# 棋牌游戏服务器架构设计分享



一、棋牌类服务器的特点 　　1、棋牌类不分区不分服 　　一般来说，棋牌游戏都是不分区不分服的。所以棋牌类服务器要满足随着用户量的增加而扩展的需要。 　　2、房间模式 　　即在同一局游戏中就是在同一个房间中，同一个房间中的人可以接收到其他人的消息。 　　3、每个房间的操作必须是顺序性 　　这个特性类似与一般游戏的回合制，每个玩家的操作都是有顺序性的。 　　二、需要解决的技术点 　　1、数据共享 　　因为棋牌类游戏不分区不分服，我们在设计服务器的时候，是按世界服的思想去设计，即服务器是一个n多台物理机的集群。当用户登陆服务器，创建房间时，可能根据负载均衡算法，它可以在任何一台服务器上面。所以，不管用户登陆到哪一台服务器上面了，都可以获得自己的数据。我们可以使用redis来做数据共享。 　　2、如何进入房间 　　在同一局游戏中，我们要求所有人都在同一个房间中，我们可以规定在同一个房间中的用户，必须登陆到同一台物理服务器上面。在创建房间完成之后，其他人根据房间号查找房间的时候，可以根据房间号，获取这个房间所在的服务器ip和端口，判断一个当前用户登陆的服务器ip与房间所在的服务器ip是否相同，如果相同，就不做切换，如果不一样，客户端就使用ip和端口，连接到房间所在的服务器上面。 　　3、保证房间操作的顺序性 　　创建房间成功之后，接下来的操作都要保证它的顺序性，所以房间需要有一个它自己的消息个队列。我们可以把每个房间到达服务器的消息封装为一个任务，把这个任务放到消息队列中，然后有一个任务执行者去按顺序执行这些任务。 　　三、系统架构 　　1、功能设计 　　登陆。一般都是需要接第三方登陆，登陆这一块是http操作，我们统一提供一个web服务，用来做登陆验证。因为在登陆时，调用第三方的http服务，这个过程可能很慢，如果放在逻辑服务器的话，可能会卡业务逻辑任务。因为可能不同的玩家业务请求可能同在一个线程中，如果有任务卡了，那么这个任务以后新来的请求请会卡住，导致消息延迟。 　　获取游戏公告，也放在web服务中。公告一般是游戏登陆的时候向服务器获取一次。把它放在web服务器中，与业务逻辑分离的好处是，当业务逻辑服务器维护或更新的时候，不影响用户的登陆，和获取公告，这样用户体验会好一些。 　　创建用户唯一的id，因为棋牌类游戏服务器是世界服，无分区，所以用户的id必须是全局唯一的。可以利用redis的incr方法，原子的递增，如果不想被别人根据userid的递增推算出有多少注册用户，递增的梯度可以随机，比如每次递增的值从1到1024中随机一个。 　　创建房间，当房间主创建房间时，房间的id需要在任何台服务器上可以查询到，所以创建房间成功后，房间id要存储在共享内存redis中，每个房间id对应一个房间所在的ip地址或服务器id.这样，当有用户要进入房间，在查询房间id时，可能判断这个房间是否和自己登陆的游戏服务器相同。 　　查找加入房间。根据房间id查询房间，查找到房间后，获取房间所在的ip地址或服务器id,如果发现和自己所登陆的服务器一样，直接可以加入房间。如果不一样，把这个房间所在的ip和端口返回给客户端，让客户端重新与房间所在的服务器建立连接，使用登陆时的token验证用户。 　　游戏脚本调用。在验证游戏是否合法时，客户端与服务器都要验证，验证的算法是一样的，所以可以使用脚本来写，写一份脚本，在服务器与客户端中同时使用。可以使用lua。同一个算法使用同一个脚本 ，这样在开发新的同类型棋牌游戏时，只需要替换一下这个脚本就行了，不用再重复开发。 　　2、后台管理系统 　　这个一般是根据运营需求开发的，每个公司不一样。不过有一点，后台管理系统可能要和游戏服务器通信，这种通信方式最好是采用redis的订阅/发布机制。这样可以把某个消息事件同时发送到所有的业务服务器上面。根据用户所在的服务器进行处理。 　　3、玩家同屏 　　玩家同屏是棋牌游戏中的一个重点，对于做过那些大型的arpg，或mmo游戏的程序员来说，这并不是什么难事。因为同屏就是服务器对客户端的消息进行转发。一个房间四个人，一个人出的牌或操作能被其他三个人同时看到。 　　因为棋牌游戏的同步数据量比较小。一般常见的同步方式有两种： 　　1、客户端主动拉取 　　客户端定时主动向服务器请求一个用户的消息队列，当一个玩家有操作需要同步到其他玩家时，在服务器端先把这个消息放到这个用户的消息队列中。等待客户端的拉取操作。这种方式的好处是，不需要考虑网络闪断或网络不好的情况，信息都是同步获取的。缺点是，定时拉取的时间间隔很短，可能不到一秒就会拉取一次。 　　2、服务器主动推送 　　当一个用户出牌的消息需要同步给其他玩家时，服务器会获得这个玩家与服务器建立的socket连接，然后服务器使用socket 主动向客户端发送消息。 　　这种方式要考虑网络闪断，消息丢失的问题。因为服务器推送的消息，客户端有可能会收不到。所以客户端需要根据心跳来判断网络是否有断开过，如果有断开，需要重新从服务器拉取整个房间状态的消息。或者根据服务器发送的消息号，如果客户端发现接收到的服务器消息号有跳号的，比如应该接收10，却收到了12，说明中间有消息丢失，需要重新拉取整个房间的状态信息。 　　这种方式的缺点是，开发复杂，需要考虑一些网络问题。优点是，只有在有消息的时候才会推送，没有的话不推送，不占用带宽等系统资源，可以增加用户同时在线量，也就是增加了服务器的承载量。 　　4、数据同步和持久化 　　1、由于棋牌类的游戏数据少，计算量也小，所以完全可以不使用内存缓存，而直接使用redis共享内存，用户的所有数据都缓存在redis中。更新也同步更新到redis中，这样不管一个用户登陆哪一台业务服务器，都能获得自己的最新数据。 　　2、更新数据库，由于数据第一缓存是redis，所以活跃的用户数据都是可以从redis中直接获得的，而不用查询数据库，所以数据库的更新可以采取异步更新，而不会产会数据的延迟。需要注意的一点是，数据的异步更新必须保证是有顺序的。那么这就会产生一个问题，怎么保证用户的更新不会乱呢? 　　3、如何保证更新的顺序性 　　因为我们的业务服务器是多个的，用户可能连接其中的任何一个，如果说登陆的是服务器A,加入的房间在服务器B上，那么连接就会切换。为了保证数据更新的顺序，我们可以做一个数据库持久化服务，把需要更新数据库的任务实时发送到这台服务器上，由数据库持久化服务执行对数据库的更新。这样不管用户连接的哪台业务服务器，它的更新都是有顺序保证的。 　　4、一种快速简单的方法 　　由于棋牌类的业务少，数据更新少，所以查询可以有redis缓存，减少数据库查询的压力，而更新实行实时更新到数据库，前期不需要开发数据库持久化服务。等用户积累到一定程序之后，发现更新数据库比较慢的时候，再单独做一个数据库持久化服务。 　　四、服务器架构 　　1、登陆时，客户端首先向登陆的web服务器请求登陆信息，登陆成功之后，返回登陆的token,为了适应大规模的web请求和登陆服务的稳定，可以使用nginx做负载均衡。 　　2、登陆成功之后，请求负载均衡服务器，获取一台连接的业务服务器。这个负载均衡服务器可以和登陆web在一个进程中，也可以独立出来。 　　3、拿到登陆成功的token和需要连接的业务服务器的ip和端口之后，再去连接业务服务器。连接成功之后，要使用token到登陆服务器去验证，这个用户是否登陆了。 　　4、同一个房间的用户要连接到同一台物理服务器上面。在上面已经说过了。 　　5、redis用来做共享缓存。 　　6、mysql做持久化存储。 　　7、数据库持久化服务器，统一做数据入库操作。 　　五、关于网关的问题 　　1、网关的作用 　　转发消息包 　　业务的负载均衡，比如A业务由服务器a处理，B业务由服务器b处理，由网关进行转发。 　　维护与客户端的连接 　　带宽的整合，一般的云服务都是按购买的服务器计算带宽的。通过一台服务器转发消息，可以只购买一个大带宽就可以了，以节约成本。 　　2、棋牌类游戏需要网关吗? 　　我认为不太需要，因为棋牌类游戏业务比较单一，做的最多的就是消息同屏转发。最多是再有一些任务或活动，这些由一台服务器直接处理完全可以搞定。而且开发网关也是一个复杂的工作，没必要在这个上面花太多的时间。