Big Data Import&Export

Tianjiqx

2021.01.19

背景

- ArgoDB导入导出功能

- 现状：
  - 导入：本地text、orc等格式原始数据put到hdfs，然后对hdfs建外表插入到argodb表，速度43MB/s
  - 导出：argodb表转成text，orc格式写到hdfs，然后用get从hdfs下载到本地。速度46MB/s
- 需求：
  - 提升导入导出性能，达到网络，或者磁盘瓶颈

HDFS I/O PATH

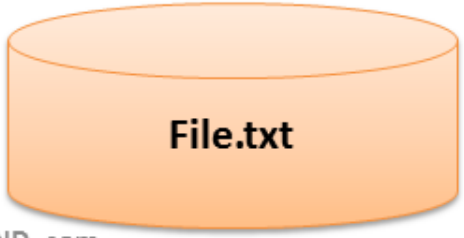
Typical Workflow

- Load data into the cluster (HDFS writes)
- Analyze the data (Map Reduce)
- Store results in the cluster (HDFS writes)
- Read the results from the cluster (HDFS reads)

Sample Scenario:

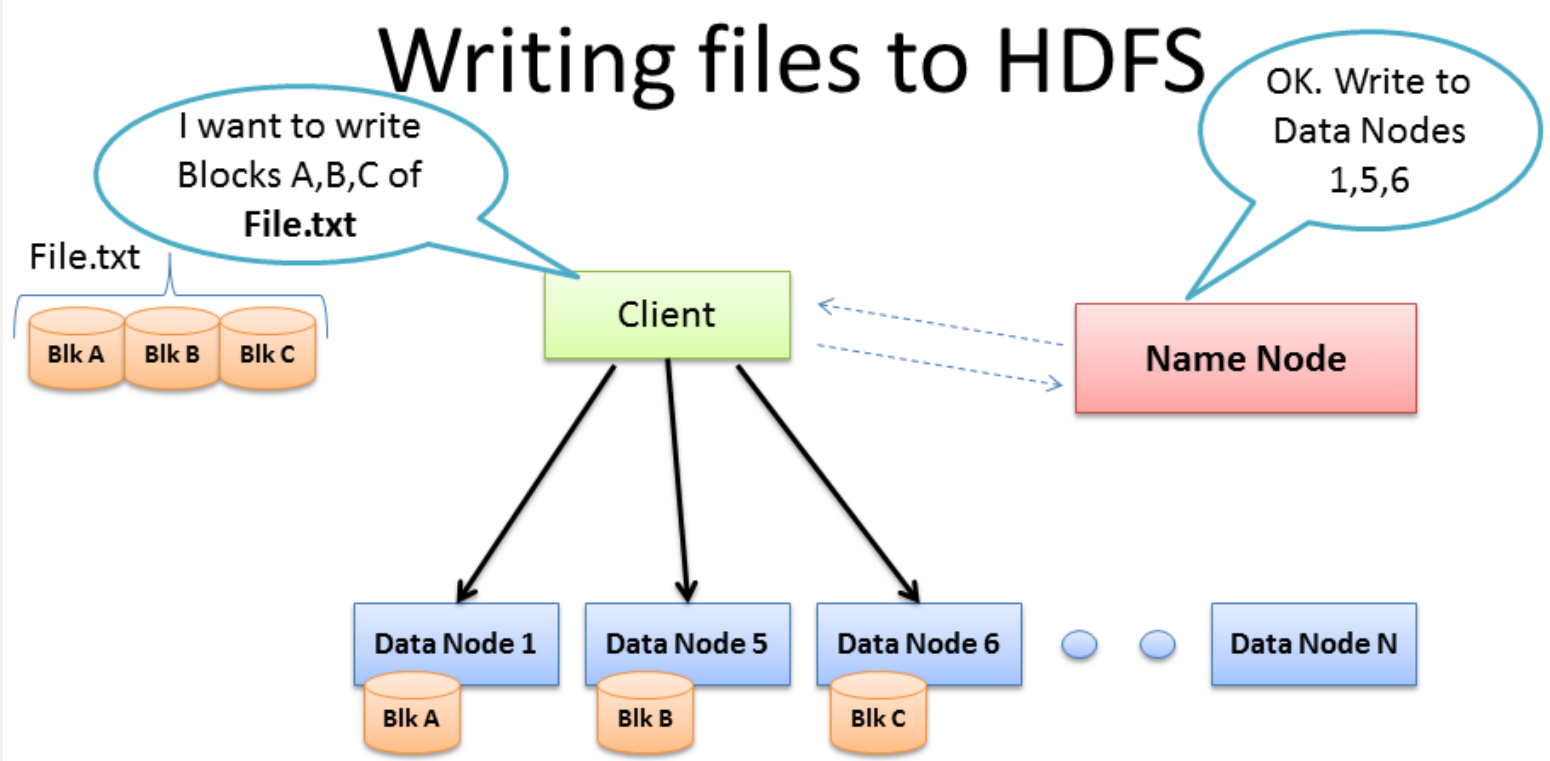
How many times did our customers type the word "**Refund**" into emails sent to customer service?

Huge file containing all emails sent to customer service



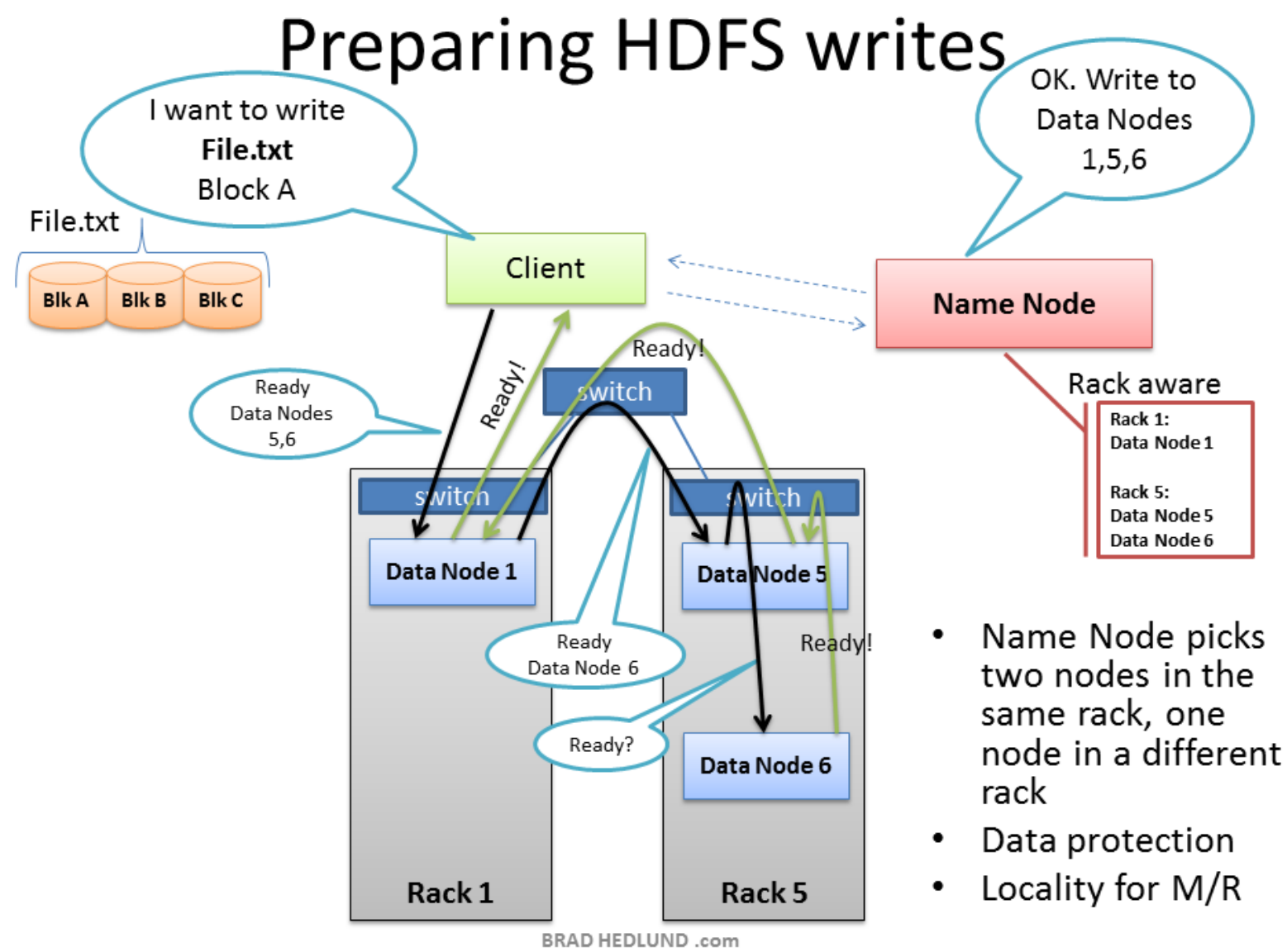
BRAD HEDLUND .com

HDFS I/O PATH



- Client consults Name Node
- Client writes block directly to one Data Node
- Data Nodes replicates block
- Cycle repeats for next block

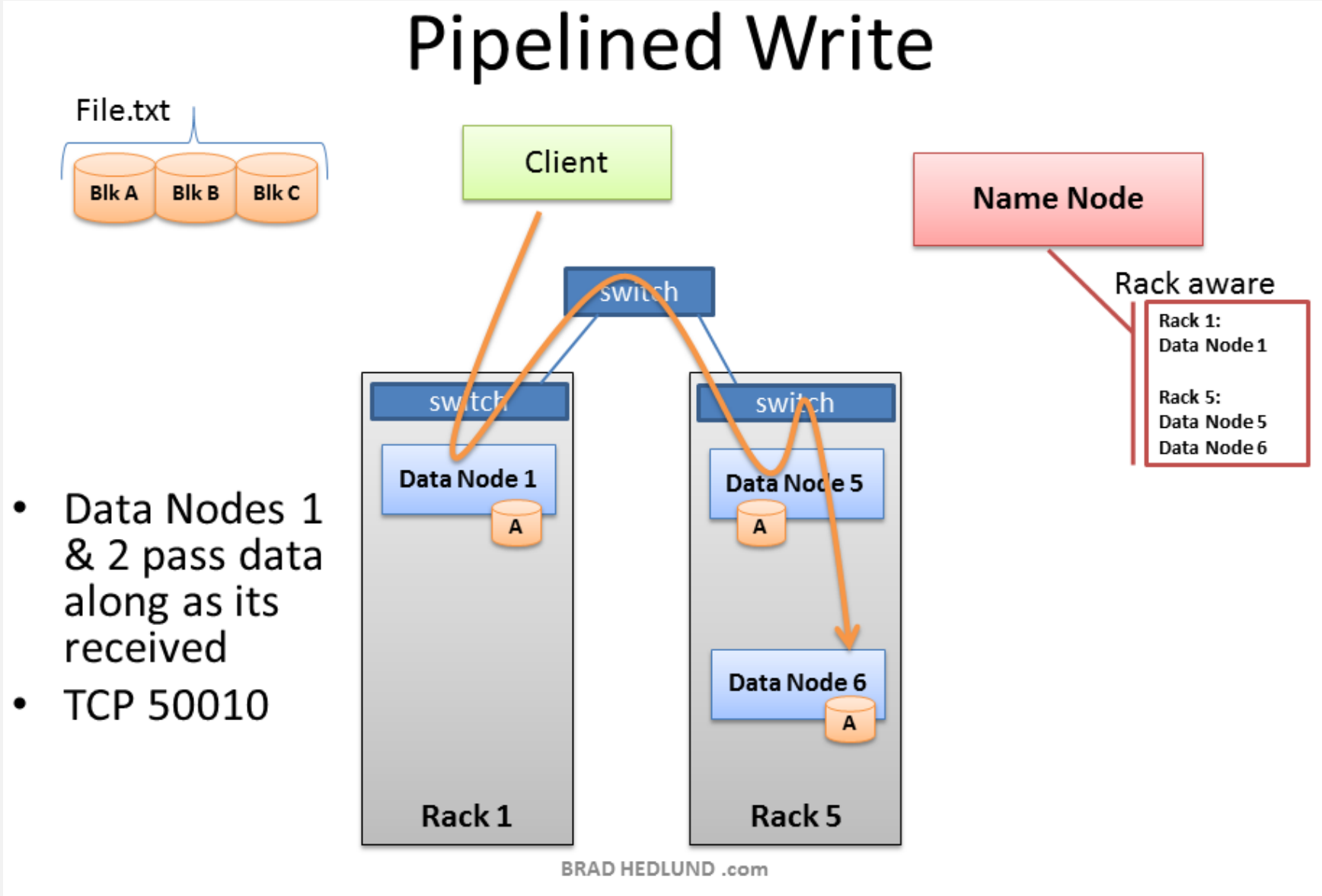
HDFS I/O PATH



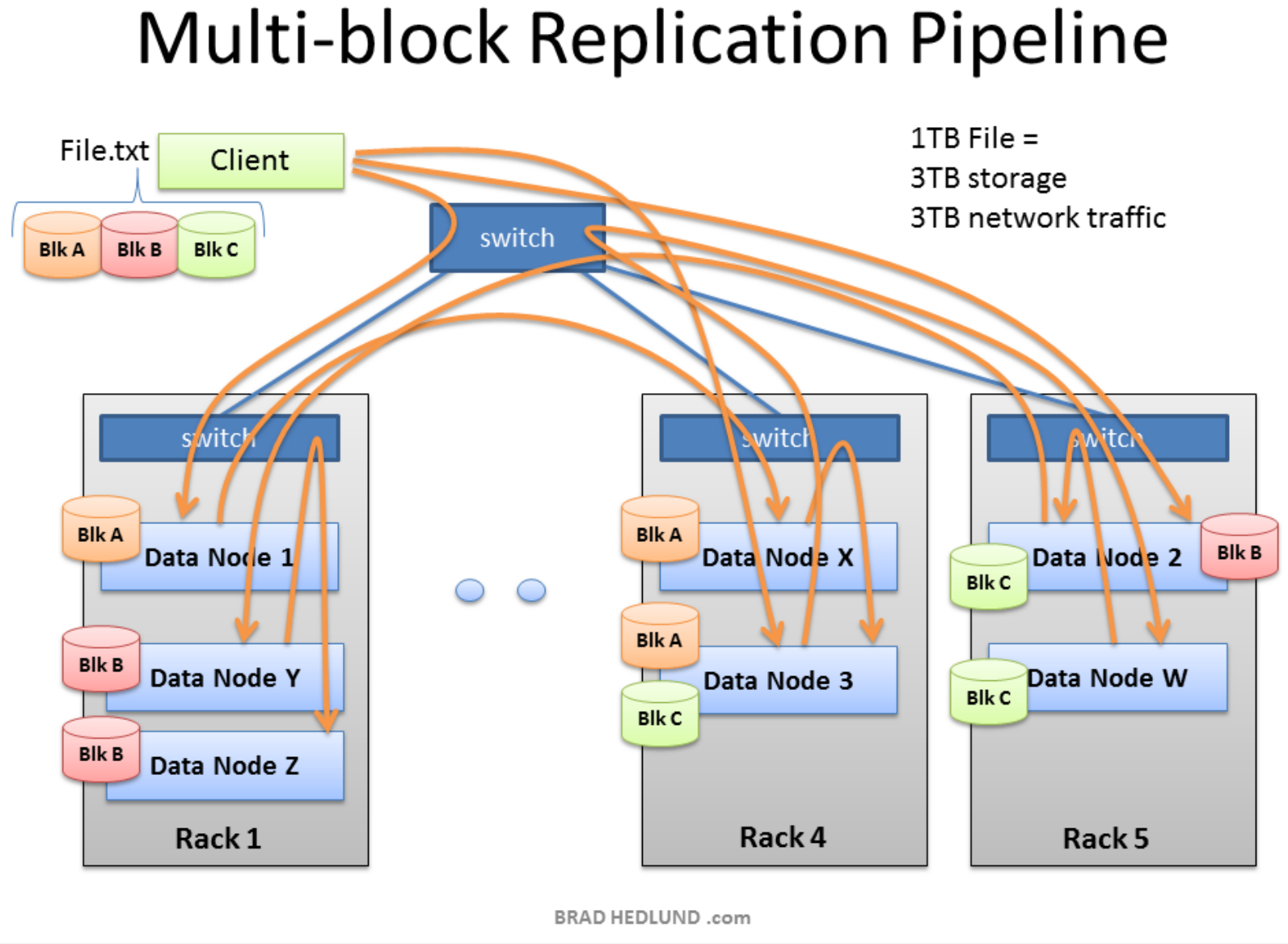
- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

HDFS I/O PATH

直到块A成功写入所有三个节点后，下一个块才会开始

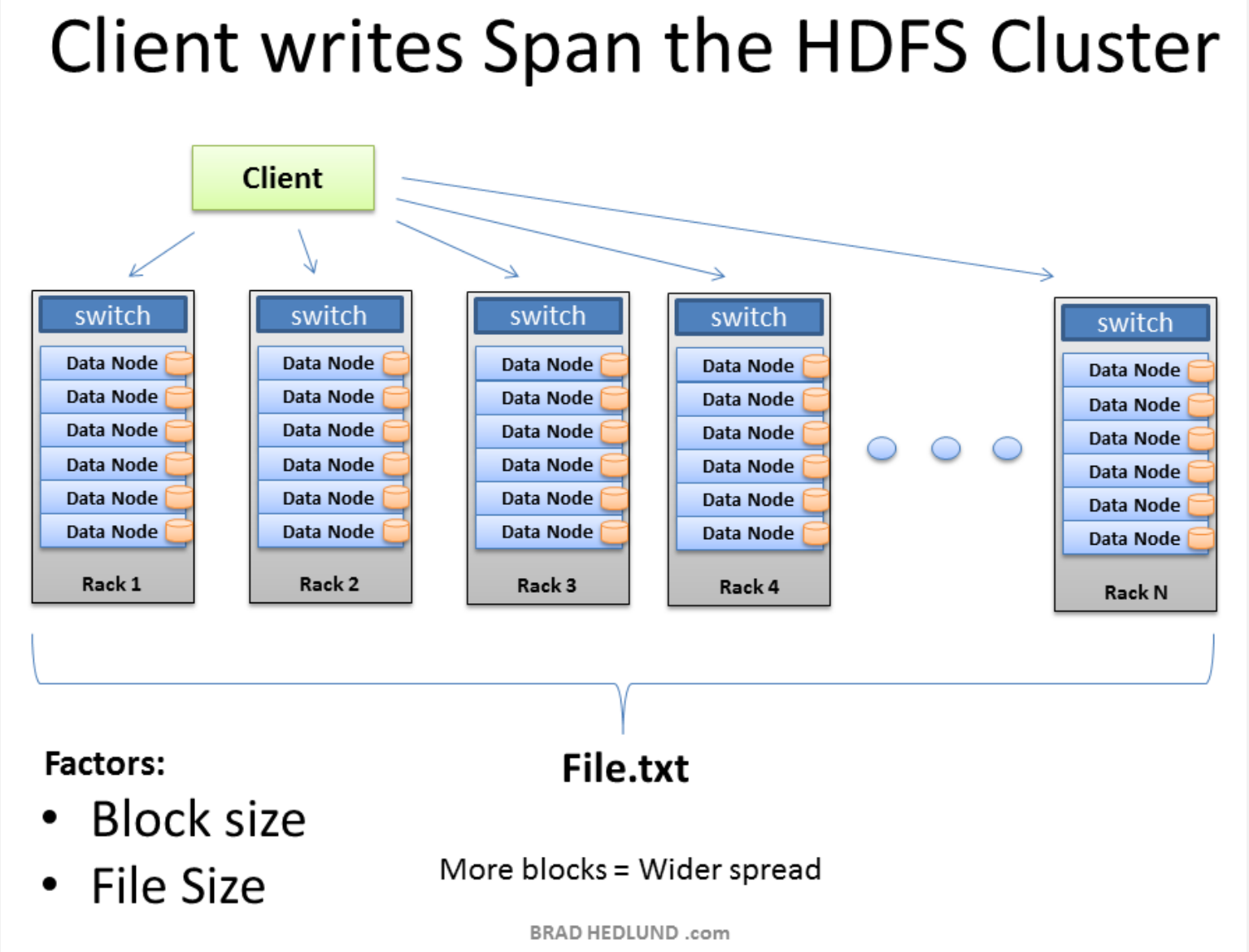


HDFS I/O PATH

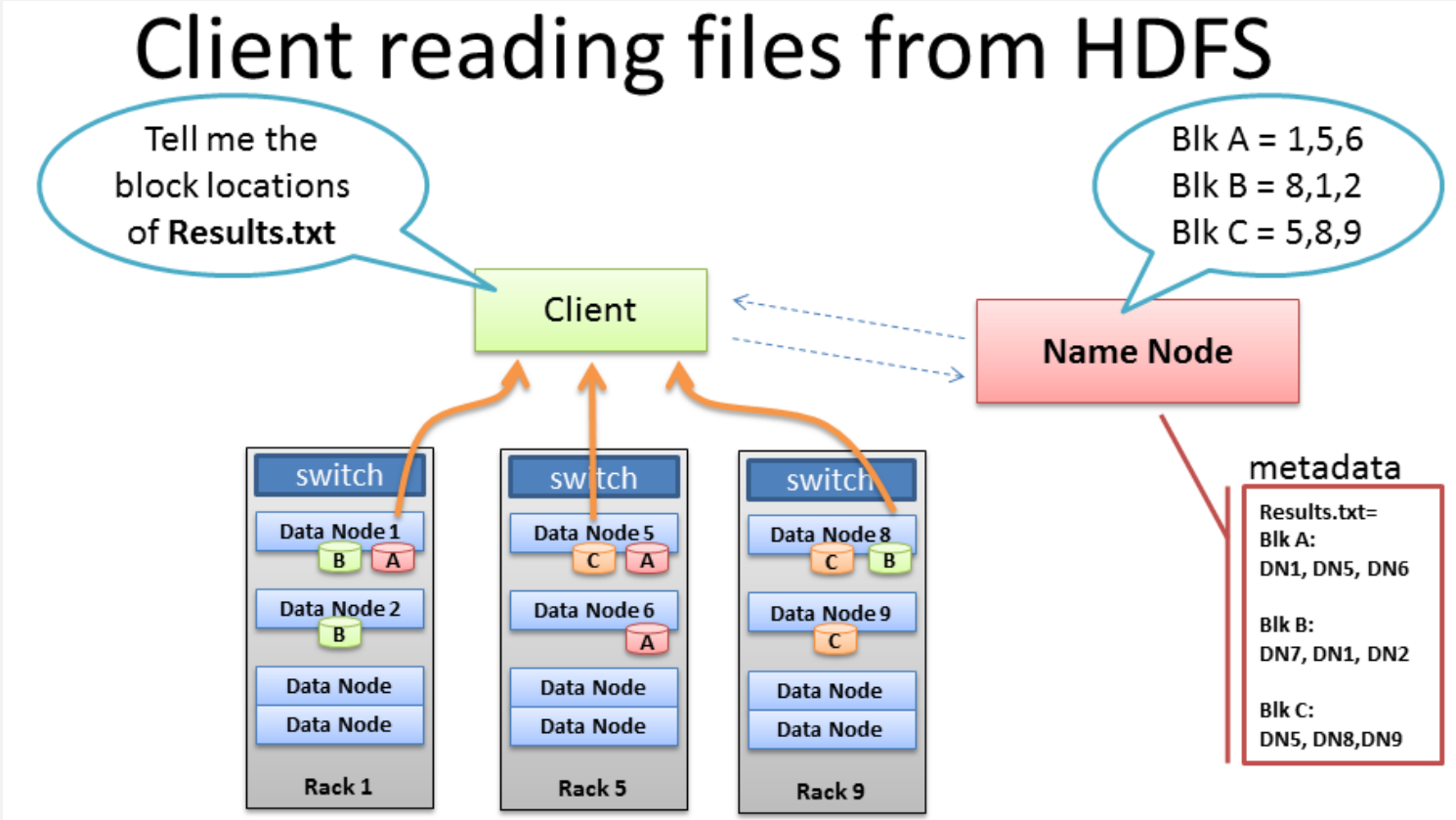

Multi-block Replication Pipeline
File.txt
Client
Blk A
Blk B
Blk C
switch
1TB File =
3TB storage
3TB network traffic
switch
Blk A
Data Node 1
Blk B
Data Node Y
Blk B
Data Node Z
Rack 1
switch
Blk A
Data Node X
Blk A
Data Node 3
Blk C
Rack 4
switch
Blk C
Data Node 2
Blk B
Blk C
Data Node W
Rack 5
BRAD HEDLUND .com

HDFS I/O PATH

组成**文件的块越多**，数据可能会传播的计算机越多，包含的数据的CPU内核和磁盘驱动器越多，则意味着**并行处理能力越强**

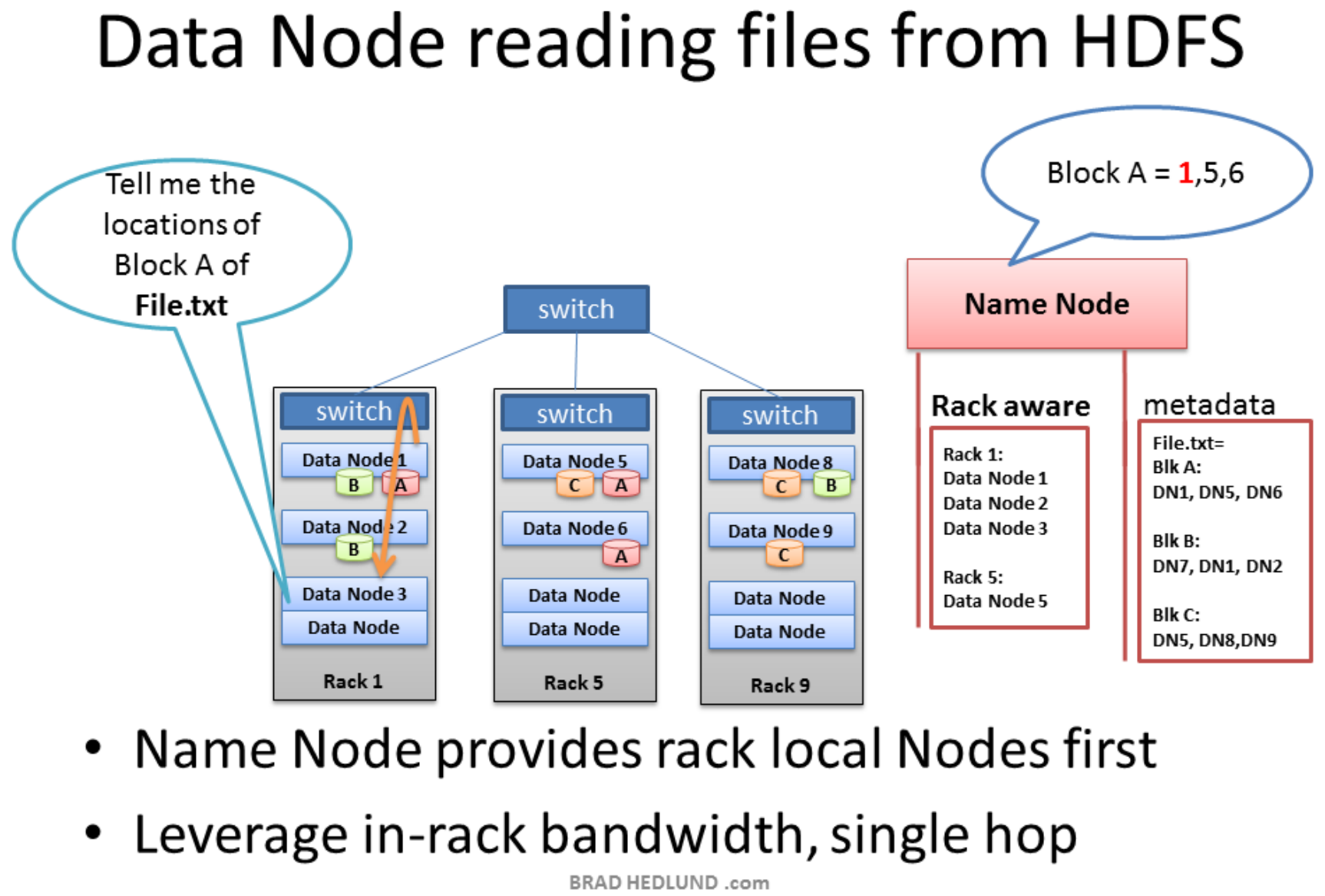


HDFS I/O PATH



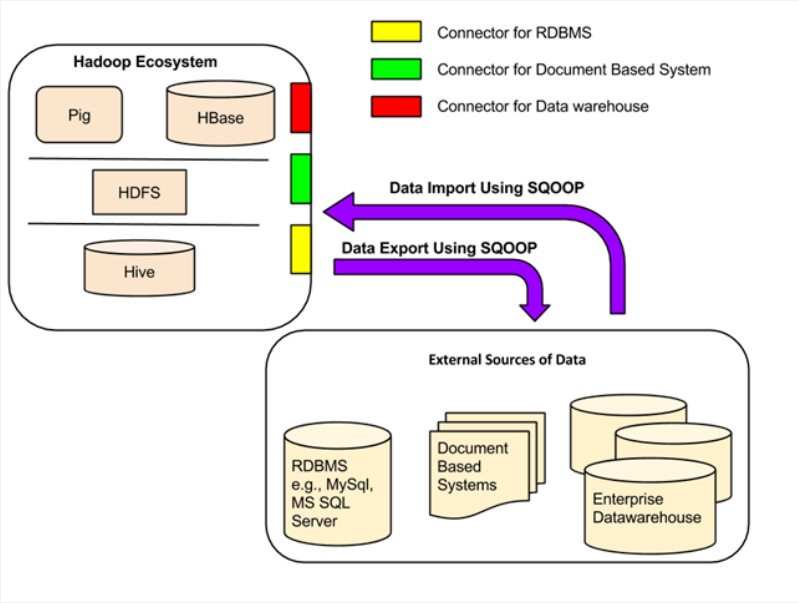
- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

HDFS I/O PATH


Data Node reading files from HDFS
Tell me the locations of Block A of File.txt
Block A = 1,5,6
Name Node
switch
switch
Data Node 1
B
A
Data Node 2
B
Data Node 3
Data Node
Rack 1
switch
Data Node 5
C
A
Data Node 6
A
Data Node
Data Node
Rack 5
switch
Data Node 8
C
B
Data Node 9
C
Data Node
Data Node
Rack 9
Rack aware
Rack 1:
Data Node 1
Data Node 2
Data Node 3
Rack 5:
Data Node 5
metadata
File.txt=
Blk A:
DN1, DN5, DN6
Blk B:
DN7, DN1, DN2
Blk C:
DN5, DN8,DN9
• Name Node provides rack local Nodes first
• Leverage in-rack bandwidth, single hop
BRAD HEDLUND .com

Sqoop

- Apache Sqoop（SQL到Hadoop）旨在支持从结构化数据存储（例如关系数据库，企业数据仓库和NoSQL系统）将数据批量导入HDFS（导出到关系数据库）。Sqoop基于连接器体系结构，该体系结构支持插件以提供与新外部系统的连接。

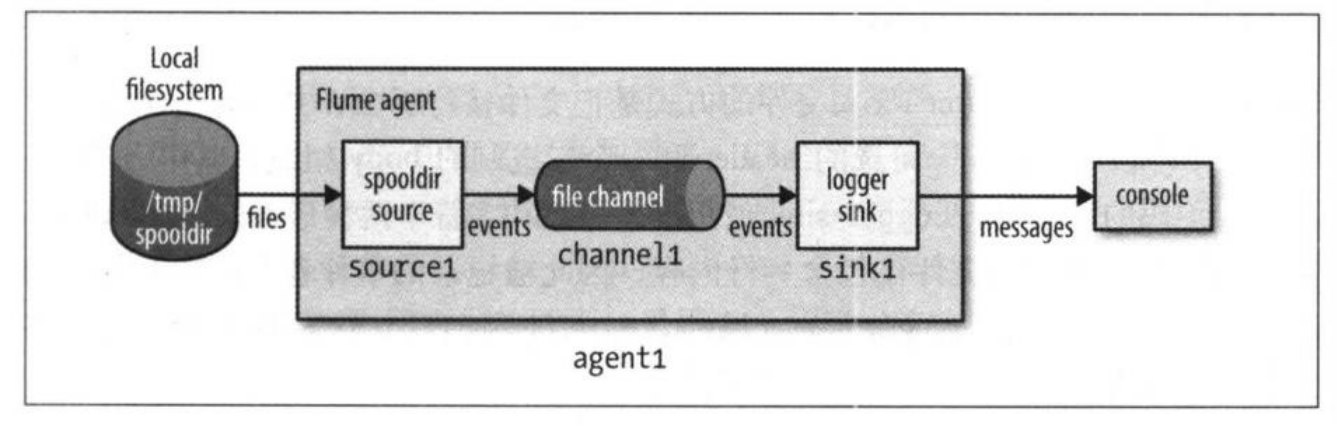


- 特点
  - 基于连接器（JDBC）
  - MapReduce架构
  - 丰富配置（直接模式导入，增量导入）
  - 可扩展
- Example
  - mysql导入，MySQLDumpInputFormat，DataDrivenDBInputFormat，DBSplitter（切分列（主键）min,max值生成filter条件，切分任务）等类

Flume

- 目的：向Hadoop导入基于事件的海量数据
  - Web服务日收集的日志文件
  - 扩展可以写入Hbase，Solr

- Flume代理（java进程）
  - 持续运行的数据源（channels）
  - 数据目标sink（channel）
  - 连接source和sink的channel，存储直至转发事件



Flume

- Spooling direcetory source
  - 监听文件夹，文件按行拆分，每行是一个Flumne事件
  - 所有事件全部传递到channel，该文件标记完成(.COMPLETED)
  - 事务
- File channel
  - 持久性，写入的channel后，代理重新启动，事件不会丢失（其他memory channel）
  - Channel到sink也是一个事务
  - At least once（代理重启前，部分事务被提交到channel，重启导致重新提交到channel）
    - 避免两阶段提交
    - MR删除重复数据
  - 写到sink的临时文件设置hdfs.inUsePrefix（下划线）

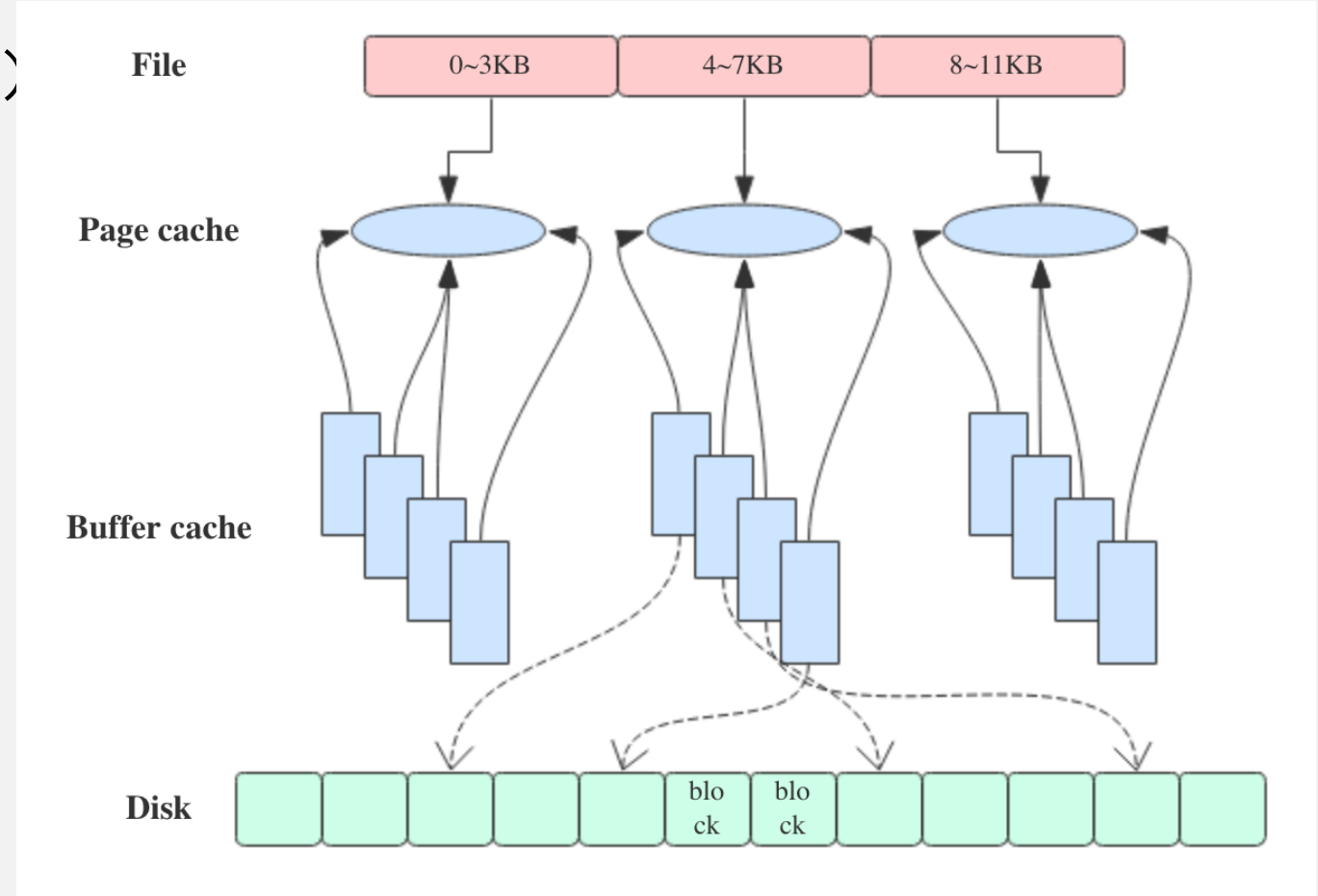
Kafka

- 高吞吐的实现
  - Broker NIO异步消息处理，IO线程与业务线程分离
  - 顺序读写磁盘
    - topic持久化
  - PageCache
  - Linux“零拷贝”
    - sendfile
  - 端到端的消息压缩

Kafka：PageCache

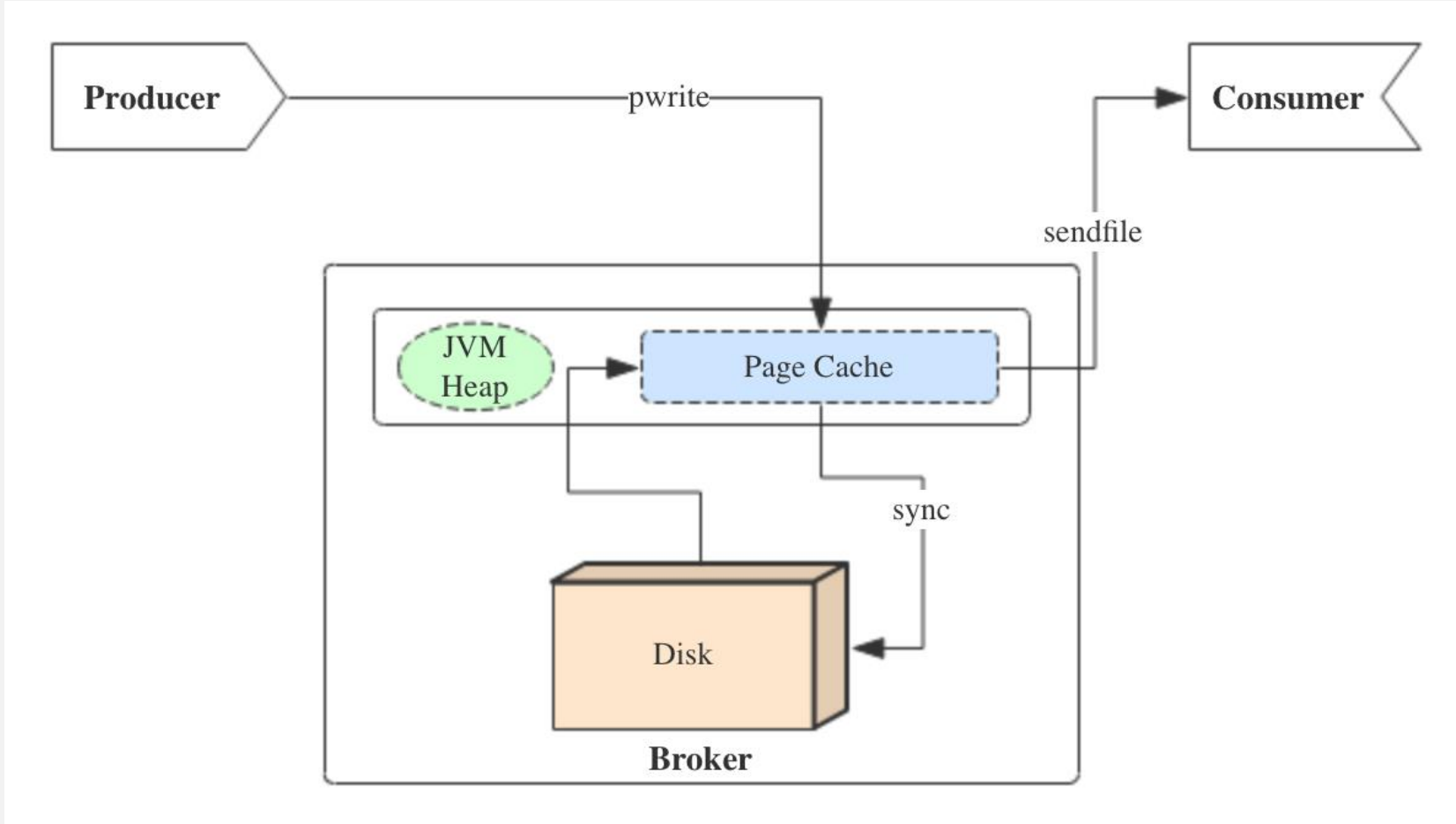
- page cache & buffer cache
  - page cache用于缓存文件的页数据，buffer cache用于缓存块设备（如磁盘）的块数据
  - Linux2.4后合并（free -m）

- block size大小为1KB
- page size大小为4KB



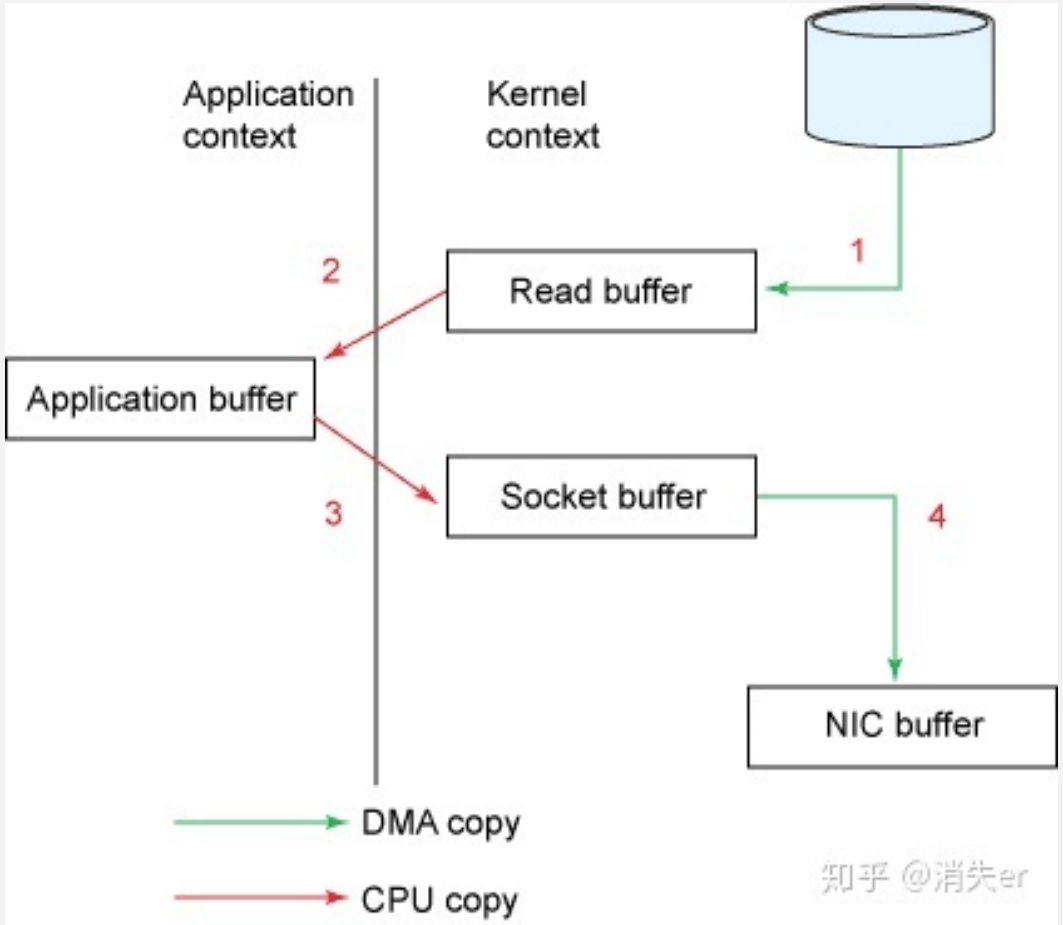
Kafka：PageCache

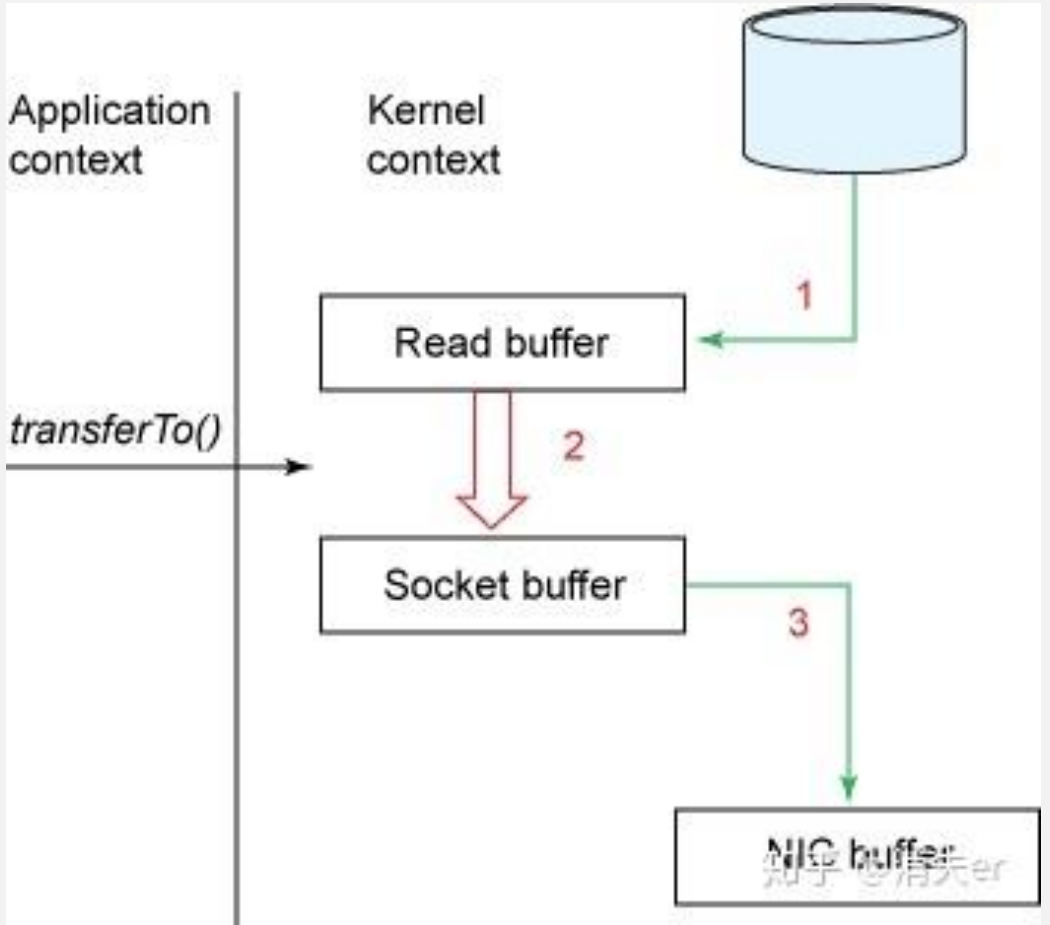
- 优势
  - 减少java对象包装开销
  - 减轻jvm GC
  - Kafka崩溃，不影响pagecahe，不丢缓存



Kafka: Sendfile

- buffer = File.read
- Socket.send(buffer) => FileChannel.transferTo()/transferFrom()





JAVA NIO

- 普通
  - Ioutils.copy
- FileChnnel
  - FileInputStream.getChnanel

- 速度比较
  - 普通io 读写118MB/s
  - Nio 读写 193MB/s

- demo演示

HIVE 压缩

- Map输出压缩

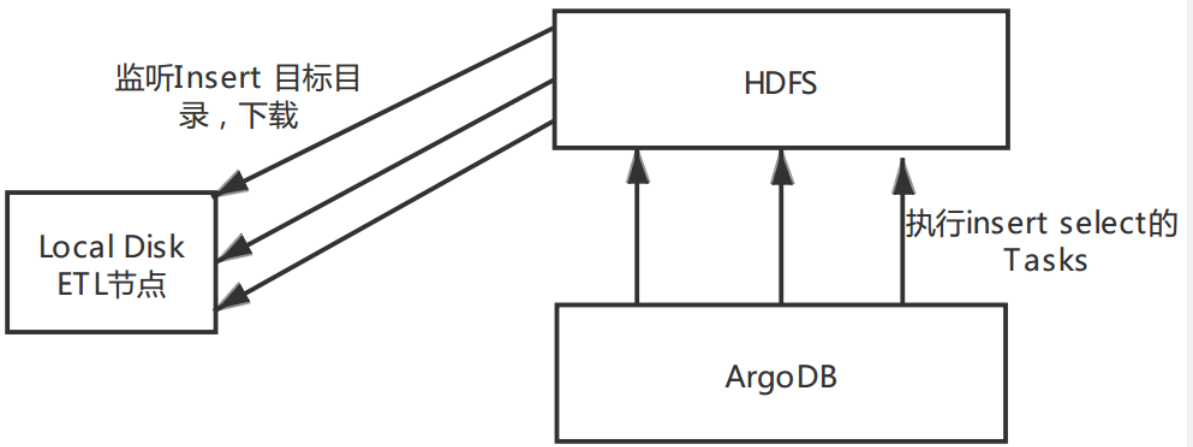
```
--(1)开启hive中间传输数据压缩功能
set hive.exec.compress.intermediate=true;
--(2)开启mapreduce中map的压缩功能
set mapreduce.map.output.compress=true;
--(3)设置mapreduce中map输出数据的压缩方式
set mapreduce.map.output.compress.codec=
org.apache.hadoop.io.compress.SnappyCodec;
```

- Reduce输出压缩

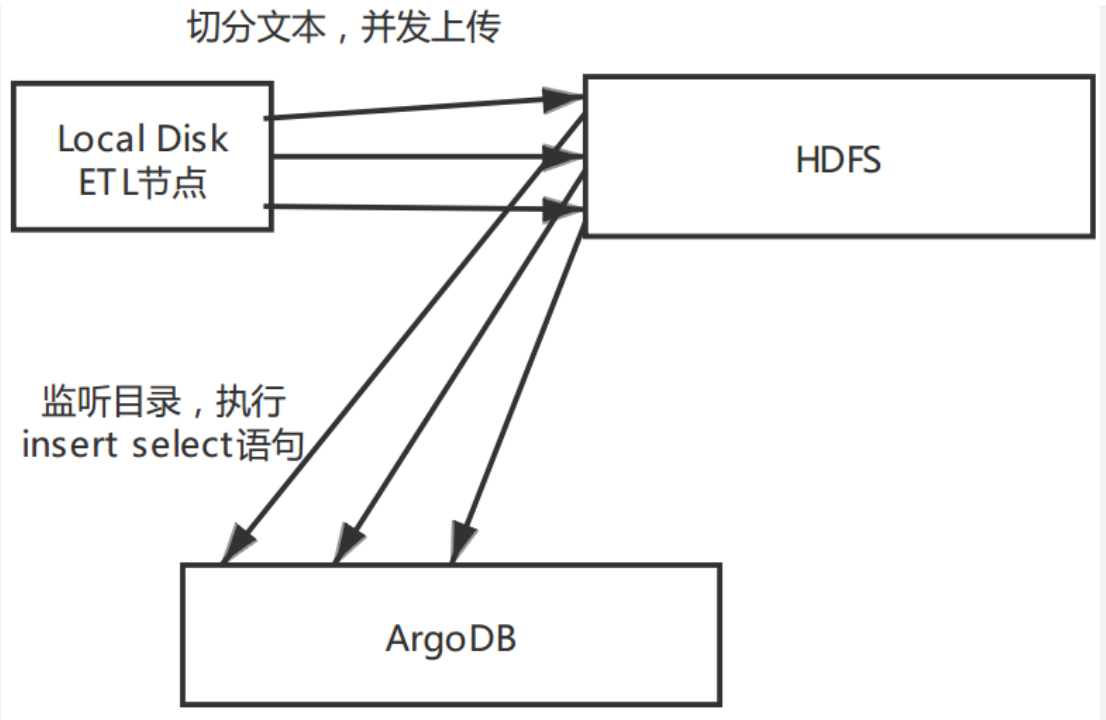
```
--（1）开启hive最终输出数据压缩功能
set hive.exec.compress.output=true;
--(2)开启mapreduce最终输出数据的压缩
set mapreduce.output.fileoutputformat.compress=true;
--(3)设置mapreduce最终的数据输出方式
set mapreduce.output.fileoutputformat.compress.codec
=org.apache.hadoop.io.compress.SnappyCodec;
--(4)设置mapreduce最终数据输出压缩为块压缩
set mapreduce.output.fileoutputformat.compress.type=BLOCK;
```

ArgoDB导入导出方案

- 导入导出方案
  - Pipeline
  - 压缩
  - fileChannel



- 导出速度
  - 非压缩73.5MB/s
  - 压缩155MB/s
  - Hdfs put/get(79MB/s,69MB/s)



参考

- https://cwiki.apache.org/confluence/display/HADOOP2/HadoopPresentations
- http://bradhedlund.com/(Understanding Hadoop Clusters and the Network)
- https://www.guru99.com/introduction-to-flume-and-sqoop.html
- https://sqoop.apache.org/docs/1.4.7/SqoopDevGuide.html
- 《Hadoop权威指南》
- 《企业大数据处理：Spark、Druid、Flume与Kafka应用实践》
- Kafka对PageCache的使用

https://blog.csdn.net/gx11251143/article/details/107620259

- Java NIO浅析https://tech.meituan.com/2016/11/04/nio.html
- 磁盘I/O那些事https://tech.meituan.com/2017/05/19/about-desk-io.html