

大规模信息系统构建技术导论

分布式MiniSQL系统个人报告

2022-2023学年 春夏学期

组员信息（第一行请写组长信息）

|  |  |
| --- | --- |
| 学号 | 姓名 |
| **3200104091** | **沈轩喆** |

2023年 5月 16日

目 录

|  |  |
| --- | --- |
| 1 引言．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 1 |
| 1.1 设计目标．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 1 |
| 1.2 设计说明．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 1 |
| 2 总体设计．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 2 |
| 2.1 系统总体架构．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 2 |
| 2.2 Master设计．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 3 |
| 2.2.1 架构设计．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 3 |
| 2.2.2 工作流程．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 4 |
| 3 核心功能模块设计．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.1 主从备份．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.2 负载均衡．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.3 容错容灾．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.3.1 Region Server失效．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.3.2 Master失效．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 5 |
| 3.4 通信规范．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 6 |
| 4 系统测试．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 7 |
| 4.1 初始化．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 7 |
| 4.2 Region Server 连接．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 7 |
| 4.3 处理Client请求．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 8 |
| 4.4 主从备份．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 8 |
| 4.5 Region Server失效处理．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 8 |
| 4.6 Master失效处理．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 8 |
| 5 总结．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．．． | 9 |

**1 引言**

1.1 设计目标

本项目是基于《数据库系统》课程学习的数据库基本知识和《大规模信息系统构建技术导论》课程学习的分布式系统与大规模软件系统构建的知识，所完成的分布式关系型miniSQL项目。

本项目设计并实现一个分布式的关系型SQL引擎，除实现数据库各类基本操作（增删改查、索引）、SQL语句执行等基本功能外，包含Zookeeper集群、客户端、主从节点等多个模块，具有数据分区、均衡负载、客户端缓存、副本管理、容错容灾等功能。

本系统使用Java语言进行项目构建，使用Github进行版本管理和协作开发，由小组五位成员共同完成。

1.2 设计说明

本程序采用Java程序设计语言，在VS code平台下编辑、编译与调试。具体程序由5人组成的小组开发而成。具体分工如表1所示：

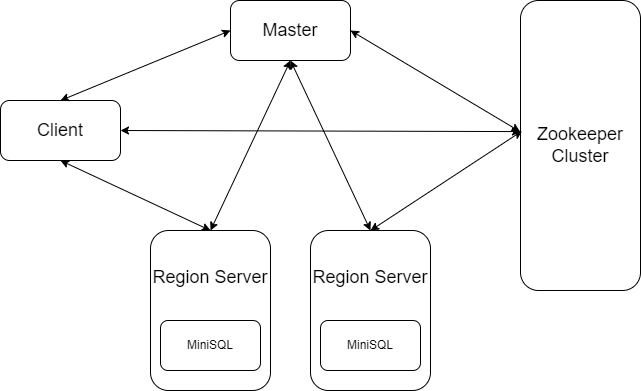
**表1 成员分工表**

|  |  |  |
| --- | --- | --- |
| **成员姓名** | **学号** | **分工** |
| 沈轩喆 | 3200104091 | Master模块的设计，负载均衡、容错容灾 |

该文档是沈轩喆的个人模块报告，仅包含我个人在本次课程项目中完成的一系列工作。

**2 总体设计**

2.1 功能模块设计

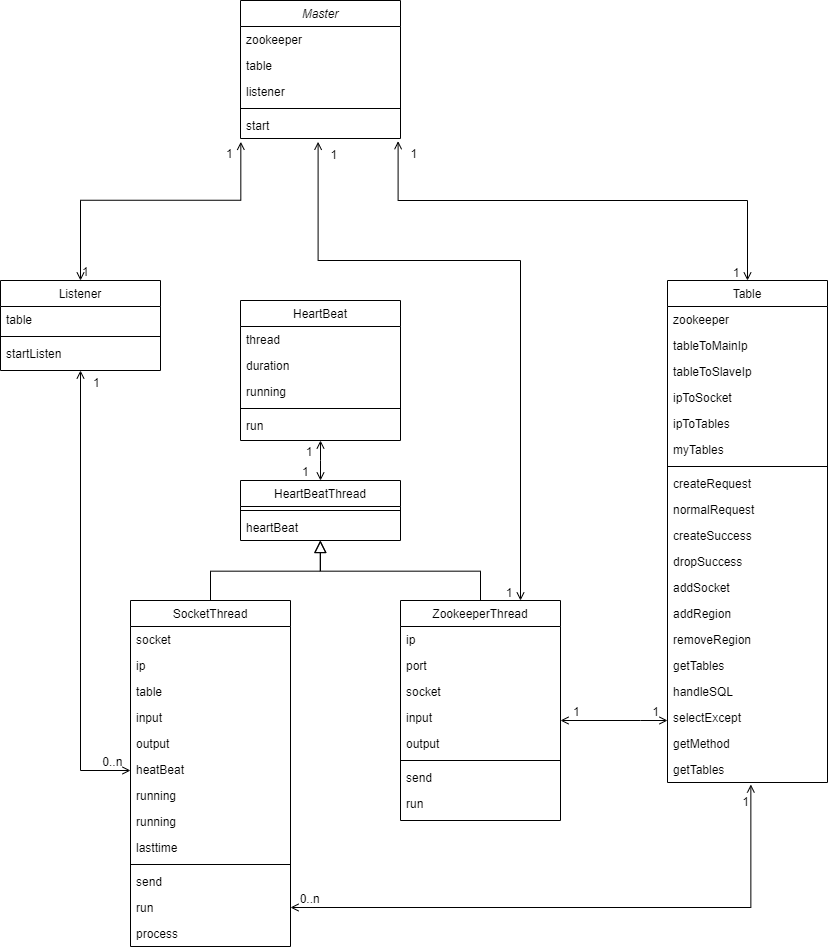
系统的总体架构设计如下图所示：

**图2.1 总体架构设计图**

系统整体架构分为 Zookeeper Cluster、Master、Region Server 和 Client 四个模块。其中，Region Server 底层由 MiniSQL 提供服务。

2.2 Master设计

2.2.1 架构设计

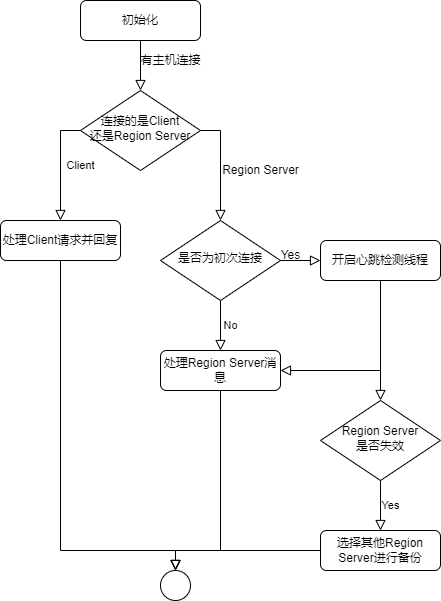


**图2.2.1 Master类图**

Master中包含Master、Listener、Table、SocketThread、ZookeeperThread、HeartBeat六个类及HeartBeatThread一个接口。

* Master，负责初始化
* Listener，持续在某一端口进行监听，当产生一个新的socket连接时，创建一个SocketThread对象来进一步处理
* Table，负责记录每张表的主从备份所在的Region Server，当有数据被修改时完成主从备份的同步更新
* HeartBeatThread，一个接口，定义了heartBeat函数
* HeartBeat，完成心跳的定时检查与发送
* SocketThread，与一台主机通信（可以是Client或者Region Server），并处理它发送的消息并回复
* ZookeeperThread，与Zookeeper通信，发送消息并定时发送心跳

2.2.2 工作流程

Master的工作流程可以用下面的流程图来表示：

**图2.2.2 Master流程图**

**3 核心功能模块设计**

3.1 主从备份

本系统对数据库中所有表进行了主从备份，分别分布在不同的两台Region Server上。来自Client的请求会先送到主备份所在Region Server处理，若请求涉及到表数据的更新，主备份的Region Server会向Master报告，再由Master通知从备份所在的Region Server更新数据。

通过主从备份，本系统在Region Server失效时不至于直接丢失数据，并在此基础上可以实现容错容灾。

3.2 负载均衡

本系统对于Region Server的负载进行了一定处理，当Client请求新建表或Region Server需要备份时，会由Master选择当前最空闲的Region Server完成请求。

3.3 容错容灾

3.3.1 Region Server失效

当一台Region Server失效时，其向Master发送的心跳消息会停止，Master定期检查会发现该Region Server未按时发送心跳信息，认为其失效。进而遍历该Region Server所拥有的所有表，选择空闲的Region Server要求它们对这些表进行备份。

通过这种方式，只要一张表的主从备份所在的两台Region Server不同时失效，就可以保证数据安全。

3.3.2 Master失效

当Master其向Zookeeper发送的心跳消息会停止，Zookeeper定期检查会发现Master未按时发送心跳信息，认为其失效。进而选择一台Region Server成为新的Master，广播通知所有的Region Server。新的Master接收其他Region Server发送的信息，并将自身原本存储的表备份。

通过这种方式，即使Master失效系统也能继续正常工作。

3.4 通信协议

Master需要与Client, Region Server, Zookeeper进行通信。

与Client的连接为短连接，Client向Master发送请求，Master回复后连接即断开。

与Region Server的连接为长连接，连接开始后Region Server首先向Master报告自己的IP，监听端口及已有的表，之后不断发送心跳信息。同时，该连接还用于主从备份的同步及其他Region Server失效时的通信。

与Zookeeper的连接也是长连接，连接开始后Master首先向Zookeeper报告自己的监听端口，之后不断发送心跳信息。

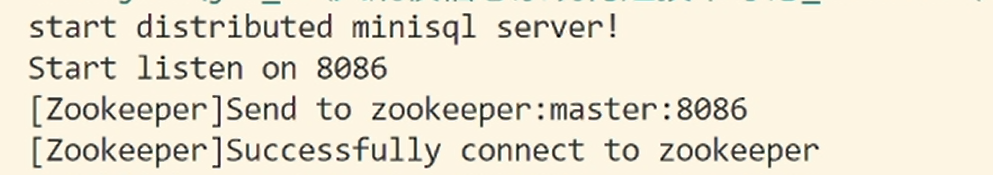
**表3.4 通信协议**

|  |  |  |  |
| --- | --- | --- | --- |
| 协议格式 | 返回格式 | 方向 | 用途 |
| Master:8086 | 无 | Master->Zookeeper | Master报告自身的监听端口 |
| ALIVE | 无 | Master->Zookeeper | Master向Zookeeper发送心跳信息 |
| <show> | {$table1$table2$table3$...} | Client->Master | Client向Master询问所有表的表名 |
| <get>tablename | ip:port | Client->Master | Client需要对表操作，Master告知对应Region Server的地址 |
| <create>tablename | ip:port | Client->Master | Client需要创建表，Master告知对应Region Server的地址 |
| <quit> | 无 | Client->Master | Client退出 |
| (hello)port:table1,table2,table3... | 无 | Region Server->Master | Region Server上线，向Master报告自己的监听端口及已有表 |
| (CREATE)tablename | 无 | Region Server->Master | Region Server成功创建表，通知Master |
| (DROP)tablename | 无 | Region Server->Master | Region Server成功删除表，通知Master |
| (MODIFY)sql statement | 无 | Region Server->Master | Region Server成功修改表，通知Master |
| (ALIVE) | 无 | Region Server->Master | Region Server发送心跳信息 |
| (sql)sql statement | 无 | Master->Region Server | 在主备份修改时，Master通知从备份同步修改 |
| (copy)ip:port:tablename | 无 | Master->Region Server | Master通知Region Server到所述ip:port上备份表 |

**4 系统测试**

4.1 初始化

一个server上线时，会先向Zookeeper报告，若Zookeeper要求其成为Master，则运行Master，完成初始化并开始监听。

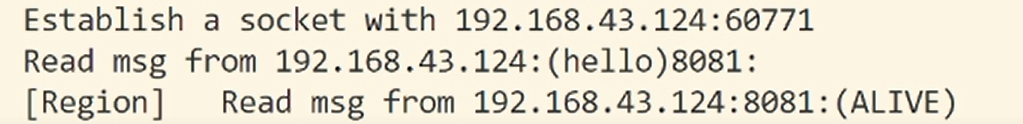


同时，Master开始向Zookeeper发送心跳信息。



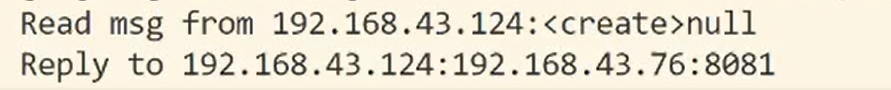
4.2 Region Server连接

当一个Region Server上线时，会与Master建立连接并发送hello消息，随后 Region Server开始向Master发送心跳信息。



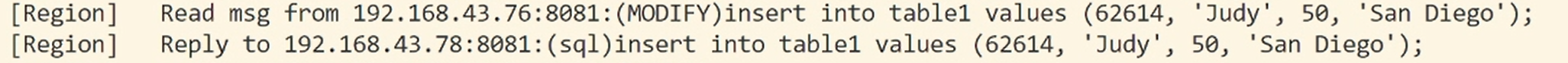
4.3 处理Client请求

Client向Master发送请求，Master会回复其处理该请求的Region Server的IP及端口。



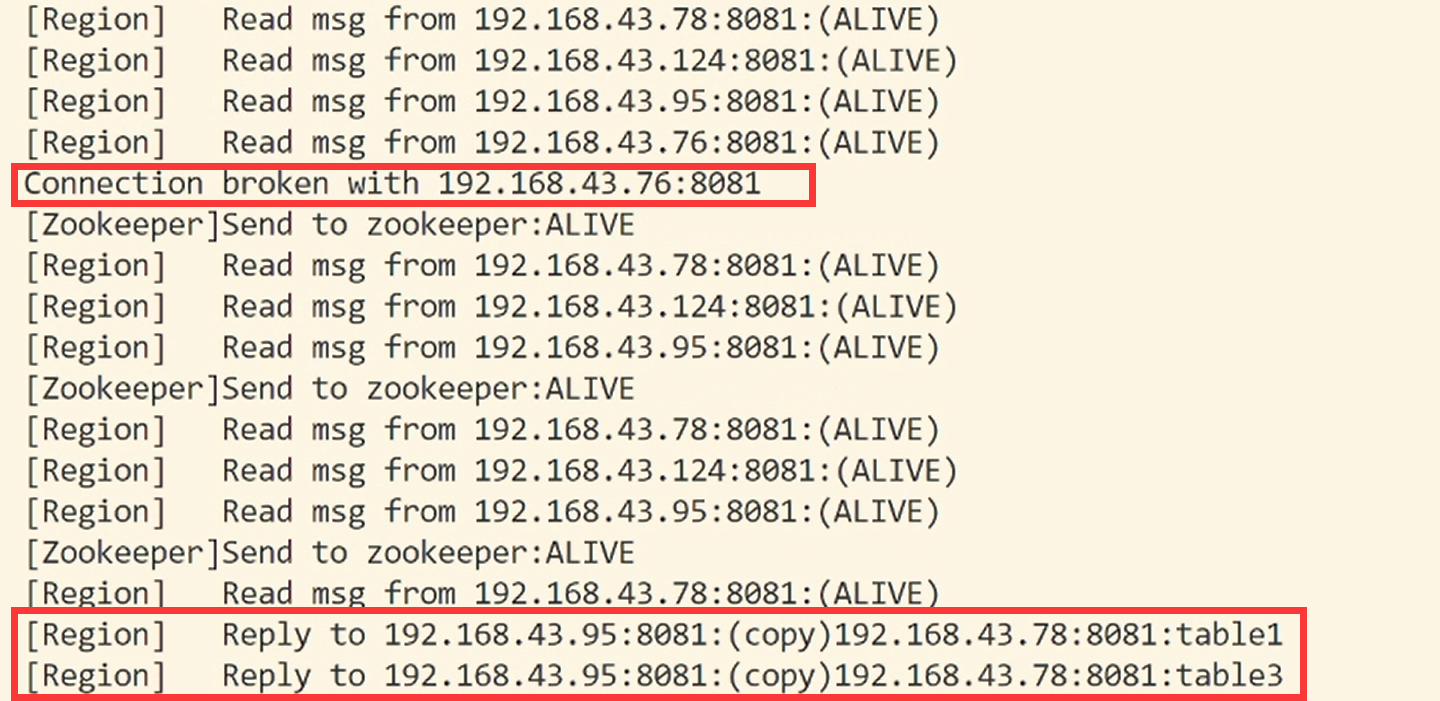
4.4 主从备份

当主备份的表数据改变时会通知Master，再由Master通知从备份同步数据。



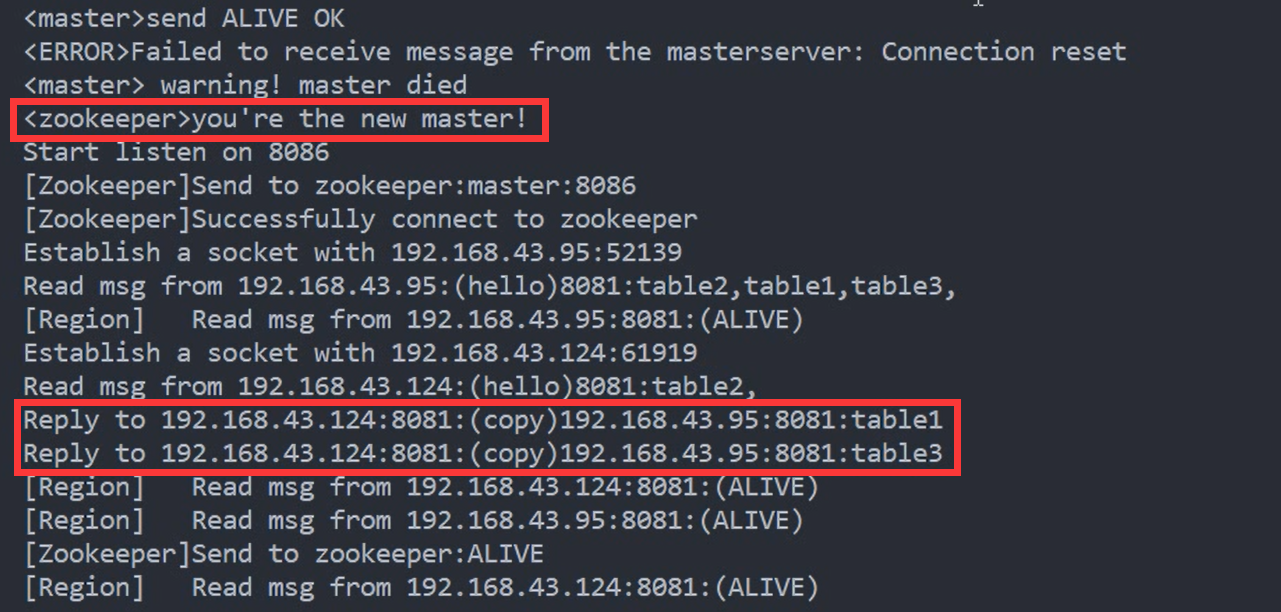
4.5 Region Server失效处理

当一个Region Server停止发送心跳信息一段时间后，Master会认为其失效，并要求其他空闲的Region Server重新备份失效Region Server上的所有表。



4.6 Master失效处理

Master失效后，Zookeeper会选择并通知一个Region Server成为新的Master，新的Master会接收其他Region Server发送的表信息，并将自己原本作为Region Server时存储的所有表备份。



**5 总结**

在这次课程项目中，我们小组分工明确，在已有的MiniSQL上实现了分布式处理，最终完成了一个具有分布式存储、负载均衡、容错容灾等功能的分布式MiniSQL。该项目难度较大，但也在很大程度上提高了我们的能力，加深了对课程内容的理解。