高级Hadoop 2.x（一）

谭唐华

课程大纲

- 分布式集群部署
- Zookeeper部署
- HA集群搭建

分布式集群安装

- 基于伪分布式环境安装进行展开
- 规划机器与服务（★★★★☆）
  - HDFS 文件系统
  - YARN “云操作系统”
  - JobHistoryServer 历史服务监控
- 修改配置文件，设置服务运行机器节点（★★★☆☆）
- 分发HADOOP安装包至各个机器节点
- 依据官方集群安装文档，分别启动各节点相应服务
- 测试 HDFS 、YARN、 MapReduce ， Web UI 监控集群（★★★☆☆）

配置主节点至各从节点 SSH 无密钥登陆

实时演练【搭建Hadoop 2.x分布式环境】

- 系统基本环境配置
  - step 1: 创建普通用户和设置密码
  - step 2: 设置系统主机名称
  - step 3: 设置主机名与IP地址映射
  - step 4: 关闭防火墙和禁用Selinux
  - step 5: 卸载系统自带Open JDK

实时演练【搭建Hadoop 2.x分布式环境】

- HDFS部署启动测试
  - step 1: 配置NTP时间同步
  - step 2: 安装JDK和配置环境变量
  - step 3: 对HDFS进行配置
  - step 4: 启动HDFS守护进程（NameNode和DataNode）
  - step 5: 测试HDFS（读写文件和WEB UI监控）

实时演练【搭建Hadoop 2.x分布式环境】

- YARN部署启动及运行MapReduce程序
  - step 1: 对YARN进行配置
  - step 2: 启动YARN守护进程（RM 和 NM）
  - step 3: YARN的WEB UI监控
  - step 4: 准备数据和在YARN上运行WordCount案例
  - step 5: 配置JobHistoryServer并启动监控应用

HADOOP练习：分布式部署

- 作业：实现分布式部署安装

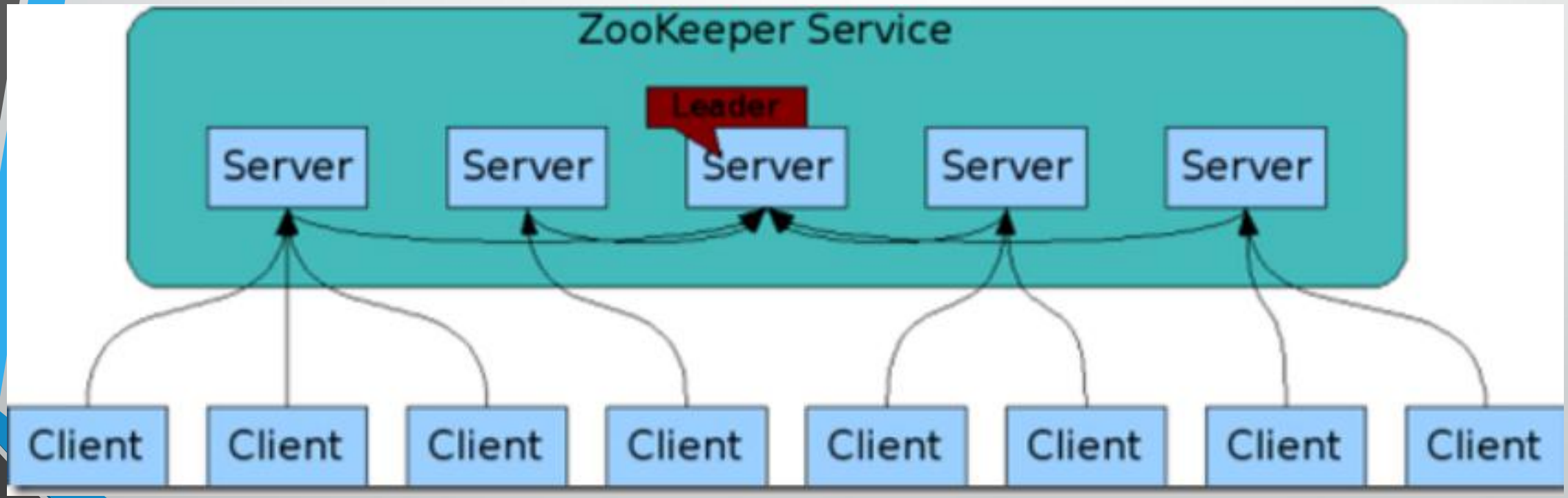
课程大纲

- 分布式集群部署
- Zookeeper部署
- HA集群搭建

什么是 ZooKeeper?

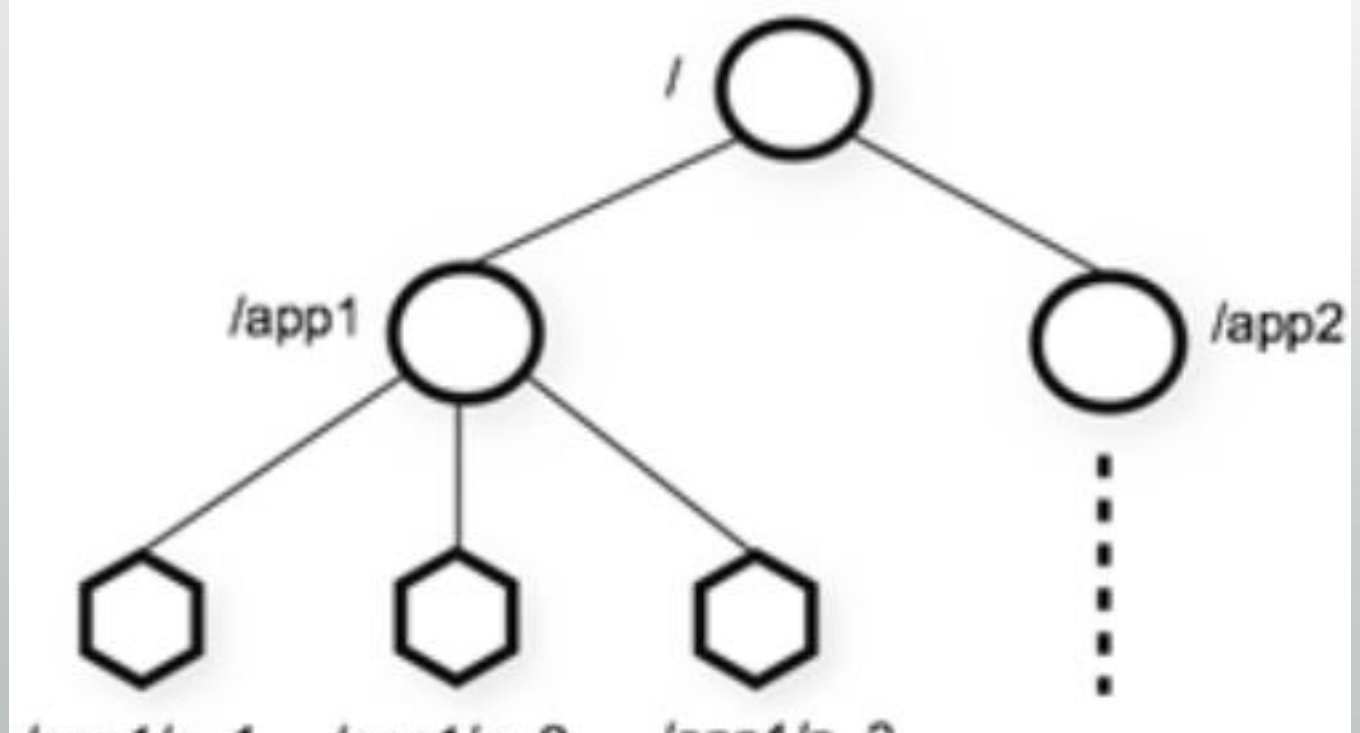
- 一个开源的分布式的，**为分布式应用提供协调服务**的Apache项目。
- 提供一个简单的原语集合，以便于分布式应用可以在它之上构建更高层次的同步服务。
- 设计非常易于编程，它使用的是类似于文件系统那样的树形数据结构。
- 目的就是将分布式服务不再需要由于协作冲突而另外实现协作服务。

ZooKeeper Service



Zookeeper数据结构

- 数据结构和分等级的命名空间
  - Zookeeper的命名空间的结构和文件系统很像。一个名字和文件一样使用/的路径表现，zookeeper的每个节点都是被路径唯一标识。
  - ZooKeeper's Hierarchical Namespace



Zookeeper 角色

角色		描述
领导者（Leader）		领导者负责进行投票的发起和决议，更新系统状态
学习者（Learner）	跟随者（Follower）	Follower 用于接收客户请求并向客户端返回结果，在选主过程中参与投票
	观察者（ObServer）	ObServer 可以接收客户端连接，将写请求转发给 leader 节点。但 ObServer 不参加投票过程，只同步 leader 的状态。ObServer 的目的是为了扩展系统，提高读取速度
客户端（Client）		请求发起方

ZooKeeper 典型应用场景

- Zookeeper 从设计模式角度来看，是一个**基于观察者模式设计**的分布式服务管理框架，它**负责存储和管理**大家都关心的数据，然后接受**观察者的注册**，一旦这些数据的状态发生变化，Zookeeper 就将负责通知已经在 Zookeeper 上注册的那些**观察者**做出相应的反应，从而实现集群中**类似 Master/Slave 管理模式**。
- 应用场景
  - 统一命名服务（Name Service）
  - 配置管理（Configuration Management）
  - 集群管理（Group Membership）
  - 共享锁（Locks）/同步锁

Zookeeper 单机模式安装

- 安装JDK、配置环境变量、验证java –version
- 下载、赋执行权限、解压
  - 下载地址：http://apache.dataguru.cn/zookeeper/
  - 权限：chmod u+x zookeeper-3.4.5.tar.gz
  - 解压：tar -zxvf zookeeper-3.4.5.tar.gz -C /opt/modules/
- 配置
  - 复制配置文件：cp conf/zoo_sample.cfg conf/zoo.cfg
  - 配置数据存储目录：dataDir=/opt/modules/zookeeper-3.4.5/data
  - 创建数据存储目录：mkdir /opt/modules/zookeeper-3.4.5/data
- 启动
  - 启动：bin/zkServer.sh start

检测

Client Shell：bin/zkCli.sh 或者 查看状态：bin/zkServer.sh status

Zookeeper 配置参数详解

- 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。
- Zookeeper Client 命令讲解
  - 命令：bin/zkCli.sh -server localhost:2181
  - 详解：ls、get、create、delete

Zookeeper 分布式安装

- 在单机模式基础之上，修改配置文件conf/zoo.cfg
  - vi conf/zoo.cfg
  - 内容如下：

    initLimit=5

    syncLimit=2

    server.1=192.168.48.128:2888:3888

    server.2=192.168.48.181:2888:3888

    server.3=192.168.48.1822:2888:3888

- 在各个机器安装的数据存储目录下创建myid文件
  - 命令：touch data/myid
  - 编辑：vi data/myid
  - 内容（不同机器，不一样）：

    192.168.48.128：1

    192.168.48.128：2

    192.168.48.128：3

Zookeeper 配置参数详解续

- initLimit：这个配置项是用来配置 Zookeeper 接受客户端（这里所说的客户端不是用户连接 Zookeeper 服务器的客户端，而是 Zookeeper 服务器集群中连接到 Leader 的 Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过 10 个心跳的时间（也就是 tickTime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败。总的时间长度就是 5*2000=10 秒。
- syncLimit：这个配置项标识 Leader 与 Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是 2*2000=4 秒。

Zookeeper 配置参数详解续

- **server.A=B:C:D**：其中 A 是一个数字，表示这个是第几号服务器；B 是这个服务器的 ip 地址；C 表示的是这个服务器与集群中的 Leader 服务器交换信息的端口；D 表示的是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。如果是伪集群的配置方式，由于 B 都是一样，所以不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号。
- 集群模式下配置一个文件 myid，这个文件在 dataDir 目录下，这个文件里面就**有一个数据就是 A 的值**，Zookeeper 启动时读取此文件，拿到里面的数据与 zoo.cfg 里面的配置信息比较从而判断到底是那个 server。

Zookeeper 访问方式-Shell命令

```
ZooKeeper -server host:port cmd args
        connect host:port
        get path [watch]
        ls path [watch]
        set path data [version]
        rmr path
        delquota [-n|-b] path
        quit
        printwatches on|off
        create [-s] [-e] path data acl
        stat path [watch]
        close
        ls2 path [watch]
        history
        listquota path
        setAcl path acl
        getAcl path
        sync path
        redo cmdno
        addauth scheme auth
        delete path [version]
```

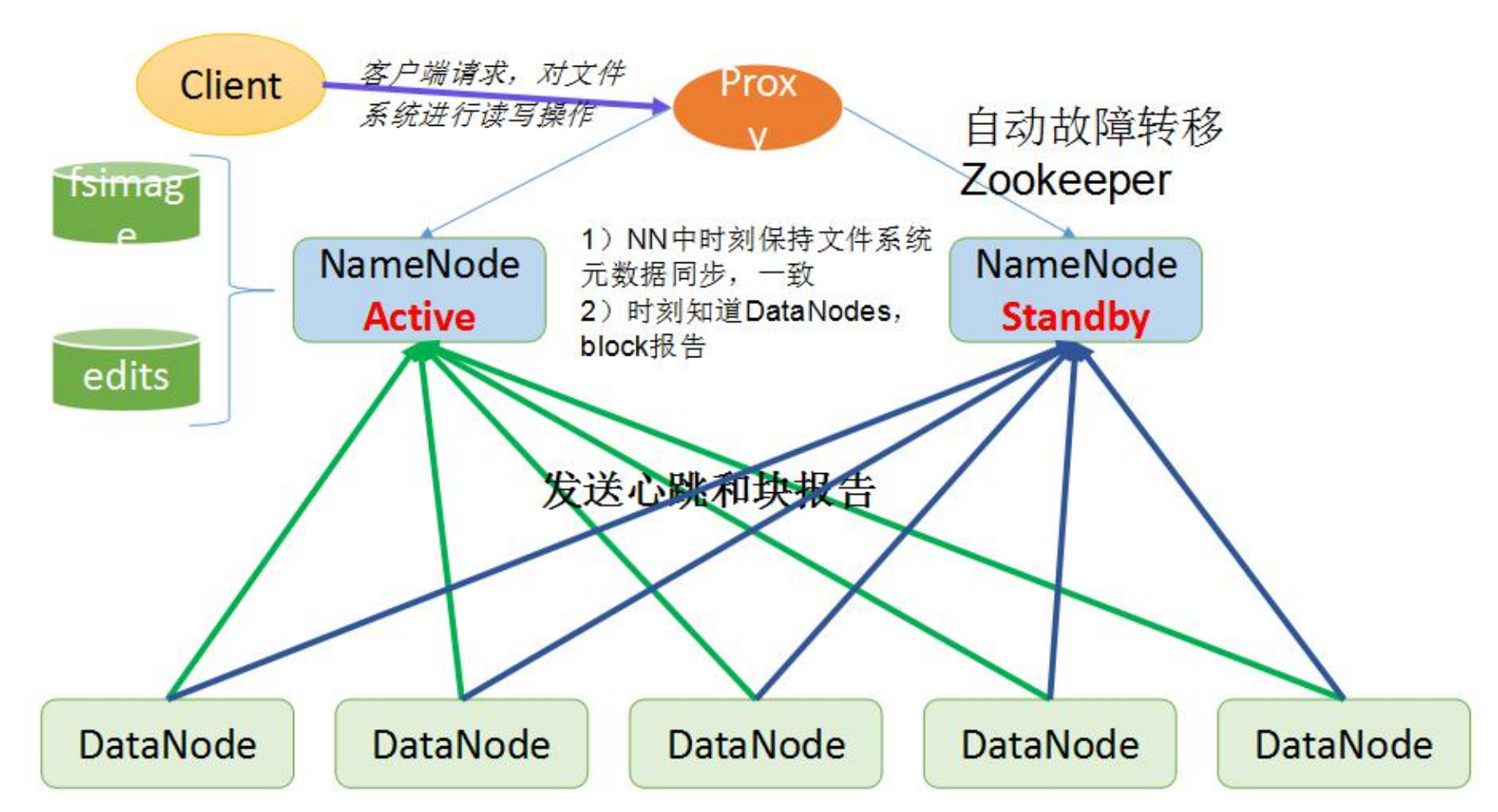
课程大纲

- 分布式集群部署
- Zookeeper部署
- HA集群搭建

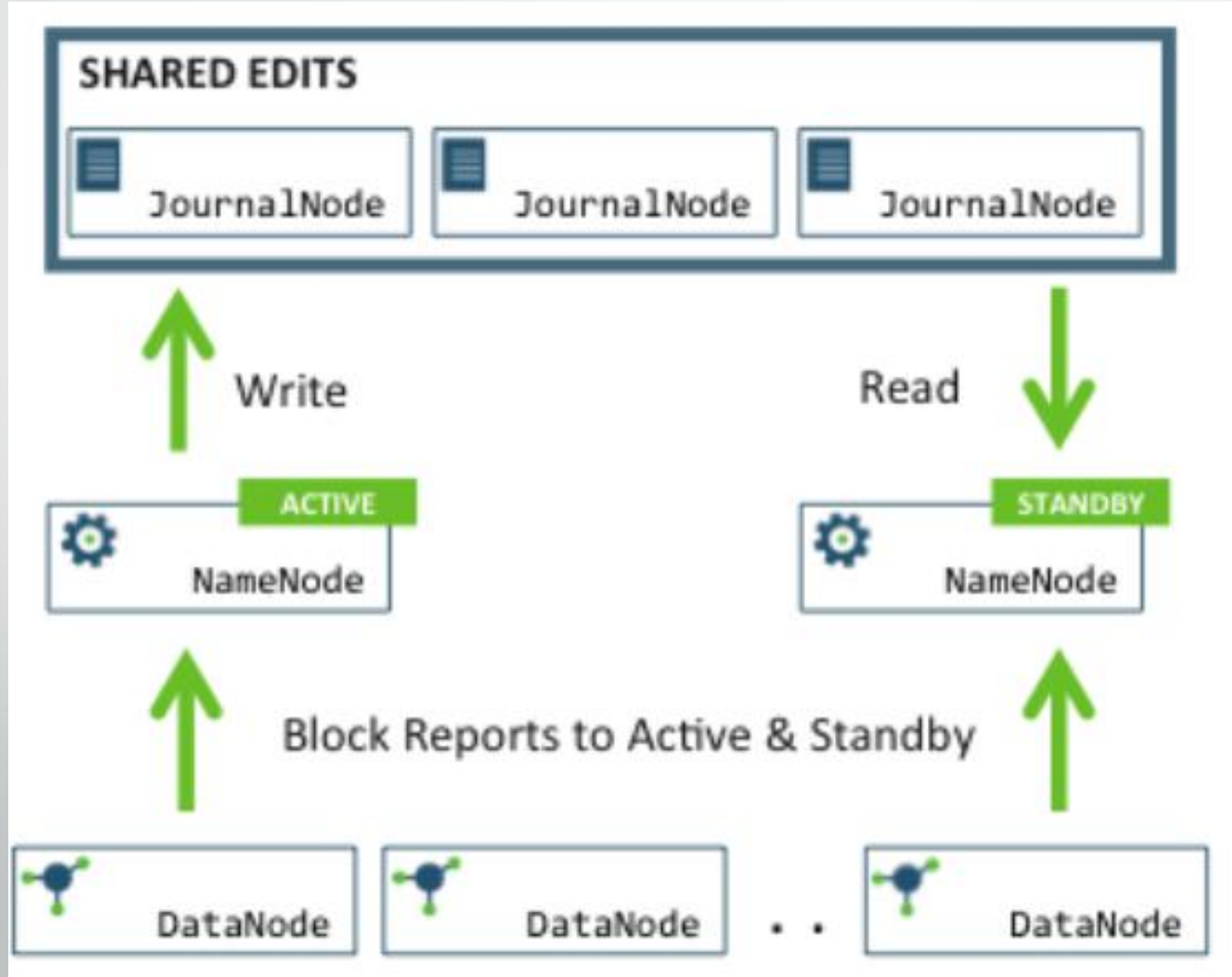
背景

- Hadoop 2.0 之前，在HDFS 集群中 NameNode 存在单点故障（SPOF）。对于只有一个NameNode 的集群，若NameNode 机器出现故障，则整个集群将无法使用，直到NameNode 重新启动。
- NameNode 主要在以下两个方面影响HDFS 集群
  - NameNode 机器发生意外，如宕机，集群将无法使用，直到管理员重启
  - NameNode 机器需要升级，包括软件、硬件升级，此时集群也将无法使用
- HDFS HA 功能通过**配置Active/Standby 两个NameNodes 实现在集群中对NameNode 的热备**来解决上述问题。如果出现故障，如机器崩溃或机器需要升级维护，这时可通过此种方式将NameNode 很快的切换到另外一台机器。

HDFS HA 设计


Client
客户端请求，对文件系统进行读写操作
Proxy
自动故障转移
Zookeeper
fsimage
edits
NameNode
Active
1）NN中时刻保持文件系统元数据同步，一致
2）时刻知道DataNodes，block报告
NameNode
Standby
发送心跳和块报告
DataNode
DataNode
DataNode
DataNode
DataNode

HDFS HA 设计


SHARED EDITS
JournalNode
JournalNode
JournalNode
Write
Read
ACTIVE
NameNode
STANDBY
NameNode
Block Reports to Active & Standby
DataNode
DataNode
DataNode

QJM HA 配置

- NameNode HA 基本配置（core-site.xml、hdfs-site.xml）
  - Active NameNode与Standby NameNode 地址配置
  - NameNode 与DataNode 本地存储路径配置
  - HDFS Namespace 访问配置
  - 隔离fencing配置（配置两节点之间的互相SSH无密钥登录）
- QJM 配置（hdfs-site.xml）
  - QJM 管理编辑日志
  - 编辑日志存储目录

手动故障转移（无需配置）

QJM HA 启动

- Step1 :在各个JournalNode节点上，输入以下命令启动journalnode服务：
  $ sbin/hadoop-daemon.sh start journalnode
- Step2:在[nn1]上，对其进行格式化，并启动：
  $ bin/hdfs namenode -format
  $ sbin/hadoop-daemon.sh start namenode
- Step3:在[nn2]上，同步nn1的元数据信息：
  $ bin/hdfs namenode -bootstrapStandby
- Step4:启动[nn2]：
  $ sbin/hadoop-daemon.sh start namenode
- Step5:将[nn1]切换为Active
  $ bin/hdfs haadmin -transitionToActive nn1
- Step6:在[nn1]上，启动所有datanode
  $ sbin/hadoop-daemons.sh start datanode

NameNode 管理命令

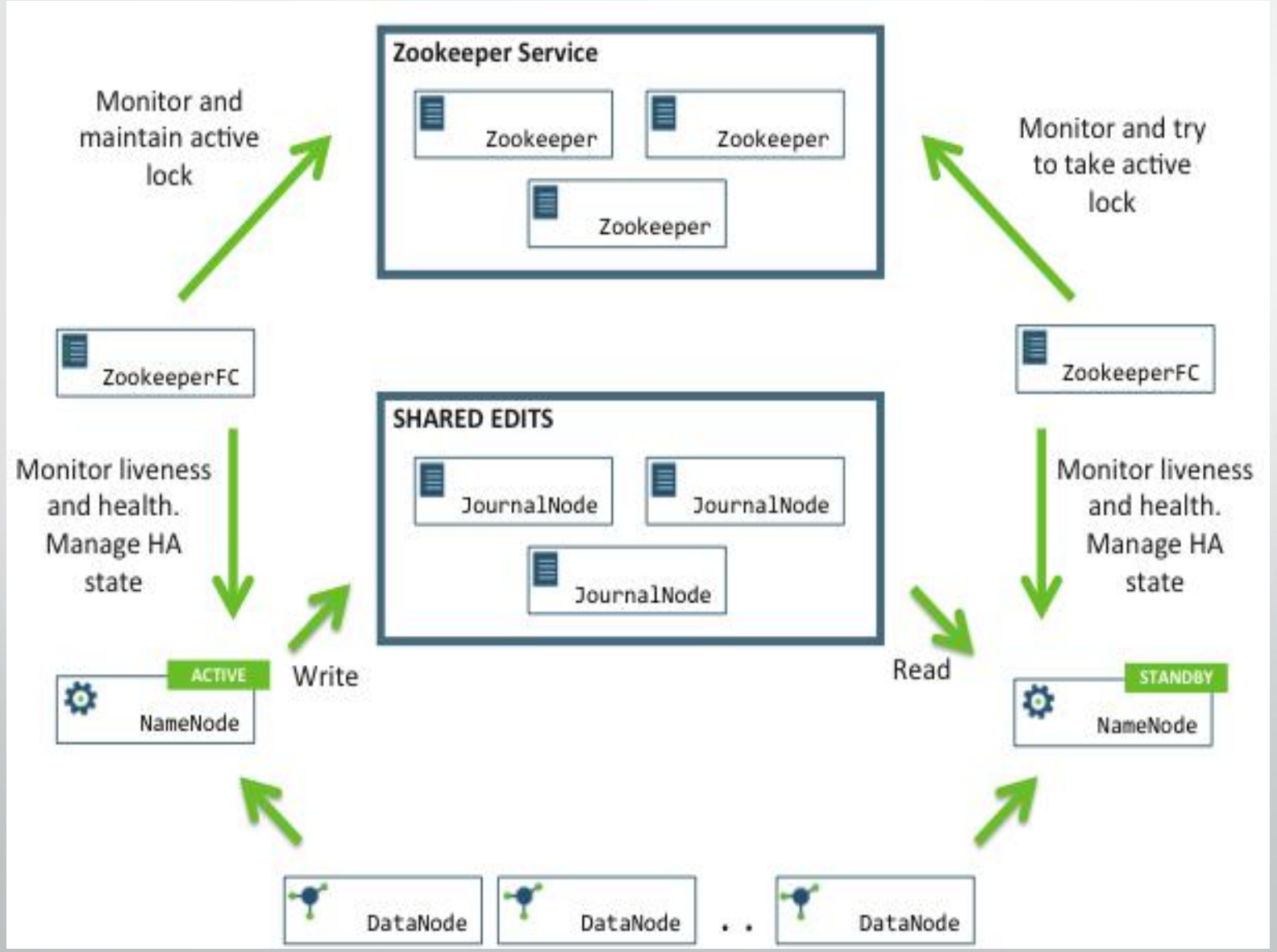
- **bin/hdfs namenode**
  [-backup]
  [-checkpoint]
  [-importCheckpoint]
  [-format [-clusterid cid] [-force] [-nonInteractive]]
  [-upgrade]
  [-rollback]
  [-finalize]
  [-initializeSharedEdits]
  [-bootstrapStandby]
  [-recover [-force]]
- bin/hdfs haadmin

NN HA 自动故障转移

- 启动
  - 关闭所有HDFS 服务 sbin/stop-dfs.sh
  - 启动Zookeeper 集群 bin/zkServer.sh start
  - 初始化 HA 在Zookeeper中状态 bin/hdfs zkfc -formatZK
  - 启动HDFS服务 sbin/start-dfs.sh
  - 在各个NameNode节点上启动DFSZK Failover Controller，先在那台机器启动，那个机器的NameNode就是Active NameNode

    sbin/hadoop-daemon.sh start zkfc
- 验证
  - 将Active NemeNode 进程kill，命令：kill -9 pid
  - 将Active NameNode 机器断开网络，命令：service network stop

NN HA 自动故障转移


Zookeeper Service
Zookeeper
Zookeeper
Zookeeper
Monitor and maintain active lock
Monitor and try to take active lock
ZookeeperFC
ZookeeperFC
SHARED EDITS
JournalNode
JournalNode
JournalNode
Monitor liveness and health. Manage HA state
Monitor liveness and health. Manage HA state
ACTIVE
NameNode
Write
Read
STANDBY
NameNode
DataNode
DataNode
. .
DataNode

HDFS Using QJM