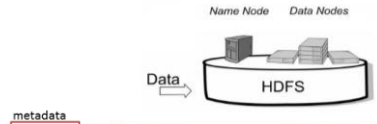
第九讲 协调服务系统
ZooKeeper

徐 辰
cxu@dase.ecnu.edu.cn
華東師範大學

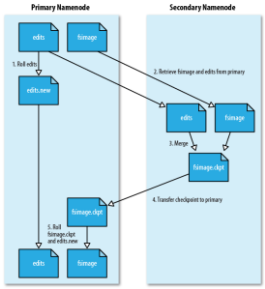
HDFS 1.0

名称节点保存元数据：
（1）在磁盘上：FsImage和EditLog
（2）在内存中：映射信息，即文件包含哪些块，每个块存储在哪个数据节点



NameNode	DataNode
• 存储元数据	• 存储文件内容
•元数据保存在内存中	•文件内容保存在磁盘
• 保存文件,block， datanode之间的映射关系	•维护了block id到datanode本地文件的映射关系

HDFS 1.0 单点故障

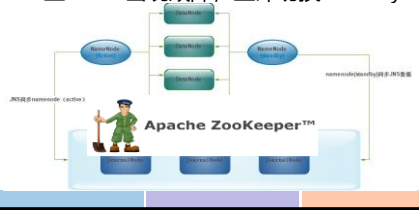


•SecondaryNameNode会定期和NameNode通信
•从NameNode上获取到FsImage和EditLog文件，并下载到本地的相应目录下
•执行EditLog和FsImage文件合并
•将新的FsImage文件发送到NameNode节点上
•NameNode使用新的FsImage和EditLog（缩小了）

第二名称节点用途：
•不是热备份
•主要是防止日志文件EditLog过大，导致名称节点失败恢复时消耗过多时间
•附带起到冷备份功能

HDFS HA (High Availability)

- NameNode之间的数据同步
  - 借助共享存储系统或JNs(Journal Nodes)实现
- NameNode之间的状态感知
  - 一旦Active出现故障，立即切换Standby

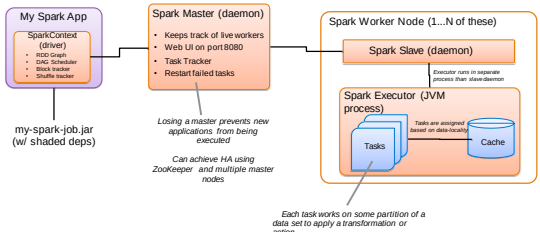


配置多个NameNode

- 问题：谁会成为active?

```
<!-- ns下面有两个NameNode，分别是nn1, nn2 -->
<property>
  <name>dfs.ha.namenodes.ns</name>
  <value>nn1,nn2</value>
</property>
<!-- nn1的RPC通信地址 -->
<property>
  <name>dfs.namenode.rpc-address.ns.nn1</name>
  <value>master1:9000</value>
</property>
<!-- nn1的http通信地址 -->
<property>
  <name>dfs.namenode.http-address.ns.nn1</name>
  <value>master1:50070</value>
</property>
<!-- nn2的RPC通信地址 -->
<property>
  <name>dfs.namenode.rpc-address.ns.nn2</name>
  <value>master2:9000</value>
</property>
<!-- nn2的http通信地址 -->
<property>
  <name>dfs.namenode.http-address.ns.nn2</name>
  <value>master2:50070</value>
</property>
```

Spark HA (High Availability)



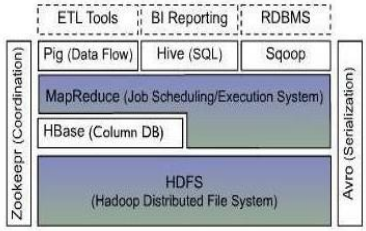
ZooKeeper简介

7

- ZooKeeper：轻量级的分布式系统
- 作用：用于解决分布式应用中通用的协作问题
  - 命名管理 Naming
  - 配置管理 Configuration Management
  - 集群管理 Group Membership
  - 同步管理 Synchronization

ZooKeeper的广泛使用

8



大纲

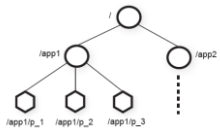
9

- 设计思想
  - 数据模型
  - 操作原语
- 体系架构
- 工作原理
- 容错机制
- 典型应用

数据模型

10

- 维护类似文件系统的层次数据结构
- Znode：子目录项
  - Znode doesn't not design for data storage, instead it store meta-data or configuration
  - Can store information like timestamp version
- Znode持久性
  - Regular(常规)
  - Ephemeral(临时)
- Znode顺序性
  - Sequential flag



Znode

11

- 是否会自动删除
  - Regular Znode: 用户需要显式的创建、删除
  - Ephemeral Znode: 用户创建后，可以显式的删除，也可以在Session结束后，由ZooKeeper Server自动删除
- 是否带有顺序号
  - 如果创建的时候指定Sequential，该Znode的名字后面会自动Append一个不断增加的SequenceNo

Znode的四种类型

12

- **PERSISTENT**(持久化节点)：客户端断开连接后，该节点依旧存在
- **PERSISTENT_SEQUENTIAL**(持久化顺序编号节点)：客户端断开连接后，该节点依旧存在；Zookeeper给该节点名称进行顺序编号
- **EPHEMERAL**(临时节点)：客户端与zookeeper断开连接后，该节点被删除
- **EPHEMERAL_SEQUENTIAL**(临时顺序编号节点)： 客户端断开连接后，该节点被删除；Zookeeper给该节点名称进行顺序编号

大纲

13

- 设计思想
  - 数据模型
  - 操作原语
- 体系架构
- 工作原理
- 容错机制
- 典型应用

Client API

14

- create(path, data, flags)
- delete(path, version)
- exist(path, watch)
- getData(path, watch)
- setData(path, data, version)
- getChildren(path, watch)
- sync(path)
  - Two version synchronous and asynchronous

常用API含义

15

- create: 在树中某一位置添加一个Znode
- delete: 删除某一Znode
- exists: 判断某一位置是否存在Znode
- get data: 从某一Znode读取数据
- set data: 向某一Znode写入数据
- get children: 查找某一Znode的子节点
- sync: 用于等待数据同步

大纲

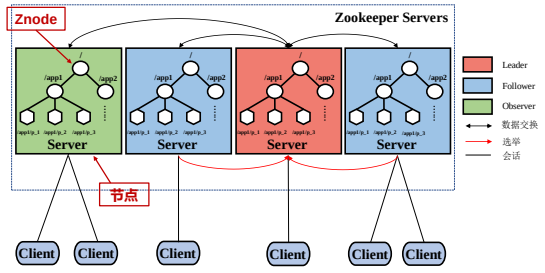
16

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型应用

架构图

17

- 与MapReduce、Spark架构图有很大区别



服务器

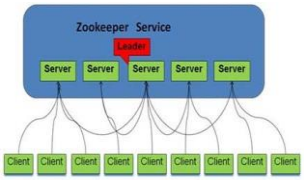
18

- 服务器Server：每台服务器都维护一份树形结构数据的备份
  - 领导者Leader：参与选主过程的服务器通过一定的算法选定一个节点作为领导者，领导者节点可以为客户端直接提供读、写服务。
  - 追随者Follower：追随者仅直接提供读服务，客户端发给追随者的写操作要将转发给领导者
  - 观察者Observer：与追随者类似，区别在于观察者不参与选举领导者的过程。观察者的角色是可选的，服务器中可以没有观察者节点

客户端

19

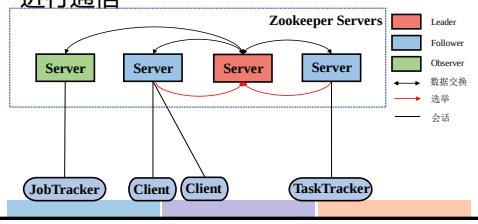
- 客户端通过执行前述API来操纵ZooKeeper中维护的数据
- 问题：谁是Client?



客户端

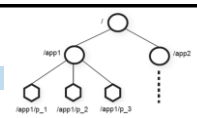
20

- MapReduce中JobTracker和TaskTracker作为ZooKeeper的客户端
- JobTracker和TaskTracker利用ZooKeeper进行通信



会话机制

21



- Client
  - Client可以在某个Znode上设置一个Watcher，来跟踪该Znode上的变化
- Server:
  - 一旦这个Znode中存储的数据的修改，子Znode的变化等，可以通知设置监控的客户端
- Session
  - A connection to server from client is a session
  - Timeout mechanism

大纲

22

- 设计思想
- 体系架构
- 工作原理
  - 领导者选举
  - 读写请求流程
- 容错机制
- 典型应用

为什么需要选领导者？

23

- ZooKeeper从某种意义上说是一个轻量级的数据存储系统，维护了数据的多个副本
- 由于写操作会改变数据的内容，所以必须保证每个节点都执行相同的操作序列，从而达到副本之间的一致性。
- ZooKeeper中需要有领导者节点来保证各节点执行相同的写操作序列。如果没有领导者节点，那么难以保证各节点副本之间的一致性。

如何进行领导者选举？

24

- ZooKeeper工作过程中最重要的问题是如何在服务器之间确定领导者，确定了领导者之后才可以进行读写操作
- 分布式一致性协议是用来解决分布式系统如何就某个提议达成一致的问题
  - 经典的实现分布式一致性协议的算法有Paxos等。这些算法的基本思想都是由某些节点发出提议，再由其它节点进行投票表决，最终所有节点达成一致
  - 本课程不深入讨论^_^

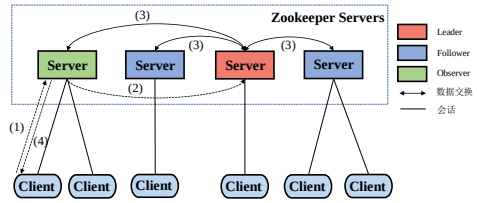
大纲

25

- 设计思想
- 体系架构
- 工作原理
  - 领导者选举
  - 读写请求流程
- 容错机制
- 典型应用

写请求流程

26



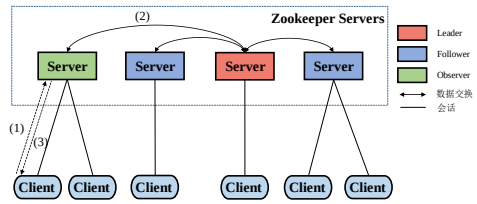
写请求流程

27

1. 客户端与某一服务器建立连接发起写请求
2. 若服务器是追随者或观察者，则将接收到写请求转发给领导者，否则领导者直接处理
3. 数据同步
   - 理论上，领导者要将写请求发给所有服务器，直到所有服务器都成功执行了写操作，该写请求才算是完成
   - 事实上，保证半数以上的节点写操作成功就可以认为写请求完成。这一过程实际上是追随者、观察者与领导者之间进行数据同步的过程，该过程需要使用分布式一致性协议
4. 数据返回

读请求流程

28



读请求流程

29

1. 当客户端发起读数据请求时，无论是领导者、追随者还是观察者都可以响应请求。
2. 数据同步（可选）
   - 为了读取最新的数据，客户端可以调用sync()操作用于等待追随者或观察者与领导者进行数据同步。之后，客户端发起的读请求将得到最新的数据。
3. 数据返回
   - 如果是领导者响应该请求，那么返回给客户端的数据必然是最新的
   - 如果是追随者或观察者响应请求，那么直接返回给客户端的数据可能不是最新的，需执行第2步得到最新的数据

大纲

30

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型应用

故障类型

31

- 领导者节点故障：ZooKeeper需要重新进行领导者选举。
- 追随者或观察者节点故障：该节点无法对外提供服务，但其它节点依然可以正常提供服务，所以不影响ZooKeeper的服务。
  - 如果追随者或观察者节点发生故障后进行重启，那么它们将可从领导者节点或其它节点进行数据恢复。

大纲

32

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型应用

ZooKeeper典型应用场景

33

- 命名管理 Naming
- 配置管理 Configuration Management
- 集群管理 Group Membership
  - 状态监控 Status Monitoring
  - 选主 Leader Election
- 同步管理 Synchronization
  - 共享锁 Locks
  - 同步队列 Synchronous Queue
  - 双屏障 Double Barrier

命名管理

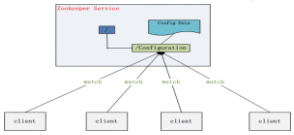
34

- 统一命名服务：树形的名称结构
  - 域名解析：某一域名的IP地址是多少?
  - HDFS目录与文件：某一文件存储在哪里?
- Name Service已经是Zookeeper内置的功能，只要调用Zookeeper的API 就能实现
  - 如调用create接口就可以创建一个Znode
  - 利用Znode存储信息，用于实现相应功能

配置管理

35

- 场景：用户命令行修改了默认的参数
- 方法：
  - 配置信息存在 ZooKeeper 某个目录节点，所有机器watch该目录节点
  - 一旦发生变化，每台机器收到通知，然后从ZooKeeper获取新的配置信息应用到系统中



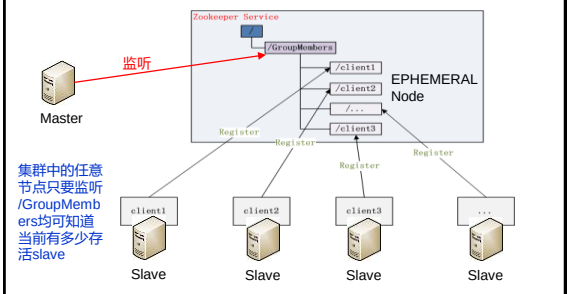
集群管理

36

- 监控集群中slave状态：新增或减少
  - Create a Znode g
  - Each process create a Znode under g in EPHEMERAL mode
  - Watch g for group information
- 从多个master选主(Leader Election)
  - Create a Znode g　注意与ZooKeeper本身选Leader的区别
  - Each process create a Znode under g in EPHEMERAL_SEQUENTIAL mode
  - Watch g and choose the one with minimal No.

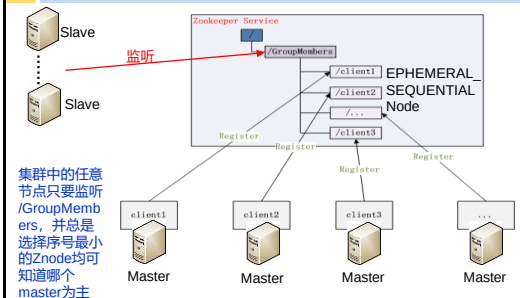
监控slave节点状态

37



多master选主

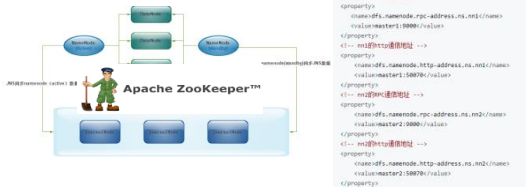
38



HDFS HA (High Availability)

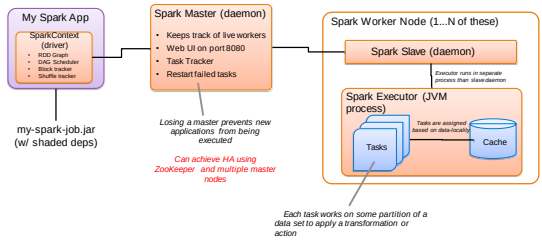
39

- 一旦Active出现故障，立即切换Standby，谁会成为active?



Spark HA

40



Double Barrier双屏障

41

- 允许客户端在计算的开始和结束时同步。当足够的进程加入到双屏障时，进程开始计算。当计算完成时，离开屏障。
- 实现方法
  - 进入屏障：创建/ready和/process节点，每个进程都先到ready中注册，注册数量达到要求时，通知所有进程启动开始执行
  - 离开屏障：在/process下把注册ready的进程都建立节点，每个进程执行结束后删掉/process下对应节点。当/process为空时，任务全结束

课后阅读（研究生）

42

- 分布式系统概念与设计，George Coulouris等著，金蓓弘等译
  - 第13、15章
- 论文
  - Hunt, P., Konar, M., Junqueira, F. P., & Reed, B. (2010). ZooKeeper : Wait-free coordination for Internet-scale systems. In USENIX Annual Technology Conference (pp. 1–14).

本讲小结

43

- 设计思想
- 体系架构
- 工作原理
- 容错机制
- 典型应用

Apache ZooKeeper™

谢谢！Q&A