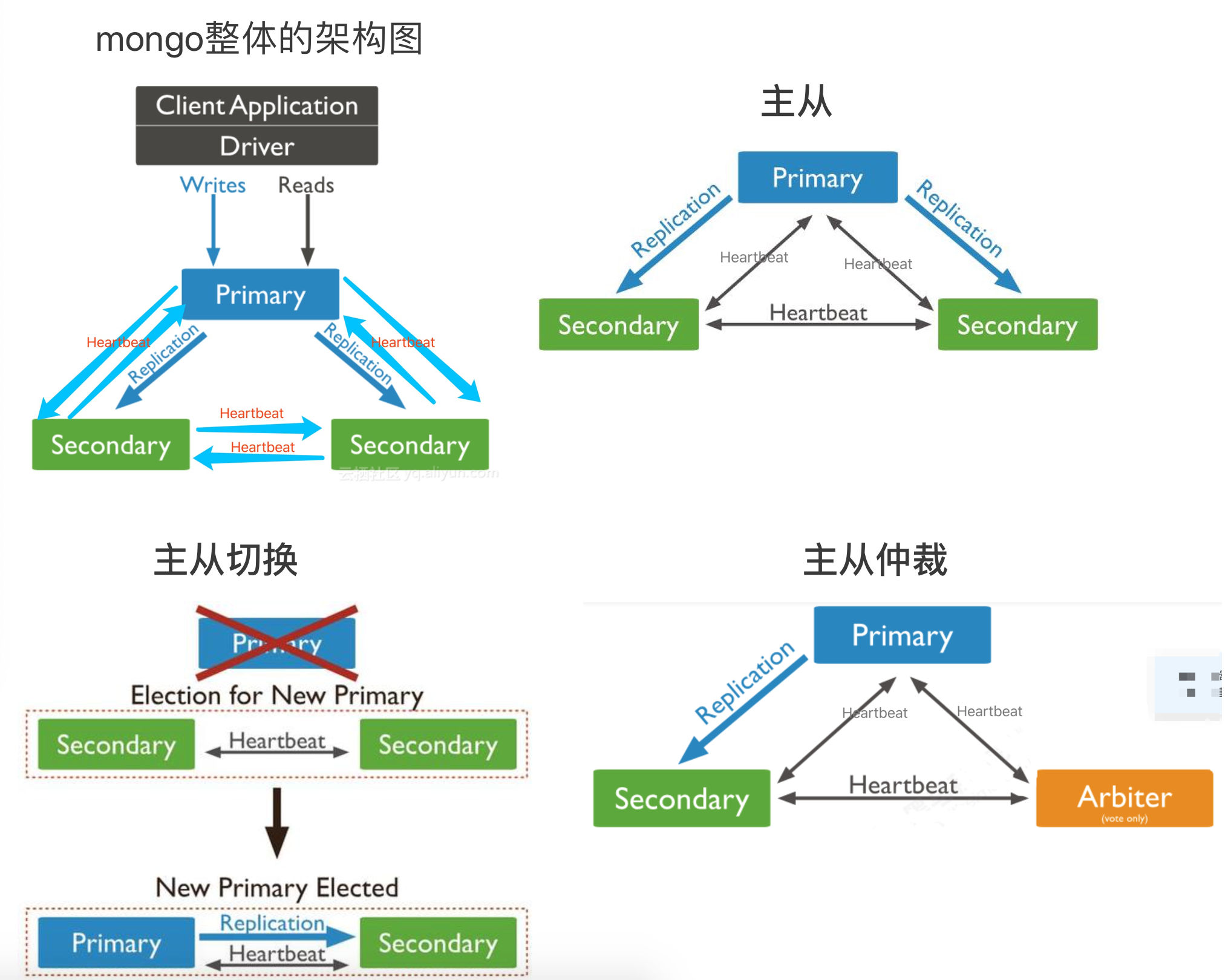
复制集原理https://developer.aliyun.com/article/64

**复制集介绍**

Mongodb复制集由一组Mongod实例（进程）组成，包含一个Primary节点和多个Secondary节点，Mongodb Driver（客户端）的所有数据都写入Primary，Secondary从Primary同步写入的数据，以保持复制集内所有成员存储相同的数据集，提供数据的高可用。典型的Mongdb复制集，包含一个Primary节点和2个Secondary节点。



**Primary选举**

**原理**

复制集通过replSetInitiate命令（或mongo shell的rs.initiate()）进行初始化，初始化后各个成员间开始发送心跳消息，并发起Priamry选举操作，获得『大多数』成员投票支持的节点，会成为Primary，其余节点成为Secondary。

初始化复制集

config = {
\_id : "my\_replica\_set",
members : [
{\_id : 0, host : "rs1.example.net:27017"},
{\_id : 1, host : "rs2.example.net:27017"},
{\_id : 2, host : "rs3.example.net:27017"},
]
}
rs.initiate(config)

『大多数』的定义

假设复制集内投票成员（后续介绍）数量为N，则大多数为 N/2 + 1，当复制集内存活成员数量不足大多数时，整个复制集将无法选举出Primary，复制集将无法提供写服务，处于只读状态。

通常建议将复制集成员数量设置为奇数，从上表可以看出3个节点和4个节点的复制集都只能容忍1个节点失效，从『服务可用性』的角度看，其效果是一样的。（但无疑4个节点能提供更可靠的数据存储）

**特殊的Secondary**

Secondary

正常情况下，复制集的Seconary会参与Primary选举（自身也可能会被选为Primary），并从Primary同步最新写入的数据，以保证与Primary存储相同的数据。

Secondary可以提供读服务，增加Secondary节点可以提供复制集的读服务能力，同时提升复制集的可用性。另外，Mongodb支持对复制集的Secondary节点进行灵活的配置，以适应多种场景的需求。

Arbiter

Arbiter节点只参与投票，不能被选为Primary，并且不从Primary同步数据。

比如你部署了一个2个节点的复制集，1个Primary，1个Secondary，任意节点宕机，复制集将不能提供服务了（无法选出Primary），这时可以给复制集添加一个Arbiter节点，即使有节点宕机，仍能选出Primary。

Arbiter本身不存储数据，是非常轻量级的服务，当复制集成员为偶数时，最好加入一个Arbiter节点，以提升复制集可用性。

Priority0

Priority0节点的选举优先级为0，不会被选举为Primary

比如你跨机房A、B部署了一个复制集，并且想指定Primary必须在A机房，这时可以将B机房的复制集成员Priority设置为0，这样Primary就一定会是A机房的成员。（注意：如果这样部署，最好将『大多数』节点部署在A机房，否则网络分区时可能无法选出Primary）

Vote0

Mongodb 3.0里，复制集成员最多50个，参与Primary选举投票的成员最多7个，其他成员（Vote0）的vote属性必须设置为0，即不参与投票。

Hidden

Hidden节点不能被选为主（Priority为0），并且对Driver不可见。

因Hidden节点不会接受Driver的请求，可使用Hidden节点做一些数据备份、离线计算的任务，不会影响复制集的服务。

Delayed

Delayed节点必须是Hidden节点，并且其数据落后与Primary一段时间（可配置，比如1个小时）。

因Delayed节点的数据比Primary落后一段时间，当错误或者无效的数据写入Primary时，可通过Delayed节点的数据来恢复到之前的时间点。

**产生选举的场景**

Primary选举除了在复制集初始化时发生，还有如下场景

复制集被reconfig

Secondary节点检测到Primary宕机时，会触发新Primary的选举

当有Primary节点主动stepDown（主动降级为Secondary）时，也会触发新的Primary选举

Primary的选举受节点间心跳、优先级、最新的oplog时间等多种因素影响。

节点间心跳

复制集成员间默认每2s会发送一次心跳信息，如果10s未收到某个节点的心跳，则认为该节点已宕机；如果宕机的节点为Primary，Secondary（前提是可被选为Primary）会发起新的Primary选举。

节点优先级

每个节点都倾向于投票给优先级最高的节点

优先级为0的节点不会主动发起Primary选举

当Primary发现有优先级更高Secondary，并且该Secondary的数据落后在10s内，则Primary会主动降级，让优先级更高的Secondary有成为Primary的机会。

Optime

拥有最新optime（最近一条oplog的时间戳）的节点才能被选为主。

网络分区

只有更大多数投票节点间保持网络连通，才有机会被选Primary；如果Primary与大多数的节点断开连接，Primary会主动降级为Secondary。当发生网络分区时，可能在短时间内出现多个Primary，故Driver在写入时，最好设置『大多数成功』的策略，这样即使出现多个Primary，也只有一个Primary能成功写入大多数。

**数据同步**

Primary与Secondary之间通过oplog来同步数据，Primary上的写操作完成后，会向特殊的[local.oplog.rs](https://docs.mongodb.org/manual/reference/local-database/#local.oplog.rs)特殊集合写入一条oplog，Secondary不断的从Primary取新的oplog并应用。

因oplog的数据会不断增加，local.oplog.rs被设置成为一个[capped集合](https://docs.mongodb.org/manual/core/capped-collections/)，当容量达到配置上限时，会将最旧的数据删除掉。另外考虑到oplog在Secondary上可能重复应用，oplog必须具有幂等性，即重复应用也会得到相同的结果。