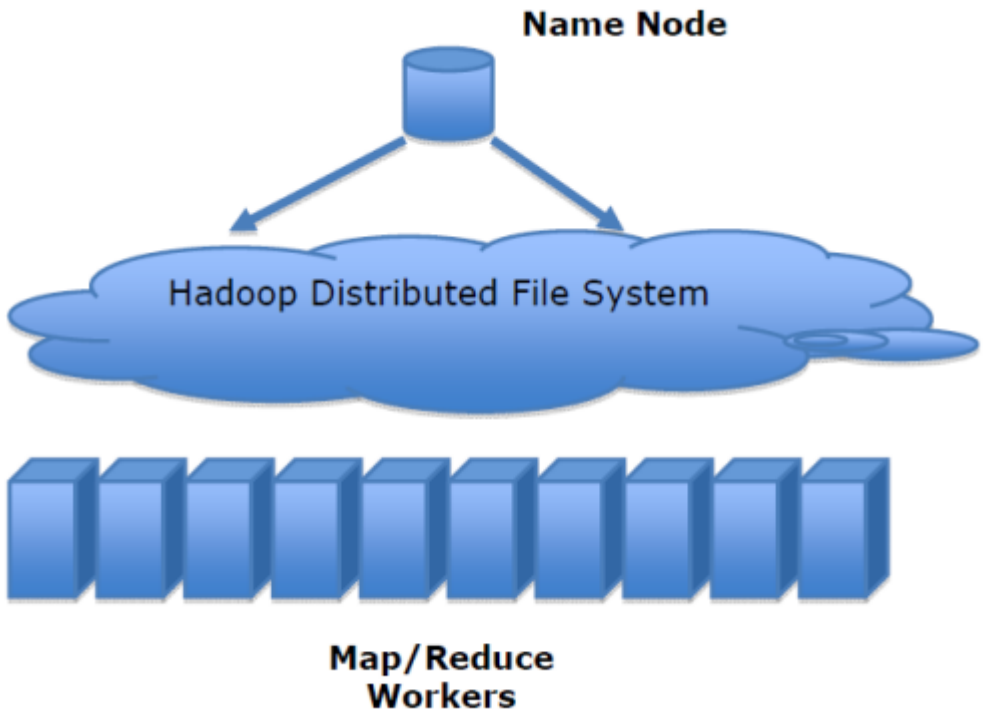
Hadoop 核心-HDFS

1. HDFS概述

1.1 介绍

在现代的企业环境中，单机容量往往无法存储大量数据，需要跨机器存储。统一管理分布在集群上的文件系统称为**分布式文件系统** 。

HDFS （Hadoop Distributed File System）是 Apache Hadoop 项目的一个子项目. Hadoop 非常适于存储大型数据 (比如 TB 和 PB), 其就是使用 HDFS 作为存储系统. HDFS 使用多台计算机存储文件, 并且提供统一的访问接口, 像是访问一个普通文件系统一样使用分布式文件系统.



1.2 历史

1. **Doug Cutting** 在做 Lucene 的时候, 需要编写一个爬虫服务, 这个爬虫写的并不顺利, 遇到了一些问题, 诸如: 如何存储大规模的数据, 如何保证集群的可伸缩性, 如何动态容错等
2. 2013年的时候, Google 发布了三篇论文, 被称作为三驾马车, 其中有一篇叫做 GFS, 是描述了 Google 内部的一个叫做 **GFS** 的分布式大规模文件系统, 具有强大的可伸缩性和容错性
3. Doug Cutting 后来根据 GFS 的论文, 创造了一个新的文件系统, 叫做 HDFS

2. HDFS应用场景

2.1 适合的应用场景

- 存储非常大的文件：这里非常大指的是几百M、G、或者TB级别，需要**高吞吐量**，对**延时没有要求**。
- 采用流式的数据访问方式: 即**一次写入、多次读取**，数据集经常从数据源生成或者拷贝一次，然后在其上做很多分析工作。
- 运行于商业硬件上: Hadoop不需要特别贵的机器，可运行于普通廉价机器，可以处**节约成本**
- 需要高**容错性**
- 为数据存储提供所需的**扩展能力**

2.2 不适合的应用场景

1） 低延时的数据访问 对延时要求在毫秒级别的应用，不适合采用HDFS。HDFS是为高吞吐数据传输设计的,因此可能**牺牲延时**

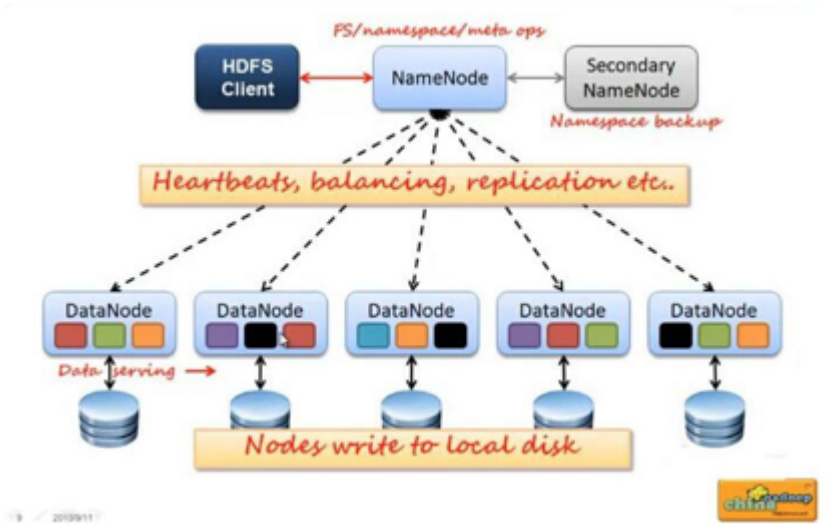
2）大量小文件 文件的元数据保存在**NameNode的内存中**， 整个文件系统的文件数量会受限于NameNode的内存大小。 经验而言，一个文件/目录/文件块一般占有150字节的元数据内存空间。如果有100万个文件，每个文件占用1个文件块，则需要大约300M的内存。因此十亿级别的文件数量在现有商用机器上难以支持。

3）多方读写，需要任意的文件修改 HDFS采用追加（append-only）的方式写入数据。**不支持文件任意offset的修改**。不支持多个写入器（writer）

3. HDFS 的架构

HDFS是一个`主/从（Mater/Slave）体系结构`，

HDFS由四部分组成，**HDFS Client**、**NameNod e**、**DataNode**和**Secondary NameNode**。



**1、Client：就是客户端。

- 文件切分。文件上传 HDFS 的时候，Client 将文件切分成 一个一个的Block，然后进行存储。
- 与 NameNode 交互，获取文件的位置信息。
- 与 DataNode 交互，读取或者写入数据。
- Client 提供一些命令来管理 和访问HDFS，比如启动或者关闭HDFS。

2、NameNode：就是 master，它是一个主管、管理者。

- 管理 HDFS 的名称空间
- 管理数据块（Block）映射信息
- 配置副本策略
- 处理客户端读写请求。

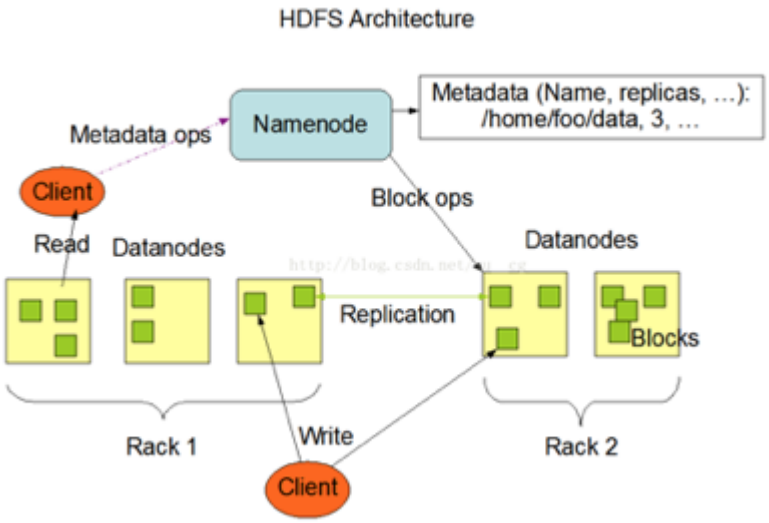
3、DataNode：就是Slave。NameNode 下达命令，DataNode 执行实际的操作。

- 存储实际的数据块。
- 执行数据块的读/写操作。

4、Secondary NameNode：并非 NameNode 的热备。当NameNode 挂掉的时候，它并不能马上替换 NameNode 并提供服务。

- 辅助 NameNode，分担其工作量。
- 定期合并 fsimage和fsedits，并推送给NameNode。
- 在紧急情况下，可辅助恢复 NameNode。

4:NameNode和DataNode




```
### 4.1 NameNode作用
```

- NameNode在内存中保存着整个文件系统的**名称 空间**和文件数据块的**地址映射**
- 整个HDFS可存储的文件数受限于**NameNode的内存大小**

1、NameNode元数据信息 文件名，文件目录结构，文件属性(生成时间，副本数，权限)每个文件的块列表。 以及列表中的块与块所在的DataNode之间的地址映射关系 在内存中加载文件系统中每个文件和每个数据块的引用关系(文件、block、datanode之间的映射信息) 数据会定期保存到本地磁盘（fsImage文件和edits文件）

2、NameNode文件操作 NameNode负责文件元数据的操作 DataNode负责处理文件内容的读写请求，数据流不经过NameNode，会询问它跟那个DataNode联系

3、NameNode副本 文件数据块到底存放到哪些DataNode上，是由NameNode决定的，NN根据全局情况做出放置副本的决定

4、NameNode心跳机制 全权管理数据块的复制，周期性的接受心跳和块的状态报告信息（包含该DataNode上所有数据块的列表） 若接受到心跳信息，NameNode认为DataNode工作正常，如果在10分钟后还接受到不到DN的心跳，那么NameNode认为DataNode已经宕机 ,这时候NN准备要把DN上的数据块进行重新的复制。 块的状态报告包含了一个DN上所有数据块的列表，blocks report 每个1小时发送一次.

4.2 DataNode作用

提供真实文件数据的存储服务。

1、Data Node以数据块的形式存储HDFS文件

2、Data Node 响应HDFS 客户端读写请求

3、Data Node 周期性向NameNode汇报心跳信息

4、 Data Node 周期性向NameNode汇报数据块信息

5、 Data Node 周期性向NameNode汇报缓存数据块信息

5:HDFS的副本机制和机架感知

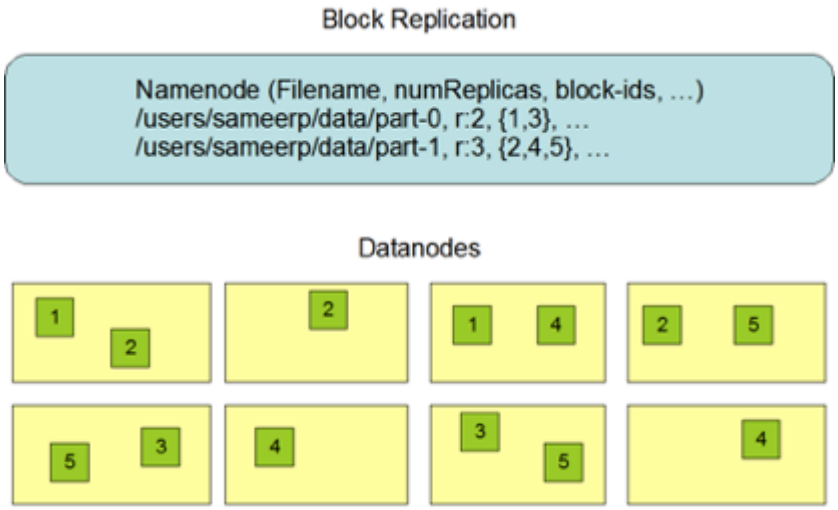
5.1 HDFS 文件副本机制

所有的文件都是以 block 块的方式存放在 HDFS 文件系统当中,作用如下

1. 一个文件有可能大于集群中任意一个磁盘，引入块机制,可以很好的解决这个问题
2. 使用块作为文件存储的逻辑单位可以简化存储子系统
3. 块非常适合用于数据备份进而提供数据容错能力

在 Hadoop1 当中, 文件的 block 块默认大小是 64M, hadoop2 当中, 文件的 block 块大小默认是 128M, block 块的大小可以通过 hdfs-site.xml 当中的配置文件进行指定

```
<property>
    <name>dfs.block.size</name>
    <value>块大小 以字节为单位</value>
</property>
```



5.2 机架感知

HDFS分布式文件系统的内部有一个副本存放策略：以默认的副本数=3为例：

1、第一个副本块存本机

2、第二个副本块存跟本机同机架内的其他服务器节点

3、第三个副本块存不同机架的一个服务器节点上

6、hdfs的命令行使用

`ls`

```
格式:   hdfs dfs -ls  URI
作用:类似于Linux的ls命令，显示文件列表
```

```
 hdfs  dfs   -ls  /
```

`lsr`

```
格式  :   hdfs  dfs -lsr URI
作用  :在整个目录下递归执行ls，与UNIX中的ls-R类似
```

```
 hdfs  dfs   -lsr  /
```

`mkdir`

```
格式 :  hdfs  dfs [-p] -mkdir <paths>
作用 :   以<paths>中的URI作为参数，创建目录。使用-p参数可以递归创建目录
```

`put`

```
格式   :  hdfs dfs -put <localsrc >  ... <dst>
作用 :   将单个的源文件src或者多个源文件srcs从本地文件系统拷贝到目标文件系统中（<dst>对应的路径）。也可以从标准输入中读取输入，写入目标文件系统中
```

```
hdfs dfs -put  /rooot/a.txt  /dir1
```

`moveFromLocal`

```
格式:  hdfs  dfs -moveFromLocal  <localsrc>   <dst>
作用:   和put命令类似，但是源文件localsrc拷贝之后自身被删除
```

```
hdfs  dfs -moveFromLocal  /root/install.log  /
```

moveToLocal

```
未实现
```

get

```
格式   hdfs dfs  -get [-ignorecrc ]  [-crc]  <src> <localdst>

作用：将文件拷贝到本地文件系统。 CRC 校验失败的文件通过-ignorecrc选项拷贝。 文件和CRC校验和可以通过-CRC选项拷贝
```

```
hdfs dfs  -get   /install.log  /export/servers
```

mv

```
格式  :  hdfs  dfs -mv URI   <dest>
作用:  将hdfs上的文件从原路径移动到目标路径（移动之后文件删除），该命令不能夸文件系统
```

```
hdfs  dfs  -mv  /dir1/a.txt   /dir2
```

rm

```
格式:  hdfs dfs -rm [-r] 【-skipTrash】 URI 【URI 。。。】
作用:    删除参数指定的文件，参数可以有多个。    此命令只删除文件和非空目录。
如果指定-skipTrash选项，那么在回收站可用的情况下，该选项将跳过回收站而直接删除文件；
否则，在回收站可用时，在HDFS Shell 中执行此命令，会将文件暂时放到回收站中。
```

```
hdfs  dfs  -rm  -r  /dir1
```

cp

```
格式:     hdfs  dfs  -cp URI [URI ...] <dest>
作用:     将文件拷贝到目标路径中。如果<dest>  为目录的话，可以将多个文件拷贝到该目录下。
-f
选项将覆盖目标，如果它已经存在。
-p
选项将保留文件属性（时间戳、所有权、许可、ACL、XAttr）。
```

```
hdfs dfs -cp /dir1/a.txt  /dir2/b.txt
```

cat

```
hdfs dfs  -cat  URI [uri  ...]
作用：将参数所指示的文件内容输出到stdout
```

```
hdfs dfs  -cat /install.log
```

chmod

```
格式：      hdfs   dfs  -chmod  [-R]  URI[URI  ...]
作用：      改变文件权限。如果使用  -R 选项，则对整个目录有效递归执行。使用这一命令的用户必须是文件的所属用户，或者超级用户。
```

```
hdfs dfs -chmod -R 777 /install.log
```

chown

```
格式：      hdfs   dfs  -chmod  [-R]  URI[URI  ...]
作用：      改变文件的所属用户和用户组。如果使用  -R 选项，则对整个目录有效递归执行。使用这一命令的用户必须是文件的所属用户，或者超级用户。
```

```
hdfs  dfs  -chown  -R hadoop:hadoop  /install.log
```

appendToFile

```
格式：hdfs dfs -appendToFile <localsrc> ... <dst>
作用：追加一个或者多个文件到hdfs指定文件中.也可以从命令行读取输入.
```

```
 hdfs dfs -appendToFile  a.xml b.xml  /big.xml
```

7、hdfs的高级使用命令

7.1、HDFS文件限额配置

在多人共用HDFS的环境下，配置设置非常重要。特别是在Hadoop处理大量资料的环境，如果没有配额管理，很容易把所有的空间用完造成别人无法存取。Hdfs的配额设定是针对目录而不是针对账号，可以 让每个账号仅操作某一个目录，然后对目录设置配置。

hdfs文件的限额配置允许我们以文件个数，或者文件大小来限制我们在某个目录下上传的文件数量或者文件内容总量，以便达到我们类似百度网盘网盘等限制每个用户允许上传的最大的文件的量。

```
1    hdfs dfs -count -q -h /user/root/dir1  #查看配额信息
```

所谓的空间限额

7.1.1、数量限额

```
1   hdfs dfs  -mkdir -p /user/root/dir    #创建hdfs文件夹
2   hdfs dfsadmin -setQuota 2  dir      # 给该文件夹下面设置最多上传两个文件，发现只能上传一个文件
```

```
1   hdfs dfsadmin -clrQuota /user/root/dir  # 清除文件数量限制
```

7.1.2、空间大小限额

在设置空间配额时，设置的空间至少是block_size * 3大小

```
1   hdfs dfsadmin -setSpaceQuota 4k /user/root/dir   # 限制空间大小4KB
2   hdfs dfs -put  /root/a.txt  /user/root/dir
```

生成任意大小文件的命令:

```
1   dd if=/dev/zero of=1.txt  bs=1M count=2     #生成2M的文件
```

清除空间配额限制

```
1   hdfs dfsadmin -clrSpaceQuota /user/root/dir
```

7.2、hdfs的安全模式

安全模式是hadoop的一种**保护机制**，用于保证集群中的数据块的安全性。当集群启动的时候，会首先进入安全模式。当系统处于安全模式时会检查数据块的完整性。

假设我们设置的副本数（即参数dfs.replication）是3，那么在datanode上就应该有3个副本存在，假设只存在2个副本，那么比例就是2/3=0.666。hdfs默认的副本率0.999。我们的副本率0.666明显小于0.999，因此系统会自动的复制副本到其他dataNode，使得副本率不小于0.999。如果系统中有5个副本，超过我们设定的3个副本，那么系统也会删除多于的2个副本。

在安全模式状态下，文件系统只接受读数据请求，而不接受删除、修改等变更请求。在，当整个系统达到安全标准时，HDFS自动离开安全模式。

安全模式操作命令

```
hdfs  dfsadmin  -safemode  get #查看安全模式状态
hdfs  dfsadmin  -safemode  enter #进入安全模式
hdfs  dfsadmin  -safemode  leave #离开安全模式
```

8. HDFS基准测试

实际生产环境当中，hadoop的环境搭建完成之后，第一件事情就是进行压力测试，测试我们的集群的读取和写入速度，测试我们的网络带宽是否足够等一些基准测试

8.1 测试写入速度

向HDFS文件系统中写入数据,10个文件,每个文件10MB,文件存放到/benchmarks/TestDFSIO中

```
hadoop jar /export/servers/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar  TestDFSIO -write -nrFiles 10  -fileSize 10MB
```

完成之后查看写入速度结果

```
hdfs dfs -text  /benchmarks/TestDFSIO/io_write/part-00000
```

8.2 测试读取速度

测试hdfs的读取文件性能

在HDFS文件系统中读入10个文件,每个文件10M

```
hadoop jar /export/servers/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar  TestDFSIO -read -nrFiles 10 -fileSize 10MB

```

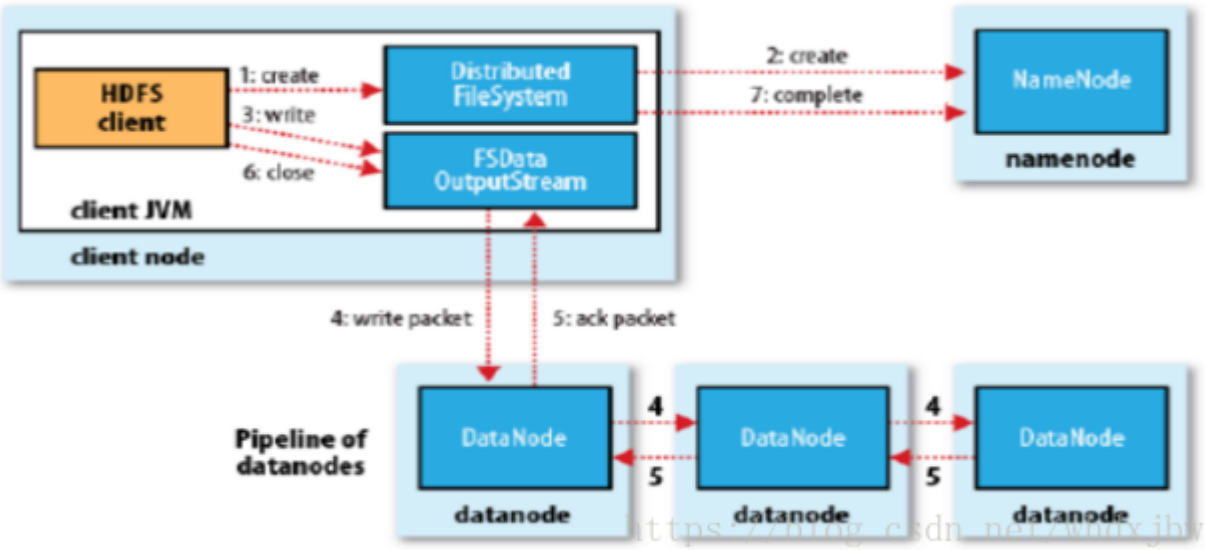
查看读取果

```
hdfs dfs -text /benchmarks/TestDFSIO/io_read/part-00000
```

8.3 清除测试数据

```
hadoop jar /export/servers/hadoop-2.7.5/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.7.5.jar   TestDFSIO -clean
```

9.HDFS 文件写入过程



1. Client 发起文件上传请求, 通过 RPC 与 NameNode 建立通讯, NameNode 检查目标文件是否已存在, 父目录是否存在, 返回是否可以上传
2. Client 请求第一个 block 该传输到哪些 DataNode 服务器上
3. NameNode 根据配置文件中指定的备份数量及机架感知原理进行文件分配, 返回可用的 DataNode 的地址如: A, B, C
    - Hadoop 在设计时考虑到数据的安全与高效, 数据文件默认在 HDFS 上存放三份, 存储策略为本地一份, 同机架内其它某一节点上一份, 不同机架的某一节点上一份。

4. Client 请求 3 台 DataNode 中的一台 A 上传数据（本质上是一个 RPC 调用，建立 pipeline），A 收到请求会继续调用 B, 然后 B 调用 C, 将整个 pipeline 建立完成, 后逐级返回 client
5. Client 开始往 A 上传第一个 block（先从磁盘读取数据放到一个本地内存缓存），以 packet 为单位（默认64K），A 收到一个 packet 就会传给 B, B 传给 C. A 每传一个 packet 会放入一个应答队列等待应答
6. 数据被分割成一个个 packet 数据包在 pipeline 上依次传输, 在 pipeline 反方向上, 逐个发送 ack（命令正确应答），最终由 pipeline 中第一个 DataNode 节点 A 将 pipelineack 发送给 Client
7. 当一个 block 传输完成之后, Client 再次请求 NameNode 上传第二个 block 到服务 1

10.HDFS 文件读取过程

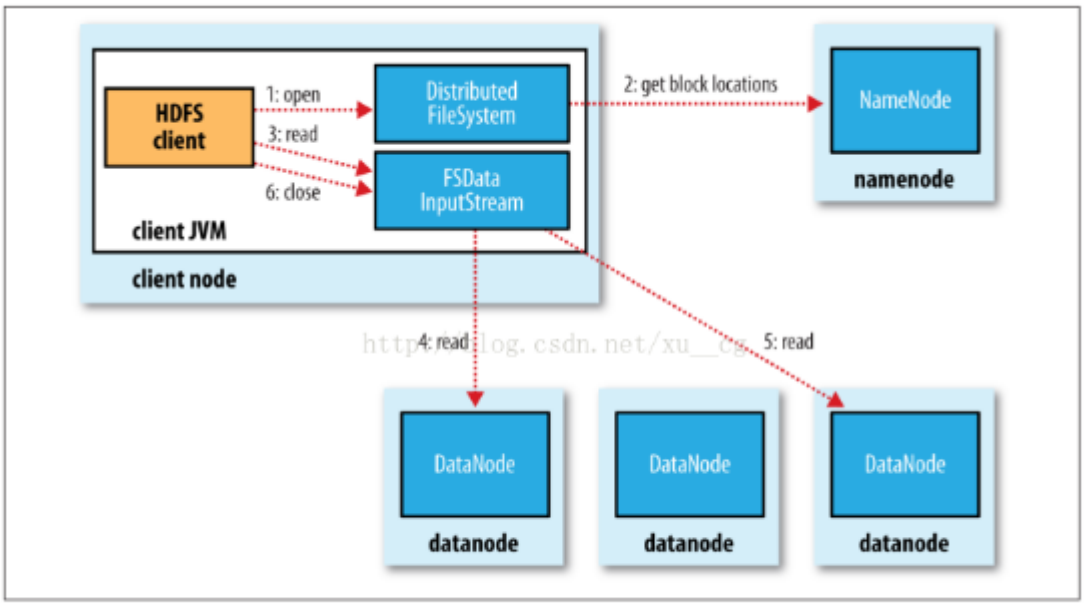


Figure 3-2. A client reading data from HDFS

1. Client向NameNode发起RPC请求，来确定请求文件block所在的位置；
2. NameNode会视情况返回文件的部分或者全部block列表，对于每个block，NameNode 都会返回含有该 block 副本的 DataNode 地址； 这些返回的 DN 地址，会按照集群拓扑结构得出 DataNode 与客户端的距离，然后进行排序，排序两个规则：网络拓扑结构中距离 Client 近的排靠前；心跳机制中超时汇报的 DN 状态为 STALE，这样的排靠后；
3. Client 选取排序靠前的 DataNode 来读取 block，如果客户端本身就是DataNode,那么将从本地直接获取数据(短路读取特性)；
4. 底层上本质是建立 Socket Stream（FSDataInputStream），重复的调用父类 DataInputStream 的 read 方法，直到这个块上的数据读取完毕；
5. 当读完列表的 block 后，若文件读取还没有结束，客户端会继续向NameNode 获取下一批的 block 列表；

6. 读取完一个 block 都会进行 checksum 验证，如果读取 DataNode 时出现错误，客户端会通知 NameNode，然后再从下一个拥有该 block 副本的DataNode 继续读。
7. read 方法是并行的读取 block 信息，不是一块一块的读取；NameNode 只是返回Client请求包含块的DataNode地址，并不是返回请求块的数据；
8. 最终读取来所有的 block 会合并成一个完整的最终文件。

11.HDFS 的元数据辅助管理

当 Hadoop 的集群当中, NameNode的所有元数据信息都保存在了 FsImage 与 Eidts 文件当中, 这两个文件就记录了所有的数据的元数据信息, 元数据信息的保存目录配置在了 `hdfs-site.xml` 当中

```
<property>
    <name>dfs.namenode.name.dir</name>
    <value>
        file:///export/servers/hadoop2.7.5/hadoopDatas/namenodeDatas,
                    file:///export/servers/hadoop-
2.7.5/hadoopDatas/namenodeDatas2
    </value>
</property>
<property>
     <name>dfs.namenode.edits.dir</name>
     <value>file:///export/servers/hadoop-
2.7.5/hadoopDatas/nn/edits</value
</property>>
```

11.1 FsImage 和 Edits 详解

- `edits`
  - `edits` 存放了客户端最近一段时间的操作日志
  - 客户端对 HDFS 进行写文件时会首先被记录在 `edits` 文件中
  - `edits` 修改时元数据也会更新
- `fsimage`
  - NameNode 中关于元数据的镜像, 一般称为检查点, `fsimage` 存放了一份比较完整的元数据信息
  - 因为 `fsimage` 是 NameNode 的完整的镜像, 如果每次都加载到内存生成树状拓扑结构，这是非常耗内存和CPU, 所以一般开始时对 NameNode 的操作都放在 edits 中

- `fsimage` 内容包含了 NameNode 管理下的所有 DataNode 文件及文件 block 及 block 所在的 DataNode 的元数据信息.
- 随着 `edits` 内容增大, 就需要在一定时间点和 `fsimage` 合并

11.2 fsimage 中的文件信息查看

使用命令 `hdfs oiv`

```
cd /export/servers/hadoop2.7.5/hadoopDatas/namenodeDatas
hdfs oiv -i fsimage_0000000000000000864 -p XML -o hello.xml
```

11.3. edits 中的文件信息查看

使用命令 `hdfs oev`

```
cd /export/servers/hadoop2.7.5/hadoopDatas/namenodeDatas
hdfs oev -i  edits_0000000000000000865-0000000000000000866 -p XML -o myedit.xml
```

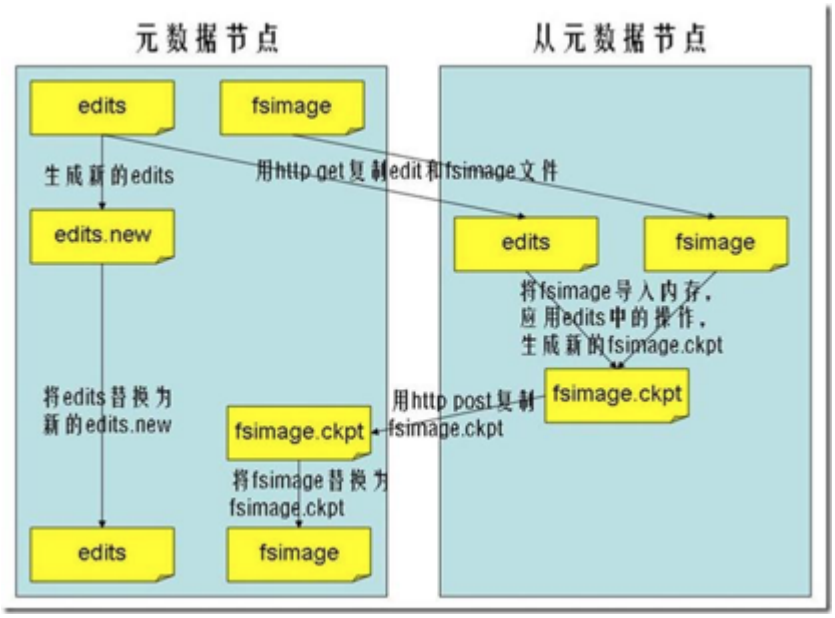
11.4 SecondaryNameNode 如何辅助管理 fsimage 与 edits 文件?

- SecondaryNameNode 定期合并 fsimage 和 edits, 把 edits 控制在一个范围内
- 配置 SecondaryNameNode
  - SecondaryNameNode 在 `conf/masters` 中指定
  - 在 masters 指定的机器上, 修改 `hdfs-site.xml`

```
<property>
  <name>dfs.http.address</name>
  <value>host:50070</value>
</property>
```

  - 修改 `core-site.xml` , 这一步不做配置保持默认也可以

```
<!-- 多久记录一次 HDFS 镜像，默认 1小时 -->
<property>
  <name>fs.checkpoint.period</name>
  <value>3600</value>
</property>
<!-- 一次记录多大，默认 64M -->
<property>
  <name>fs.checkpoint.size</name>
  <value>67108864</value>
</property>
```



1. SecondaryNameNode 通知 NameNode 切换 editlog
2. SecondaryNameNode 从 NameNode 中获得 fsimage 和 editlog(通过http方式)
3. SecondaryNameNode 将 fsimage 载入内存, 然后开始合并 editlog, 合并之后成为新的 fsimage
4. SecondaryNameNode 将新的 fsimage 发回给 NameNode
5. NameNode 用新的 fsimage 替换旧的 fsimage

特点

- 完成合并的是 SecondaryNameNode, 会请求 NameNode 停止使用 edits, 暂时将新写操作放入一个新的文件中 `edits.new`
- SecondaryNameNode 从 NameNode 中通过 Http GET 获得 edits, 因为要和 fsimage 合并, 所以也是通过 Http Get 的方式把 fsimage 加载到内存, 然后逐一执行具体对文件系统的操作, 与 fsimage 合并, 生成新的 fsimage, 然后通过 Http POST 的方式把 fsimage 发送给 NameNode. NameNode 从 SecondaryNameNode 获得了 fsimage 后会把原有的 fsimage 替换为新的 fsimage, 把 edits.new 变成 edits. 同时会更新 fstime

- Hadoop 进入安全模式时需要管理员使用 dfsadmin 的 save namespace 来创建新的检查点
- SecondaryNameNode 在合并 edits 和 fsimage 时需要消耗的内存和 NameNode 差不多, 所以一般把 NameNode 和 SecondaryNameNode 放在不同的机器上