走向架构师之路

分布式设计与开发（三）------高一致性服务ZooKeeper

标签： 分布式应用 string null 服务器 byte 算法

2010-08-18 22:57 40223人阅读 评论(9) 收藏 举报

分类： 技术积累（62）

分布式环境中大多数服务是允许部分失败，也允许数据不一致，但有些最基础的服务是需要高可靠性，高一致性的，这些服务是其他分布式服务运转的基础，比如naming service、分布式lock等，这些分布式的基础服务有以下要求：

- 高可用性
- 高一致性
- 高性能

对于这种有些挑战CAP原则 的服务该如何设计，是一个挑战，也是一个不错的研究课题，Apache的ZooKeeper也许给了我们一个不错的答案。ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务， 它暴露了一个简单的原语集，分布式应用程序可以基于它实现同步服务，配置维护和命名服务等。关于ZooKeeper更多信息可以参见 官方文档

ZooKeeper的基本使用

搭一个分布式的ZooKeeper环境比较简单，基本步骤如下：

1）在各服务器安装 ZooKeeper

下载ZooKeeper后在各服务器上进行解压即可

tar -xzf zookeeper-3.2.2.tar.gz

2）配置集群环境

分别各服务器的zookeeper安装目录下创建名为zoo.cfg的配置文件，内容填写如下：

```
[xhtml]
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
dataDir=/home/admin/zookeeper-3.2.2/data
# the port at which the clients will connect
clientPort=2181
server.1=zoo1:2888:3888
server.2=zoo2:2888:3888
```

(56537)
分布式设计与开发（二）
(47555)
Tomcat源码分析（一）--
(44363)
JVM学习笔记（三）------
(42108)
构建高性能web之路------
(42029)
分布式设计与开发（三）
(40210)

评论排行

迈向架构师的第一步 (77)
分布式设计与开发（一） (42)
单点登录SSO的实现原理 (40)
在AMD的WIN7上搭建IOS (38)
JVM学习笔记（一）------ (28)
JVM学习笔记（四）------ (23)
JVM学习笔记（三）------ (22)
phonegap源码分析（一） (22)
从Jetty、Tomcat和Mina (20)
最后一天 (19)

推荐文章

*Android自定义ViewGroup打造各种风格的SlidingMenu
* Android 6.0 运行时权限处理完全解析
* 数据库性能优化之SQL语句优化
*Animation动画详解(七)——ObjectAnimator基本使用
* Chromium网页URL加载过程分析
* 大数据三种典型云服务模式

最新评论

迈向架构师的第一步
haryhouqin: 加油！
IT土鳖混外企（二）------ 语言障
神的理想: 不错，15000词放口袋
架构师的楷模
youngke: 加油，学习中
JVM学习笔记（二）------Java代
brianway: 思路清晰，几张图画的挺好的，学习了。
探索WebKit内核（一）------ 菜鸟
l360220954: 楼主，可不可以给一个详尽版的编译方法呢，编译了好几次，用了好几种在网上找的方法，有一个错误实在过不去...
探索WebKit内核（一）------ 菜鸟
l360220954: @NiYongYuanDeErDuo:我也要编译这个了，请问你的问题解决得怎么样了呀？
Hessian源码分析（三）------ He
zwllxs: io部分我可以 补上，并且还改写过大量io下的包，扩充了序列化和反序列化模块
从Jetty、Tomcat和Mina中提炼N
yange102688: 假如一个场景，用户上传文件，某些用户网速较慢，同时存在100个这样的用户，如果BIO且最大线程设为1...
单点登录SSO的实现原理
笑破苍穹: 调理清晰
基于libevent, libuv和android Loc
yangchangeu: epoll是同步非阻塞I/O

其中zoo1和zoo2分别对应集群中各服务器的机器名或ip，server.1和server.2中1和2分别对应各服务器的zookeeper id，id的设置方法为在dataDir配置的目录下创建名为myid的文件，并把id作为其文件内容即可，在本例中就分为设置为1和2。其他配置具体含义可见官方文档。

3）启动集群环境

分别在各服务器下运行zookeeper启动脚本

/home/admin/zookeeper-3.2.2/bin/zkServer.sh start

4）应用zookeeper

应用zookeeper可以在是shell中执行命令，也可以在java或c中调用程序接口。

在shell中执行命令，可运行以下命令：

bin/zkCli.sh -server 10.20.147.35:2181

其中 10.20.147.35为集群中任一台机器的ip或机器名。执行后可进入zookeeper的操作面板，具体如何操作可见官方文档

在java中通过调用程序接口来应用zookeeper较为复杂一点，需要了解watch和callback等概念，不过试验最简单的CURD倒不需要这些，只需要使用ZooKeeper这个类即可，具体测试代码如下：

```
[java]
01. public static void main(String[] args) {
02.     try {
03.         ZooKeeper zk = new ZooKeeper("10.20.147.35:2181", 30000, null);
04.         String name = zk.create("/company", "alibaba".getBytes(),
05.                 Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT_SEQUENTIAL);
06.         Stat stat = new Stat();
07.         System.out.println(new String(zk.getData(name, null, stat)));
08.         zk.setData(name, "taobao".getBytes(), stat.getVersion(), null, null);
09.         System.out.println(new String(zk.getData(name, null, stat)));
10.         stat = zk.exists(name, null);
11.         zk.delete(name, stat.getVersion(), null, null);
12.         System.out.println(new String(zk.getData(name, null, stat)));
13.     } catch (Exception e) {
14.         e.printStackTrace();
15.     }
16. }
```

以上代码比较简单，查看一下zooKeeper的api doc就知道如何使用了

ZooKeeper的实现机理

ZooKeeper的实现机理是我看过的开源框架中最复杂的，它的解决是分布式环境中的一致性问题，这个场景也决定了其实现的复杂性。看了两三天的源码还是有些摸不着头脑，有些超出了我的能力，不过通过看文档和其他高人写的文章大致清楚它的原理和基本结构。

1）ZooKeeper的基本原理

ZooKeeper是以Fast Paxos算法为基础的，在前一篇 blog 中大致介绍了一下paxos，而没有提到的是paxos存在活锁的问题，也就是当有多个 proposer交错提交时，有可能互相排斥导致没有一个proposer能提交成功，而Fast Paxos作了一些优化，通过选举产生一个leader，只有leader才能提交propose，具体算法可见Fast Paxos 。因此，要想弄得ZooKeeper首先得对Fast Paxos有所了解。

2）ZooKeeper的基本运转流程

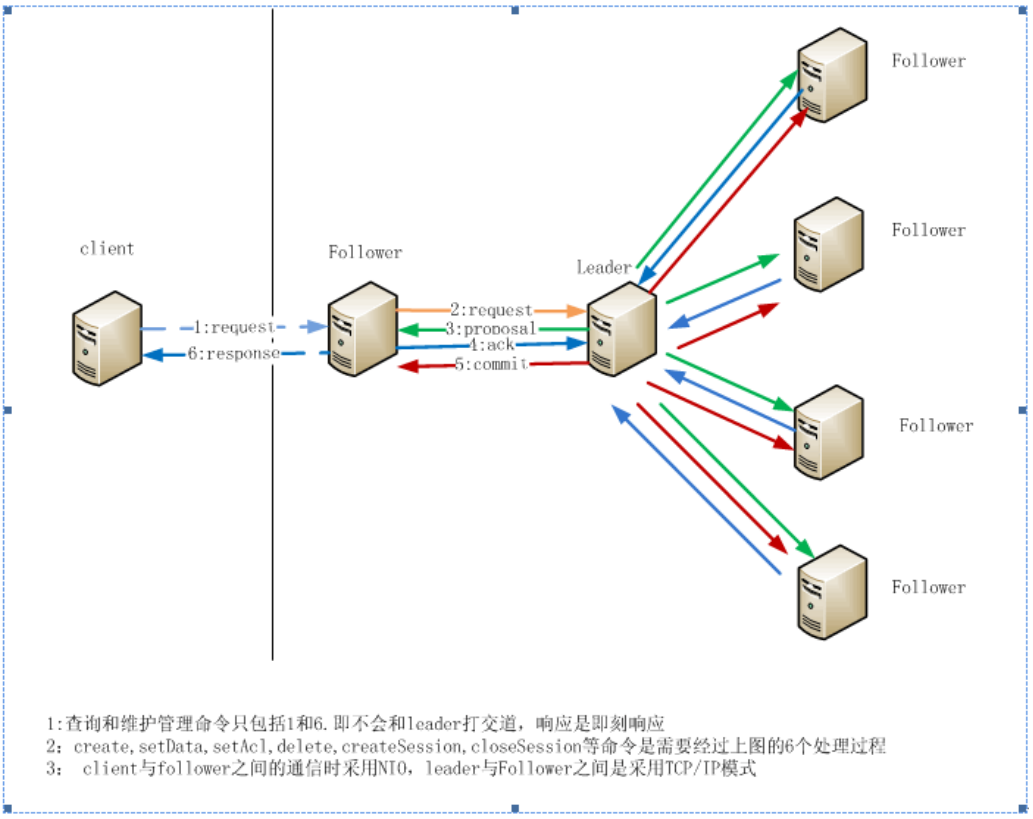
ZooKeeper主要存在以下两个流程：

- 选举Leader
- 同步数据

选举Leader过程中算法有很多，但要达到的选举标准是一致的：

- Leader要具有最高的zxid
- 集群中大多数的机器得到响应并follow选出的Leader

同步数据这个流程是ZooKeeper的精髓所在，并且就是Fast Paxos算法的具体实现。一个牛人画了一个ZooKeeper数据流动图，比较直观地描述了ZooKeeper是如何同步数据的。



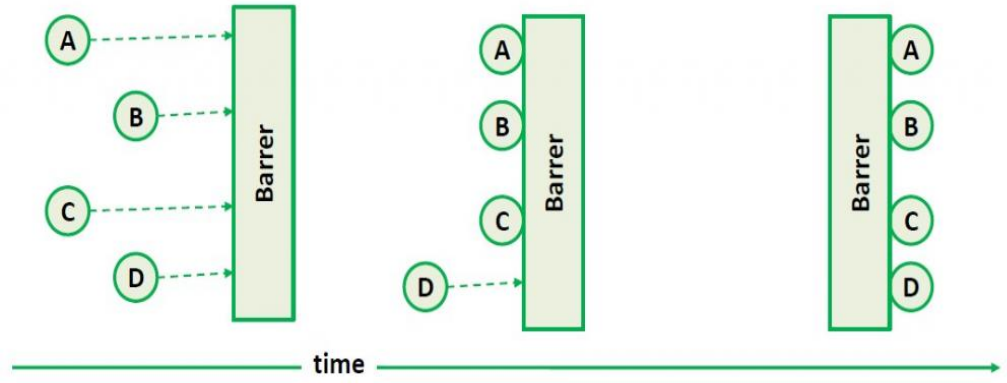
以上两个核心流程我暂时还不能悟透其中的精髓，这也和我还没有完全理解Fast Paxos算法有关，有待后续深入学习

ZooKeeper的应用领域

Tim在blog中提到了Paxos所能应用的几个主要场景，包括database replication、naming service、config配置管理、access control list等等，这也是ZooKeeper可以应用的几个主要场景。此外， ZooKeeper官方文档中提到了几个更为基础的分布式应用，这也算是ZooKeeper的妙用吧

1）分布式Barrier

Barrier是一种控制和协调多个任务触发次序的机制，简单说来就是搞个闸门把欲执行的任务给拦住，等所有任务都处于可以执行的状态时，才放开闸门。它的机理可以见下图所示：



在单机上JDK提供了CyclicBarrier这个类来实现这个机制，但在分布式环境中JDK就无能为力了。在分布式里实现Barrer需要高一致性做保障，因此 ZooKeeper可以派上用场，所采取的方案就是用一个Node作为Barrer的实体，需要被Barrer的任务通过调用exists()检测这个Node的存在，当需要打开Barrier的时候，删掉这个Node，ZooKeeper

的watch机制会通知到各个任务可以开始执行。

2） 分布式 Queue

与 Barrier类似 分布式环境中 实现Queue也需要高一致性做保障， ZooKeeper提供了一个种简单的方式，ZooKeeper通过一个Node来维护Queue的实体，用其children来存储Queue的内容，并且 ZooKeeper的create方法中提供了顺序递增的模式，会自动地在name后面加上一个递增的数字来插入新元素。可以用其 children来构建一个queue的数据结构，offer的时候使用create，take的时候按照children的顺序删除第一个即可。 ZooKeeper保障了各个server上数据是一致的，因此也就实现了一个 分布式 Queue。take和offer的实例代码如下所示：

```java
/**
 * Removes the head of the queue and returns it, blocks until it succeeds.
 * @return The former head of the queue
 * @throws NoSuchElementException
 * @throws KeeperException
 * @throws InterruptedException
 */
public byte[] take() throws KeeperException, InterruptedException {
    TreeMap<Long,String> orderedChildren;
    // Same as for element.  Should refactor this.
    while(true){
        LatchChildWatcher childWatcher = new LatchChildWatcher();
        try{
            orderedChildren = orderedChildren(childWatcher);
        }catch(KeeperException.NoNodeException e){
            zookeeper.create(dir, new byte[0], acl, CreateMode.PERSISTENT);
            continue;
        }
        if(orderedChildren.size() == 0){
            childWatcher.await();
            continue;
        }
        for(String headNode : orderedChildren.values()){
            String path = dir +"/"+headNode;
            try{
                byte[] data = zookeeper.getData(path, false, null);
                zookeeper.delete(path, -1);
                return data;
            }catch(KeeperException.NoNodeException e){
                // Another client deleted the node first.
            }
        }
    }
}
/**
 * Inserts data into queue.
 * @param data
 * @return true if data was successfully added
 */
public boolean offer(byte[] data) throws KeeperException, InterruptedException{
    for(;;){
        try{
            zookeeper.create(dir+"/"+prefix, data, acl, CreateMode.PERSISTENT_SEQUENTIAL);
            return true;
        }catch(KeeperException.NoNodeException e){
            zookeeper.create(dir, new byte[0], acl, CreateMode.PERSISTENT);
        }
    }
}
```

3）分布式lock

利用 ZooKeeper实现 分布式lock，主要是通过一个Node来代表一个Lock，当一个client去拿锁的时候，会在这个Node下创建一个自增序列的child，然后通过getChildren()方式来check创建的child是不是最靠前的，如果是则拿到锁，否则就调用exist()来check第二靠前的child，并加上watch来监视。当拿到锁的child执行完后归还锁，归还锁仅仅需要删除自己创建的child，这时watch机制会通知到所有没有拿到锁的client，这些child就会根据前面所讲的拿锁规则来竞争锁。

顶 2　踩 0

上一篇 分布式设计与开发（二）------几种必须了解的分布式算法
下一篇 基于Jupiter建立code review机制

我的同类文章

技术积累（62）

- 探索WebKit内核（三）------ … 2013-05-12 阅读 10721
- 探索WebKit内核（二）------ … 2013-04-28 阅读 18249
- 从Samples中入门IOS开发… 2013-01-15 阅读 8386
- 从Samples中入门IOS开发… 2013-01-14 阅读 8272
- 从Samples中入门IOS开发… 2013-01-14 阅读 13958
- 从Samples中入门IOS开发… 2013-01-12 阅读 8463
- 从Samples中入门IOS开发… 2013-01-11 阅读 8191
- 从Samples中入门IOS开发… 2013-01-10 阅读 8005
- 从几个sample来学习Java… 2012-12-12 阅读 16470
- 推荐近年来印象最深的几本书 2012-11-08 阅读 7541

更多文章

主题推荐 apache 分布式 zookeeper 设计 分布式应用 服务器

猜你在找

数据结构和算法
有趣的算法（数据结构）
以性别预测为例，谈谈数据挖掘中常见的分类算法
传统IT环境与PaaS环境下的应用开发模式
《C语言/C++学习指南》加密解密篇（安全相关算法）
分布式与集群的区别
分布式与集群的区别
分布式与集群的区别
java分布式学习
关于Mongodb的全面总结学习mongodb的人可以从这里开

查看评论

8楼 jiang2011jiang 2014-07-22 14:43发表

Thanks!

7楼 mgjmuying 2014-05-12 15:33发表


看了楼主的博客，思路清晰明白，让我对分布式很多的知识加深了理解。请问下您的花名是？谢谢！

6楼 一见 2013-04-21 19:37发表


赞，介绍的不错

5楼 showdy1984 2013-03-19 17:59发表

ZooKeeper 看不太明白，目的是大致了解了。。实现原理看不懂。

4楼 phoenix2xp 2012-07-24 09:53发表

分布式开发还是复杂啊

3楼 fh13760184 2012-01-04 18:03发表

NX

2楼 wujianjun1219 2011-05-10 16:35发表

[e01]

1楼 j_now 2010-11-16 12:26发表

[e01]

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap