**ZooKeeper**

目录

[1. ZooKeeper: 分布式应用程序的分布式协调服务 2](#_Toc501886436)

# ZooKeeper: 分布式应用程序的分布式协调服务

ZooKeeper 是分布式开源的协调分布式应用程序的工具。它公开了一组简单的基元分布式应用程序可以利用这些原语实现更高层次的同步、配置维护、组合命名服务。它被设计成易于编程，并使用一个数据模型的样式后，熟悉的目录树结构的文件系统。它在java中运行，并且具有Java和C的绑定。

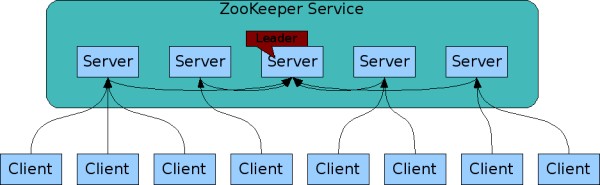
众所周知协调服务是很难的到正确的。它们特别容易出现诸如竞争条件和死锁等错误。ZooKeeper的目的是减轻分布式应用程序的责任，从零开始执行协调服务。

## 1.1 设计目标

**ZooKeeper 是简单的.** ZooKeeper允许分布式进程通过类似于标准文件系统的共享层次命名空间相互协调。命名空间由数据寄存器组成----znodes，在ZooKeeper的说法中这些都是类似的文件和目录。与设计用于存储的典型文件系统不同，ZooKeeper数据保存在内存中，这以为着ZooKeeper可以实现高吞吐量和低延迟数量。

ZooKeeper实现对高性能，包可用性强制有序访问。ZooKeeper的性能方面以为着它可以用于大型分布式系统。可靠性方面使它不能成为单一故障点。严格的顺序访问以为着可以在客户端实现负载的同步原语。

**ZooKeeper 被复制.** 就像它所协调的分布式进程一样，ZooKeeper本身也被复制到一组成为集合的主机上。



ZooKeeper 服务

构成ZooKeeper服务的服务器必须全部相互了解。他们维护了一个内存状态的镜像，以及一个持久存储的事务日志和快照。只要大部分服务器都可用，ZooKeeper服务将可用。

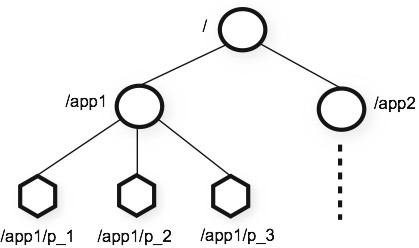
客户端连接到一台ZooKeeper服务器。客户端维护一个TCP连接，通过它发送请求，获取响应，获取监听事件，并发送心跳。如果到服务器的TCP连接总段，客户端将连接到不同的服务器。

**ZooKeeper 是有序的.** ZooKeeper使用映射所有的ZooKeeper事务顺序的数字来标记每个更新。后续操作可以使用该命令实现更高级别的抽象，例如同步原语。

**ZooKeeper 是非常快的.** It 它在“read-dominant”的工作负载中非常快。ZooKeeper应用程序在数千台机器上运行，并且在读取比写入更为普遍的情况下，性能表现最佳，比例约为10:1.

## 1.2 数据模型和分层命名空间

ZooKeeper提供的命名空间与标准的文件系统非常相似。命名是由”/”分割的一些列路径元素。ZooKeeper命名空间中的每个节点都由一个路径标识。



ZooKeeper's Hierarchical Namespace

## 1.3 节点和短暂节点

与标准文件系统不同的是，ZooKeeper命名空间中的每个节点都可以拥有与其相关的数据以及子节点。这就像有一个文件系统，允许一个文件也是一个目录。(ZooKeeper被设计用于存储协调数据:状态信息，配置，位置信息等，英雌存储在每个节点的数据通常很小，在字节到千字节范围内)。我们使用术语znode来说明我们正在谈论的ZooKeeper数据节点。

Znodes维护一个统计结构，其中包括数据更改的版本号，ACL(Access Control List)更改和时间戳，以及允许缓存验证和协调更新。每当一个znode的数据发生变化，版本号就会增加。例如，每当客户端检索数据时，它也会受到数据的版本。

存储在命名空间中的每个节点上的数据是以原子方式读取和写入的。读取获得与znode关联的所有数据字节，写入将替换所有数据。每个节点都有一个访问控制列表(ACL)，

限制谁可以做什么。

ZooKeeper 也有短暂节点的概念。只要创建znode的会话处于活动状态，就会存在这些znode。当会话结束时，znode被删除。当你想实现[tbd](待决定)时，短暂节点很有用。

## 1.4 有条件的更新和监控

ZooKeeper支持监控的概念。客户端可以在znodes上设个一个监控。当znode改变时，监控将被触发并移除。当监控触发时，客户端受到一个数据包说znode已经改变。如果客户端和ZooKeeper服务器之间的连接中断，客户端将收到本地通知。这些可以用来[tbd]

## 1.5 担保

ZooKeeper非常快速，非常简单。既然它的目标是构建比较复杂的服务，比如同步，就提供了一套保证。这些事:

* 顺序一致性----来自客户端的更新将按照发送顺讯进行应用。
* 原子性----更新成功或失败。没有部分结果。
* 单个系统镜像----无论服务器连接到那个服务器，客户端都将看到相同的服务视图。
* 可靠性----一旦应用更新，它将一直持续到客户端覆盖更新。
* 及时性----系统的客户端视角在一定的时间范围内保证是最新的。

有关这些的更多信息，以及如何使用它们，请参考[tbd]

## 1.6 简单的 API

ZooKeeper的一个设计目标是提供一个非常简单的编程接口。因此，它只支持这些操作:

create

creates a node at a location in the tree

delete

deletes a node

exists

tests if a node exists at a location

get data

reads the data from a node

set data

writes data to a node

get children

retrieves a list of children of a node

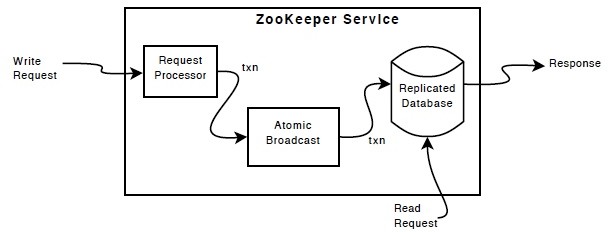
sync

waits for data to be propagated

有关这些更深入的探讨，以及如何使用它们来实现更高层次的操作，请参考*[tbd]*

## 1.7 实现

ZooKeeper Components 展示了ZooKeeper服务的高级组件。除了请求处理器之外，构成ZooKeeper服务的每个服务器赋值其每个组件的自己的副本。



ZooKeeper Components

赋值的数据库是包含整个数据树的内存数据库。将更新记录到磁盘以实现可恢复性，并在写入内存数据库之前将其写入磁盘。

每个ZooKeeper服务器都为客户端提供服务。客户端连接到一个服务器提交irequest。读取请求是从每个服务器数据库的本地副本服务的。请求改变服务的状态，写请求，由协议处理。

作为协议的一部分，来自客户端的所有写入请求都将被转发给一个称谓leader的服务器。ZooKeeper服务器的其余部分(follower)，接收leader的消息建议并同意消息传递。消息传递层负责替换失败的leader，并与leader同步follows。

ZooKeeper使用自定义的原子消息传递协议。由于消息传递层是原子的，因此ZooKeeper可以保证本地副本不会发生分离。当leader接收到一个写请求时，它将计算什么时候系统的状态是什么时候写入被应用并且把它转换成捕获这个新状态的事务。

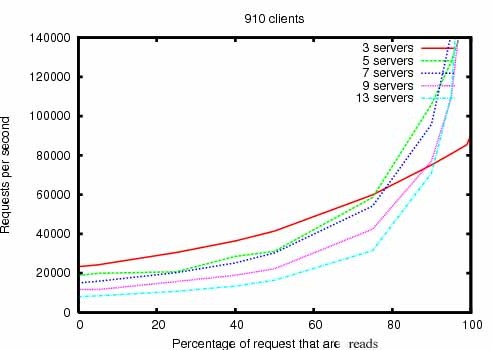
## 1.8 用途

ZooKeeper的编程接口是有目的的简单的。但是，有了它，你可以实现更高阶的操作，例如同步原语，组成员资格，所有权等。一些分布式应用程序已经使用它来:

## 1.9 性能

ZooKeeper被设计为高性能的。但是呢？Yahoo！ZooKeeper开发团队的成果研究表明它是。在读取数量大于写入的应用程序中，性能特别高，因为写入操作设计到同步所有服务器的状态。

ZooKeeper Throughput as the Read-Write Ratio Varies



ZooKeeper吞吐量作为读写比率的变化是ZooKeeper发布的3.2对2Ghz至强服务器和两个SATA 15K RPM驱动器运行吞吐量图。一个驱动器作为一个专门的管理员登录器。快照被写入OS驱动器。写请求被1k写入和读取1k。服务器表名ZooKeeper集合的大小，服务器组成的服务数量。大约30个其他服务器被用来模拟客户机。ZooKeeper集成配置等，Leader不允许来自客户端的连接。

Note:

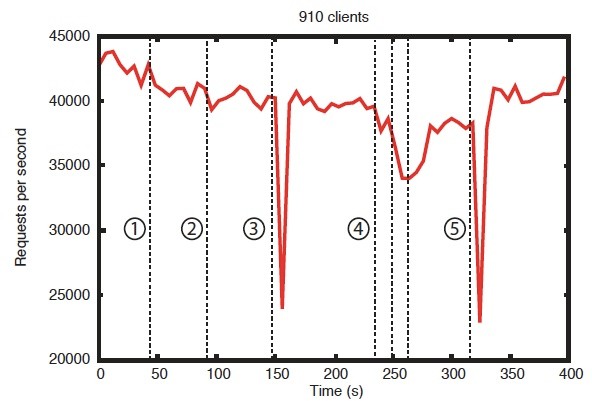
In version 3.2 r/w performance improved by ~2x compared to the [previous 3.1 release](http://zookeeper.apache.org/docs/r3.1.1/zookeeperOver.html#Performance).

Benchmarks 也表名它也是可靠的。存在错误时的可靠性显示了部署对各种故障的响应方式。图中标出的事件如下:

1. Follow的失败和恢复
2. 不同的Follow的故障与恢复
3. Leader失败
4. 两个Follow的失败和恢复
5. 另一个Leader失败

## 1.10 可靠性

显示系统在发生故障的行为注入，我们跑了7台ZooKeeper服务器。我们和以前运行的是相同的饱和基准，但是这次我们将写入百分比保持在恒定的30%，这是我们预期工作负载的一个保守的比率。



Reliability in the Presence of Errors

这是这个图的一些重要的观察结果。首先，如果follow失败和恢复的很快，ZooKeeper能够维持高吞吐量尽管失败，但也许更重要的是，leader选举算法允许系统恢复足够快，以防止吞吐量大幅下降。在我们的观察中，管理员需要小玉200ms选出新leader。第三，作为追随者恢复，ZooKeeper能够再次提高吞吐量，一旦他们开始处理请求。

## 1.11 ZooKeeper 项目

ZooKeeper以成功地应用在许多工业应用。它在Yahoo上使用！作为Yahoo的协调和故障恢复服务！消息代理，它是高度可伸缩的发布订阅系统，管理数以千计的topic用于赋值和数据传输。它由Yahoo的抓取服务使用！它还负责管理故障恢复。一些Yahoo！广告系统也用它来实现可靠的服务。

鼓励所有用户和开发人员加入社区并贡献他们的专业知识。更多信息见ApacheZooKeeper项目。