Egyptian Big Data Geeks 1st Meeting

Big Data Architecture and Hadoop Echo-systems

Mostafa Alaa, Big Data Developer.

MustafaAlaa.Mohamed@gmail.com

1. Big Data “Hadoop/HDFS” Architecture.

Processing
Map-Reduce/ Spark

Resource
Management/Schedul
er

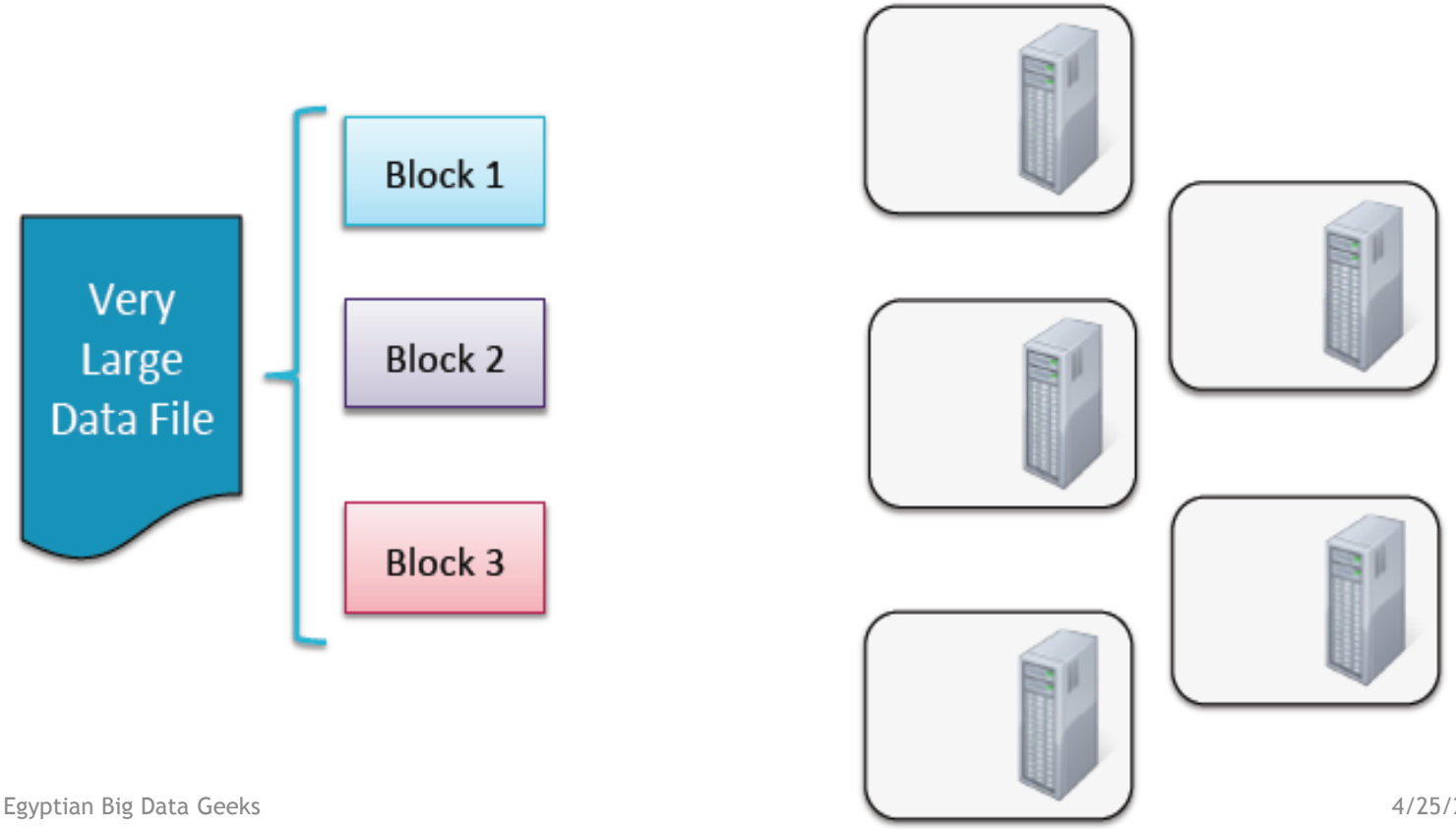
Storage
HDFS/Amazon S3/…

Master/Name
Node/Queen

Workers/Nodes/Data
Nodes/Slaves

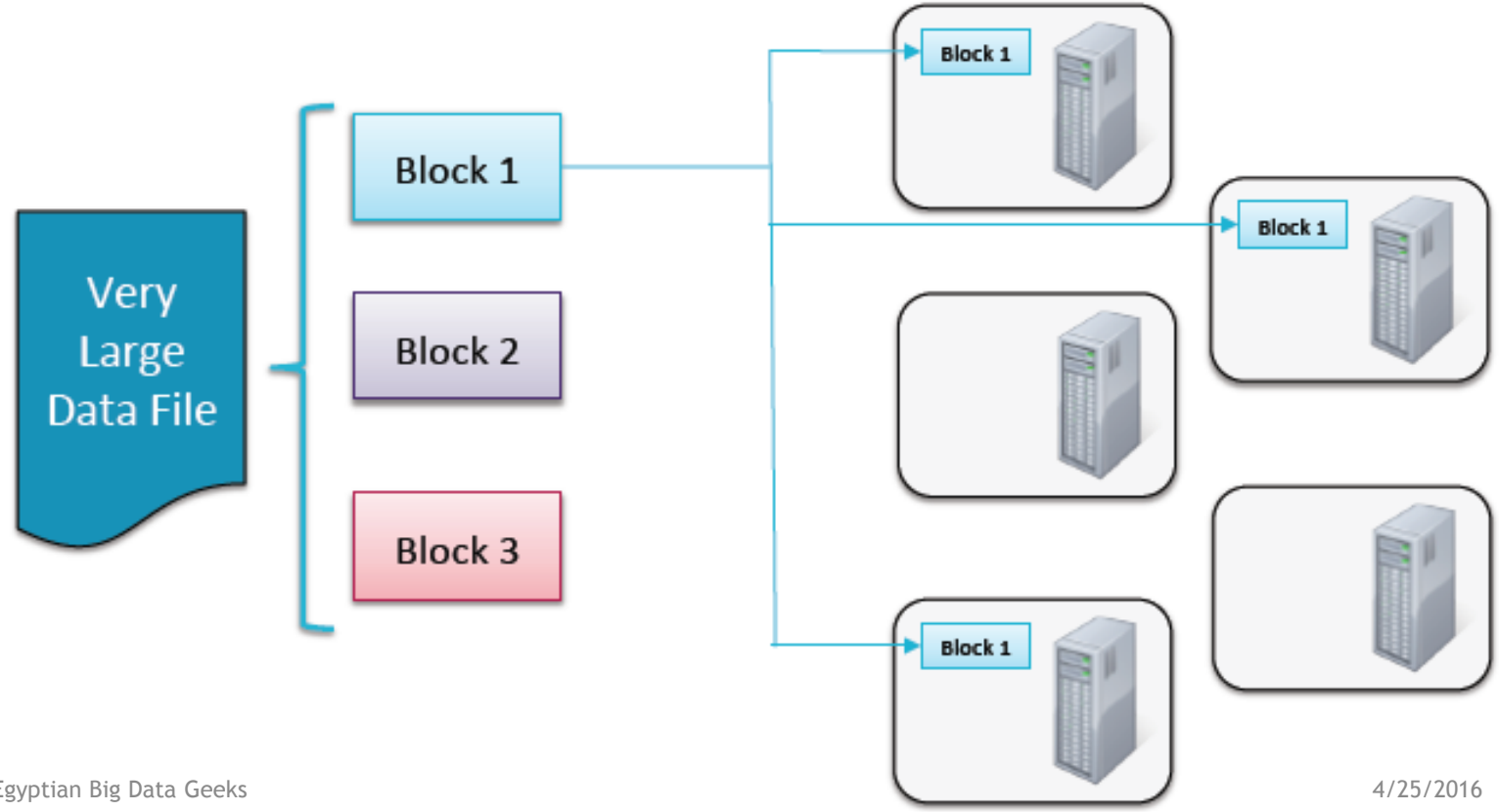
1.1 Storage

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



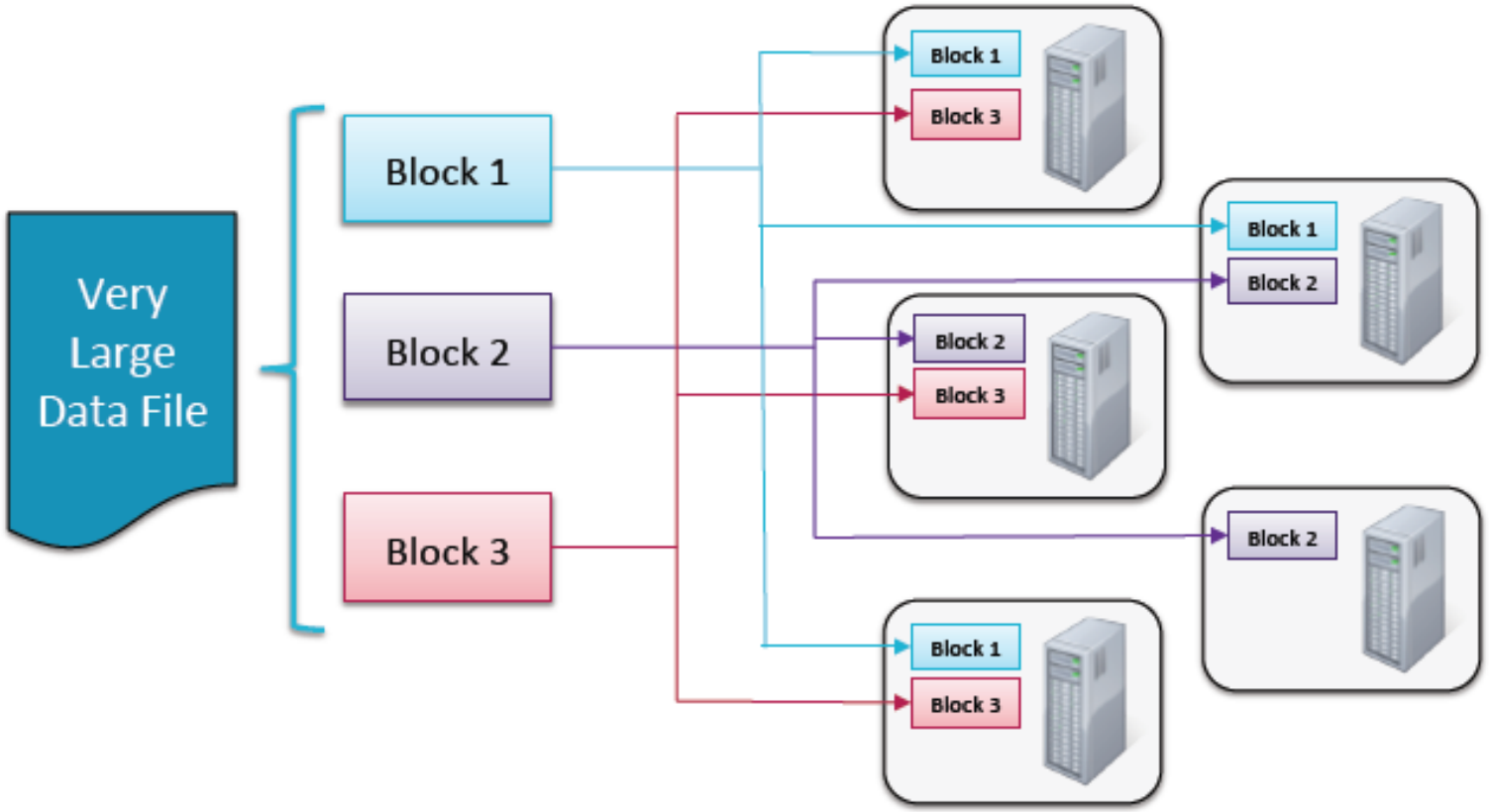
1.1 Storage

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



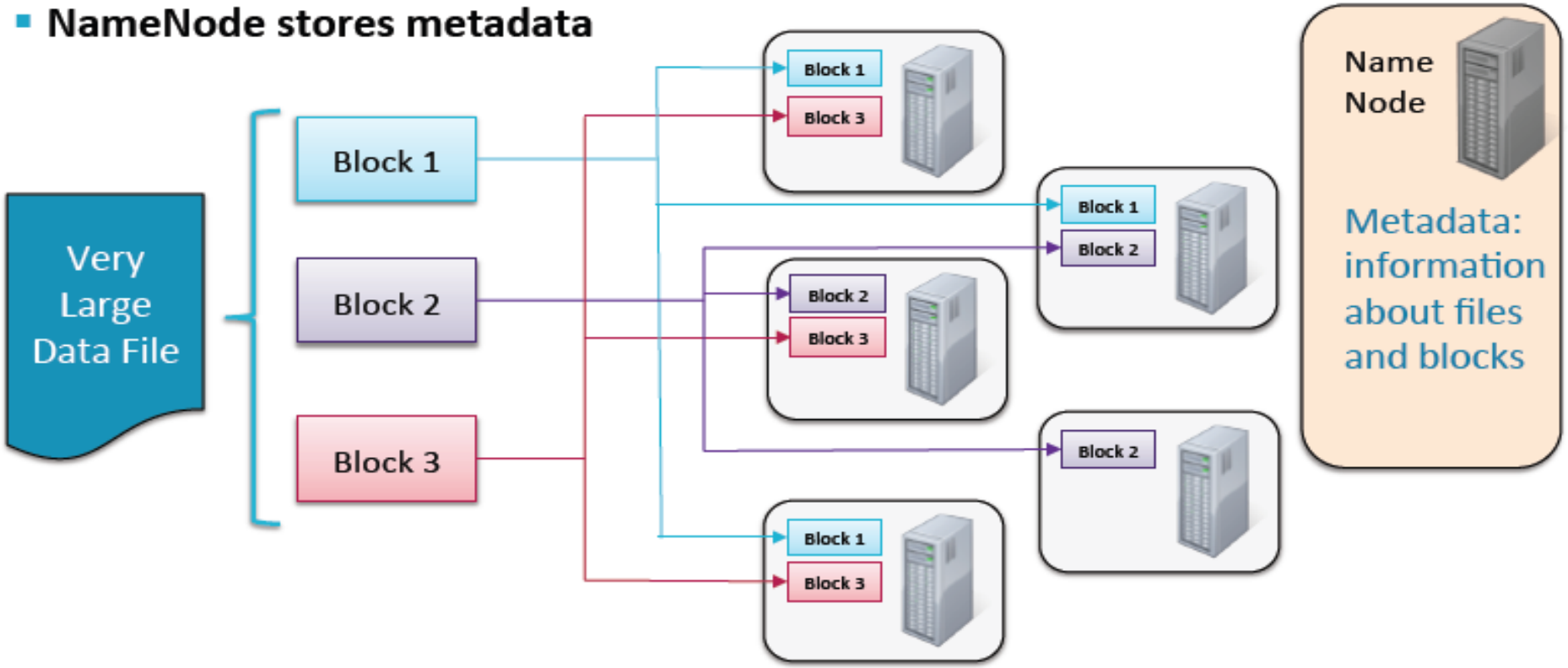
1.1 Storage

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

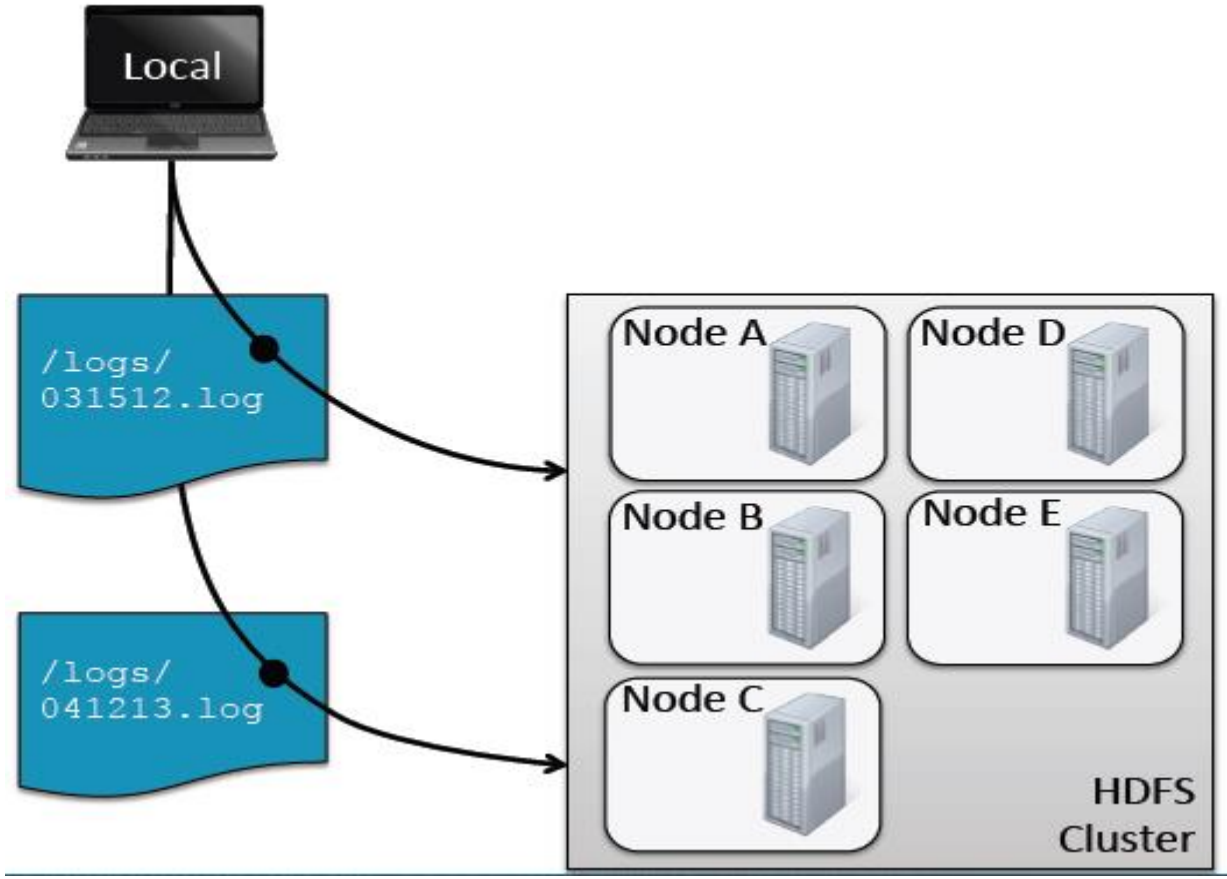


1.1 Storage

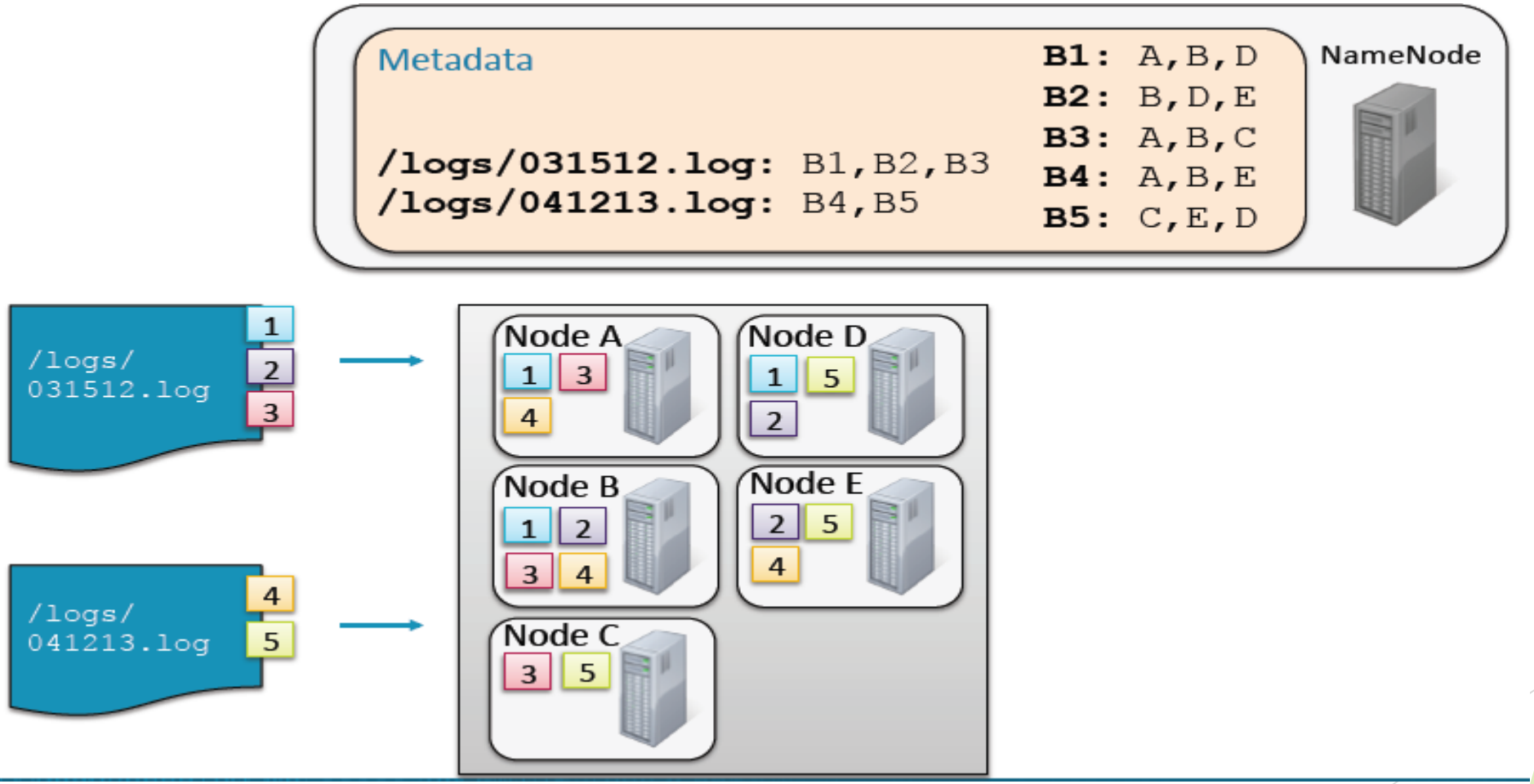
- **Data files are split into blocks and distributed to data nodes**
- **Each block is replicated on multiple nodes (default 3x)**
- **NameNode stores metadata**



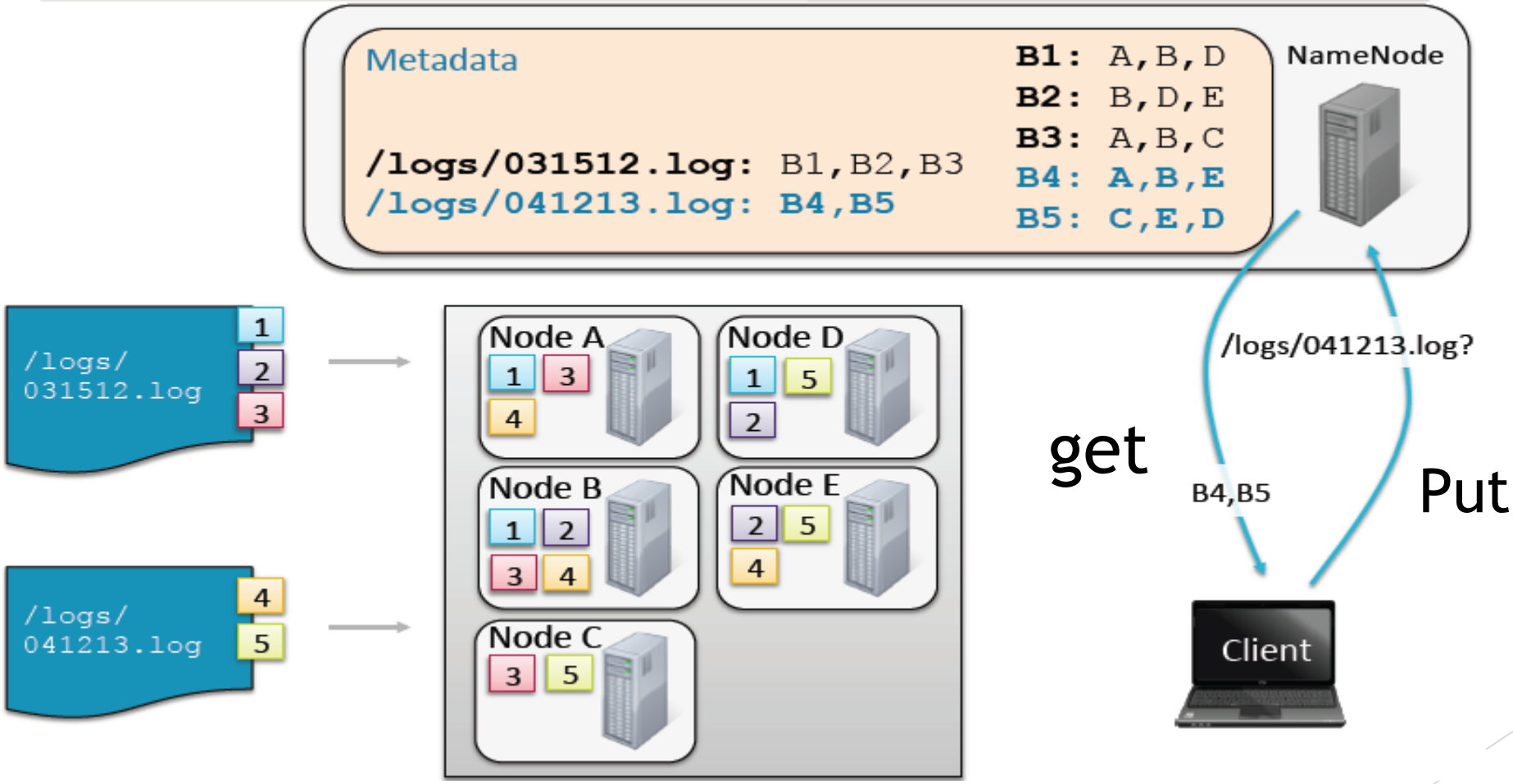
1.1 Storage


Local
/logs/
031512.log
/logs/
041213.log
Node A
Node D
Node B
Node E
Node C
HDFS
Cluster

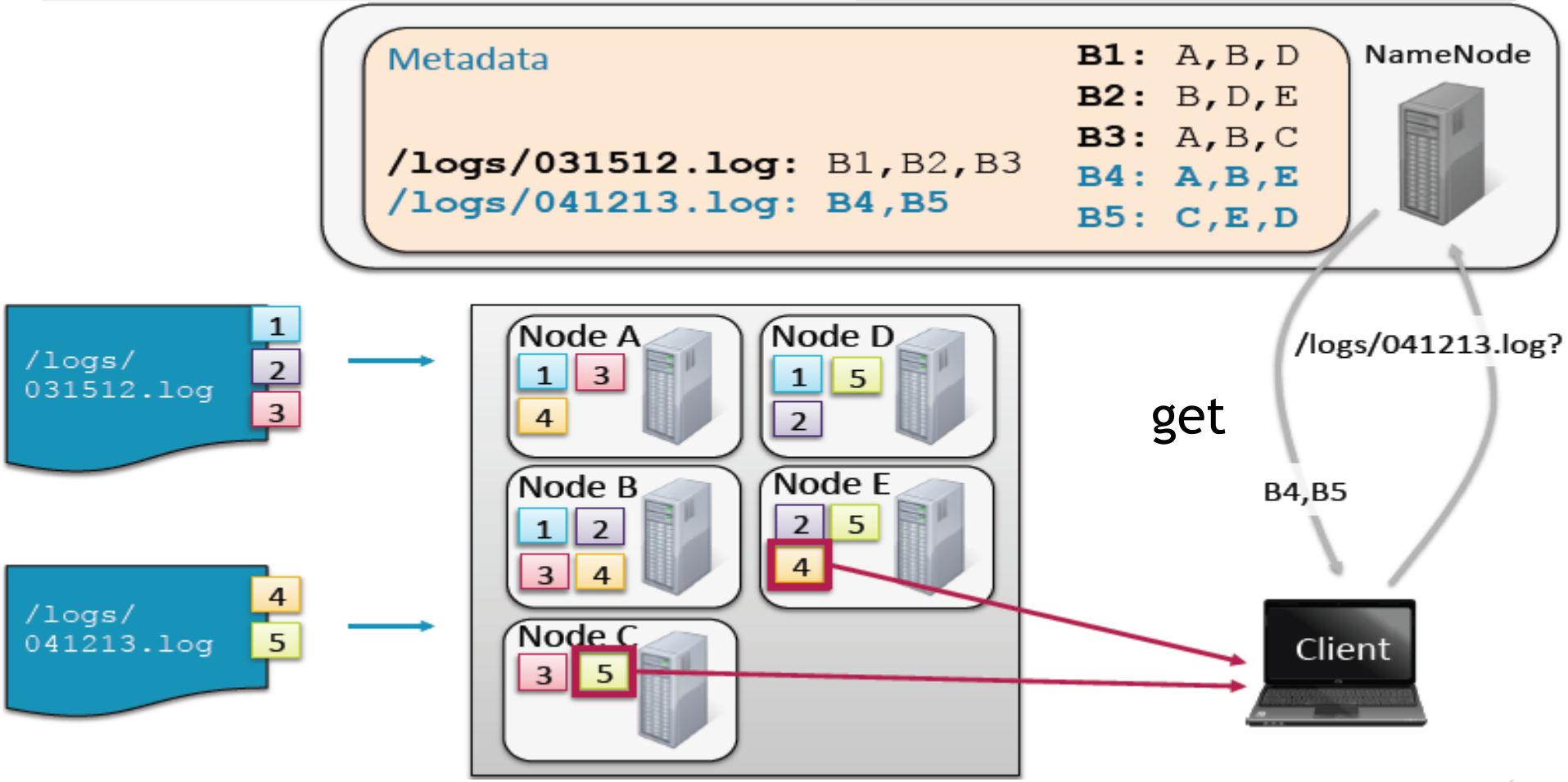
1.1 Storage


Metadata
/logs/031512.log: B1,B2,B3
/logs/041213.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/ 031512.log
1
2
3
/logs/ 041213.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

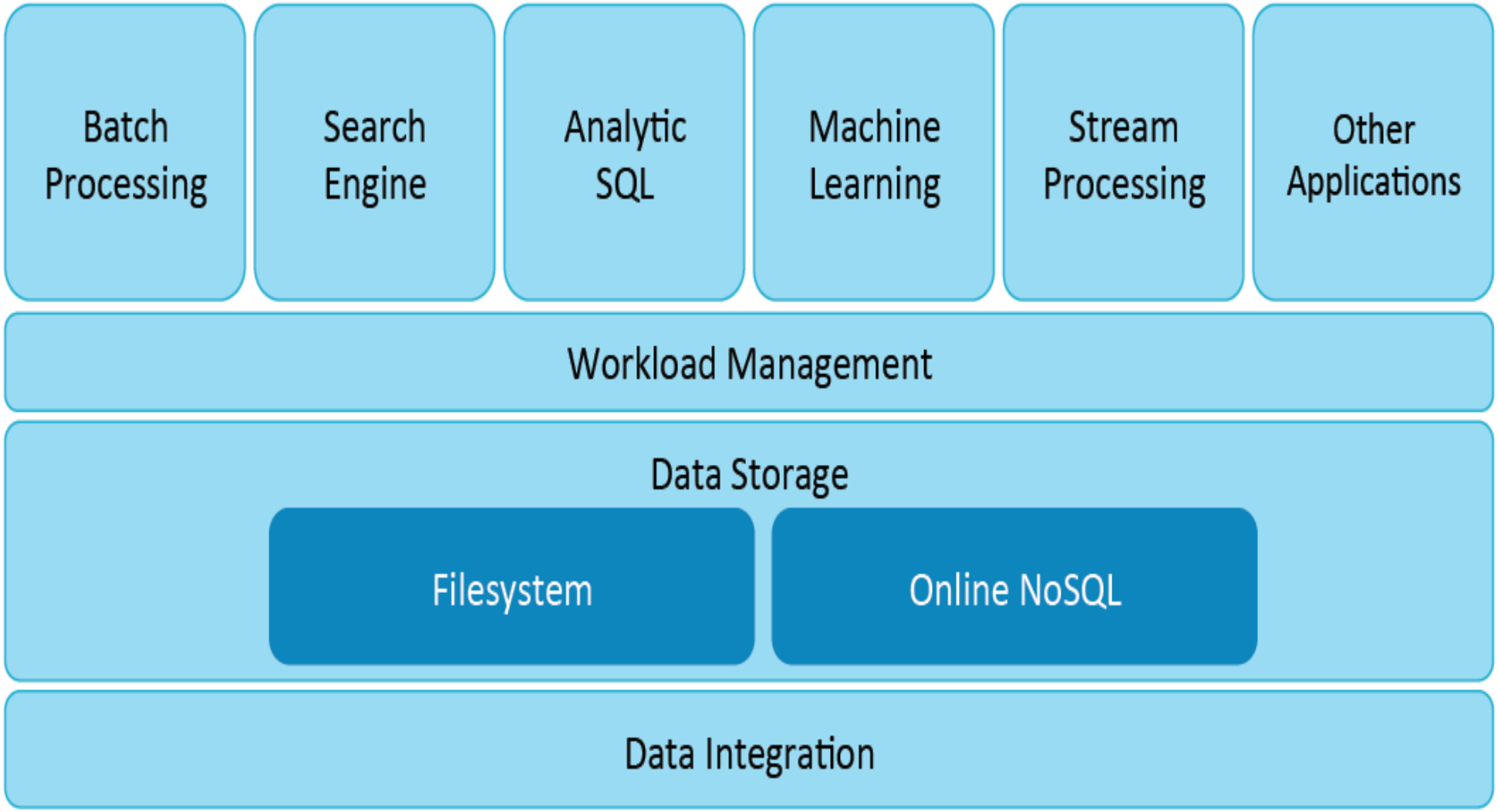
1.1 Storage


Metadata
/logs/031512.log: B1,B2,B3
/logs/041213.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/031512.log
1
2
3
/logs/041213.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/041213.log?
get
B4,B5
Put
Client

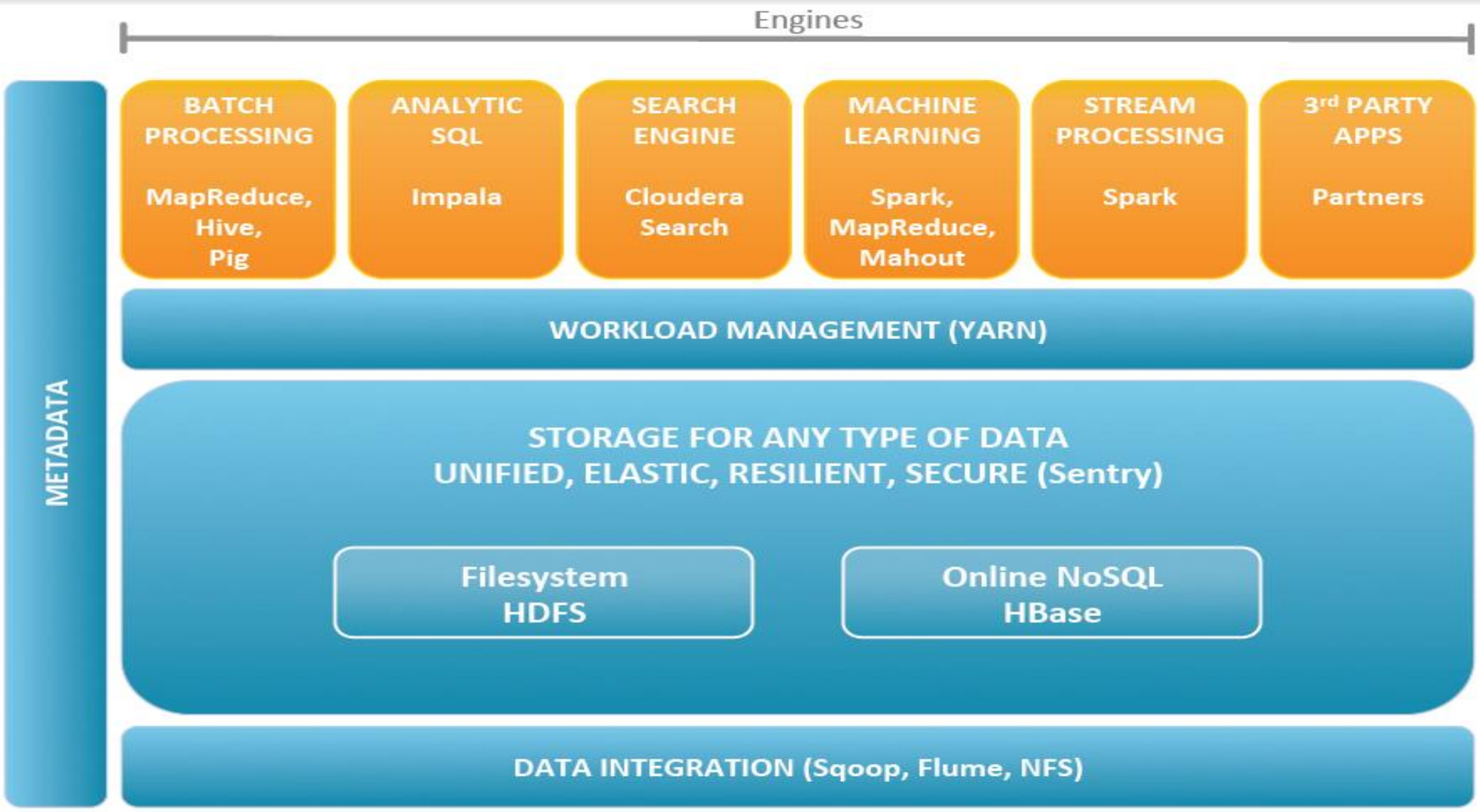
1.1 Storage


Metadata
/logs/031512.log: B1,B2,B3
/logs/041213.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/031512.log
1
2
3
/logs/041213.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/041213.log?
get
B4,B5
Client

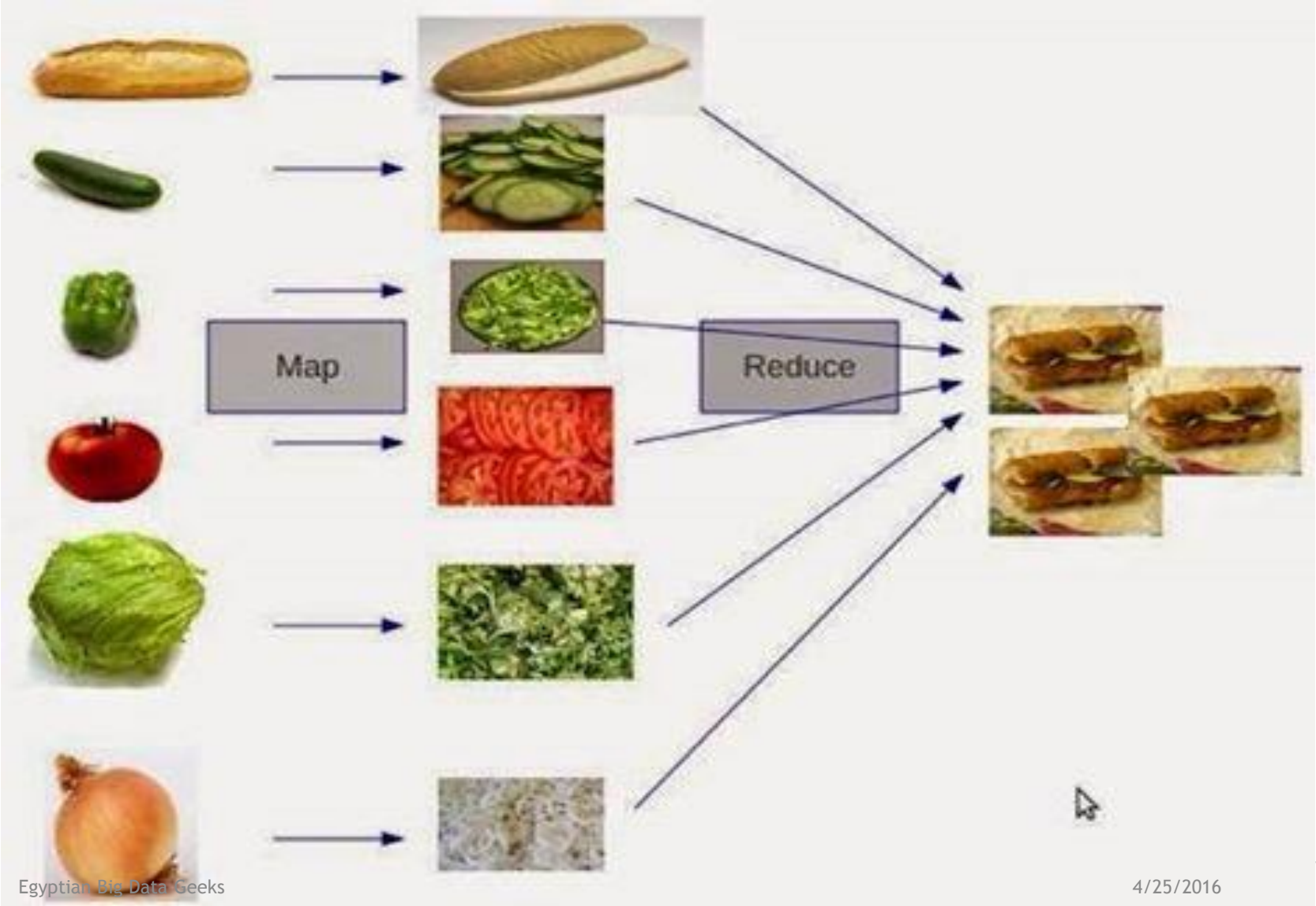
1.2 Processing



1.2 Processing

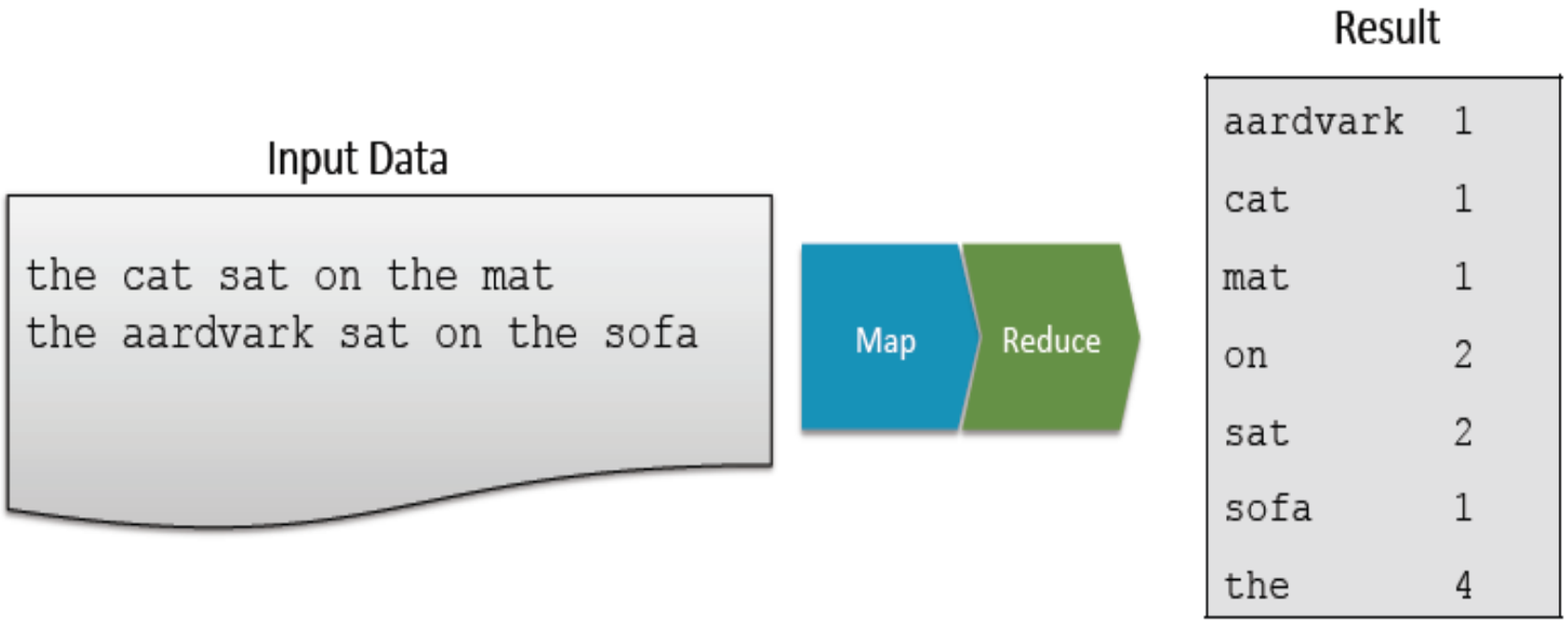


1.2 Batch Processing- Map-Reduce Concept

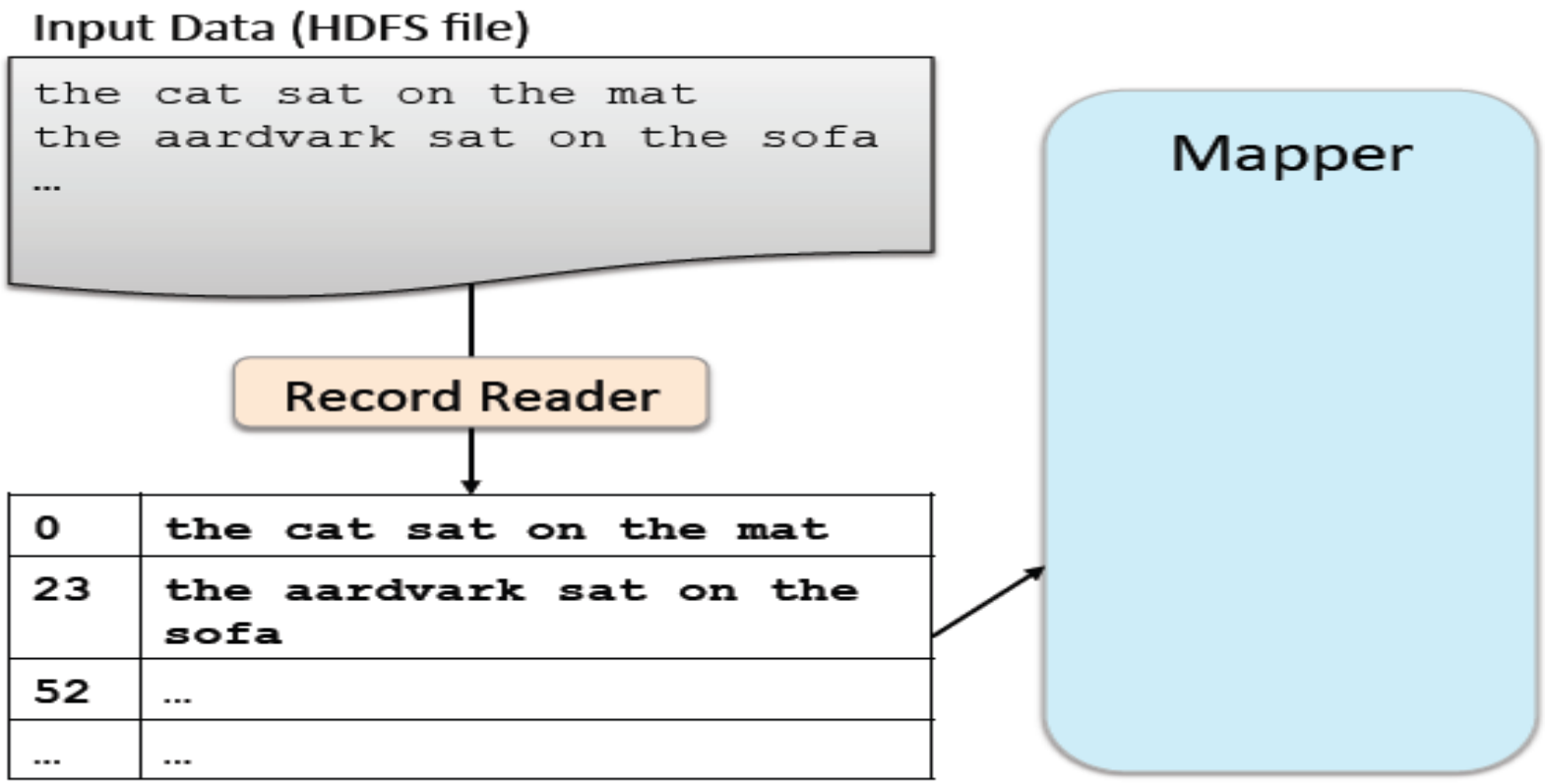


By Haider Haider

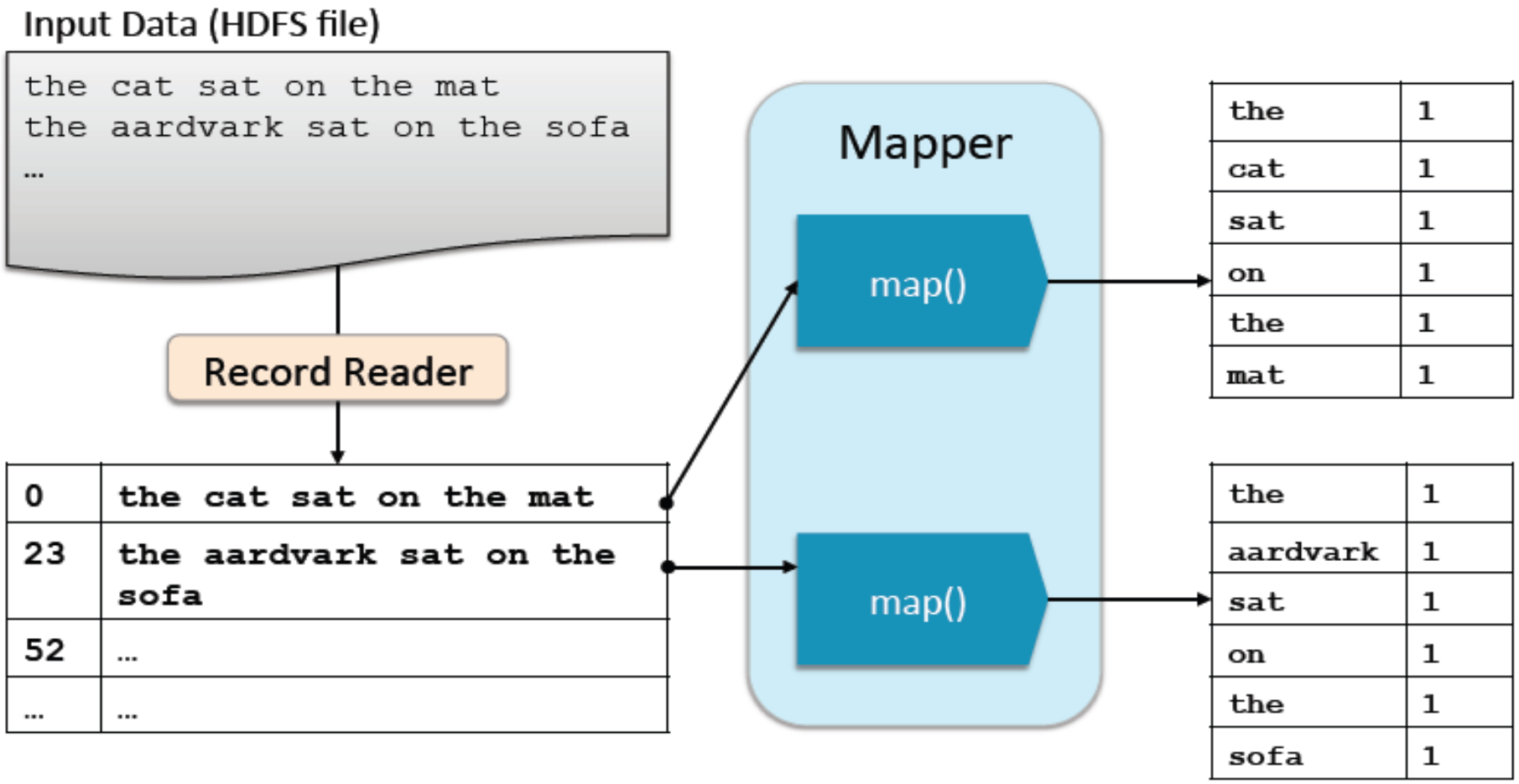
1.2 Map-Reduce, Word Count Example



1.2 Map-Reduce, Word Count Example

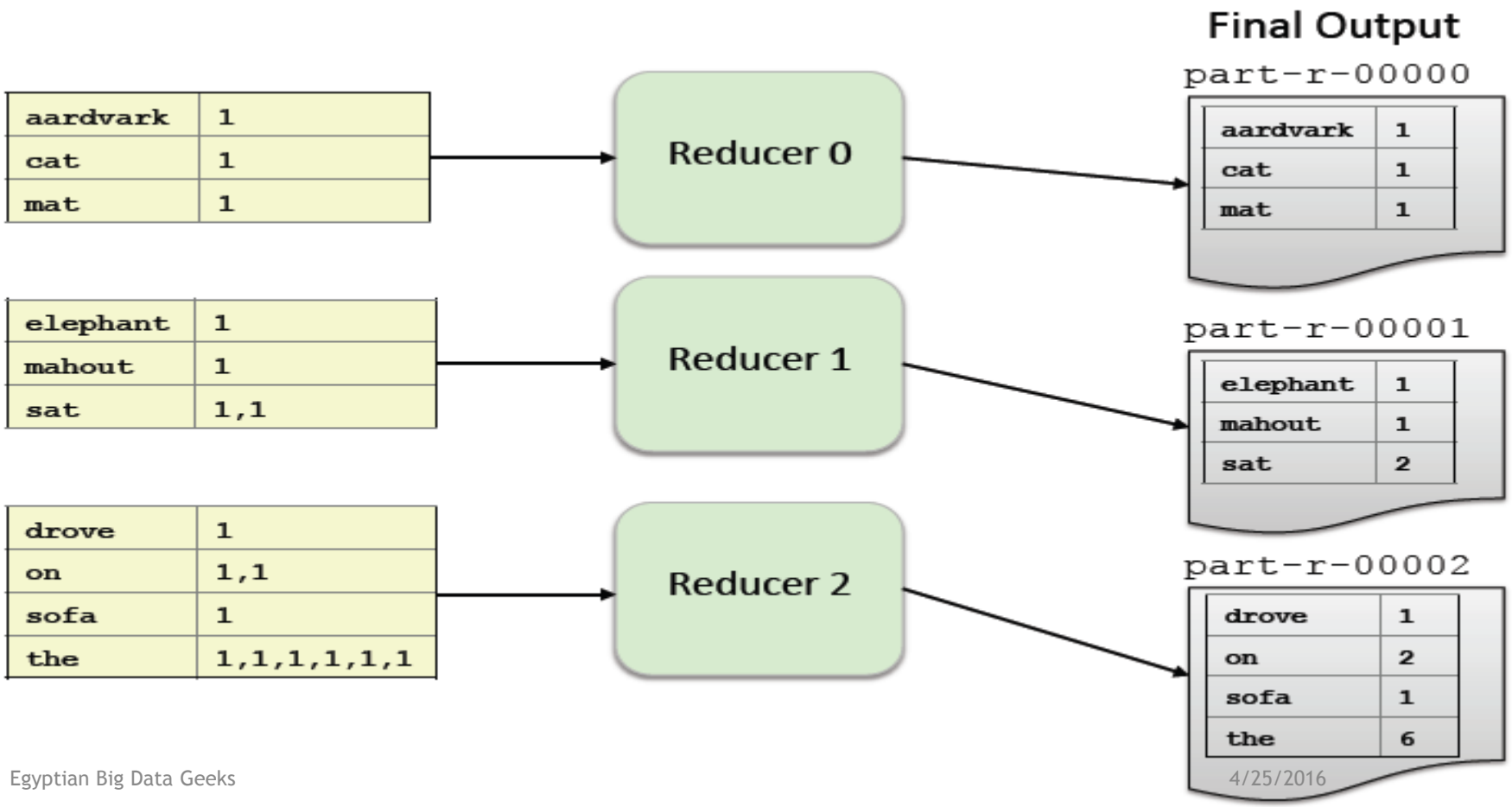


1.2 Map-Reduce, Word Count Example



1.2 Map-Reduce, Word Count Example

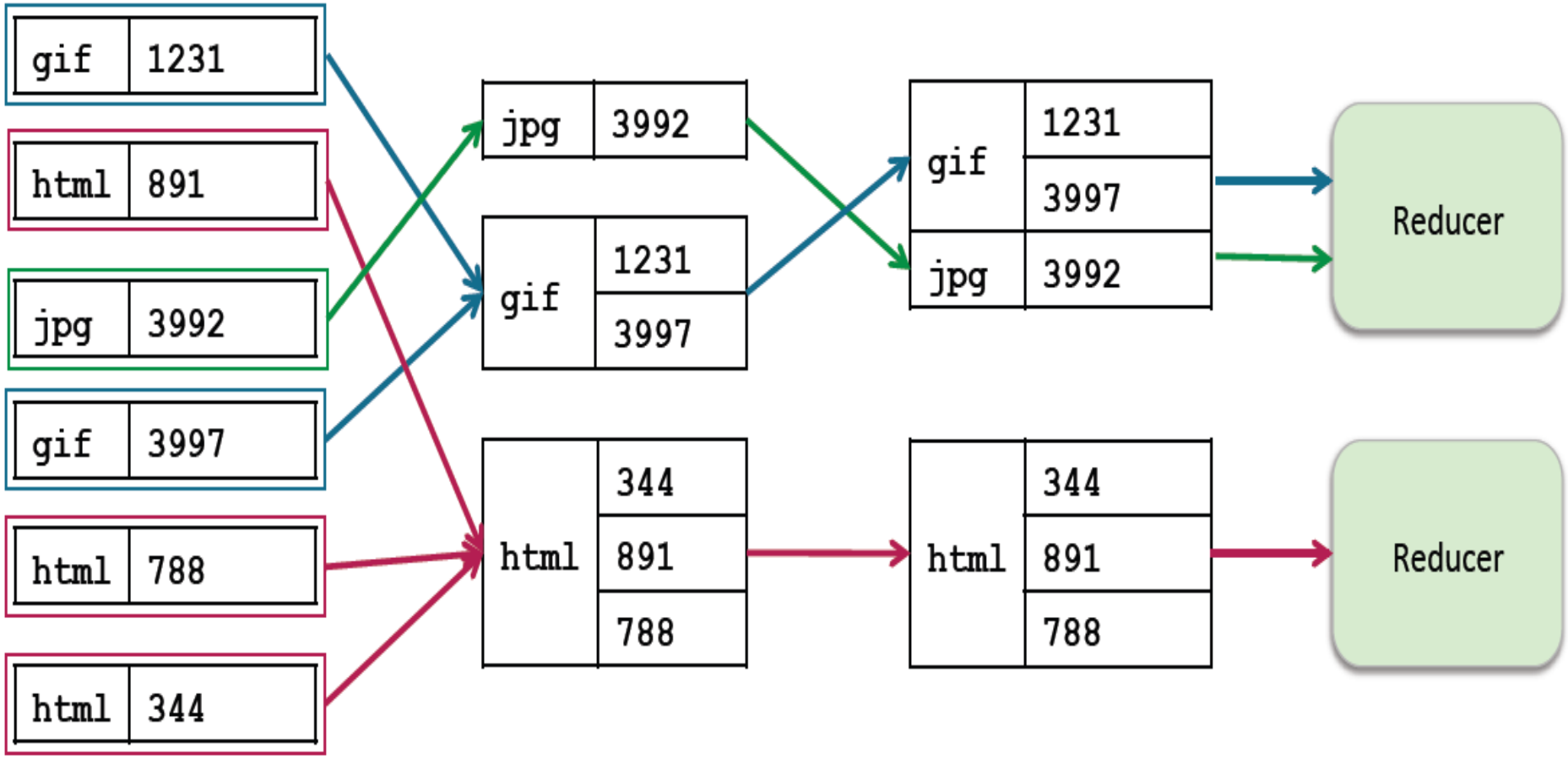
Reduce phase



1.2 Map-Reduce, Word Count Example

- What happened Between Map and Reduce?
- How the data Shuffled and moved to the reducer?
- Can we choose which reducer to send the data?

1.2 Map-Reduce, Word Count Example



1.2 Map-Reduce, Word Count Example

- **Map-Reduce Program Structure.**
  - **Mapper Class.**
  - **Reducer Class.**
  - **Driver Class.**

1.2 Map-Reduce, Word Count Example

- **Map-Reduce Program Structure.**
  - **Mapper Class.**
  - **Reducer Class.**
  - **Driver Class.**

1.2.1 Mapper Class Simple Example

```
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException
{
    String line = value.toString();
    for (String word : line.split(" "))
    {
        if (word.length() > 0)
        {
            context.write(new Text(word), new IntWritable(1));
        }
    }
}
```

1.2.1 Reduce Class Simple Example

```
public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException
{
    int wordCount = 0;
    for (IntWritable value : values)
    {
        wordCount += value.get();
    }
    context.write(key, new IntWritable(wordCount));
}
```

1.2.1 Driver Class

- The driver code runs on the client machine.
- It configures the job, then submits it to the cluster.

```
Job job = new Job();
job.setJarByClass(WordCount.class);
job.setJobName("Word Count");
FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(WordMapper.class);
job.setReducerClass(SumReducer.class);
```

1.2.1 Job Deployment

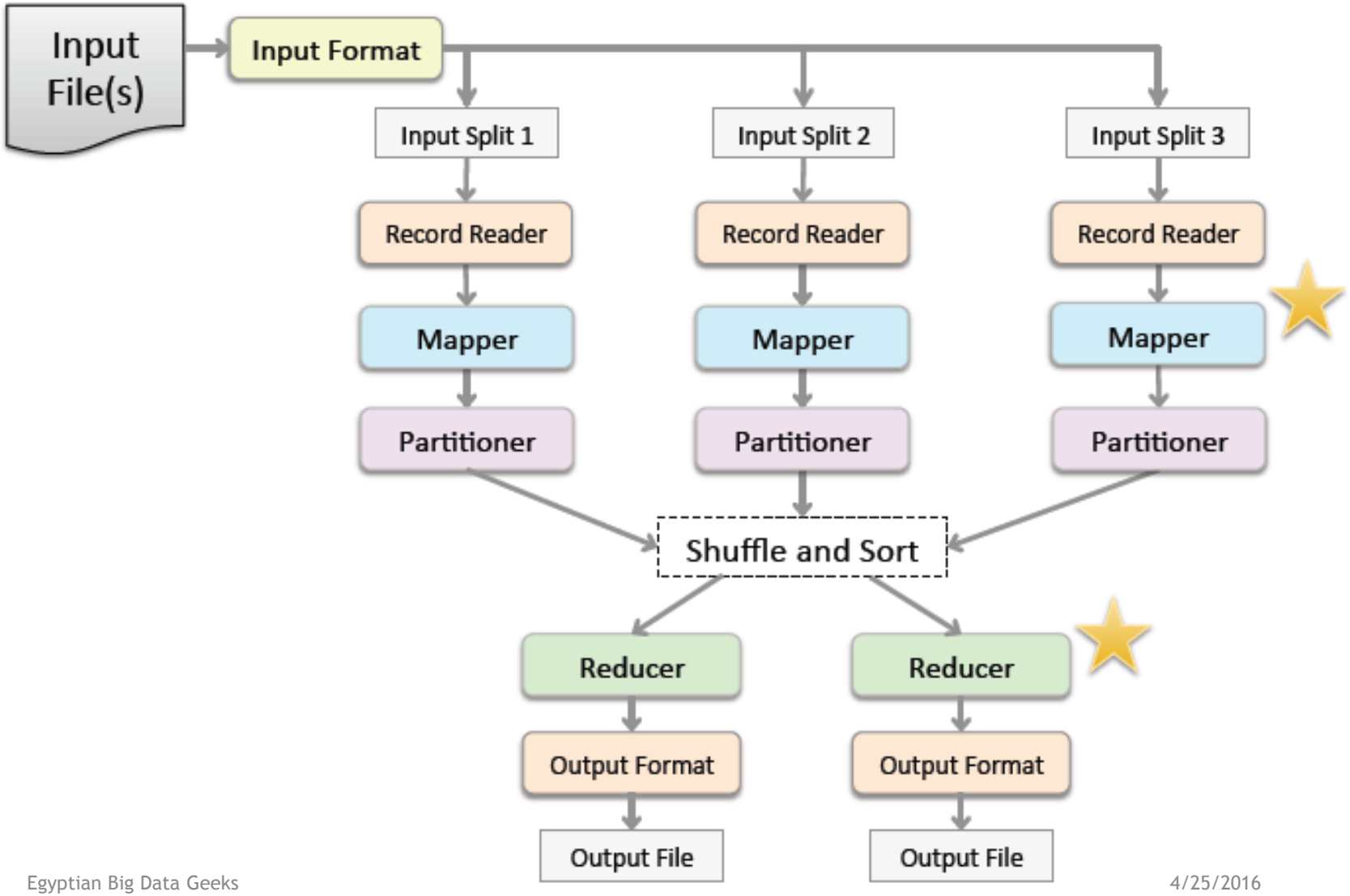
- Simple Export the Project as Jar then Run it from your client machine into the Cluster.

```
hadoop jar wc.jar stubs.WordCount
input_Dir output_Dir
```

Input directory from HDFS

output directory to HDFS

1.2 Map-Reduce, Word Count Example

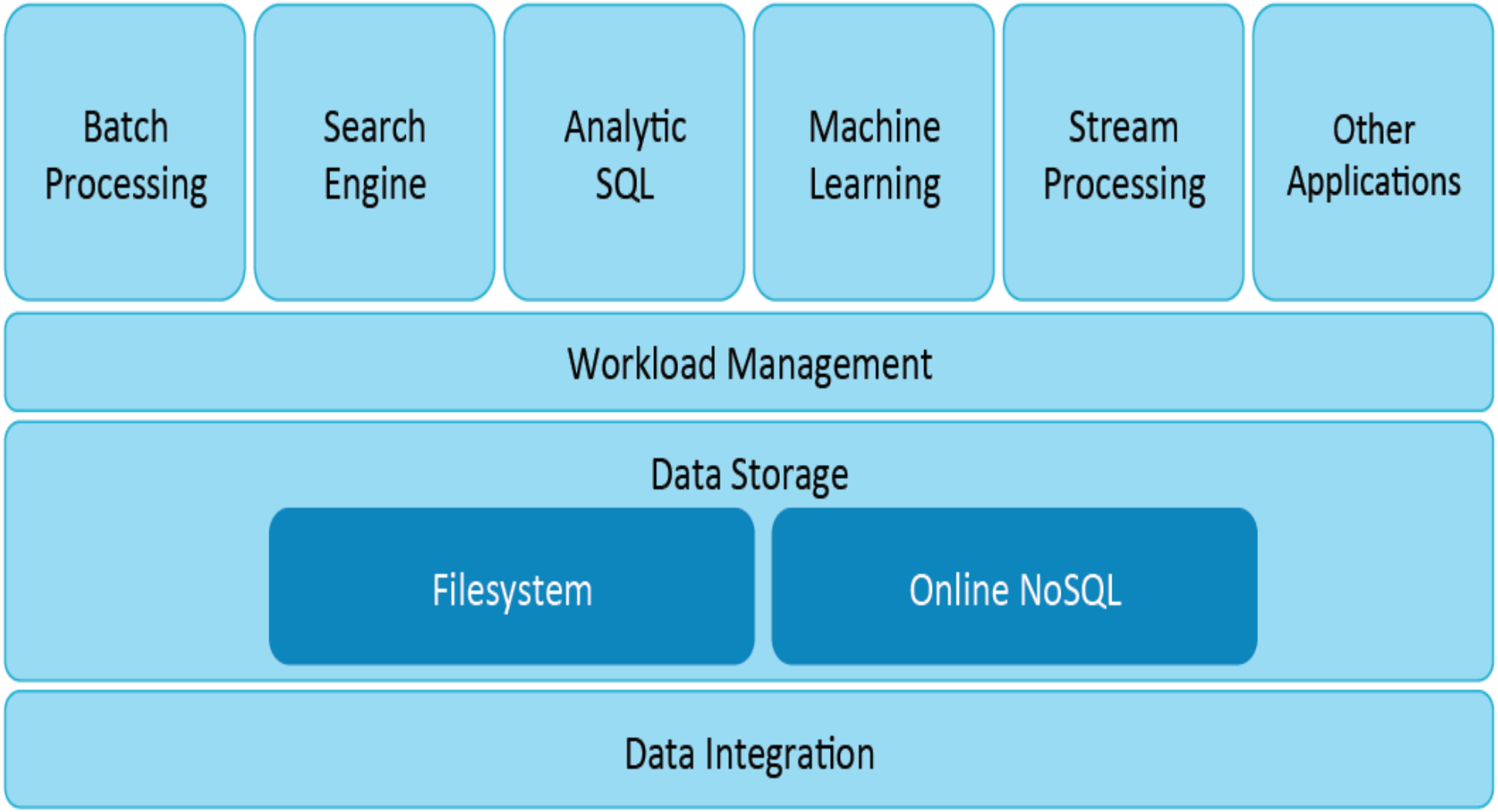


Map-Reduce Concept

- Full example will be uploaded into our group Egyptian Big Data Geeks..

https://www.facebook.com/groups/big.data.egypt/

1.2 Processing



1.2.2 Processing -Hive

1.2.2 Processing -Hive

- What is Hive?
  - Apache Hive is a high level abstraction on top of MapReduce.
  - Uses a SQL like language called HiveQL .
  - Generates MapReduce jobs that run on the Hadoop cluster.
  - Originally developed by Facebook for data warehousing.
  - Now an open-source Apache project.

1.2.2 Processing -Hive

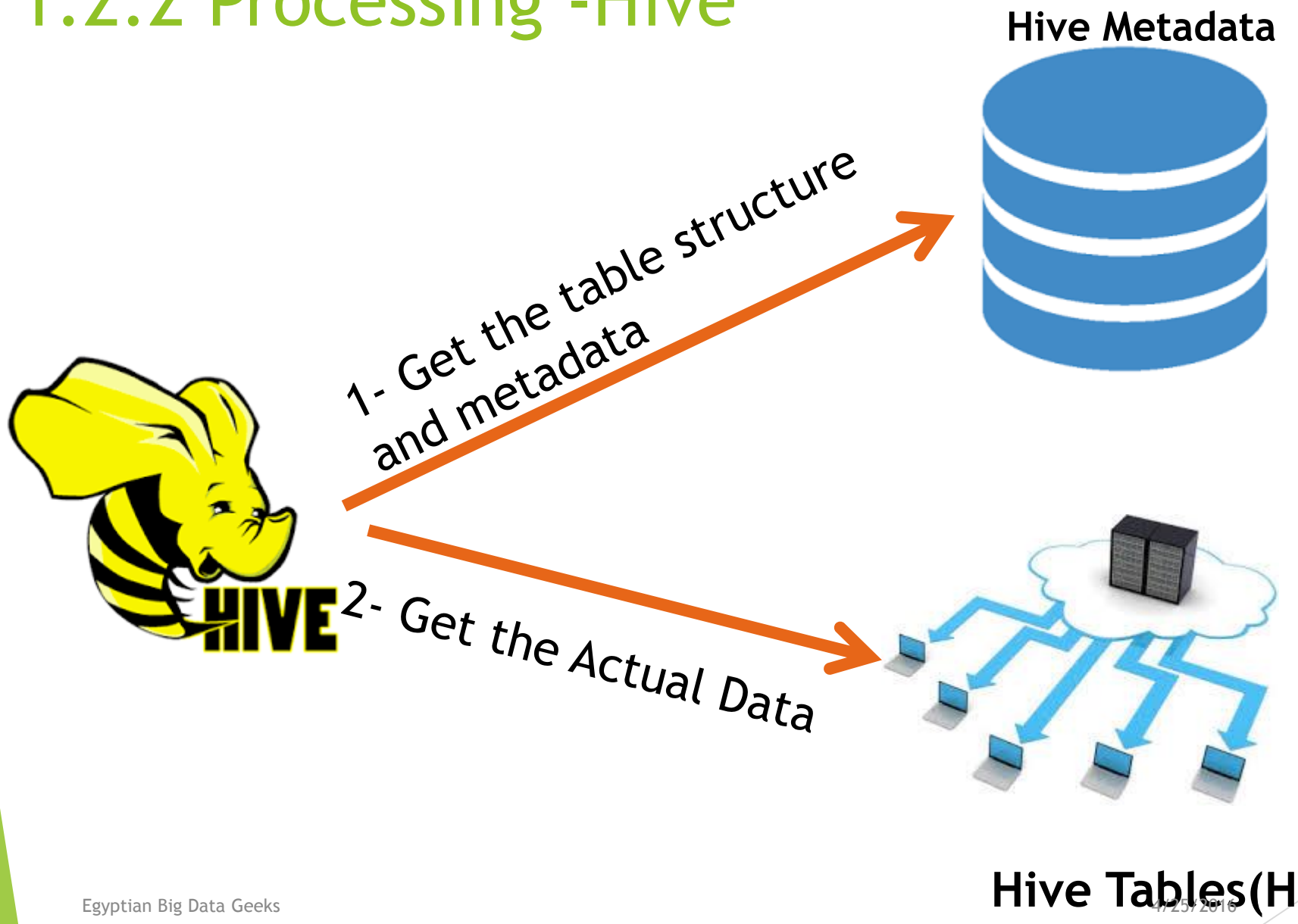
- Hive Example

```
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;
```

1.2.2 Processing -Hive

- How Hive Loads and Stores Data?
  - Hive's queries operate on tables, just like in an RDBMS.
  - A table is simply an HDFS directory containing one or more files – Default path: /user/hive/warehouse/<table_name>
  - Hive supports many formats for data storage and retrieval.
- How does Hive know the structure and location of tables?
  - These are specified when tables are created.
  - This metadata is stored in Hive's metastore. Contained in an RDBMS such as MySQL.

1.2.2 Processing -Hive



1.2.2 Hive

```
CREATE TABLE jobs
(id INT, title STRING,
    salary INT, posted TIMESTAMP )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

Data Example

```
1,Data Analyst,100000,2013-06-21 15:52:03
```

1.2.3 Spark

1.2.3 Spark

- What is Spark?
  - Apache Spark is a fast and general engine for large--scale data processing.
  - Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation that has maintained it since.
  - Written in Scala.
  - Spark Shell
    - Interactive – for learning or data exploration.
    - Python or Scala.

1.2.3 Spark

- Why Spark?
  - Faster
    - Cutting down on the number of reads and writes to the disc.
    - In Spark the concept of RDDs (Resilient Distributed Datasets) lets you save data on memory and preserve it to the disc if and only if it is required.
  - Easy Management.
  - Spark Streaming –Real Time Method to Process Streams.
  - Caching.
  - Recovery.
  - Iterative computations.

RDD (Resilient Distributed Dataset)

- RDD (Resilient Distributed Dataset)
  - Resilient – if data in memory is lost, it can be recreated.
  - Distributed – processed across the cluster.
  - Dataset – initial data can come from a file or be created programmatically.
- RDDs are the fundamental unit of data in Spark.
- Most Spark programming consists of performing operations on RDDs.

Creating RDD

- Three ways to create an RDD
  - From a file or set of files.
  - From data in memory.
  - From another RDD.

```
val mydata = sc.textFile("purplecow.txt")
```

Word Count Example in Spark

```
val counts = textFile.flatMap(line => line.split(" "))
                     .map(word => (word, 1))
                     .reduceByKey(_ + _)
```

Spark Operations

- Transformations.
  - Transformations – define a new RDD based on the current one(s).
    - map(function)–creates a new RDD by performing a function on each record in the base RDD.

```
map(line => line.toUpperCase)
```

```
egyptian big data geeks 1st meeting
----
EGYPTIAN BIG DATA GEEKS 1ST MEETING
```

Spark Operations

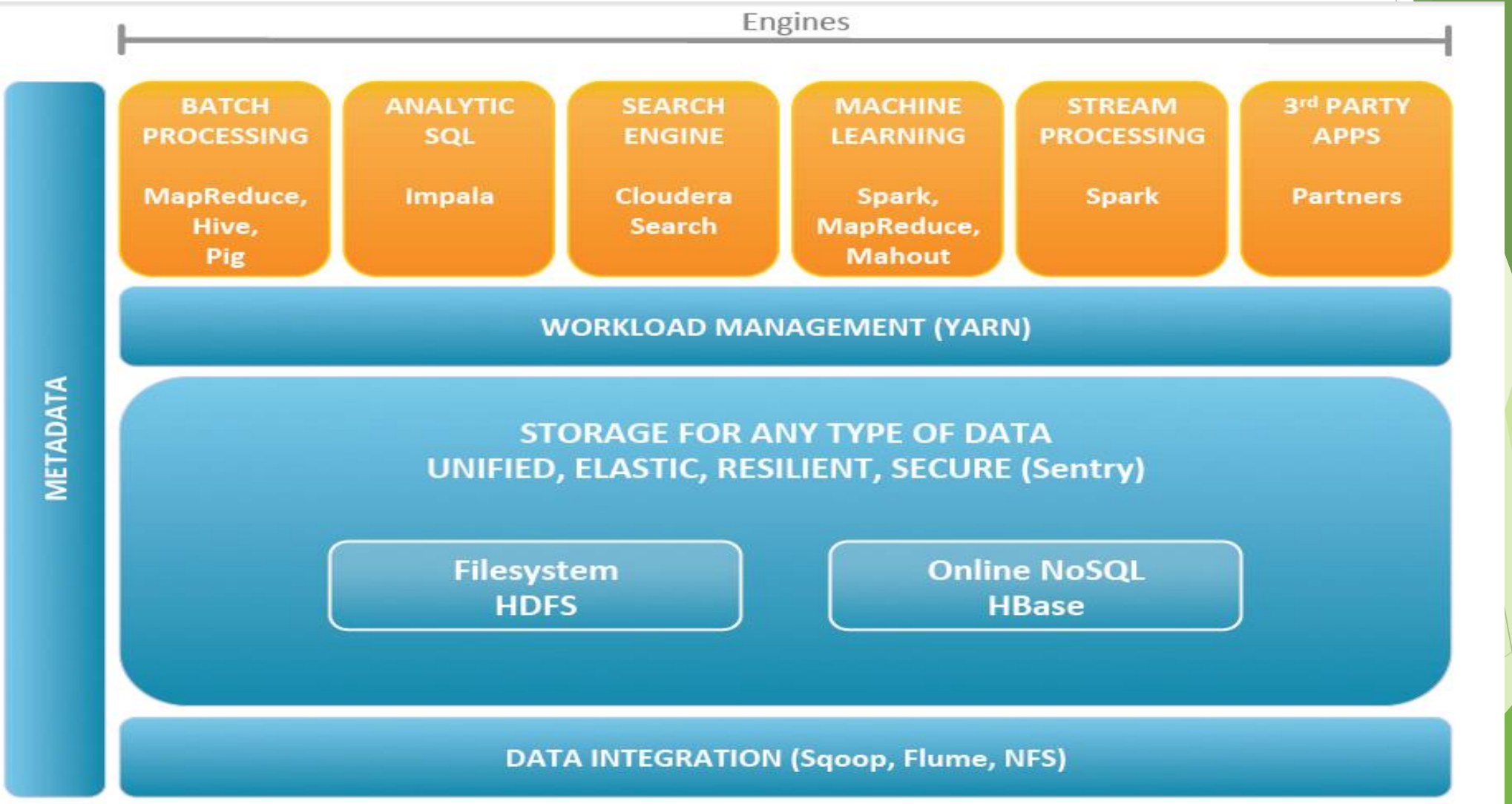
- Actions.
  - Actions–return values.

```
mydata.count()
take(n)
saveAsTextFile(file)
```

Spark Opertaions

- Lazy Execution.
- Lineage.
- Pipelining.

1.2 Stack


Engines
BATCH PROCESSING
MapReduce, Hive, Pig
ANALYTIC SQL
Impala
SEARCH ENGINE
Cloudera Search
MACHINE LEARNING
Spark, MapReduce, Mahout
STREAM PROCESSING
Spark
3rd PARTY APPS
Partners
WORKLOAD MANAGEMENT (YARN)
METADATA
STORAGE FOR ANY TYPE OF DATA
UNIFIED, ELASTIC, RESILIENT, SECURE (Sentry)
Filesystem
HDFS
Online NoSQL
HBase
DATA INTEGRATION (Sqoop, Flume, NFS)

YARN [Yet Another Resource Nevigator] Resource Management

- YARN is the Hadoop processing layer that contains
  - A resource manager.
  - A job scheduler.
- YARN allows multiple data processing engines to run on a single Hadoop cluster
  - Batch programs (e.g. Spark, MapReduce)
  - Interactive SQL (e.g. Impala)
  - Advanced analytics (e.g. Spark, Impala)
  - Streaming (e.g. Spark Streaming)

References.

Hadoop the definitive guide 4th edition.

Cloudera Training for Developers.

Wikipedia