ZooKeeper是一种分布式协调服务，用于管理大型主机。在分布式环境中协调和管理服务是一个复杂的过程。ZooKeeper通过其简单的架构和API解决了这个问题。ZooKeeper允许开发人员专注于核心应用程序逻辑，而不必担心应用程序的分布式特性。

ZooKeeper框架最初是在“Yahoo!"上构建的，用于以简单而稳健的方式访问他们的应用程序。 后来，Apache ZooKeeper成为Hadoop，HBase和其他分布式框架使用的有组织服务的标准。 例如，Apache HBase使用ZooKeeper跟踪分布式数据的状态。

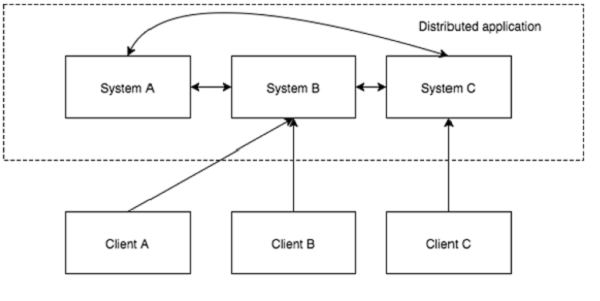
在进一步深入之前，我们了解关于分布式应用的一两件事情是很重要的。因此，让我们开始分布式应用的概述的快速讨论。

## 分布式应用

分布式应用可以在给定时间（同时）在网络中的多个系统上运行，通过协调它们以快速有效的方式完成特定任务。通常来说，对于复杂而耗时的任务，非分布式应用（运行在单个系统中）需要几个小时才能完成，而分布式应用通过使用所有系统涉及的计算能力可以在几分钟内完成。

通过将分布式应用配置为在更多系统上运行，可以进一步减少完成任务的时间。分布式应用正在运行的一组系统称为**集群**，而在集群中运行的每台机器被称为**节点**。

分布式应用有两部分，**Server（服务器）**和**Client（客户端）**应用程序。服务器应用程序实际上是分布式的，并具有通用接口，以便客户端可以连接到集群中的任何服务器并获得相同的结果。 客户端应用程序是与分布式应用进行交互的工具。

[](http://219.83.161.15:9999/zentao/file-read-61.png)

### 分布式应用的优点

* **可靠性** - 单个或几个系统的故障不会使整个系统出现故障。
* **可扩展性** - 可以在需要时增加性能，通过添加更多机器，在应用程序配置中进行微小的更改，而不会有停机时间。
* **透明性** - 隐藏系统的复杂性，并将其显示为单个实体/应用程序。

### 分布式应用的挑战

* **竞争条件** - 两个或多个机器尝试执行特定任务，实际上只需在任意给定时间由单个机器完成。例如，共享资源只能在任意给定时间由单个机器修改。
* **死锁** - 两个或多个操作等待彼此无限期完成。
* **不一致** - 数据的部分失败。

## 什么是Apache ZooKeeper？

Apache ZooKeeper是由集群（节点组）使用的一种服务，用于在自身之间协调，并通过稳健的同步技术维护共享数据。ZooKeeper本身是一个分布式应用程序，为写入分布式应用程序提供服务。

ZooKeeper提供的常见服务如下 :

* **命名服务** - 按名称标识集群中的节点。它类似于DNS，但仅对于节点。
* **配置管理** - 加入节点的最近的和最新的系统配置信息。
* **集群管理** - 实时地在集群和节点状态中加入/离开节点。
* **选举算法** - 选举一个节点作为协调目的的leader。
* **锁定和同步服务** - 在修改数据的同时锁定数据。此机制可帮助你在连接其他分布式应用程序（如Apache HBase）时进行自动故障恢复。
* **高度可靠的数据注册表** - 即使在一个或几个节点关闭时也可以获得数据。

分布式应用程序提供了很多好处，但它们也抛出了一些复杂和难以解决的挑战。ZooKeeper框架提供了一个完整的机制来克服所有的挑战。竞争条件和死锁使用**故障安全同步方法**进行处理。另一个主要缺点是数据的不一致性，ZooKeeper使用**原子性**解析。

## ZooKeeper的好处

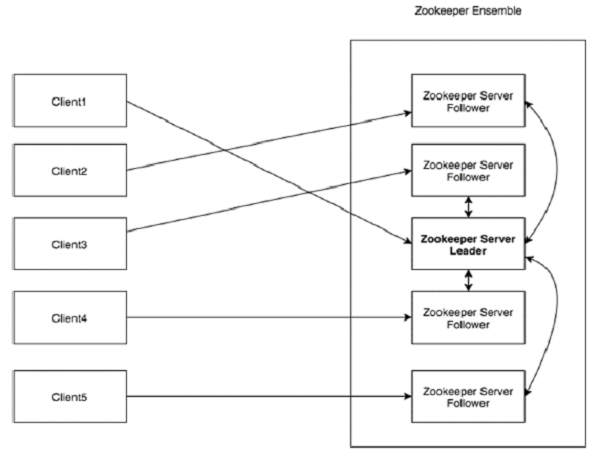
以下是使用ZooKeeper的好处：

* **简单的分布式协调过程**
* **同步** - 服务器进程之间的相互排斥和协作。此过程有助于Apache HBase进行配置管理。
* **有序的消息**
* **序列化** - 根据特定规则对数据进行编码。确保应用程序运行一致。这种方法可以在MapReduce中用来协调队列以执行运行的线程。
* **可靠性**
* **原子性** - 数据转移完全成功或完全失败，但没有事务是部分的。

在深入了解ZooKeeper的运作之前，让我们来看看ZooKeeper的基本概念。我们将在本章中讨论以下主题：  
1、Architecture（架构）  
2、Hierarchical namespace（层次命名空间）  
3、Session（会话）  
4、Watches（监视）

**ZooKeeper的架构**

看看下面的图表。它描述了ZooKeeper的“客户端-服务器架构”。

[](http://219.83.161.15:9999/zentao/file-read-58.jpg)

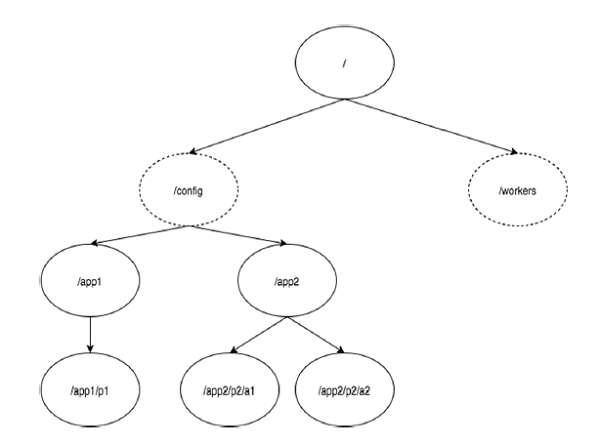
作为ZooKeeper架构的一部分的每个组件在下表中进行了说明。

|  |  |
| --- | --- |
| **部分** | **描述** |
| Client（客户端） | 客户端，我们的分布式应用集群中的一个节点，从服务器访问信息。对于特定的时间间隔，每个客户端向服务器发送消息以使服务器知道客户端是活跃的。  类似地，当客户端连接时，服务器发送确认码。如果连接的服务器没有响应，客户端会自动将消息重定向到另一个服务器。 |
| Server（服务器） | 服务器，我们的ZooKeeper总体中的一个节点，为客户端提供所有的服务。向客户端发送确认码以告知服务器是活跃的。 |
| Ensemble | ZooKeeper服务器组。形成ensemble所需的最小节点数为3。 |
| Leader | 服务器节点，如果任何连接的节点失败，则执行自动恢复。Leader在服务启动时被选举。 |
| Follower | 跟随leader指令的服务器节点。 |

**层次命名空间**

下图描述了用于内存表示的ZooKeeper文件系统的树结构。ZooKeeper节点称为**znode**。每个znode由一个名称标识，并用路径(/)序列分隔。

* 在图中，首先有一个由“/”分隔的znode。在根目录下，你有两个逻辑命名空间**config**和**workers**。
* **config**命名空间用于集中式配置管理，**workers**命名空间用于命名。
* 在**config**命名空间下，每个znode最多可存储1MB的数据。这与UNIX文件系统相类似，除了父znode也可以存储数据。这种结构的主要目的是存储同步数据并描述znode的元数据。此结构称为**ZooKeeper数据模型**。

[](http://219.83.161.15:9999/zentao/file-read-59.jpg)

ZooKeeper数据模型中的每个znode都维护着一个**stat**结构。一个stat仅提供一个znode的**元数据**。它由版本号，操作控制列表(ACL)，时间戳和数据长度组成。

* **版本号** - 每个znode都有版本号，这意味着每当与znode相关联的数据发生变化时，其对应的版本号也会增加。当多个zookeeper客户端尝试在同一znode上执行操作时，版本号的使用就很重要。
* **操作控制列表(ACL)** - ACL基本上是访问znode的认证机制。它管理所有znode读取和写入操作。
* **时间戳** - 时间戳表示创建和修改znode所经过的时间。它通常以毫秒为单位。ZooKeeper从“事务ID"(zxid)标识znode的每个更改。**Zxid**是唯一的，并且为每个事务保留时间，以便你可以轻松地确定从一个请求到另一个请求所经过的时间。
* **数据长度** - 存储在znode中的数据总量是数据长度。你最多可以存储1MB的数据。

**Znode的类型**

Znode被分为持久（persistent）节点，顺序（sequential）节点和临时（ephemeral）节点。

* **持久节点** - 即使在创建该特定znode的客户端断开连接后，持久节点仍然存在。默认情况下，除非另有说明，否则所有znode都是持久的。
* **临时节点**- 客户端活跃时，临时节点就是有效的。当客户端与ZooKeeper集合断开连接时，临时节点会自动删除。因此，只有临时节点不允许有子节点。如果临时节点被删除，则下一个合适的节点将填充其位置。临时节点在leader选举中起着重要作用。
* **顺序节点**- 顺序节点可以是持久的或临时的。当一个新的znode被创建为一个顺序节点时，ZooKeeper通过将10位的序列号附加到原始名称来设置znode的路径。例如，如果将具有路径**/myapp**的znode创建为顺序节点，则ZooKeeper会将路径更改为**/myapp0000000001**，并将下一个序列号设置为0000000002。如果两个顺序节点是同时创建的，那么ZooKeeper不会对每个znode使用相同的数字。顺序节点在锁定和同步中起重要作用。

**Sessions（会话）**

会话对于ZooKeeper的操作非常重要。会话中的请求按FIFO顺序执行。一旦客户端连接到服务器，将建立会话并向客户端分配**会话ID**。

客户端以特定的时间间隔发送**心跳**以保持会话有效。如果ZooKeeper集合在超过服务器开启时指定的期间（会话超时）都没有从客户端接收到心跳，则它会判定客户端死机。

会话超时通常以毫秒为单位。当会话由于任何原因结束时，在该会话期间创建的临时节点也会被删除。

**Watches（监视）**

监视是一种简单的机制，使客户端收到关于ZooKeeper集合中的更改的通知。客户端可以在读取特定znode时设置Watches。Watches会向注册的客户端发送任何znode（客户端注册表）更改的通知。

Znode更改是与znode相关的数据的修改或znode的子项中的更改。只触发一次watches。如果客户端想要再次通知，则必须通过另一个读取操作来完成。当连接会话过期时，客户端将与服务器断开连接，相关的watches也将被删除。