ZooKeeper集群分布式协调服务

www.huawei.com

HUAWEI

目标

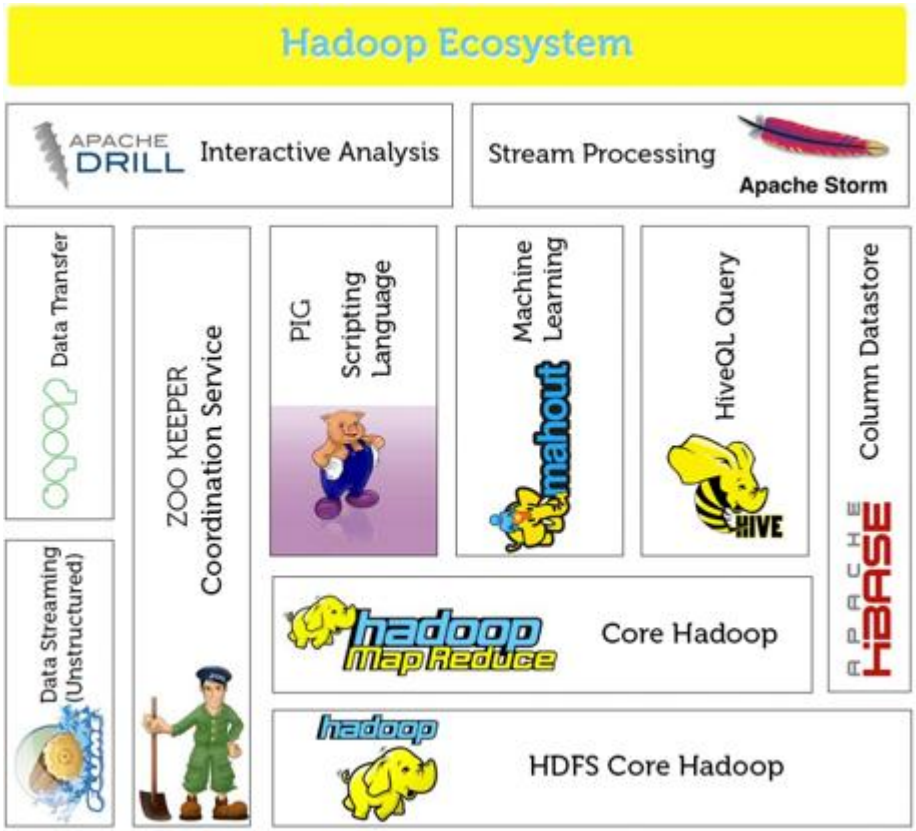
- 学完本课程后，您将能够:
  - 掌握**ZooKeeper**概念
  - 掌握**ZooKeeper**系统架构
  - 掌握**ZooKeeper**关键特性

目录

Zookeeper定义

- **Zookeeper** 分布式服务框架主要是用来解决分布式应用中经常遇到的一些数据管理问题，提供分布式、高可用性的协调服务能力,在**FusionInsight**集群中主要用途是保存上层组件的元数据，并保证其主备倒换。

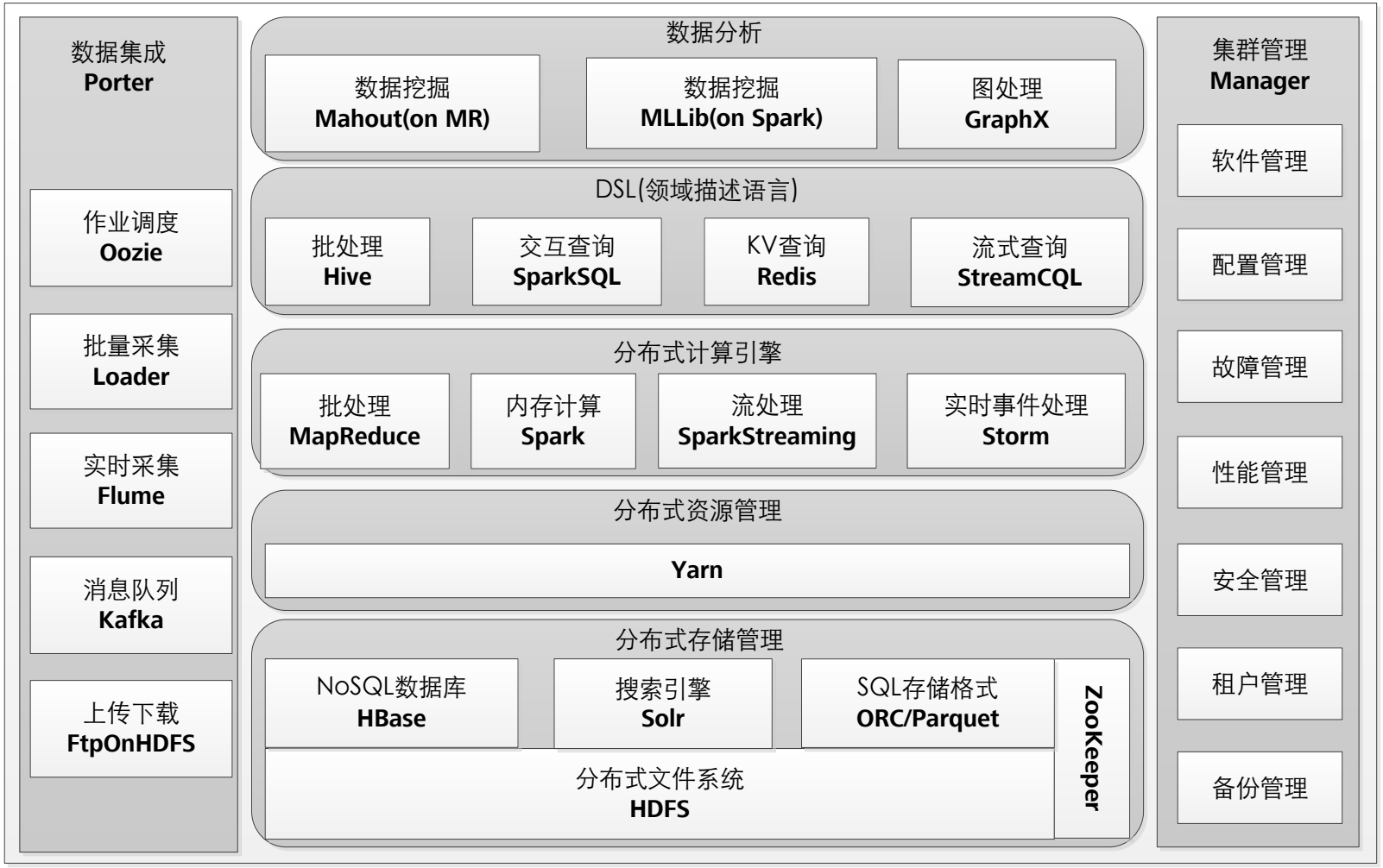
- 安全模式下**ZooKeeper**依赖于**LdapServer**和**Kerberos**。



HUAWEI

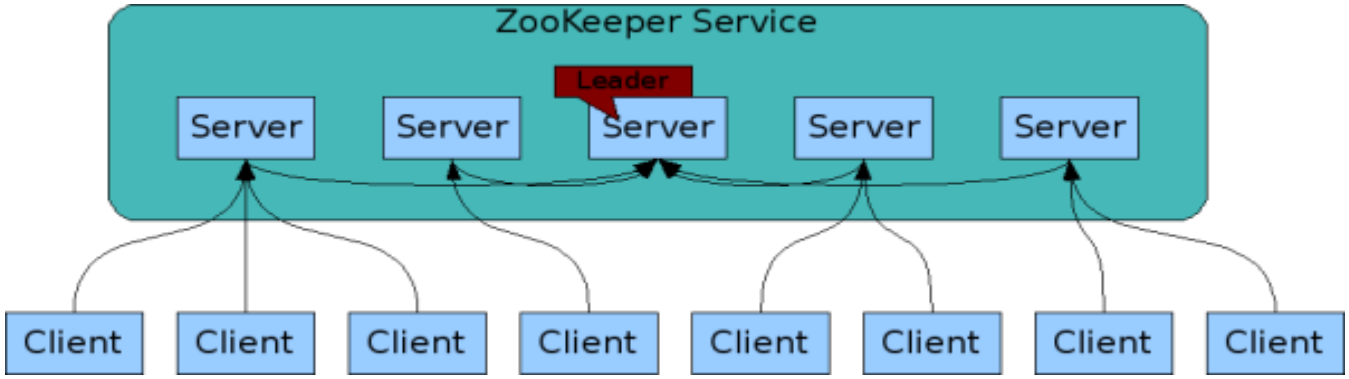
目录

ZooKeeper在产品的位置



目录

ZooKeeper服务架构--模型



1.分布式地分布在一组机器中

2.所有节点存储整份数据（在内存也在硬盘）

3.在启动时候选举出**Leader**

4.Leader会做原子广播到所有其他节点

5.严格的顺序访问控制

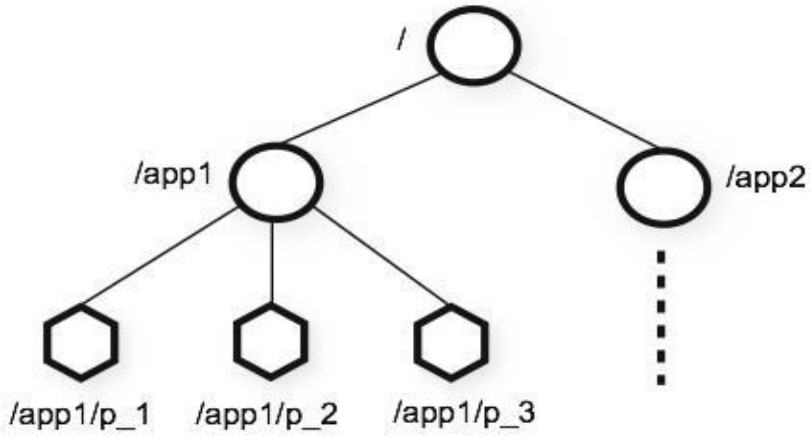
6.不会部分读写

ZooKeeper服务架构--容灾能力

- 一般情况下，**ZooKeeper**能够完成选举即能够正常对外提供服务。**ZooKeeper**选举时，当某一个实例获得了半数以上的票数时，则变为**leader**。
- 对于**n**个实例的服务，**n**可能为奇数或偶数
  - **n**为奇数时，假定**n=2x+1**，则成为**leader**的节点需获得**x+1**票，容灾能力为**x**
  - **n**为偶数时，假定**n=2x+2**，则成为**leader**的节点需要获得**x+2**票（大于一半），容灾能力为**x**。

由此可见，**2x+1**个节点与**2x+2**个节点的容灾能力相同（**3**个与**4**个相同，**5**个与**6**个相同...），而考虑到选举以及完成写操作的速度与节点数的相关性，我们建议**ZooKeeper**部署奇数个节点。

ZooKeeper数据模型



1. 分层命名空间

2. 每个命名空间的节点都叫做“**znode**”

3. 每个**znode**被路径区分（如：**/app1**）

4. **znode**节点类型– 永久节点和临时节点

5.临时节点不能有子节点

6.每个**znode**节点有数据，也可以选择有子节点

7. **znode**不能被重命名

8. 可以给**znode**增加或者删除**watchers**

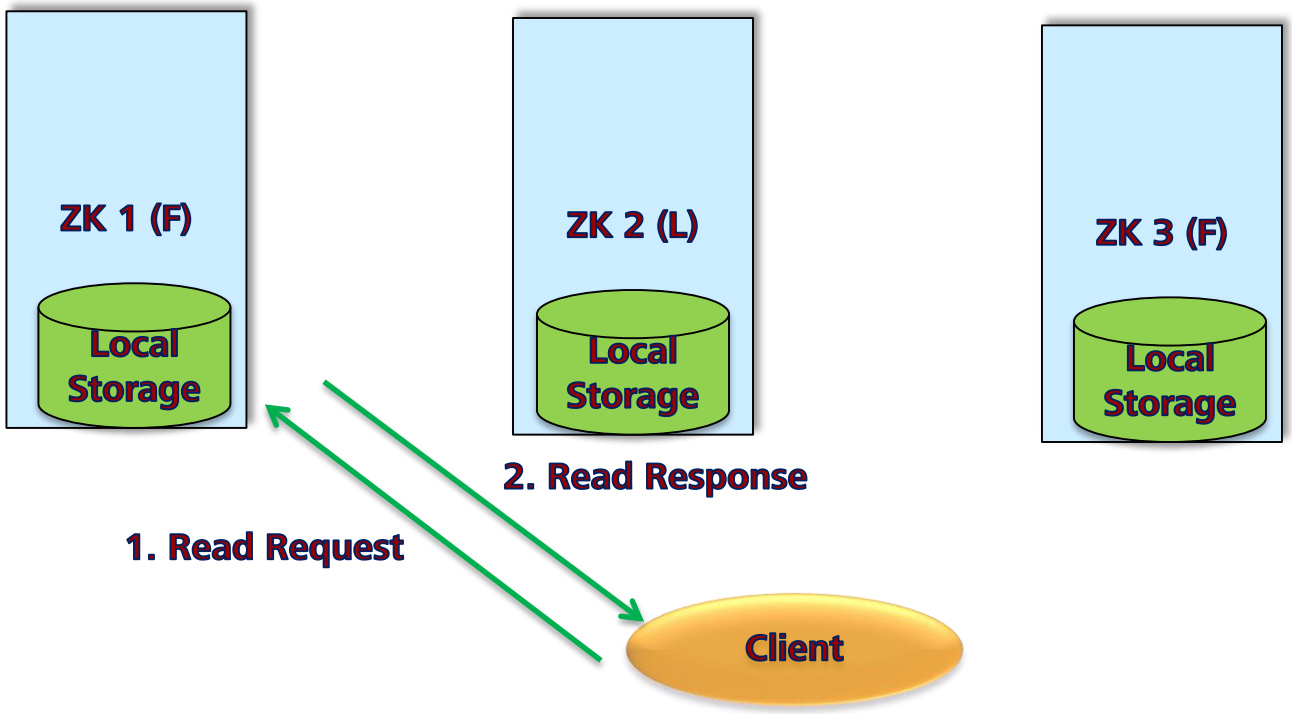
目录

ZooKeeper关键特性

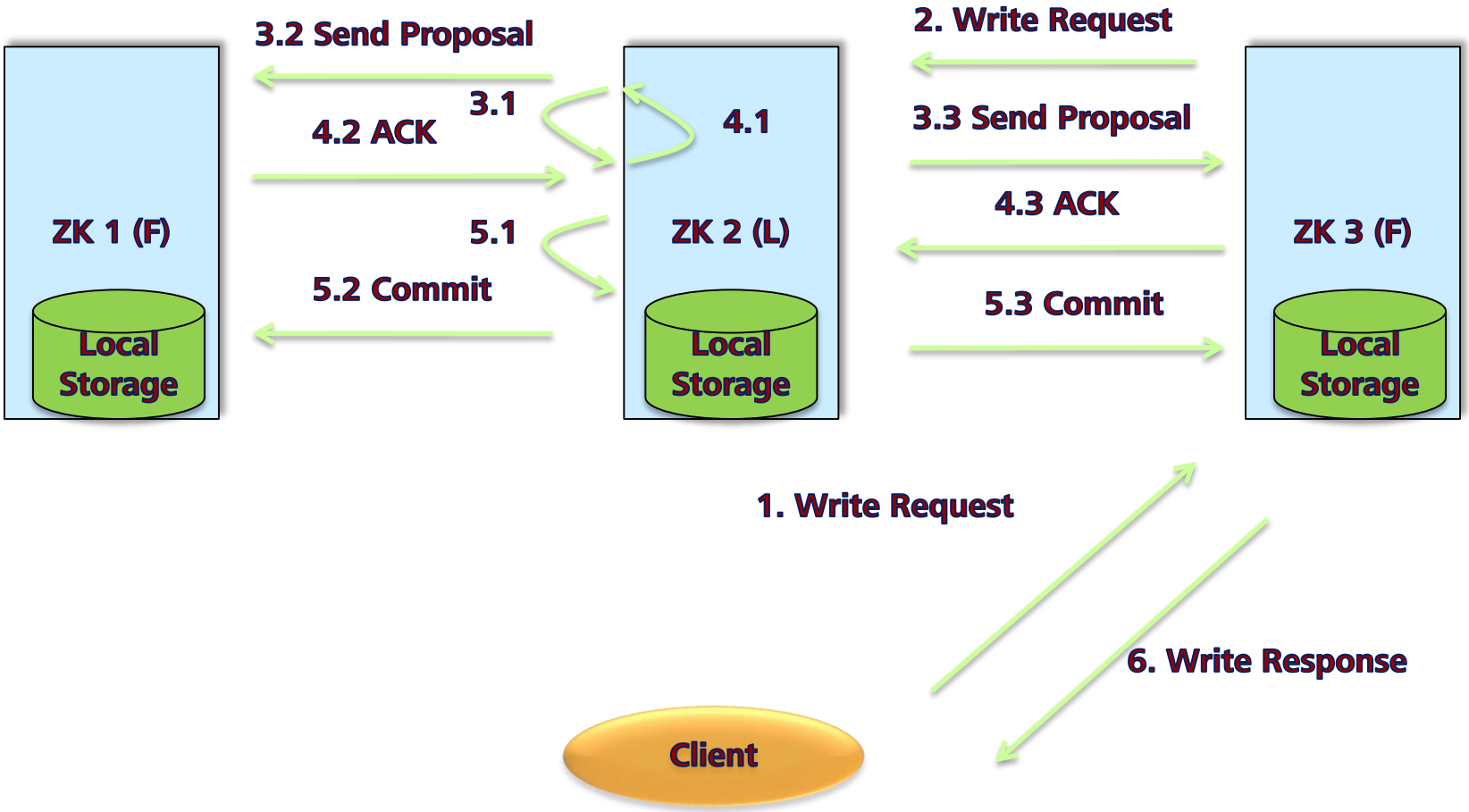
- 最终一致性：无论哪个**server**，对外展示的均是同一个视图。
- 实时性：保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息。
- 可靠性：一条消息被一个**server**接收，它将被所有**server**接受。
- 等待无关性：慢的或者失效的**client**不得干预快速的**client**的请求，使得每个**client**都能有效的等待。
- 原子性：更新只能成功或者失败，没有中间状态。
- 顺序性：客户端的更新顺序与它们被发送的顺序相一致。

ZooKeeper读特性

由**ZooKeeper**的一致性可知，客户端无论连接哪个**server**，获取的均是同一个视图。所以，读操作可以在客户端与任意节点间完成。



ZooKeeper写特性



HUAWEI

ACL（访问控制列表）

- **ACL**可以控制访问**ZooKeeper**的节点，只能应用于特定的**znode**上，而不能应用于该**znode**的所有子节点上。设置**ACL**命令为 **setAcl /znode scheme:id:perm**
- **Scheme**为认证方式， **ZooKeeper**内置了**4**种方式：
  - **world** 一个单独的**ID**，表示任何人都可以访问
  - **auth** 不使用**ID**，只有认证的用户可以访问
  - **digest** 使用**username:password**生成**MD5**哈希值作为认证**ID**
  - **IP** 使用客户端主机**IP**地址来进行认证
- **Id**：用来认证的字段，用来判断认证信息是否合法，不同的**scheme**的认证方式不同。
- **Perm**：即**permission**，通过**Acl**认证的用户对该节点可拥有的操作权限

日志增强

- **Ephemeral node**（临时节点）在**session**过期之后就会被系统删除，在审计日志中添加**Ephemeral node**被删除的审计日志，以便了解当时**Ephemeral node**的状态信息。

ZooKeeper客户端常用命令使用

调用**ZooKeeper**客户端，执行命令：

```
sh zkCli.sh -server 172.0.0.1:24002
```

执行创建节点，设置**ACL**，查询**ACL**，查询节点值，删除节点操作。

创建节点：`create /node`

获取节点子节点：`ls /node`

设置**Acl**：`setAcl /node sasl:admin@HADOOP.COM:cdrwa`

查询节点**Acl**：`getAcl /node`

创建节点数据：`set /node data`

获取节点数据：`get /node`

删除节点：`delete /node`

删除节点及所有子节点：`deleteall /node`

目录

ZooKeeper和Storm Nimbus HA的配合关系

Storm Nimbus利用**zookeeper**来选主。

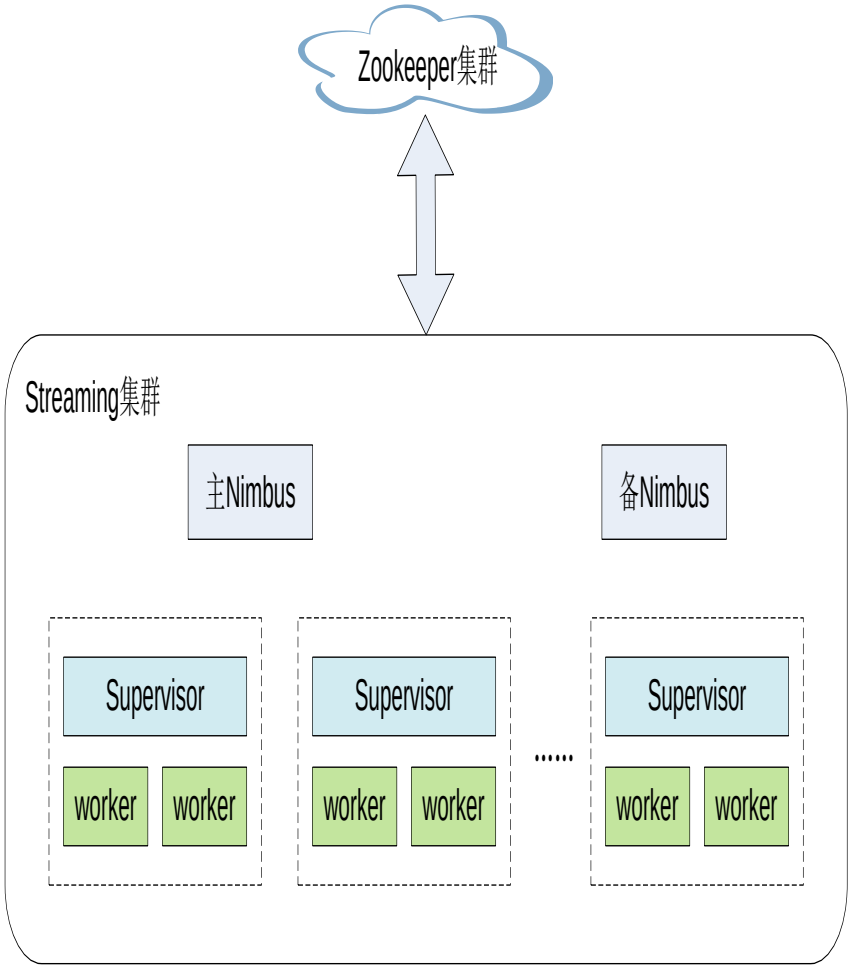
ZooKeeper提供了以下两种能力：

- 分布式锁服务

多个**Nimbus**进程都尝试着去**ZooKeeper**中写入一个对应的节点，该节点只能被一个**Nimbus**进程创建成功，创建成功的**Nimbus**进程成为主**Nimbus**。

- 事件侦听机制**--watcher**。

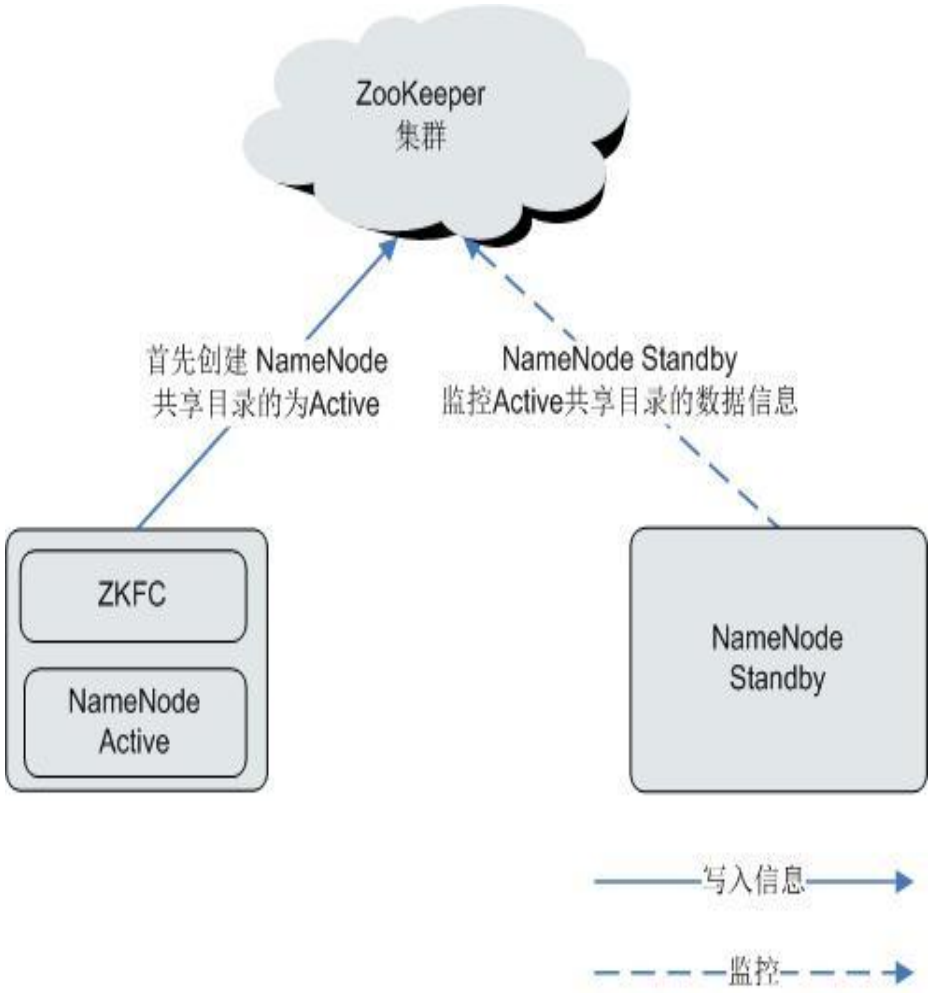
备**Nimbus**在侦听那个对应的**ZooKeeper**节点。主**Nimbus**进程宕掉之后，该节点会被删除，那么，备**Nimbus**就可以收到相应的消息。



ZooKeeper和HDFS的配合关系

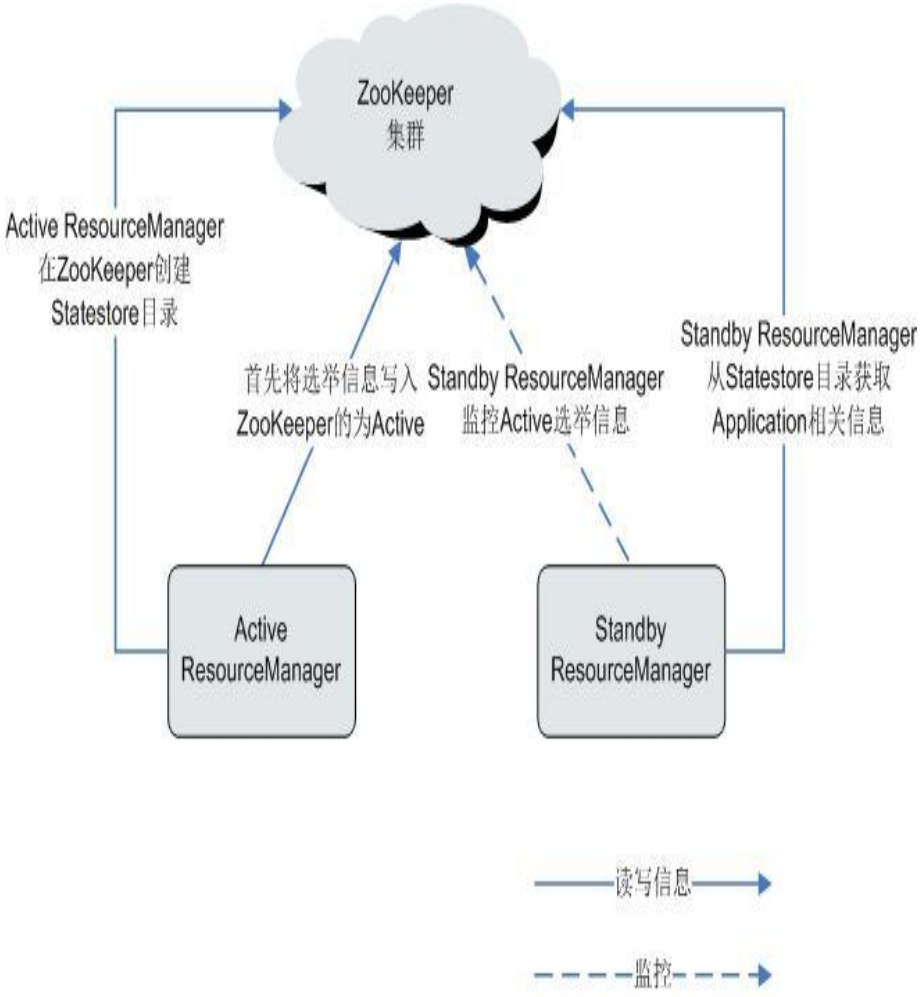
ZKFC（ZKFailoverController）作为一个**ZooKeeper**集群的客户端，用来监控**NameNode**的状态信息。**ZKFC**进程仅在部署了**NameNode**的节点中存在。**HDFS NameNode**的**Active**和**Standby**节点均部署有**zkfc**进程：

- **HDFS NameNode**的**ZKFC**连接到**ZooKeeper**，把主机名等信息保存到**ZooKeeper**中，即“**/hadoop-ha**”下的**znode**目录里。先创建**znode**目录的**NameNode**节点为主节点，另一个为备节点。**HDFS NameNode Standby**通过**ZooKeeper**定时读取**NameNode**信息。
- 当主节点进程异常结束时，**HDFS NameNode Standby**通过**ZooKeeper**感知“**/hadoop-ha**”目录下发生了变化，**NameNode**会进行主备切换。



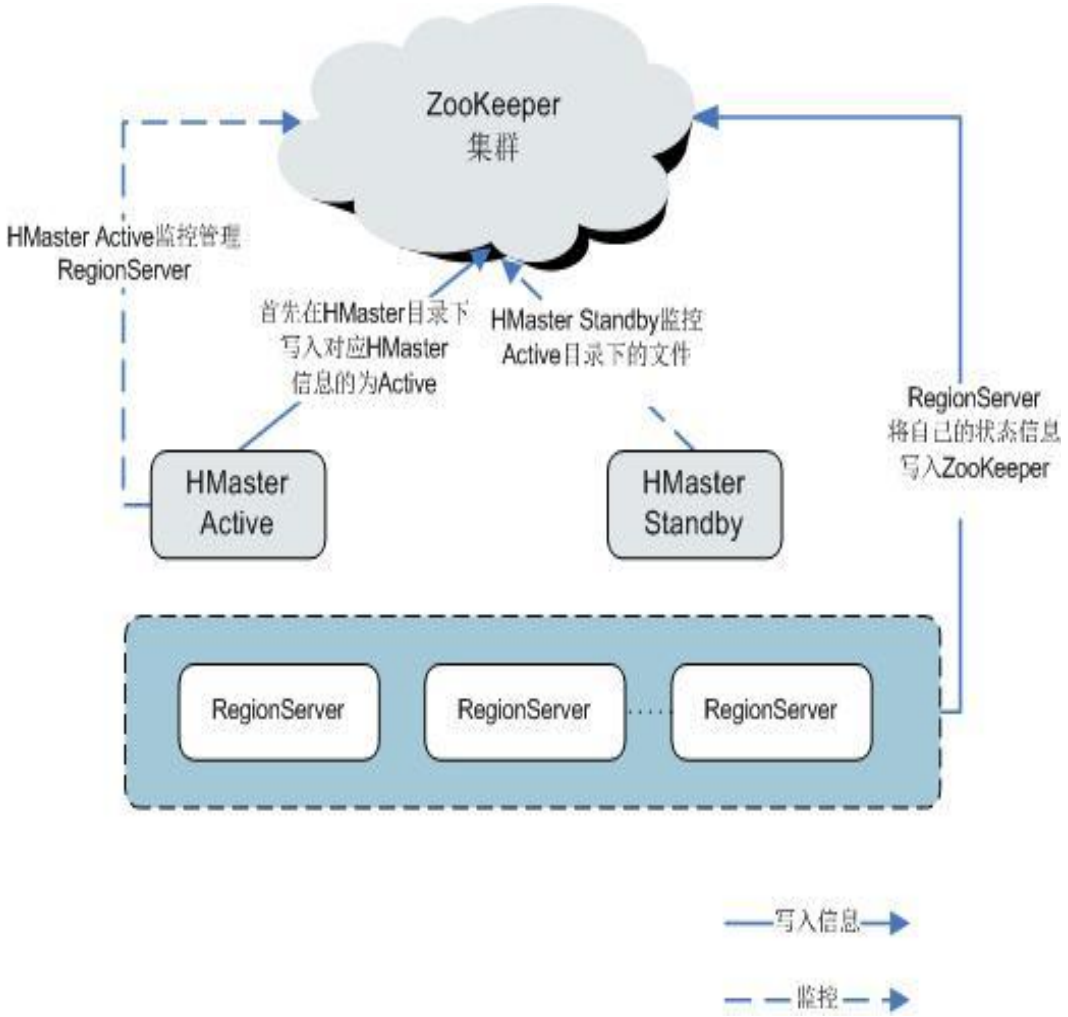
ZooKeeper和YARN的配合关系

- 在系统启动时，**ResourceManager**会尝试把选举信息写入**ZooKeeper**，第一个成功把写入**ZooKeeper**的**ResourceManager**被选举为**Active ResourceManager**，另一个为**Standby ResourceManager**。**Standby ResourceManager**定时去**ZooKeeper**监控**Active ResourceManager**选举信息。
- **Active ResourceManager**还会在**ZooKeeper**中创建**Statestore**目录，存储**Application**相关信息。当**Active ResourceManager**产生故障时，**Standby ResourceManager**会从**Statestore**目录获取**Application**相关信息，恢复数据并升为**Active ResourceManager**。



ZooKeeper和HBase的配合关系

- **HRegionServer**把自己以**Ephemeral**方式注册到**ZooKeeper**中。其中**ZooKeeper**存储**HBase**的如下信息：**HBase**元数据、**HMaster**地址。
- **HMaster**通过**ZooKeeper**随时感知各个**HRegionServer**的健康状况，以便进行控制管理。
- **HBase**也可以部署多对**HMaster**，类似**HDFS NameNode**，当**HMaster**主节点出现故障时，**HMaster**备用节点会通过**ZooKeeper**获取主**HMaster**存储的整个**HBase**集群状态信息。即通过**ZooKeeper**实现避免**HBase**单点故障问题。



总结

本章节主要介绍知识点有：

- **ZooKeeper**的作用以及在我们产品中的位置。
- **ZooKeeper**的服务架构以及数据模型。
- **ZooKeeper**的读写特性以及一致性的保证。
- **ZooKeeper**节点的创建以及节点权限设定。
- **ZooKeeper**与组件之间的关系。

思考

1 ZooKeeper在集群中的位置及作用是什么?

2 ZooKeeper节点结构是什么样子? 节点类型有哪些?

3 ZooKeeper为什么建议奇数部署?

4 ZooKeeper一致性的含义是什么?

练习题

选择题

1.ZooKeeper子节点（znode）会不会继承父节点的ACL?

A（会） B（不会）

2. ZooKeeper内置了几种认证方式（ ）

A.World B.IP C.digest D.auth E. ID

判断题

读操作只能在客户端与特定节点间完成。

附录1

1 ZooKeeper的吞吐量有其极限。经测试，**3**实例**ZooKeeper**最大吞吐量为**2~3**万次读**/**秒，**3000**次写**/**秒。若数据访问量大，请扩容**ZooKeeper**（具体数据与实际环境有关）。
2 ZooKeeper选举使用的是快速选举算法，较为复杂，可自行在网上查阅具体实现过程。
附参考网址：**http://zookeeper.apache.org/**

Thank you

www.huawei.com