集群间RootServer选主设计文档

修订历史

版本	修订日期	修订描述	作者	备注
0.1	2016-1-4	创建设计文档	庞天泽、储佳佳	

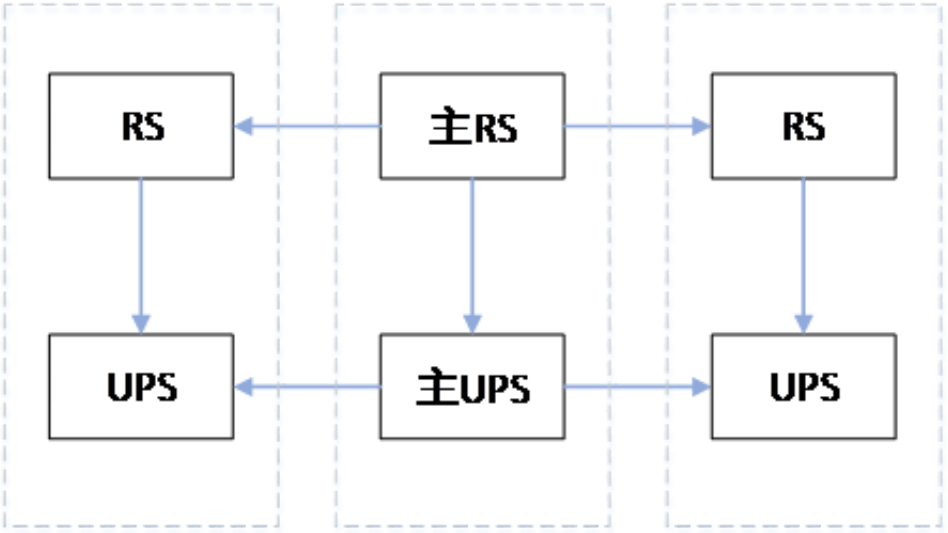
1 系统设计

1.1 综述

OceanBase是一款可扩展的分布式数据库，能够支持千亿级海量数据存储和访问。OceanBase由RootServer、UpdateServer、ChunkServer和MergeServer四类节点组成，其中RootServer是数据库的主控节点，负责数据库的管理、数据分布、配置信息维护等；UpdateServer负责数据库的事务处理，是唯一接受写入的节点。通常在数据库运行过程中，只有唯一的RootServer和UpdateServer完成其负责的功能，我们称之为主节点（如主RootServer，主UpdateServer）。当RootServer或UpdateServer故障时，数据库集群能在有限时间内自动将其它可用节点设置为新的主节点，从而保证数据库能够继续提供服务，这是OceanBase在高可用上的重要体现。

开源版本的OceanBase0.4.2中，通常在集群中部署主备两台RootServer节点，并采用Linux高可用集群方案（Corosync）实现双节点互备。在实际应用场景中，这种高可用方法存在一定的问题。RootServer依靠Corosync进行节点故障检测和主备节点切换，而Corosync一旦出现异常，就会导致方案失效，集群可能无法进行节点切换，甚至出现双主问题（两台主RootServer同时存在）。在我们OceanBase版本中，采用不同于0.4.2版本OceanBase集群架构，并实现基于Raft的RootServer选主和UpdateServer日志强同步，在一定程度上解决了0.4.2版本存在的问题。

我们在OceanBase0.4.2的基础上进行开发，尽可能做到架构直观明朗，同时又便于实现和控制，设计了三集群架构，如图1-1所示。每个集群内仅有一个RootServer节点、一个UpdateServer节点，并由各自的RootServer负责集群内的数据划分、负载均衡等管理事项。三集群架构中同一时刻只有唯一主RootServer存在，主RootServer所在的集群为主集群，主集群内的UpdateServer为主UpdateServer。非主集群均为备集群，备集群内的RootServer为备RootServer，备集群内的UpdateServer为备UpdateServer。集群间的数据同步通过主备UpdateServer间的事务日志同步实现。



三集群架构中，每个集群都是完整的OceanBase集群，只有主集群能够处理事务更新，备集群则主要用于备份数据更新，并在主集群故障时切换为新的主集群。当用户向系统发送读写请求时，MergeServer会将请求转发到主Updateserver来进行处理，并且只有当系统不在选举状态中时，才能处理读写请求。主节点和各个备节点之间通过租约保持联系，并检测节点故障。与高可用相关的主要涉及每个集群内RootServer和UpdateServer之间、三集群间各个Rootserver之间、以及各个UpdateServer之间的租约机制。

单机群内部RootServer和UpdateServer间的租约分析

集群内的RootServer与UpdateServer之间维持着租约，由RootServer主动检查该租约，如果发现UpdateServer即将过期，则发送续约请求更新租约。另外，UpdateServer也会定期检查自身租约是否过期，如果是是备节点，则向本集群内的RootServer重新注册，直到注册成功；如果是主节点，则会切换为备节点，也会不断向RootServer注册直到成功。

需要说明的是，主UpdateServer的状态会影响主RootServer向其它RootServer节点续约。主RootServer运行过程中会向其它备RootServer发送续约请求，而在发送请求前，先向主UpdateServer发送请求来获取最大日志号日志的时间戳，如果主UpdateServer故障或者已经切换为备UpdateServer，那么RootServer获取时间戳的操作就会失败。如果获取时间戳成功，则说明主UpdateServer正常运行。只有在主UpdateServer正常时候主RootServer才能进行续约。

三集群中各RootServer间的租约分析

各个RootServer记录自己的租约信息，并不断检查租约是否过期，租约期内节点处于正常状态。主RootServer在自身租约即将结束的时候，会向备RootServer发送续约请求，若超过半数RootServer成功更新租约，则主RootServer续约成功，主备节点的租约时间延长。RootServer一旦检测到租约过期，不论主备，都会在经历一个随机的等待时间后发起无主选举，直到成功选举出新的主节点。为了避免双主问题，我们要保证在选举开始时系统中不能有主节点存在，所以主RootServer被设置为先于其它RootServer过期。

三集群中各RootServer间的租约分析

UpdateServer之间通过主UpdateServer向备UpdateServer同步日志信息保持备UpdateServer的活跃状态（keep alive）。备节点定期检查最新日志的接收时间，来判断自己的租约是否过期。若备节点在规定时间内没有接收到主节点的日志信息，则认为自己过期了，于是准备向主节点重新注册。注册时，备节点首先通过集群内的RootServer获取主RootServer地址，再从主RootServer得到主UpdateServer的地址信息，最后向主UpdateServer注册。

我们设计并实现了三集群架构中RootServer选主模块，配合UpdateServer的日志强同步，使系统具有较高的可用性和可靠性。集群间RootServer选主主要包括两个部分，无主选举和主RootServer续约。当RootServer检测到租约过期时，就会发起无主选举，向系统中所有其它RootServer发投票请求，若在一定时间内能收到超过半数的支持票，则该RootServer成功当选为主RootServer，并和其它所有RootServer维持一个租约时间，在该租约时间内，主RootServer的身份都是有效的。为了避免频繁地换主，主RootServer租约即将过期时，会向其它所有RootServer发起续约请求，更新它们的租约。若在一定时间内能收到超过半数的响应，则说明续约成功，主RootServer会更新自己的租约，并在下个租约期内继续作为主节点运行。

1.2 名词解释

- leader：选举角色，表示RootServer此时是主节点，它所在集群作为主机群对外服务，同时向其他RootServer和本集群的UpdateServer不断发送租约消息，以确保这些节点承认自己的leader地位。
- follower：选举角色，表示RootServer是备节点，仅管理自己所在的备集群。
- candidate：选举角色，表示RootServer正处于选举过程中。
- obi_role：集群角色，值为master表示主集群，值为slave表示备集群。

1.3 功能

RootServer选主模块实现的功能包括以下几点：

- 主集群异常时，只要剩余可用集群的RootServer节点占总RootServer数量一半以上，即可保证能够在没有人工干预情况下，选举出新的主节点接管集群，使系统继续正常提供服务。集群异常包括RootServer不可用、UpdateServer不可用和网络故障等，我们在设计选主功能时，都细致考虑了这些问题。
- 增加RootServer间的租约机制。在系统正常运行时，主RootServer不断更新各个RootServer节点的租约，以避免因租约过期发生选举，保持系统稳定；与此同时，各个RootServer也不断检查租约是否过期，判断是否需要发起选举。
- 支持用户使用rs_admin工具主动发起RootServer选举。如果用户不希望当前主集群继续服务，可以通过运维工具向系统发送命令，重新发起选举。具体命令为：

```
$rs_admin -r < master_rootserver_ip > -p < master_rootserver_port > reelect
```

2 模块设计

2.1 RootServer选举子模块设计

2.1.1 结构

模块中使用ObElectionNode类存储选举相关的变量，如RootServer的选举角色、RootServer地址列表、选举信息、租约等。

```
Class ObElectionNode{
    /// 选举角色，包括leader、candidate、follower三种
    ObElectionRole role;
    /// 除本节点外的其它RootServer地址
    vector<ObServer> other_rs;
    /// 除本节点外的RootServer个数
    slave_rs_count;
    /// 选举阶段成功投票的对象
    ObServer votefor_
    /// 选举阶段确定的系统主RootServer
    ObServer leader_info;
    /// 租约值
    int64_t lease;
    .....
}
```

使用单独的线程循环检查ObElectionNode的信息，根据集群角色、选举角色、租约等信息执行租约检查、续租、选举等的选举模块的各种操作。该线程会依据选举角色实时修改集群角色，使选举结果能够反映到系统中，选举角色和集群角色的对应关系如下：

选举角色	集群角色
leader	master
candidate	slave
follower	slave

RootServer之间需要进行大量的信息交互，我们将选举模块涉及的信息封装成一个消息包进行传输，数据结构中设置标识来区别不同类型的消息。使用相同的接口发送消息包，接收方则根据消息包类型调用相应函数处理并回复结果。消息包中数据结构如下：

```
struct ObMsgRsElection{
    /// 发送消息包的节点地址
    ObServer addr;
    /// 发送方希望接收包更新的最新租约
    int64_t lease;
    /// 发送方集群的UpdateServer最大日志号日志的时间戳
    int64_t max_logtimestamp;
    /// 消息包类型
    int64_t type;
    /// 能否选举标记
    bool auto_elect_flag;
}
```

2.1.2 流程

选举线程根据选举角色和系统状态执行相关操作，主要包括选举、维持租约等，下面对不同选举角色下地执行过程进行具体描述。

选举角色为leader

检查自身租约情况，为避免系统出现双主，我们将正常租约过期点前一段时间的时间点设为leader的租约过期点（目前设置为200ms）。RootServer的当前时间超过该过期点，即认为租约过期，此时RootServer选举相关状态会重置为集群初启动时的值。如果租约没有过期，RootServer会在租约过期前2000ms内向其它RootServer发送续租请求。

续租时，RootServer首先检查本集群的UpdateServer是否运行正常，如果正常，则将新的租约（目前设置为5000ms）发送给所有备RootServer。备RootServer接收到续租请求后，也要在集群内UpdateServer正常情况下才同意续租请求，并更新租约。主RootServer发送续租请求后，收集回复信息，当多数RootServer同意时，主RootServer更新本身租约。

选举角色为candidate

RootServer是该角色时，正处于选举状态中。选举线程会检查选举是否超时，如果超时，RootServer会重置选举状态，选举角色也变为follower。下次租约过期时继续进行选举。

选举角色为follower

选举线程检查租约是否过期，如果过期，RootServer等待一个随机时间后（时间范围为300ms~800ms），发起选举。

选举包括两个阶段：投票阶段和广播阶段。在投票阶段中，主RootServer向其它RootServer发送投票请求，并把UpdateServer日志最大时间戳放入消息包中。当接收方发现投票请求包中的时间戳不小于本集群内的最大时间戳时，便同意投票请求，否则予以拒绝。如果收到多个RootServer的投票请求，则会向同意的第一个RootServer承诺不会再同意其它RootServer，保证仅投出一个赞成票，使得在投票阶段只有一个RootServer能够收集到多数赞同票。

若投票阶段中，发起投票的RootServer得到多数节点赞同，则可以当选为leader，即主节点。但该阶段中，RootServer仅对投票请求作出回应，并不会确认leader当选者，因此需要当选的RootServer进入广播阶段，向其它节点发送广播，通知当选消息，接收方收到该消息后将新主RootServer的信息保存下来，结束选举，开始作为备节点正常服务。而新的主RootServer收到多数节点对于当选广播的确认回复后，将选举角色变为leader，同时集群角色变为master，结束选举。

3 模块接口

3.1对外接口

RootServer可以调用ObCheckElection::get_election_state()函数判断是否处于选举过程。

```
/**
 * @brief get election state, including  INIT, DURING_ELECTION, AFT
ER_ELECTION
 * @return election state
 */
ObElectionRole::State get_election_state()
```

RootServer可以调用ObElectionNode::set_is_extendlease(bool flag)函数设置RootServer能否发送租约。

```
/**
 * @brief set the flag to allow lease extending open or closed
 * @param flag [in], true means open, false means closed
 */
ObElectionNode::set_is_extendlease(bool flag);
```

4 使用限制条件和注意事项

- 首次启动系统时，所有集群均作为备集群启动，用户需要使用命令设置主集群，具体命令为：

  > $rs_admin -r < master_rootserver_ip > -p < master_rootserver_port > set_obi_master_first

- 用户设置主集群后，需要等待备设置的集群切换为主集群，再进行boot_strap操作。用户可以使用命令查看集群的obi_role是否为MASTER：

```
$rs_admin -r < master_rootserver_ip > -p < master_rootserver_port > get_obi_role
```

- 用户只能够发起选举，并不能指定哪个RootServer成为新的主节点。
- 三集群架构中，如果只有一个集群出现异常，整个系统仍可以正常服务；一个以上集群异常，则系统将处于不可服务状态。
- 系统提供两个用户可配置的RootServer参数，具体如下：

参数名	默认值	参数描述
rs_election_random_wait_time	300ms	发起投票前需要等待的最少时间。参数值越大，选举时间越长
rs_election_protection_time_us	200ms	leader可以提前租约过期的时间。参数值越大，选举时的无主时间越长，而系统出现双主的可能性越小