DISTRIBUTED COMPUTING, HADOOP, HDFS

DS8003 – MGT OF BIG DATA AND TOOLS

Ryerson University

Instructor: Kanchana Padmanabhan

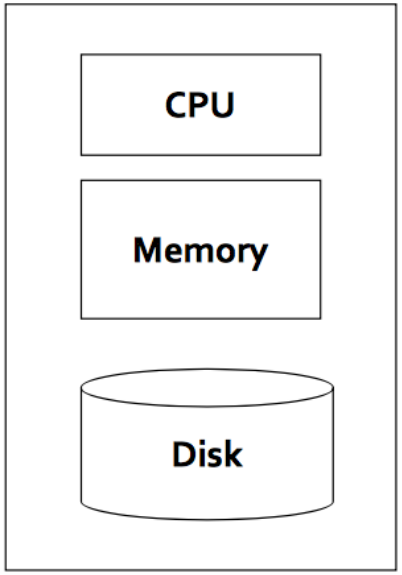
Lecture 2 - Outline

- Distributed Computing
- Hadoop HDFS

Distributed Computing

Single Node Architecture

- Traditionally, computation has been CPU bound
  - Complex computation on small data
- For decades, the primary push is to increase the computing power of a single machine



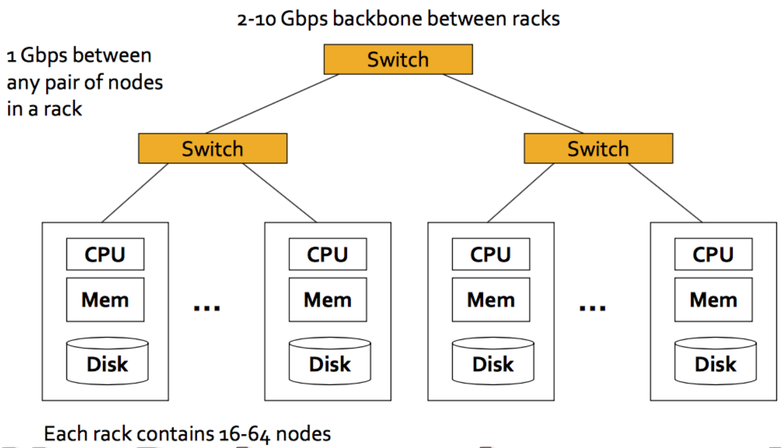
Scale Up vs. Scale Out

- Single Node Architecture
  - Scaling up advantage
    - Programming is easier than distributed computing
    - Faster processing on smaller data
  - Scale up disadvantage
    - Hardware cost
    - Scalability
    - Reliability
- Advantage of scale-out systems
  - Scalability
  - Reliability
  - Cost

Data Becomes the Bottleneck

- Traditional distributed systems don't scale to today's Internet-scale data
- We cannot process the data until we have read it
- Getting data to the computer processor becomes the bottleneck
  - Disk I/O is slow [It takes 4 hours to read a 3TB disk]
  - Network bandwidth is bottleneck
- Solution → moving computation to the data!

• ***Internet***	◦ ***2.5 exabytes ($2.5x10^{18}$) per day – 2012*** ◦ ***2.3 zettabytes ($2.3x10^{21}$) per day - 2014***
• ***Facebook***	◦ ***500+ terabytes per day*** ◦ ***100+ petabytes in a single Hadoop cluster***



MapReduce to the rescue!

Disk Capacity

- While disk capacity increased, the cost has decreased significantly

Year	Capacity (GB)	Cost per GB (USD)
1997	2.1	$157
2004	200	$1.05
2012	3,000	$0.05

Disk Performance - Hadoop

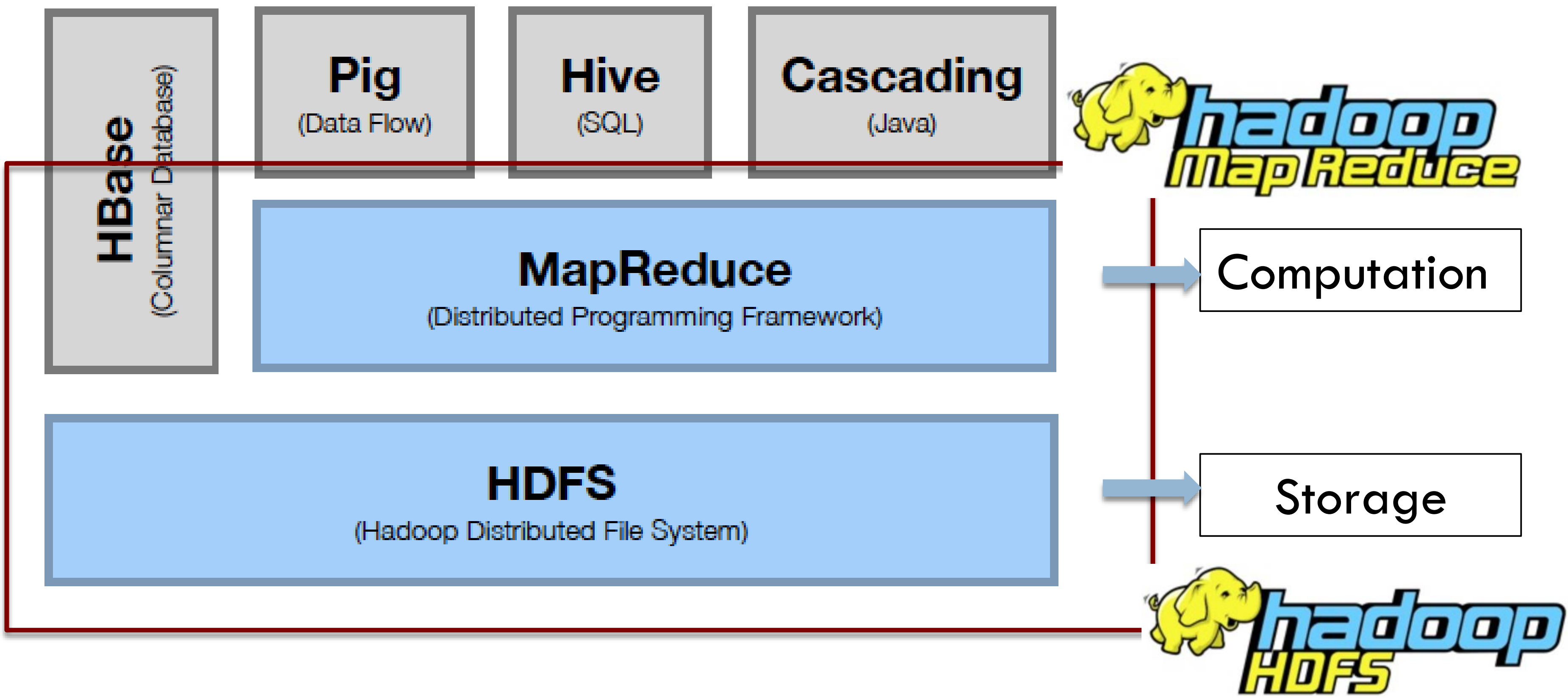
Year	Capacity (GB)	Transfer Rate (MB/s)	Disk Read Time
1997	2.1	16.6	126 seconds
2004	200	56.5	59 minutes
2012	3,000	210	3 hours, 58 minutes

- Hadoop → Reading 1000 disks in parallel
  - 3TB in 15 seconds

Hadoop

Software Library

Hadoop CORE


HBase
(Columnar Database)
Pig
(Data Flow)
Hive
(SQL)
Cascading
(Java)
MapReduce
(Distributed Programming Framework)
HDFS
(Hadoop Distributed File System)
hadoop
Map Reduce
Computation
Storage
hadoop
HDFS

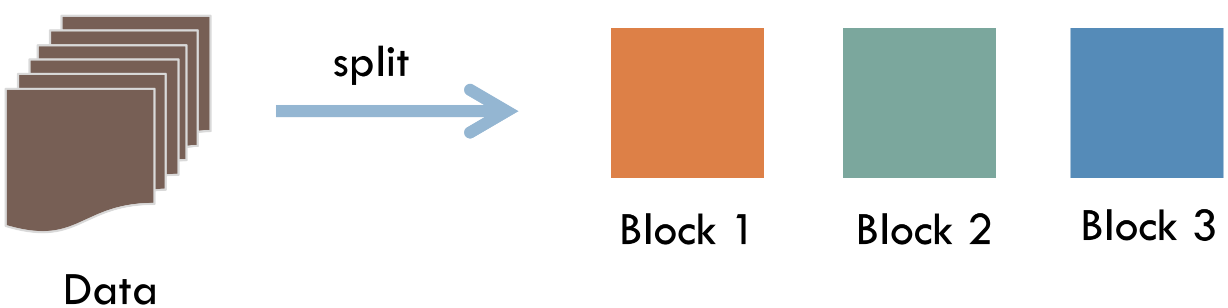
HDFS - Hadoop Distributed File System

HDFS

- A distributed file system that runs on large clusters of commodity machines
- Based on Google GFS paper
- Provides redundant storage for massive datasets

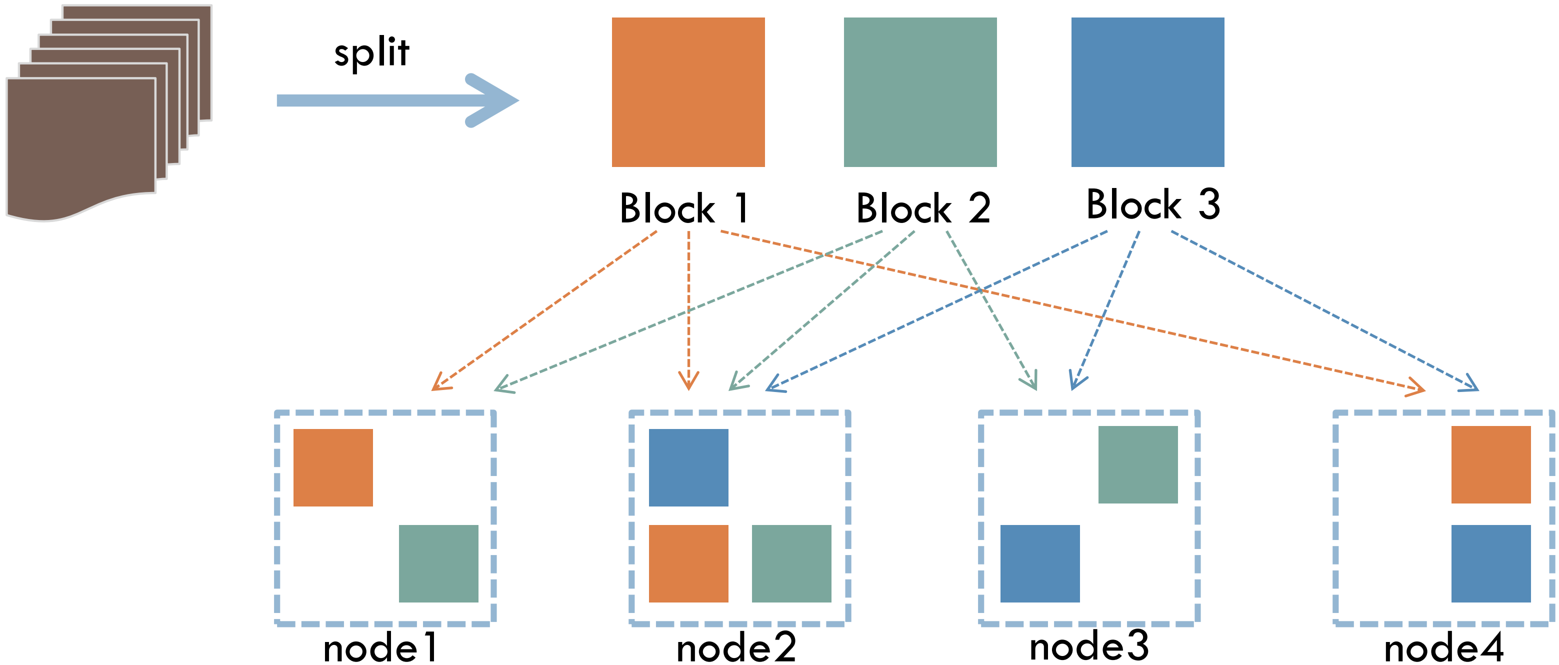
HDFS - Blocks

- When a file is added to HDFS, it is split into blocks
  - 64M by default,
  - Can be configured to 128M, 256M, 1G, etc.
  - Should not be very small - Map tasks depends on number of number of blocks
- Why blocks?
  - Replication (fault tolerance)
  - Large file gets chunked and distributed easily (There could be files that will not fit on the disk of a single machine)
  - Data-local distributed computation (MapReduce)



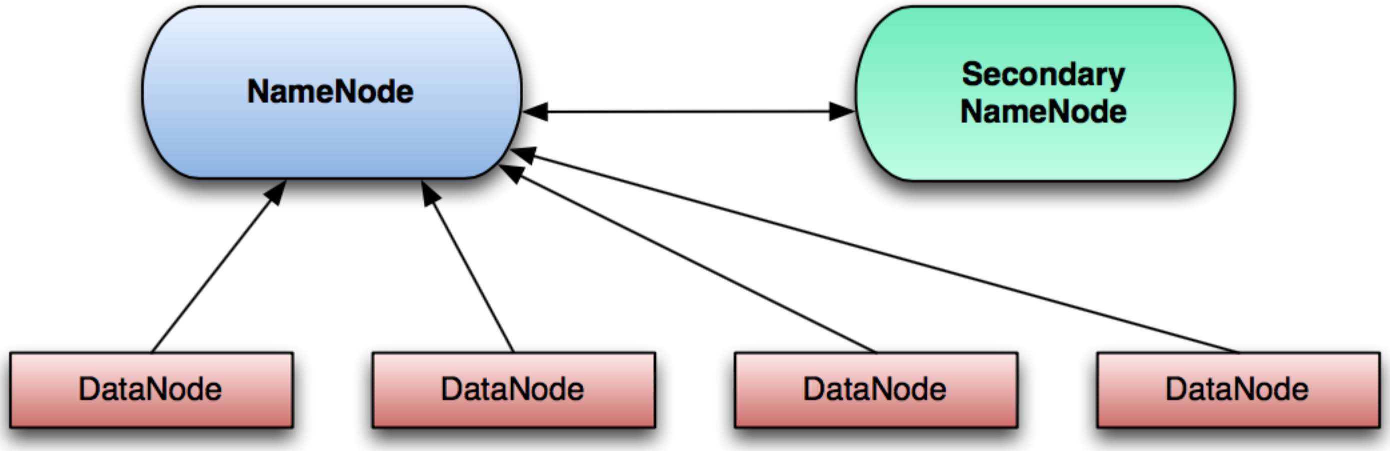
HDFS - Replication

- The blocks are replicated to nodes throughout the cluster
  - Based on the *replication factor* (3 by default)
- Replication increases reliability and performance
  - Reliability: can tolerate data loss
  - Performance: more opportunities for data locality



HDFS Architecture

- There're 3 daemons in "classical" HDFS
  - NameNode (master)
  - Secondary NameNode (master)
  - DataNode (slave)

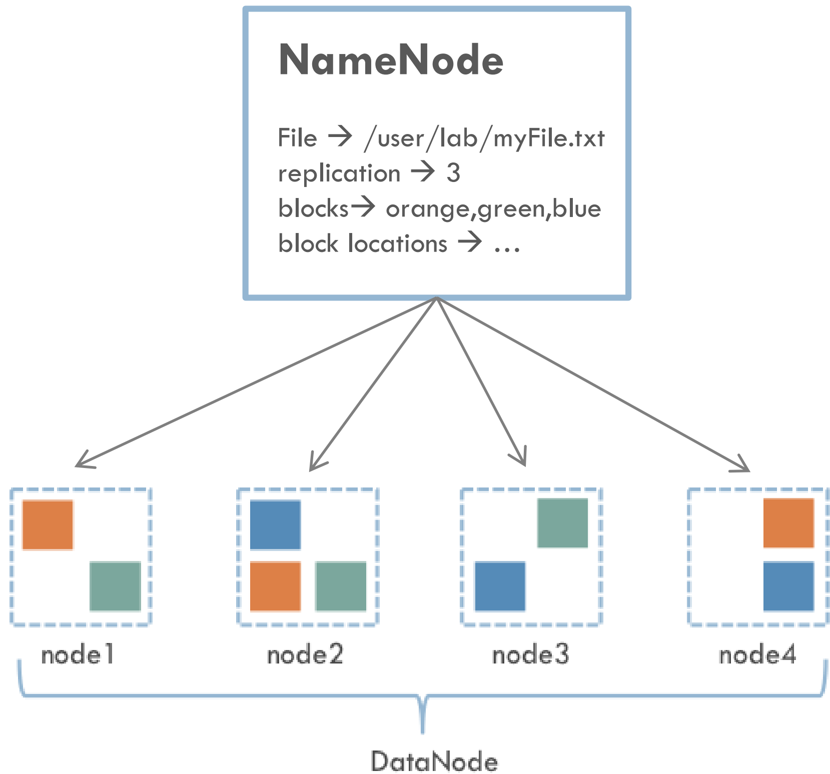


HDFS – NameNode and DataNode

- NameNode → master
  - Maintains filesystem tree and metadata in tree
  - Knows where all the blocks are stored for a file
- DataNode → worker
  - Store and retrieve data blocks
  - Report periodically with lists of blocks they stored

NameNode (master)

- The NameNode stores all metadata
  - Information about file locations in HDFS
  - Information about file ownership and permissions
  - Name of the individual blocks
  - Locations of the blocks
- Metadata is stored on disk and read into memory when the NameNode daemon starts up
- Changes/Edits to the files are written to the logs

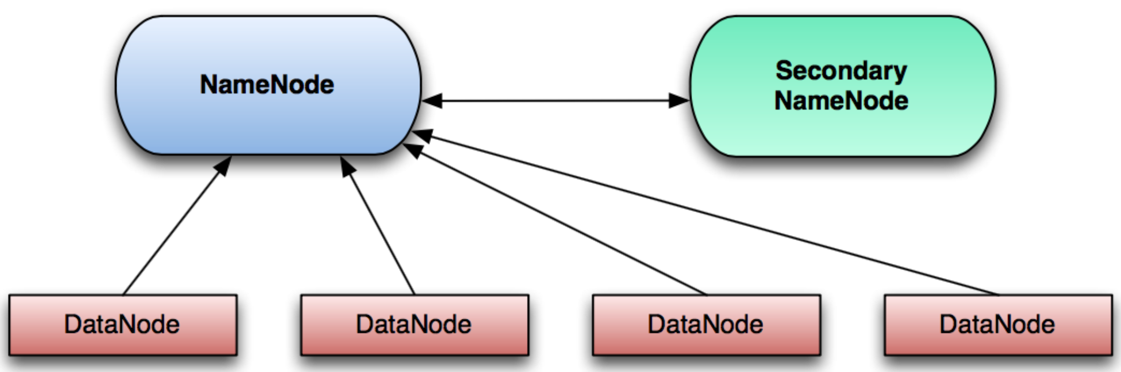


DataNode (slave)

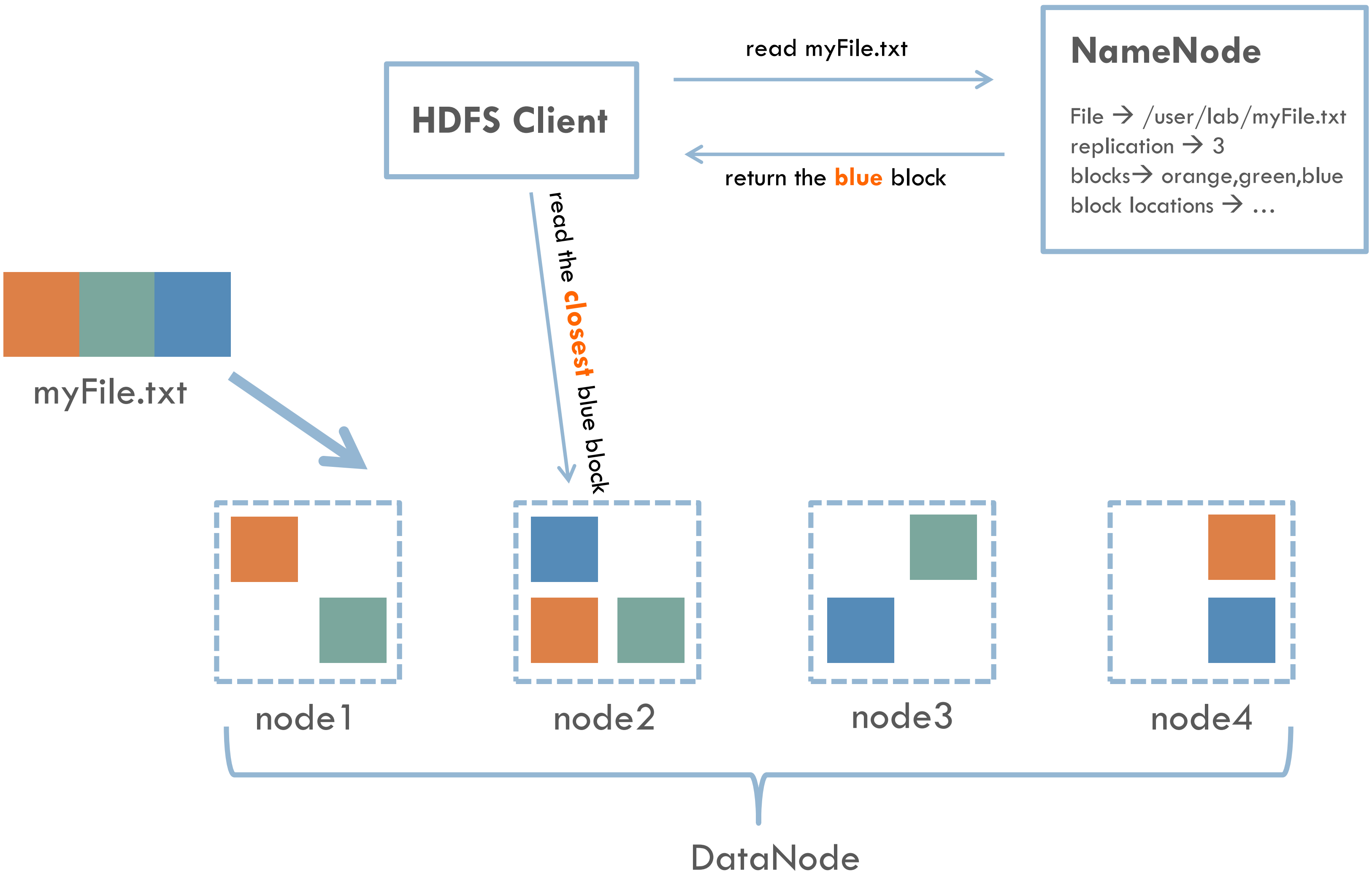
- Actual files/data are chunked into blocks and stored on the data nodes
  - Block name “blk_xxxxx” maps to a block name in the NameNode
  - The location of the blocks are stored in NameNode instead
- Each block is replicated to different nodes for redundancy
- The DataNode daemon controls access to the blocks and communicates with the NameNode

Secondary NameNode (master)

- The Secondary NameNode is not a backup for the NameNode
  - It provides memory-intensive administrative functions for the NameNode
    - Secondary NameNode periodically combines a prior snapshot of the file system metadata and edit logs into a new snapshot
    - It then transmits the new snapshot back to the NameNode

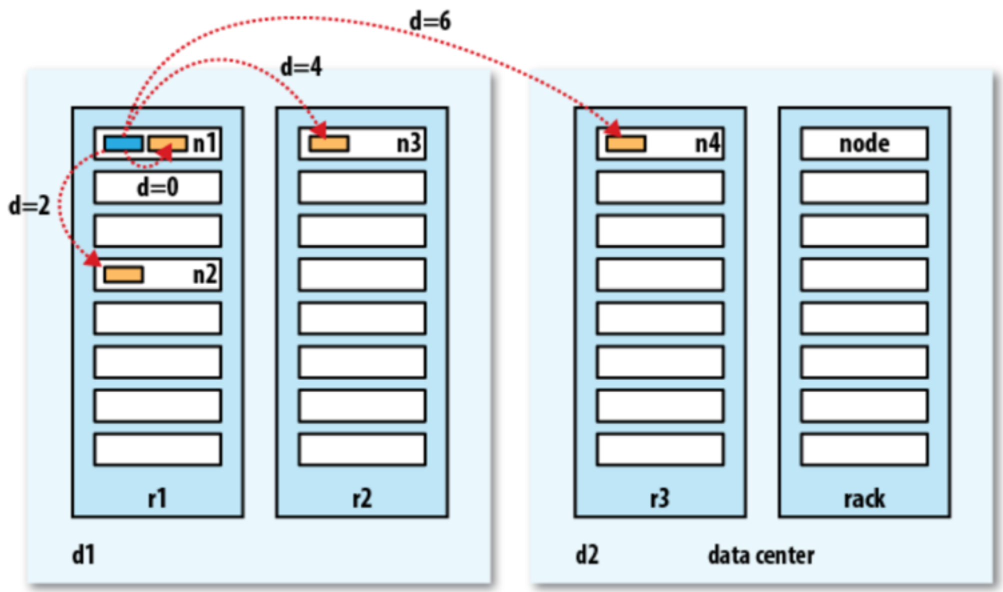


Anatomy of a File Read

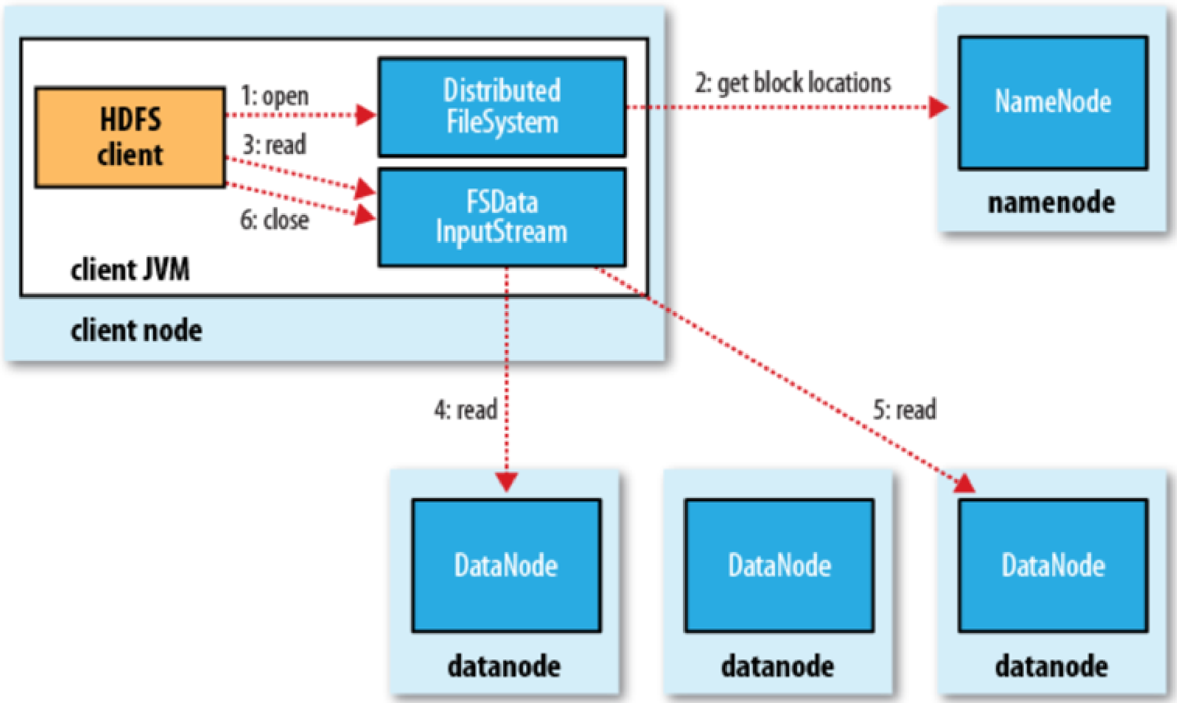


Network Distance in Hadoop

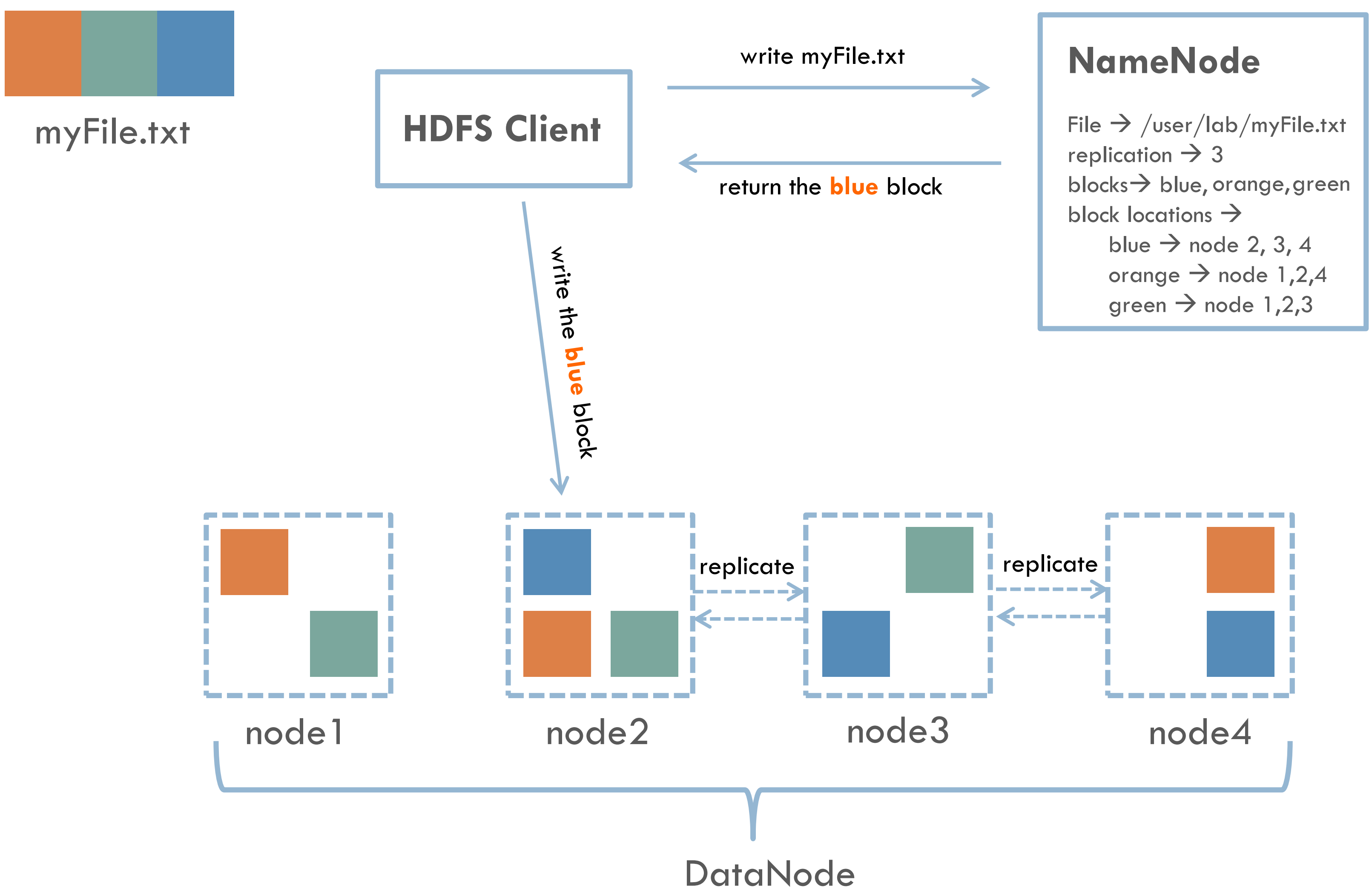
- How does Hadoop decide which block of data is closest?



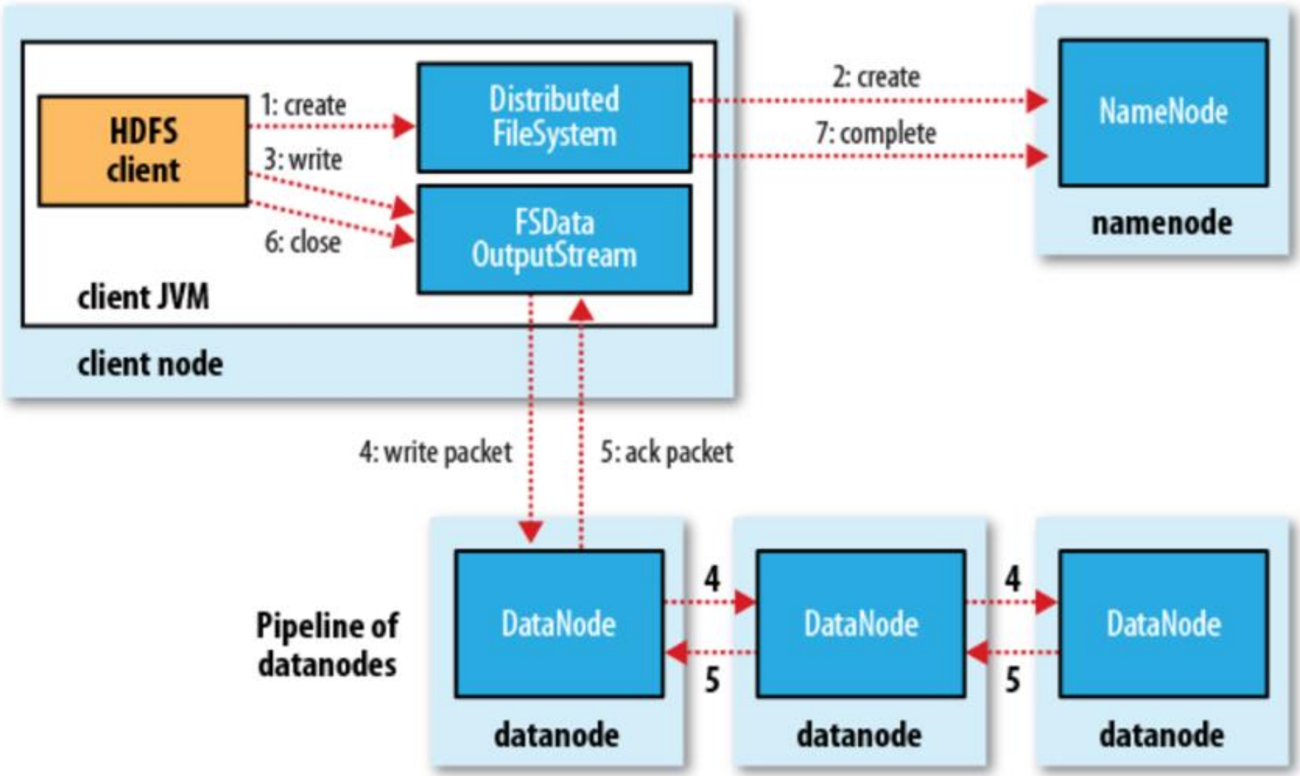
Anatomy of a File Read



Anatomy of a File Write



Anatomy of a File Write

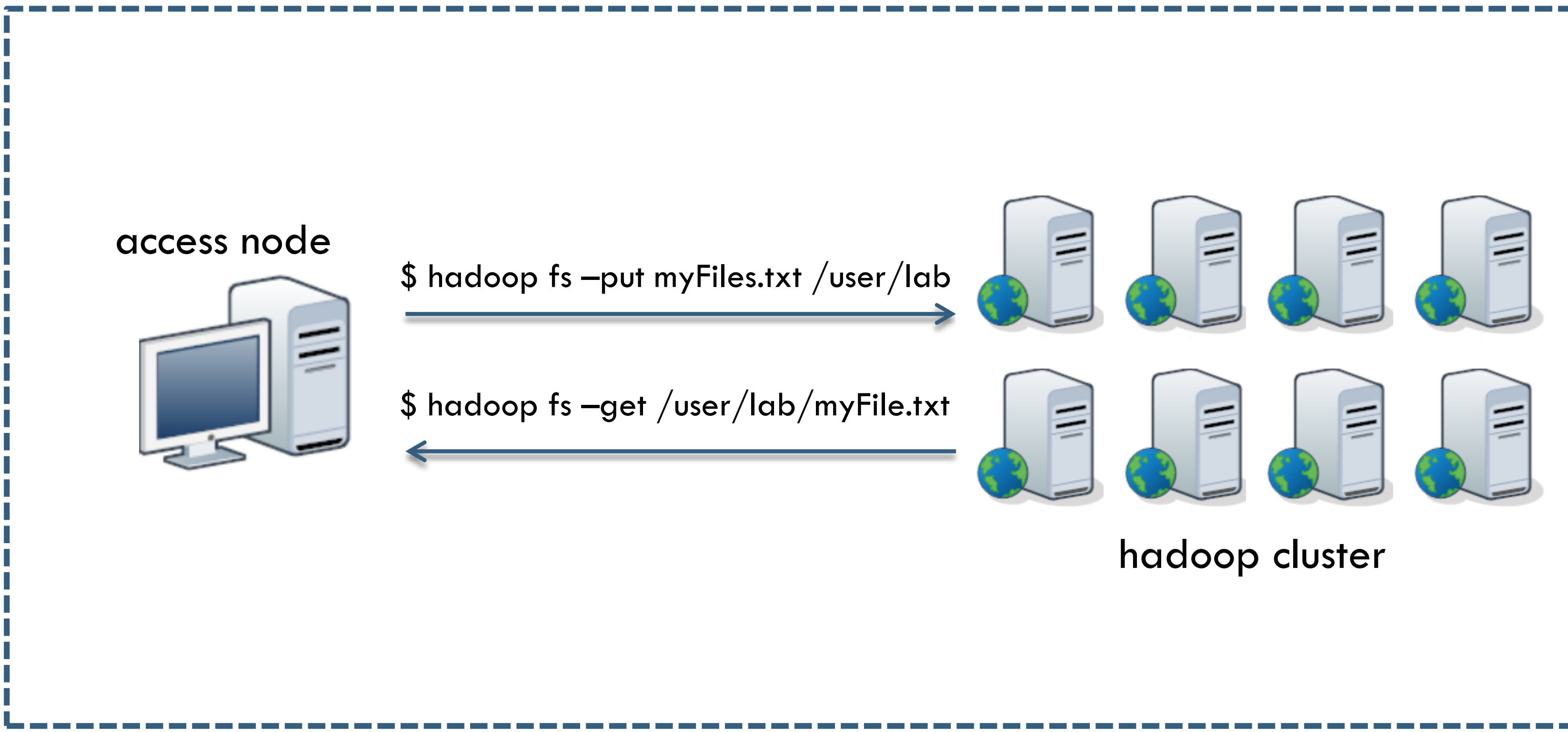


Block Replication Strategy

- Hadoop's default strategy is to place the first replica on the same node as the client
- The second replica is placed on a different rack from the first (off-rack), chosen at random.
- The third replica is placed on the same rack as the second, but on a different node chosen at random.

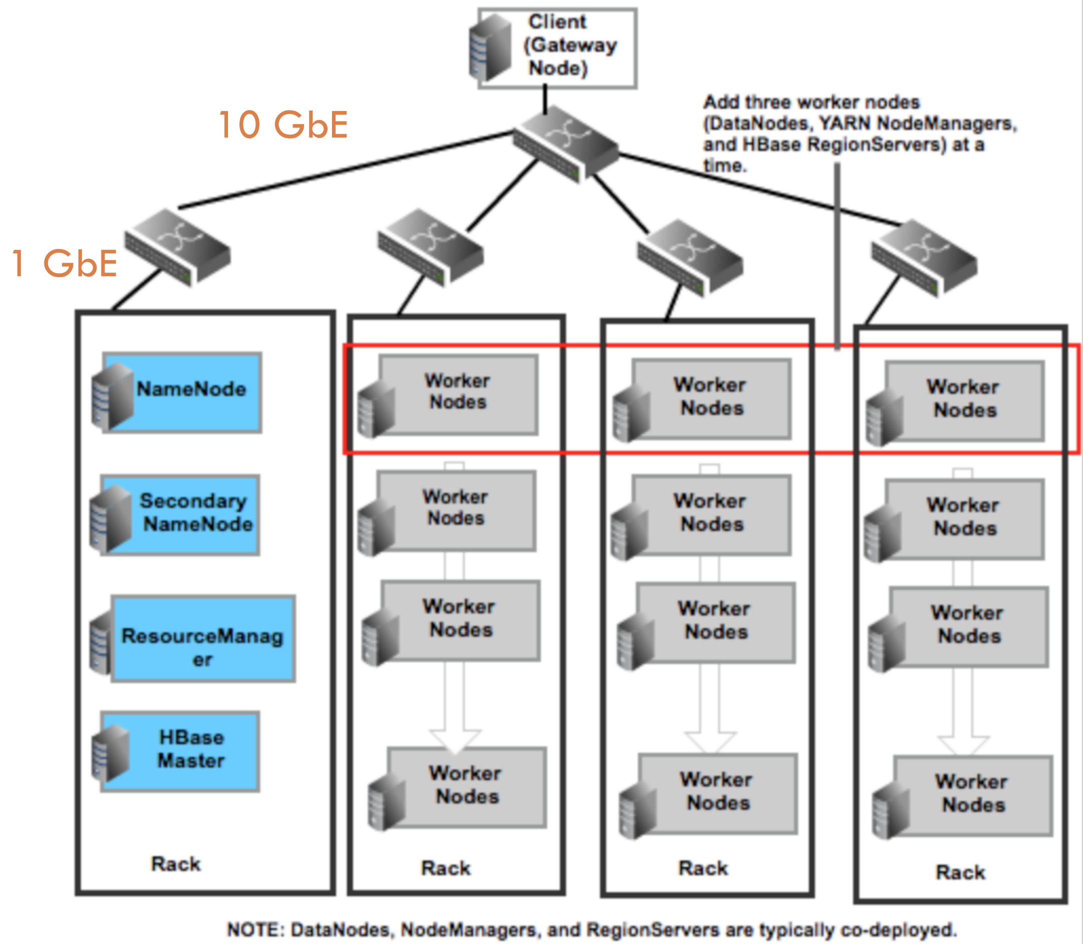
HDFS – CLI (command line)

- Users typically access HDFS via *hadoop fs* command



Modern Distributed Computing Cluster

- Cluster architecture
  - A medium-to -large Hadoop cluster consists of a two-level or three-level architecture built with rack-mounted servers. Each rack of servers is interconnected using a 1 Gigabyte Ethernet switch. Each rack-level switch is connected to a cluster-level switch (which is typically a larger port-density 10GbE switch).



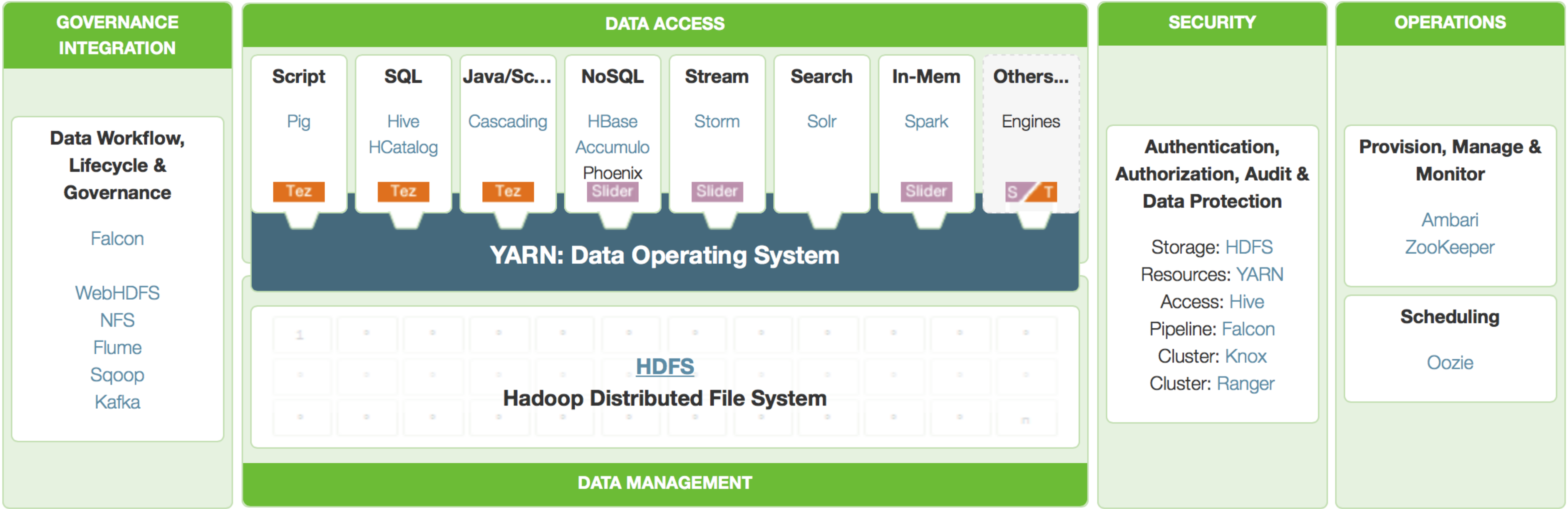
Stunning Photos Of Google's Massive Data Centers: http://www.forbes.com/pictures/edej45emjgl/up-above-the-massive-floor/

An Ideal Distributed System

- Handles failures well
  - (automatic) job should complete without manual intervention
  - (transparent) tasks assigned to a failed component are picked up by others
  - (graceful) failure only results in a proportional loss of load capacity
  - (recoverable) the capacity is reclaimed when the component is later replaced
  - (consistent) failure does not produce corruption or invalid results
- Scalability
  - Linear horizontal scalability (scale-out)
    - Adding new nodes should increase capacity proportionally
    - Shared nothing architecture
    - At a reasonable cost (commodity machines)
- Simple programming model
  - Programmers only focus on key functions while not worrying about distribution, parallelism, data transfer, failures etc.
  - Support many languages

Hadoop Ecosystem

- Data analysis
  - Hive, Pig, Spark
- Machine Learning
  - Mahout, Spark (Mllib)
- Graph processing
  - Giraph, Spark (GraphX)
- Database Integration
  - Sqoop
- Scheduling & Workflow
  - Oozie
- Cluster management
  - Ambari
- Search
  - Solr
- NoSQL
  - Hbase, Cassandra
- Stream Processing
  - Storm



Required Reading

- HDFS Comics
  - http://bigdatahandler.com/2013/10/30/understanding-hdfs-architecture-in-comic-format-2/
- More info on HDFS
  - http://www3.nd.edu/~dthain/courses/cse40822/fall2014/slides/cse40822-hadoop-lec1.pptx