# 一：名称

Zookeeper -- 分布式服务框架

# 二：描述

分布式协调服务框架

是Apache Hadoop 的一个子项目

是一个针对分布式应用的可靠协调系统。

可以解决分布式环境中经常遇到的一些数据管理问题：

如：

统一命名服务、

状态同步服务、

集群管理、

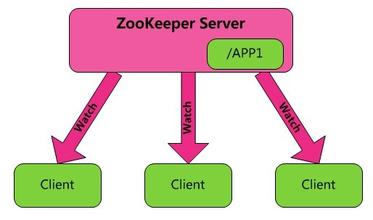
分布式应用配置项的管理

提供分布式环境中的 **管理数据服务** 和 **协调数据服务**

实例：

**动物园管理员**　职责　**管理动物** 和 **协调游客去哪观看动物，（做一些动物标识，引导游客）**

# 三：Zookeeper结构



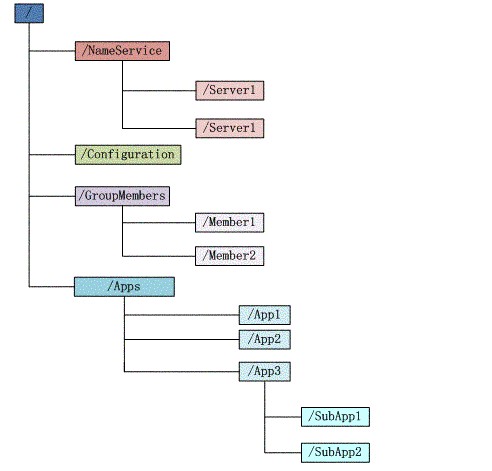
**角色：Zookeeper = 服务端 + 客户端**

|  |  |
| --- | --- |
| 服务端 | 支持群集，包括文件系统和通知机制 |
| 客户端 | 连接服务端，操作数据，注册要监听的数据，接收通知 |

**功能：Zookeeper = 文件系统 + 通知机制**

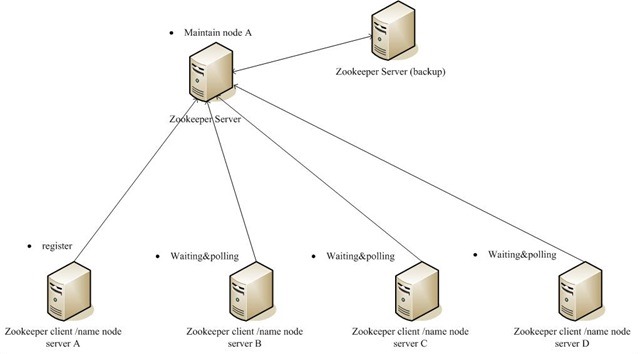
|  |  |
| --- | --- |
| 文件系统 | 存储和管理数据，以树型的结构存储 |
| 通知机制 | 监控文件系统的数据变化，通知监听的客户端 |

## 1：文件系统



|  |  |
| --- | --- |
| Zookeeper数据结构特点： | |
| **路径唯一标识** | 树型存储结构，与文件系统目录树结构类似  例：/Apps/App3/SubApp1 |
| **节点下可以创建子节点** | 可以针对节点进行增，删，改，查，创建子节点操作  各个节点下可以存储数据,数据可以有多个版本。 |
| **临时节点** | 一旦创建这个 节点 的客户端与服务器失去联系，这个 节点 也将自动删除，  Zookeeper 的客户端和服务器通信采用长连接方式，  每个客户端和服务器通过心跳来保持连接，  这个连接状态称为 session，如果 节点 是临时节点，  这个 session 失效，节点 也就删除了.  临时节点下不能创建子节点 |
| **永久节点** | 创建这个 节点 的客户端与服务器失去联系，这个节点不会删除。 |
| **节点的名称可以自动编号** | 如 App1 已经存在，再创建的话，将会自动命名为 App2 |
| **节点可以被监控** | 节点中存储数据的修改，子节点目录的变化等，一旦变化可以通知设置监控的客户端，这个是 Zookeeper 的核心特性，Zookeeper 的很多功能都是基于这个特性实现的. |

## 2：通知机制

Zookeeper 的客户端和服务器通信采用长连接方式，

客户端

watch

watch

watch

watch

Zookeeper服务端

客户端　注册监听　它关心的目录节点，

当目录节点发生变化（数据改变、被删除、子目录节点增加删除）时，

每个客户端和服务器通过心跳来保持连接，zookeeper会通知客户端。

## 3：总结



watch

watch

watch

watch

watch

watch

watch

watch

watch

**存储数据**

**Zookeeper原理：**

|  |  |  |
| --- | --- | --- |
| 1 | 存储 | 负责存储和管理数据 |
| 2 | 注册 | 然后接受 客户端 的注册，监听相关的数据 |
| 3 | 变化 | 一旦这些数据的状态发生变化 |
| 4 | 通知 | Zookeeper 负责通知已经在 Zookeeper 上注册的那些客户端 |
| 5 | 业务处理 | 客户端做出相应的反应 |

Zookeeper 特点：

|  |  |
| --- | --- |
| 顺序一致性 | 按照客户端发送请求的顺序更新数据。 |
| 原子性 | 更新要么成功，要么失败，不会出现部分更新。 |
| 单一性 | 无论客户端连接哪个server，都会看到同一个视图。 |
| 可靠性 | 一旦数据更新成功，将一直保持，直到新的更新。 |
| 及时性 | 客户端会在一个确定的时间内得到最新的数据。 |

Zookeeper功能：

|  |  |  |
| --- | --- | --- |
| 1 | 名称服务 | 统一命名服务, 唯一的名称, 树形的名称结构,  便于人识别和记住, 不会重复 |
| 2 | 配置维护 | 分布式应用配置项的管理 |
| 3 | 分布式同步 | 状态同步服务 |
| 4 | 组服务 | 集群管理 |

注：

Zoopkeeper 提供了一套很好的分布式集群管理的机制，

从而可以设计出多种多样的分布式的数据管理模型，

而不仅仅局限于下面提到的几个常用应用场景。

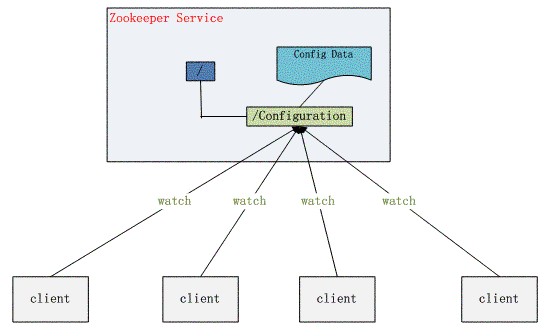
# 四：****Zookeeper应用场景****

## 1：配置数据的管理

场景描述：

数据发布与订阅模型 (配置中心)

发布者将数据发布到某节点上，供订阅者动态获取数据



实现：

应用配置集中到节点上，

应用启动时主动获取，并在节点上注册一个watcher，

每次配置更新都会通知到应用,

达到获取最新配置信息的目的。

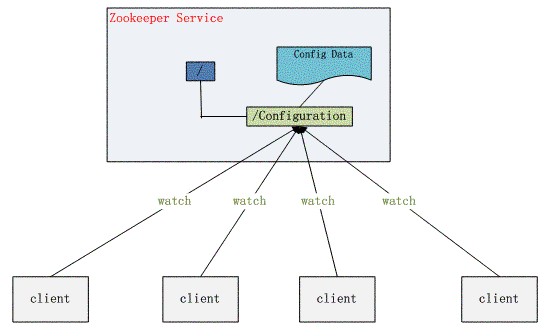
好处：

实现配置信息的集中式管理和动态更新，实时同步。

注：

在上面提到的应用场景中，有个默认前提是：数据量很小，但是数据更新可能会比较快的场景。

例：



全局配制

数据字典

渠道ＣＲＭ

行业ＣＲＭ

电销ＣＲＭ

资源平台

## 2：软负载均衡

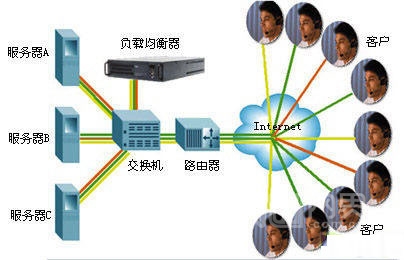
场景描述：

随着业务量的提高，访问量和数据流量的快速增长，其处理能力和计算强度也相应地增大，使得单一的服务根本无法承担。将负载（工作任务）进行平衡、分摊到多个操作单元上进行执行，在分布式环境中，为了保证高可用性，通常同一个应用或同一个服务的提供方都会部署多份，达到对等服务。

而消费者就须要在这些对等的服务器中选择一个来执行相关的业务逻辑。

好处：

高可用性,减轻服务的压力，实现软负载均衡



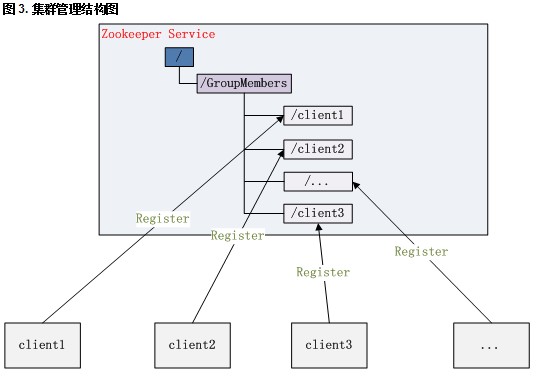
服务中间件

Zookeeper

实现：

客户请求服务，服务中间件读取Zookeeper服务端取得提供服务的列表，根据规则，把请求定位到某个服务器执行相关的业务。

## 3：集群管理



集群机器管理:

监控 　　　：监控集群中机器状态,监控机器在线率，实时检测集群机器是否存活

实现：

1.客户端启动时，向　Zookeeper上注册.

2.客户端在节点 x 上注册一个Watcher，那么如果 x的子节点变化了，会通知该客户端。

3.创建临时类型的节点，一旦客户端和服务器的会话结束或过期，那么该节点就会删除。

Master选举:

有些业务逻辑（例如一些耗时的计算，网络I/O处理），往往只需要让整个集群中的某一台机器进行执行，提高性能。

实现：

创建节点时，分配一个序号，选举时，取序列号最小的那个机器作为Master

## 4：分布式锁

锁：

不同线程间，操作一个资源，存在资源竞争，要用到锁。

分布式锁在进程与进程之间提供了一种互斥机制，在任何时刻，只有一个进程可以持有锁。

锁服务可以分为两类：

保持独占：

得益于ZooKeeper为我们保证了数据的强一致性。

就是所有试图来获取这个锁的客户端，最终只有一个可以成功获得这把锁。

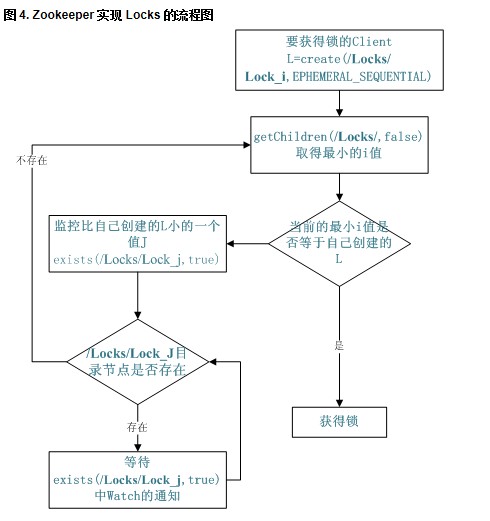
(实现：大家都在某个节点下创建节点，只有一个成功，其它都是失败的)

控制时序：

就是所有视图来获取这个锁的客户端，最终都是会被安排执行，

只是有个全局时序了。

（实现：创建有编号节点，查询按编号排序，取最小的编号，就能得到锁）



一个用户创建一个节点，分配数值

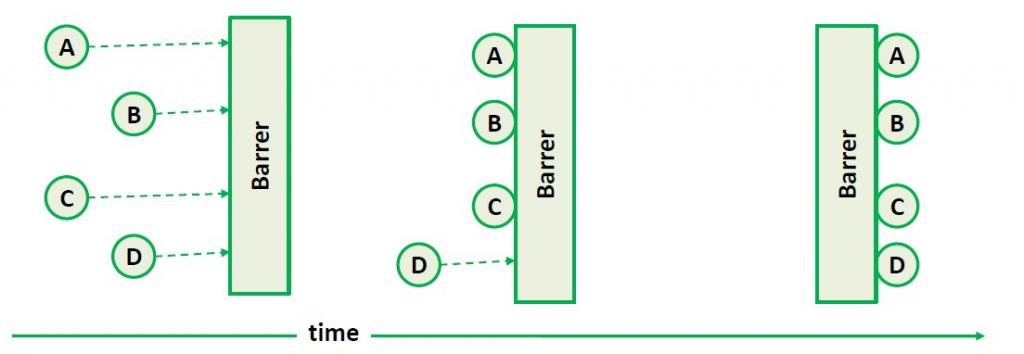
检测父节点下的节点列表的最小数据是否为自已创建的数据

如果是，代表获得锁。

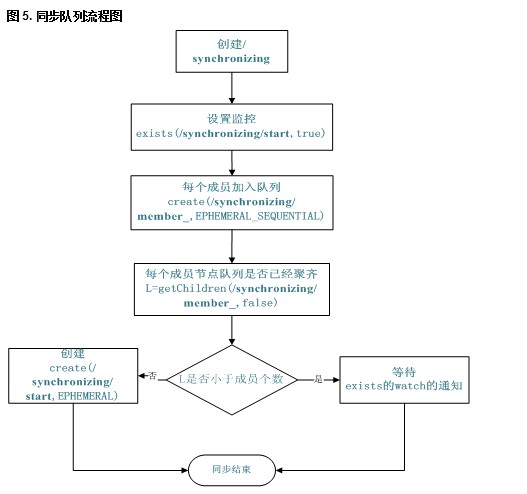
如果不是，代表别的用户已经锁住.

## 5：队列管理

|  |  |  |
| --- | --- | --- |
| 1 | 同步队列 | 描述：  当一个队列的成员都聚齐时，这个队列才可用，否则一直等待所有成员到达，这种是同步队列.  例如实现 **(发并处理)**  实现：  在约定目录下创建临时目录节点，监听节点数目是否是我们要求的数目。 |
| 2 | FIFO队列 | 描述：  队列按照 FIFO 方式进行入队和出队操作,  例如实现 (**生产者和消费者模型**)  实现：  和分布式锁服务中的控制时序场景基本原理一致，入队有编号，出队按编号。 |



**初始化**



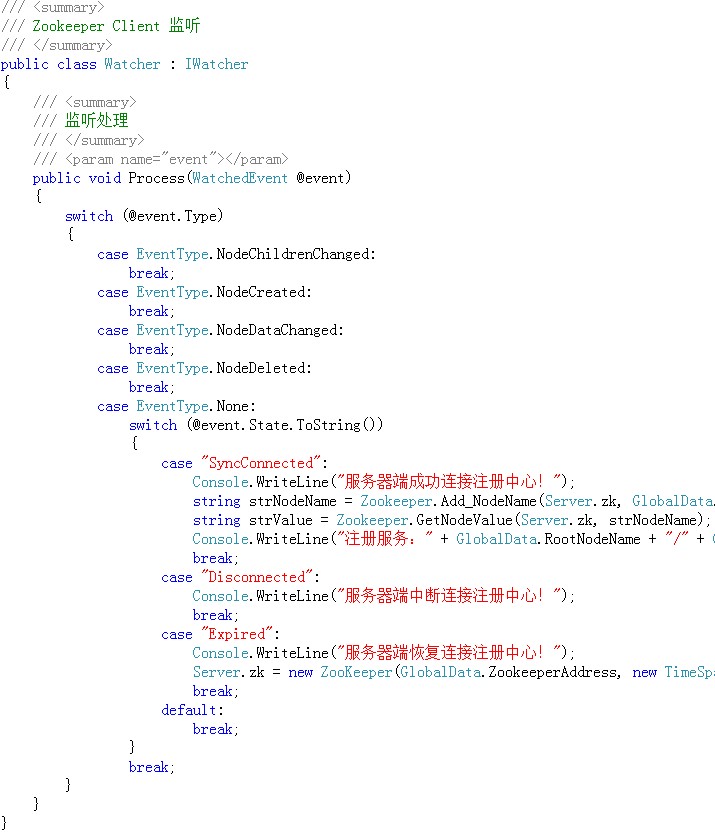
# 五：常用接口



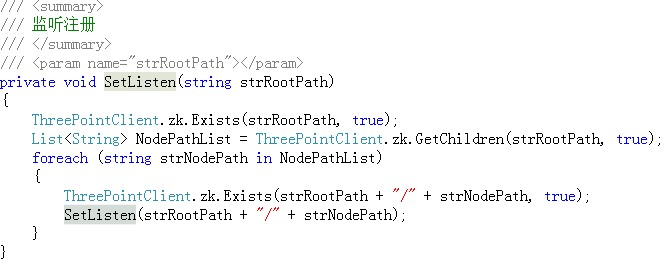


**注：**

**1：Zookeeper Client Watcher监听事件**



**2：监听只会执行一次，必需重新注册监听**



# 六：Zookeeper配置

## 1：下载

<http://hadoop.apache.org/zookeeper/>

## 2：安装

获取到 Zookeeper 的压缩包并解压到某个目录D:\Home\zookeeper-3.4.3

## 3：配置

D:\Home\zookeeper-3.4.3\conf\zoo\_sample.cfg

把这个文件名字改成

D:\Home\zookeeper-3.4.3\conf\zoo.cfg

zoo.cfg内容：

tickTime=2000

dataDir=D:/devtools/zookeeper-3.2.2/build

clientPort=2181

zoo.cfg说明：

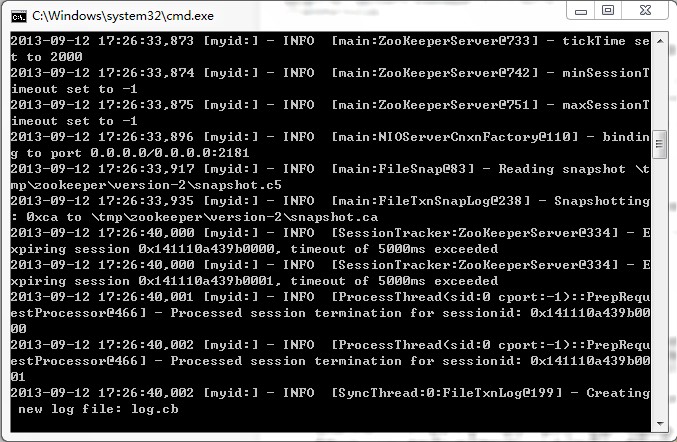
tickTime：这个时间是作为 Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。

dataDir：顾名思义就是 Zookeeper 保存数据的目录，默认情况下，Zookeeper 将写数据的日志文件也保存在这个目录里。

clientPort：这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。

## 4：启动服务器

执行文件： D:\ProgramFiles\zookeeper-3.4.3\bin\zkServer.cmd



# 七：实例Demo

注：

1：.Net Framework 4.0

2：引用 ZookeeperNet.dll