HDFS

Леонид Налчаджи
leonid.nalchadzhi@gmail.com

Яндекс

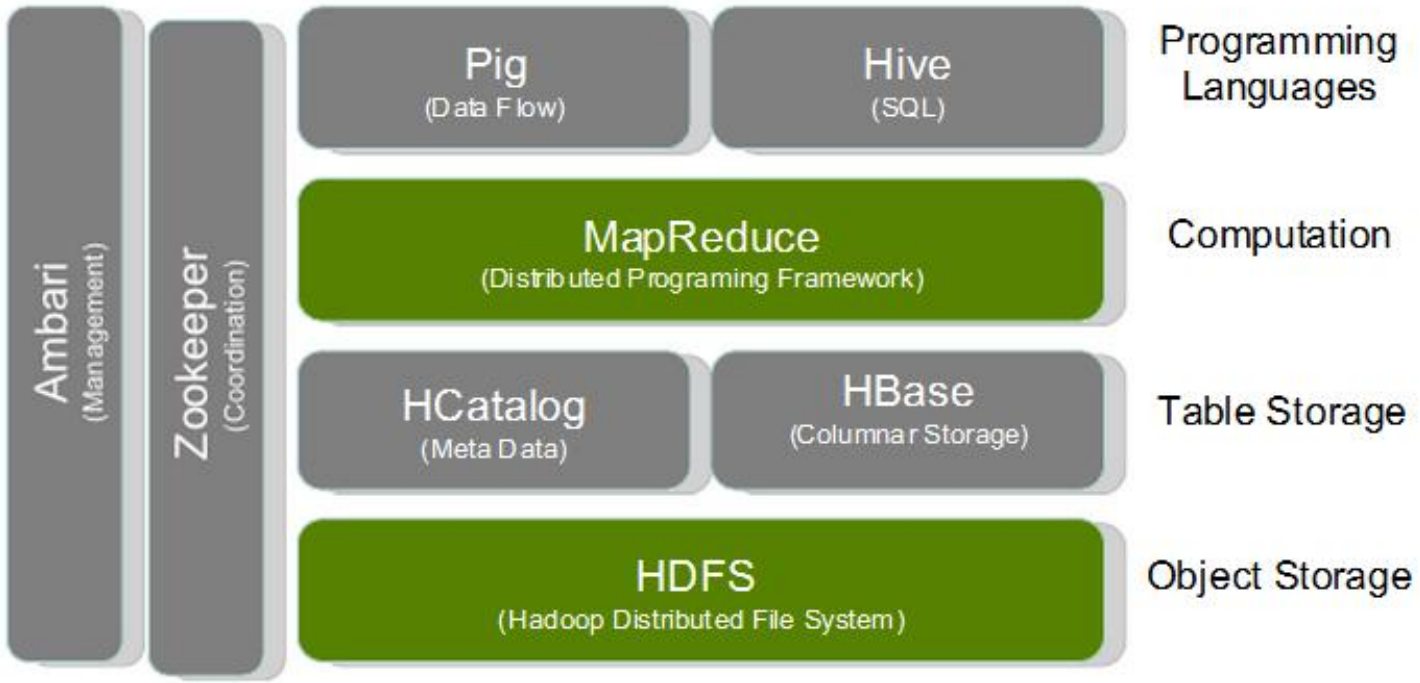
Intro

- Hadoop ecosystem
- History
- Setups

HDFS

- Features
- Architecture
- NameNode
- DataNode
- Data flow
- Tools
- Performance

Hadoop ecosystem



Doug Cutting

History

- 2002: Apache Nutch
- 2003: GFS
- 2004: Nutch Distributed Filesystem (NDFS)
- 2005: MapReduce
- 2006: subproject of Lucene => Hadoop

History

- 2006: Yahoo! Research cluster: 300 nodes
- 2006: BigTable paper
- 2007: Initial HBase prototype
- 2007: First usable HBase (Hadoop 0.15.0)
- 2008: 10,000-core Hadoop cluster

Яндекс

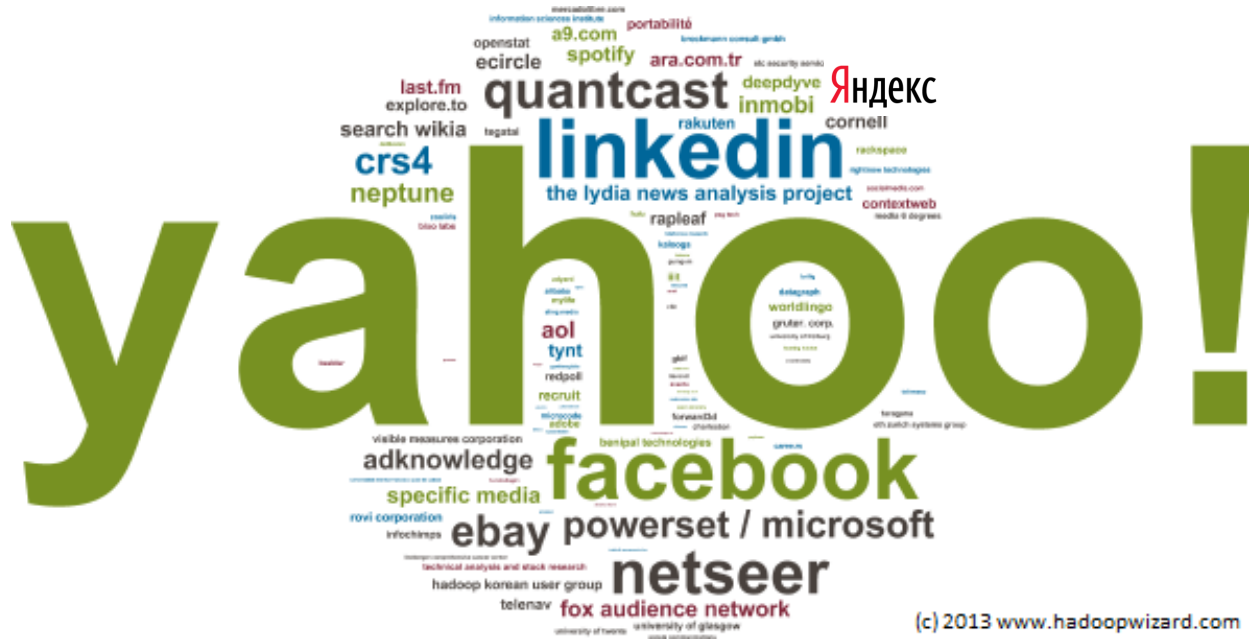
- 2 MapReduce clusters MapReduce (50 nodes, 100 nodes)
- 2 HBase clusters
- 500 nodes cluster is on its way :)
- Fact extraction, data mining, statistics, reporting, analytics

Facebook

- 2 major clusters (1100 and 300 nodes)
- Messages; machine learning, log storage, reporting, analytics
- 8B messages/day, 2Pb online data, growing 250 Tb/ months

EBay

- 532 nodes cluster (8 * 532 cores, 5.3PB).
- Heavy usage of MapReduce, HBase, Pig, Hive
- Search optimization and Research


yahoo!
facebook
linkedin
quantcast
netseer
ebay
powerset / microsoft
adknowledge
specific media
fox audience network
hadoop korean user group
the lydia news analysis project
crs4
neptune
search wikia
explore.to
last.fm
ecircle
openstat
a9.com
spotify
ara.com.tr
portabilité
deepdyve
inmobi
Яндекс
cornell
rakuten
contextweb
rapleaf
aol
tynt
redpoll
recruit
worldlingo
visible measures corporation
benipal technologies
rovi corporation
infochimps
telenav
(c) 2013 www.hadoopwizard.com

HDFS

Features

Features

- Distributed file system
- Replicated, highly fault-tolerant
- Commodity hardware
- Aimed for Map Reduce
- Pure java
- Large files, huge datasets

Features

- Traditional hierarchical file organization
- Permissions
- No soft links

Features

- Streaming data access
- Write-once-read-many access model for files
- Strictly one writer

Hardware Failure

- Hardware failure is the norm rather than the exception
- Detection of faults and quick, automatic recovery

Not good for

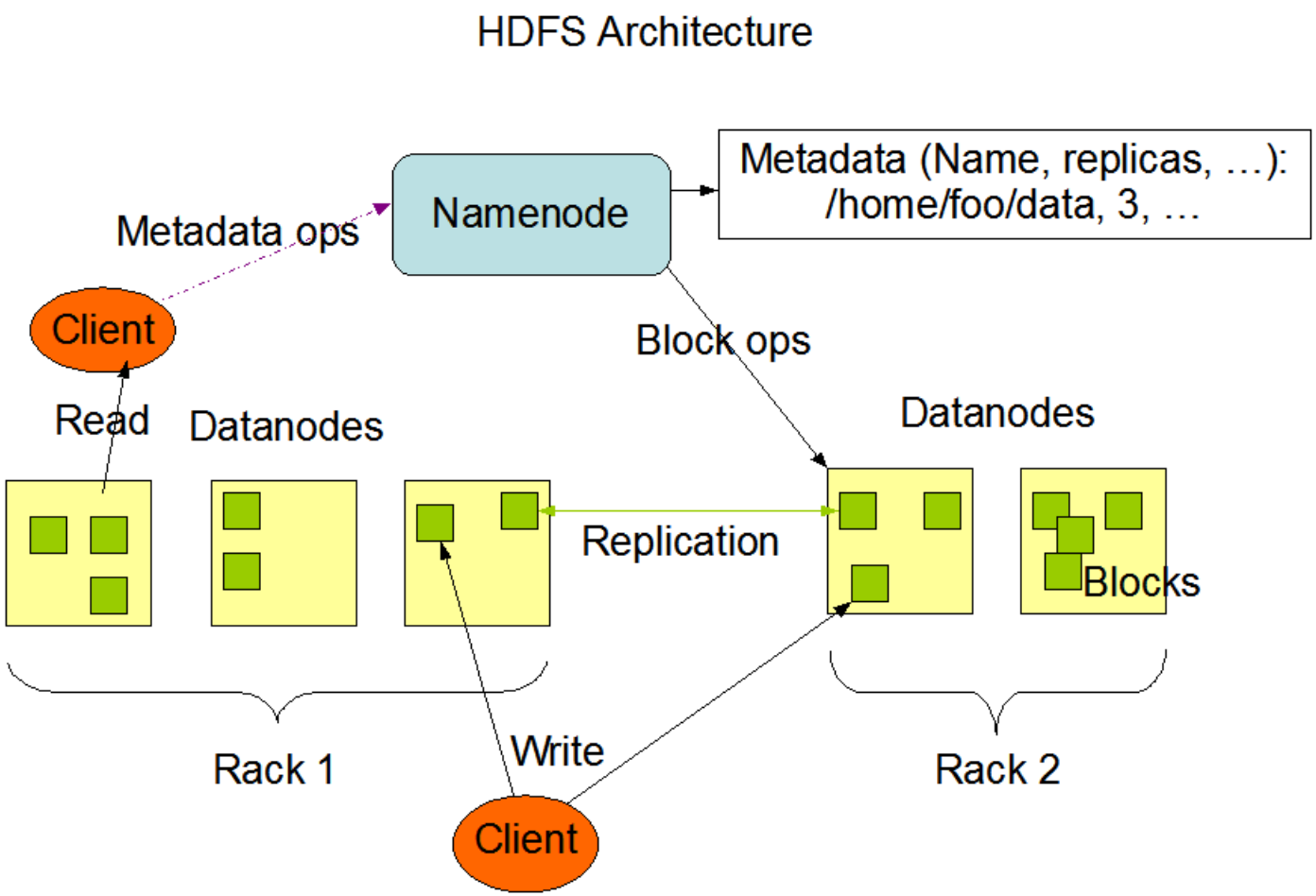
- Multiple data centers
- Low latency data access (<10 ms)
- Lots of small files
- Multiple writers

Architecture

Architecture

- One NameNode, many DataNodes
- File is a sequence of blocks (64 Mb)
- Meta data in memory of namenode
- Client interacts with datanodes directly

Architecture



Big blocks

- Let's say we want to achieve 1% seek overhead
- Seek time ~10ms, transfer rate ~100 Mb/s
- So block size must be ~100Mb

NameNode

NameNode

- Manages the FS namespace
- Keeps in Memory all files and blocks
- Executes FS operations like opening, closing, and renaming
- Makes all decisions regarding replication of blocks
- SPOF

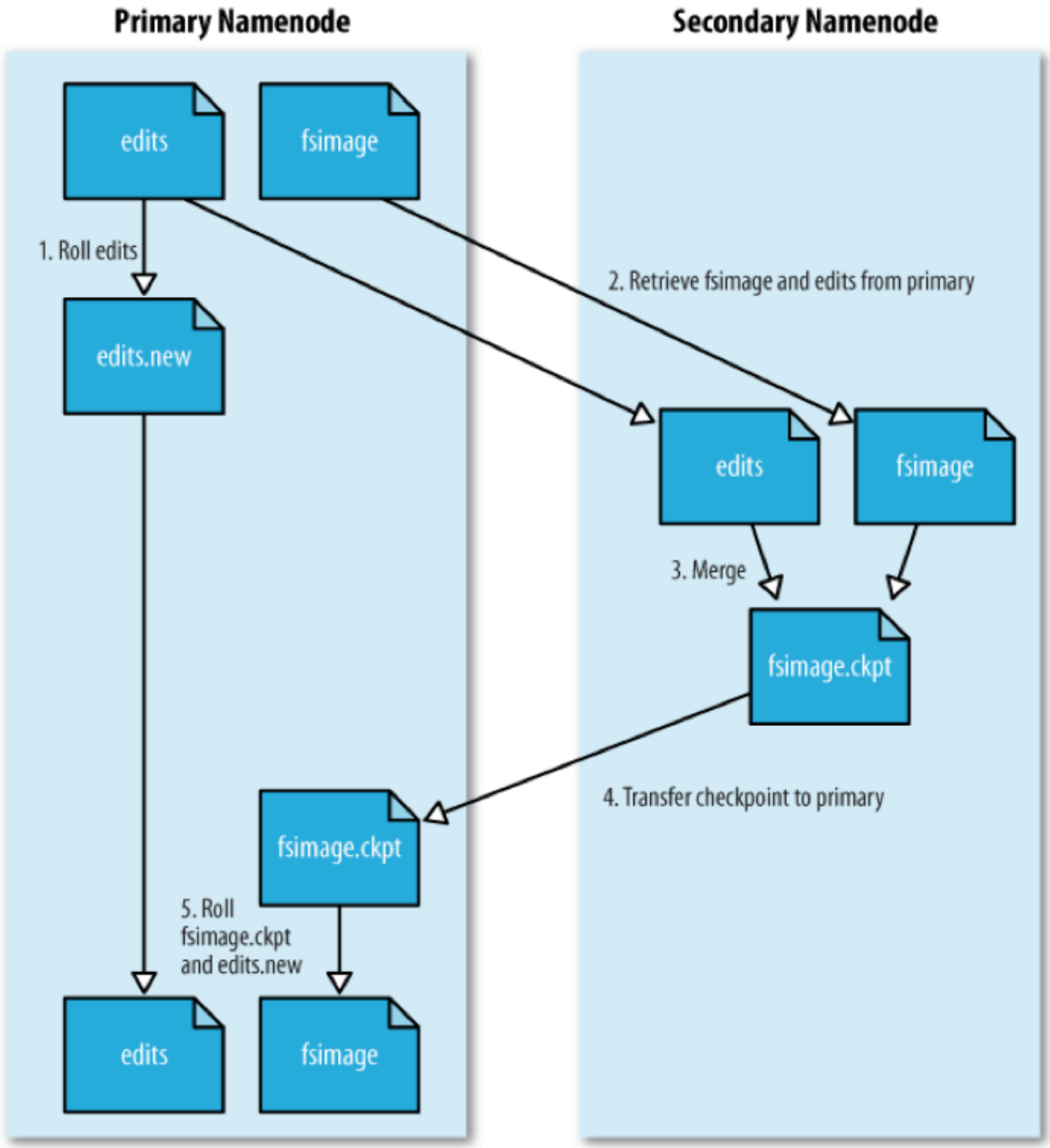
NameNode

- NS stored in 2 files: namespace image, edit log
- Determines the mapping of blocks to DataNodes
- Receives a Heartbeat and a Blockreport from each of the DataNodes

NameNode

NameNode

- Store persistent state to several file system
- Secondary MasterNode
- HDFS High-Availability
- HDFS Federation


Primary Namenode
Secondary Namenode
edits
fsimage
1. Roll edits
edits.new
2. Retrieve fsimage and edits from primary
edits
fsimage
3. Merge
fsimage.ckpt
4. Transfer checkpoint to primary
fsimage.ckpt
5. Roll
fsimage.ckpt
and edits.new
edits
fsimage

DataNode

DataNode

- Usually 1 DataNode 1 machine
- Serves read and write client requests
- Performs block creation, deletion, and replication

DataNode

- Each block represented as 2 files: data, metadata
- Metadata contains checksums, generation stamp
- Handshake while starting the cluster to verify ID of namespace and SW version

DataNode

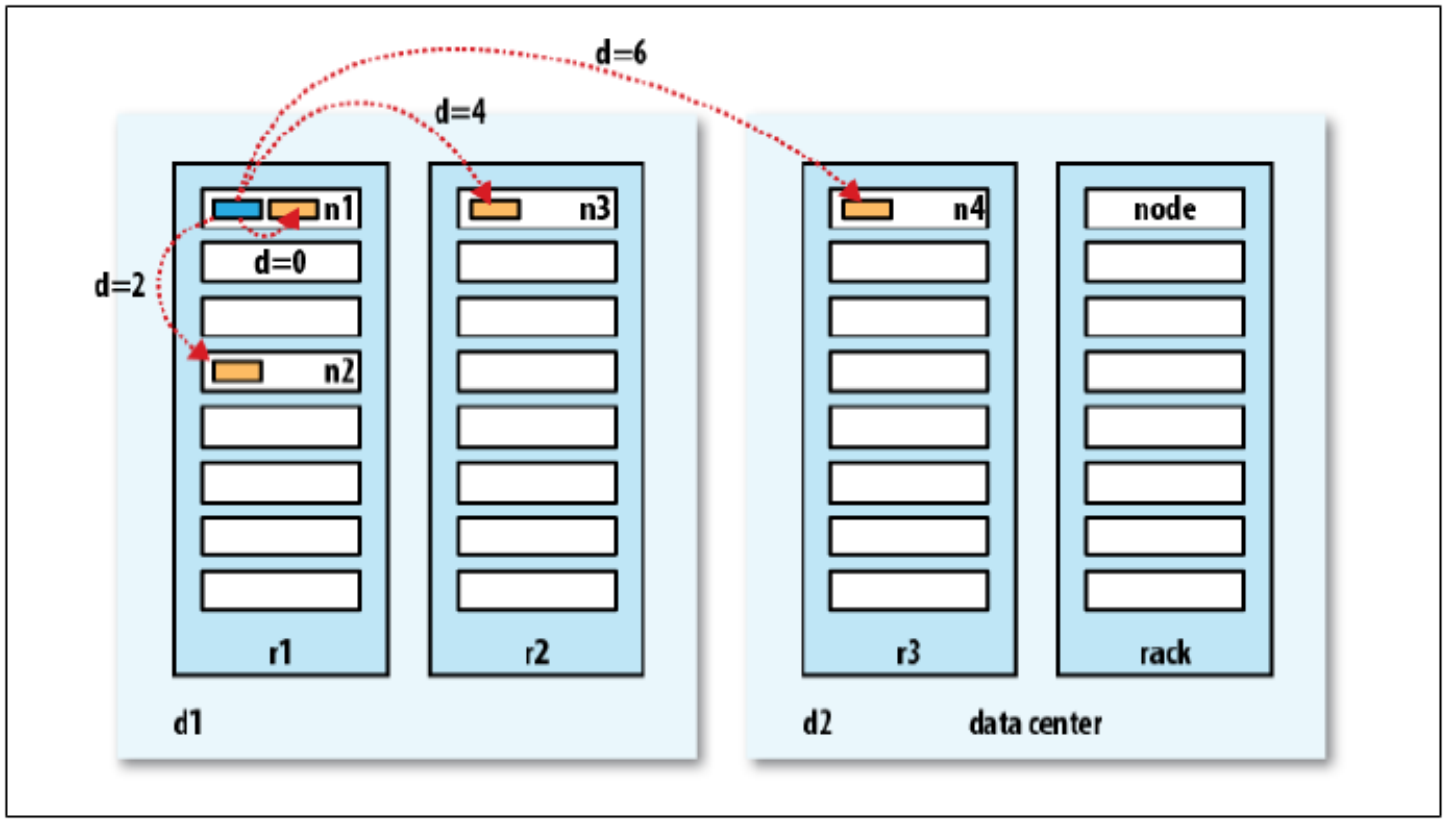
- NameNode never calls DataNodes
- Replicate block
- Remove local replica
- Re-register/shut down
- Send an immediate block report

Data Flow

Distance

- HDFS is rack aware
- Network is presented as tree
- d(n1, n2) = d(n1, A) + d (n2, A), A -- closest common ancestor

Distance



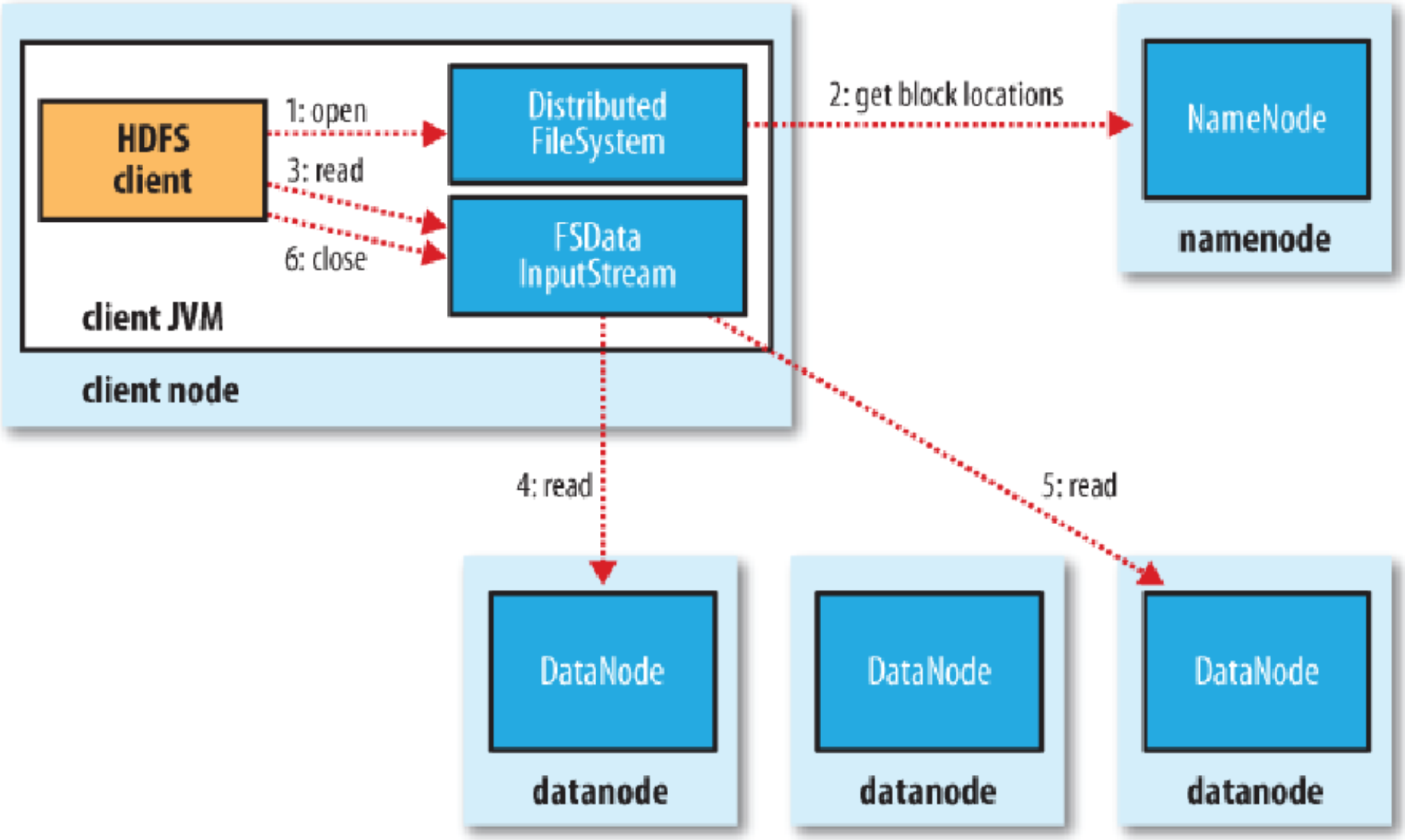
Read

- Get list of blocks locations from NameNode
- Iterate over blocks and read
- Verifies checksums

Read

- On fail or when unable to connect DN:
  - try another replica
  - remember that
  - tell NameNode

Read


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

Read

```
final Configuration conf = new Configuration();
final FileSystem fs = FileSystem.get(conf);
InputStream in = null;
try {
        in = fs.open(new Path("/user/test-hdfs/somefile"));
        //do whatever with inpustream
} finally {
        IOUtils.closeStream(in);
}
```

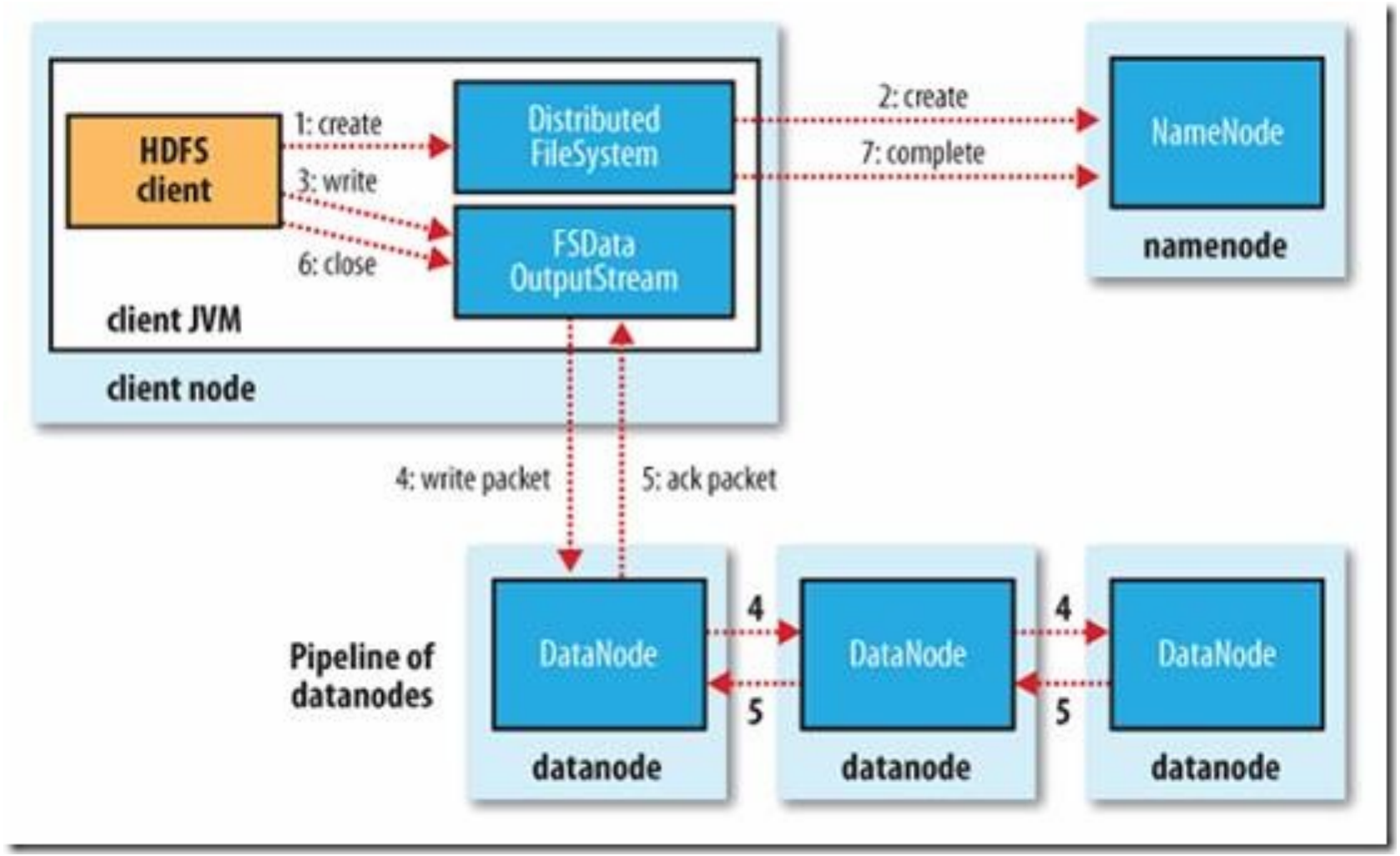
Write

- Ask NameNode to create file
- NameNode performs some checks
- Client ask for list of DataNodes
- Forms a pipeline and ack queue

Write

- Send packets to the pipeline
- In case of failure
  - close pipeline
  - remove bad DN, tell NN
  - retry sending to good DNs
  - block will be replicated asynchronously

Write


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

Write

```
final Configuration conf = new Configuration();
final FileSystem fs = FileSystem.get(conf);
FSDataOutputStream out = null;
try {
        out = fs.create(new Path("/user/test-hdfs/newfile"));
        //write to out
} finally {
        IOUtils.closeStream(in);
}
```

Block Placement

- Reliability/Bandwidth trade off
- By default: same node, 2 random nodes from another rack, other random nodes
  - No Datanode contains more than one replica of any block
  - No rack contains more than two replicas of the same block

Tools

Balancer

- Compare utilization of node with utilization of cluster
- Guarantees that the decision does not reduce either the number of replicas or the number of racks
- Minimizes the inter-rack data copying

Block scanner

- Periodically scans replica, verifies checksums
- Notifies NameNode if checksum fails
- Replicate first, then delete

Performance

Performance

- Cluster ~3500 nodes
- Total bandwidth is linear to number of nodes
- DFSIO Read: 66 MB /s per node
- DFSIO Write: 40 MB /s per node

Performance

- Open file for read: 126100
- Create file: 5600
- Rename file: 8300
- Delete file: 20700
- DataNode heartbeat: 300000
- Block reports (block/s): 639700

Sources

- The Hadoop Distributed File System (K. Shvacko)
- HDFS Architecture Guide
- Hadoop: The Definitive Guide (O'Reilly)

HDFS

Леонид Налчаджи
leonid.nalchadzhi@gmail.com

Яндекс