COMP9313

Week 2

Lecturer：

Elijah

Python 教程推荐

morvanzhou.github.io

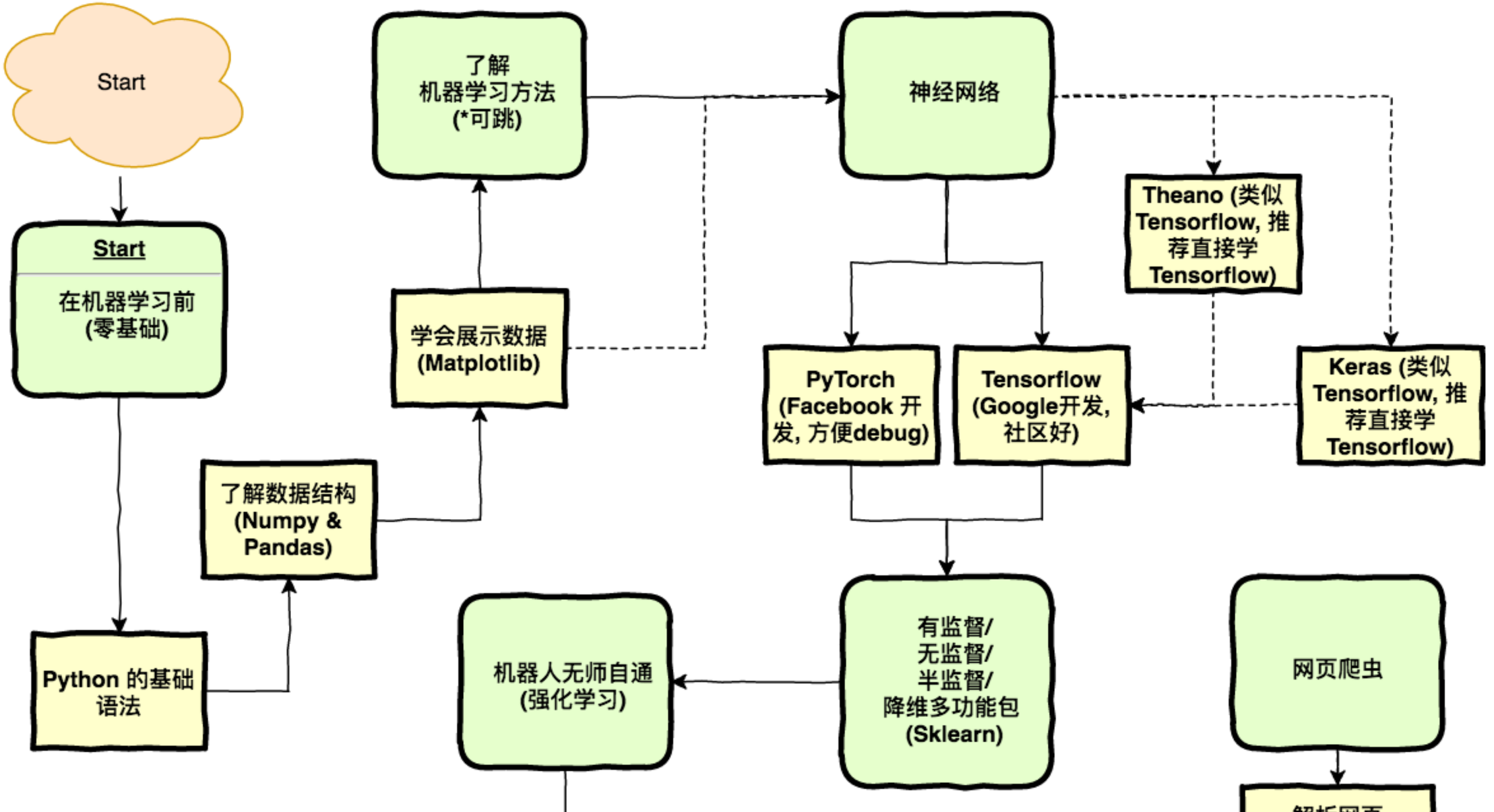
近期更新

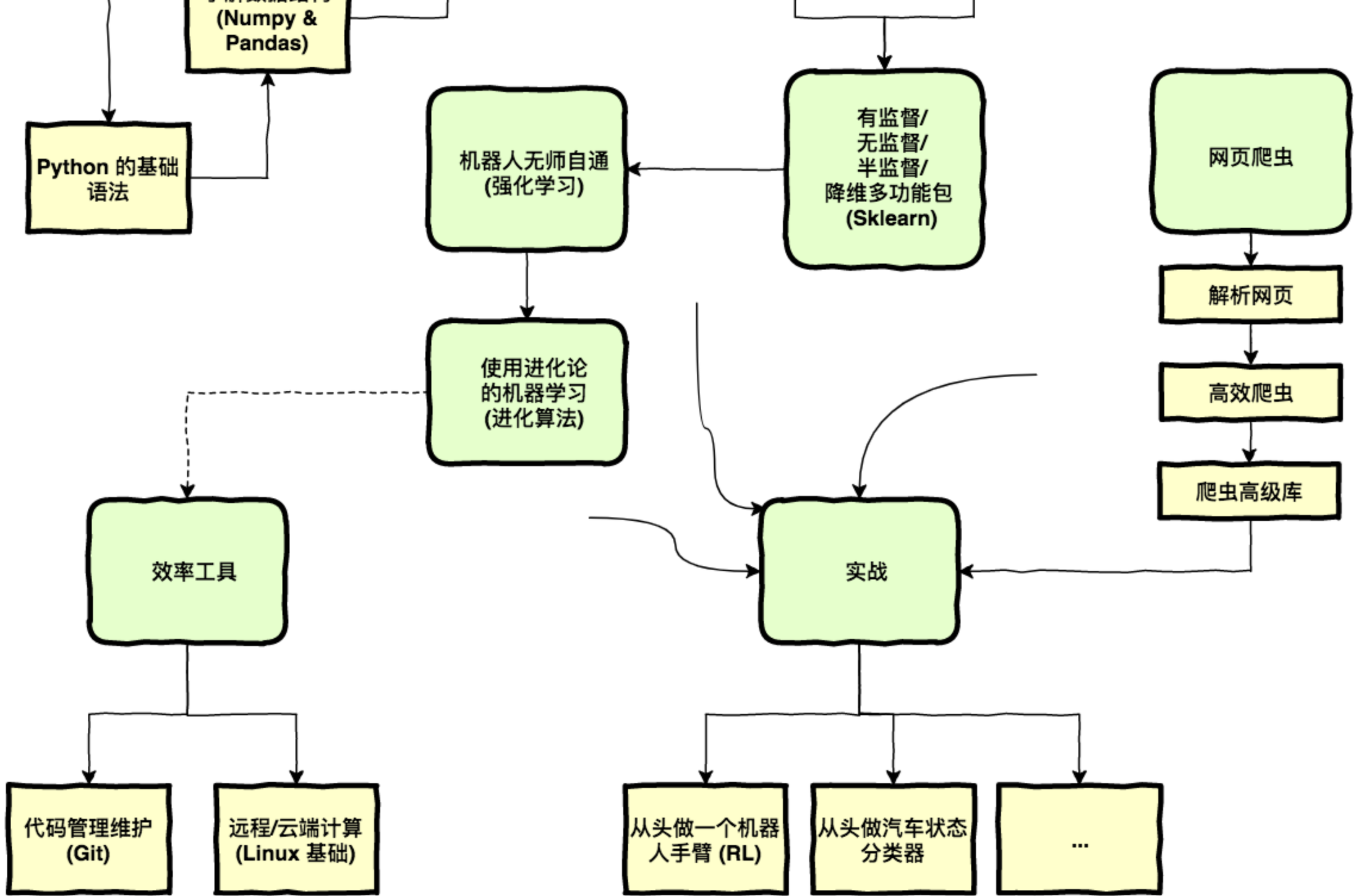
More update...

机器学习

- Pandas & Numpy & matplotlib
- 多线程
- 窗口化
- 正则表达式
- Tensorflow & Pytorch
- 网络爬虫

莫烦Python


Start
Start
在机器学习前
(零基础)
Python 的基础
语法
了解数据结构
(Numpy &
Pandas)
学会展示数据
(Matplotlib)
了解
机器学习方法
(*可跳)
神经网络
Theano (类似
Tensorflow, 推
荐直接学
Tensorflow)
PyTorch
(Facebook 开
发, 方便debug)
Tensorflow
(Google开发,
社区好)
Keras (类似
Tensorflow, 推
荐直接学
Tensorflow)
机器人无师自通
(强化学习)
有监督/
无监督/
半监督/
降维多功能包
(Sklearn)
网页爬虫


(Numpy &
Pandas)
Python 的基础
语法
机器人无师自通
(强化学习)
有监督/
无监督/
半监督/
降维多功能包
(Sklearn)
网页爬虫
解析网页
高效爬虫
爬虫高级库
使用进化论
的机器学习
(进化算法)
效率工具
实战
代码管理维护
(Git)
远程/云端计算
(Linux 基础)
从头做一个机器
人手臂 (RL)
从头做汽车状态
分类器
...

- Week 2 内容回顾

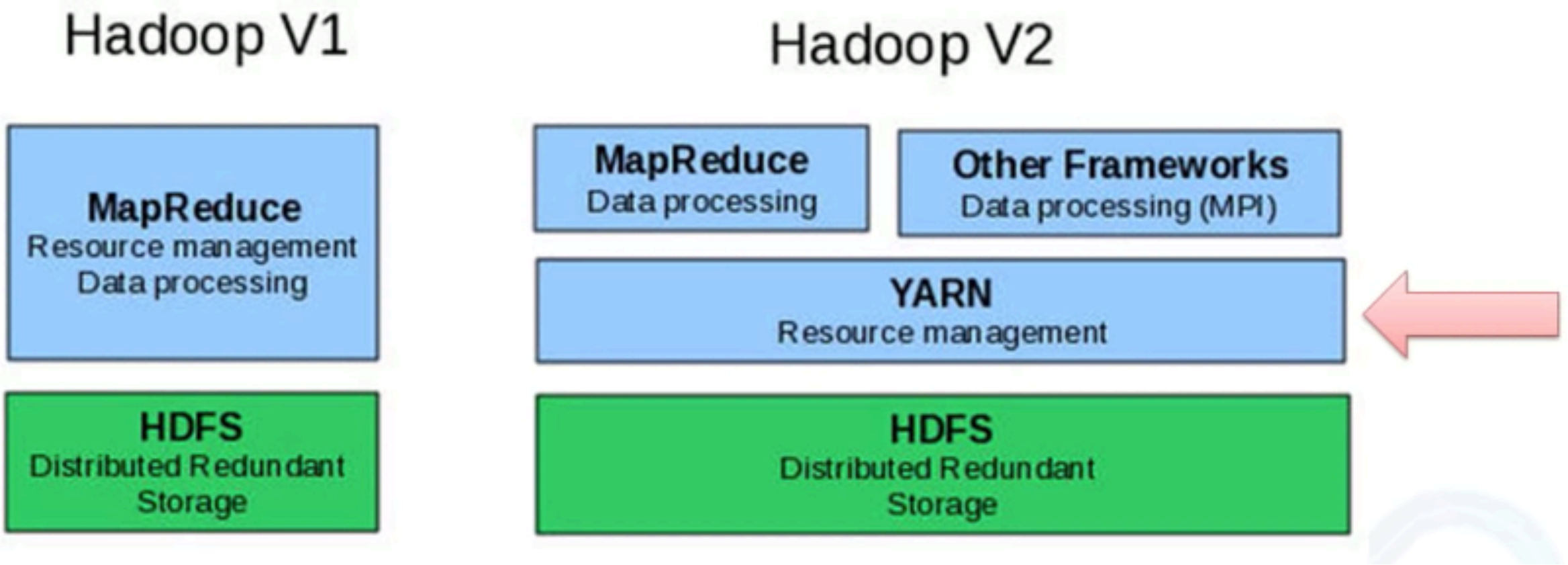
- Hadoop 框架概况
- Hadoop HDFS


Week2
Hadoop 框架概述
发展历史（了解）
Hadoop 框架内容分布（重点）
HDFS
MapReduce
YARN
程序员不需要担心的事情
Where file is located?
How to handle failures & data lost?
How to divide computation?
How to program for scaling?
Hadoop HDFS
Master-Slave Architecture
HDFS Architecture（重点）
NameNode
Secondary NameNode
1NN 2NN 的工作机制
DataNodes
分块 Block （重点）
为什么分块?
小文件问题
块的复制（重点）
为什么默认块的复制数量为3?
机架感知 Rack Awareness
HDFS读写过程

- **Hadoop 框架概况**

Hadoop Ecosystem

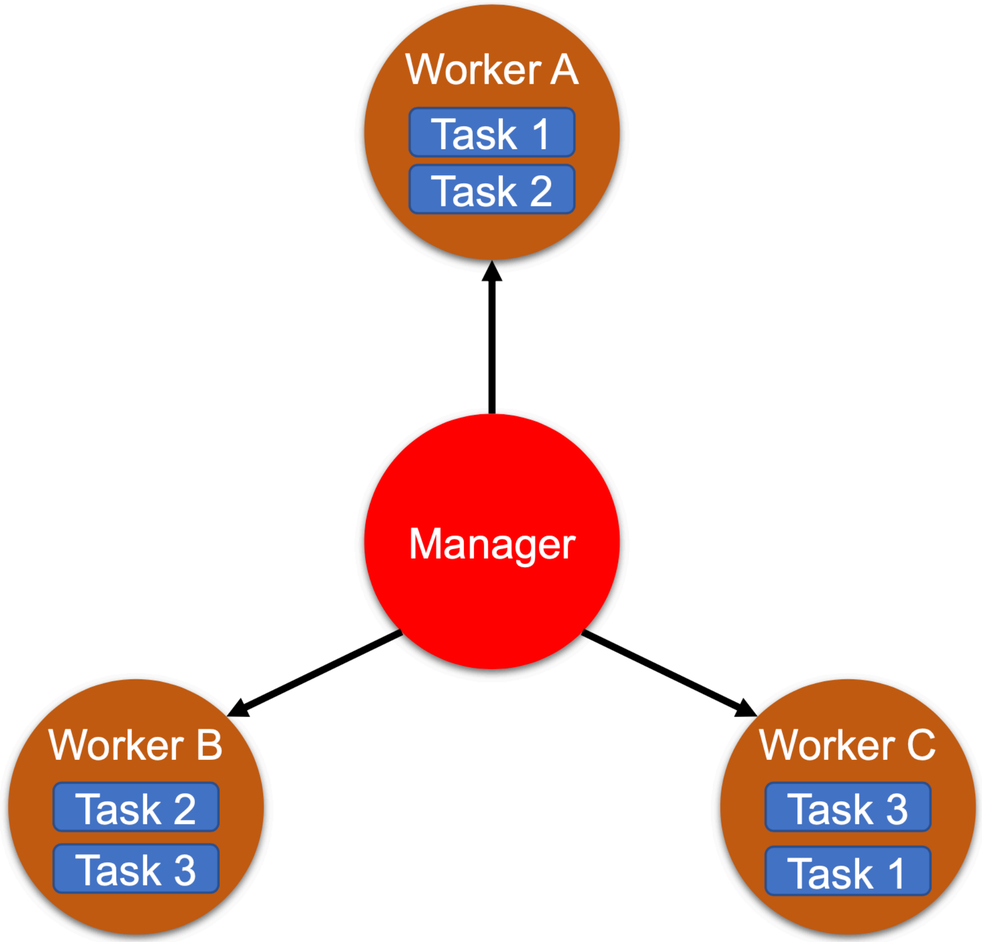
- Core of Hadoop
  - Hadoop distributed file system (HDFS)
  - MapReduce
  - YARN (Yet Another Resource Negotiator) (from Hadoop v2.0)
- Additional software packages
  - Pig
  - Hive
  - Spark
  - HBase
  - …


Hadoop V1
MapReduce
Resource management
Data processing
HDFS
Distributed Redundant
Storage
Hadoop V2
MapReduce
Data processing
Other Frameworks
Data processing (MPI)
YARN
Resource management
HDFS
Distributed Redundant
Storage

- **Hadoop HDFS**

讲完 MapReduce 会来再来理解

The Master-Slave Architecture of Hadoop



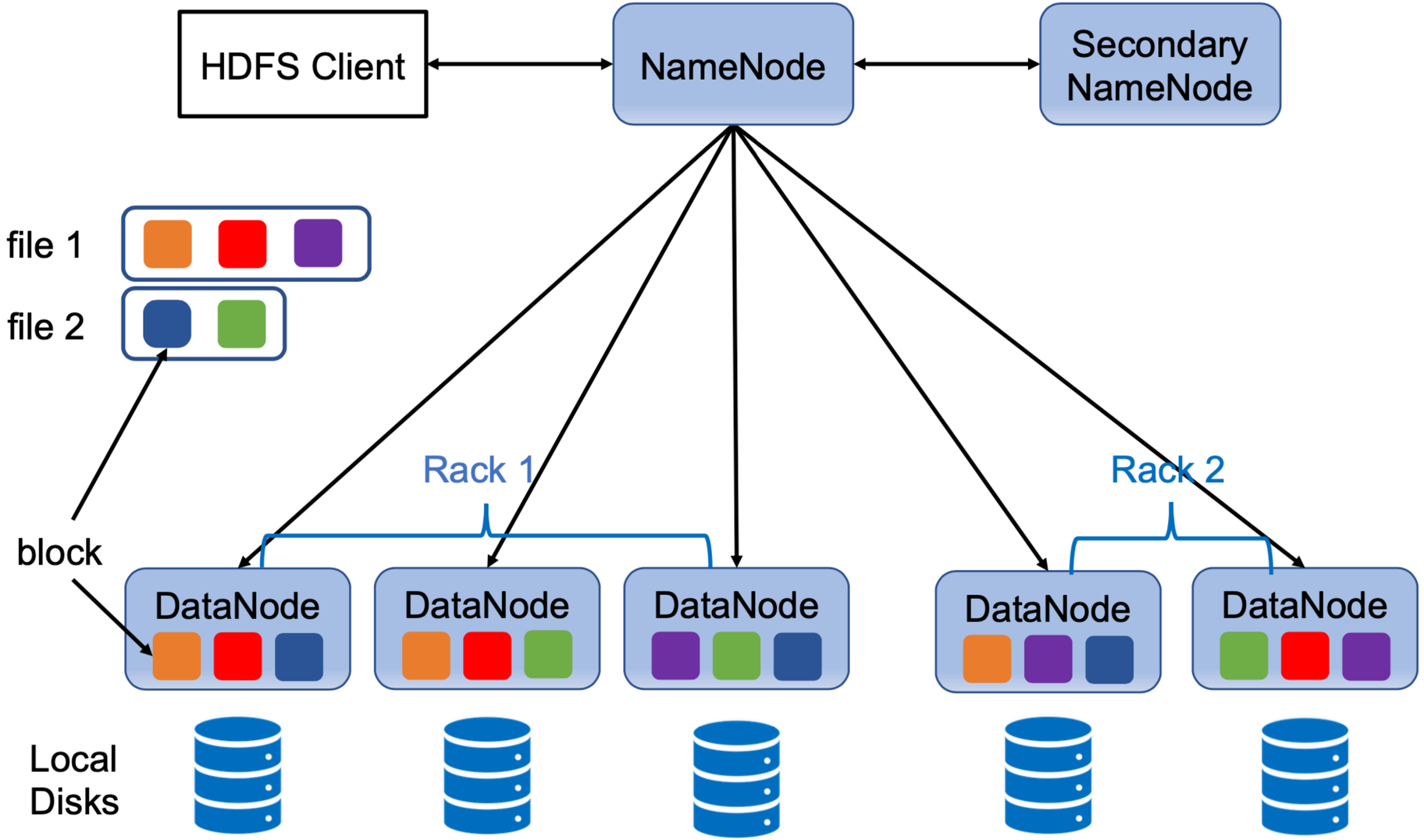
Hadoop Distributed File Systems (HDFS)

- HDFS is a file system that
  - follows master-slave architecture
  - allows us to store data over multiple nodes (machines) ,
  - allows multiple users to access data.
  - just like file systems in your PC

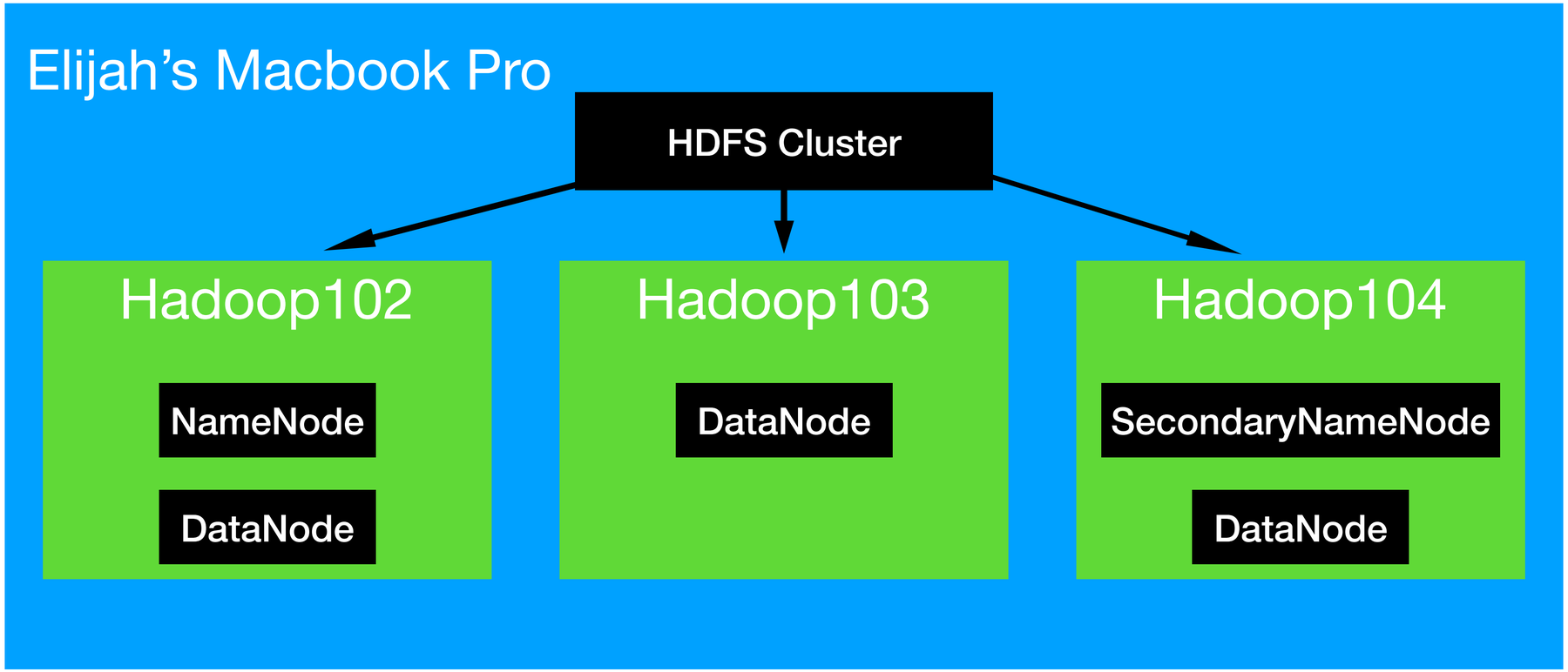
- HDFS supports
  - distributed storage
  - distributed computation
  - horizontal scalability

- **HDFS 各个组成部分**

HDFS Architecture



- **Hadoop HDFS 完全分布式集群 Demo**



- 以下内容为 HDFS 实操
- 不是这个课程的重点，以理解为主

1. 群起集群 hdfs/yarn： start-dfs.sh start-yarn.sh （理解 HDFS 架构）

2. 在集群上创建一个 /20T2/ 目录

   hadoop fs -mkdir -p /20T2/

3. 在本地创建 comp9313.txt 并且上传到 HDFS 的集群 /20T2/ 目录下

 hadoop fs -copyFromLocal comp9313.txt /20T2

 hadoop fs -ls /20T2/

4. 打开 namenode web interface — hadoop102:50070 （理解 Hadoop 文件目录）

5. 看一眼 block 和 replication （理解 block 和 replication）

DataNode

- A commodity hardware stores the data
  - Slave node

- Functions
  - stores actual data
  - perform the read and write requests
  - report the health to NameNode (heartbeat)

NameNode

- NameNode maintains and manages the blocks in the DataNodes (slave nodes).
  - Master node

- Functions:
  - records the metadata of all the files
    - FsImage: file system namespace
    - EditLogs: all the recent modifications
  - records each change to the metadata
  - regularly checks the status of datanodes
  - keeps a record of all the blocks in HDFS
  - if the DataNode failure, handle data recoveray

上 Demo ！
data/tmp/dfs/name/current

上 Demo！

1. cd 到 data/tmp/dfs/name/current （理解 Fsimage 文件和 edit 文件）

2. 使用 ovi 命令把 Fsimage 文件转化成 xml 文件 （理解什么是 MeteData）

hdfs oiv -p XML -i fsimage_0000000000000000025

-o /home/elijah/Desktop/Fsimage.xm l

NameNode vs. DataNode

	NameNode	DataNode
Quantity	One	Multiple
Role	Master	Slave
Stores	Metadata of files	Blocks
Hardware Requirements	High Capacity Memory	High Volume Hard Drive
Failure rate	Lower	Higher
Solution to Failure	Secondary NameNode	Replications

Secondary NameNode

- Take checkpoints of the file system metadata present on NameNode
  - It is not a backup NameNode!

- Functions:
  - Stores a copy of FsImage file and Editlogs
  - Periodically applies Editlogs to FsImage and refreshes the Editlogs.
  - If NameNode is failed, File System metadata can be recovered from the last saved FsImage on the Secondary NameNode.

理解为什么要引入 Secondary？

1.如果所有的 Record 都存内存 —> 一断电就没了

2.如果所有的 Record 都存磁盘 —> 效率太低

3. 于是引入 fsimage edits 两个文件

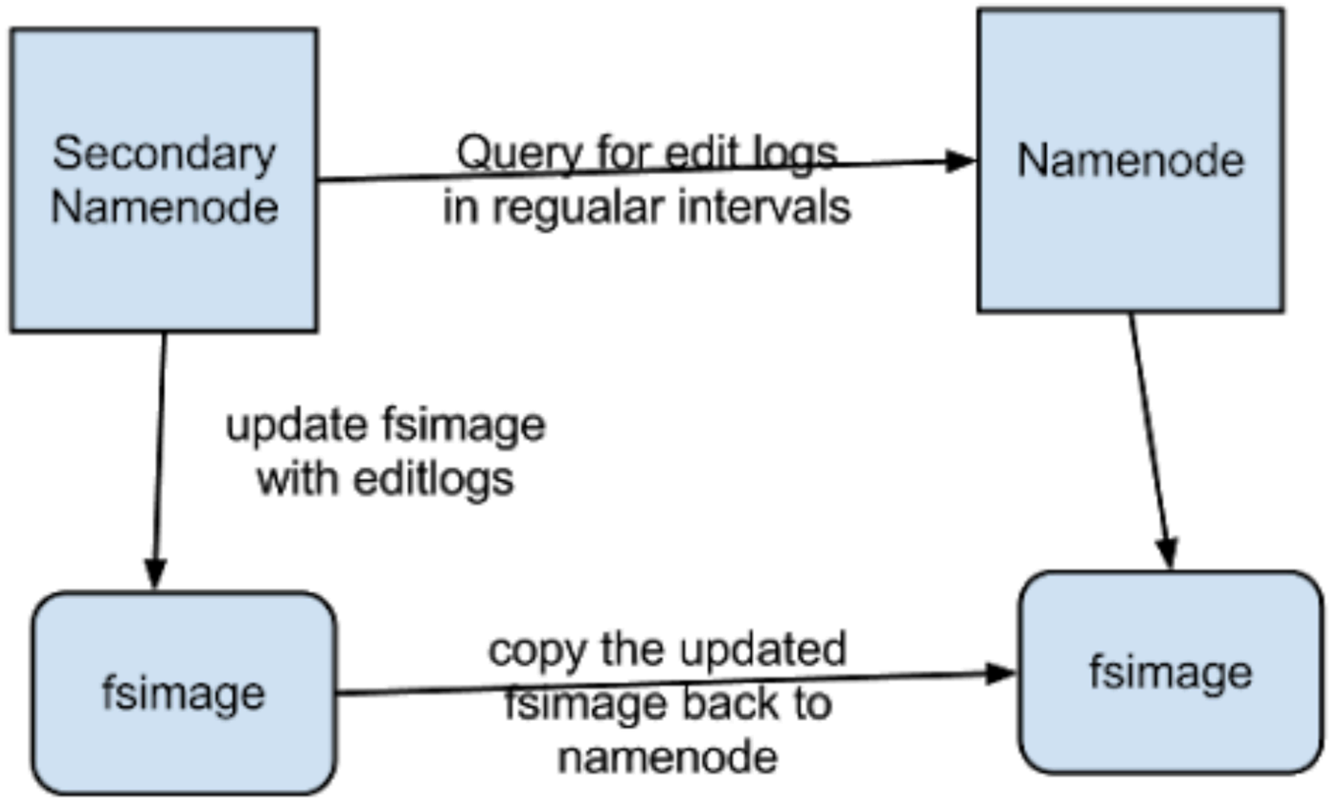
4. 如果每次都是 NameNode 自己启动的时候合并 —> 启动速度太慢

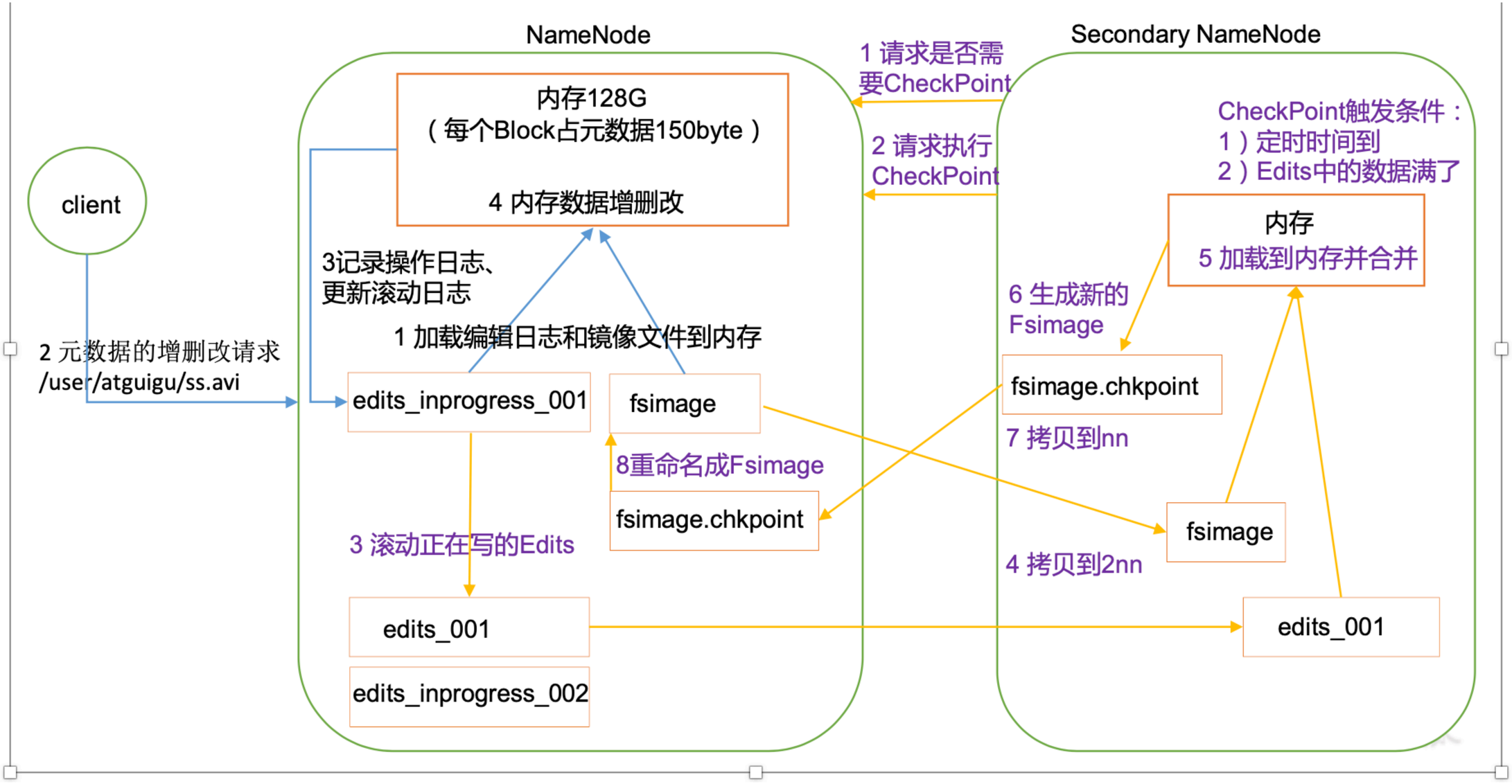
5. 于是引入 Secondary Name Node 提醒他，帮他合并

6. NameNode 挂掉的时候，还可以把 Secondary NameNode 里面的文件拷贝过来进行补救

- **HDFS 1NN 2NN 工作机制**

NameNode vs. Secondary NameNode




NameNode
Secondary NameNode
client
内存128G
（每个Block占元数据150byte）
4 内存数据增删改
1 请求是否需要CheckPoint
2 请求执行CheckPoint
CheckPoint触发条件：
1）定时时间到
2）Edits中的数据满了
内存
5 加载到内存并合并
3记录操作日志、
更新滚动日志
1 加载编辑日志和镜像文件到内存
2 元数据的增删改请求
/user/atguigu/ss.avi
edits_inprogress_001
fsimage
6 生成新的
Fsimage
fsimage.chkpoint
7 拷贝到nn
8重命名成Fsimage
fsimage.chkpoint
fsimage
3 滚动正在写的Edits
4 拷贝到2nn
edits_001
edits_001
edits_inprogress_002

NameNode 和 SecondaryNameNode 的工作机制（重点面试题）

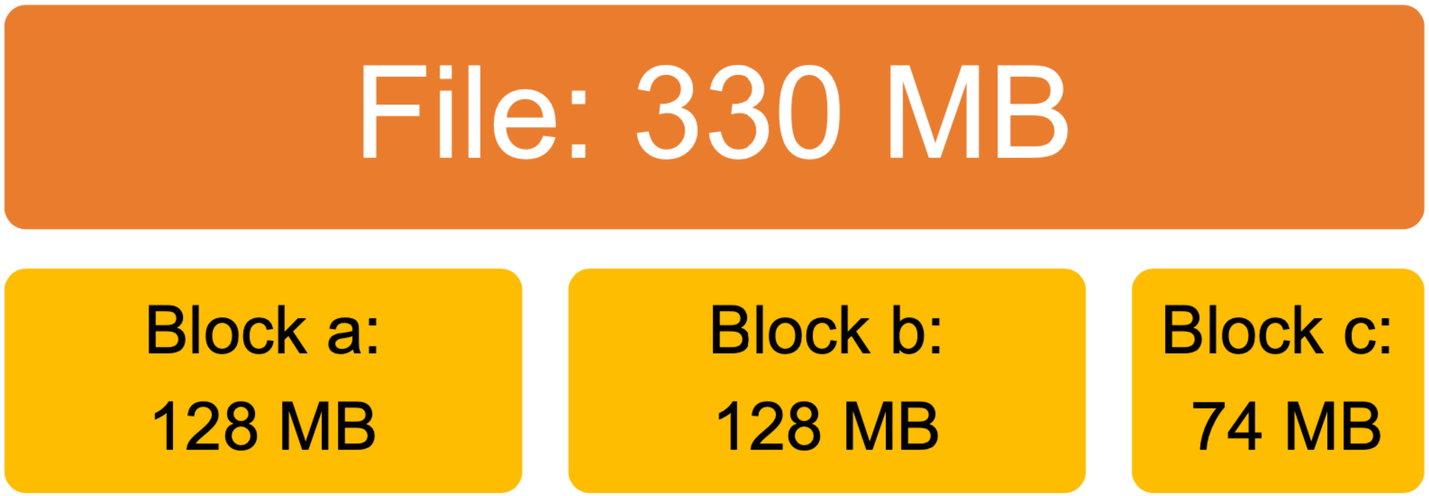
- 首先知道 NameNode 是用来存元数据的，元数据是通过两个不同的文件 edits，fdimage 共同存储的
- 其次 SecondaryNameNode 的任务就是提醒 NN 该合并了，并且帮他合并，以免 edits 这个 log 文件过大， 拖慢 NN 启动速度

> **正式的流程：**
>
> 1. NN 被启动的时候，第一件事就是合并上次生成的 edits，fdimage 文件然后加载到内存，并且生成一个新的空的 edits_inprogress_001 文件
> 2. 当 NameNode 中的数据需要修改更新时，这些更新的操作首先会被记录到这个 edits_inprogress_001 中, 然后才在内存中作修改
> 3. 整个过程，每到一定时间（一小时），或者每到一定的 transcation 次数时(这个次数时 2nn 每过一分钟去询问 NN 你到 100万 次了吗，这个也是个参数可以设置)(取决于配置参数的设置)，SecondaryNameNode 会发出 checkPoint 请求
> 4. NN 收到 checkpoint 请求后，会另起一个 edits_inprogress_002 文件记录后续的更新， 此时把 edits_inprogress_001 (重命名为 edits_001 –001) 和 fdimage 拷贝给 SecondaryNameNode，供其合并
> 5. SecondaryNameNode 合并之后生成新的 fsimage.chkpoint 文件传回给 NN
> 6. NN 把收到的 fsimage.chkpoint 重命名成 fsimage 如此往复

- **HDFS 分块 Block**

Blocks

- Block is a sequence of bytes that stores data
  - Data stores as a set of blocks in HDFS
  - Default block size is 128MB (Hadoop 2.x and 3.x)
  - A file is spitted into multiple blocks



Why Large Block Size?

- HDFS stores huge datasets
- If block size is small (e.g., 4KB in Linux), then the number of blocks is large:
  - too much metadata for NameNode
  - too many seeks affect the read speed
  - harm the performance of MapReduce too

- We don’t recommend using HDFS for small files due to similar reasons.
  - Even a 4KB file will occupy a whole block.

为什么要分块?

1.占空间（每个在集群上的文件无论大小,

在 NameNode 上都占 150B）

2.小文件太多增加寻址时间

3. 影响 MapReduce 性能

为什么要复制 Data？

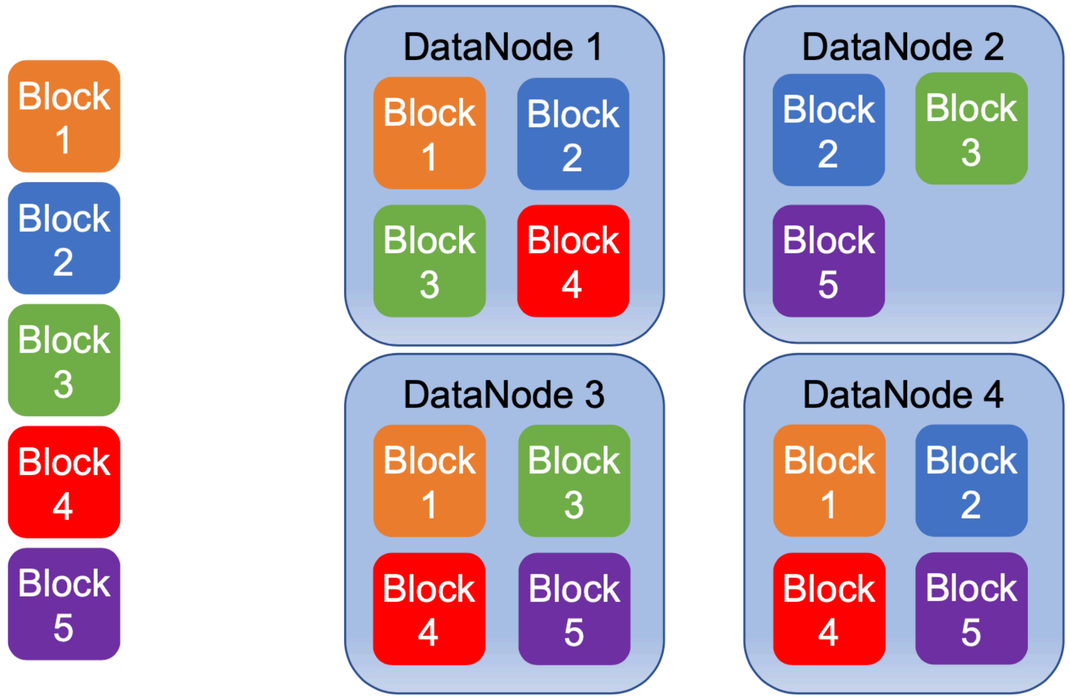
If DataNode Failed…

- Commodity hardware fails
  - If NameNode hasn't heard from a DataNode for 10mins, The DataNode is considered dead…
- HDFS guarantees data reliability by generating multiple replications of data
  - each block has 3 replications by default
  - replications will be stored on different DataNodes
  - if blocks were lost due to the failure of a DataNode, they can be recovered from other replications
  - the total consumed space is 3 times the data size
- It also helps to maintain data integrity

注意区分分块和复制数的概念，大文件分好块了之后，每个块都会被复制

Replication Management

- Each block is replicated 3 times and stored on different DataNodes



上 Demo！

1.Web interface 看一下 分块 和 replica 的情况

2.去 etc/hdfs-site.xml 看一下默认的参数 3

What about Simultaneous Failure?

等到有题出现的时候再将

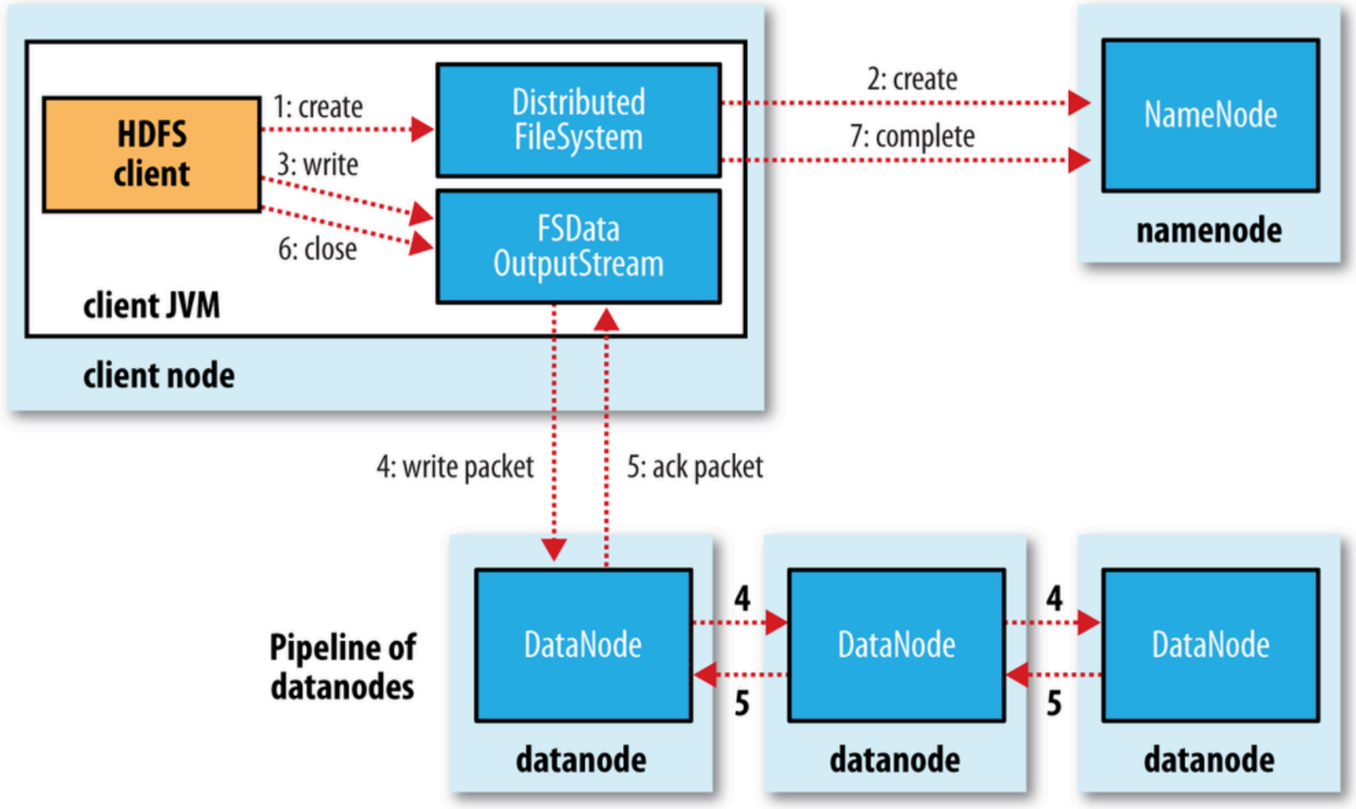
- In general
  - L1(k,N) = k*B-2*L2(k,N)-3*L3(k,N)
  - L2(k,N) = 2*L1(k-1,N)/(N-k+1)+L2(k-1,N)-L3(k,N)
  - L3(k,N) = L2(k-1,N)/(N-k+1)+L3(k-1,N)
- Let N = 4000, B = 750, we have

Failed Nodes	1st replicas lost	2nd replicas lost	3rd replicas lost
50	36,629	433	2
100	72,002	1,479	13
150	107,374	2,504	39
200	143,963	2,905	76

HDFS 写流程

Write in HDFS

- Create file – Write file – Close file



Write in HDFS

- There is only single writer allowed at any time
- The blocks are writing simultaneously
- For one block, the replications are replicating sequentially
- The choose of DataNodes is random, based on replication management policy, rack awareness, …

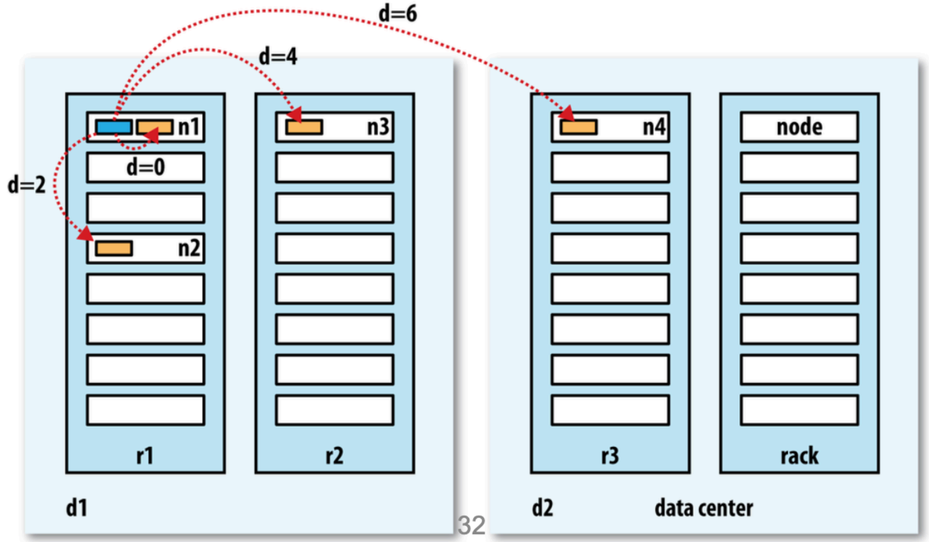
客户端向 NameNode 请求上传文件的时候，NameNode 挑选那些 DataNode

↓

机架感知 rack awareness

Why Rack Awareness?

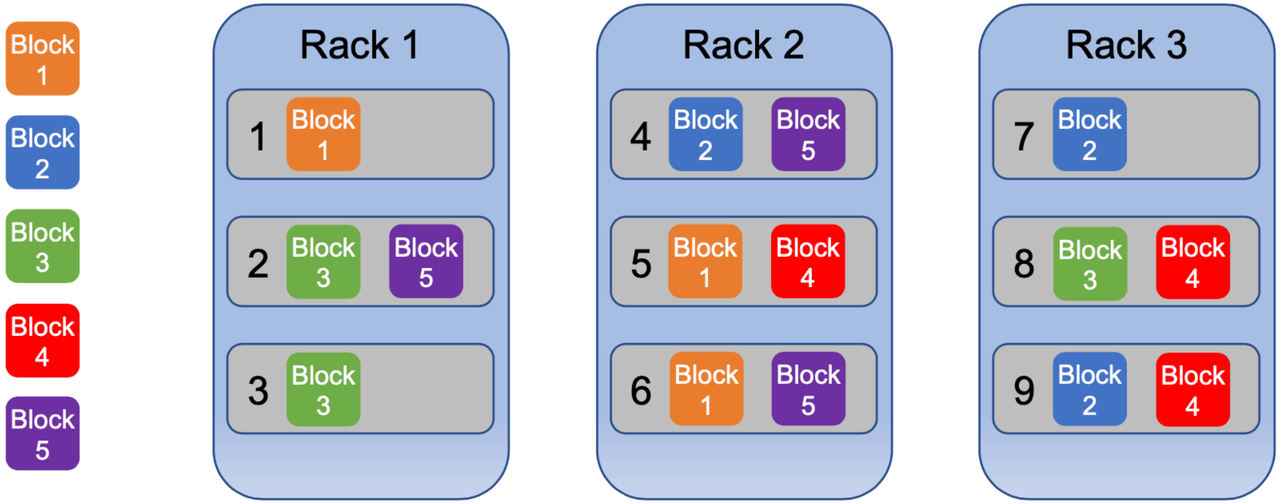
- Reduce latency
  - Write: to 2 racks instead of 3 per block
  - Read: blocks from multiple racks
- Fault tolerance
  - Never put your eggs in the same basket



影响因素：网络 I/O传输

Rack Awareness Algorithm

- If the replication factor is 3:
  - 1st replica will be stored on the local DataNode
  - 2nd on a different rack from the first.
  - 3rd on the same rack as 2nd, but on a different node.



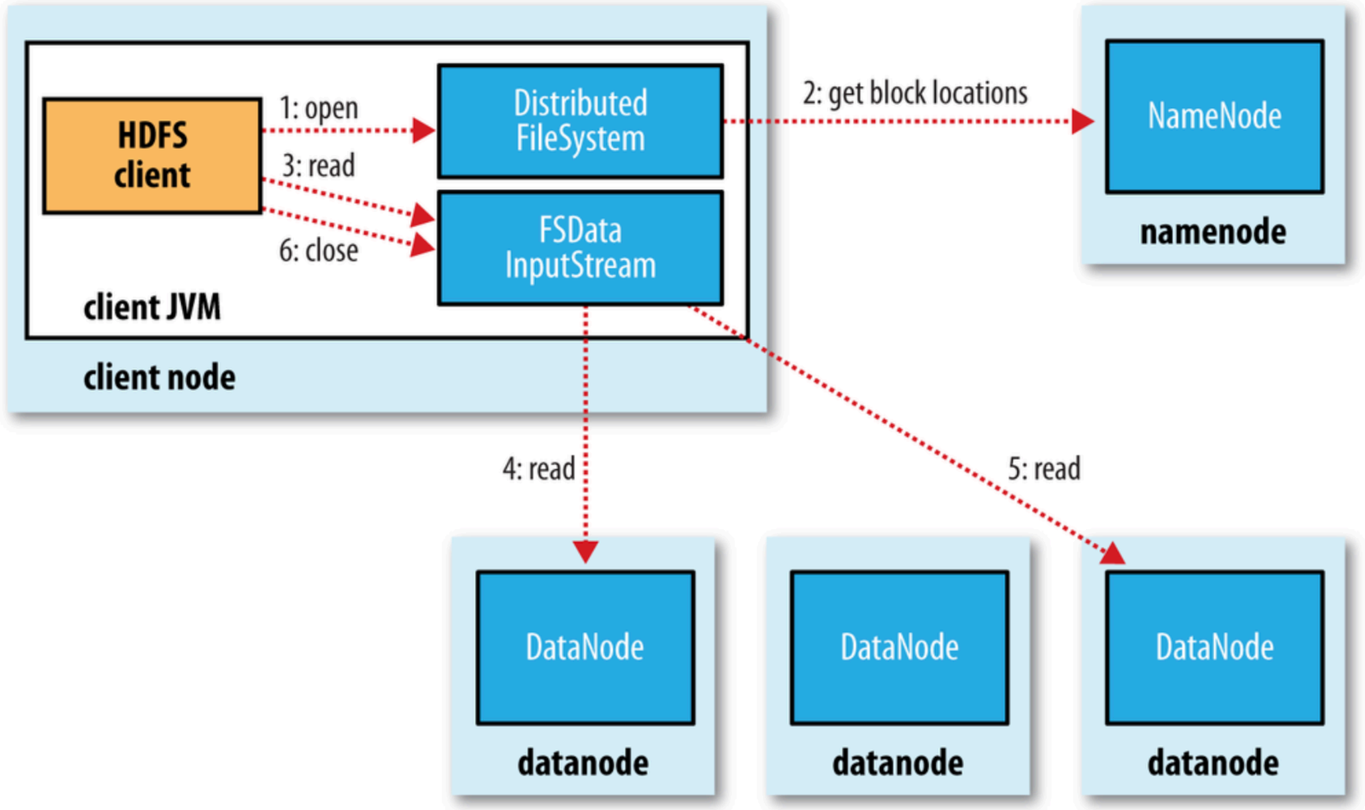
Hadoop 3.2.1 官方解释：

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Replication

For the common case, when the replication factor is three, HDFS’s placement policy is to put one replica on the local machine if the writer is on a datanode, otherwise on a random datanode in the same rack as that of the writer, another replica on a node in a different (remote) rack, and the last on a different node in the same remote rack.

HDFS 读流程

Read in HDFS



Read in HDFS

- Multiple readers are allowed to read at the same time
- The blocks are reading simultaneously
- Always choose the closest DataNodes to the client (based on the network topology)
- Handling errors and corrupted blocks
  - avoid visiting the dataNode again
  - report to NameNode

网络拓扑

NetWork Topology

网络拓扑计算距离

> **如何计算两个节点之间的距离?**
> 节点之间的距离 = 两个节点到达最近的公共路由/交换机的距离之和

Distance(/d1/r1/n0, /d1/r1/n0)=0（同一节点上的进程）

Distance(/d1/r1/n1, /d1/r1/n2)=2（同一机架上的不同节点）

Distance(/d1/r2/n0, /d1/r3/n2)=4（同一数据中心不同机架上的节点）

Distance(/d1/r2/n1, /d2/r4/n1)=6（不同数据中心的节点）