HDFS分布式文件系统

刘军（liujun@bupt.edu.cn）
北京邮电大学 数据科学中心

每周新鲜事


ORACLE
+
腾讯云
amazon.com
web services
+
光环新网
sinnet
Windows Azure
IBM
+
世纪互联
www.21vianet.com
iCloud
+
中国电信
CHINA TELECOM

HDFS起因

海量数据的存储需求

- 数据量级
  - 百度：> 200PB
  - Facebook：> 100PB
  - Yahoo：> 100PB
  - 淘宝：> 15PB
  - ebay：> 10PB
  - 中国电信云公司：>30PB

- 环境特点：
  - 低成本：大量廉价PC构成的集群作为硬件基础，单节点故障率较高
  - 大文件：大量大尺寸的文件（≥100MB – GB级）
  - 读写特性：顺序读写，极少随机读写；写入后，一般不会再修改
  - 目标：要求系统整体高吞吐量，而非低延时

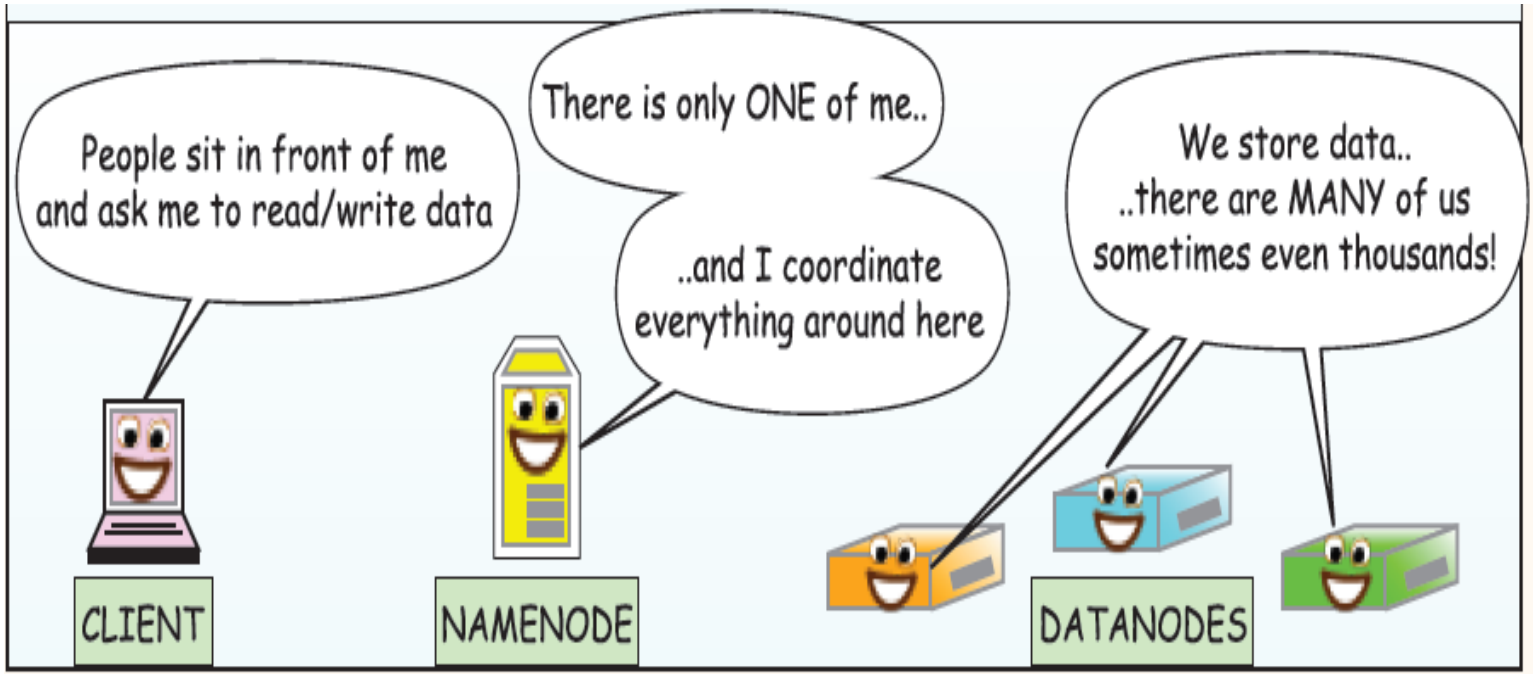
HDFS

The Hadoop Distributed File System (HDFS) is a **distributed** file system designed to run on **commodity hardware**. It has many **similarities** with existing distributed file systems. However, the **differences** from other distributed file systems are significant. HDFS is **highly fault-tolerant** and is designed to be deployed on low-cost hardware. HDFS provides **high throughput** access to application data and is suitable for applications that have **large data sets**.

Source : http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction

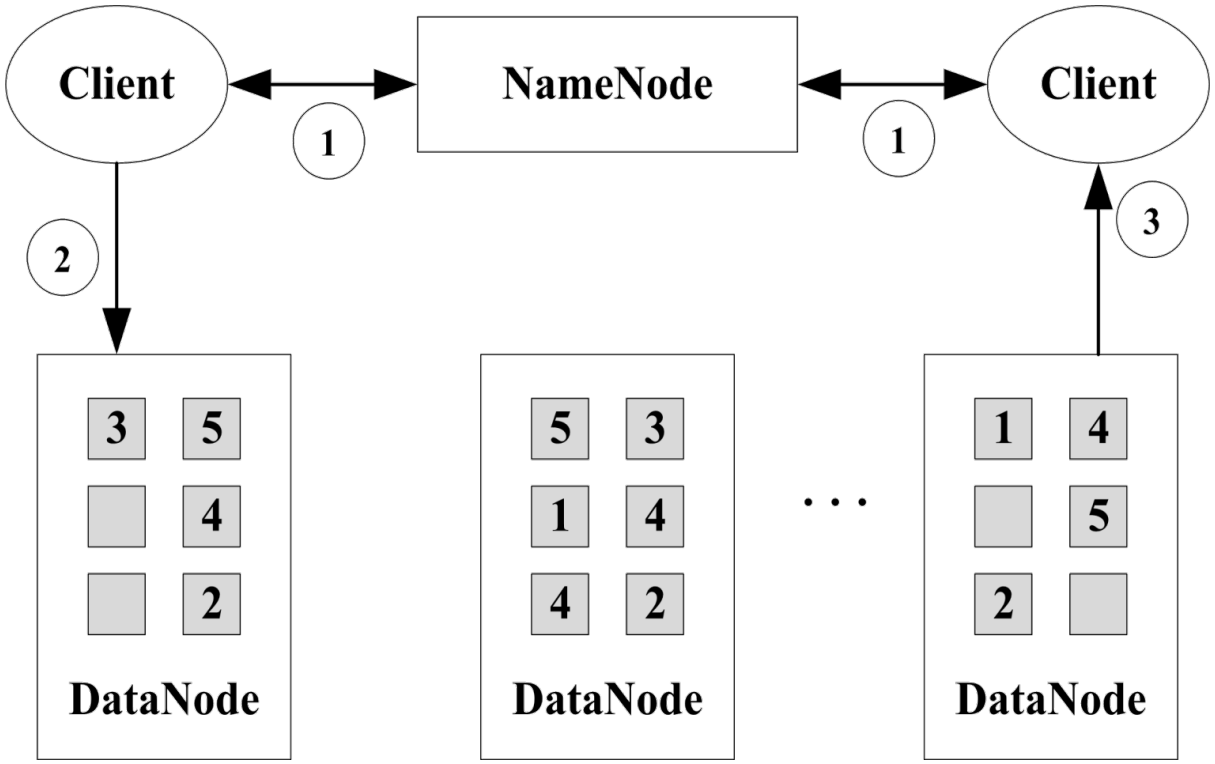
HDFS架构

节点分工



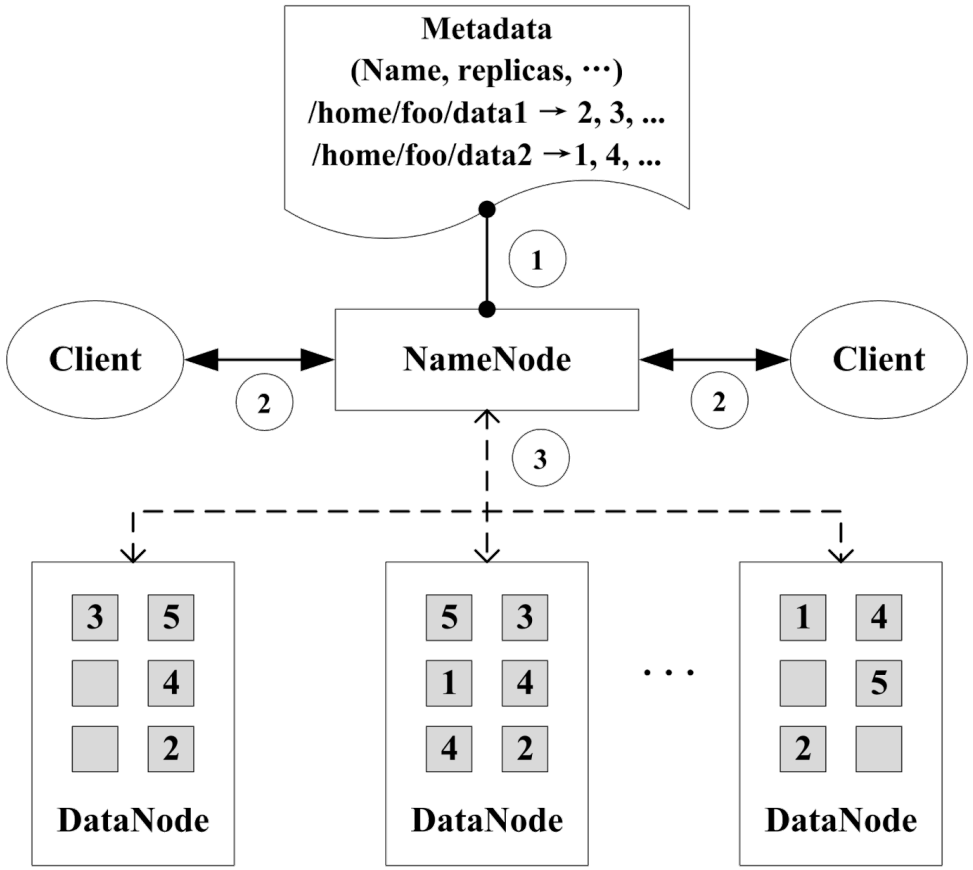
By Maneesh Varshney, mvarshney@gmail.com（后同）

Client



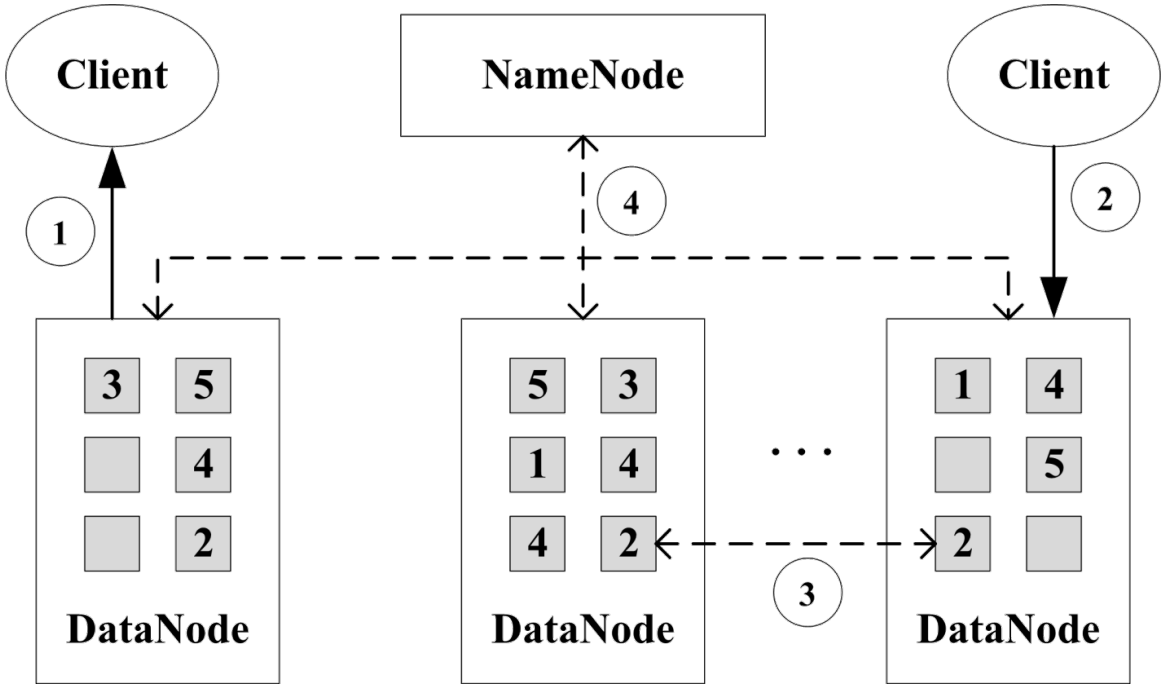
- Client
  - 系统使用者，调用 HDFS API 操作文件
  - 与NN交互，进行文件元数据操作，以及获取文件元数据（①）
  - 与DN交互，进行数据读写（②③）

NameNode（NN）



- NameNode
  - 系统唯一管理者
  - 管理元数据（①）
    - 保存和管理文件系统的命名空间，包括目录树、文件和目录的元数据等
    - 保存和构造文件与数据块所在DN的对应关系（①）
  - 与Client交互，进行文件操作与元数据查询（②）
  - 接收DN的数据块报告，更新元数据，分配数据存储节点（③）

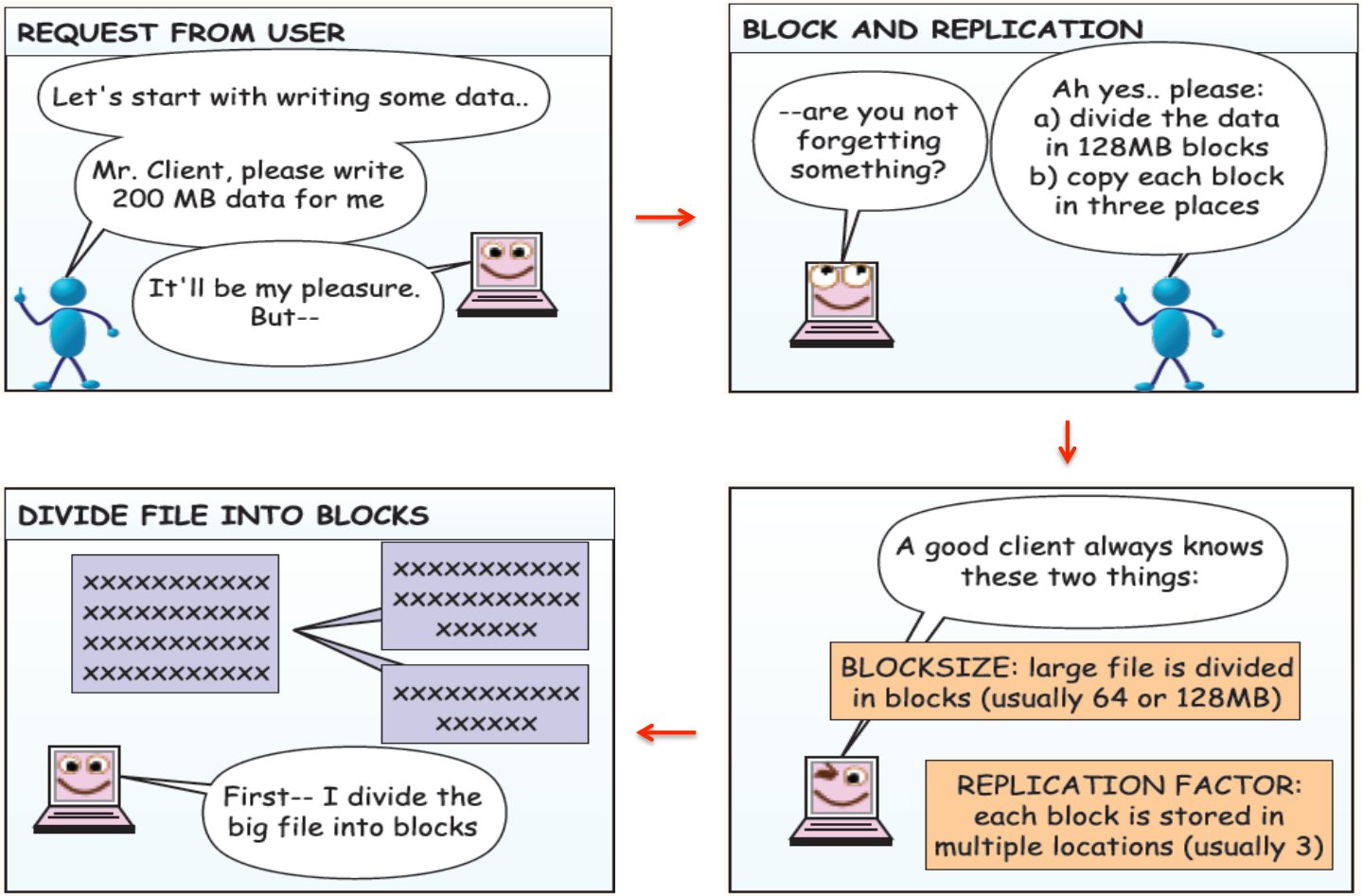
DataNode（DN）



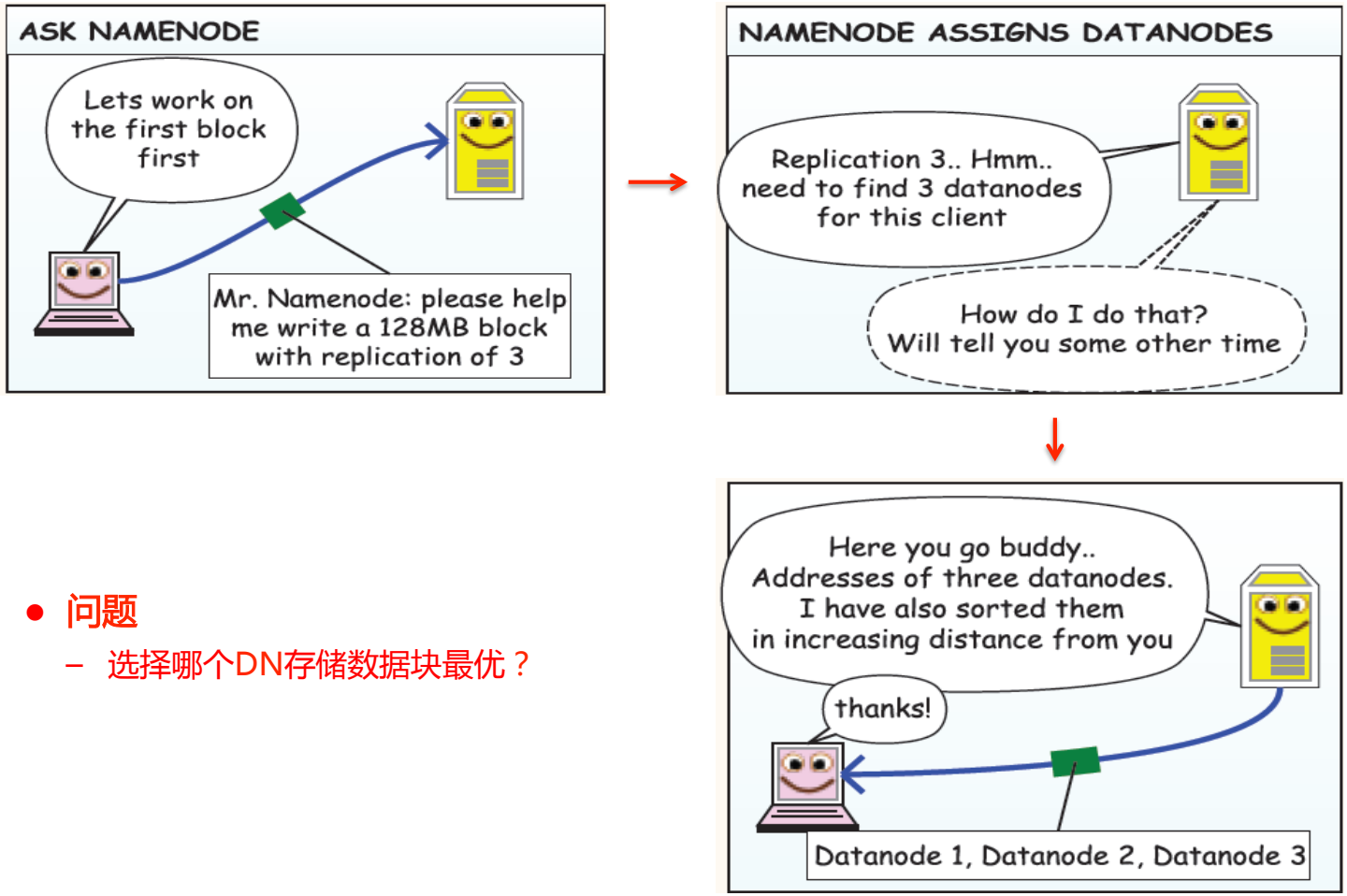
- DataNode
  - 数据存储节点
  - 与Client交互，进行数据读写（①②）
  - 文件分割为数据块，冗余备份存储在多个DN中（③）
  - 向NN定期报告数据块存储情况（④）Client

HDFS数据写入

（1）文件拆分

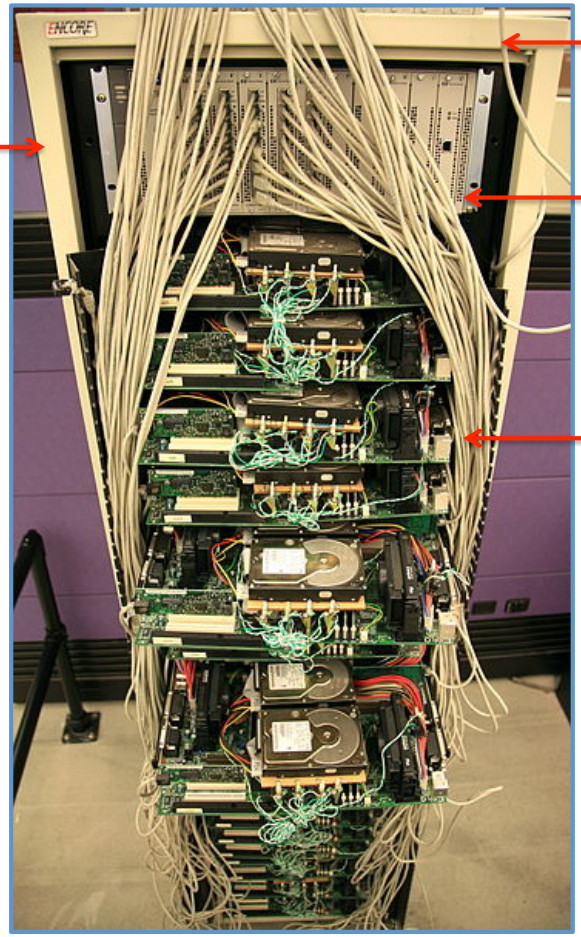

REQUEST FROM USER
Let's start with writing some data..
Mr. Client, please write 200 MB data for me
It'll be my pleasure. But--
BLOCK AND REPLICATION
--are you not forgetting something?
Ah yes.. please: a) divide the data in 128MB blocks b) copy each block in three places
A good client always knows these two things:
BLOCKSIZE: large file is divided in blocks (usually 64 or 128MB)
REPLICATION FACTOR: each block is stored in multiple locations (usually 3)
DIVIDE FILE INTO BLOCKS
First-- I divide the big file into blocks

（2）分配DN



- 问题
  - 选择哪个DN存储数据块最优？

先了解下服务器机架



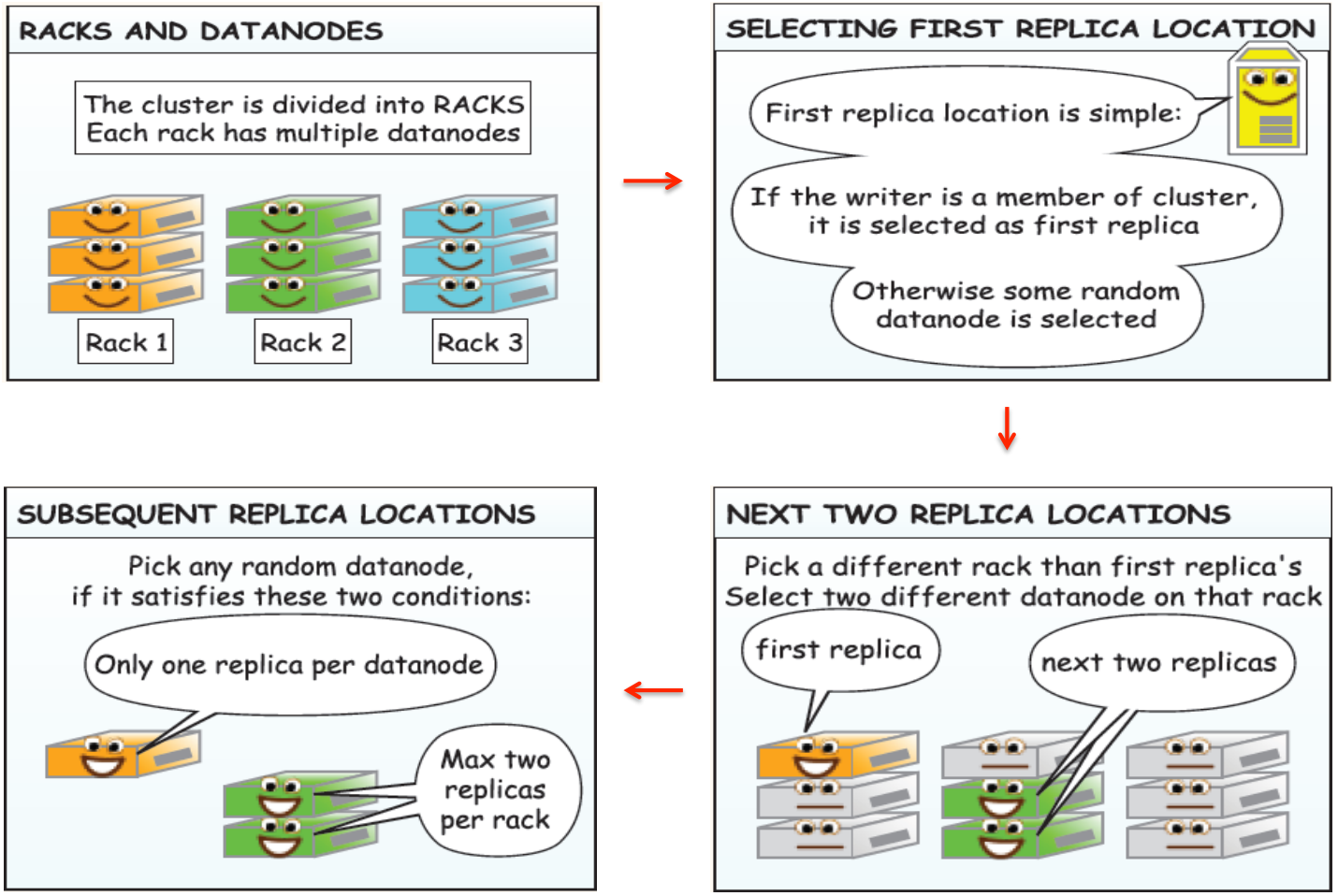
机架

架间交换机（共享，拥挤）

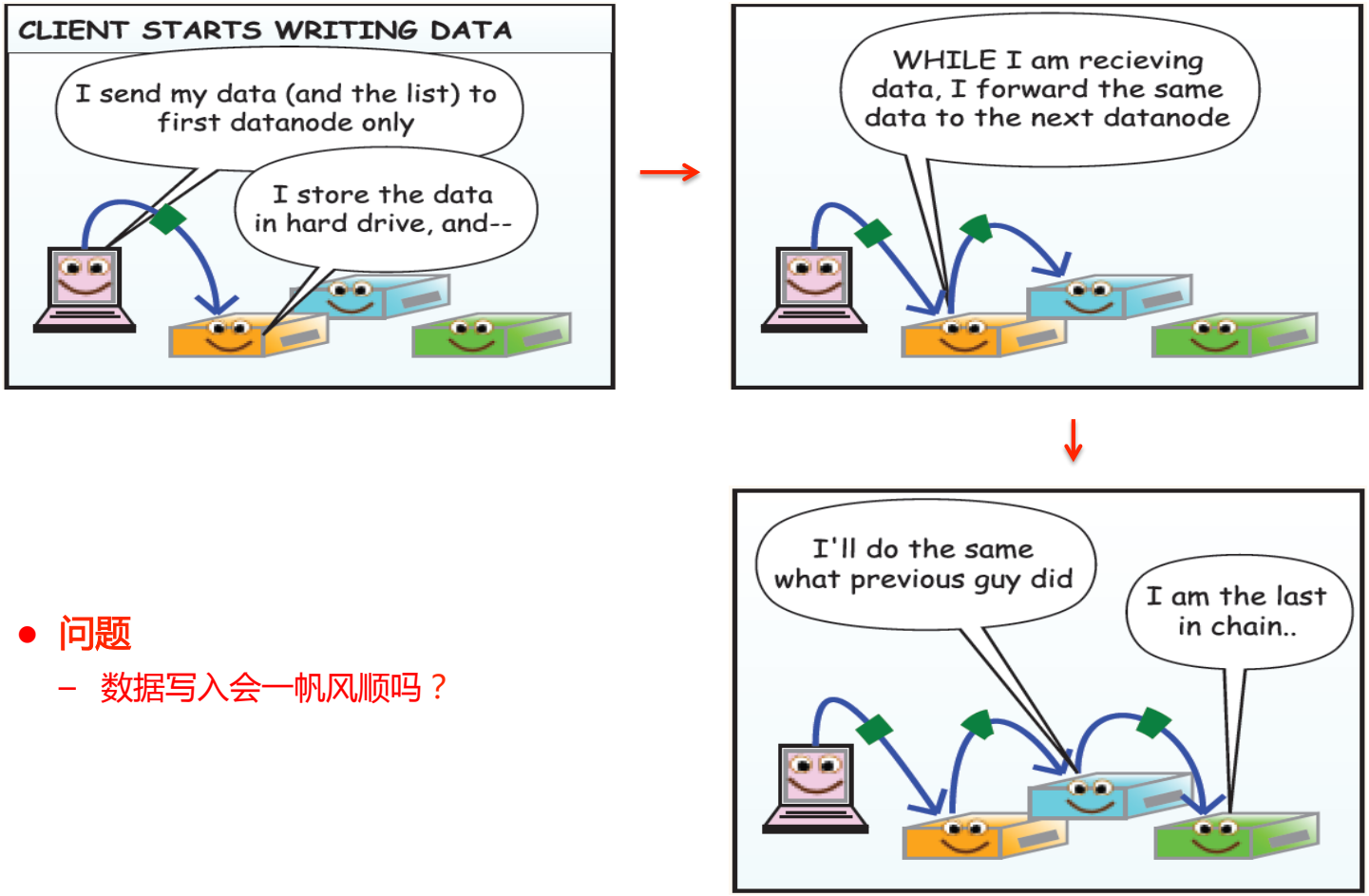
机架交换机（专用，快）

服务器

机架感知的写入策略

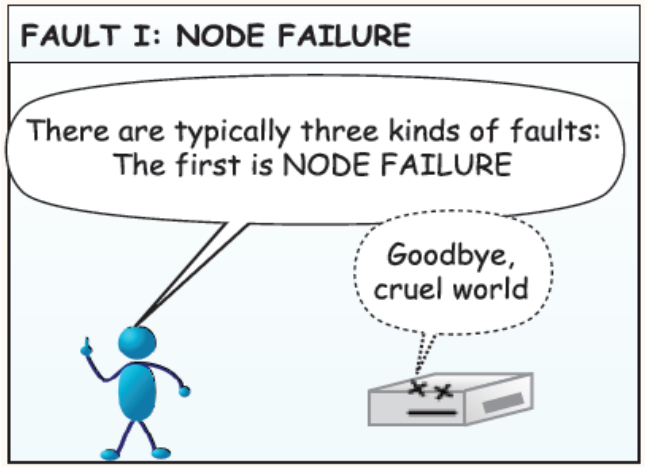

RACKS AND DATANODES
The cluster is divided into RACKS
Each rack has multiple datanodes
Rack 1
Rack 2
Rack 3
SELECTING FIRST REPLICA LOCATION
First replica location is simple:
If the writer is a member of cluster, it is selected as first replica
Otherwise some random datanode is selected
NEXT TWO REPLICA LOCATIONS
Pick a different rack than first replica's
Select two different datanode on that rack
first replica
next two replicas
SUBSEQUENT REPLICA LOCATIONS
Pick any random datanode,
if it satisfies these two conditions:
Only one replica per datanode
Max two replicas per rack

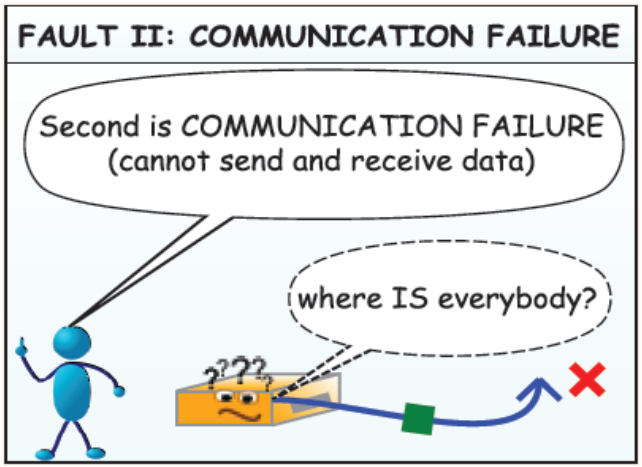
（3）写入DN

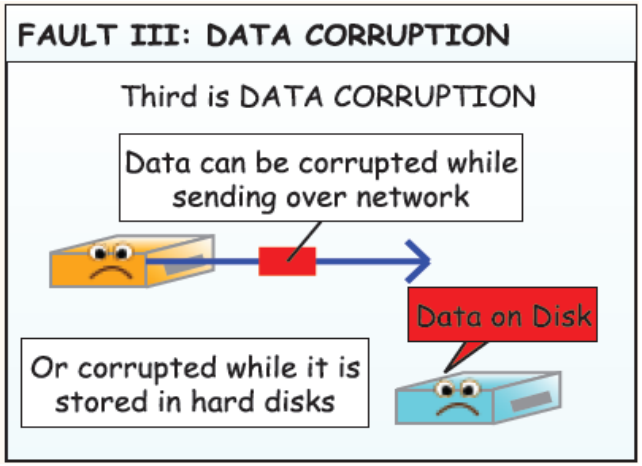


- 问题
  - 数据写入会一帆风顺吗？

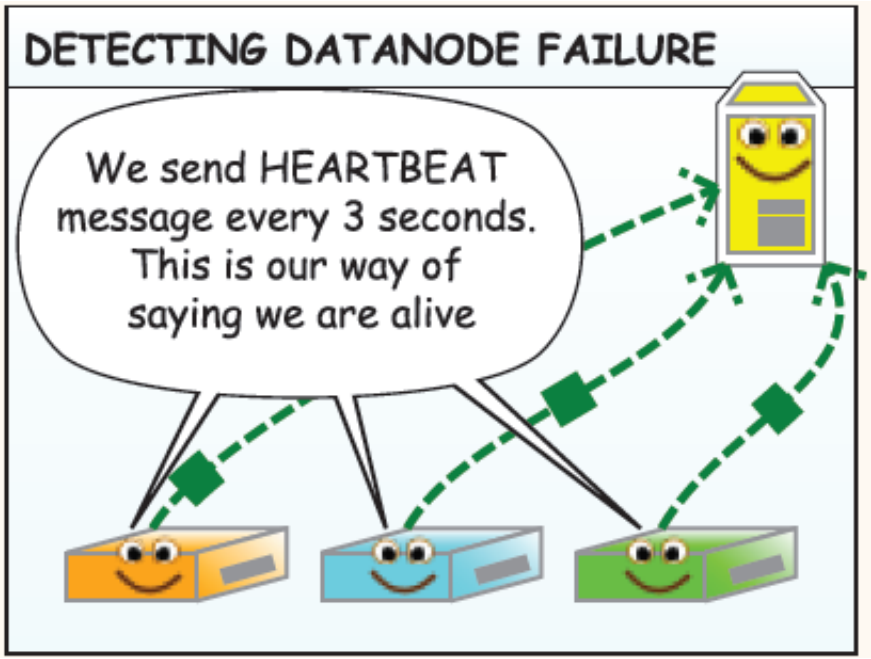
不会，所以要容错：节点、网络、存储

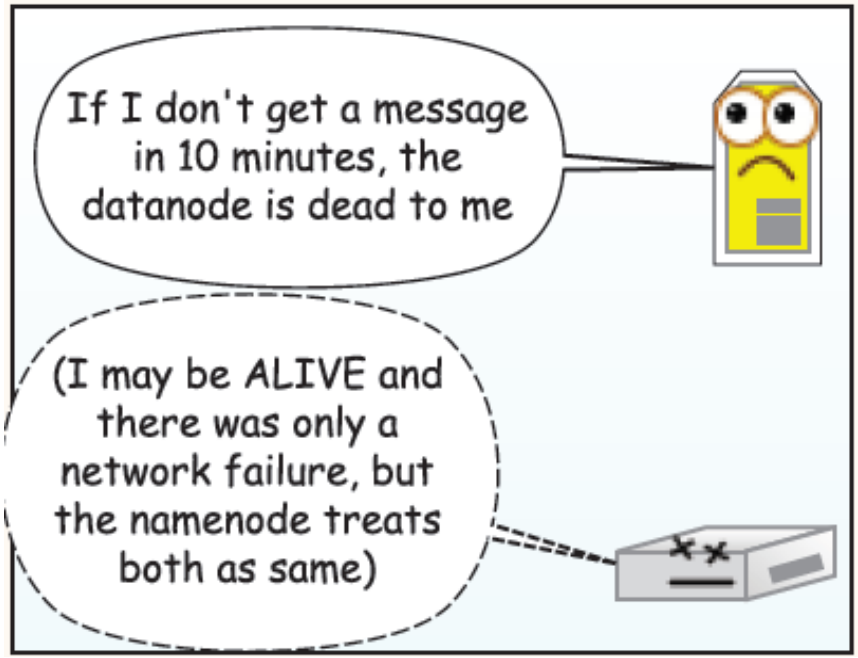




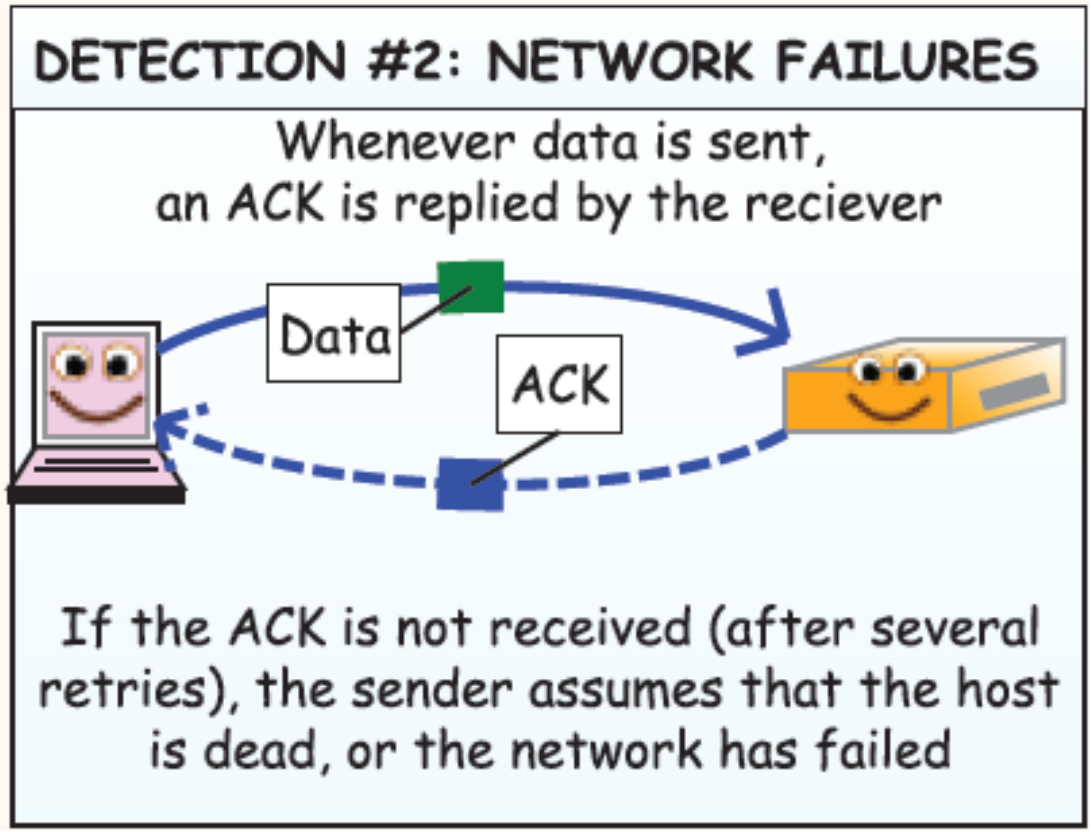


容错之前，得先能监测错误。如何监测节点错误？

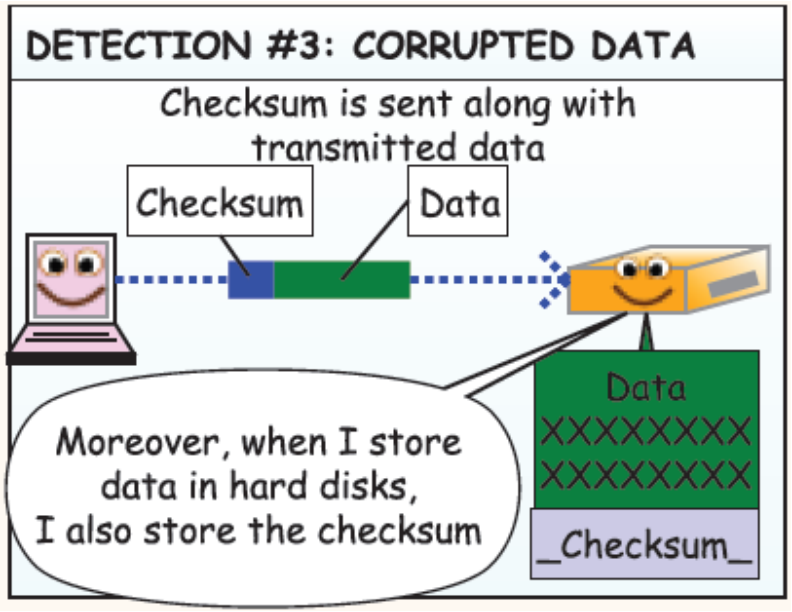


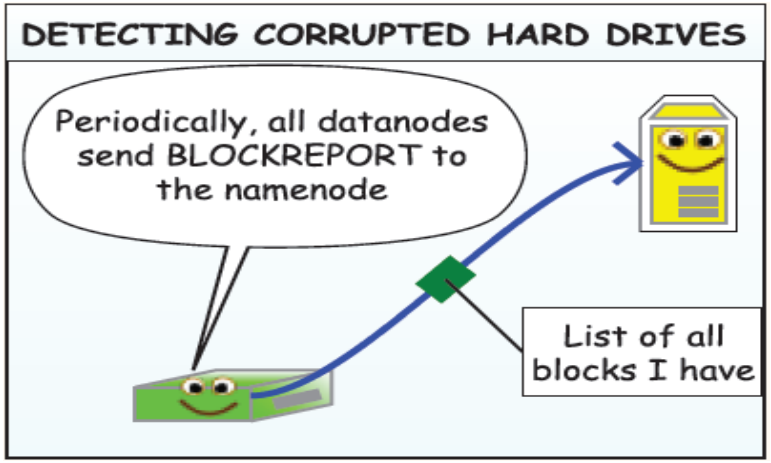


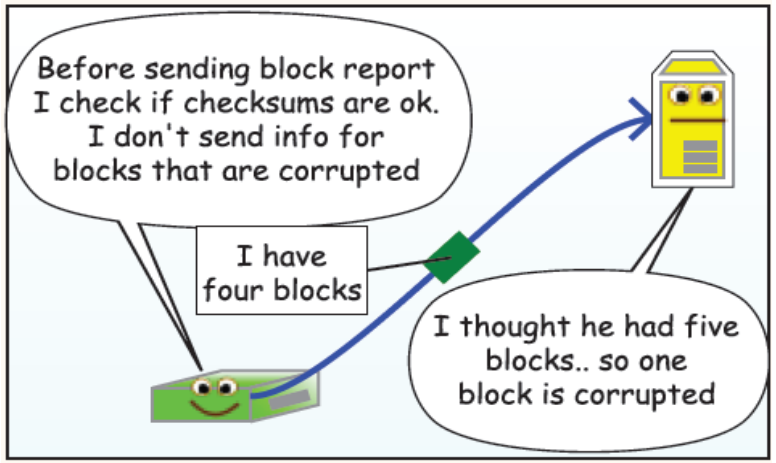
如何监测网络错误？


DETECTION #2: NETWORK FAILURES
Whenever data is sent,
an ACK is replied by the reciever
Data
ACK
If the ACK is not received (after several retries), the sender assumes that the host is dead, or the network has failed

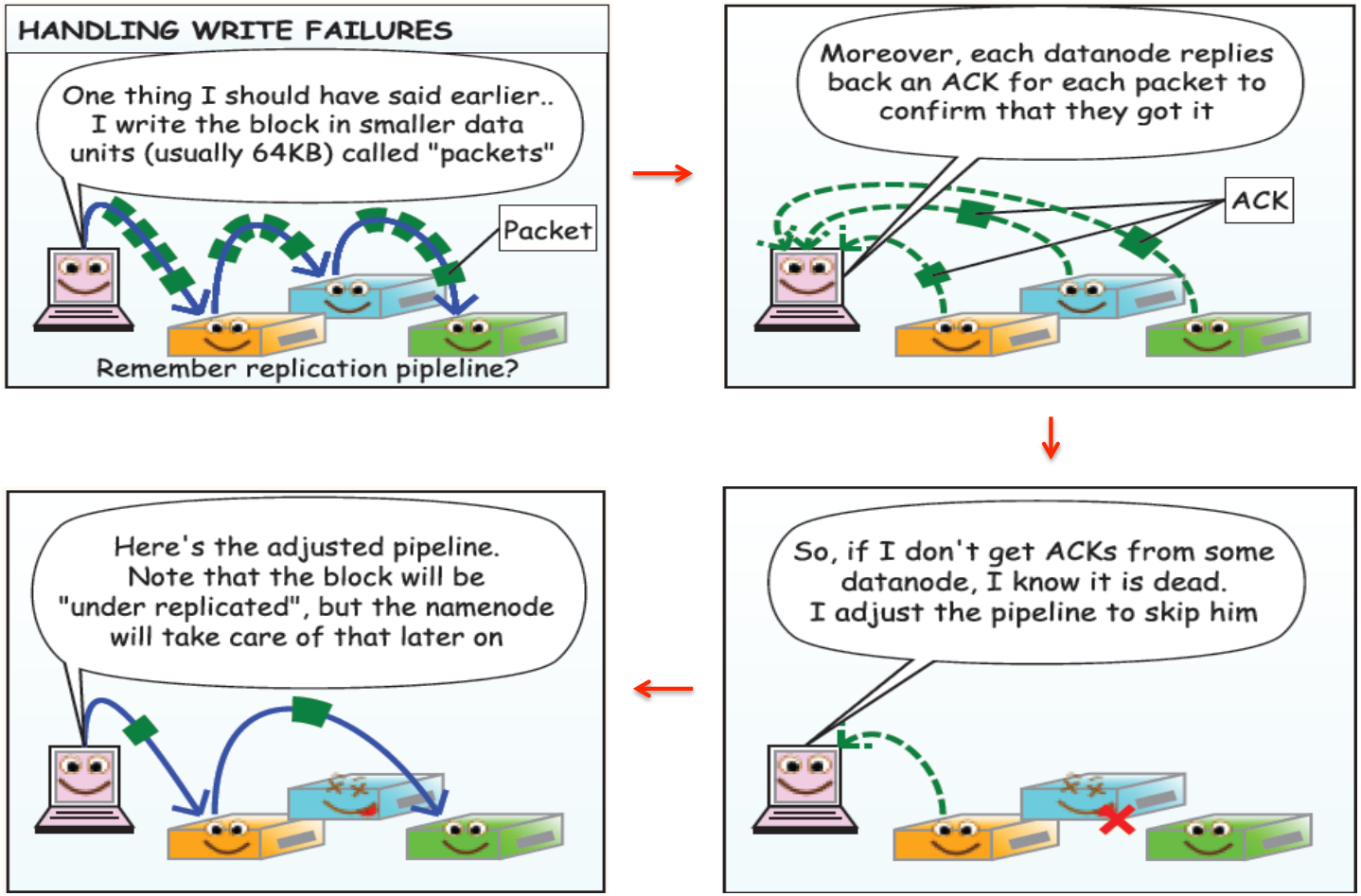
如何监测存储错误？


DETECTION #3: CORRUPTED DATA
Checksum is sent along with transmitted data
Checksum
Data
Moreover, when I store data in hard disks, I also store the checksum
Data
XXXXXXXX
XXXXXXXX
Checksum

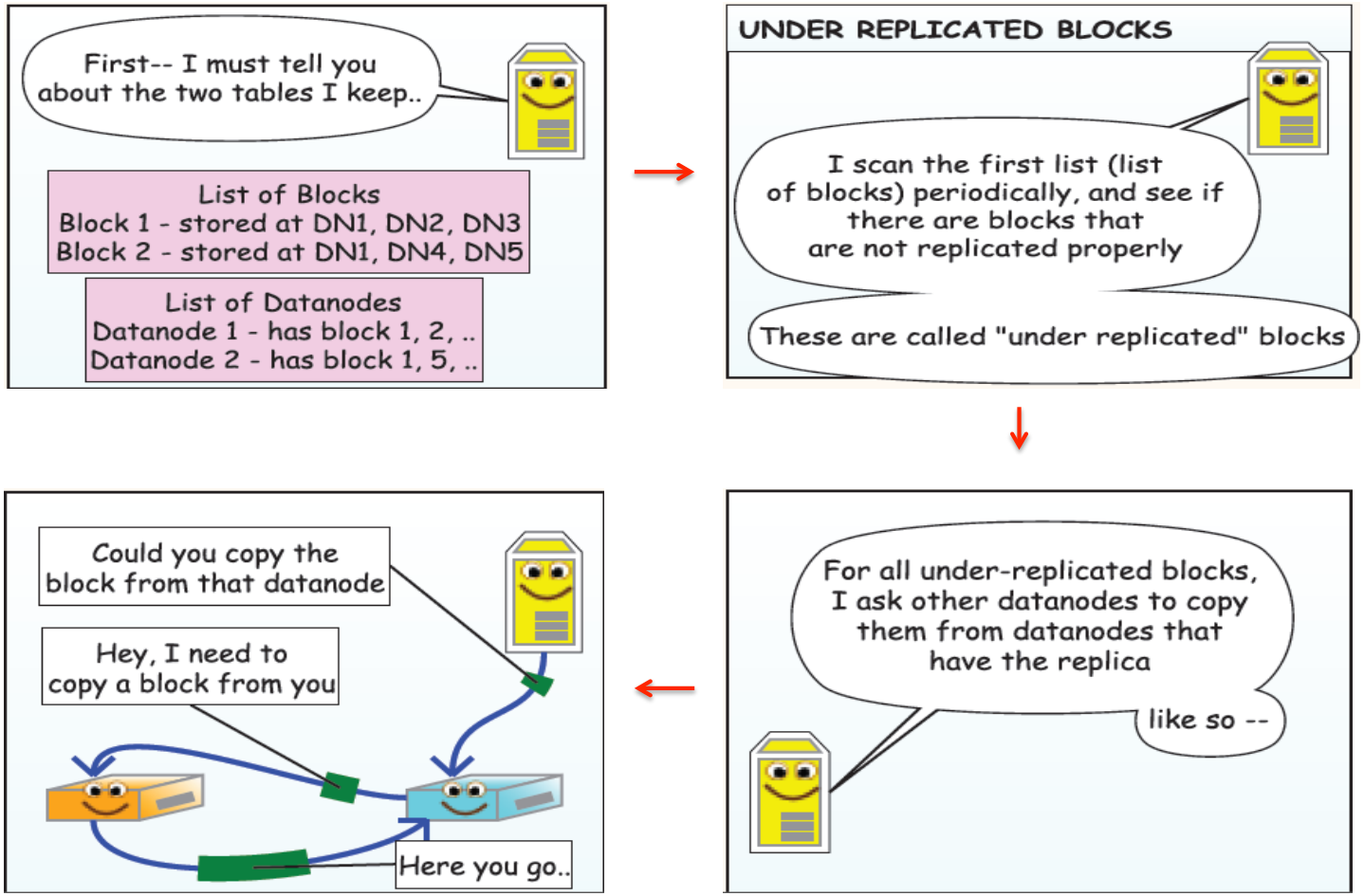

DETECTING CORRUPTED HARD DRIVES
Periodically, all datanodes send BLOCKREPORT to the namenode
List of all blocks I have


Before sending block report I check if checksums are ok. I don't send info for blocks that are corrupted
I have four blocks
I thought he had five blocks.. so one block is corrupted

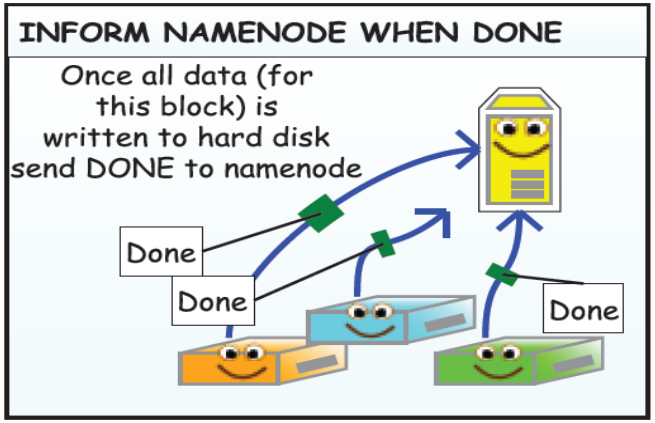
写数据的容错：写入时

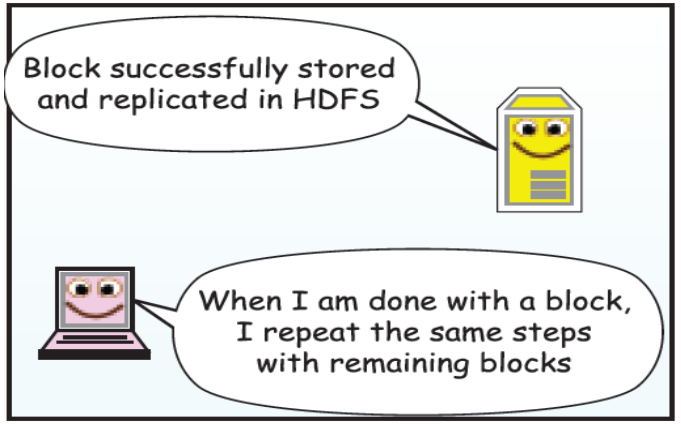

HANDLING WRITE FAILURES
One thing I should have said earlier.. I write the block in smaller data units (usually 64KB) called "packets"
Packet
Remember replication pipeline?
Moreover, each datanode replies back an ACK for each packet to confirm that they got it
ACK
So, if I don't get ACKs from some datanode, I know it is dead. I adjust the pipeline to skip him
Here's the adjusted pipeline. Note that the block will be "under replicated", but the namenode will take care of that later on

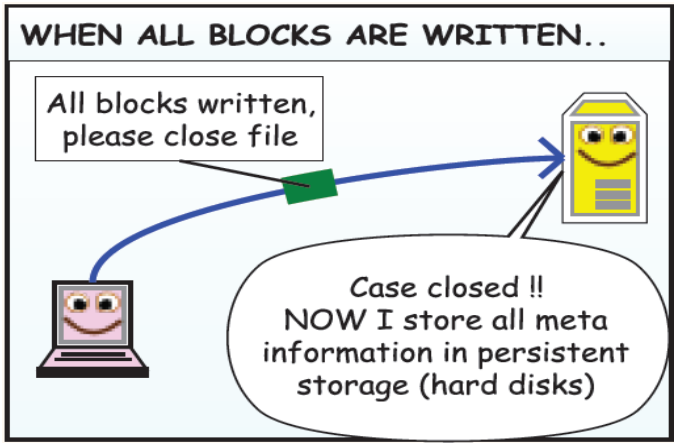
写数据的容错：写入后


First-- I must tell you about the two tables I keep..
List of Blocks
Block 1 - stored at DN1, DN2, DN3
Block 2 - stored at DN1, DN4, DN5
List of Datanodes
Datanode 1 - has block 1, 2, ..
Datanode 2 - has block 1, 5, ..
UNDER REPLICATED BLOCKS
I scan the first list (list of blocks) periodically, and see if there are blocks that are not replicated properly
These are called "under replicated" blocks
For all under-replicated blocks, I ask other datanodes to copy them from datanodes that have the replica
like so --
Could you copy the block from that datanode
Hey, I need to copy a block from you
Here you go..

（4）完成写入

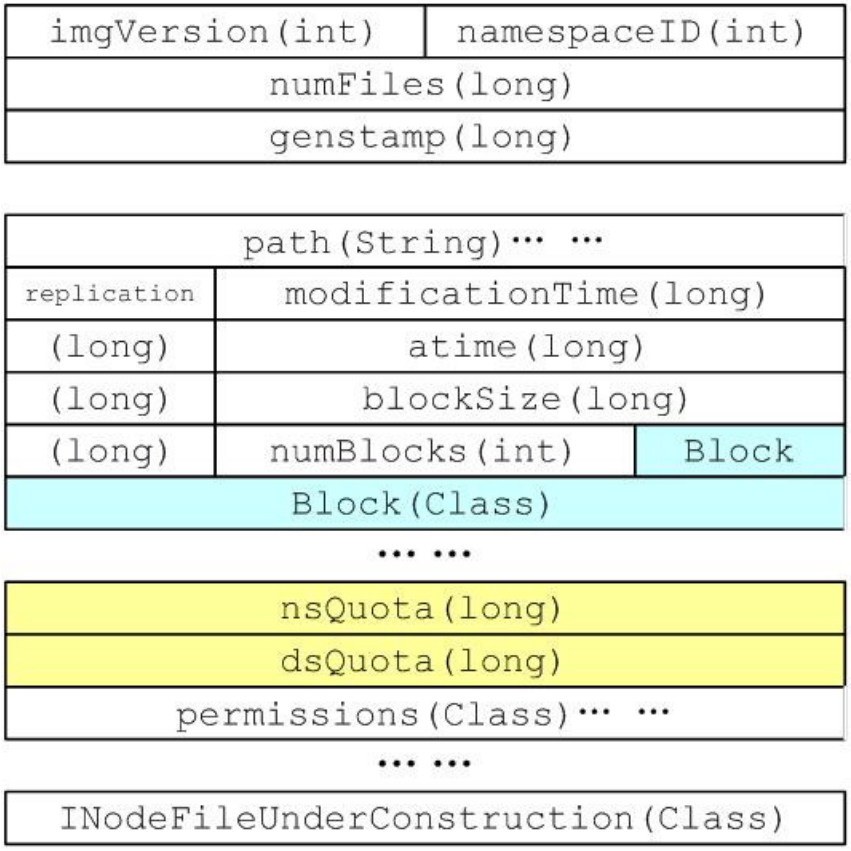






- 问题
  - NameNode看起来很闲嘛，是这样吗？

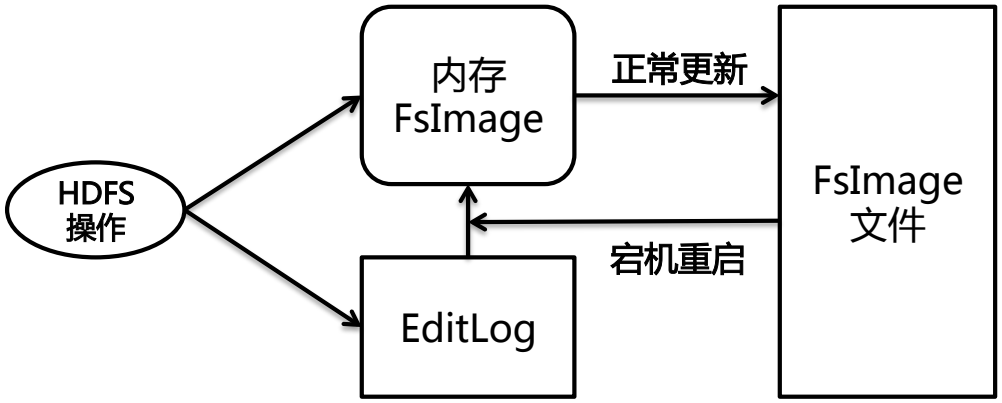
NameNode中目录/文件的元数据



FsImage文件

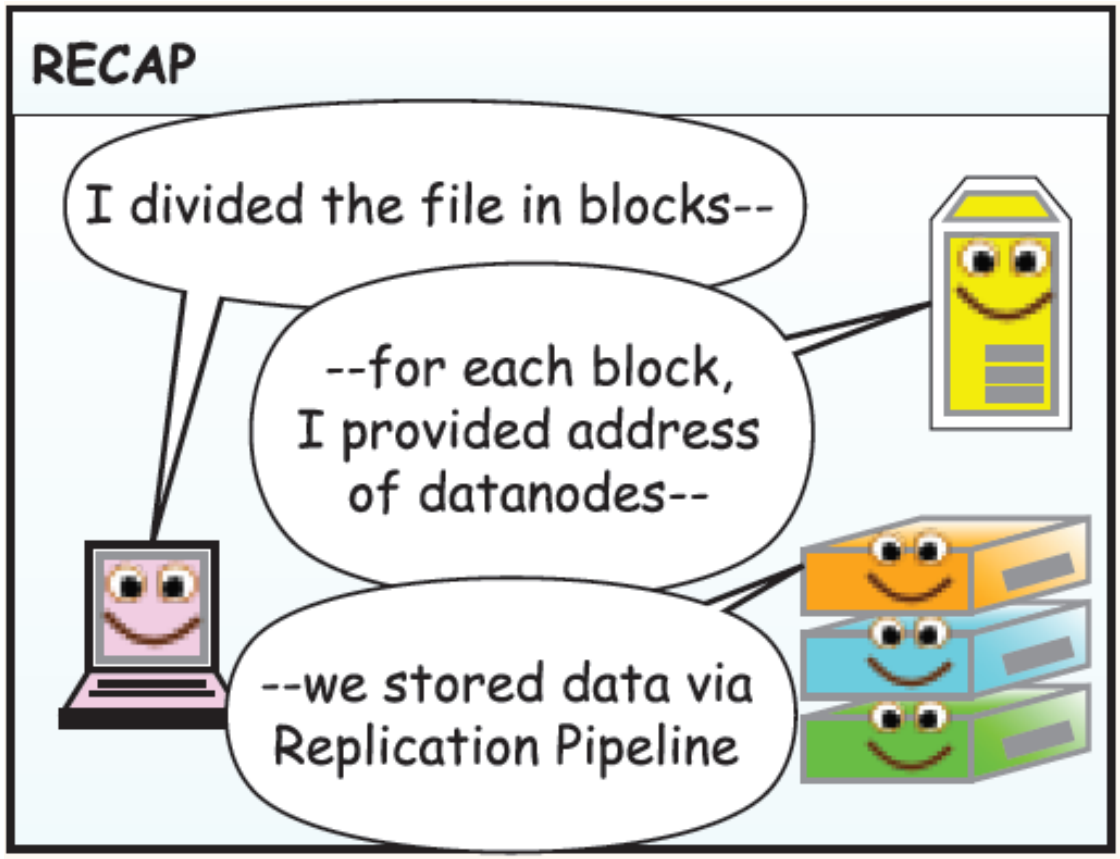
NameNode的效率与可靠性

- 每次HDFS文件操作更新FsImage文件？
  - NO！效率太低！ 有没有办法解决？
- FsImage保存到内存中？
  - 有改进，但NN宕机怎么办？
- **FsImage + 内存元数据 + EditLog = 效率 + 安全性**



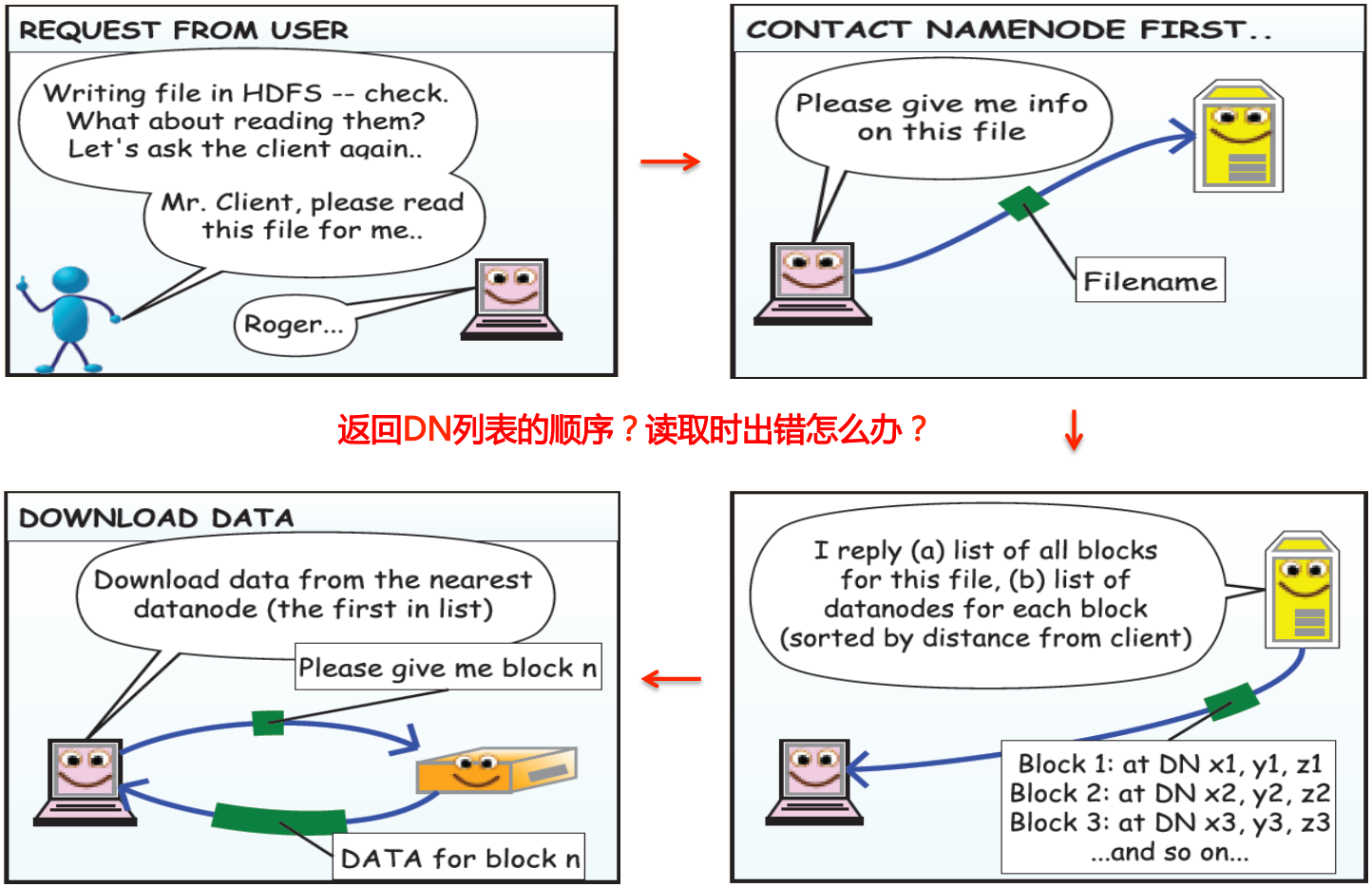
  - EditLog达到一定的大小（fs.checkpoint.size）或从上次保存过后一段时间后（fs.checkpoint.period），NN将内存中文件和目录元数据更新到FsImage中
  - NN启动时，将EditLog与FsImage合并重构内存元数据

写入数据总结


RECAP
I divided the file in blocks--
--for each block, I provided address of datanodes--
--we stored data via Replication Pipeline

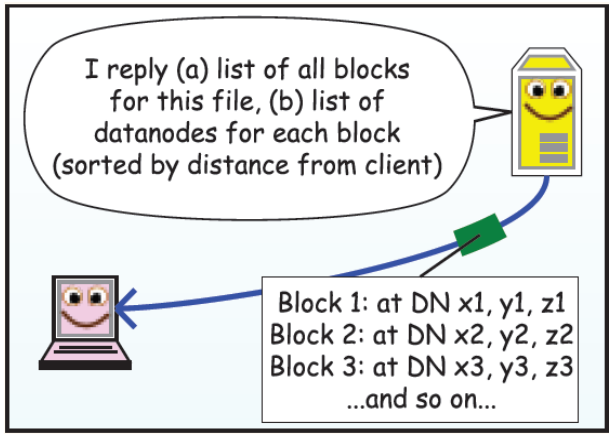
HDFS数据读取

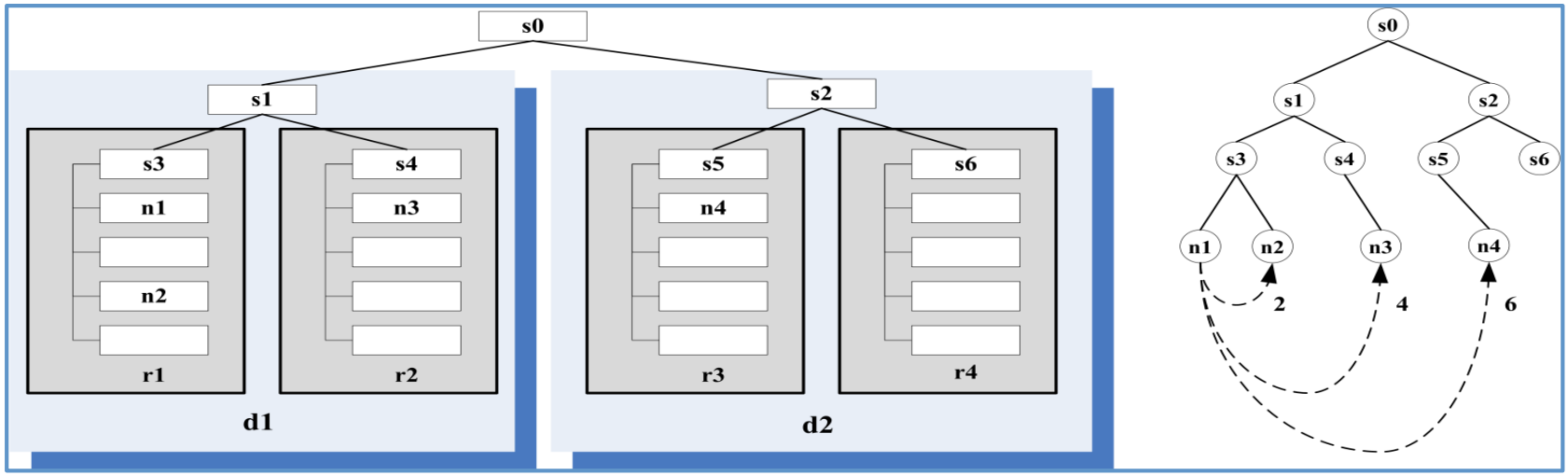
读取数据


REQUEST FROM USER
Writing file in HDFS -- check.
What about reading them?
Let's ask the client again..
Mr. Client, please read this file for me..
Roger...
CONTACT NAMENODE FIRST..
Please give me info on this file
Filename
返回DN列表的顺序？读取时出错怎么办？
DOWNLOAD DATA
Download data from the nearest datanode (the first in list)
Please give me block n
DATA for block n
I reply (a) list of all blocks for this file, (b) list of datanodes for each block (sorted by distance from client)
Block 1: at DN x1, y1, z1
Block 2: at DN x2, y2, z2
Block 3: at DN x3, y3, z3
...and so on...

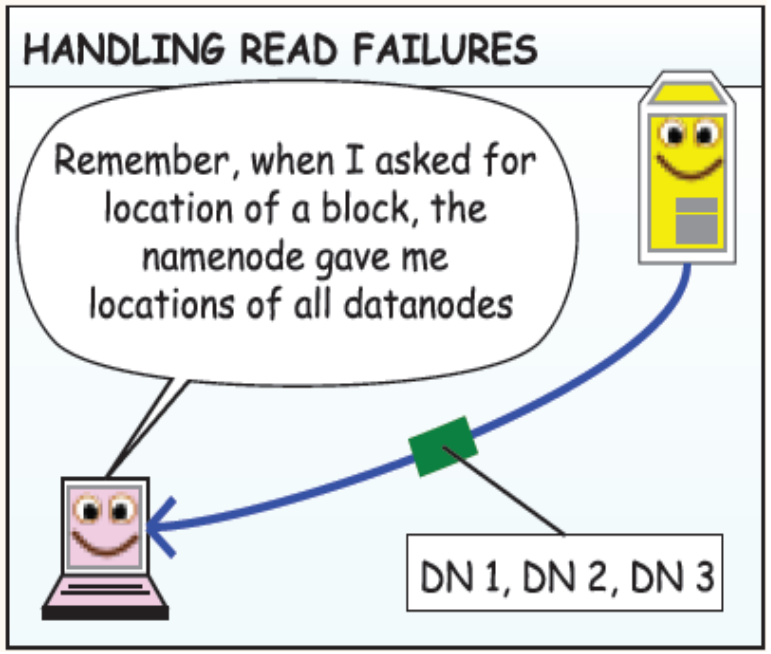
机架感知的读取策略

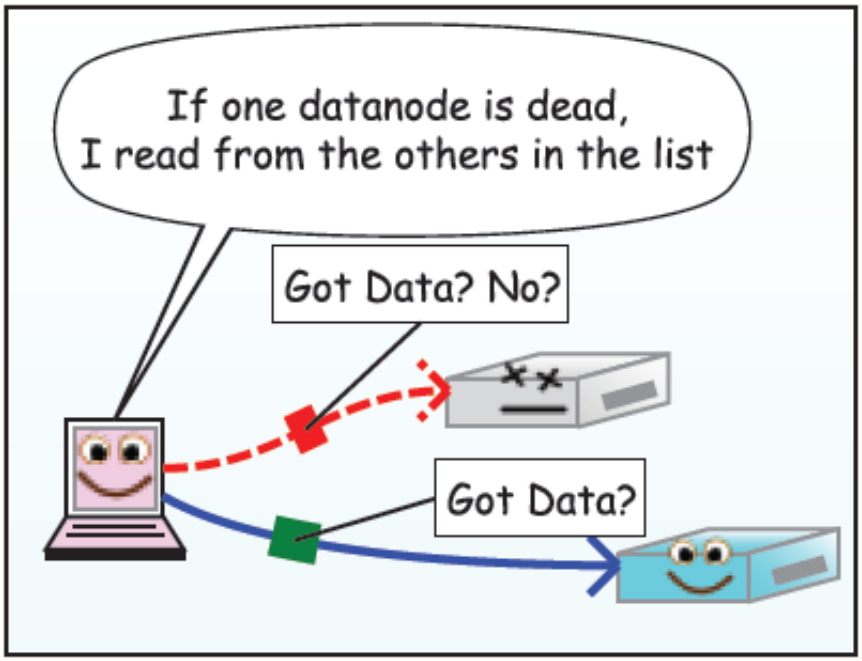
- 机架感知的数据读取
  - NN返回Client按距离排序后的DN列表
  - Client选择距离最小的DN读取





如何处理读取数据时的错误？





使用HDFS

Shell

- HDFS的命令行格式：hadoop fs –cmd *<args>*
  - cmd：具体的指令内容
  - *<args>*：一系列可变的参数

命令	说明	示例
cat	显示非压缩文件的内容	-cat hdfs://nn1.exp.com/file1
chmod	改变文件或目录的访问权限	-chmod 777 hdfs://nn1.exp.com/file1
chown	变更文件或目录的所有者	-chown user1 hdfs://nn1.exp.com/file1
copyFromLocal	从本地复制文件到HDFS文件系统中	-copyFromLocal localf1 /usr/hadoop/file1
copyToLocal	从HDFS复制文件到本地	-copyToLocal /usr/hadoop/file1 localf1
cp	复制文件或目录	-cp /usr/hadoop/file1 /usr/hadoop/file2
get	从HDFS复制文件到本地	-get /usr/hadoop/file1 localf1
ls	列出文件和子目录	-ls /usr/hadoop/file1
mkdir	创建目录	-mkdir /usr/hadoop/dir2
moveFromLocal	将文件或目录从本地移动到HDFS	-moveFromLocal localf1 /usr/hadoop/file1
mv	将文件或目录源路径移动到目的路径	-mv /usr/hadoop/file1 /usr/hadoop/file2
put	将文件或目录从本地移动复制HDFS	-put localfile1 /usr/hadoop/file1
rm	删除文件或空目录	-rm /usr/hadoop/dir1
tail	显示文件的最后1KB数据	-tail /usr/hadoop/file1

Java API

- 示例：合并多个本地话单文件写入HDFS

```
1:  public class MergeCDRFiles{
2:    public static void main(String[] args) throws IOException {
3:      Configuration conf = new Configuration();
4:      FileSystem localInputFS = FileSystem.getLocal(conf);
5:      FileSystem hdfsOutputFS = FileSystem.get(conf);
6:      Path inputLocalDir = new Path(args[0]);
7:      Path hdfsOutputFile = new Path(args[1]);
8:      FileStatus[] inputFiles = localInputFS.listStatus(inputDir);
9:      FSDataOutputStream out = hdfsOutputFS.create(hdfsOutputFile);
10:     for (int i=0; i<inputFiles.length; i++) {
11:       FSDataInputStream in= localInputFS.open(inputFiles[i].getPath());
12:       byte buffer[] = new byte[256];
13:       int bytesRead = 0;
14:       while( (bytesRead = in.read(buffer)) > 0) {
15:         out.write(buffer, 0, bytesRead);  }
16:       in.close();
17:     }
18:     out.close();
19:     localInputFS.delete(inputLocalDir);
20:   }
21: }
```

- **实例化FileSystem对象**
  - 本地文件：LocalFileSystem
  - HDFS文件：DistributedFileSystem
- **获得文件列表**
  - FileSystem.listStatus()
  - FileStatus：文件和目录的路径、文件长度、块大小、备份数量、修改时间、所属用户以及权限信息等
- **创建文件**
  - FileSystem.create()
- **读取文件内容**
  - FileSystem.open()
  - FSDataInputStream.read()
- **数据写入文件**
  - FSDataOutputStream.write()
- **删除文件**
  - FileSystem.delete()

下周课为：实践课

- 在课程平台观看第4章《HDFS原理与操作》的原理部分，并练习：
  - HDFS基本操作
- 提前自学Git的使用方法
- 上课时请携带笔记本电脑，并安装SSH客户端

数据科学中心
Center for Data Science

刘军
北京邮电大学 信通院 宽带网络监控教研中心
北邮本部 明光楼七层
邮件地址：liujun@bupt.edu.cn
新浪微博：北邮刘军
电　　话：010-62283742