Brief on Hadoop

What is Hadoop?

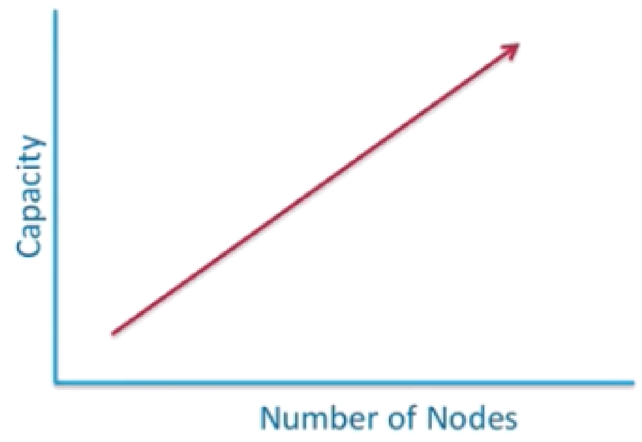
- **Hadoop is a distributed data storage and processing platform**
  - Stores massive amounts of data in a very resilient way
  - Handles low-level distributed system details and enables your developers to focus on the business problems
- **Tools built around Hadoop (the 'Hadoop ecosystem') can be configured/ extended to handle many different tasks**
  - Extract Transform Load (ETL)
  - BI environment
  - General data storage
  - Predictive analytics
  - Statistical analysis
  - Machine learning
  - ...

Core Hadoop is a File System and a Processing Framework

- **The Hadoop Distributed File System (HDFS)**
  - Any type of file can be stored in HDFS
  - Data is split into chunks and replicated as it is written
    - Provides resiliency and high availability
    - Handled automatically by Hadoop
- **YARN (Yet Another Resource Negotiator)**
  - Manages the processing resources of the Hadoop cluster
  - Schedules jobs
  - Runs processing frameworks
- **MapReduce**
  - A distributed processing framework

Hadoop is Scalable

- **Adding nodes (machines) adds capacity proportionally**
- **Increasing load results in a graceful decline in performance**
  - Not failure of the system



Hadoop is Fault Tolerance

- **Node failure is inevitable**
- **What happens?**
  - System continues to function
  - Master re-assigns work to a different node
  - Data replication means there is no loss of data
  - Nodes which recover rejoin the cluster automatically

The Hadoop Ecosystem (2)

- **Examples of Hadoop ecosystem projects (all included in CDH):**

Project	What does it do?
Spark	In-memory and streaming processing framework
HBase	NoSQL database built on HDFS
Hive	SQL processing engine designed for batch workloads
Impala	SQL query engine designed for BI workloads
Parquet	Very efficient columnar data storage format
Sqoop	Data movement to/from RDBMSs
Flume, Kafka	Streaming data ingestion
Solr	Powerful text search functionality
Hue	Web-based user interface for Hadoop
Sentry	Authorization tool, providing security for Hadoop

Why Do You Need Hadoop? (1)

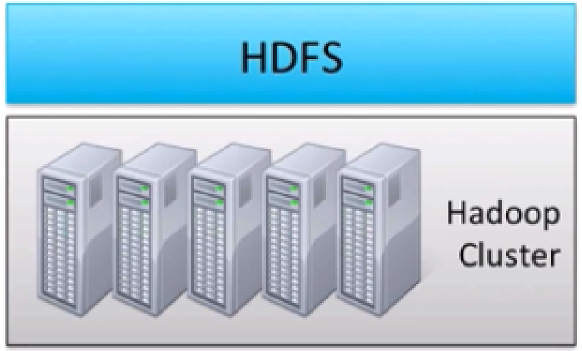
- **More data is coming**
  - Internet of things
  - Sensor data
  - Streaming
- **More data means bigger questions**
- **More data means better answers**
- **Hadoop easily scales to store and handle all of your data**
- **Hadoop is cost-effective**
  - Typically provides a significant cost-per-terabyte saving over traditional, legacy systems
- **Hadoop integrates with your existing datacenter components**

The Hadoop Distributed File System (HDFS)

- **HDFS is the storage layer for Hadoop**
- **A filesystem which can store any type of data**
- **Provides inexpensive and reliable storage for massive amounts of data**
  - Data is replicated across computers
- **HDFS performs best with a 'modest' number of large files**
  - Millions, rather than billions, of files
  - Each file typically 100MB or more
- **File in HDFS are 'write once'**
  - Appends are permitted
  - No random writes are allowed

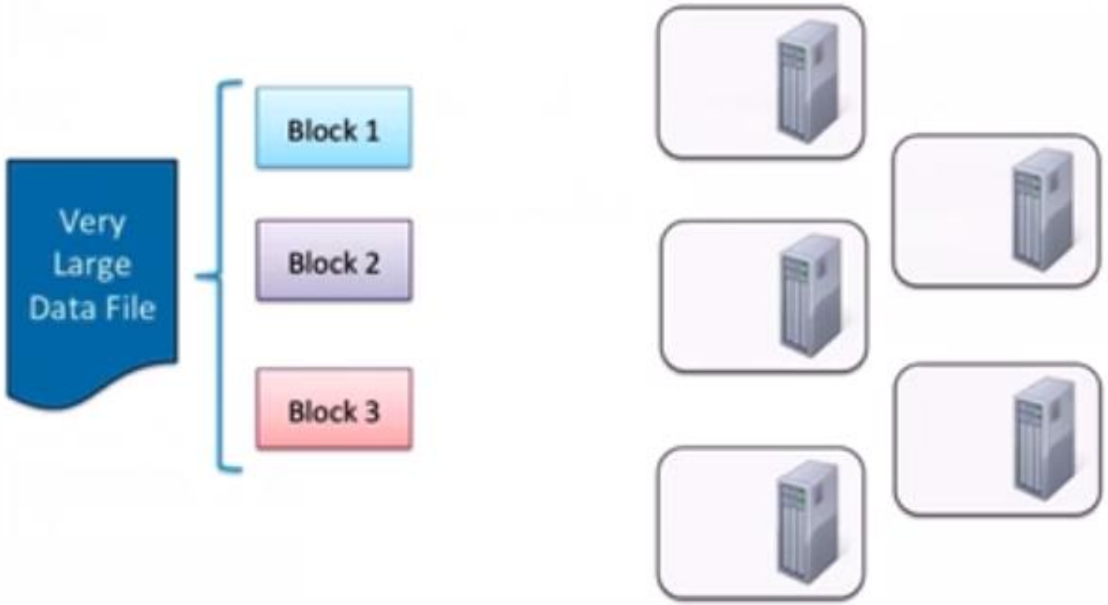
HDFS Basic Concepts

- HDFS is a filesystem written in Java
- Sits on top of a native filesystem
- Scalable
- Fault tolerant
- Supports efficient processing with MapReduce, Spark, and other frameworks



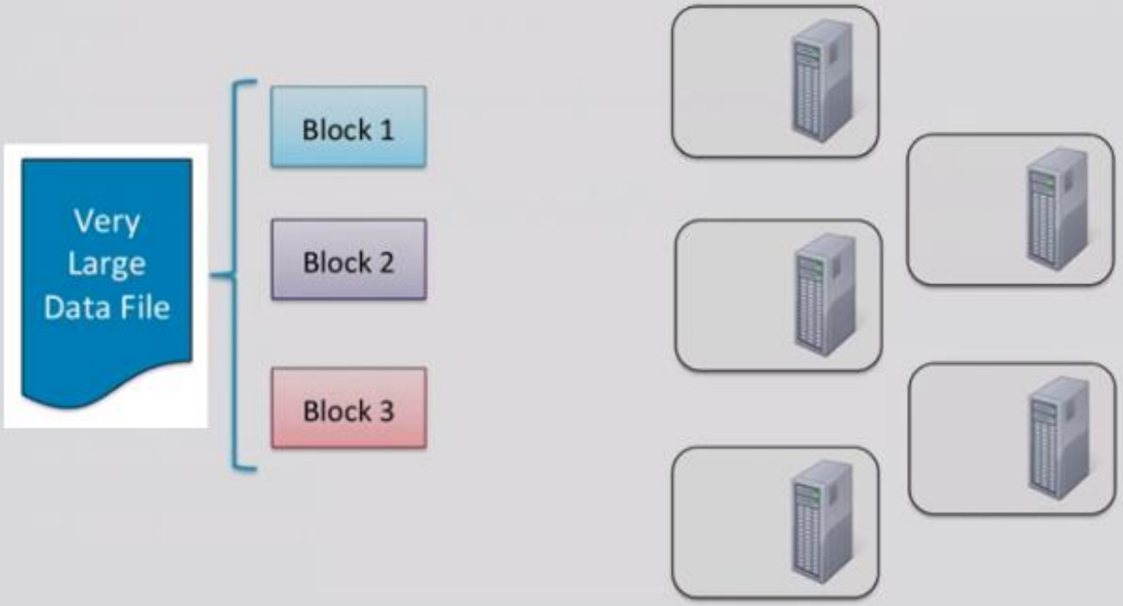
How Files are Stored (1)

- **Data files are split into blocks and distributed to data nodes**



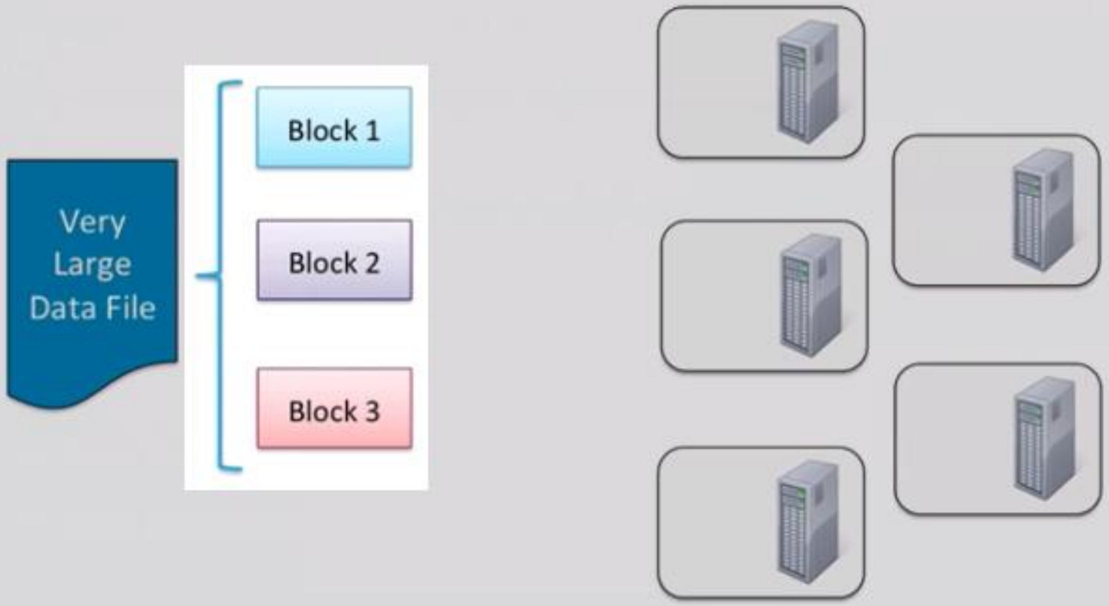
How Files are Stored (1)

- **Data files are split into blocks and distributed to data nodes**



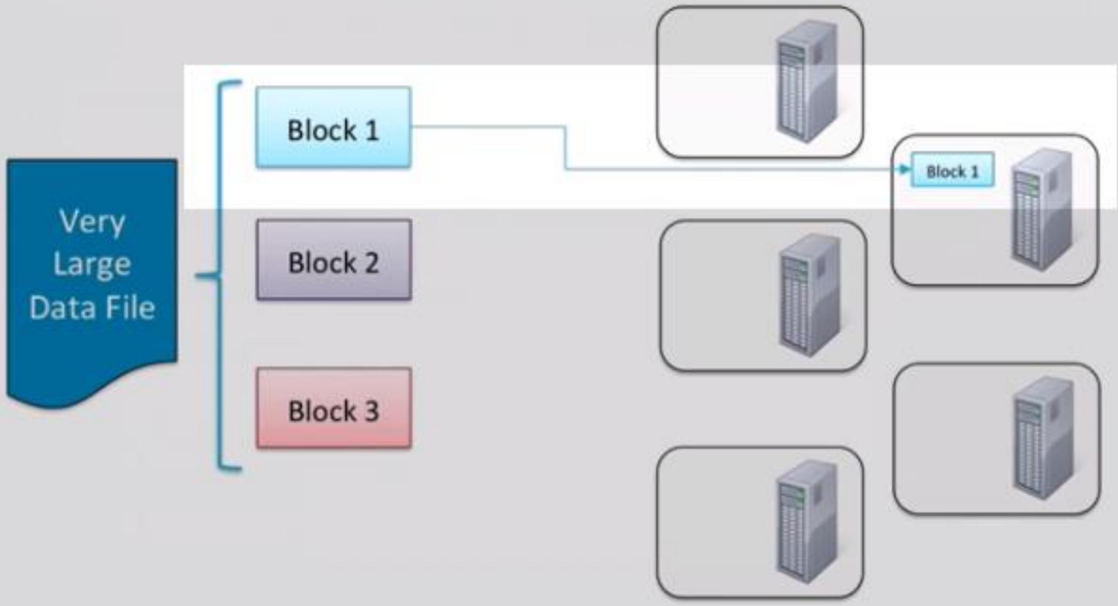
How Files are Stored (1)

- **Data files are split into blocks and distributed to data nodes**



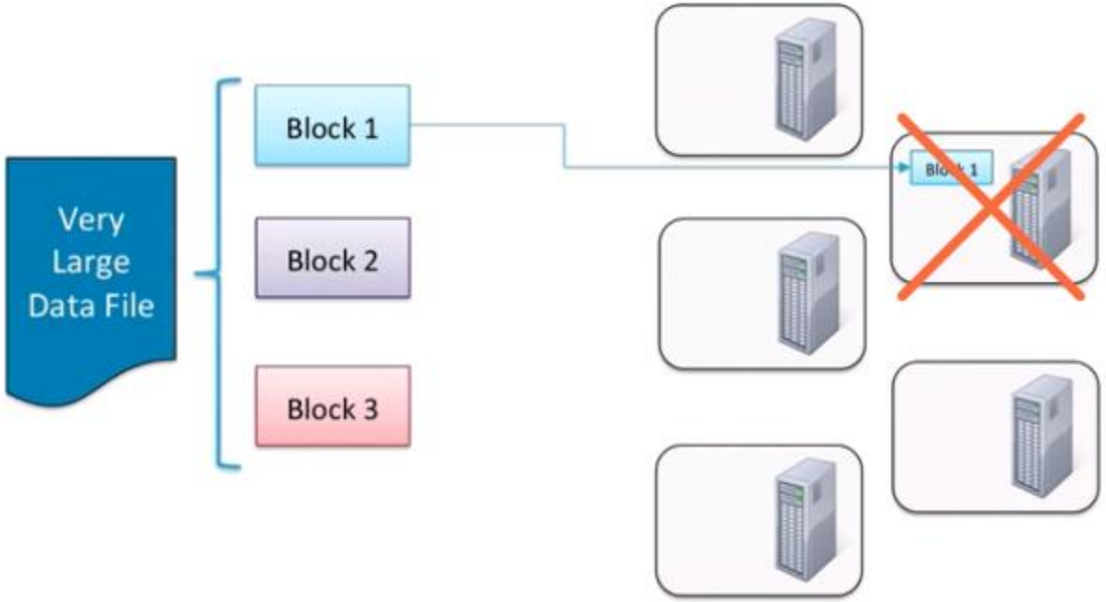
How Files are Stored (2)

- **Data files are split into blocks and distributed to data nodes**



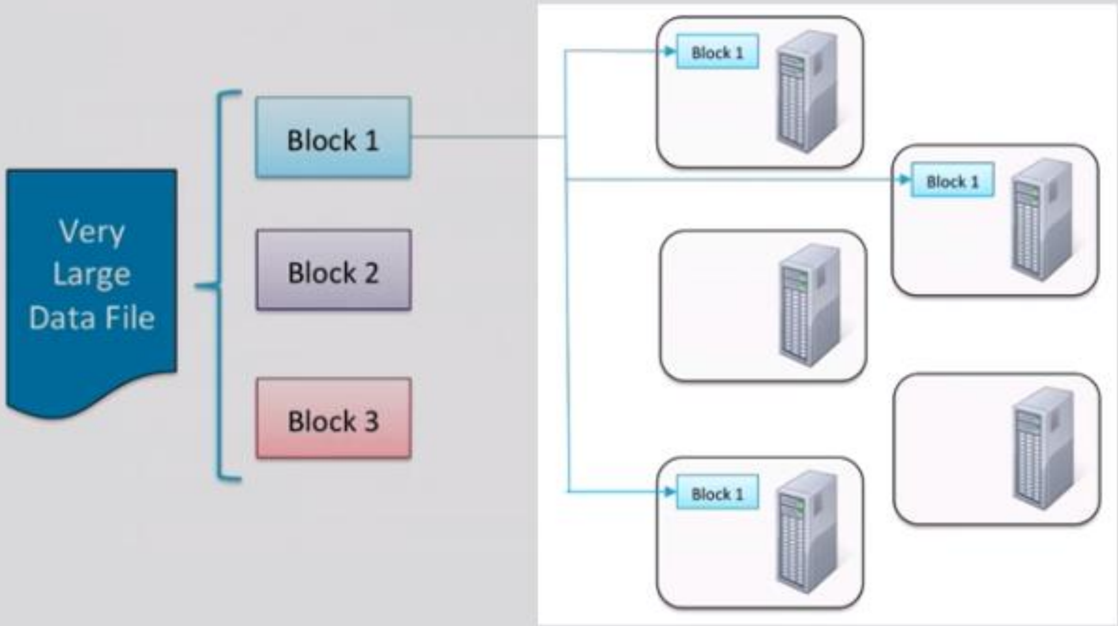
How Files are Stored (2)

- **Data files are split into blocks and distributed to data nodes**



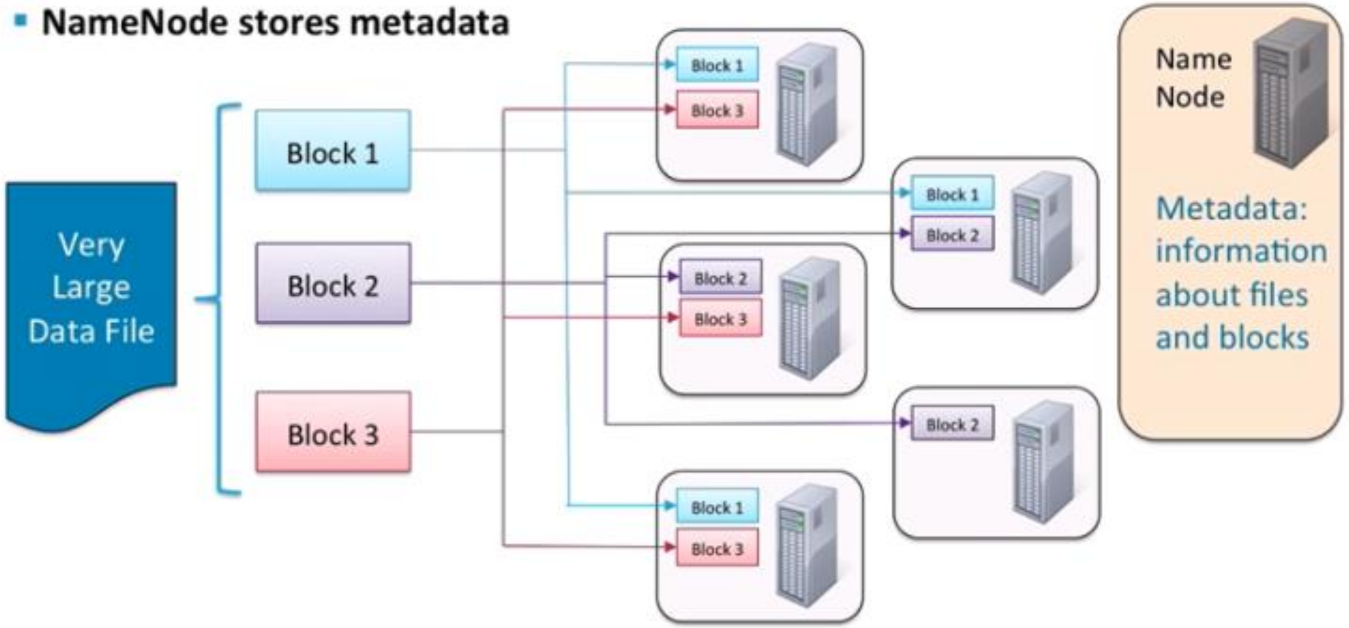
How Files are Stored (3)

- **Data files are split into blocks and distributed to data nodes**
- **Each block is replicated on multiple nodes (default: 3x replication)**



How Files are Stored (5)

- **Data files are split into blocks and distributed to data nodes**
- **Each block is replicated on multiple nodes (default: three-fold replication)**
- **NameNode stores metadata**

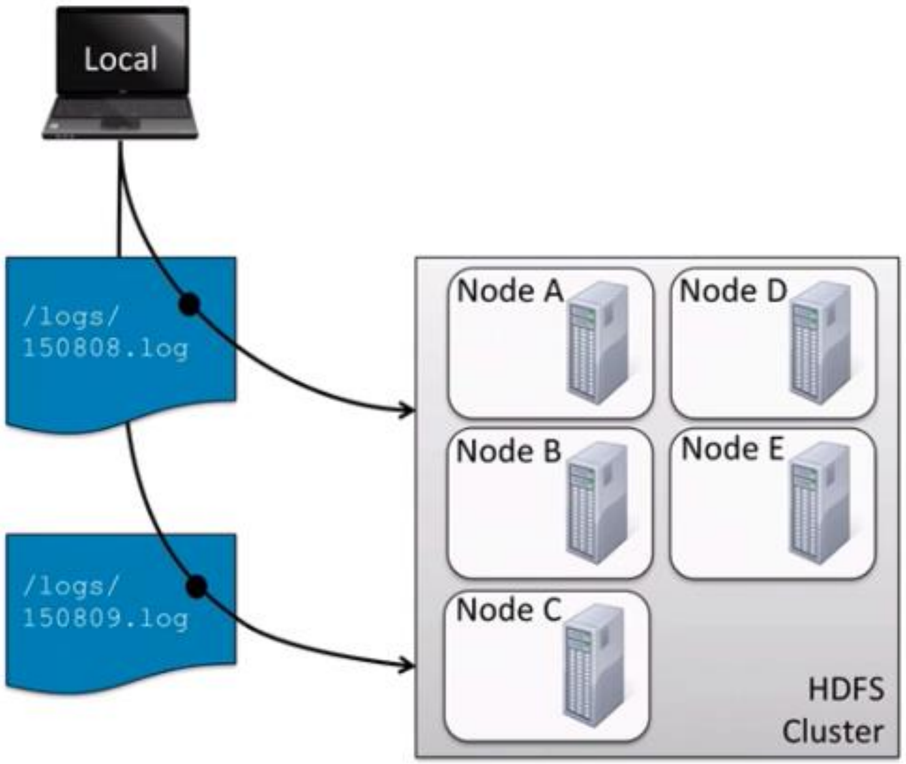


Getting Data In and Out of HDFS

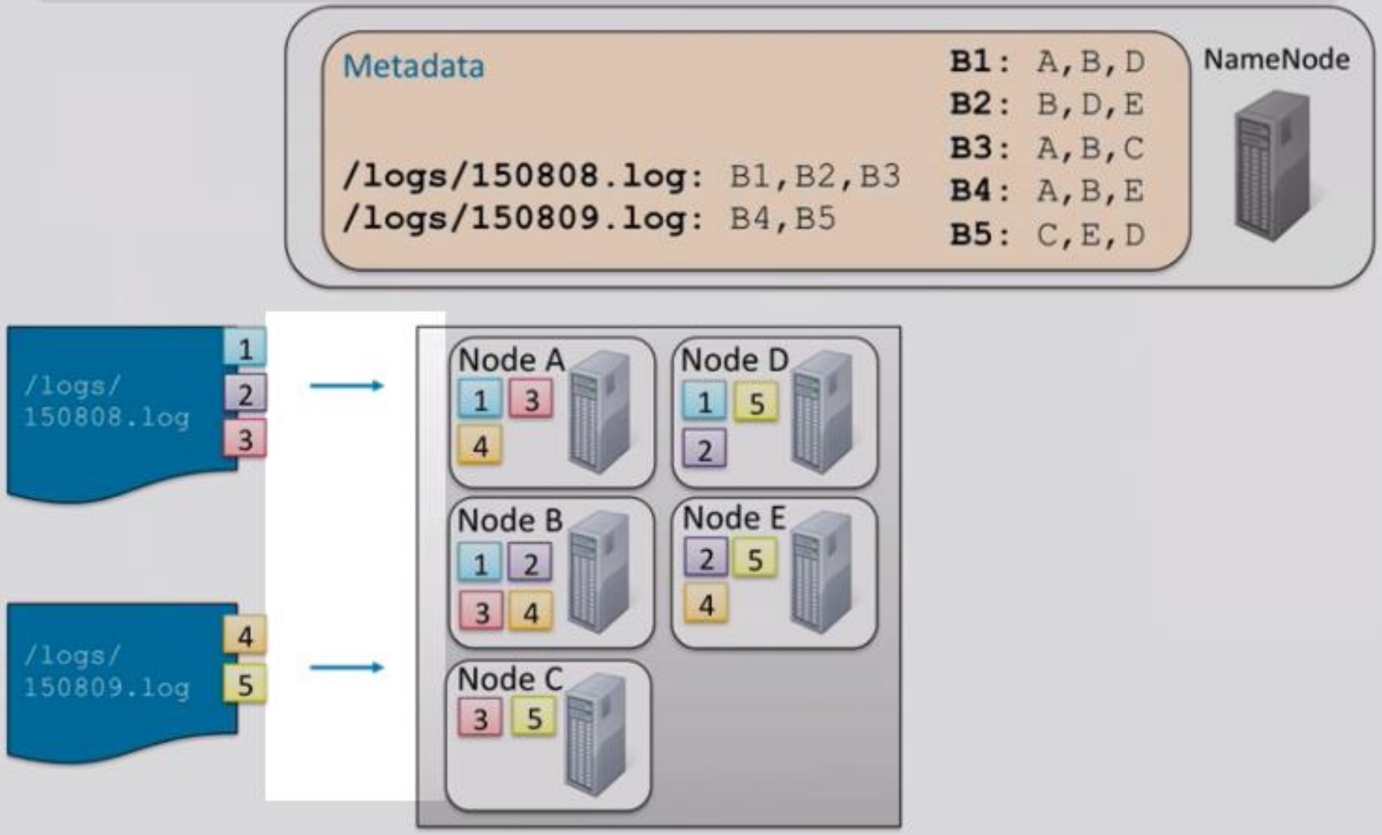
- **Hadoop**
  - Copies data between client (local) and HDFS (cluster)
  - API or command line
- **Ecosystem Projects**
  - Flume
    - Collects data from network sources (e.g., websites, system logs)
  - Sqoop
    - Transfers data between HDFS and RDBMSs
- **Business Intelligence Tools**

RDBMS: Relational Database Management System

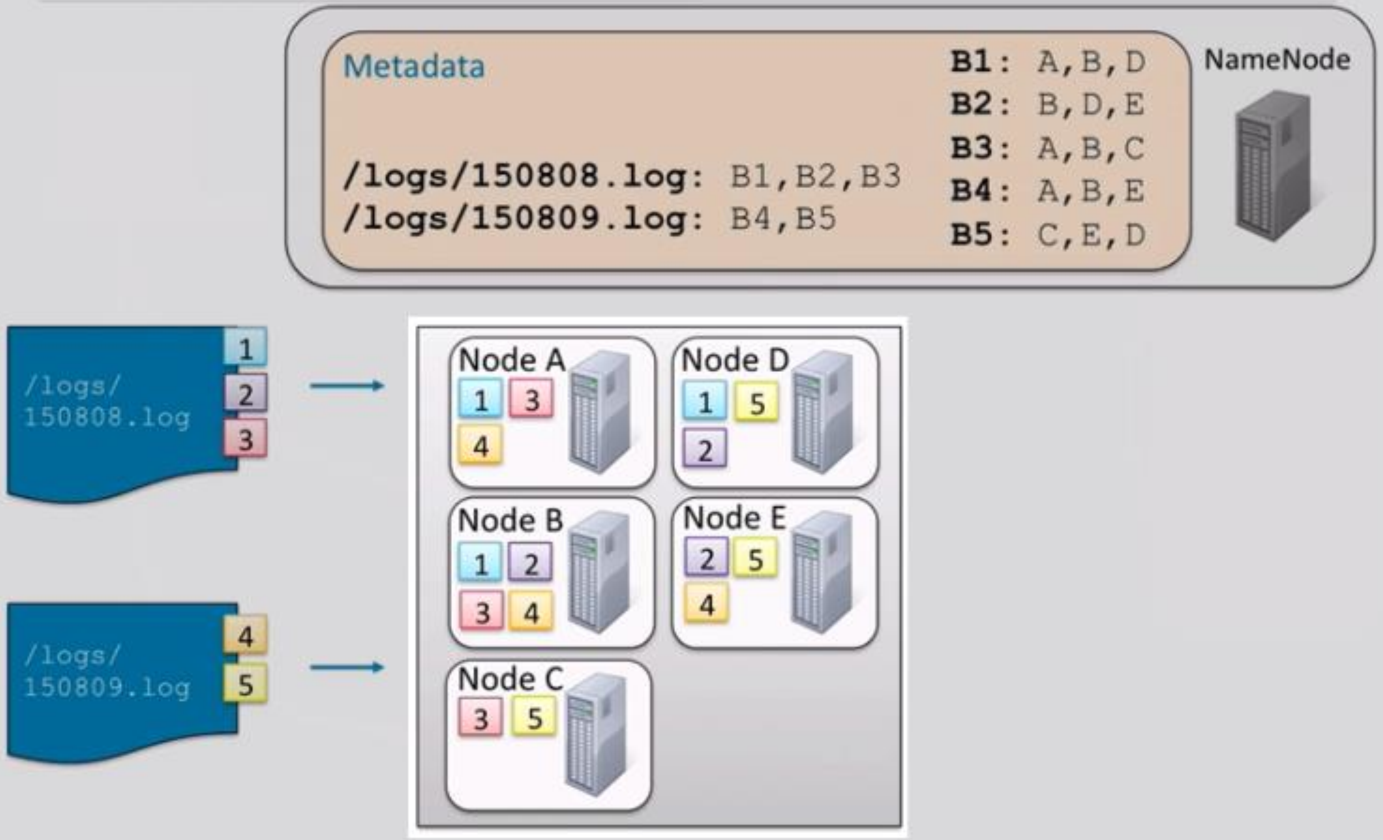
Example: Storing and Retrieving Files (1)

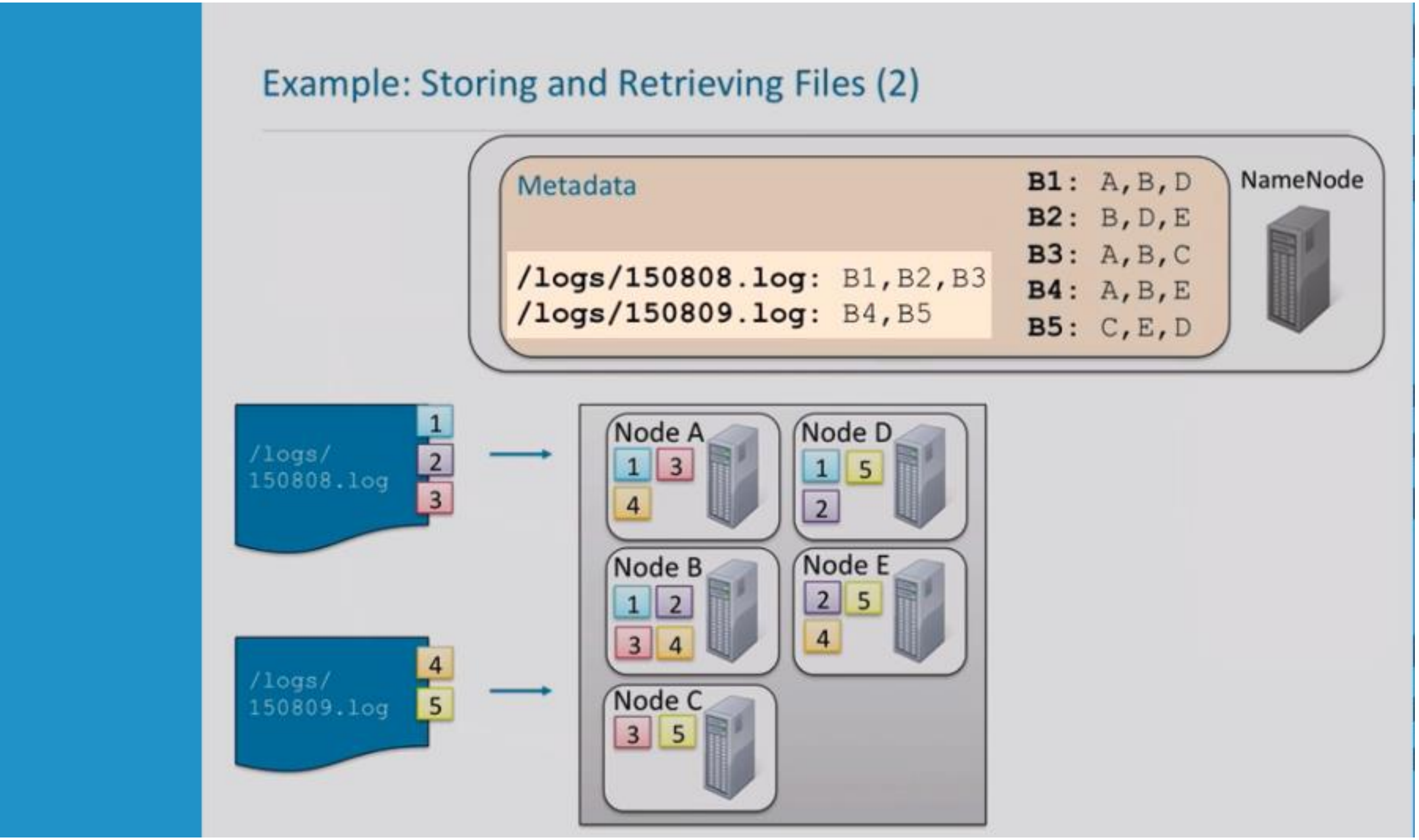


Example: Storing and Retrieving Files (2)

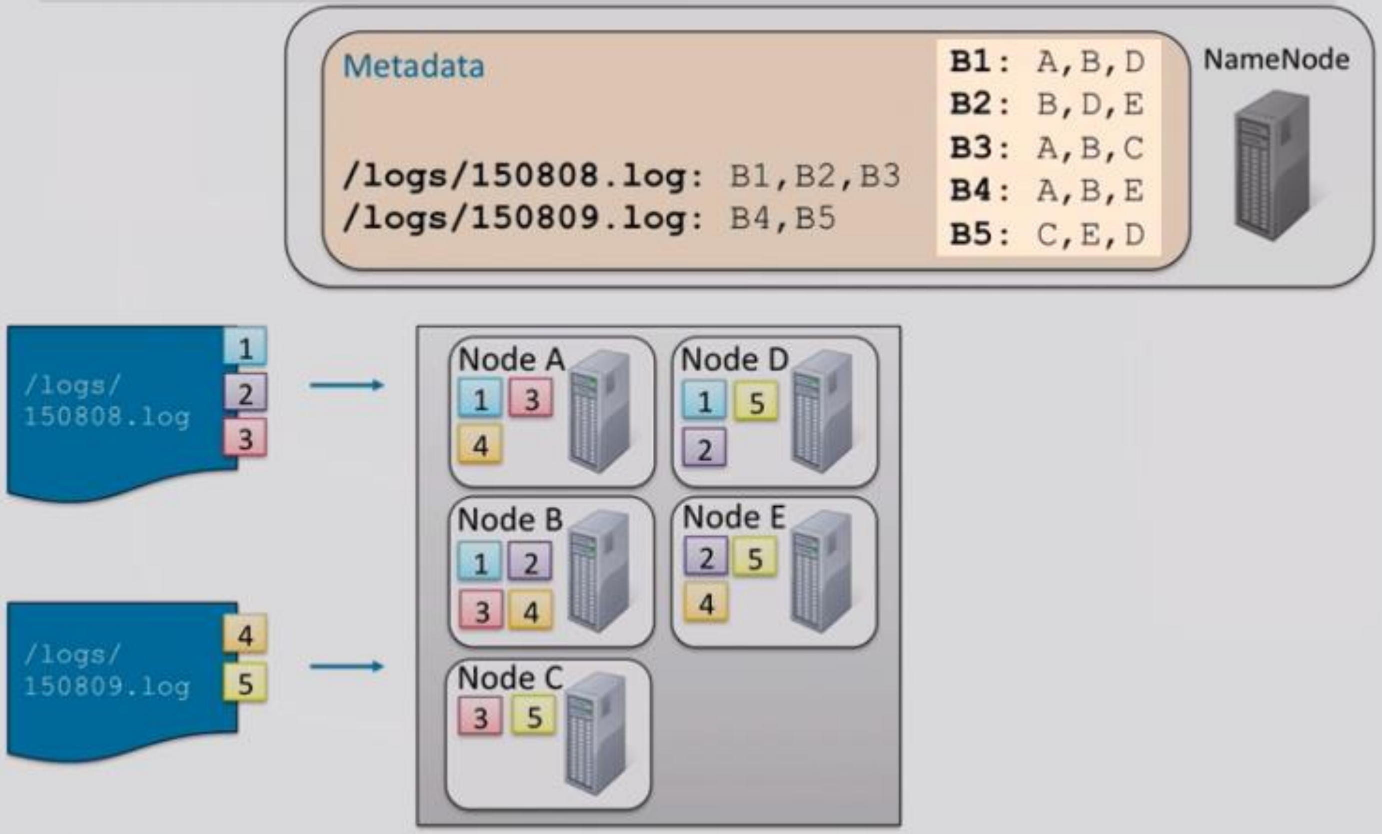


Example: Storing and Retrieving Files (2)



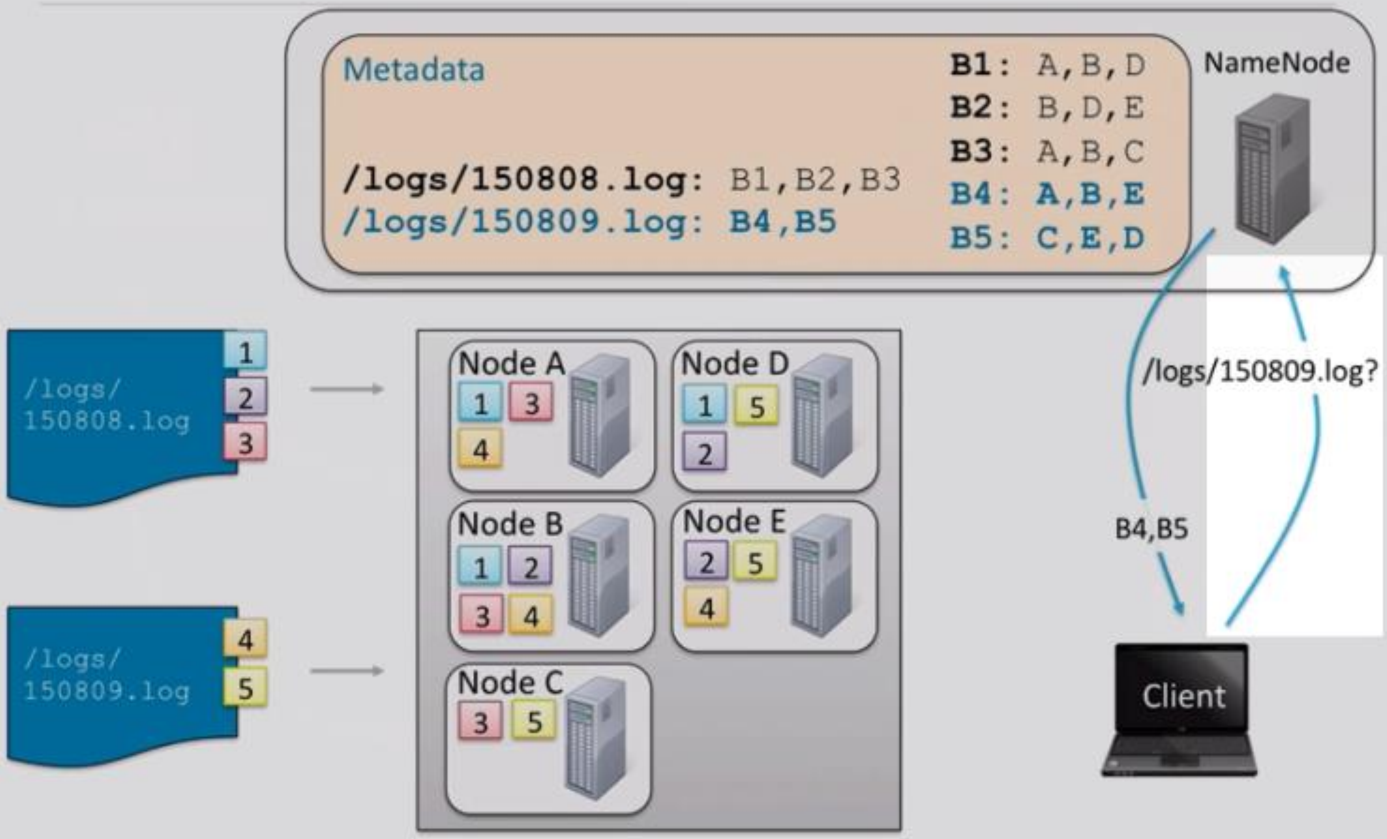

Example: Storing and Retrieving Files (2)
Metadata
/logs/150808.log: B1,B2,B3
/logs/150809.log: B4,B5
B1: A,B,D
B2: B,D,E
B3: A,B,C
B4: A,B,E
B5: C,E,D
NameNode
/logs/
150808.log
1
2
3
/logs/
150809.log
4
5
Node A
1 3
4
Node D
1 5
2
Node B
1 2
3 4
Node E
2 5
4
Node C
3 5

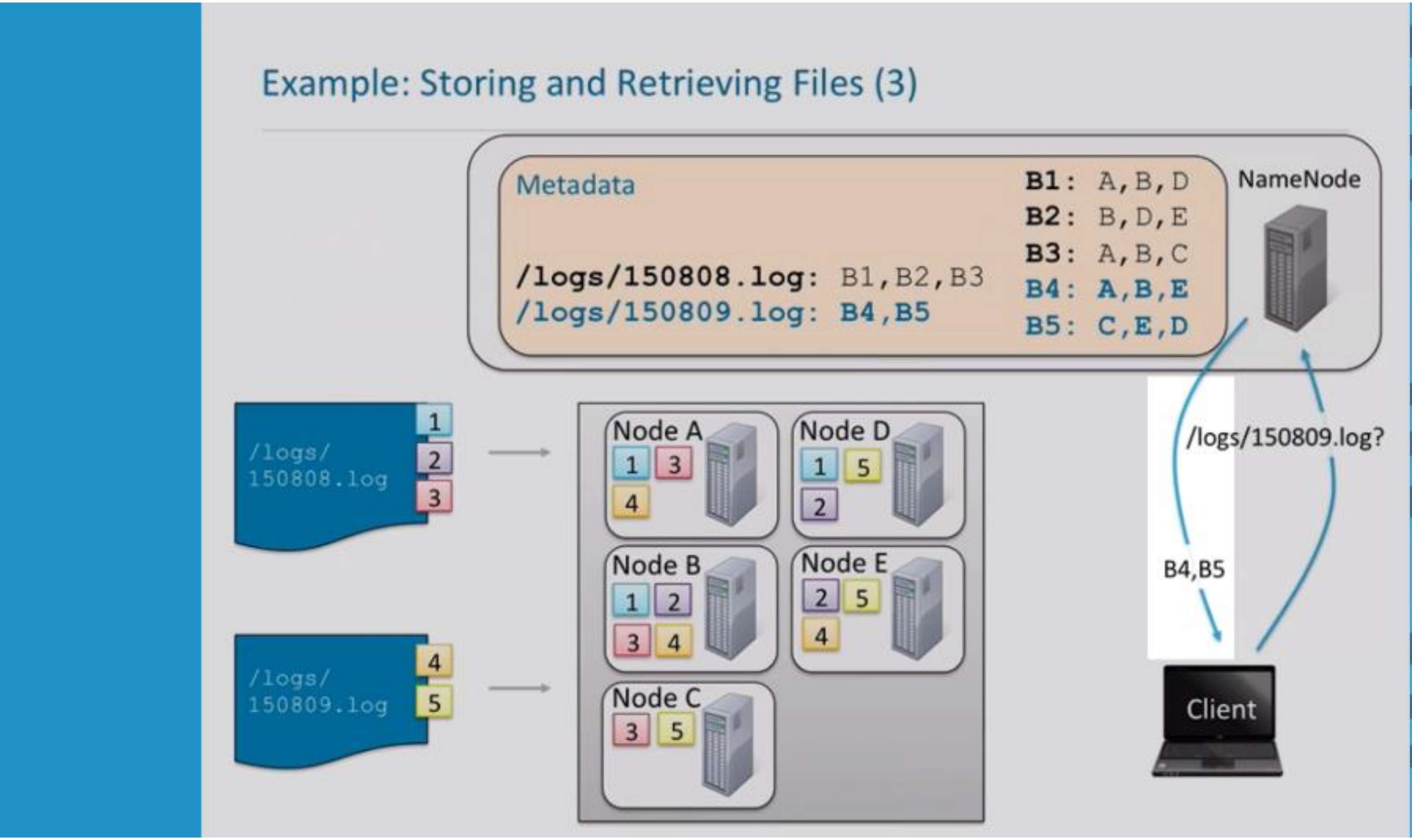
Example: Storing and Retrieving Files (2)



cloudera

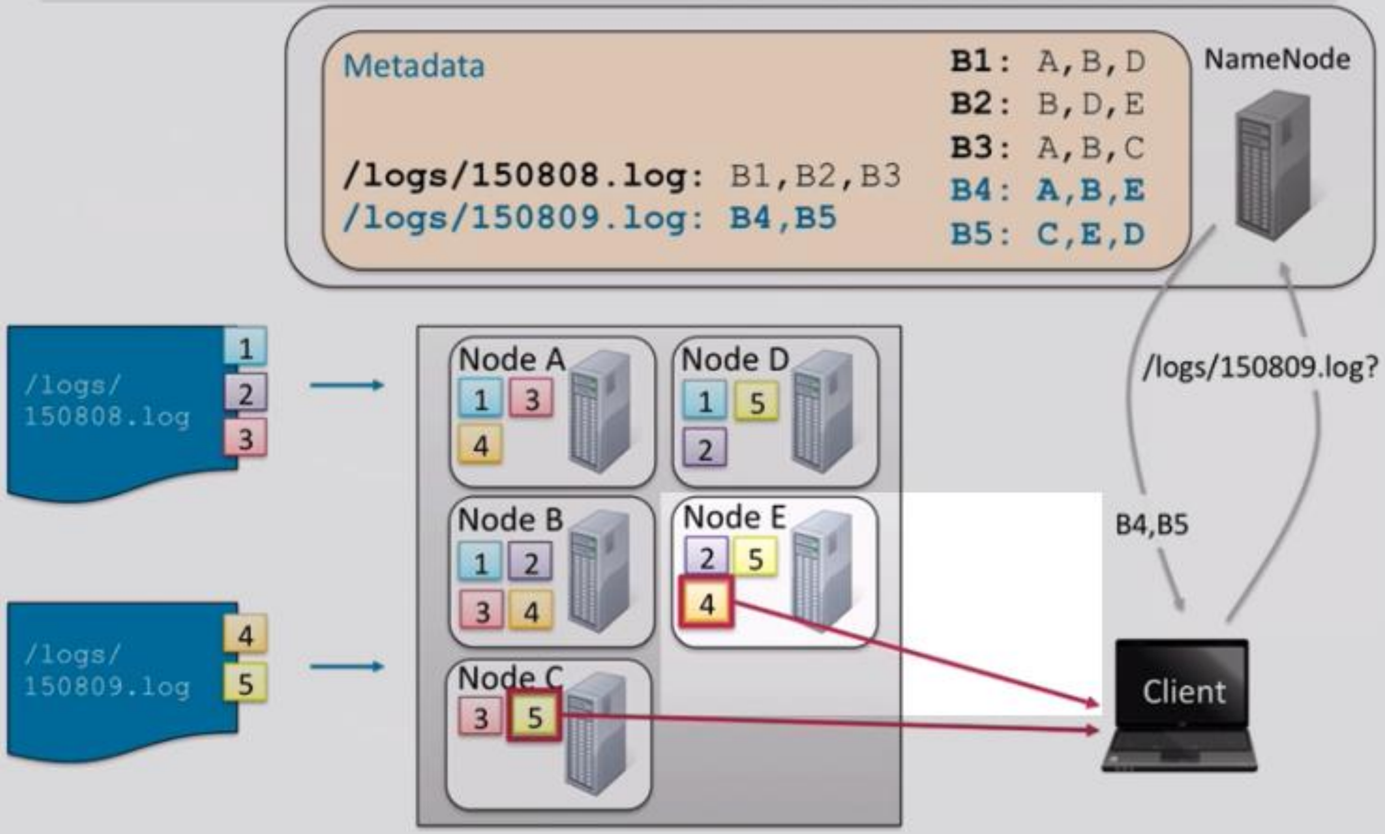
Example: Storing and Retrieving Files (3)





Example: Storing and Retrieving Files (3)
Metadata
/logs/150808.log: B1,B2,B3
/logs/150809.log: B4,B5
B1: A,B,D
B2: B,D,E
B3: A,B,C
B4: A,B,E
B5: C,E,D
NameNode
/logs/
150808.log
1
2
3
/logs/
150809.log
4
5
Node A
1
3
4
Node D
1
5
2
Node B
1
2
3
4
Node E
2
5
4
Node C
3
5
/logs/150809.log?
B4,B5
Client

Example: Storing and Retrieving Files (4)



MapReduce: Key Features

- **MapReduce is a programming model**
  - Neither platform- nor language-specific
  - Record-oriented data processing (key and value)
  - Facilitates task distribution across multiple nodes
- **MapReduce was the original processing framework available on Hadoop**
  - Still widely used, although other frameworks are replacing it for many types of workload
- **MapReduce code is typically written in Java**

The Motivation for YARN

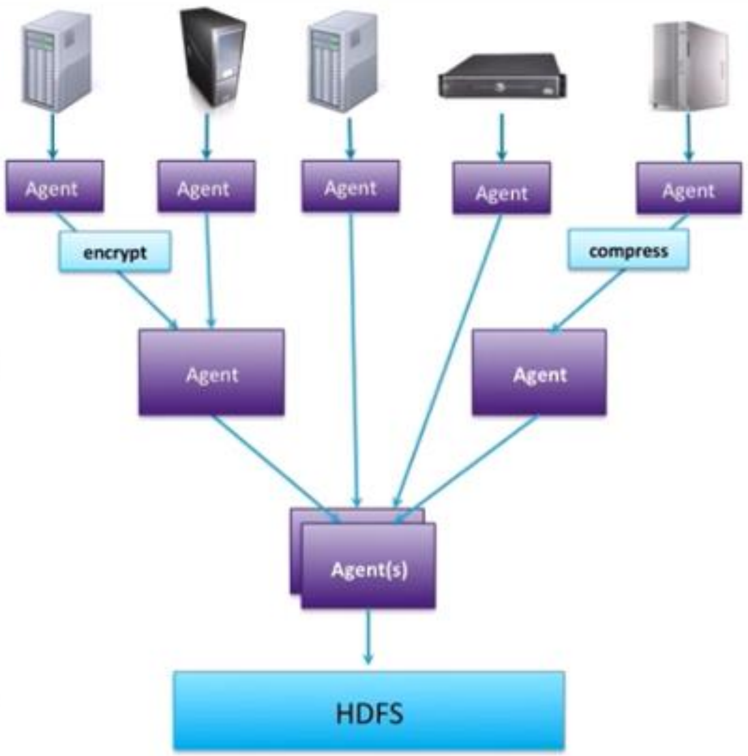
- **Originally, Hadoop only supported MapReduce as a processing framework**
- **MapReduce used all of the cluster's processing resources**
- **Now, multiple frameworks may exist on a single cluster**
  - MapReduce
  - Spark
- **Each framework competes for compute and memory resources on the nodes**
- **YARN (Yet Another Resource Negotiator) was developed to manage this contention**
  - Allocates resources to different frameworks based on demand, and on system administrator settings

Flume and Kafka: What Are They?

- **Flume and Kafka are tools for ingesting event data into Hadoop as that data is being generated**
  - Log files
  - Sensor data
  - Streaming data from social media such as Twitter
  - Etc.
- **Flume is typically easier to configure, but Kafka provides more functionality**
  - Flume generally provides a path from a data source to HDFS or to a streaming framework such as Spark
  - Kafka uses a 'Publish/Subscribe' model
    - Allows data to be consumed by many different systems, including writing to HDFS

Example Flume Pipeline

- Collect data as it is produced
  - Files, syslogs, stdout or custom source
- Process in place
  - e.g., encrypt, compress
- Pre-process data before storing
  - e.g., transform, scrub, enrich
- Write in parallel
  - Scalable throughput
- Store in any format
  - Text, compressed, binary, or custom sink

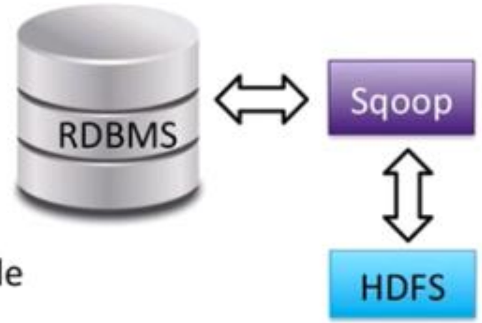


Flume and Kafka: Why Should I Use Them?

- **Flume and Kafka are ideal for aggregating event data from many sources into a centralized location (HDFS)**
- **Well-suited for event driven data**
  - Network traffic
  - Social-media-generated
  - Email messages
  - GPS tracking information
  - Digital sensors
  - Log files
- **Allow you to process streaming data, as that data is being generated**
  - Vital for applications such as fraud prevention, threat detection

Sqoop: What Is It?

- **Sqoop rapidly moves large amounts of data between relational database management systems (RDBMSs) and HDFS**
  - Import tables (or partial tables) from an RDBMS into HDFS
  - Export data from HDFS to a database table



- **Uses JDBC to connect to the database**
  - Works with virtually all standard RDBMSs
- **Custom 'connectors' for some RDBMSs provide much higher throughput**
  - e.g., Teradata, Oracle

Spark: What Is It?

- **Apache Spark is large-scale data processing engine**
- **Supports a wide range of workloads**
  - Machine learning
  - Interactive analytics
  - Batch applications
  - Iterative algorithms
  - Business Intelligence
  - Etc.
- **Spark Streaming provides the ability to process data as that data is being generated**
  - Typically in conjunction with Flume or Kafka

Spark: Why Should I Use It?

- **Faster than MapReduce**
- **Spark code can be written in Python, Scala, or Java**
  - Easier to develop for than MapReduce
- **Spark is well-suited to iterative processing algorithms such as many of those used in machine learning applications**
- **Spark Streaming provides real-time data processing features**
- **Spark is replacing MapReduce at many organizations**
  - Organizations new to Hadoop will typically start with Spark and never write MapReduce code

Apache Hive: What Is It?

- **Hive is an abstraction layer on top of Hadoop**
  - Hive uses a SQL-like language called HiveQL
- **The Hive interpreter uses MapReduce or Spark to actually process the data**
- **JDBC and ODBC drivers are available**
  - Allows Hive to integrate with BI and other applications

```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

Hive: Why Should I Use It?

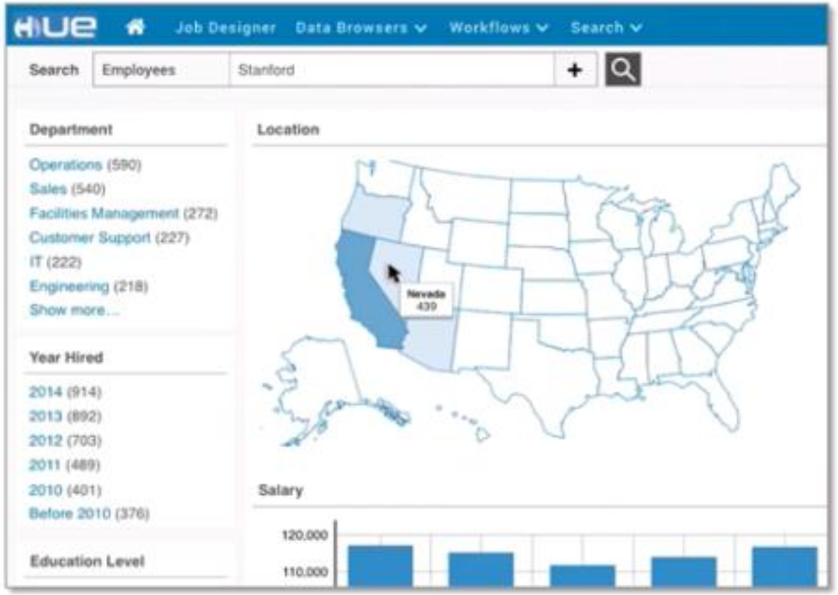
- **Data can be loaded before the table is defined**
  - Schema-on-Read
  - You do not need to know the data's structure prior to loading it
- **Does not require a developer who knows Java, Scala, Python or other traditional programming languages**
  - Anyone who knows SQL can process and analyze the data on the cluster
- **Well suited for dealing with structured data, or data which can have a structure applied to it**

Comparing Hive to an RDBMS

Feature	RDBMS	Hive
Query language	SQL	SQL
Update and delete records	Yes	Experimental
Transactions	Yes	Experimental
Stored procedures	Yes	No
Index support	Extensive	Limited
Latency	Very low	High
Scalability	Low	Very high
Data format flexibility	Minimal	Very high
Storage cost	Very expensive	Inexpensive

Hue: What Is It?

- **Hue provides a Web front-end to a Hadoop**
  - Upload data
  - Browse data
  - Query tables in Impala and Hive
  - Search
  - And much more
- **Provides access control for the cluster by requiring users to log in before they can use the system**
- **Makes Hadoop easier to use**



HBase: What Is It?

- **HBase is a NoSQL distributed database**
- **Stores data in HDFS**
- **Scales to support very high throughput for both reads and writes**
  - Millions of inserts or updates per second
- **A table can have many thousands of columns**
  - Handles sparse data well
- **Designed to store very large amounts of data (Petabytes+)**

Comparing HBase to a Relational Database

	HBase	RDBMS
Data layout	Column Family-oriented	Row- or column-oriented
Transactions	Single row only	Yes (ACID)
Query language	get/put/scan	SQL
Indexes	Row-key only (limited support for secondary indexes	Yes
Max data size	PB+	TBs
Read/write throughput limits	Millions of queries/second	1000s of queries/second

HBase: When Should I Used It?

- **Use HBase if…**
  - You need random reads
  - You need random writes
  - You need to do thousands of operations per second on terabytes of data
  - Your access patterns are simple and well-known
- **Don’t use HBase if…**
  - You only append to your dataset and typically read the entire table
  - You primarily perform ad-hoc analytics (ill-defined access patterns)
  - Your data easily fits on one large node