2017.09

Hadoop离线大数据分析

Hadoop分布式文件系统

贝毅君 beiyj@zju.edu.cn

Hadoop Distributed File System

数据量越来越多，在一个操作系统管辖的范围存不下了，那么就分配到更多的操作系统管理的磁盘中，但是不方便管理和维护，因此迫切需要一种系统来管理多台机器上的文件，这就是分布式文件管理系统（DFS）。

通透性。让实际上是通过网络来访问文件的动作，由程序与用户看来，就像是访问本地的磁盘一般。
容错。即使系统中有某些节点脱机，整体来说系统仍然可以持续运作而不会有数据损失。

分布式文件管理系统很多，hdfs只是其中一种。适用于一次写入多次查询的情况，不支持并发写情况，小文件不合适。对于需要频繁写入的小文件，更适合用Hbase进行存储（详见Hbase部分）

2 HDFS架构

自己设计一个HDFS架构：

思路：

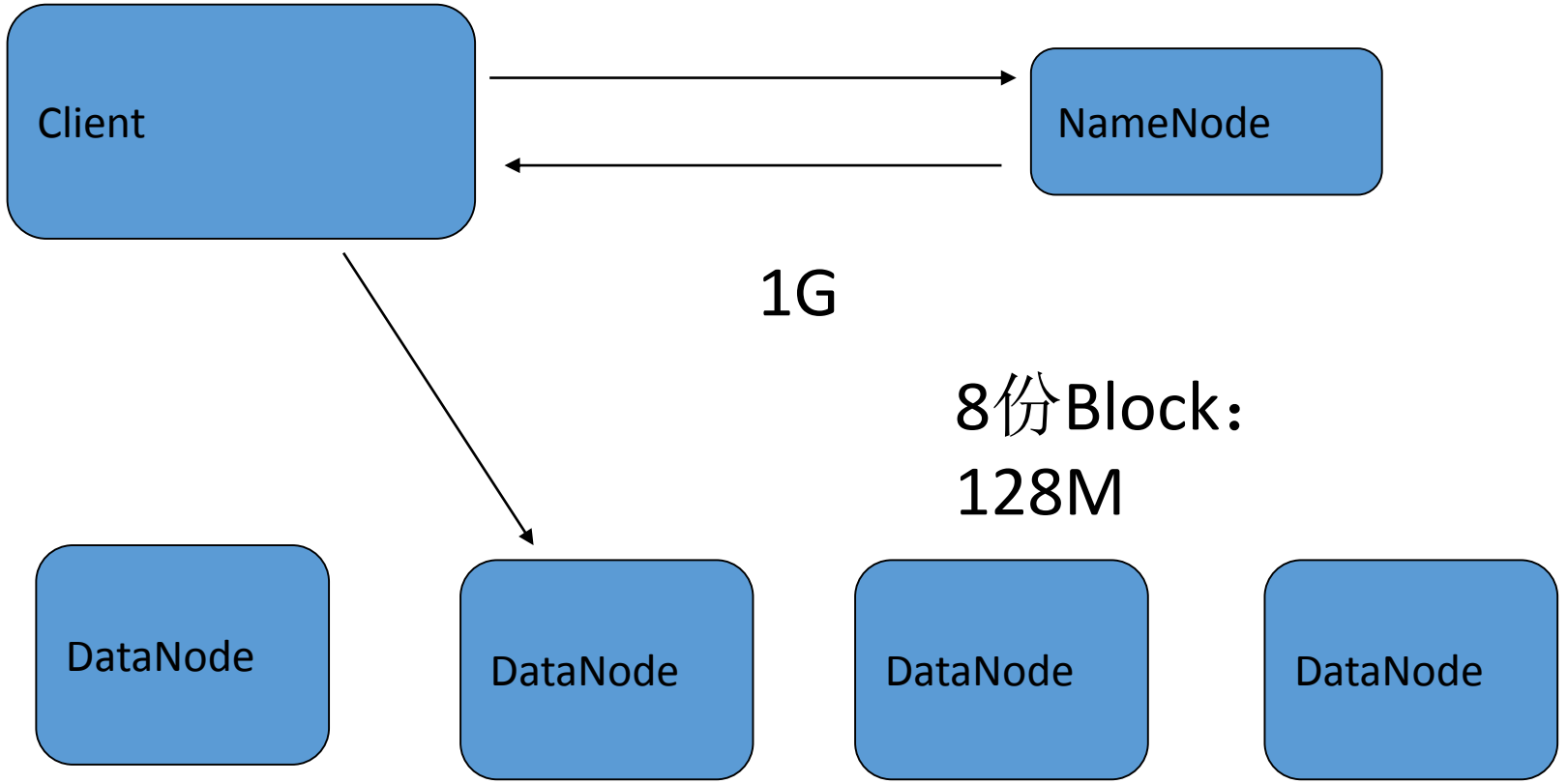
有多个分布式机器用于存储数据

有一个集群管理员的“老大”角色

用于和客户端进行信息交互

管理员角色同时负责用于关于下面

用于实际存储数据的“小弟”



主从结构

主节点，只有一个: *namenode*

（2.0之后的版本可以存在多个）

从节点，有很多个: *datanode*

namenode负责：

接收用户操作请求

维护文件系统的目录结构

管理文件与block之间关系，block与datanode之间关系

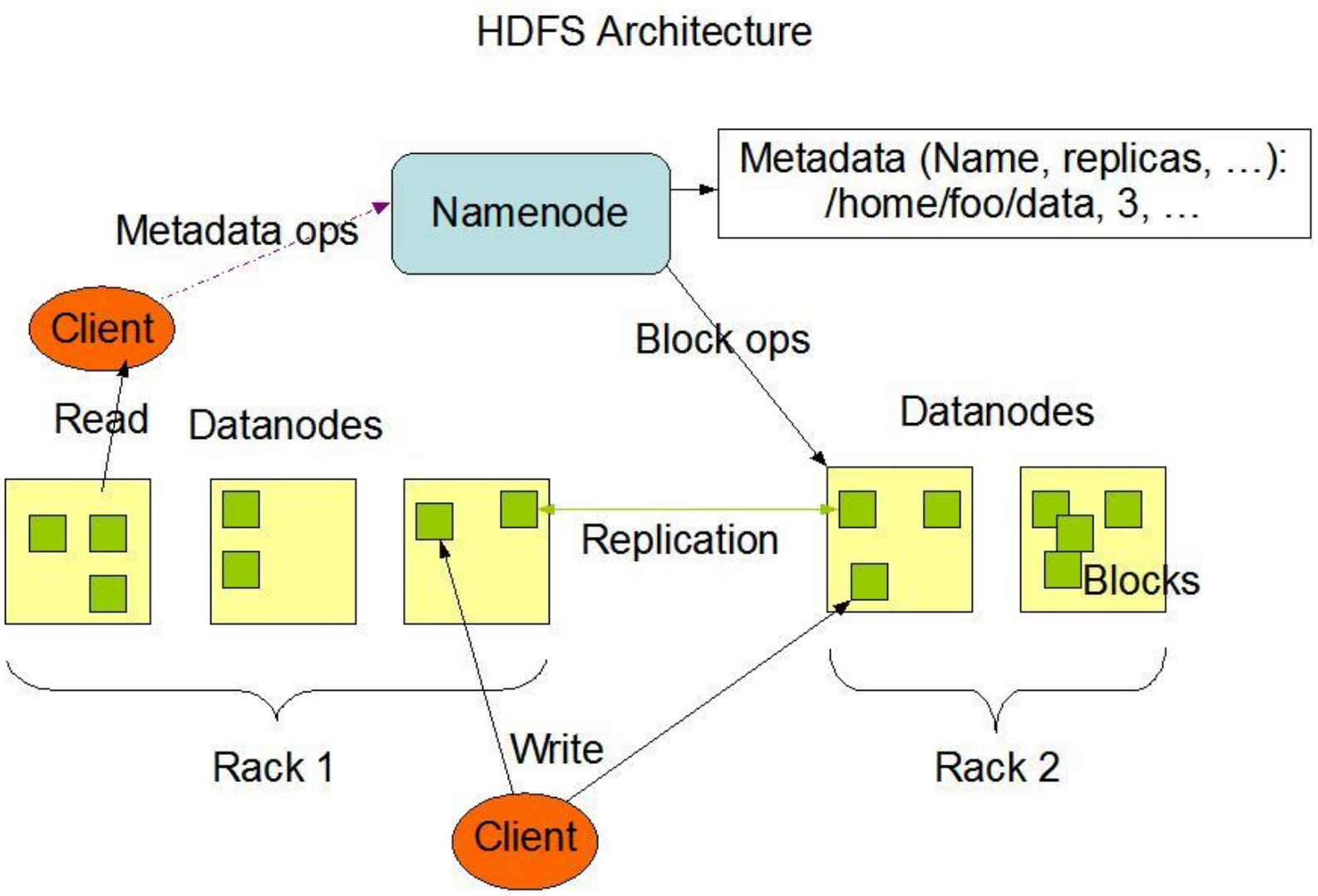
datanode负责：

存储文件

文件被分成block存储在磁盘上

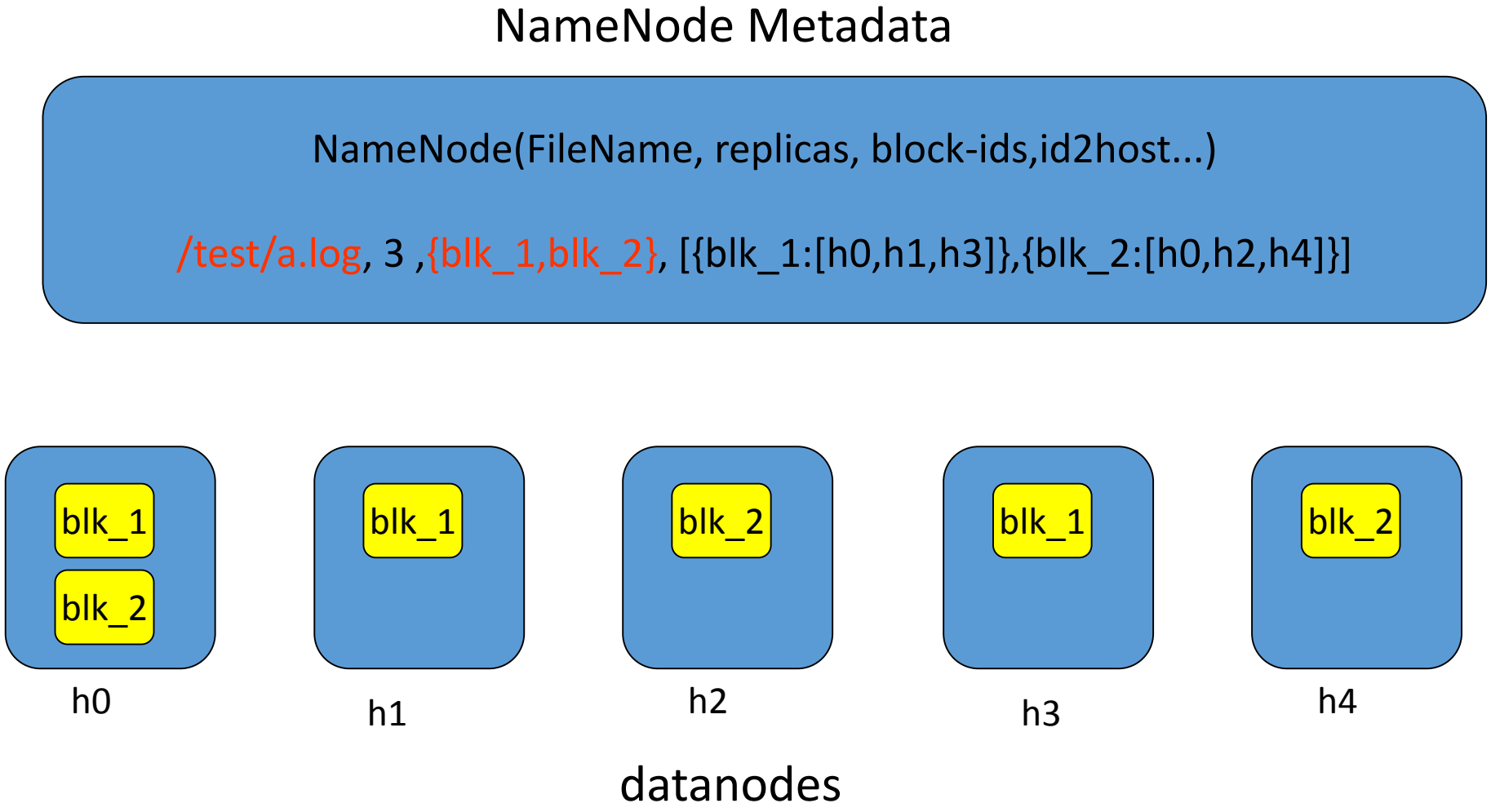
为保证数据安全，文件会有多个副本

2 HDFS架构



- Namenode：用于保管元数据（metadata），并响应client发来的读写请求，将相应的元数据信息返回给客户端
- DataNode：实际保存数据，一个datanode包括数个block，每个block保存一份数据，在Hadoop1.0版本中每个block默认64M，2.0中修改为128M
- Rack：每个机架上包含数个datanode，每一份数据都要在datanode中保存多次，在选择冗余存储的datanode结点时，尽量选择不在一个机架上的不同datanode
- Replication：HDFS默认将每份数据切分成3个block，并将每个block在不同的datanode上
- Client：客户端向namenode发起读写请求，和datanode交互实际读写数据

2 Namenode元数据



- 为什么需要元数据：namenode在HDFS中充当老大的角色，作为领导，不需要知道每个小弟具体的工作内容细节，但需要知道每个人的分工，当分配任务给小弟，当有小弟生病（宕机）时，及时进行容错处理
- 在元数据中保存了以下信息：文件名，该文件被切分成了哪几个block，每个block被保存在哪些台机器上

2 Namenode

是整个文件系统的管理节点。它维护着整个文件系统的文件目录树，文件/目录的元信息和每个文件对应的数据块列表。接收用户的操作请求。

文件包括： hdfs-site.xml的dfs.name.dir属性

①fsimage:元数据镜像文件。存储某一时段NameNode内存元数据信息。

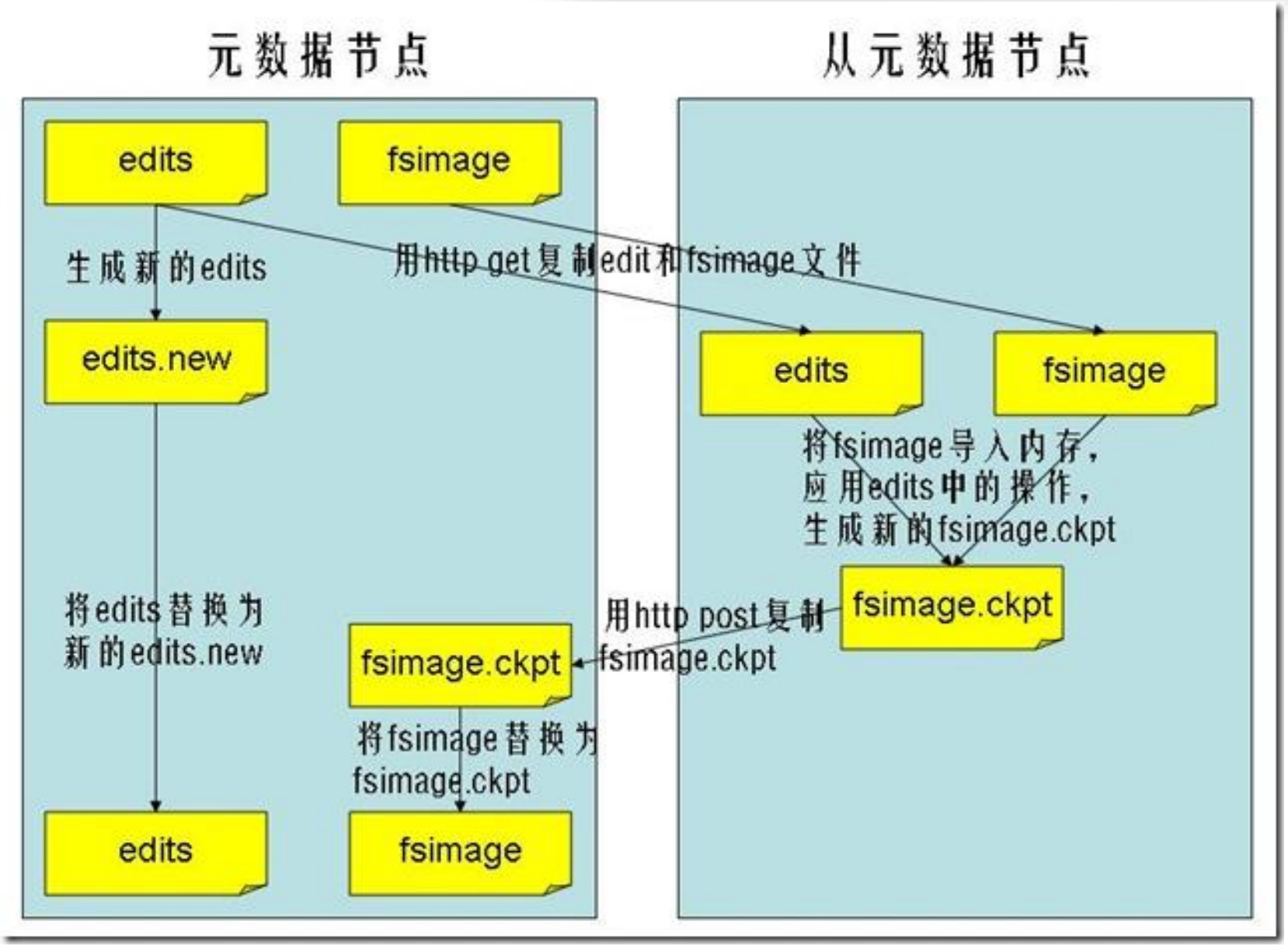
②edits:操作日志文件。

③fstime:保存最近一次checkpoint的时间

以上这些文件是保存在linux的文件系统中。

- Namenode始终在内存中保存metedata，用于处理“读请求”
- 到有“写请求”到来时，namenode会首先写editlog到磁盘，即向edits文件中写日志，成功返回后，才会修改内存，并且向客户端返回
- Hadoop会维护一个fsimage文件，也就是namenode中metedata的镜像，但是fsimage不会随时与namenode内存中的metedata保持一致，而是每隔一段时间通过合并edits文件来更新内容。
- Secondary namenode就是用来合并fsimage和edits文件来更新NameNode的metedata的。

2 SecondaryNameNode



- HA的一个解决方案。但不支持热备配置即可。
- 工作流程：
- secondary通知namenode切换edits文件
- secondary从namenode获得fsimage和edits(通过http)
- secondary将fsimage载入内存，然后开始合并edits
- secondary将新的fsimage发回给namenode
- namenode用新的fsimage替换旧的fsimage
- 什么时候checkpoint：
- fs.checkpoint.period 指定两次checkpoint的最大时间间隔，默认3600秒。
- fs.checkpoint.size　规定edits文件的最大值，一旦超过这个值则强制checkpoint，不管是否到达最大时间间隔。默认大小是64M。

2 HDFS的shell

- -help [cmd] //显示命令的帮助信息
- -ls(r) <path> //显示当前目录下所有文件
- -du(s) <path> //显示目录中所有文件大小
- -count[-q] <path> //显示目录中文件数量
- -mv <src> <dst> //移动多个文件到目标目录
- -cp <src> <dst> //复制多个文件到目标目录
- -rm(r) //删除文件(夹)
- -put <localsrc> <dst> //本地文件复制到hdfs
- -copyFromLocal //同put

2 HDFS的shell

- -moveFromLocal　　//从本地文件移动到hdfs
- -get [-ignoreCrc] <src> <localdst>　//复制文件到本地，可以忽略crc校验
- -getmerge <src> <localdst>　　//将源目录中的所有文件排序合并到一个文件中
- -cat <src>　　//在终端显示文件内容
- -text <src>　　//在终端显示文件内容
- -copyToLocal [-ignoreCrc] <src> <localdst>//复制到本地
- -moveToLocal <src> <localdst>
- -mkdir <path>　　//创建文件夹
- -touchz <path>　　//创建一个空文件

使用fileSystem读文件内容

```
public class FileSystemCat {

  public static void main(String[] args) throws Exception {
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), conf);
    InputStream in = null;
    try {
      in = fs.open(new Path(uri));
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
```

```
% hadoop FileSystemCat hdfs://localhost/user/tom/quangle.txt
On the top of the Crumpetty Tree
The Quangle Wangle sat,
But his face you could not see,
On account of his Beaver Hat.
```

使用fileSystem写文件

```
public class FileCopyWithProgress {
  public static void main(String[] args) throws Exception {
    String localSrc = args[0];
    String dst = args[1];

    InputStream in = new BufferedInputStream(new FileInputStream(localSrc));

    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(dst), conf);
    OutputStream out = fs.create(new Path(dst), new Progressable() {
      public void progress() {
        System.out.print(".");
      }
    });

    IOUtils.copyBytes(in, out, 4096, true);
  }
}
```

2 RPC介绍

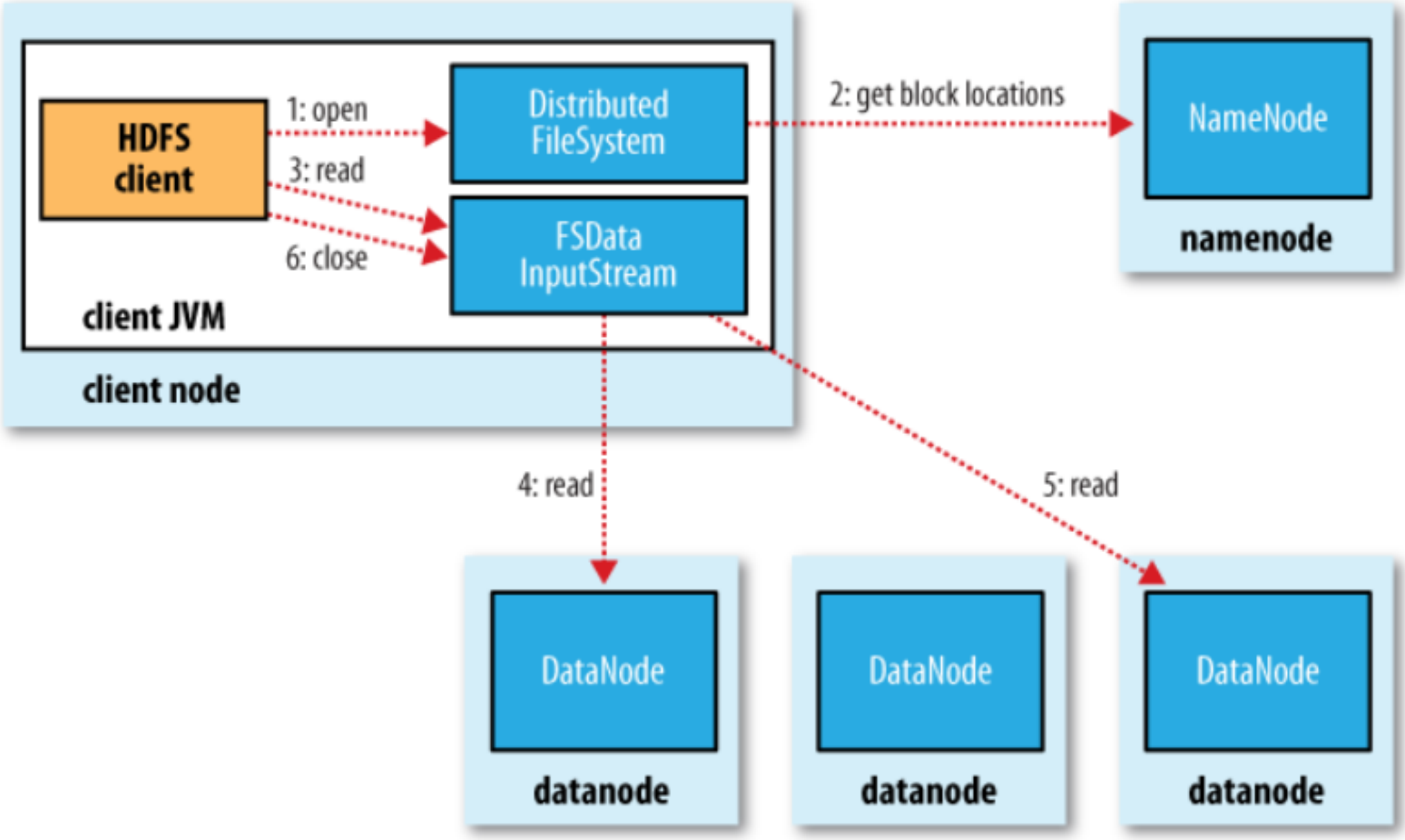
Remote Procedure Call

RPC——远程过程调用协议，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC协议假定某些传输协议的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI网络通信模型中，RPC跨越了传输层和应用层。RPC使得开发包括网络分布式多程序在内的应用程序更加容易

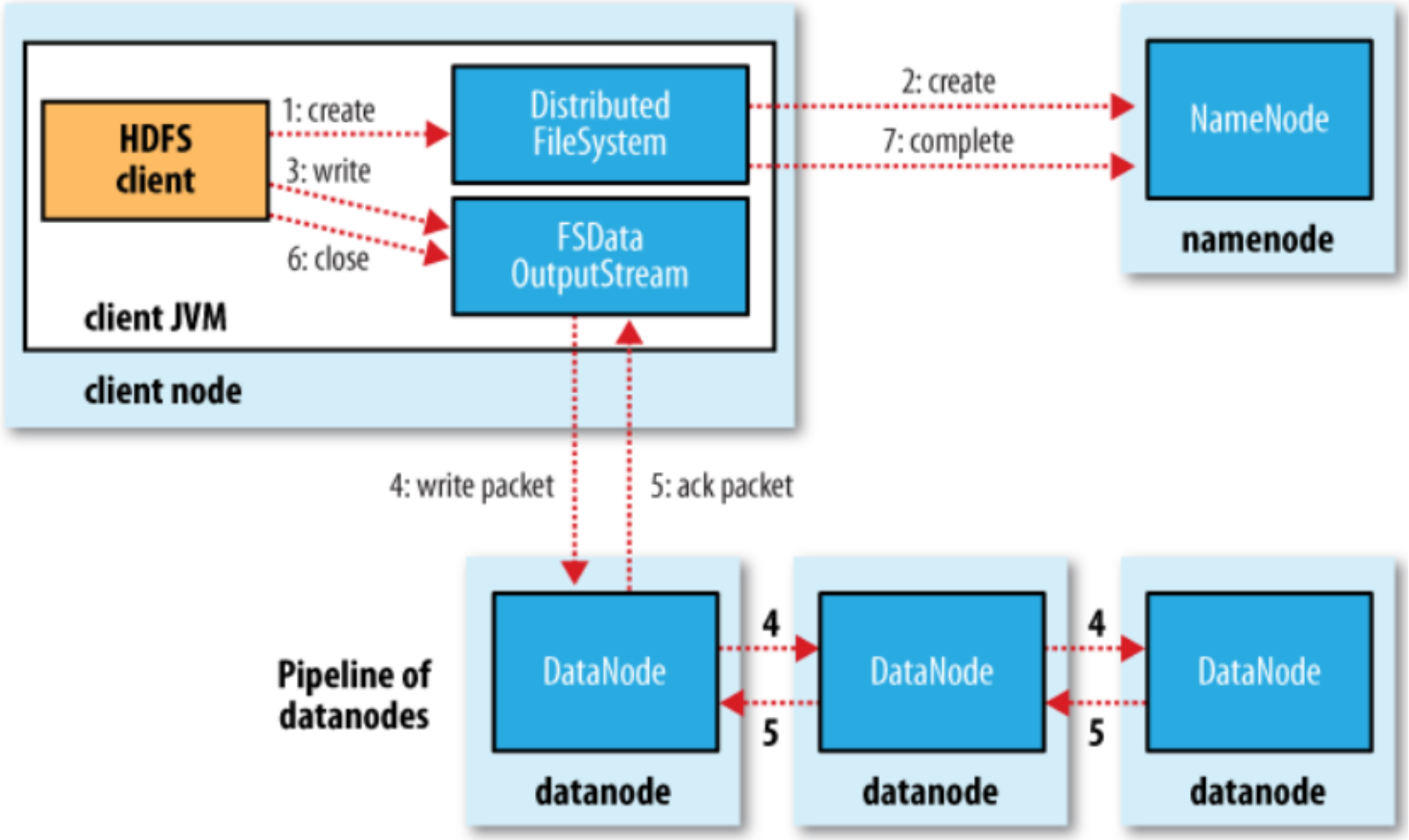
RPC采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。首先，客户机调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息的到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行。

hadoop的整个体系结构就是构建在RPC之上的(见org.apache.hadoop.ipc)。

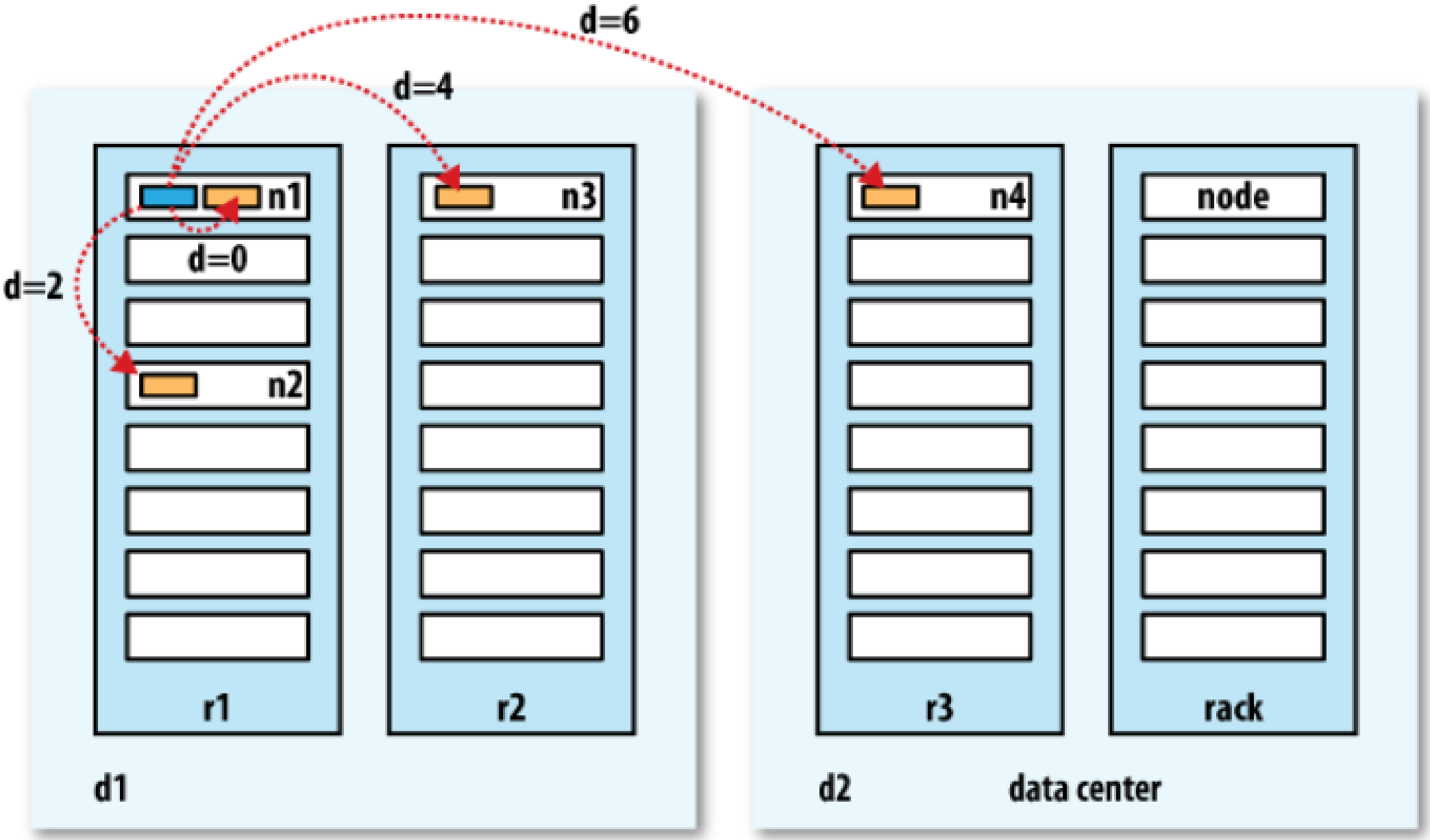
2 HDFS读过程


HDFS client
1: open
3: read
6: close
Distributed FileSystem
FSData InputStream
client JVM
client node
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

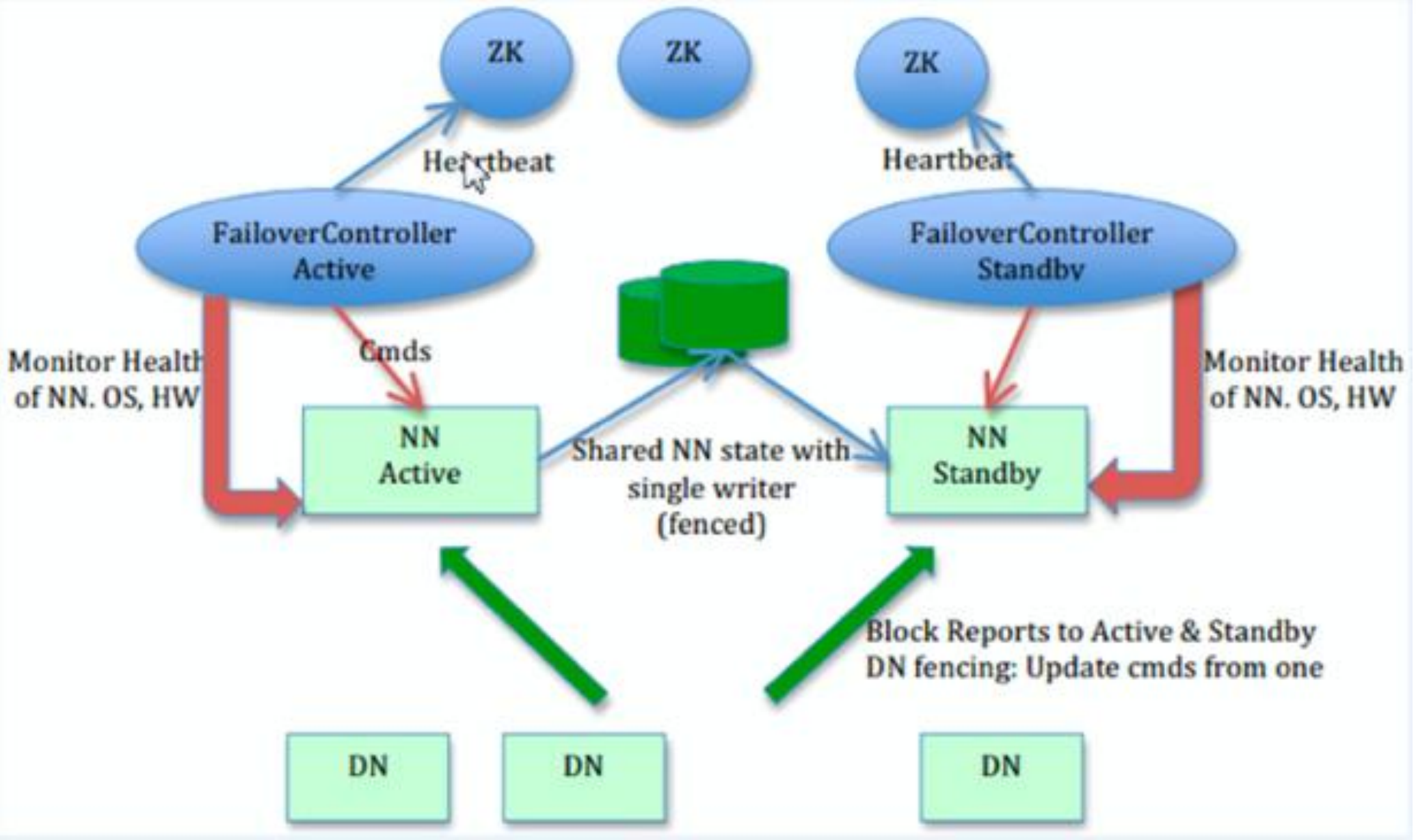
2 HDFS写过程


HDFS client
1: create
3: write
6: close
Distributed FileSystem
FSData OutputStream
client JVM
client node
2: create
7: complete
NameNode
namenode
4: write packet
5: ack packet
Pipeline of datanodes
DataNode
4
5
DataNode
4
5
DataNode
datanode
datanode
datanode

2 Hadoop网络距离



HDFS HA


ZK
ZK
ZK
Heartbeat
Heartbeat
FailoverController
Active
FailoverController
Standby
Monitor Health
of NN. OS, HW
Cmds
NN
Active
Shared NN state with
single writer
(fenced)
NN
Standby
Monitor Health
of NN. OS, HW
Block Reports to Active & Standby
DN fencing: Update cmds from one
DN
DN
DN

2 HDFS HA

在集群模式中，Hadoop用nameservice的概念代替namenode的概念，一个nameservice中包括两个namenode其中一个为active状态，第二个为standby状态。

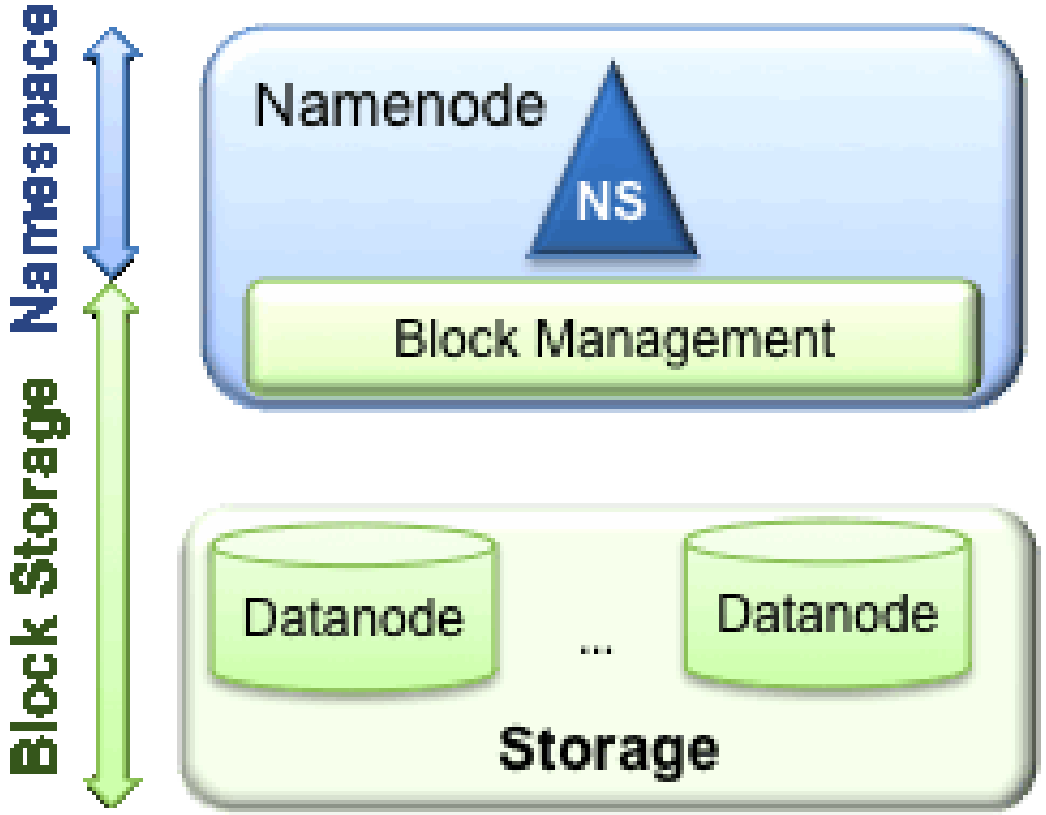
nameservice其实是一个对外访问的逻辑名，这样用户程序访问时，不用知道具体哪个namenode处于active

failover控制器用来监控namenode的状态，并通过心跳机制向zk集群汇报情况，如果当前active的namenode宕机，failover监控器会立即向zk集群汇报，集群会通知当前处于standby状态的namenode变为active状态，开始接手下来的工作

单机环境下的NN通过edits文件和fsimage来协调存储元信息，在集群模式中，edits文件单独存储在一个journalNode分布式日志系统中，

用来存储edits文件的journalNode节点同样由zk负责协调，当处于active节点的namenode在向journalNode写文件的时候，会同时向所有journalNode写数据，因为journalNode节点比较多，所以只要多数节点写入成功，就算成功

2 HDFS联邦



问题：如何进行NameSpace的水平扩展，从而使用多个NN

HDFS主要包括两层：

Namespace

由目录，文件和块组成；支持所有文件系统操作包括增加、删除、修改和列出文件和目录

Block Storage Service 有两个部分：

Block管理（被NameNode包含）

提供datanode集群的注册和定期的心跳检查

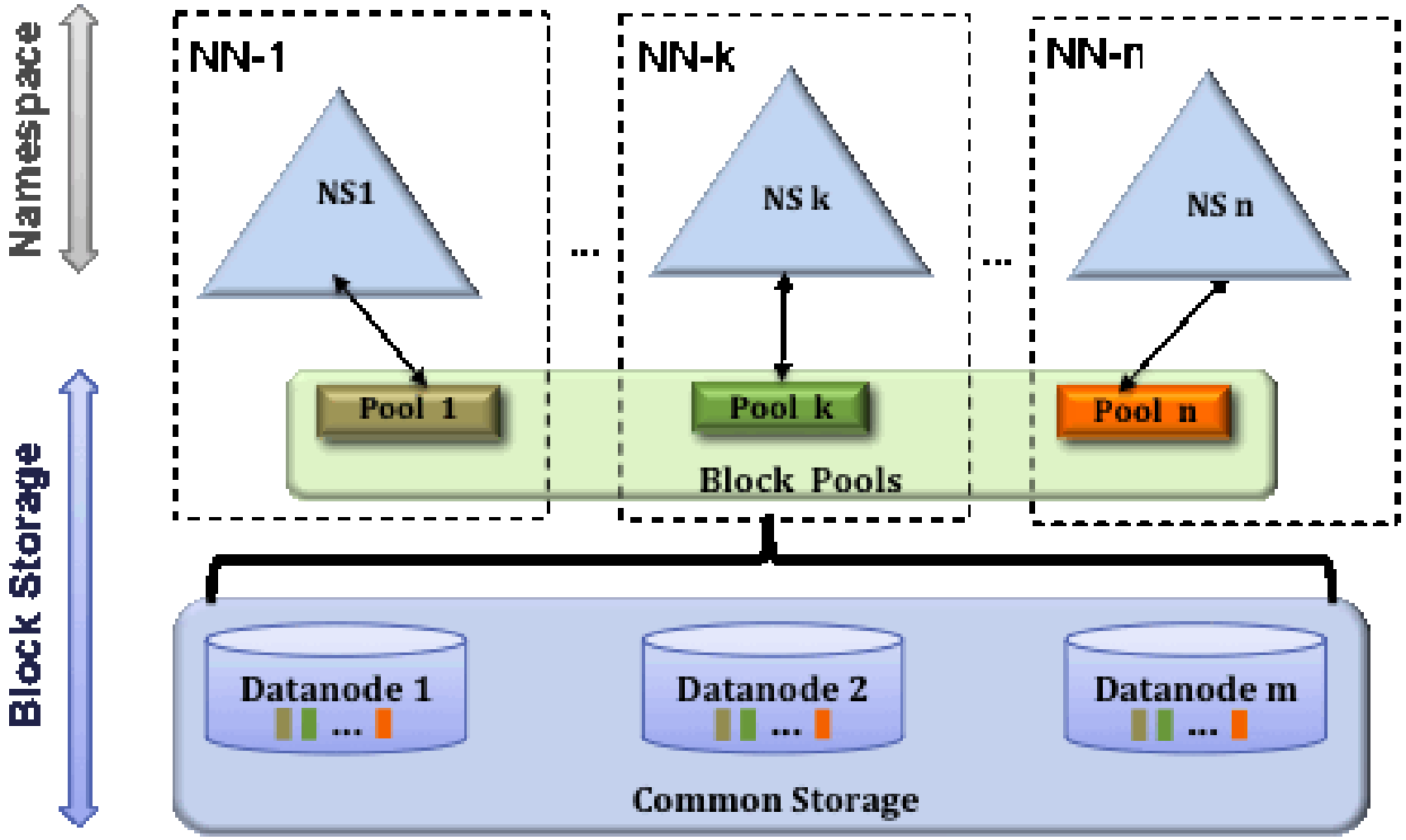
处理block的报告并掌握block的位置

支持block的相关操作，如增删改查和得到block的位置

管理副本位置，管理副本的复制和删除

存储-由提供datanodes的本地系统的提供，允许读写。

2 HDFS联邦



为了水平扩展，联盟使用多个独立的Namenodes/namespaces。这些NameNode之间是相连的，即Namenodes是独立的,不需要互相协调。DataNode被所有的NameNode使用用来作为通用的数据块存储设备。每一个DataNode注册集群中所有的NameNode。Datanodes发送心跳和block报告并且处理NameNode发送的命令

2017.09

THE END