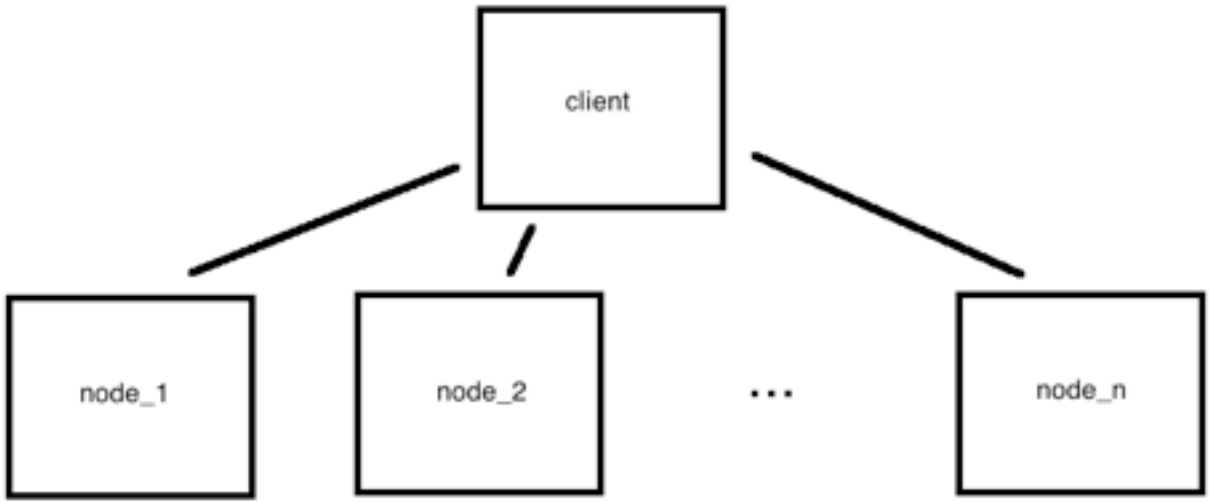
GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 17: MORE DISTRIBUTED SYSTEMS

I. BIG DATA

We can visualize this horizontal cluster architecture as a single client-multiple server relationship



How do we process data in a distributed architecture?

- *move code to data*
  - *map-reduce → less overhead (network traffic, disk I/O)*

"Computing nodes are the same as storage nodes."

II. HADOOP ECOSYSTEM

Hadoop *is a popular open-source Java-based implementation of the map-reduce framework (including file storage for input/output).*

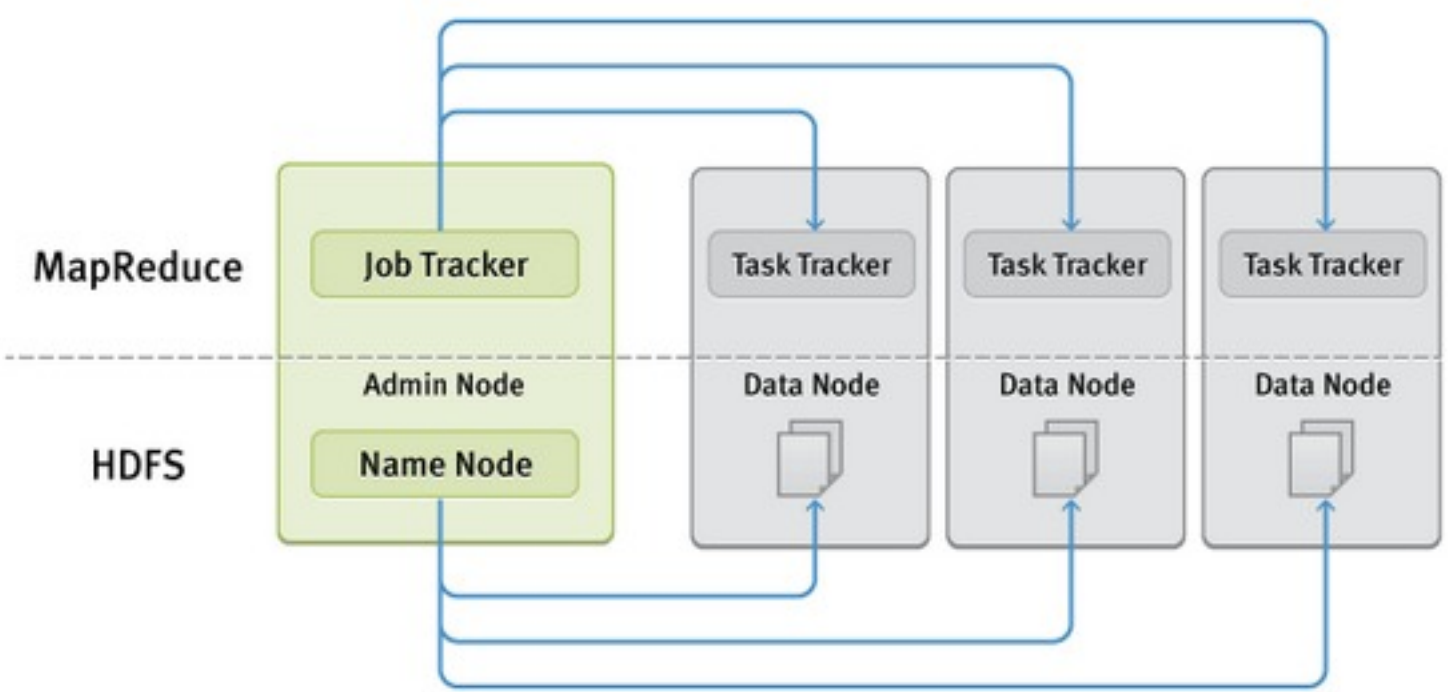
Google	Open-source	Function
GFS	HDFS	Distributed file system
MapReduce	MapReduce	Batch distributed data processing
Bigtable	HBase	Distributed DB/key-value store
Protobuf/Stubby	Thrift or Avro	Data serialization/RPC
Pregel	Giraph	Distributed graph processing
Tenzing	Hive	Scalable SQL on MapReduce
Dremel/F1	Cloudera Impala	Scalable interactive SQL (MPP)
FlumeJava	Crunch	Abstracted data pipelines on Hadoop

Hadoop *is a popular open-source Java-based implementation of the map-reduce framework (including file storage for input/output).*

More often, Hadoop refers to the ecosystem of tools around distributed computing with two main components:

- distributed filesystem (HDFS)

- map-reduce job scheduler


MapReduce
HDFS
Job Tracker
Admin Node
Name Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node

HDFS benefits include:

- *Push compute tasks to data nodes to avoid data transfer*

- *Data replication: data is replicated so if a single machine fails another still contains the data*

The map-reduce framework handles a lot of messy details for you:

- parallelization & distribution (eg, input splitting)

- partitioning (shuffle/sort/redirect)

- fault-tolerance (fact: tasks/nodes will fail!)

- I/O scheduling

- status and monitoring

Apache Hive

- *SQL language to query data on HDFS*
- *Queries are translated behind the scenes into map-reduce jobs*
- *Data is stored on HDFS, but a metadata database contains the table schemas*

Cloudera Impala

- ***ANOTHER*** *SQL language to query data on HDFS*

- *Similar interface to Hive*

BUT:

- *Impala contains its own scheduling engine, queries are not translated map-reduce jobs*

    - *Leads to faster queries, but no fault tolerance*

III. MAP-REDUCE PROGRAMMING

As we've discussed, the map-reduce approach involves splitting a problem into subtasks and processing these subtasks in parallel.

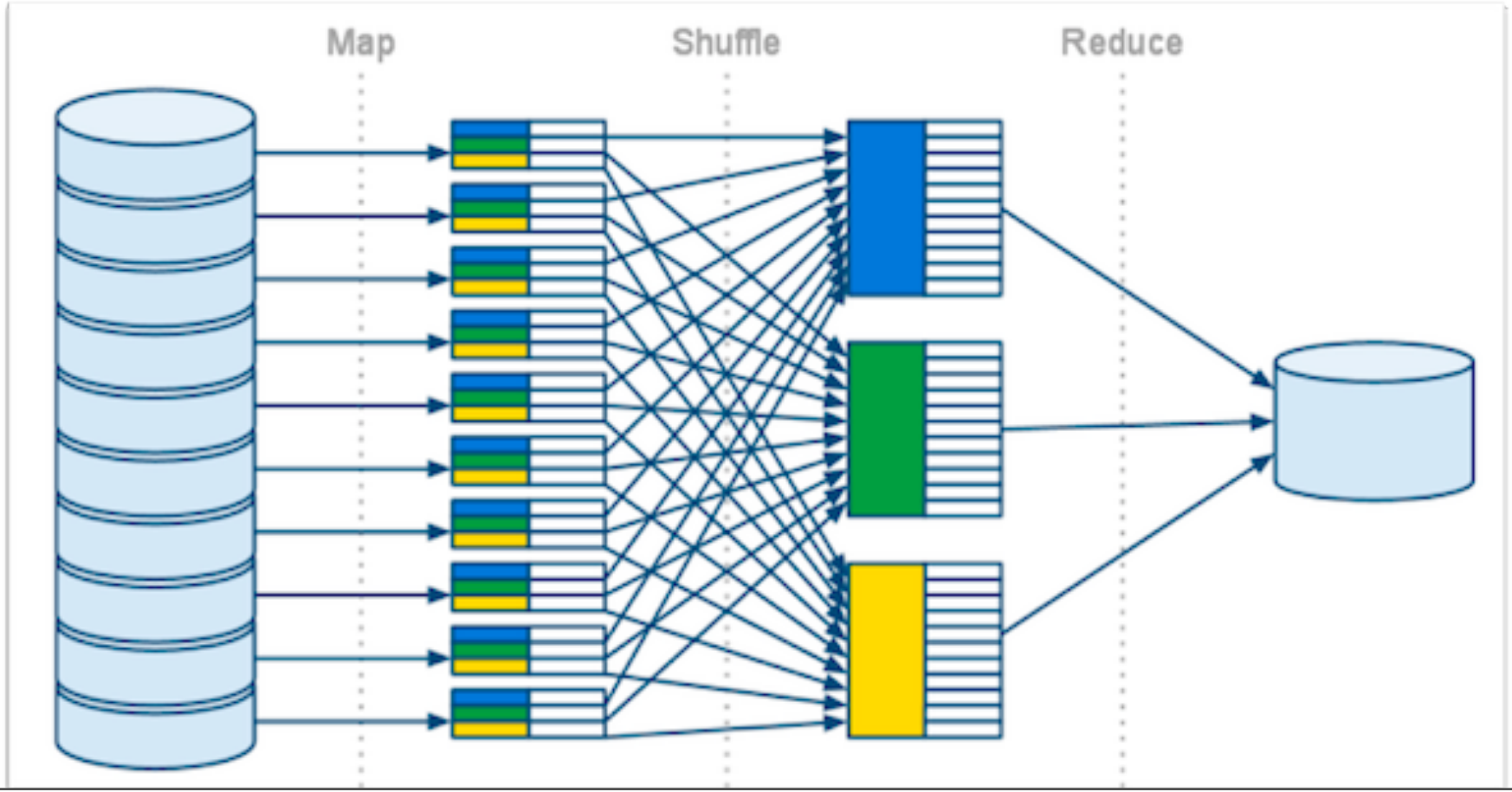
This takes place in two phases:

1) *the* **mapper** *phase*
2) *the* **reducer** *phase*

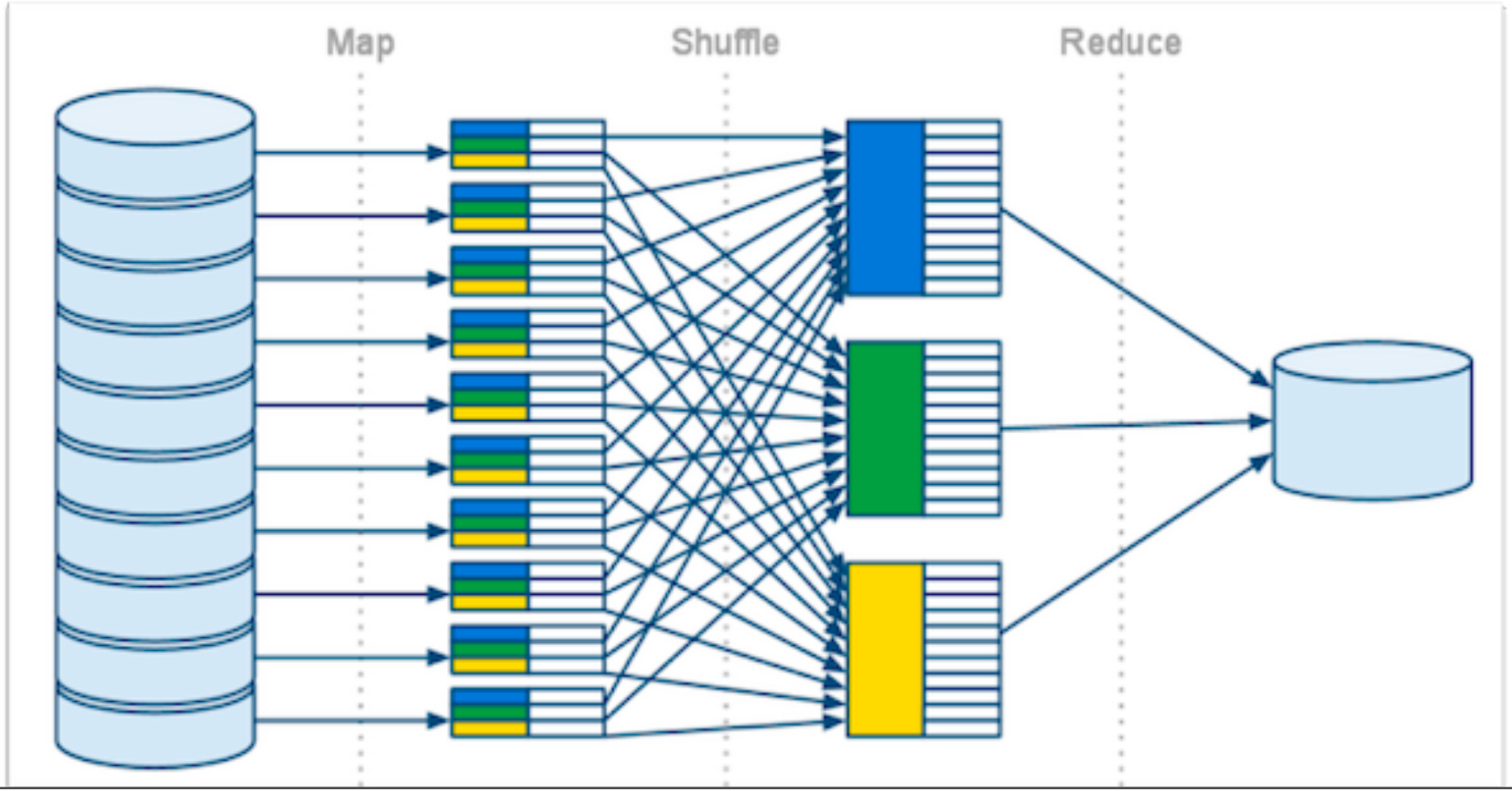
Map-reduce uses a functional programming paradigm. The data processing primitives are mappers and reducers, as we've seen.

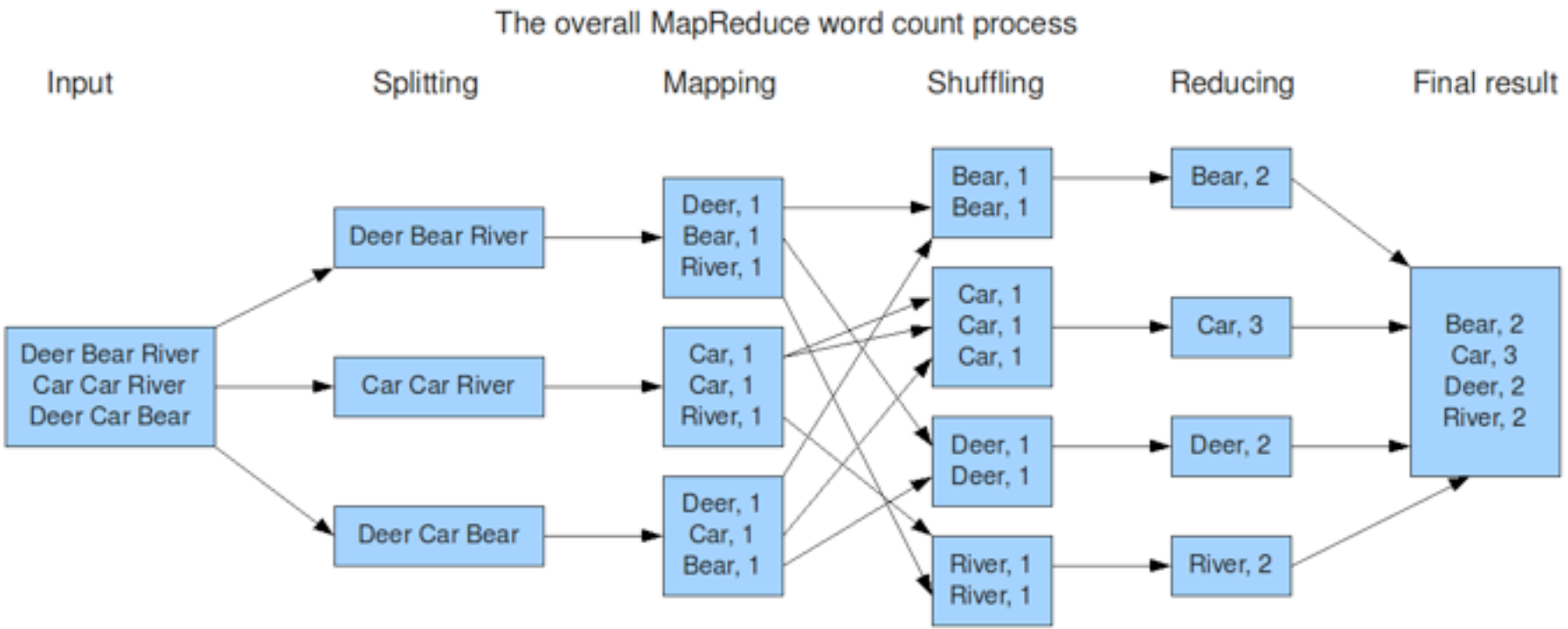
mappers – *filter & transform data*
reducers – *aggregate results*

MAP-REDUCE



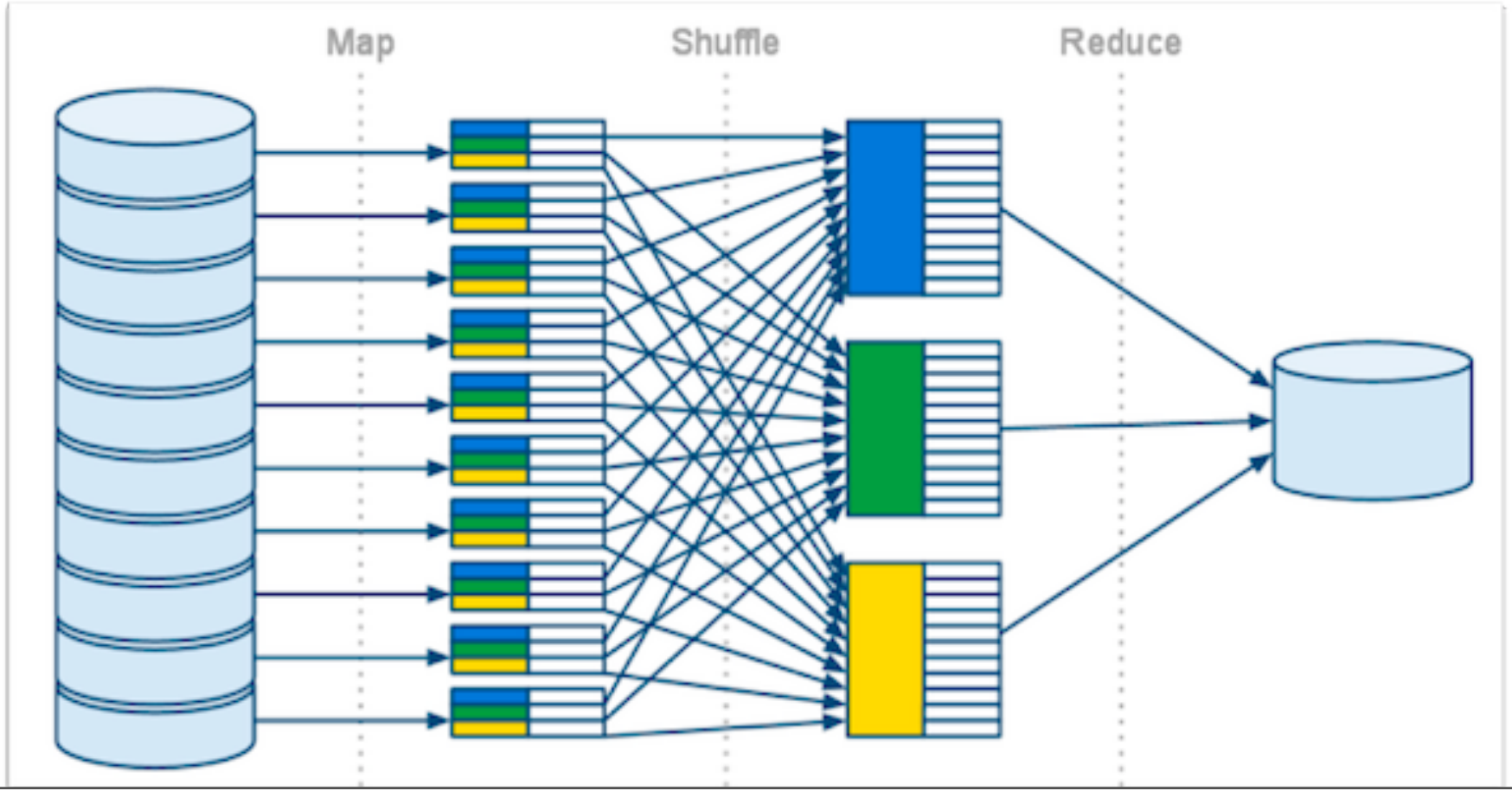
MAP-REDUCE




The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

IV. APACHE SPARK

MAP-REDUCE



Apache Spark is a project of of Berkeley AMPLab for:

> "fast and general engine for large-scale data processing"

Apache Spark is a project of of Berkeley AMPLab for:

"fast and general engine for large-scale data processing"

Goals:

- Support iterative algorithms
- Support interactive data mining
- Offer same advantages as Hadoop
  (HDFS, fault-tolerance, data locality)

The main primitive in Spark is the **RDD**

An RDD is a **Resilient Distributed Dataset**

The main primitive in Spark is the **RDD**

An RDD is a **Resilient Distributed Dataset**

RDDs are partitioned datasets with many supported manipulation operations:

- count, filter, groupBy, join, map, reduce

Transformations (define a new RDD)	map filter sample groupByKey reduceByKey sortByKey	flatMap union join cogroup cross mapValues
Actions (return a result to driver program)	collect reduce count save lookupKey	

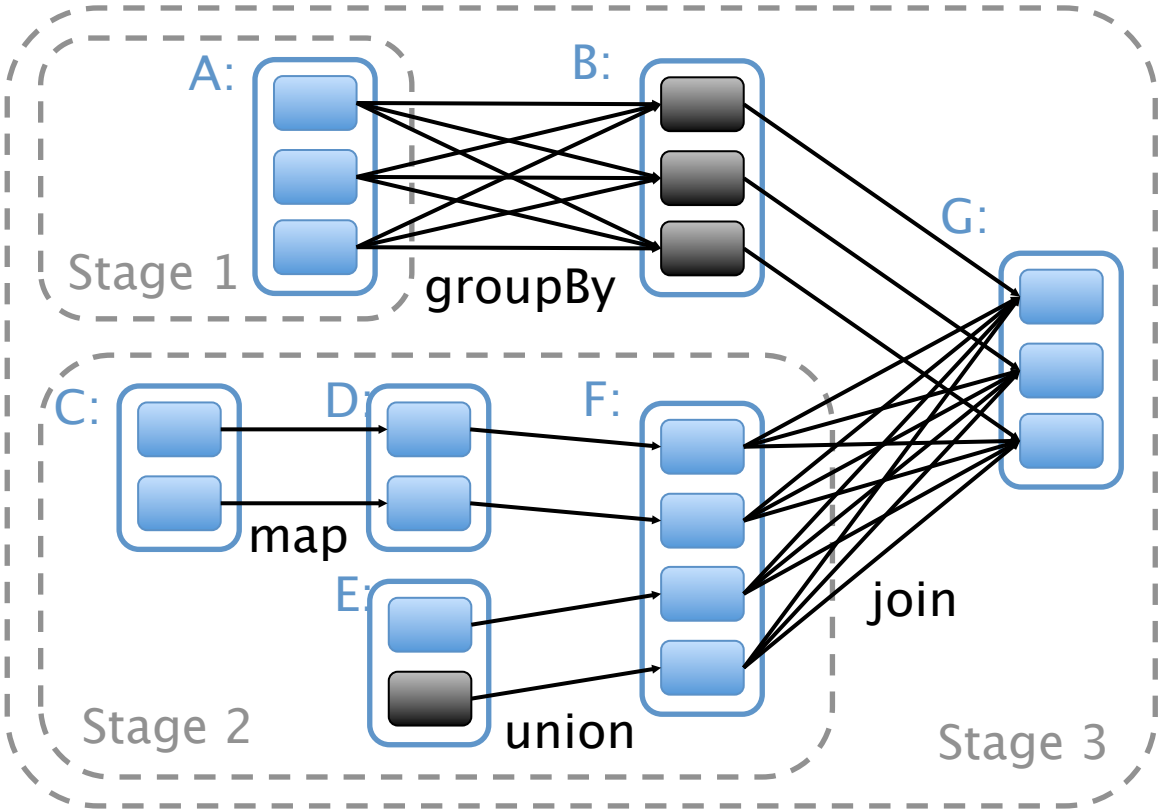
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()
```

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains(“foo”)).count
```

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
```


A:
B:
G:
Stage 1
groupBy
C:
D:
F:
map
E:
join
Stage 2
union
Stage 3

Other benefits of Spark:

Interactivity: *spark-shell*

Python support: *pyspark*

Spark Ecosystem:

Other benefits of Spark:

Interactivity: *spark-shell*

Python support: *pyspark*

Spark Ecosystem: