1.ZooKeeper 是什么?

ZooKeeper是一个开放源码的`分布式协调服务`，它是集群的管理者，监视着集群中各个节点的状态根据节点提交的反馈进行下一步合理操作。最终，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

分布式应用程序可以基于Zookeeper实现诸如数据发布/订阅、负载均衡、命名服务、分布式协调/通知、集群管理、Master选举、分布式锁和分布式队列等功能。

Zookeeper保证了如下分布式一致性特性：

- 顺序一致性
- 原子性
- 单一视图
- 可靠性
- 实时性（最终一致性）

客户端的读请求可以被集群中的任意一台机器处理，如果读请求在节点上注册了监听器，这个监听器也是由所连接的zookeeper机器来处理。对于写请求，这些请求会同时发给其他zookeeper机器并且达成一致后，请求才会返回成功。因此，随着zookeeper的集群机器增多，读请求的吞吐会提高但是写请求的吞吐会下降。

有序性是zookeeper中非常重要的一个特性，所有的更新都是全局有序的，每个更新都有一个唯一的时间戳，这个时间戳称为`zxid（Zookeeper Transaction Id）`。而读请求只会相对于更新有序，也就是读请求的返回结果中会带有这个zookeeper最新的zxid。

2.ZooKeeper和dubbo的区别?

Zookeeper: zookeeper用来注册服务和进行负载均衡，哪一个服务由哪一个机器来提供必需让调用者知道， 简单来说就是ip地址和服务名称的对应关系。当然也可以通过硬编码的方式把这种对应关系在调用方业务代码中实现，但是如果提供服务的机器挂掉，调用者无法知晓，如果不更改代码会继续请求挂掉的机器提供服务。 zookeeper通过`心跳机制`可以检测挂掉的机器并将挂掉机器的ip和服务对应关系从列表中删除。至于支持高并发,简单来说就是横向扩展,在不更改代码的情况通过添加机器来提高运算能力。通过添加新的机器向 zookeeper注册服务，服务的提供者多了能服务的客户就多了。

dubbo: 是管理中间层的工具，在业务层到数据仓库间有非常多服务的接入和服务提供者需要调度，dubbo提供一个框架解决这个问题。

zookeeper和 dubbo的关系： Dubbo将注册中心进行抽象，它可以外接不同的存储媒介给注册中心提供服务，有 ZooKeeper, Memcached, Redis等。

> 注意这里的 dubbo只是一个框架，这个框架中要完成调度必须要有一个分布式的注册中心，储存所有服务的元数据，可以用zk，也可以用别的。

3.Zookeeper的java客户端都有哪些?

- zk自带的 zkclient
- Apache开源的 Curator

4.ZooKeeper提供了什么?

- 文件系统
- 通知机制

5.说说ZooKeeper文件系统

Zookeeper提供一个多层级的节点命名空间(节点称为 znode)。与文件系统不同的是，这些节点都可以设置关联的数据，而文件系统中只有文件节点可以存放数据而目录节点不行。

Zookeeper为了保证高吞吐和低延迟，在内存中维护了这个树状的目录结构，这种特性使得 Zookeeper不能用于存放大量的数据,每个节点的存放数据上限为1M。

6.说说ZAB协议?

ZAB协议是为分布式协调服务Zookeeper专门设计的一种支持`崩溃恢复的原子广播协议`。

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，所有进程（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来进行事务请求处理。

7.Znode有哪些类型

- `PERSISTENT-持久节点` 除非手动删除，否则节点一直存在于Zookeeper上
- `EPHEMERAL-临时节点` 临时节点的生命周期与客户端会话绑定，一旦客户端会话失效（客户端与zookeeper连接断开不一定会话失效），那么这个客户端创建的所有临时节点都会被移除。
- `PERSISTENT_SEQUENTIAL-持久顺序节点` 基本特性同持久节点，只是增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。
- `EPHEMERAL_SEQUENTIAL-临时顺序节点` 基本特性同临时节点，增加了顺序属性，节点名后边会追加一个由父节点维护的自增整型数字。

8.Zookeeper节点宕机如何处理?

Zookeeper本身也是集群，推荐配置`不少于3个服务器`。Zookeeper自身也要保证当一个节点宕机时，其他节点会继续提供服务。 如果是一个Follower宕机，还有2台服务器提供访问，因为Zookeeper上的数据是有多个副本的，数据并不会丢失； 如果是一个Leader宕机，Zookeeper会选举出新的Leader。

ZK集群的机制是只要超过半数的节点正常，集群就能正常提供服务。 只有在ZK节点挂得太多，只剩一半或不到一半节点能工作，集群才失效。

所以：

- 3个节点的cluster可以挂掉1个节点(leader可以得到2票>1.5)
- 2个节点的cluster不能挂掉任何1个节点(leader可以得到1票<=1)

9.Zookeeper有哪几种几种部署模式?

Zookeeper有三种部署模式：

- `单机部署`：一台集群上运行
- `集群部署`：多台集群运行
- `伪集群部署`：一台集群启动多个 Zookeeper实例运行

10.Zookeeper的典型应用场景?

Zookeeper是一个典型的`发布/订阅模式`的分布式数据管理与协调框架，开发人员可以使用它来进行分布式数据的发布和订阅。 通过对 Zookeeper中丰富的数据节点进行交叉使用，配合`Watcher事件通知机制`，可以非常方便的构建一系列分布式应用，会涉及的核心功能：

- 数据发布/订阅
- 负载均衡
- 命名服务
- 分布式协调/通知
- 集群管理
- Master选举
- 分布式锁
- 分布式队列

11.说一下Zookeeper Watcher机制

Zookeeper允许客户端向服务端的某个 Znode注册个 Watcher监听，当服务端的一些指定事件触发了这个 Watcher，服务端会向指定客户端发送一个事件通知来实现分布式的通知功能，客户端根据Watcher通知状态和事件类型做出业务上的改变。

工作机制：

- 客户端注册 watcher
- 服务端处理watcher
- 客户端回调 watcher

12.客户端注册Watcher的流程?

1、客户端注册`Watcher实现` 2、调用`getData()` / `getChildren()` / `exist()` 三个API，传入Watcher对象 3、标记请求request，封装Watcher到`WatchRegistration` 4、封装成`Packet`对象，发服务端发送request 5、收到服务端响应后，将Watcher注册到`ZKWatcherManager`中进行管理 6、请求返回，完成注册。

13.服务端处理Watcher的流程?

1、`服务端接收Watcher并存储` 接收到客户端请求，处理请求判断是否需要注册Watcher，需要的话将数据节点的节点路径和ServerCnxn（ServerCnxn代表一个客户端和服务端的连接，实现了Watcher的process接口，此时可以看成一个Watcher对象）存储在WatcherManager的WatchTable和watch2Paths中去。

2、`Watcher触发`

- 以服务端接收到 setData() 事务请求触发NodeDataChanged事件为例：
- 封装WatchedEvent
- 将通知状态（SyncConnected）、事件类型（NodeDataChanged）以及节点路径封装成一个WatchedEvent对象
- 查询Watcher
- 从WatchTable中根据节点路径查找Watcher 没找到；说明没有客户端在该数据节点上注册过Watcher 找到；提取并从WatchTable和Watch2Paths中删除对应Watcher（从这里可以看出Watcher在服务端是一次性的，触发一次就失效了）

3、`调用process方法来触发Watcher` 这里process主要就是通过ServerCnxn对应的TCP连接发送Watcher事件通知。

14.客户端回调 Watcher流程?

1、客户端`SendThread`线程接收事件通知，交由 EventThread线程回调 Watcher。 2、客户端的Watcher机制同样是一次性的，一旦被触发后，该 Watcher就失效了。

15.Zookeeper对节点的 watch监听通知是永久的吗?为什么不是永久的?

不是永久的。 官方声明：一个 Watch事件是一个次性的触发器，当被设置了 Watch的数据发生了改变的时候，则服务器将这个改变发送给设置了 Watch的客户端，以便通知它们。

原因： 如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，给网络和服务器造成很大压力。一般是客户端执行`getData`（“/节点”，true），如果节点A发生了变更或删除，客户端会得到它的 watch事件，但是在之后节点A又发生了变更，而客户端又没有设置 watch事件，就不再给客户端发送。

> 在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。

16.说说ACL权限控制机制

1、`权限模式（Scheme）` IP：从IP地址粒度进行权限控制 Digest：最常用，用类似于username:password 的权限标识来进行权限配置，便于区分不同应用来进行权限控制 World：最开放的权限控制方式，是一种特殊的digest模式，只有一个权限标识“world:anyone” Super：超级用户

2、`授权对象` 授权对象指的是权限赋予的用户或一个指定实体，例如IP地址或是机器灯。

3、`权限 Permission`

- CREATE：数据节点创建权限，允许授权对象在该Znode下创建子节点
- DELETE：子节点删除权限，允许授权对象删除该数据节点的子节点
- READ：数据节点的读取权限，允许授权对象访问该数据节点并读取其数据内容或子节点列表等
- WRITE：数据节点更新权限，允许授权对象对该数据节点进行更新操作
- ADMIN：数据节点管理权限，允许授权对象对该数据节点进行ACL相关设置操作
-

17.服务器有哪些角色?

1、`Leader` 事务请求的唯一调度和处理者，保证集群事务处理的顺序性 集群内部各服务的调度者

2、`Follower` 处理客户端的非事务请求，转发事务请求给Leader服务器 参与事务请求Proposal的投票 参与Leader选举投票

3、`Observer` 处理客户端的非事务请求，转发事务请求给Leader服务器 不参与任何形式的投票

18.Zookeeper 下 Server工作状态有哪些?

1、`LOOKING`：寻找Leader状态。当服务器处于该状态时，它会认为当前集群中没有Leader，因此需要进入Leader选举状态。 2、`FOLLOWING`：跟随者状态。表明当前服务器角色是Follower。 3、`LEADING`：领导者状态。表明当前服务器角色是Leader。 4、`OBSERVING`：观察者状态。表明当前服务器角色是Observer。

19.集群支持动态添加机器吗?

其实就是水平扩容，Zookeeper在这方面不太好。 两种方式：

- `全部重启`：关闭所有 Zookeeper服务，修改配置之后启动。不影响之前客户端的会话。

- `逐个重启`：在过半存活即可用的原则下一台机器重启不影响整个集群对外提供服务。（这是比较常用的方式）

> 3.5版本开始支持动态扩容。

20.什么是Chroot?

3.2.0版本后，添加了特性，该特性允许每个客户端为自己设置一个命名空间。如果一个客户端设置了Chroot，那么该客户端对服务器的任何操作，都将会被限制在其自己的命名空间下。

> 通过设置 Chroot，能够将一个客户端应用与 Zookeeper服务端的一颗子树相对应，在那些多个应用