Massive Mini SQL 总体设计文档

学号	姓名
3180105698	马梓睿
3180105415	黄泽煊
3180101872	李保宏
3180103654	沈乐明
3180105874	林瑞潇

1. 程序概要

项目背景

大数据是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。而随着社会和科学技术的发展，人们之间的交流越来越密切，生活也越来越方便，大数据就是这个高科技时代的产物。阿里巴巴创办人马云来台演讲中就提到，未来的时代将不是IT时代，而是DT的时代，DT就是 Data Technology 数据科技，显示大数据对于阿里巴巴集团来说举足轻重。

大数据的价值体现在以下几个方面：

对大量消费者提供产品或服务的企业可以利用大数据进行精准营销 做小而美模式的中小微企业可以利用大数据做服务转型 面临互联网压力之下必须转型的传统企业需要与时俱进充分利用大数据的价值。

但是，面对越来越多的数据，一个能够分布式管理大量数据并且保证数据一致性的数据库亟待开发。不同于类似亚马逊搭建的Dynmo数据库平台，本项目开发的数据库更强调其一致性而非可用性，因此更类似于HBase。

项目前景

大数据的发展着重凸显出以下趋势：

- 数据的资源化
- 与云计算的深度结合
- 科学理论的突破
- 数据科学和数据联盟的建立
- 数据泄露泛滥
- 数据管理成为核心竞争力
- 数据质量是商业智能成功的关键
- 数据生态系统复合化程度加强

同时，经李克强总理签批，2015年9月，国务院印发《促进大数据发展行动纲要》，系统部署大数据发展工作。《纲要》明确，推动大数据发展和应用，在未来5至10年打造精准治理、多方协作的社会治理新模式，建立运行平稳、安全高效的经济运行新机制，构建以人为本、惠及全民的民生服务新体系，开启大众创业、万众创新的创新驱动新格局，培育高端智能、新兴繁荣的产业发展新生态。 在兼具天时地利人和的情况下，本项目开发的平台将具有重大的意义和非凡的影响力，将会大大减轻用户的数据管理压力。

项目需求

我们将要实现一个拥有完整功能的分布式数据库模型，需要实现以下功能：

- 完整的数据库功能
- 同时部署多个客户端
- 保证大量客户端访问过程中的数据一致性
- 少数节点错误的数据恢复

2. 模块划分

MSQL

底层的数据库实现，实现一个完整的数据库，包括数据的增加修改删除等基本操作。 MSQL采用C++编写，不涉及大规模框架。

MSQL Controller

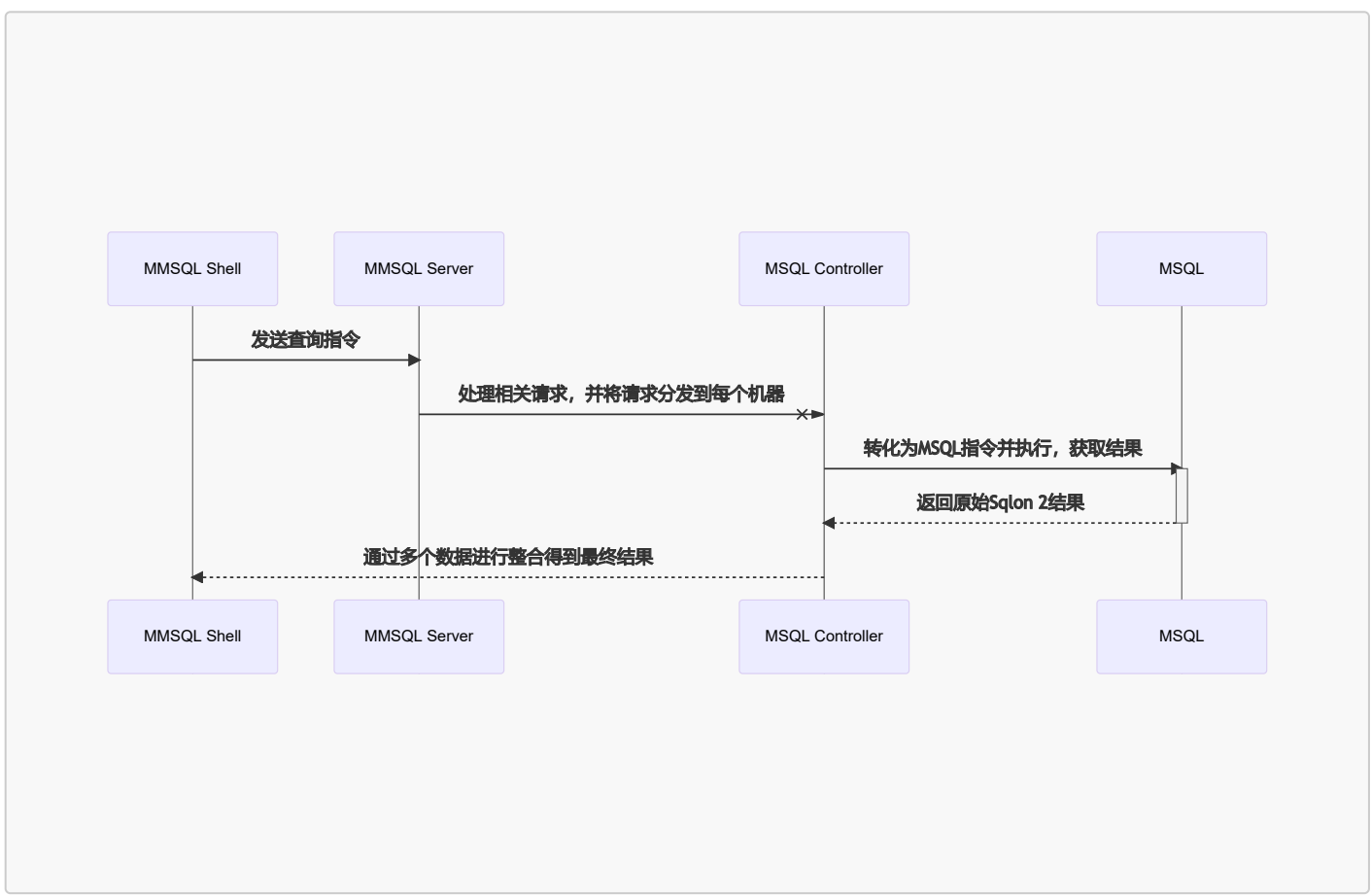
Controller需要集成Zookeeper，需要告诉其它机器自己保存了哪些表。同时通过Zookeeper来标识自己机器状态，副本管理、容灾相关的内容也要在这部分完成，通过主从节点的方式实现读写。还有一点就是需要调用MSQL程序并完成一些内容。

MMSQL Server

Server和主节点不同，主节点是我们的数据中心，主要用于保存数据，而Server保存的是元数据等信息，也就是存一个表在哪些机器上面以及有哪些机器可以访问。同时，MMSQL Server也负责负载均衡等工作。同时它也和Zookeeper交互。

MMSQL Shell

Shell是客户端，用Java写，主要需要解析SQL语句，然后获取所需要的表，如果多个表还要把数据传输过来在本地join。



核心技术

- Sqlon 2：Sqlon是我们受到JSON的数据模型启发而为了开发MSQL独立设计的一个通用序列化库，特点是描述文件和数据文件分离。而Sqlon 2则是在Sqlon的基础上进行了一定的改进，包括更加数学上完整的数据模型以及对流传输的优化。
- MSQL B+ Tree：这是对普通的B+树经过一定的修改后得到的一个牺牲一定效率但操作更加灵活的B+树，特点是和序列化模块紧密结合，其中包括节点缓存功能等高级操作。
- MSQL File Block Alocator：为了解决能够在一个文件存放多个表和索引等信息并且为了能够灵活分配空间而实现的一种特殊的数据结构，这个数据结构受到了FAT文件系统、内存管理机制等的启发，能够实现基本的按页分配空间功能。
- ZooKeeper：为了解决同步等问题引入的成熟解决方案，是项目的核心。

3. 功能设计

MSQL

负责基础数据库功能的实现，包括：

- 数据增添
- 数据修改
- 数据删除
- 数据查询
- 数据索引

MSQL运行在服务端上，和MSQL Controller一一对应。用户发送的一个请求最终会转化为MSQL的直接操作。而MSQL作为一个C++程序具有一定的独立性，MSQL和其Controller通过网络通信的方式进行交互。MSQL启动时会接收到Controller传送的参数，MSQL根据参数完成初始化操作并作为客户端连接Controller对应的端口，建立本地TCP长连接。

MSQL保存整个数据库的部分表，但其不负责全局的操作。比如当两个表要进行Join时，这个操作不是由MSQL完成的，Controller会负责这些操作。同时，MSQL本身的稳定性是拥有冗余的。当MSQL崩溃时，MSQL Controller会帮助重新启动，同时完成相应的数据恢复操作。因此当出现一些严重错误时，我们借鉴Docker的思想，直接Kill掉MSQL然后重新启动，防止从错误中恢复的过程影响业务。

MSQL Controller

每一个MSQL Controller都对应控制一个MSQL，通过接收MSQL Shell发来的请求返回对应的查询结果或者修改结果，它将MSQL Shell的请求转换为MSQL指令并且通知MSQL实现，然后直接将结果返回给MSQL Shell，同时，它还需要和ZooKeeper协同，完成相应的分布式功能。

MSQL Controller运行在服务端上，和MSQL一一对应。启动服务端时，首先启动的便是MSQL Controller。它会启动一个MSQL实例，并连接ZooKeeper，通知集群上的机器自己已经加入。这时，MSQL Controller会通过指令通知MSQL，获取MSQL中存储的所有表的元信息，将其放到ZooKeeper上向其他机器共享，主要是通知MMSQL Server。此后Controller就进入工作状态，能够接收MMSQL Shell发来的指令并执行。

MSQL Controller遵从主从模式架构。集群初始化时会选定一个Master机器，任何写操作都会发到Master，Master会根据拥有该表的机器而将写操作发送到所有机器，只有所有机器都写成功了这次写操作才成功。而读操作可以在任何机器上进行。整个操作的协同也是通过ZooKeeper来实现的。

MSQL Controller本身承载数据分布、集群管理等功能，而不提供负载均衡等功能，这些在MMSQL Server上面承载。

在执行Join操作时，其中一个机器（根据负载均衡算法选择的）会获取对应的表，并在对应的MSQL上执行Join操作。最终得到的结果直接返回给用户。因此Join操作需要MMSQL Shell自行选择机器、传输对应表、进行Join请求。整个过程MSQL Controller只是作为命令的执行者。

MMSQL Server

MMSQL Server储存的是MSQL Controller的信息，包括所有表的位置信息还有节点是否有效，MMSQL Shell每次查询或者修改时都需要事先在MMSQL Server中找到表的具体位置和有效节点，才能与MSQL Controller通讯。它也需要和ZooKeeper协同，保证信息的正确和完备。

MMSQL Server需要首先启动，然后才能启动其它Controller。Server中保存的信息包括启动的机器的信息、一台机器（Controller）保存了哪些表。当用户进行一个查询时，它会首先向MMSQL Server发送一个请求，Server采用负载均衡算法从拥有该表的机器中选择一个机器，然后用户直接向对应的Controller发送请求，网络流量不经过Server。同样地，如果执行一个命令的过程中失败了，那么用户也会通知Server，Server可以重新进行命令的分发。

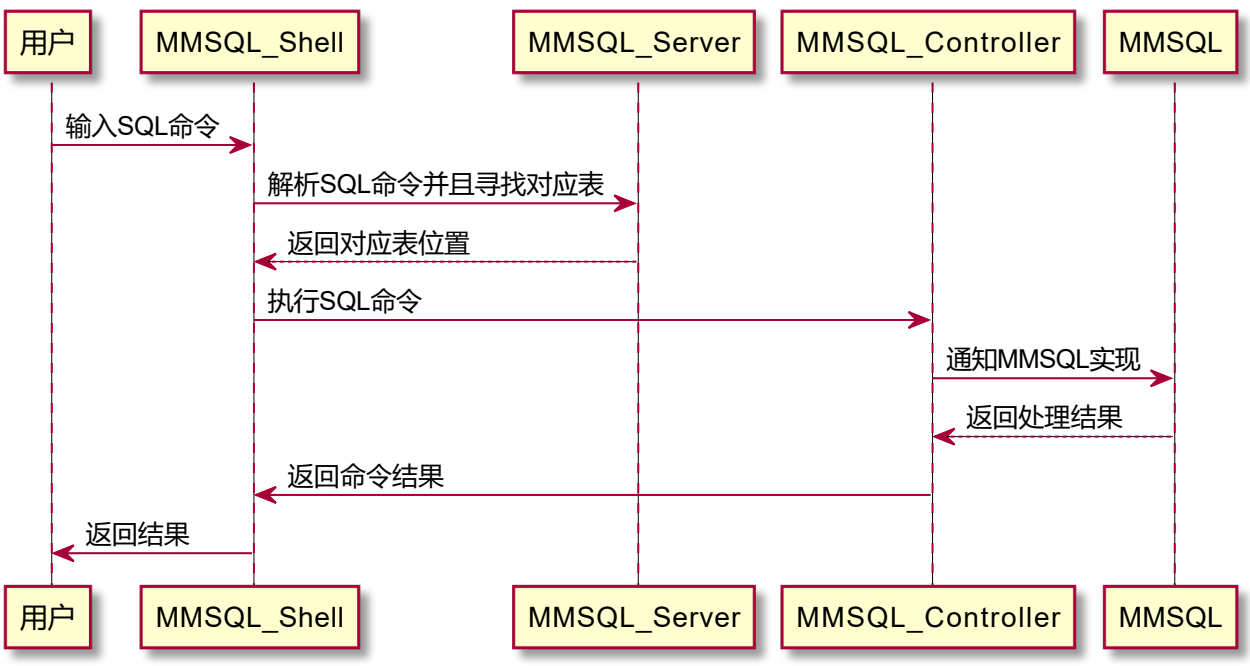
MMSQL Shell

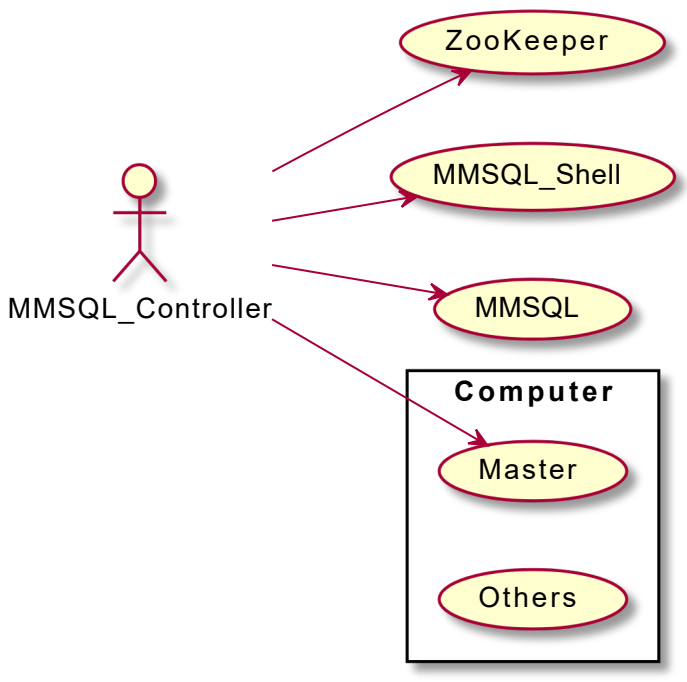
客户端，通过接收使用者的输入，先在MSQL Server中找到对应的表的位置，然后通过和对应表的MSQL Controller进行通信，进行查询或者修改操作

Shell需要对用户的SQL命令进行解释，然后向MMSQL Server发送相应的请求，最终拿到真正执行命令的Controller主机。Shell接下来与Controller进行通信，返回的结果也需要MMSQL Shell进行解释，转化为人类可读的数据。

在执行Join操作时，Shell需要首先向MMSQL Server发送相应请求，Server会返回两张表的机器，以及一个建议执行Join的机器。Shell会和这个执行Join的机器进行通信，按照需求发送一个从其它Controller同步对应表的请求。拿到表后Controller会执行Join操作，从而得到最终的结果。

4. 数据流图





用例编号	MS-MC-01	用例名称	MMSQL_Controller
角色	控制MSQL	需求来源	MMSQL
描述	用以接收请求，通过接收MSQL Shell发来的请求返回对应的查询结果或者修改结果		

用例编号	MS-MC-02	用例名称	MMSQL_Controller
角色	控制MSQL	需求来源	MMSQL
描述	将MSQL Shell的请求转换为MSQL指令并且通知MSQL实现，然后直接将结果返回给MSQL Shell		

用例编号	MS-MC-03	用例名称	MMSQL_Controller
角色	控制MSQL	需求来源	MMSQL
描述	通知MSQL实现由Controller转换而来的请求，返回结果给MSQL Shell		

用例编号	MS-MS-01	用例名称	MMSQL_Shell
角色	接收用户输入	需求来源	MMSQL
描述	对用户的SQL命令进行解释，然后向MMSQL Server发送相应的请求，最终拿到真正执行命令的Controller主机		

用例编号	MS-MS-02	用例名称	MMSQL_Shell
角色	接收用户输入	需求来源	MMSQL
描述	在MSQL Server中找到对应的表的位置，然后通过和对应表的MSQL Controller进行通信，进行查询或者修改操作		

用例编号	MS-MS-03	用例名称	MMSQL_Shell
角色	接收用户输入	需求来源	MMSQL
描述	与Controller进行通信，返回的结果需要MMSQL Shell进行解释，转化为人类可读的数据		

用例编号	MS-ME-01	用例名称	MMSQL_Server
角色	保存表信息	需求来源	MMSQL
描述	储存MSQL Controller的信息，包括所有表的位置信息还有节点是否有效		

用例编号	MS-ME-02	用例名称	MMSQL_Server
角色	保存表信息	需求来源	MMSQL
描述	每次查询或者修改时都事先在MMSQL Server中找到表的具体位置和有效节点，然后与MSQL Controller通讯		

用例编号	MS-ME-03	用例名称	MMSQL_Server
角色	保存表信息	需求来源	MMSQL
描述	首先向MMSQL Server发送一个请求，Server采用负载均衡算法从拥有该表的机器中选择一个机器		

用例编号	MS-MQ-01	用例名称	MSQL
角色	实现基础数据库	需求来源	MMSQL
描述	从MMSQL_Controller拿到需要的数据并且向数据库中增添数据，返回添加是否成功		

用例编号	MS-MQ-02	用例名称	MSQL
角色	实现基础数据库	需求来源	MMSQL
描述	从MMSQL_Controller拿到需要的数据并且修改数据库中指定数据，返回修改是否成功		

用例编号	MS-MQ-03	用例名称	MSQL
角色	实现基础数据库	需求来源	MMSQL
描述	从MMSQL_Controller拿到需要的数据并且删除数据库中指定数据，返回删除是否成功		

用例编号	MS-MQ-01	用例名称	MSQL
角色	实现基础数据库	需求来源	MMSQL
描述	从MMSQL_Controller拿到需要的数据并且查询数据库中该数据的信息，返回查询是否成功		

用例编号	MS-MQ-02	用例名称	MSQL
角色	实现基础数据库	需求来源	MMSQL
描述	Shell输入建立索引指令后，给本地对应的表建立索引，返回建立是否成功		

5. 非功能需求

5.1 性能需求

能够在网络可接受的范围内使MSQL数据库部分能够达到50%以上负载运行，也就是大规模传输部分不应过多占用网络资源且负载均衡功能能够合理分配请求。

5.2 一致性需求

在非极端情况下，需要保证系统的最终一致性。

5.3 容灾需求

在任意不超过1/3的机器出现问题而无法访问的情况下，应该能够保证数据的完整性和系统的可用性。