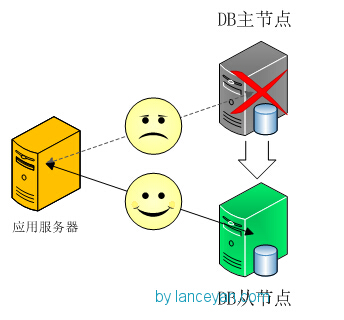
MongoDB集群部署

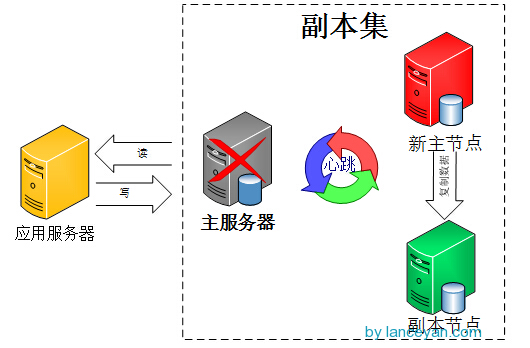
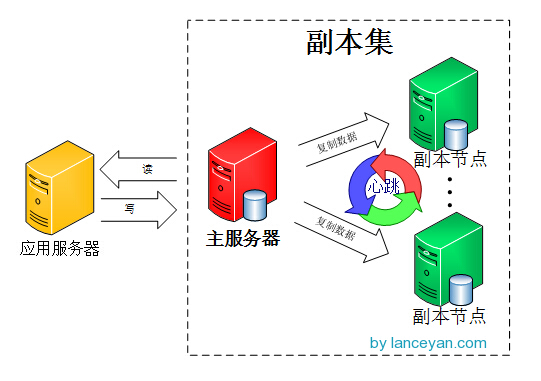
1. 主从复制
   1. 模型图



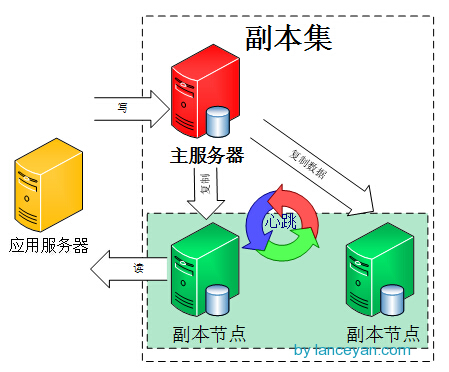
* 1. 架构优势：
     1. 数据备份，主节点数据自动同步到副节点，当主节点挂掉，可以切换到副节点；
  2. 架构缺点：
     1. 当主节点挂掉，需要手动切换；
     2. 副节点不支持读写入操作，造成主节点读写压力过大；
     3. 每个副节点的数据都是对数据库全量拷贝；
  3. 总结：

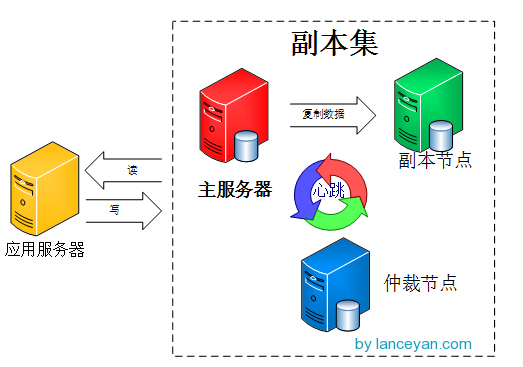
所以这种简单的模式无法做的高扩展性，高性能，需要引用副本集和分片的功能。

1. 副本集
   1. 模型图：



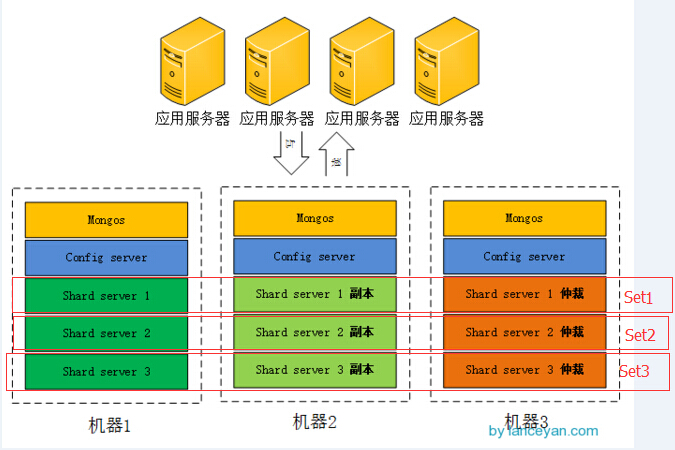
读写分离：





仲裁节点：不存储数据，占用资源少，只是负责故障转移的群体投票，即当主节点挂掉后，负责选出新的主节点：

1. 得到每个服务器节点的最后操作时间戳。每个mongodb都有oplog机制会记录本机的操作，方便和主服务器进行对比数据是否同步还可以用于错误恢复；
2. 如果集群中大部分服务器都挂掉了，保留活着的节点都为 secondary状态并停止选举；
3. 如果集群中选举出来的主节点或者所有从节点最后一次同步时间太旧，停止选举等待人为操作；
4. 如果上面条件都没问题，则选择操作时间戳最新（保证数据是最新的）的服务器节点作为主节点；
   1. 架构优势：
      1. 故障转移：实现了主节点挂掉，集群内会自动切换主节点，主服务器负责整个副本集的读写，副本集定期同步数据备份，一但主节点挂掉，副节点就会选举一个新的主服务器；
      2. 数据备份；
      3. 读写分离：减轻了主节点读写压力过大的问题；
   2. 架构缺点：
      1. 每个副节点的数据都是对数据库全量拷贝，副节点复制压力过大；
      2. 数据压力大时，无法扩展；
5. 分片+副本集
   1. 模型图



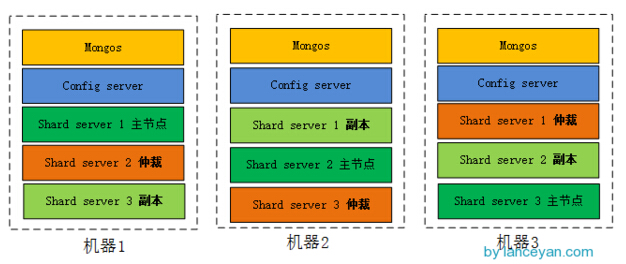
**mongos**：数据库集群请求的入口，所有的请求都通过mongos进行协调，不需要在应用程序添加一个路由选择器，mongos自己就是一个请求分发中心，它负责把对应的数据请求请求转发到对应的shard服务器上。在生产环境通常有多mongos作为请求的入口，防止其中一个挂掉所有的mongodb请求都没有办法操作；

**config Server**：配置服务器，存储所有数据库元信息（路由、分片）的配置。mongos本身没有物理存储分片服务器和数据路由信息，只是缓存在内存里，配置服务器则实际存储这些数据。mongos第一次启动或者关掉重启就会从 config server 加载配置信息，如果配置服务器信息变化则通知到所有的 mongos 更新自己的状态，这样 mongos 就能继续准确路由。在生产环境通常有多个 config server 配置服务器；

**shard**：分片，将一个数据表逻辑上分为多个分片处理；

**replica Set**：副本集，对其中一个分片配置副本集；

**优化**：仲裁节点不进行数据读写操作，将仲裁节点配置在各个服务器，分担服务器压力；



* 1. 架构优势：
     1. 对数据分片存储，解决副节点复制压力过大问题；
     2. 当数据压力大时，可扩展；
  2. 架构缺点：
     1. 集群负载均衡需要搭配haproxy分担服务器处理请求的压力；