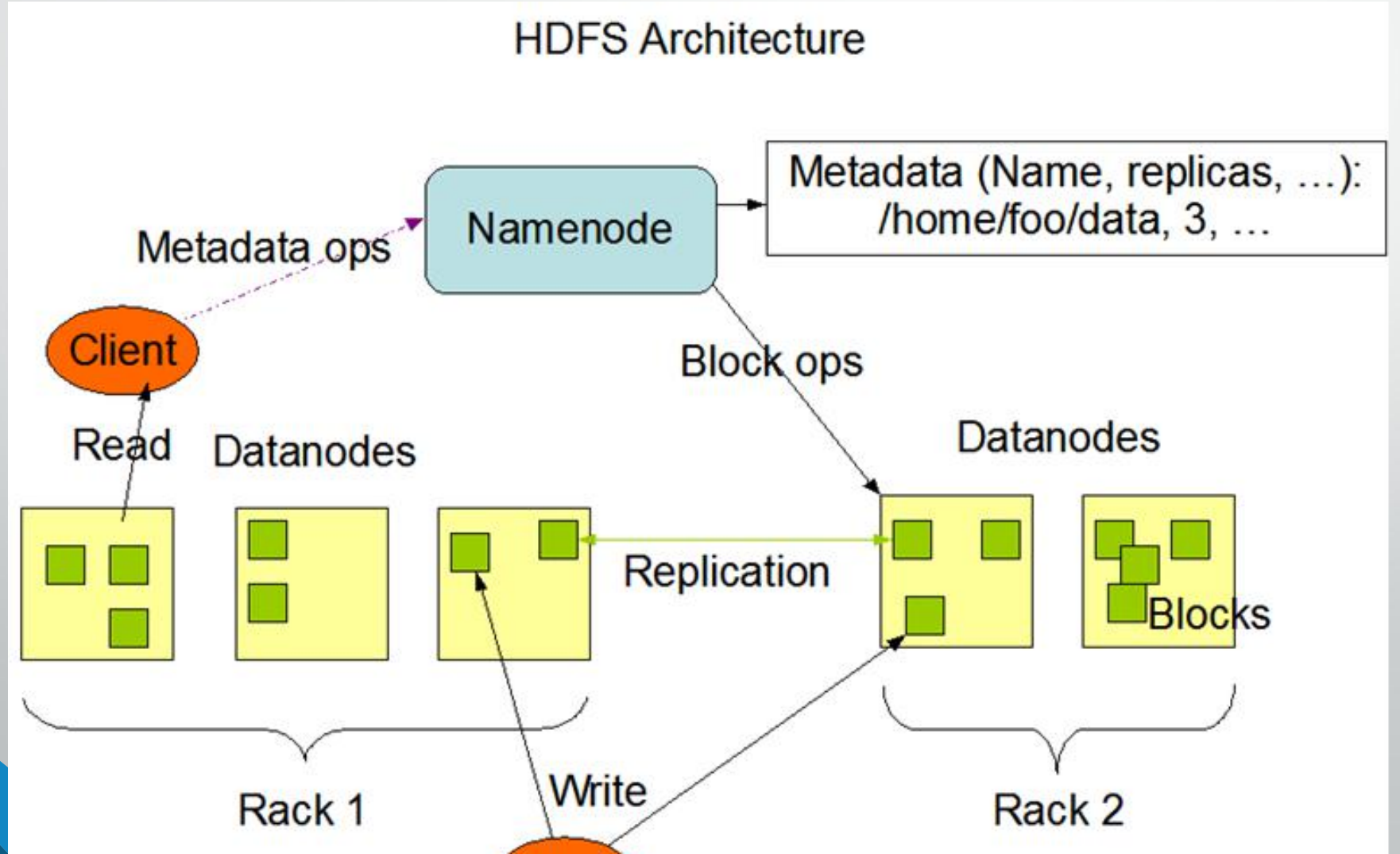
深入Hadoop 2.x（一）

谭唐华

课程大纲

- HDFS存储架构剖析
- NameNode启动流程
- Yarn平台架构剖析
- HDFS JAVA API讲解
- HDFS 文件读写流程

HDFS分布式文件系统



NameNode

- Namenode 是一个中心服务器，单一节点（简化系统的设计和实现），负责管理文件系统的名字空间(namespace)以及客户端对文件的访问。
- 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，跟文件内容相关的数据流不经过NameNode，只会询问它跟那个DataNode联系，否则NameNode会成为系统的瓶颈。
- 副本存放在哪些DataNode上由 NameNode来控制，根据全局情况做出块放置决定，读取文件时NameNode尽量让用户先读取最近的副本，降低带块消耗和读取时延
- Namenode 全权管理数据块的复制，它周期性地从集群中的每个Datanode接收心跳信号和块状态报告(Blockreport)。接收到心跳信号意味着该Datanode节点工作正常。块状态报告包含了一个该Datanode上所有数据块的列表。

DataNode

- 一个数据块在DataNode以文件存储在磁盘上，包括两个文件，一个是数据本身，一个是元数据包括数据块的长度，块数据的校验和，以及时间戳
- DataNode启动后向NameNode注册，通过后，周期性（1小时）的向NameNode上报所有的块信息。
- 心跳是每3秒一次，心跳返回结果带有NameNode给该DataNode的命令如复制块数据到另一台机器，或删除某个数据块。如果超过10分钟没有收到某个DataNode 的心跳，则认为该节点不可用。
- 集群运行中可以安全加入和退出一些机器

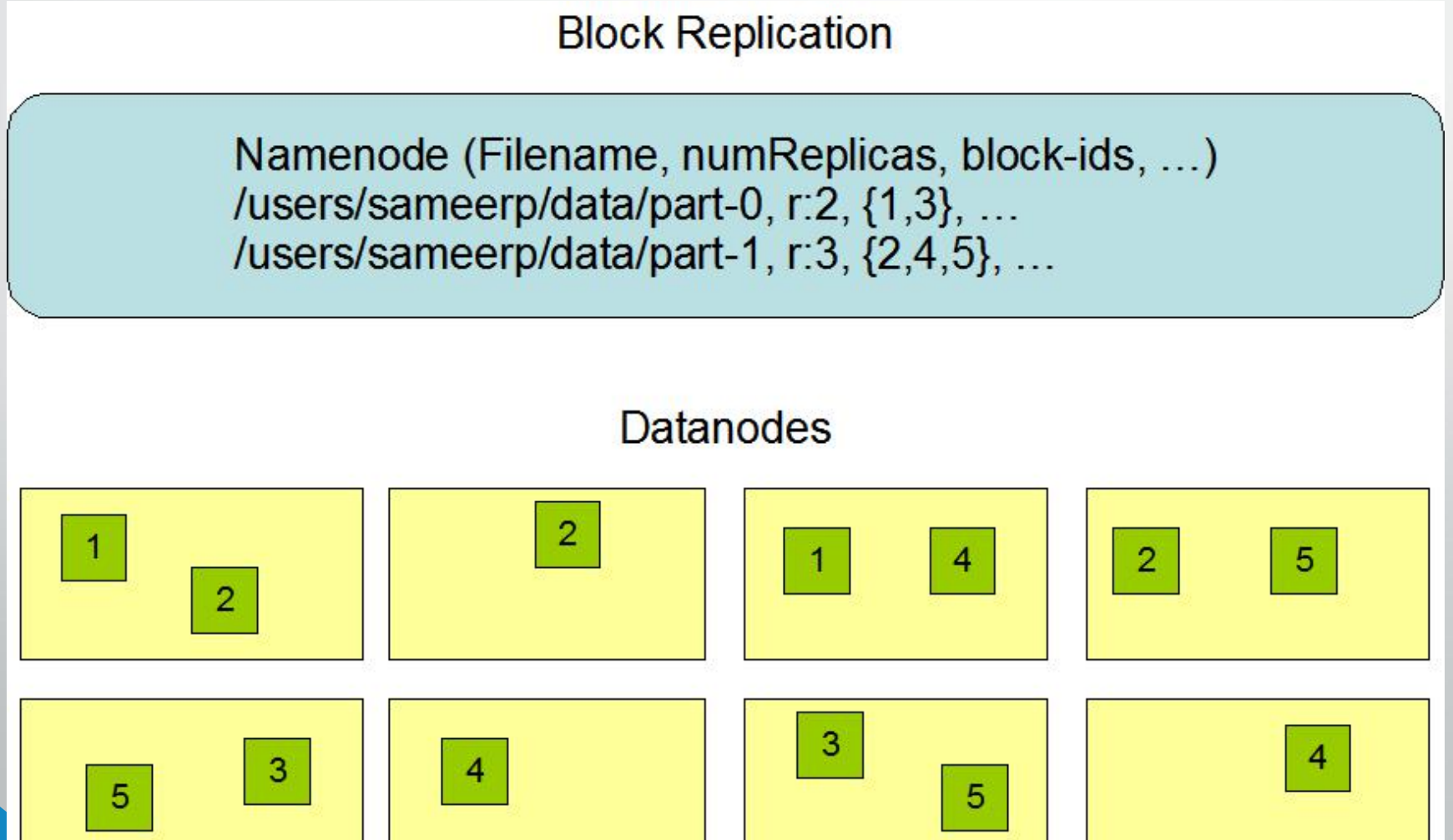
文件

- 文件切分成块（默认大小128M），以块为单位，每个块有多个副本存储在不同的机器上，副本数可在文件生成时指定（默认3）
- NameNode 是主节点，存储文件的元数据如文件名，文件目录结构，文件属性（生成时间,副本数,文件权限），以及每个文件的块列表以及块所在的DataNode等等
- DataNode 在本地文件系统存储文件块数据，以及块数据的校验和。
- 可以创建、删除、移动或重命名文件，当文件创建、写入和关闭之后不能修改文件内容。

数据损坏处理

- 当DataNode读取block的时候，它会计算checksum
- 如果计算后的checksum，与block创建时值不一样，说明该block已经损坏。
- Client读取其它DN上的block。
- NameNode标记该块已经损坏，然后复制block达到预期设置的文件备份数
- DataNode 在其文件创建后三周验证其checksum

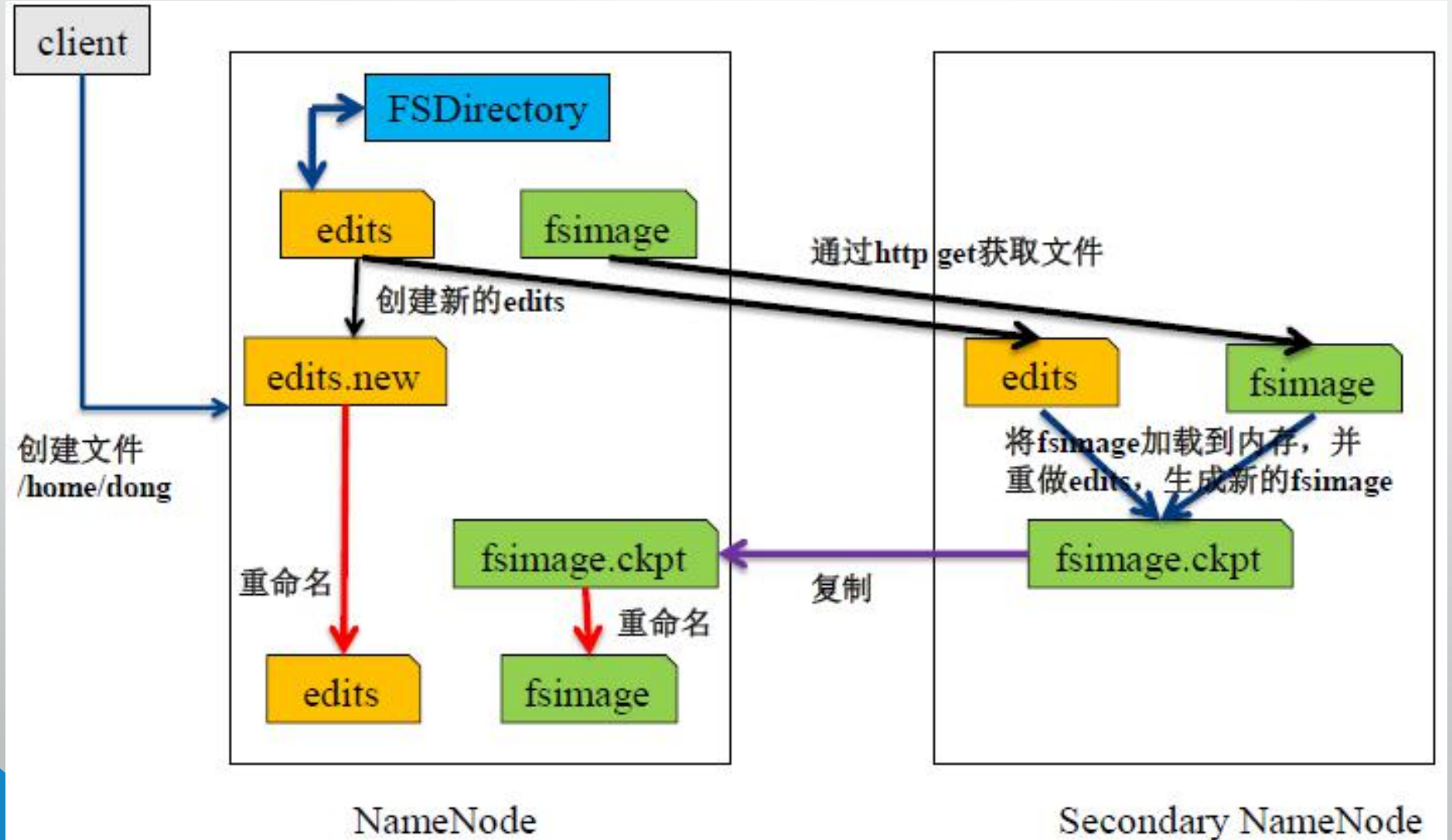
数据复制


Block Replication
Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …
Datanodes
1
2
2
1
4
2
5
5
3
4
3
5
4

课程大纲

- HDFS存储架构剖析
- NameNode启动流程
- Yarn平台架构剖析
- HDFS JAVA API讲解
- HDFS 文件读写流程

NameNode启动过程



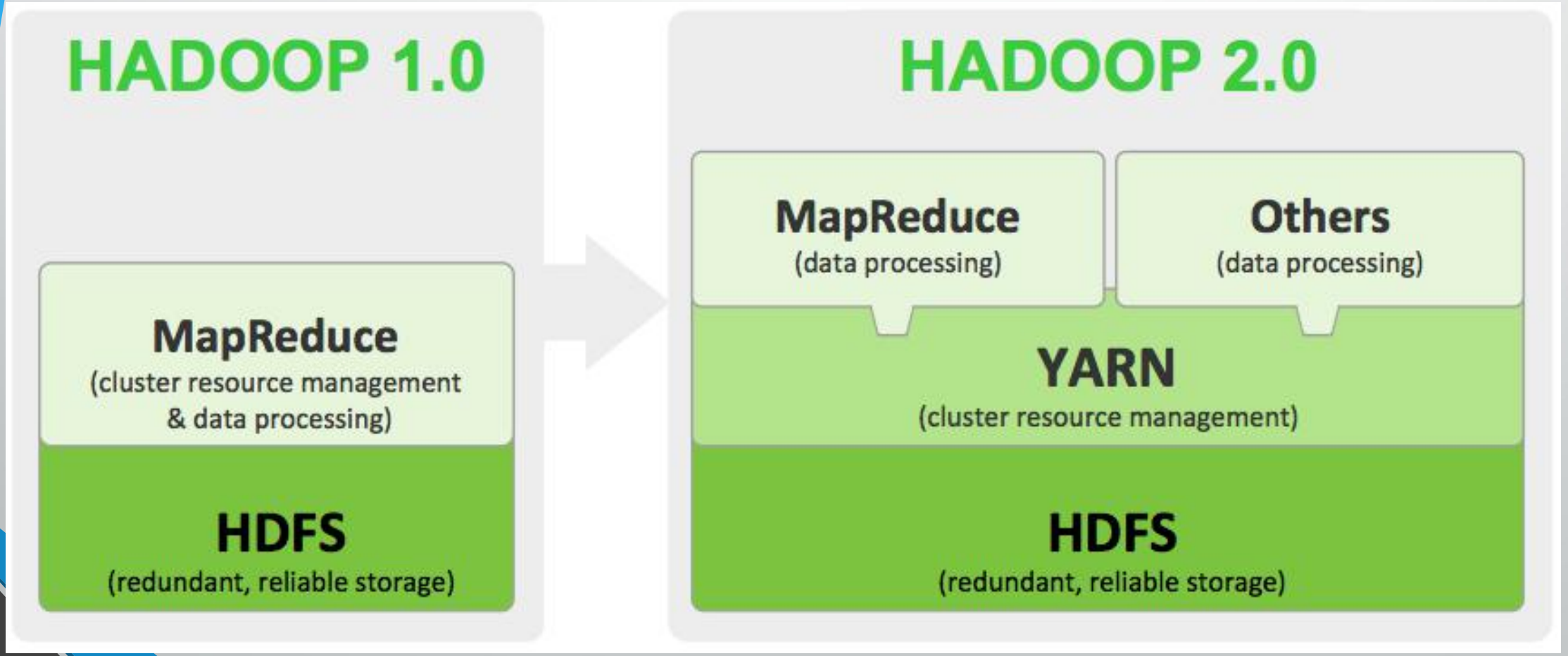
HDFS Shell命令

```
[hadoop@hadoop-yarn hadoop-2.2.0]$ bin/hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setrep [-R] [-w] <rep> <path> ...]
        [-stat [format] <path> ...]
        [-tail [-f] <file>]
        [-test -[defsz] <path>]
        [-text [-ignoreCrc] <src> ...]
        [-touchz <path> ...]
        [-usage [cmd ...]]
```

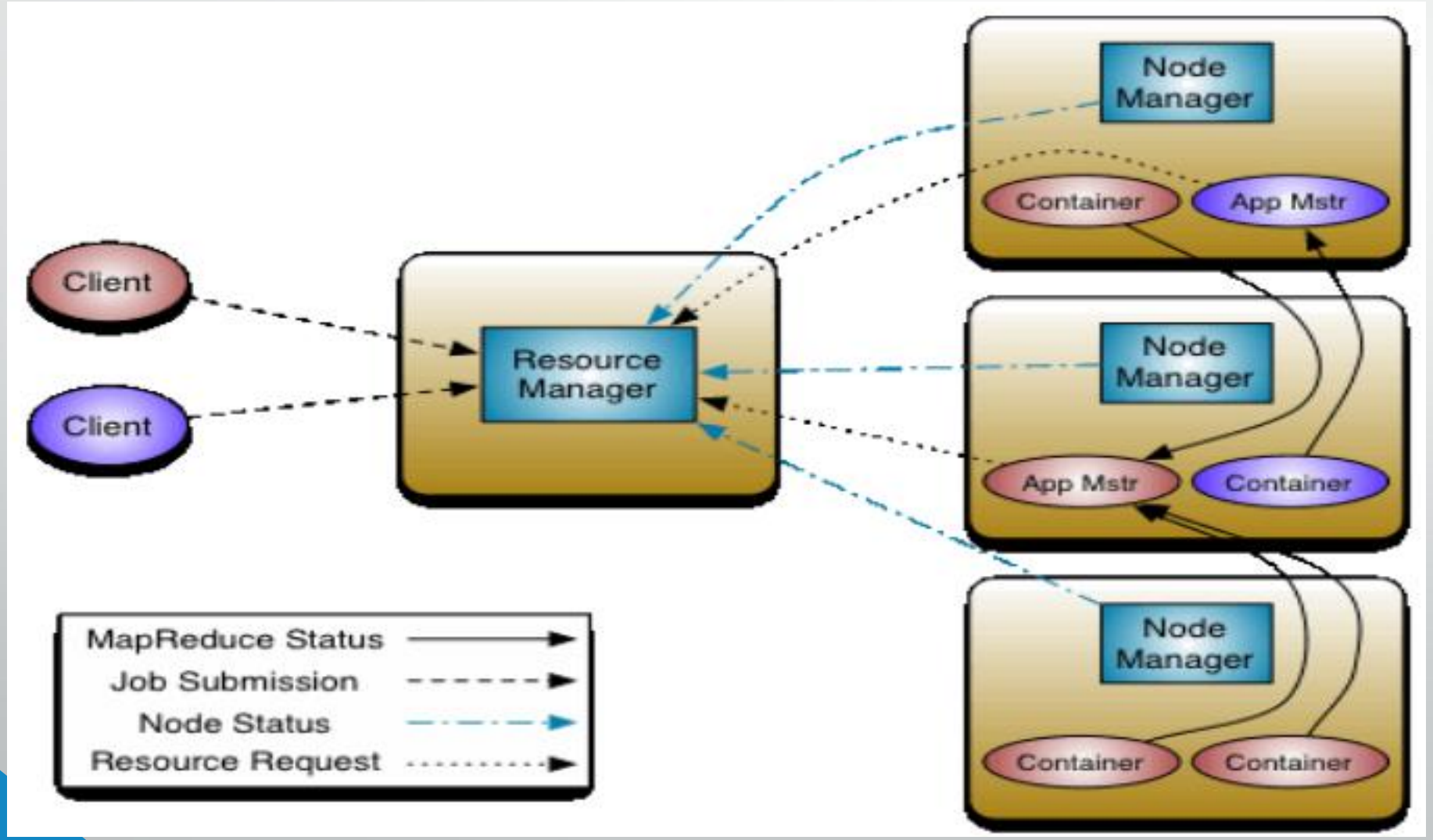
课程大纲

- HDFS存储架构剖析
- NameNode启动流程
- YARN平台架构剖析
- HDFS JAVA API讲解
- HDFS 文件读写流程

YARN版本对比


HADOOP 1.0
MapReduce
(cluster resource management
& data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)

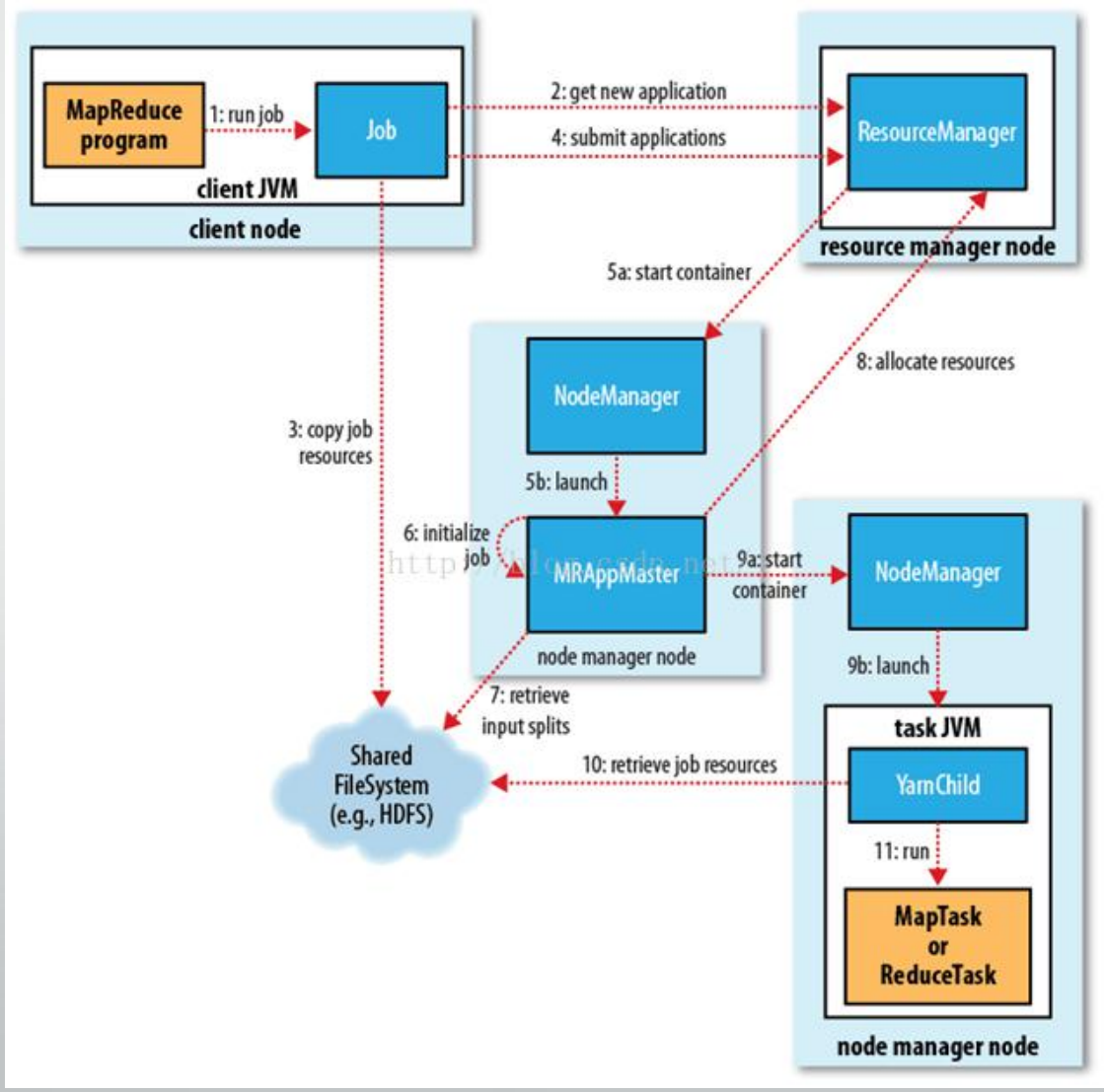
YARN架构


Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

YARN服务功能

- ResourceManager
  - 处理客户端请求
  - 启动/监控ApplicationMaster
  - 监控NodeManager
  - 资源分配与调度
- NodeManager
  - 单个节点上的资源管理
  - 处理来自ResourceManager的命令
  - 处理来自ApplicationMaster的命令
- ApplicationMaster
  - 数据切分
  - 为应用程序申请资源，并分配给内部任务
  - 任务监控与容错
- Container
  - 对任务运行环境的抽象，封装了CPU、内存等多维资源以及环境变量、启动命令等任务运行相关的信息

YARN工作流程


MapReduce program
1: run job
Job
client JVM
client node
2: get new application
4: submit applications
ResourceManager
resource manager node
5a: start container
NodeManager
5b: launch
8: allocate resources
3: copy job resources
6: initialize job
MRAppMaster
node manager node
9a: start container
NodeManager
9b: launch
task JVM
YarnChild
11: run
MapTask or ReduceTask
node manager node
7: retrieve input splits
Shared FileSystem (e.g., HDFS)
10: retrieve job resources

MapReduce计算框架

- 将计算过程分为两个阶段，Map和Reduce
  - ✓Map 阶段并行处理输入数据
  - ✓Reduce阶段对Map结果进行汇总
- Shuffle连接Map和Reduce两个阶段
  - ✓Map Task将数据写到本地磁盘
  - ✓Reduce Task从每个Map Task上读取一份数据
- 仅适合离线批处理
  - ✓具有很好的容错性和扩展性
  - ✓适合简单的批处理任务
- 缺点明显
  - ✓启动开销大、过多使用磁盘导致效率低下等

YARN资源管理

- YARN允许用户配置每个节点上可用的物理内存资源，注意，这里是“可用的”，因为一个节点上的内存会被若干个服务共享，比如一部分给YARN，一部分给HDFS，一部分给HBase等，YARN配置的只是自己可以使用的，配置参数如下：

（1）yarn.nodemanager.resource.memory-mb

表示该节点上YARN可使用的物理内存总量，默认是8192（MB），注意，如果你的节点内存资源不够8GB，则需要调减小这个值，而YARN不会智能的探测节点的物理内存总量。

（2）yarn.nodemanager.vmem-pmem-ratio

任务每使用1MB物理内存，最多可使用虚拟内存量，默认是2.1。

（3）yarn.nodemanager.pmem-check-enabled

是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，则直接将其杀掉，默认是true。

（4）yarn.nodemanager.vmem-check-enabled

是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，则直接将其杀掉，默认是true。

（5）yarn.scheduler.minimum-allocation-mb

单个任务可申请的最少物理内存量，默认是1024（MB），如果一个任务申请的物理内存量少于该值，则该对应的值改为这个数。

（6）yarn.scheduler.maximum-allocation-mb

单个任务可申请的最多物理内存量，默认是8192（MB）。

YARN资源管理

- 目前的CPU被划分成虚拟CPU（CPU virtual Core），这里的虚拟CPU是YARN自己引入的概念，初衷是，考虑到不同节点的CPU性能可能不同，每个CPU具有的计算能力也是不一样的，比如某个物理CPU的计算能力可能是另外一个物理CPU的2倍，这时候，你可以通过为第一个物理CPU多配置几个虚拟CPU弥补这种差异。用户提交作业时，可以指定每个任务需要的虚拟CPU个数。在YARN中，CPU相关配置参数如下：

（1）yarn.nodemanager.resource.cpu-vcores

表示该节点上YARN可使用的虚拟CPU个数，默认是8，注意，目前推荐将该值设值为与物理CPU核数数目相同。如果你的节点CPU核数不够8个，则需要调减小这个值，而YARN不会智能的探测节点的物理CPU总数。

（2） yarn.scheduler.minimum-allocation-vcores

单个任务可申请的最小虚拟CPU个数，默认是1，如果一个任务申请的CPU个数少于该数，则该对应的值改为这个数。

（3）yarn.scheduler.maximum-allocation-vcores

单个任务可申请的最多虚拟CPU个数，默认是32。

课程大纲

- HDFS存储架构剖析
- NameNode启动流程
- Yarn平台架构剖析
- HDFS JAVA API讲解
- HDFS 文件读写流程

开发环境准备

- 安装Maven，用于管理项目依赖包（apache-maven-3.0.5-bin.tar）
- 安装Eclipse（eclipse-jee-kepler-SR1-linux-gtk-x86_64.tar）
- 配置Eclipse与Maven插件（JEE，自带配置好）
- 设置Linux下Eclipse快捷键
- 设置Eclipse字体大小（Java文件和XML文件）
- 创建Maven Project，配置POM文件

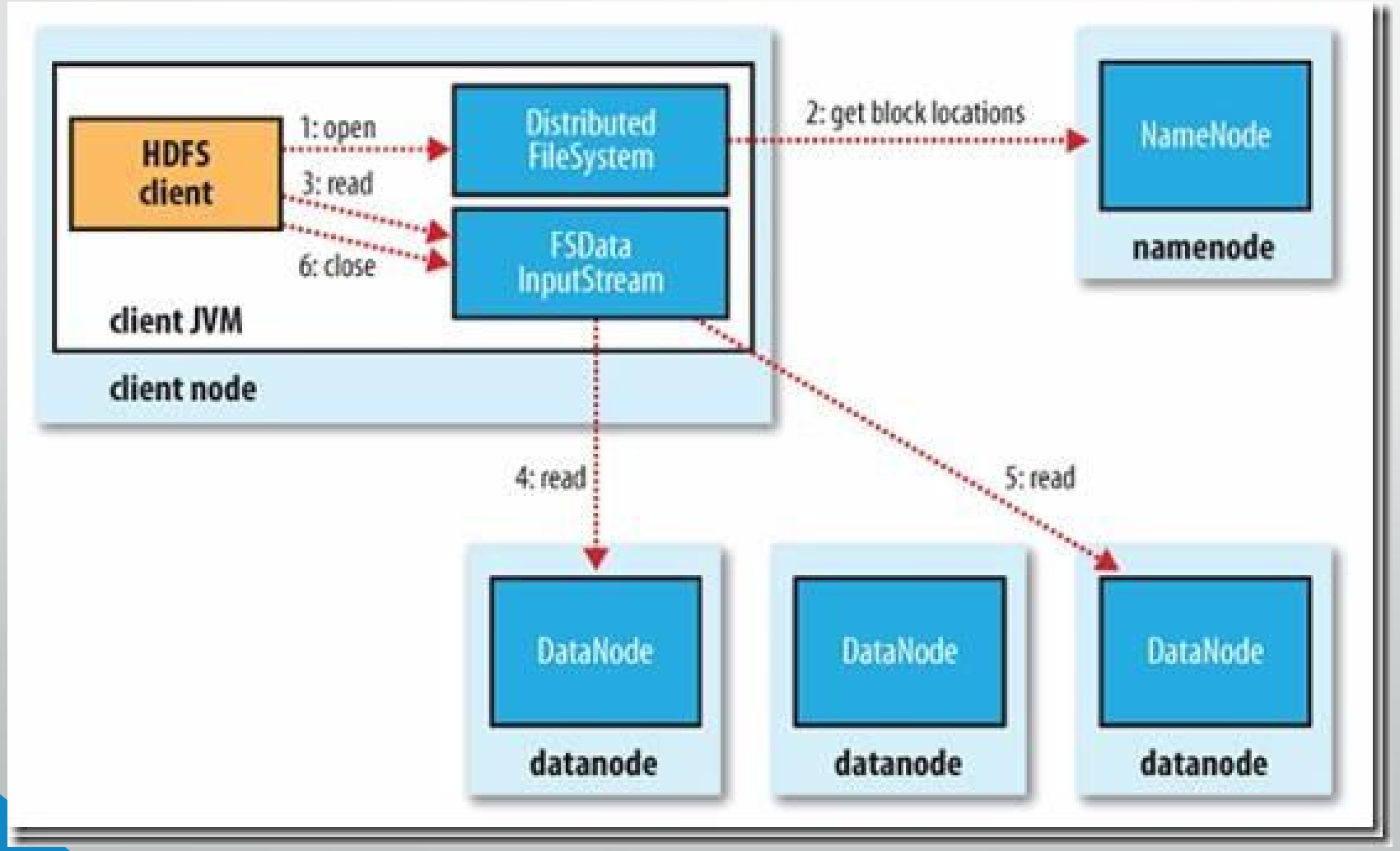
对HDFS API 操作

- 文件操作
  - 上传本地文件到HDFS
  - 读取文件
  - 在hadoop fs中新建文件，并写入
  - 重命名文件
  - 删除hadoop fs上的文件
- 目录操作
  - 读取某个目录下的所有文件
  - 在hadoop fs上创建目录
  - 删除目录

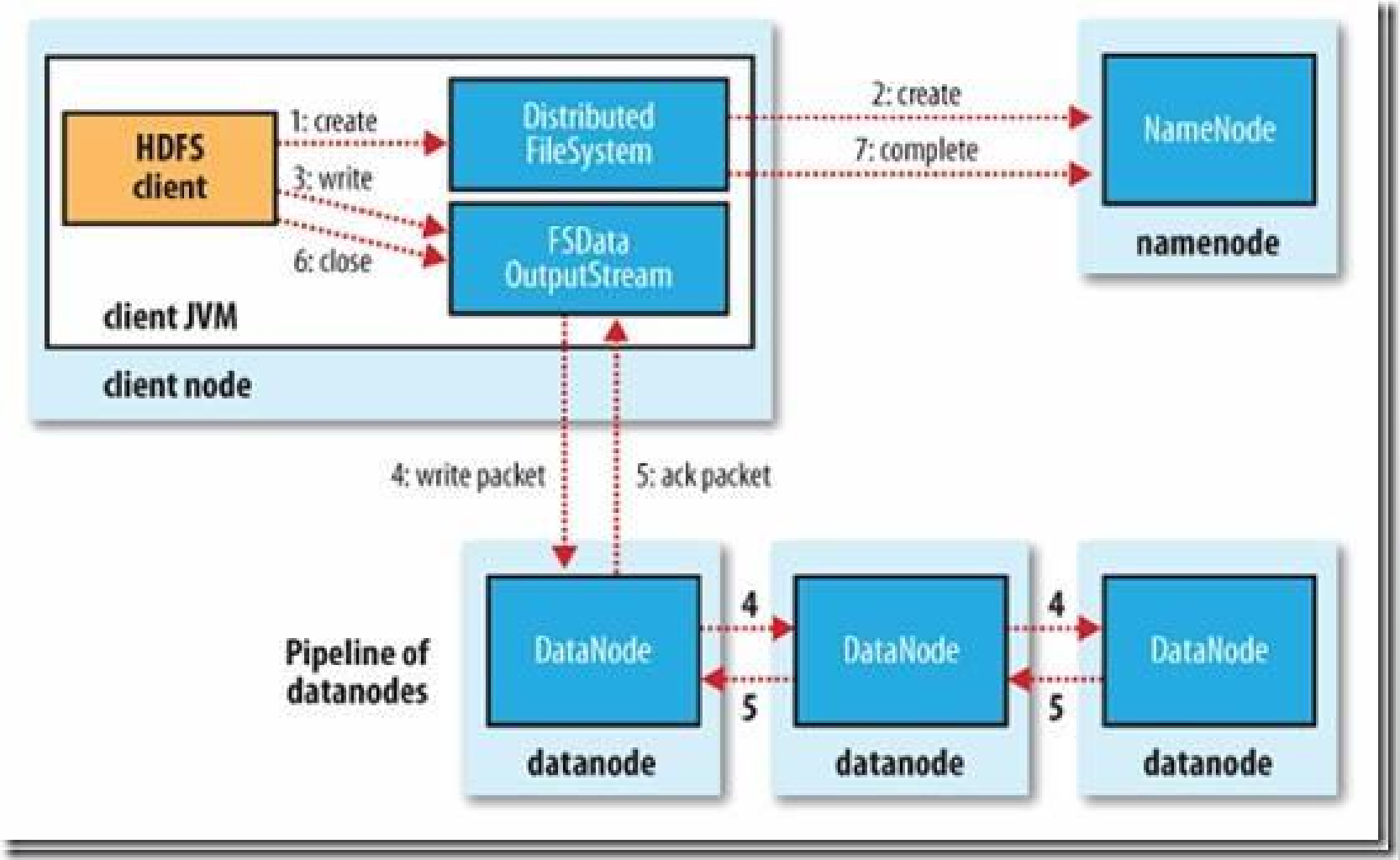
课程大纲

- HDFS存储架构剖析
- NameNode启动流程
- Yarn平台架构剖析
- HDFS JAVA API讲解
- HDFS 文件读写流程

读HDFS上文件流程


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

写HDFS上文件流程


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
DataNode
4
5
datanode
DataNode
datanode

总结

- 理解HDFS架构优缺点
- MapReduce任务执行流程
- Maven和eclipse基本配置
- HDFS API基本使用