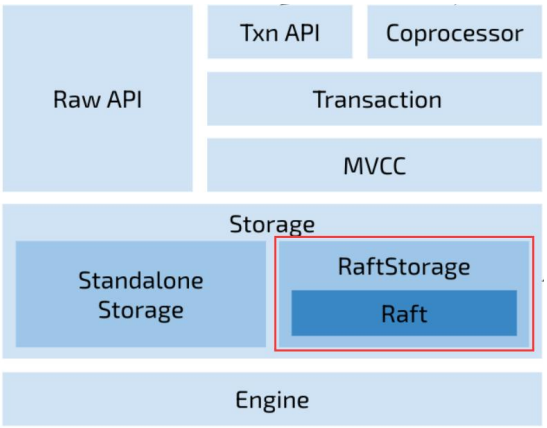

图 2-2 Project 2 结构

2.3.1 Part A

Part A 需要根据论文实现基本的 Raft 一致性共识算法。

需要实现的代码位于 raft/目录下，该文件夹中已经包含了一些框架代码和测试用例。该部分的实现包括三个步骤：

1) 领导者选举：我们需要实现领导者的选举过程，以确定集群的领导者。

2) 日志复制：在确定领导者之后，我们需要实现日志的复制过程，以确保所有节点上的日志一致。

3) 原始节点接口：我们需要提供一个原始节点接口，使得领导者可以与其他节点进行通信。

2.3.2 Part B

Part B 要求我们在单个 Raft 组上构建可容灾的键值服务。

首先，我们需要使用 Raft 算法在多服务器之间复制日志，日志项包含有待应用到键值服务器的操作（例如：Put、Get 和 Delete 请求）。然后在日志被复制多数服务器并应用到状态机后，对客户端发出的请求进行相应的回复。

该部分需要理解三个术语：Store 为 tinykv-server 的一个实例，Peer 代表运行在 Store 上的一个 Raft 节点，Region 是 Peer 的集合，也被称为 Raft 组。Project2 进行了简化，在一个 Store 上只有一个 Peer，一个集群中只有一个 Region。所以现在不需要考虑 Region 的范围。在 project3 中，我们会进一步介绍多 Region 的情况。

2.3.3 Part C

Part C 主要实现 Raft 日志压缩和快照功能。

RaftLog 是一种非常关键的数据结构，它保证了分布式数据的一致性。但是持续的数据写入会导致日志的无限增长，占用大量的存储空间，这在实际环境中是不可行的，所以我们需要实现垃圾收集机制（GC）。我们会引入快照机制进行垃圾回收，并且当系统需要恢复的时候，快照可以用于数据恢复。同时我们要保证在用快照进行数据恢复的过程中，系统可以正常运行且高可用。

2.4 Project 3 Multi-raft KV

2.4.1 Part A

单个 Raft 组难以进行扩展，因此需要实现多 Raft 组分别负担部分数据，如下图 2-3。

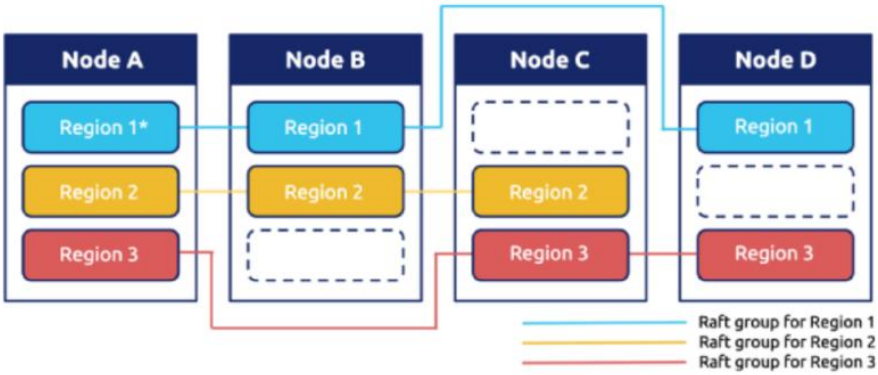

图 2-3 multi-raft 结构

在这一部分，我们将实现成员变动及领导权变动机制，针对基础的 Raft 算法进行扩展，这是后续两个部分所需的功能。成员变动，即配置更改，被用来增加或移除 Raft 组的成员，这会改变 Raft 组的法定人数，所以要小心处理。领导权的变动，即领导权的转移，被用于将领导权转移到另一个成员，这对于实现负载均衡十分有用。

2.4.2 Part B

由于 Raft 模块已经支持了成员变动和领导权变动，我们需要基于 Part A 来支持这些管理命令，包括：

1) CompactLog（已在 Project 2 的 Part C 实现）
2) TransferLeader
3) ChangePeer
4) Split

其中，TransferLeader 和 ChangePeer 是基于 Raft 的领导权和成员变动支持的命令。它们将被用作平衡调度器的基本操作步骤。Split 命令将一个 Region 分割成两个区域，这是实现多 Raft 的基础。我们将一步步实现这些命令。

2.4.3 Part C

在本部分，我们将为调度器实现两项功能。调度器用于平衡 TinyKV 集群的负载，它通过接收来自每个区域的定期心跳请求来更新 region 信息，并根据这些信息进行决策。例如，如果发现有 store 拥有过多的 region，它会将一些 region 移动到其它 store 中。这些操作将会随着对应区域心跳请求的响应进行。我们将

按照指导和框架，逐步完成这些工作。

2.5 Project 4 Transaction

在项目四中，我们需要实现图 2-4 中红色部分。

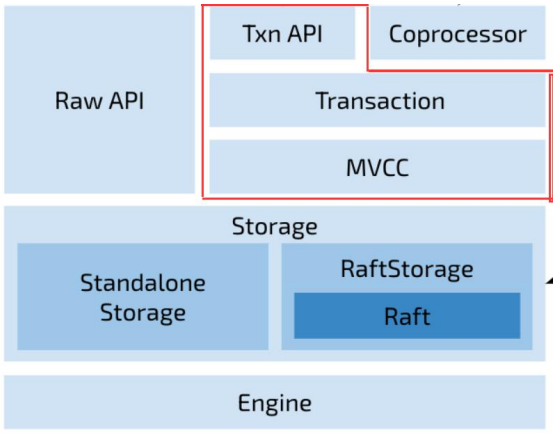

图 2-4 Project 4 结构

在之前的项目中，我们已经创建了一个键值数据库，它利用 Raft 在多个节点间保持一致性。然而，要想实现真正的可扩展性，数据库必须能够处理多个客户端。面临多个客户端时，一个问题出现了：如果两个客户端“同时”试图写入同一个键会发生什么情况？如果客户端写入并立即读取该键，他们是否应该期望读取的值与写入的值相同？在 Project 4 中，我们将通过在数据库中构建一个事务系统来解决这类问题。

该事务系统将是客户端（TinySQL）和服务器（TinyKV）之间的合作协议。只有两者都正确实现，才能保证事务属性。我们将为事务请求提供一个完整的 API，与在 Project1 中实现的 raw request 无关。

事务承诺提供快照隔离（Snapshot Isolation，SI）。这意味着在一个事务内，客户端将在整个事务的开启时读取被冻结在时间中的数据库（事务看到了数据库的一致视图）。事务操作的所有内容要么全部写入数据库，要么一个也不写入（如果与其他事务发生冲突）。

为了提供 SI，我们需要改变在后台存储中存储数据的方式。我们将为每个键和时间（由时间戳表示）存储一个值，而不仅仅是存储每个键的值。这被称为多版本并发控制（Multi-Version Concurrency Control，MVCC），因为对每个键

来说，我们存储了多个不同版本的值。

我们将在 Part A 中实现 MVCC。在 Part B 和 Part C 中我们将实现事务接口。

3 评分标准

本课程的最终提交材料包括：

1) 工程代码

2) Talent Plan 线上结业证书（可选）

3) 简要课程报告，内容主要包括项目实现、论文、算法的学习记录、整个项目架构的理解、测试结果、总结和心得等。

具体的评分标准如下表 3-1：

表 3-1 评分标准

<table>
<tr><td colspan="2">项目</td><td colspan="2">成绩占比</td><td>标准</td></tr>
<tr><td colspan="2">Project1</td><td>10</td><td>10</td><td rowspan="9">代码完善
通过测试</td></tr>
<tr><td rowspan="3">Project2</td><td>2a</td><td>15</td><td rowspan="3">40</td></tr>
<tr><td>2b</td><td>15</td></tr>
<tr><td>2c</td><td>10</td></tr>
<tr><td rowspan="3">Project3</td><td>3a</td><td>5</td><td rowspan="3">20</td></tr>
<tr><td>3b</td><td>10</td></tr>
<tr><td>3c</td><td>5</td></tr>
<tr><td colspan="2">Project4</td><td>10</td><td>10</td></tr>
<tr><td colspan="2">报告</td><td>20</td><td>20</td><td>按模板完善</td></tr>
</table>

4 学习建议

4.1 推荐系统配置

推荐的硬件配置包括四核八线程的 CPU，16GB 以上内存，以及 SSD 固态硬盘。对于环境配置，建议使用 Linux 虚拟机或 WSL2，并配置好 Golang 编程环境。

4.2 Golang 编程

学习并熟悉 Slice、Map、Channel、Interface 以及一些并发编程的基础知识。同时，需要熟悉 Golang 的单元测试工具。

4.3 Raft 算法

认真阅读论文、项目文档与注释，以深入理解 Raft 算法。可以参考 etcd 的实现来提高理解，但是请不要直接抄袭。值得注意的是，TinyKV 中的 Raft 实现与论文中的有一些区别，在需要时可以采用面向测试编程。

4.4 分布式系统 Debug 建议

在进行分布式系统的 Debug 时，建议多打日志，学习使用各种工具（如命令行）来筛选日志内容。同时，进行多次的运行测试脚本，尽可能地触发各种 corner case。在 Debug 的过程中保持耐心和冷静也是非常重要的。

4.5 报告建议

在报告中记载过程中遇到的 bug 与解决方案，并且不要局限于项目中完成的模块，需要关注整体的架构。

5 课程资源

本课程参考资源网址如下：

1) Golang
   - Learn X in Y minutes Where X=Go
   - A Tour of Go
2) Raft
   - 英文版本、中文翻译、扩展版本
   - etcd 源码、动画演示
3) Percolator
   - 英文版本、中文翻译
4) 有帮助的博客
   - 2021 Talent Plan KV 学习营结营总结
   - Talent-Plan - HUSTPORT | 综合交流平台
5) 视频资料
   - 2021 Talent Plan KV 学习营分享视频
   - 2021 Talent Plan KV 学习营推荐课