**ZooKeeper**

**CAP定理：**

一个分布式系统不可能在满足分区容错性（P）的情况下同时满足一致性（C）和可用性（A）。在此ZooKeeper保证的是CP，ZooKeeper不能保证每次服务请求的可用性，在极端环境下，ZooKeeper可能会丢弃一些请求，消费者程序需要重新请求才能获得结果。另外在进行leader选举时集群都是不可用，所以说，ZooKeeper不能保证服务可用性。

**BASE理论**

BASE理论是基本可用，软状态，最终一致性三个短语的缩写。BASE理论是对CAP中一致性和可用性（CA）权衡的结果，其来源于对大规模互联网系统分布式实践的总结，是基于CAP定理逐步演化而来的，它大大降低了我们对系统的要求。

1. 基本可用：基本可用是指分布式系统在出现不可预知故障的时候，允许损失部分可用性。但是，这绝不等价于系统不可用。比如正常情况下，一个在线搜索引擎需要在0.5秒之内返回给用户相应的查询结果，但由于出现故障，查询结果的响应时间增加了1~2秒。
2. 软状态：软状态指允许系统中的数据存在中间状态，并认为该中间状态的存在不会影响系统的整体可用性，即允许系统在不同节点的数据副本之间进行数据同步的过程存在延时。
3. 最终一致性：最终一致性强调的是系统中所有的数据副本，在经过一段时间的同步后，最终能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。

**ZooKeeper特点**

1. 顺序一致性：同一客户端发起的事务请求，最终将会严格地按照顺序被应用到 ZooKeeper 中去。
2. 原子性：所有事务请求的处理结果在整个集群中所有机器上的应用情况是一致的，也就是说，要么整个集群中所有的机器都成功应用了某一个事务，要么都没有应用。
3. 单一系统映像：无论客户端连到哪一个 ZooKeeper 服务器上，其看到的服务端数据模型都是一致的。
4. 可靠性：一旦一次更改请求被应用，更改的结果就会被持久化，直到被下一次更改覆盖。

**ZAB协议：**

ZAB协议包括两种基本的模式：崩溃恢复和消息广播。当整个 Zookeeper 集群刚刚启动或者Leader服务器宕机、重启或者网络故障导致不存在过半的服务器与 Leader 服务器保持正常通信时，所有服务器进入崩溃恢复模式，首先选举产生新的 Leader 服务器，然后集群中 Follower 服务器开始与新的 Leader 服务器进行数据同步。当集群中超过半数机器与该 Leader 服务器完成数据同步之后，退出恢复模式进入消息广播模式，Leader 服务器开始接收客户端的事务请求生成事物提案（超过半数同意）来进行事务请求处理。

**选举算法和流程：FastLeaderElection(默认提供的选举算法)**

目前有5台服务器，每台服务器均没有数据，它们的编号分别是1,2,3,4,5,按编号依次启动，它们的选择举过程如下：

1. 服务器1启动，给自己投票，然后发投票信息，由于其它机器还没有启动所以它收不到反馈信息，服务器1的状态一直属于Looking。
2. 服务器2启动，给自己投票，同时与之前启动的服务器1交换结果，由于服务器2的编号大所以服务器2胜出，但此时投票数没有大于半数，所以两个服务器的状态依然是LOOKING。
3. 服务器3启动，给自己投票，同时与之前启动的服务器1,2交换信息，由于服务器3的编号最大所以服务器3胜出，此时投票数正好大于半数，所以服务器3成为leader，服务器1,2成为follower。
4. 服务器4启动，给自己投票，同时与之前启动的服务器1,2,3交换信息，尽管服务器4的编号大，但之前服务器3已经胜出，所以服务器4只能成为follower。
5. 服务器5启动，后面的逻辑同服务器4成为follower。

**zk中的监控原理**

zk类似于linux中的目录节点树方式的数据存储，即分层命名空间，zk并不是专门存储数据的，它的作用是主要是维护和监控存储数据的状态变化，通过监控这些数据状态的变化，从而可以达到基于数据的集群管理，zk中的节点的数据上限时1M。

client端会对某个znode建立一个watcher事件，当该znode发生变化时，这些client会收到zk的通知，然后client可以根据znode变化来做出业务上的改变等。