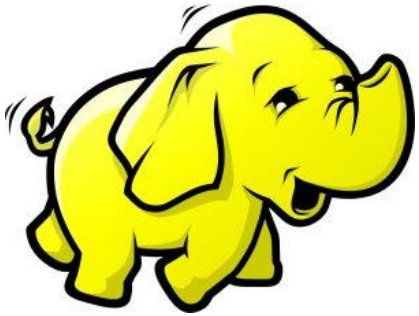
Cloud Computing

SEWP ZG527

BITS Pilani



Hadoop

Name of the elephant!!

Or have u thought of the following?

- How do “big bazaar/more/D’Mart” target promotions guaranteed to make you buy?

- How can Airtel(4G) increase Ad-campaign efficiency?

- What’s in your search? How is Google able to make such good predictions about your search?

- I have huge amount of data(nw sites) data(twitter) data(blog) data(feeds) data(forums) ? What do I do with it?

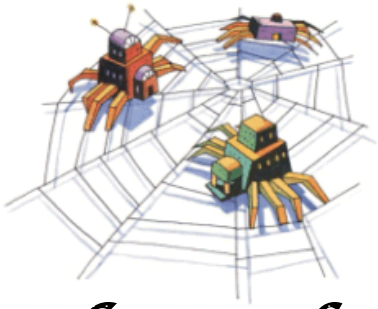
Wow, that's so much of DATA to process!!!

: exactly, and that what we call as "BIG Data"

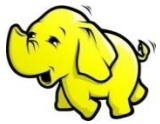
- Hadoop is one of the best-known cloud platforms for big data today
- It solves a specific class of data-crunching problems that frequently comes up in the domain of Internet computing and high-performance computing.
- Managing lots of information (growing by the day and doubling by year)
- Working with many new types of data (totally unstructured)

One of the research in the year 2012, Hadoop held the world record for the fastest system to sort large data (500 GB of data in59 sec and 100 terabytes of data in 68 seconds)

Designed to answer the question: **“How to process big data with reasonable cost and time?”**



Super, so tell me more about Hadoop, the data cruncher

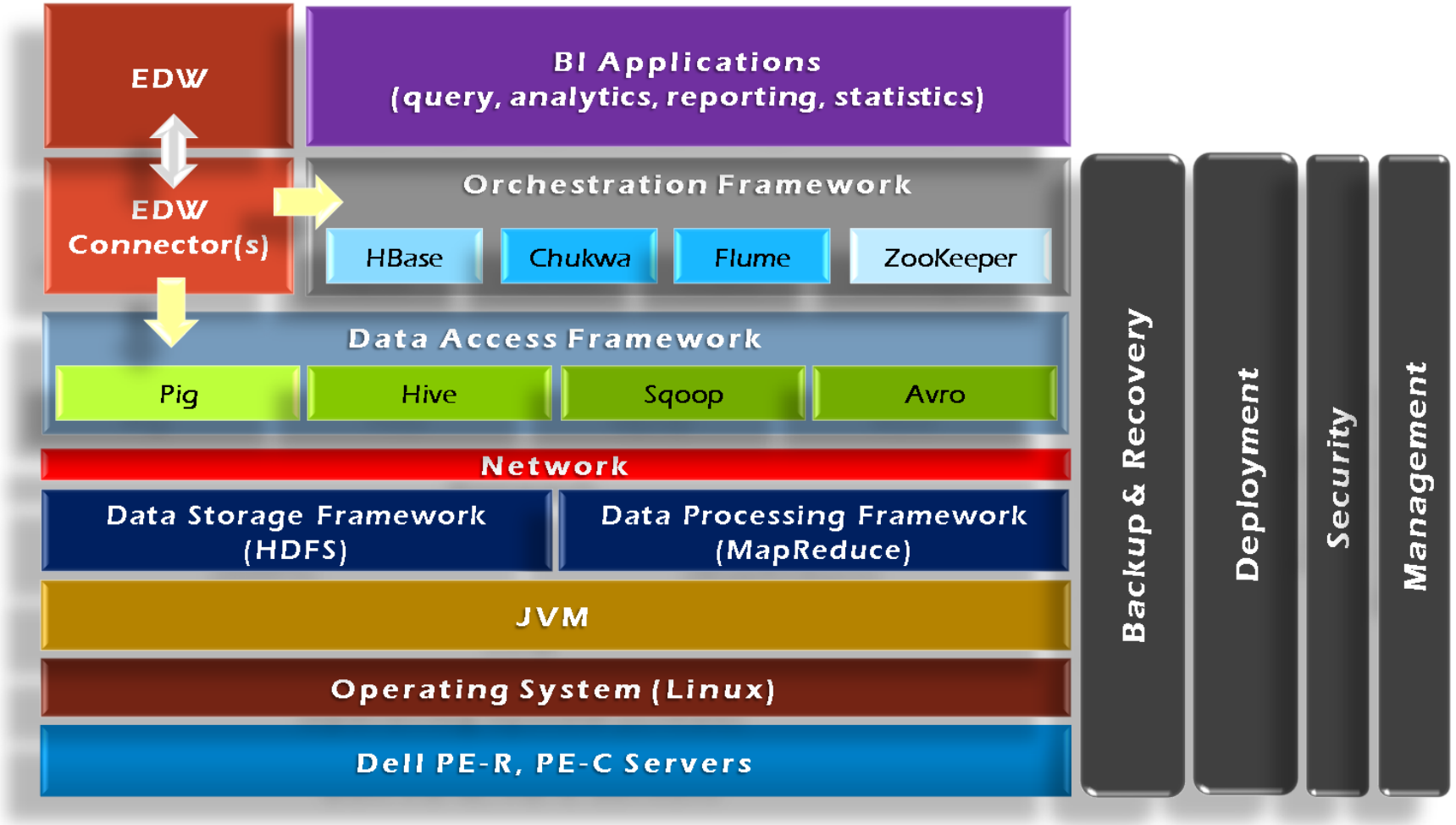


Okay, okay…. Sit tight,

Hadoop Features

- Hadoop is optimized for batch-processing applications, and scales to the number of CPUs available in the cluster
- Provides Framework for Massive parallel processing
- Programmer can focus on their program, and the framework takes care of the details of parallelization, fault-tolerance, locality optimization, load balancing
- **Paradigm shift:** In MapReduce programming model, computation goes to data rather than data coming to program. Processing takes place where data is.

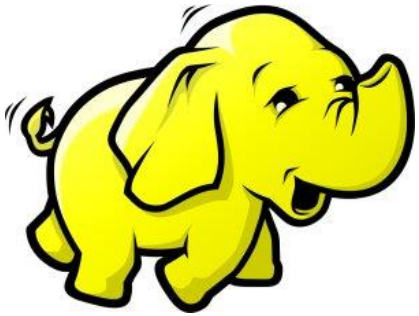
Hadoop Framework Tools



Cloud Computing

SEWP ZG527

BITS Pilani



Hadoop

Name of the elephant!!

Hadoop common Component

MapReduce – offline computing engine (Data Processing Framework)

HDFS – Hadoop Distributed file system (Data Storage Framework

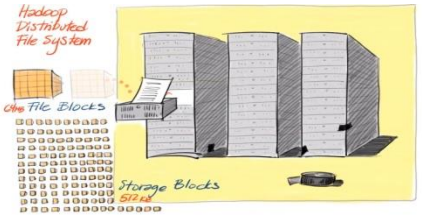
Frameworks like **Hbase**, **Pig** and **Hive** have been built on top of Hadoop.

- **Pig** is a dataflow language and execution environment over Hadoop.
- **Hbase** is a distributed key-value store which supports SQL-like queries similar to Google's BigTable
- **Hive** is a distributed data warehouse to manage data stored in the Hadoop File System.

MapReduce (Data Processing Framework)

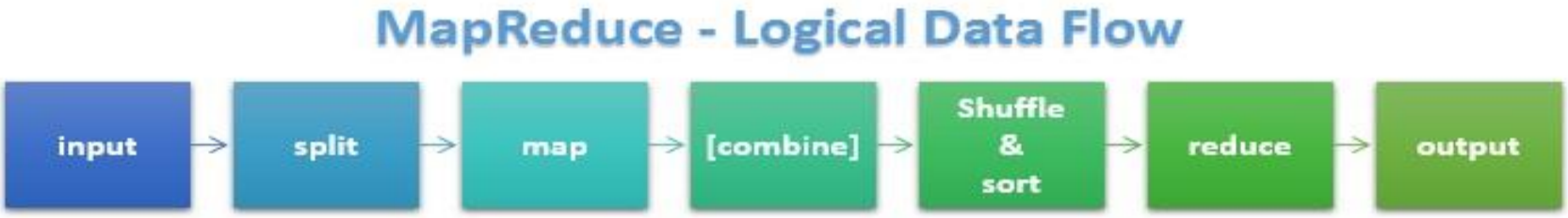
MapReduce

Software Framework for easily running applications	Processes large amount of data in parallel	Using large clusters having thousands of nodes	Nodes of commodity hardware	In a reliable and fault-tolerant manner

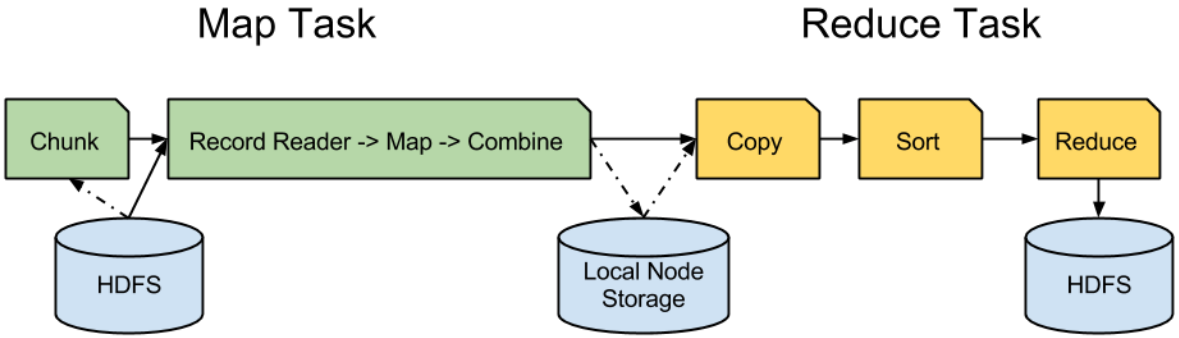


MapReduce Processing flow

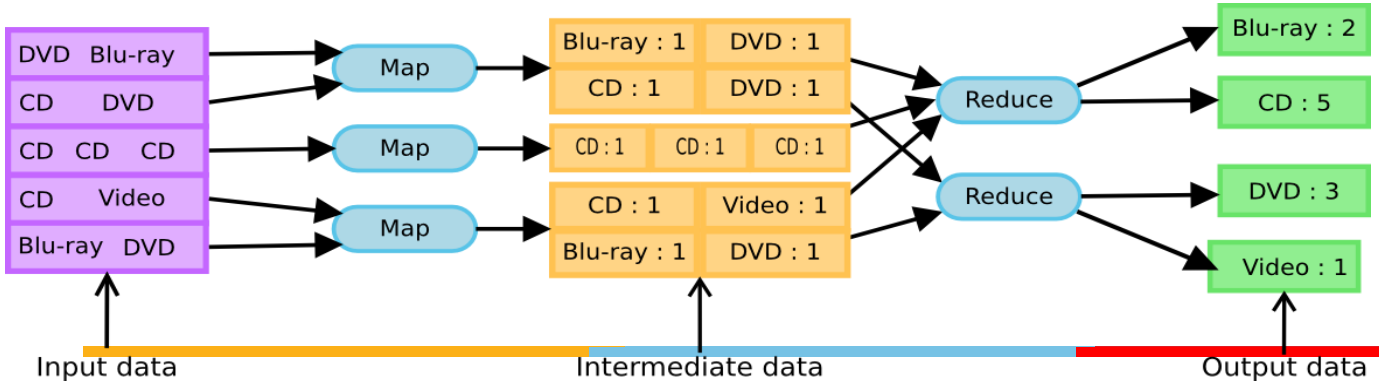
Logical flow:



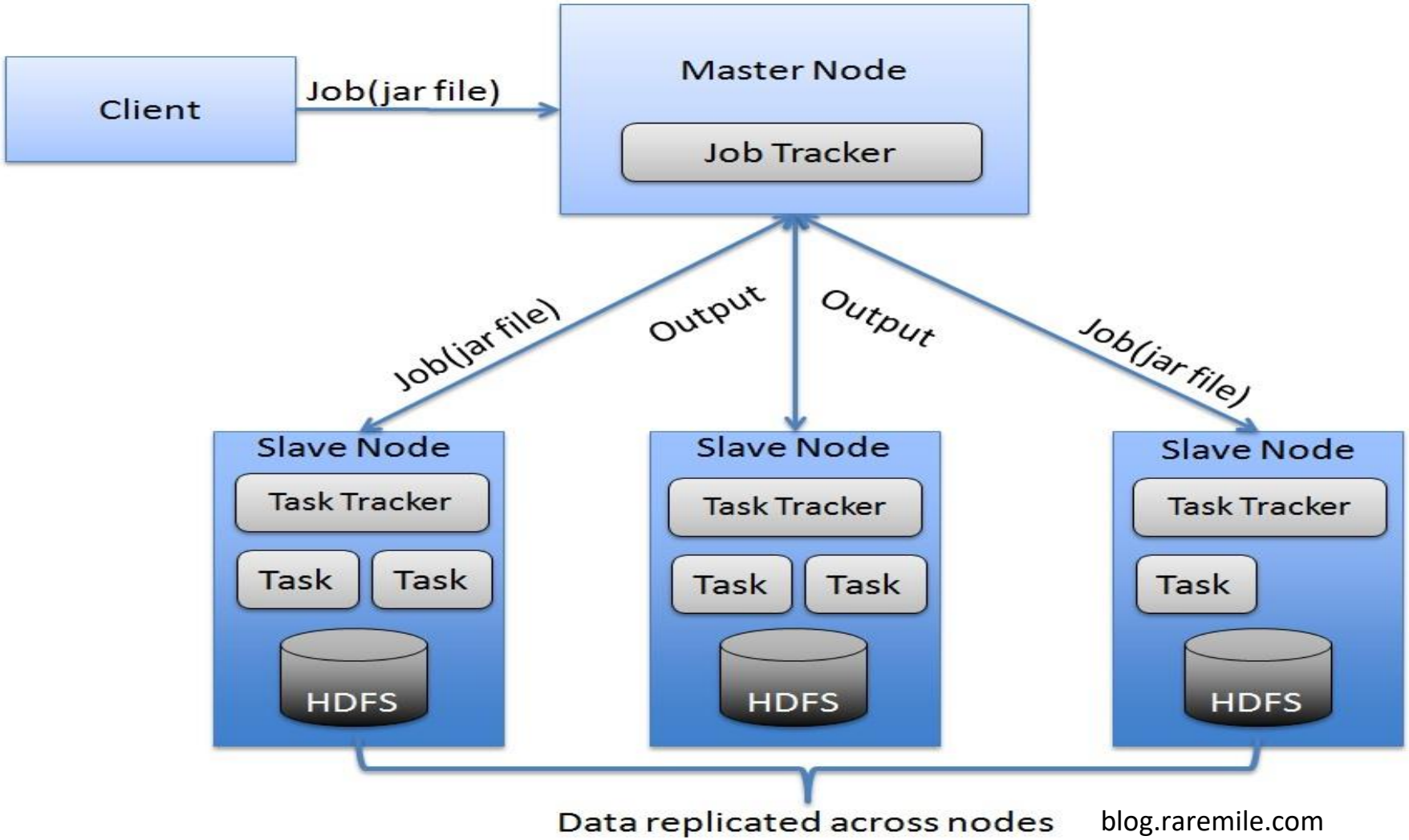
Two phases



Example:



Architecture Overview



blog.raremile.com

Architecture (cont.)

NameNode/ Master Node:

- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc.
- Handles creation of more replica blocks when necessary after a DataNode failure

DataNode/ Slave Node:

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere

Architecture (cont.)

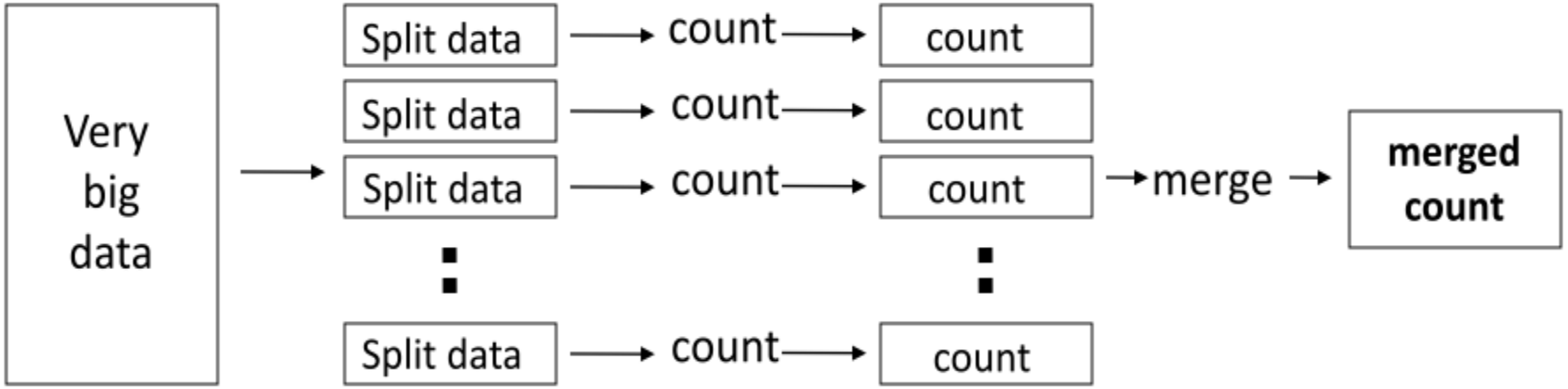
The Job Tracker:

- Central authority for the complete MapReduce cluster and responsible for scheduling and monitoring MapReduce jobs
- Responds to client request for job submission and status

The Task Tracker:

- Workers that accepts map and reduce tasks from job tracker, launches them and keeps track of their progress, reports the same to job tracker.
- Keeps track of resource usage of tasks and kills the tasks that overshoots their memory limits

Distributed Word Count



MapReduce Programming Model

Data type: key-value *records*

Map function:

$(K_{in}, V_{in}) \rightarrow list(K_{inter}, V_{inter})$

Reduce function:

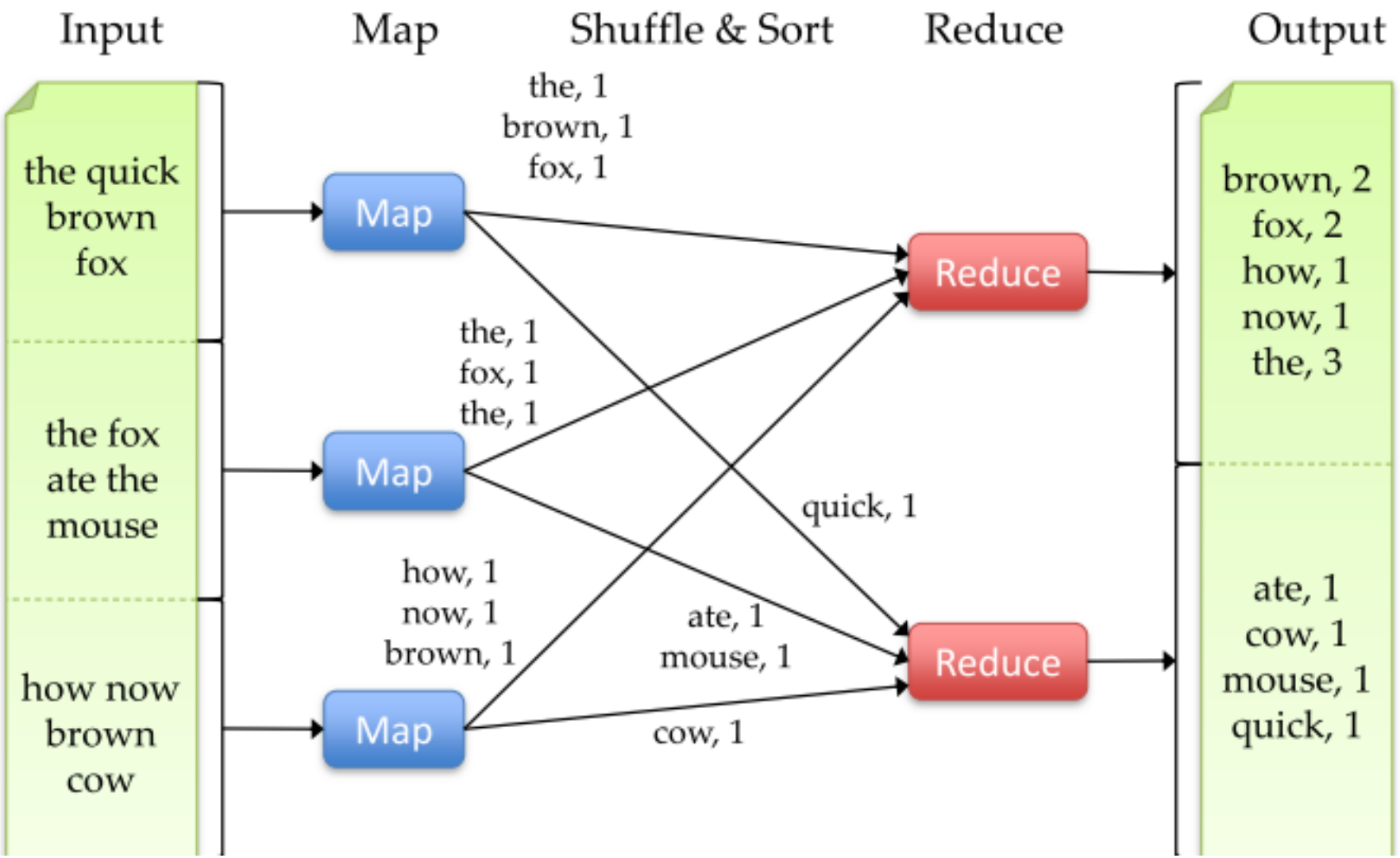
$(K_{inter}, list(V_{inter})) \rightarrow list(K_{out}, V_{out})$

Example: Word Count

```
def mapper(line):
    foreach word in line.split():
        output(word, 1)


def reducer(key, values):
    output(key, sum(values))
```

Word Count Execution



An Optimization: The Combiner

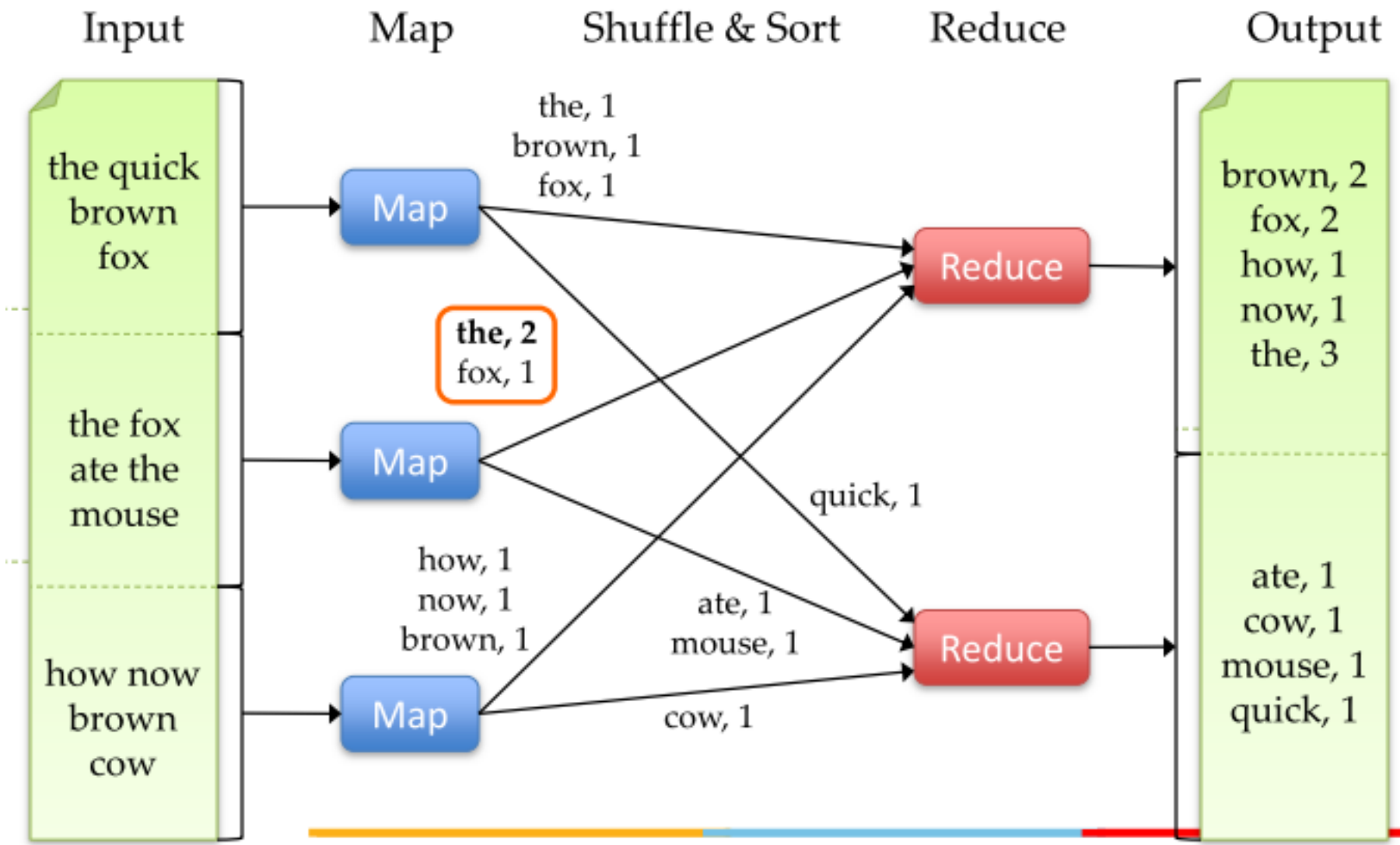
- Local reduce function for repeated keys produced by same map
- Decreases amount of intermediate data

• Example: local counting for Word Count:

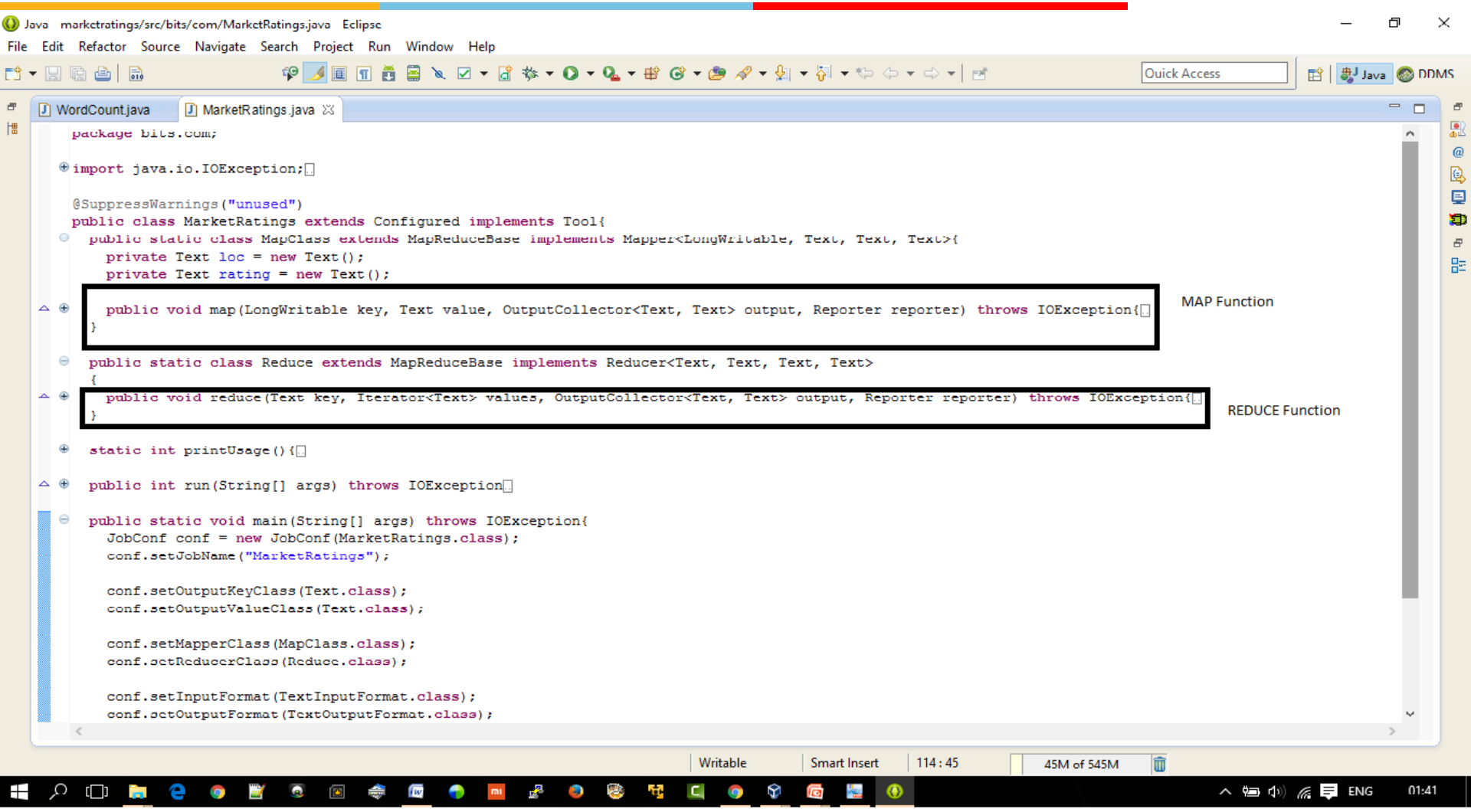
An Optimization: The Combiner

```
def combiner(key, values):
output(key, sum(values))
```

Word Count with Combiner



Snapshot of MarketRatings example and Program demo



MapReduce Execution Details

Mappers preferentially scheduled on same node or same rack as their input block

- Minimize network use to improve performance

Mappers save outputs to local disk before serving to reducers

- Allows recovery if a reducer crashes
- Allows running more reducers than # of nodes

Fault Tolerance in MapReduce

1. If a task crashes:
   – Retry on another node
      • OK for a map because it had no dependencies
      • OK for reduce because map outputs are on disk
   – If the same task repeatedly fails, fail the job or ignore that input block
2. If a node crashes:
   – Relaunch its current tasks on other nodes
   – Relaunch any maps the node previously ran
      • Necessary because their output files were lost along with the crashed node

Fault Tolerance in MapReduce

3. If a task is going slowly (straggler):
   – Launch second copy of task on another node
   – Take the output of whichever copy finishes first, and kill the other one
      • Critical for performance in large clusters

Challenges of Cloud Environment

Cheap nodes fail, especially when you have many
– Mean time between failures for 1 node = 3 years
– MTBF for 1000 nodes = 1 day
– Solution: Build fault tolerance into system

Commodity network = low bandwidth
– Solution: Push computation to the data

Programming distributed systems is hard
– Solution: Restricted programming model: users write data-parallel “map” and “reduce” functions and system handles work distribution and failures

Hadoop

Thanks Hadoop