

1.mongodb是一个分布式数据库，在集群模式下，其一般结构如图所示。数据存在在分片中，图上有三个分片，分别是shard1,shard2,shard3。为了高可用性，使用了副本集，一个shard有三个mongod组成，即一个master，2个slave。客户端连接mongos，其相当于一个代理，隐去了集群中的操作。mongos主要实现路由功能，比如我们查询db.role.find({roleid:id})根据roleid来路由到那个分片，假设是shard1，那么mongos会向shard1发送请求，获得返回后,mongos整理结果返回客户端。configserver是配置服务器，记录了集群中分片的信息，mongos本身不具有持久化能力，其路由信息均从configserver获取。

上面说一下最重要的片键。如何将数据放到存储到哪个分片上，mongodb使用了片键来确定存储到哪个分片上。假设我们玩家ID是从1增长的，那么我们选取片键是ID，min代表无限小的数，max代表无限大的数据，假设min-30分到shard1,31-60分到shard2，61-max分到shard3,那么如果插入一个{ID:61,name:"李磊"}的文档，就会被分到shard3分片去。如果61-max的分片越来越大，那么mongos会变成shar3的路由信息变成61-90,91-max，拆分开来。如果某个分片上的数据太多，那么会有一个均衡器来迁移分片数据到另一个分片，保持负载均衡。现在我们倒退一下，假设初始就是min-max，玩家ID是从1开始增长，第一个分片分到shard1上去，然后不停插入文档，shard1上，然后再不停被拆分，迁移，导致性能的下降。这种片键被称为升序片键。

如果我们使用随机数作为片键呢？插入一个{ID:random,name:"李雷"，)}的文档，以key作为片键，按照上面的分片规则，那么他就会被随机分配到shard中，达到负载均衡，看着挺好的，但是如果我们查询ID<100的玩家的时候，mongos就可能把请求发到所有的分片中，这样会增加延迟。

还有一种，我们要先说明一下，如果片键的值相同，那么都会被分在同一个分片中，假设1亿个ID=100的玩家，他们只能分到一个分片中，即便其他分片的负载很低。假设数据有个位置信息，比如{ID:1,name:"李雷", addr:"广州"}，我们以addr作为片键，那么如果addr的是广州的都会被分到同一个分片，那么当我们查询addr="广州"的玩家的时候，性能就非常高。

还有一种，是使用复合片键，简单地说就是有多个key主程的片键。假设广州，北京的可能有好几百万的数据，但是南京只有1万，不希望放到一个分片上，也不希望放到所有分片上。那么设置了片键为{key,addr}，{ID:1,name:"李雷", addr:"广州"，key=random(2)}，这里key的范围是1~2，由于key在前面，所以所有文档只能被分到1,2两个key指定的分片上，假设被分到shard1,shard2,不会被分配到shard3上去。并且同一个分片内，会以addr进行排序，查找addr=广州的，只需要发到shard1,shard2上去，并且由于以addr排序，所以一个分片内查找addr=广州的非常快。

从上面看，对于片键的选择和重要，他直接影响数据的分布和效率。

好了，我们现在来考虑世界服框架直接使用mongodb集群和多台单机的会遇到什么问题。

如果是单个数据库的mongodb集群

1.选择片键，假设使用[随机数处理,roleid]，这个只是将玩家数据分散到所有分片中，并且片键选择后就基本不能修改，如果后面想使用[随机数,roleid,name]，提升使用name来索引玩家的时候，就不行了，必须在设计之初确定好片键

2.如果某个分片宕机，那么集群功能就不用，为了高可用，每个分片必须配置副本集，来备份数据，并且支持选举，而且为了达到数据不丢失，写入数据时，处了主节点，还需要同步过半的备份节点，这样会增加响应的时间。

3. 对于扩容和减低配置，可以通过增加分片和删除分片来处理，这个可以线上处理。但是配置和维护集群成本比较高。

4. 可以通过部署多个mongos来提高并发

如果使用多个mongodb数据库来创建的话

1.需要自己设定路由，通过router(roleid)=db\_name来获取数据，全局数据也通过类似的路由grouter(global\_name) = db\_name。游戏服可能会连接到所有的数据库。

2.创建角色时，需要选定db后，再创建角色。

3.扩容可以线上进行，减低配置的话，需要通过合拼数据库，要停止部分数据库。

4.进行某种范围查询时候，需要连接到所有数据库，比如查询所有角色的名字，就需要对所有数据库进行查询。

从上面来看，其各有千秋，如果离开其使用场景对比的话就是耍流氓。从游戏的特点来说，其读的追求大于写，而片键的选取又不能发挥作用，由于集群数据状态时刻在改变，很难有一份完美的一致的集群数据库快照。使用集群，对于游戏来说，最大的问题是运维成本。而对于多个数据库实例，需要在业务框架上提供对应路由功能，这个功能也不算难。而由于游戏也会伴随停服更新，所以合拼数据库这种也没什么大事情。而对于全角色查询这种操作，就不好处理，只能由具体玩法处理。不过其优点就是运维成本低，而且理解成本也不会太高。如果说有技术挑战性的话，集群模式会更有趣，但是如果按照合适原则，多数据库实例会更适合，而且成本更低。