Hadoop & HDFS

Some Theory

The Computational Setting

Computations that need the power of many computers

- large datasets
- use of thousands of CPUs in parallel

Big data management, storage, and analytics

- cluster as a computer

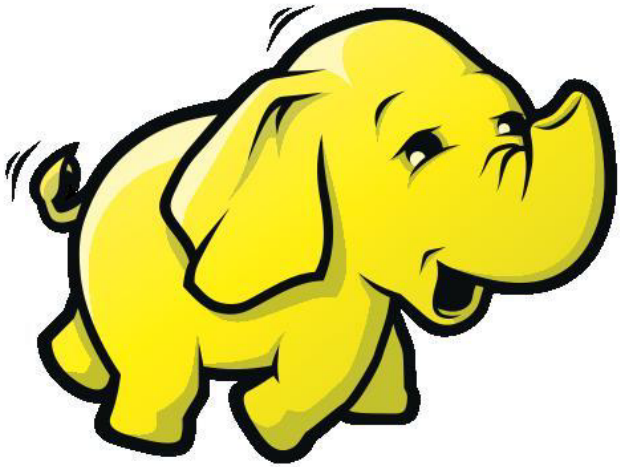
MapReduce & Hadoop: Historical Background

- **2003: Google published about its cluster architecture & distributed file system (GFS)**
- **2004: Google published about its MapReduce programming model used on top of GFS**
  - both GFS and MapReduce are written in C++ and are closed-source, with Python and Java APIs available to Google programmers only
- **2006: Apache & Yahoo! -> Hadoop & HDFS**
  - **open-source**, Java implementations of Google MapReduce and GFS with a diverse set of APIs available to public
  - evolved from Apache Lucene/Nutch open-source web search engine (Nutch MapReduce and NDFS)
- **2008: Hadoop becomes an independent Apache project**
  - Yahoo! uses Hadoop in production
- **Today: Hadoop is used as a general-purpose storage and analysis platform for big data**
  - other Hadoop distributions from several vendors including EMC, IBM, Microsoft, Oracle, Cloudera, etc.
  - many users (http://wiki.apache.org/hadoop/PoweredBy)
  - research and development actively continues…

Hadoop

What is Hadoop?

- Hadoop is an ecosystem of tools for processing “Big Data”.
- Hadoop is an open source project.

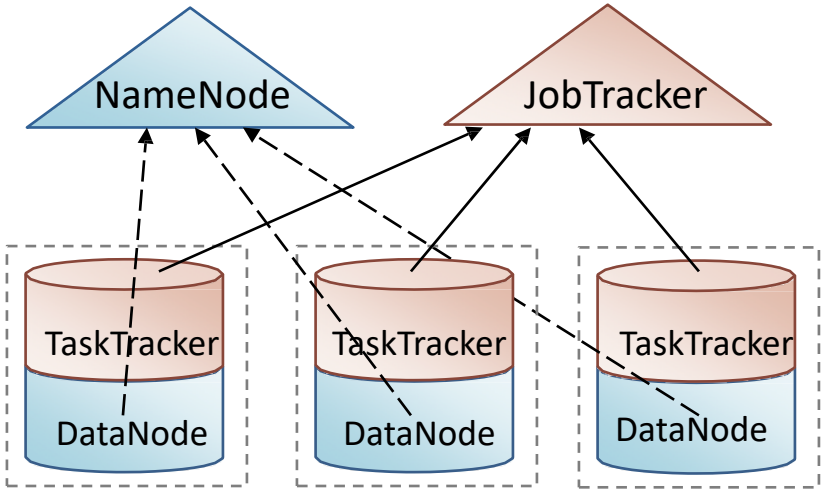


The Hadoop Family

MapReduce	**Distributed computation framework (data processing model and execution environment)**
HDFS	**Distributed file system**
HBase	Distributed, column-oriented database
Hive	Distributed data warehouse
Pig	Higher-level data flow language and parallel execution framework
ZooKeeper	Distributed coordination service
Avro	Data serialization system (RPC and persistent data storage)
Sqoop	Tool for bulk data transfer between structured data stores (e.g., RDBMS) and HDFS
Oozie	Complex job workflow service
Chukwa	System for collecting management data
Mahout	Machine learning and data mining library
BigTop	Packaging and testing

Hadoop Main Cluster Components

- **HDFS daemons**
  - **NameNode**: namespace and block management (~ master in GFS)
  - **DataNodes**: block replica container (~ chunkserver in GFS)
- **MapReduce daemons**
  - **JobTracker**: client communication, job scheduling, resource management, lifecycle coordination (~ master in Google MR)
  - **TaskTrackers**: task execution module (~ worker in Google MR)



HDFS

Hadoop Distributed File System (HDFS)

- Distributed file systems manage the storage across a network of machines.
- Hadoop has a general-purpose file system abstraction (i.e., can integrate with several storage systems such as the local file system, HDFS, Amazon S3, etc.).
- HDFS is Hadoop's flagship file system.

HDFS Design

- **Very large files**
- **Streaming data access**
  - write-once, read-many-times pattern
  - time to read the whole dataset is more important
- **Commodity hardware**
  - fault-tolerance
- **HDFS is not a good fit for**
  - low-latency data access
  - lots of small files
  - multiple writers, arbitrary file modifications

Blocks

- **HDFS files are broken into block-sized chunks**

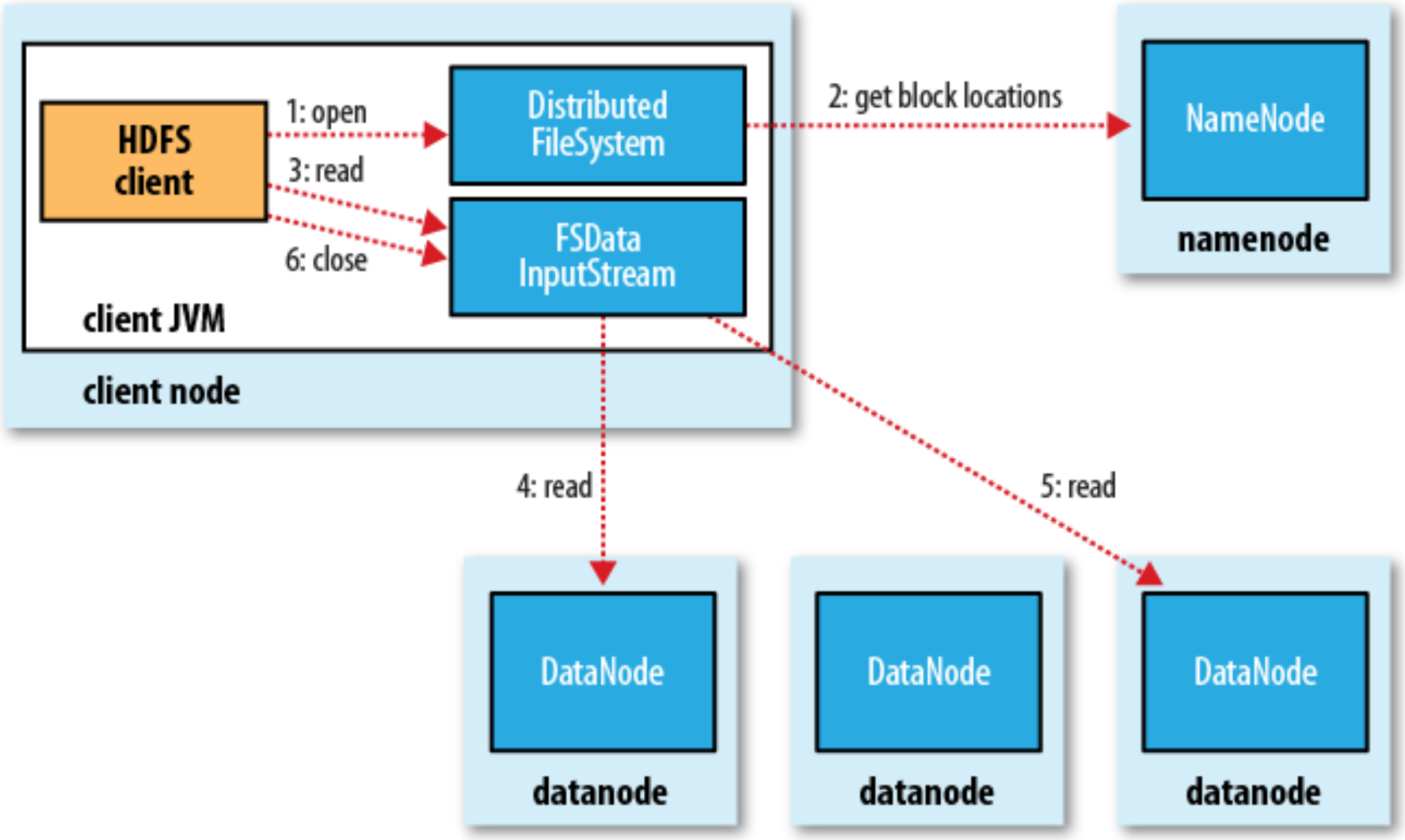
  (64 MB by default)

- **With the (large) block abstraction:**
  - a file can be larger than any single disk in the network
  - storage subsystem is simplified (e.g., metadata bookkeeping)
  - replication for fault-tolerance and availability is facilitated

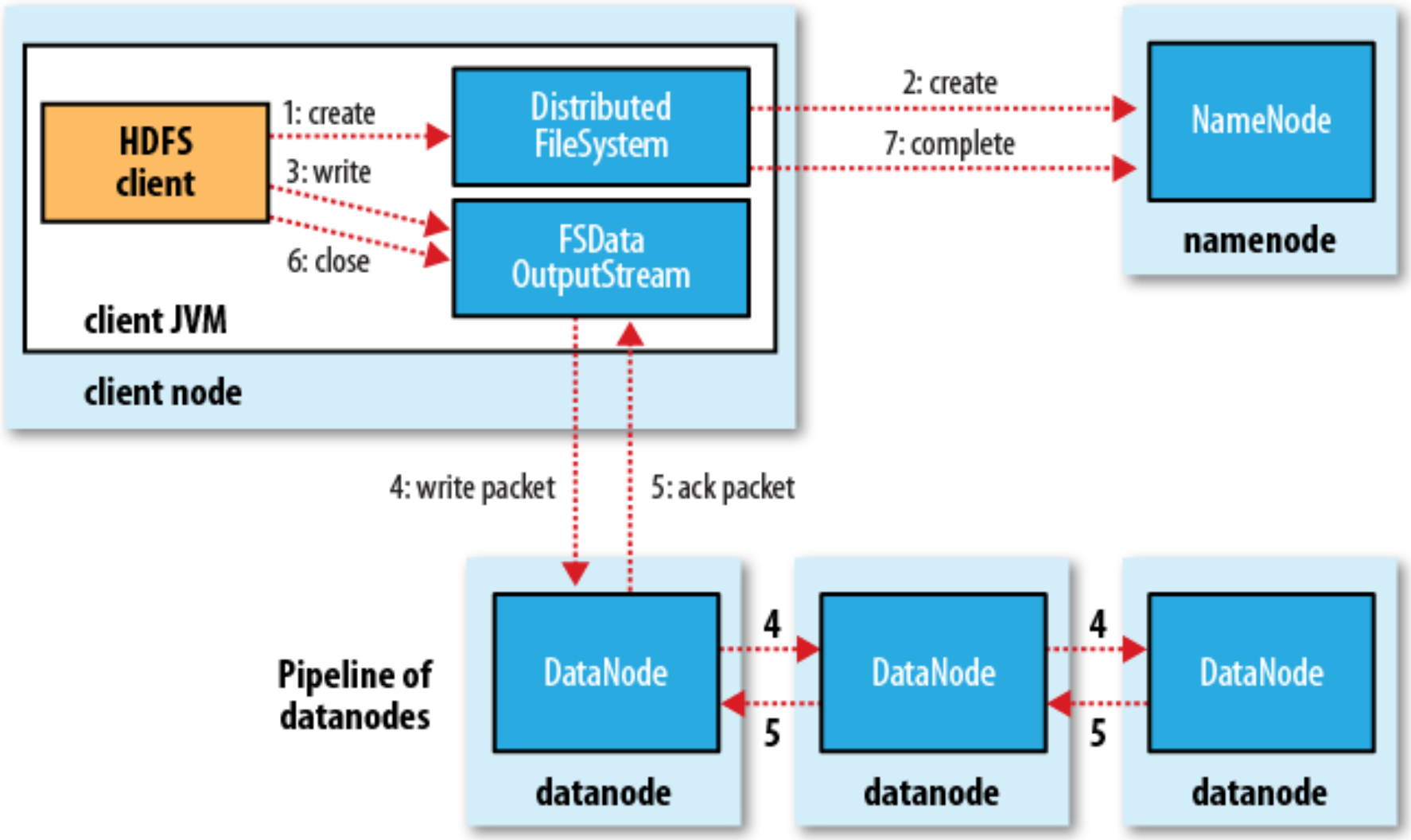
Namenodes and Datanodes

- **Two types of HDFS nodes:**
  - one Namenode (the master)
  - multiple Datanodes (workers)
- **Namenode manages the filesystem namespace.**
  - file system tree and metadata, stored persistently
  - block locations, stored transiently
- **Datanodes store and retrieve data blocks when they are told to by clients or the Namenode.**
- **Datanodes report back to the Namenode periodically with lists of blocks that they are storing.**

Reading from HDFS



Writing to HDFS



Coherency Model

- **Coherency model describes the data visibility of reads and writes for a file.**
- **In HDFS:**
  - The metadata for a newly created file is visible in the file system namespace.
  - The current data block being written is not guaranteed to be visible to other readers.
- **To force all buffers to be synchronized to all relevant datanodes, you can use the sync() method.**
- **Without sync(), you may lose up to a block of (newly written) data in the event of client or system failure.**

Tools for Ingesting Data into HDFS

- **Apache Flume**
  - to move large quantities of streaming data into HDFS (e.g., log data from a system)

- **Apache Sqoop**
  - to perform bulk imports of data into HDFS from structured data stores, such as relational databases