RDBMS, MapReduce and Spark

- With MapReduce, process data 10x faster than RDBMS
- With Spark, process data 10x faster than MapReduce

Features of MapReduce

- Simple programming model
  - No system programming (OS)
- Fault Tolerant (Reliable)
- Scalable

- Popular → Already in use across many projects

Limitations of MapReduce

- MapReduce applications are high latency jobs
- Doesn’t go well for ML type applications (applications that are iterative)
- Too many disk read writes (intermediate phases)
- No capabilities for processing live streams of data (MapReduce is a batch processing engine)
- Mostly Java was used as the programming language
- No Interactive Environment

Spark Research

- https://spark.apache.org/research.html

What is Spark?

- A general purpose, large scale, unified data processing engine
- Polyglot → Spark jobs can be written in Python, Scala, R, Java and SQL
- Spark offers capabilities to process live data streams in near real time
- Spark offers libraries for implementing machine learning
- Spark offers in-memory computation capabilities

How to launch a Spark Application?

Launch pyspark

```
$ pyspark

>>> sc

# sc --> The Spark Context object (Connection to the Spark cluster)
```

Getting started with RDDs

```
>>> sc.setLogLevel("ERROR")

>>> x = sc.textFile("/user/cloudera/Stocks")
>>> x.collect()
>>> x.take(10)
>>> x.first()

>>> for i in x.take(10): print(i)

>>> y = x.first()
>>> y
>>> type(y)
>>> y.split()
>>> y.split(',')
>>> y.split(',')[1]

# Get distinct stock symbols

>>> z = x.map(lambda y: y.split(',')[1]).distinct()
>>> z.collect()

>>> for i in z.collect(): print(i)
>>> z.count()
```

A simple Spark program

Get distinct stocks

```
# Get distinct stock symbols

>>> stocksRDD = sc.textFile("/user/cloudera/Stocks")

>>> stockSymbolRDD = stocksRDD.map(lambda y: y.split(',')[1]).distinct()

>>> stockSymbolRDD.collect()
```

Get maximum close price per stock symbol

```
# Get maximum close price per stock symbol
>>> stocksRDD = sc.textFile("/user/cloudera/Stocks")
>>> stockSymbolCloseRDD = stocksRDD.map(lambda y: (y.split(',')[1], float(y.split(',')[6])))
>>> maxClosePriceRDD = stockSymbolCloseRDD.reduceByKey(lambda a, b: round(max(a, b)))
>>> maxClosePriceRDD.collect()
```

RDD Examples

```
>>> x = sc.parallelize([(1, 2), (3, 4)])
>>> y = x.keys()
>>> y.collect()

>>> y = x.values()
>>> y.collect()

>>> x = sc.parallelize([1,2,3,4,5])
>>> y = sc.parallelize([3,4,5,6,7])

>>> z = x.union(y)
>>> z.collect()

>>> z = x.intersection(y)
>>> z.collect()

>>> z = x.subtract(y)
>>> z.collect()

>>> x = sc.parallelize([2,4,1])
>>> x.max()
>>> x.sum()
>>> x.mean()
>>> x.stdev()
>>> sc.parallelize([1, 2, 3]).variance()
>>> sc.parallelize([1, 2, 3]).stats()
```

RDD Examples

```
>>> x = sc.parallelize([("a", 1), ("b", 2)])
>>> y = sc.parallelize([("a", 3), ("a", 4), ("b", 5)])
>>> z = x.join(y)
>>> z.collect()

>>> x = sc.parallelize([1, 2])
>>> y = sc.parallelize([3, 4])
>>> z = x.cartesian(y)
>>> z.collect()
```

2 ways to create RDDs

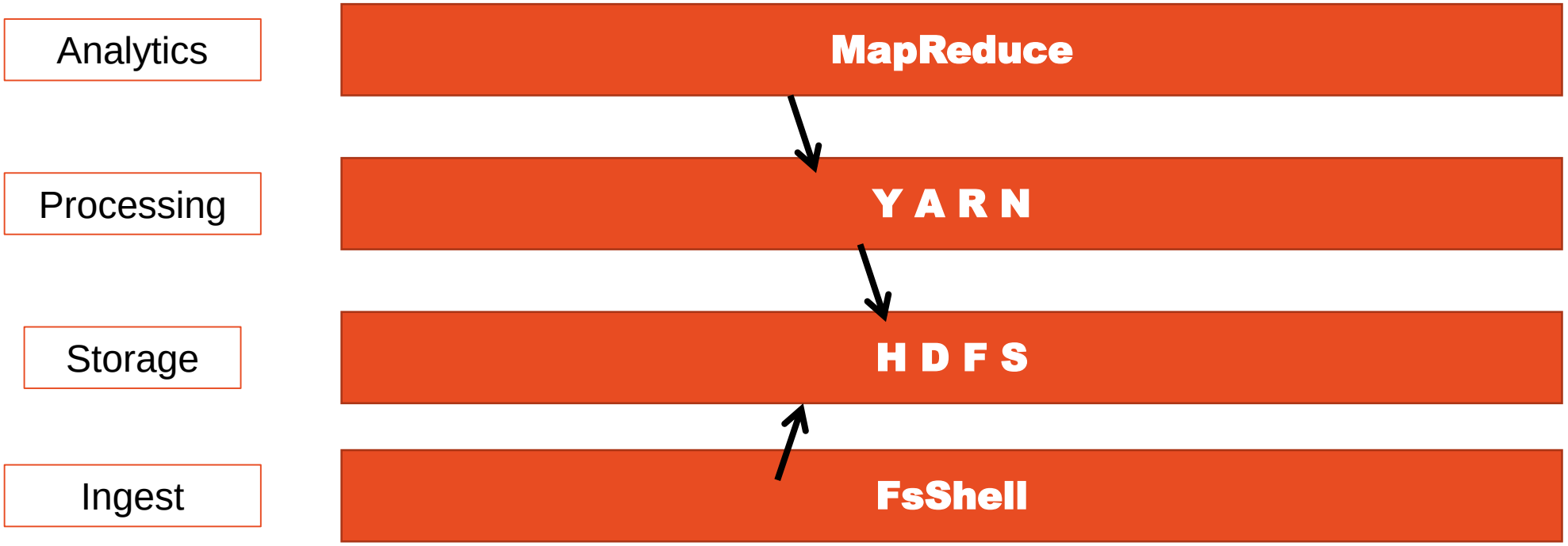
- sc.parallelize()
  - parallelize a collection
- sc.textFile()
  - reference data stored in an external storage system (Ex. HDFS)

RDDs Operations (2 types)

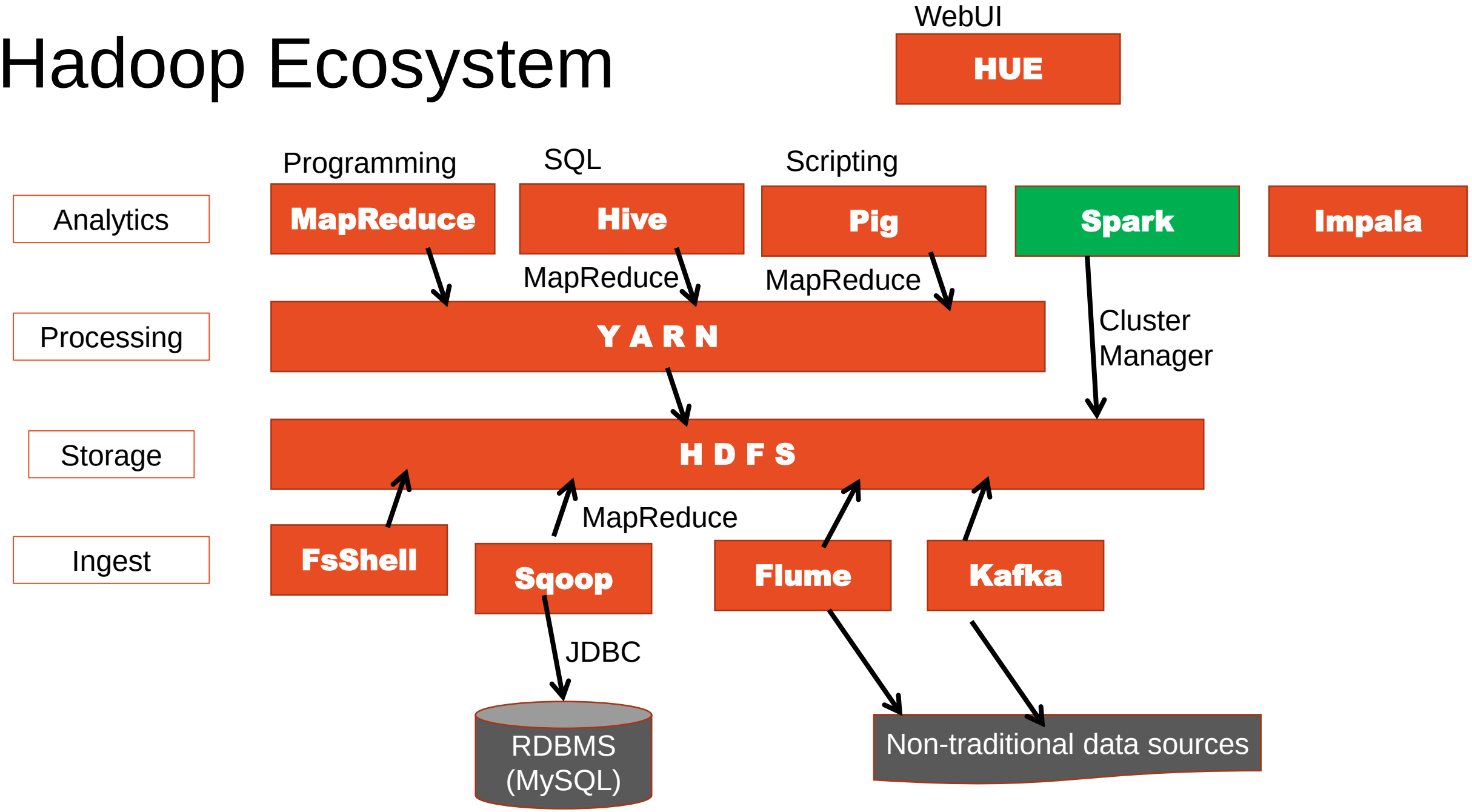
- Transformations
- Actions

- *All operations on RDDs are either 'Transformations' or 'Actions'*
- *RDDs are immutable*

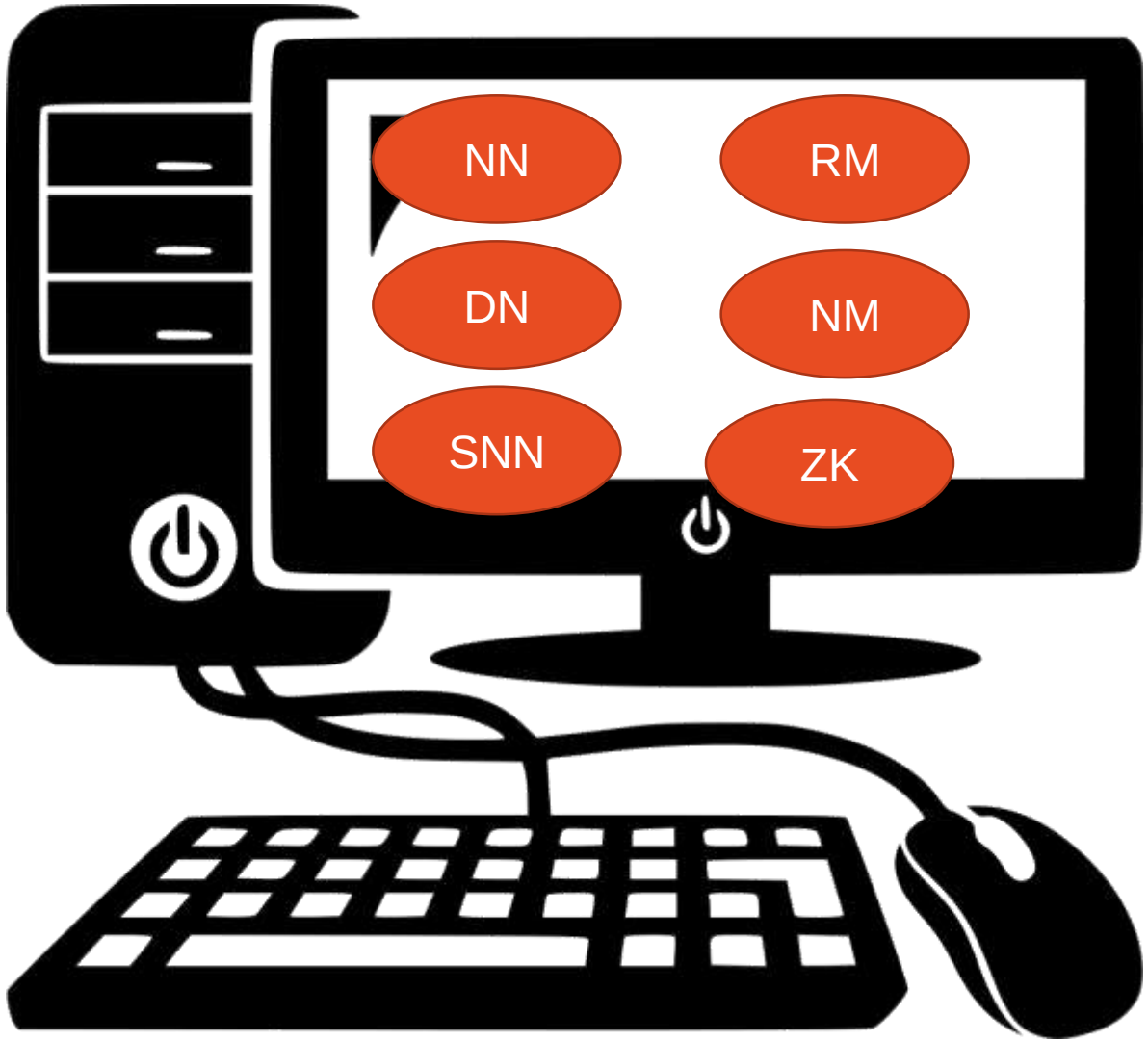
Core Hadoop

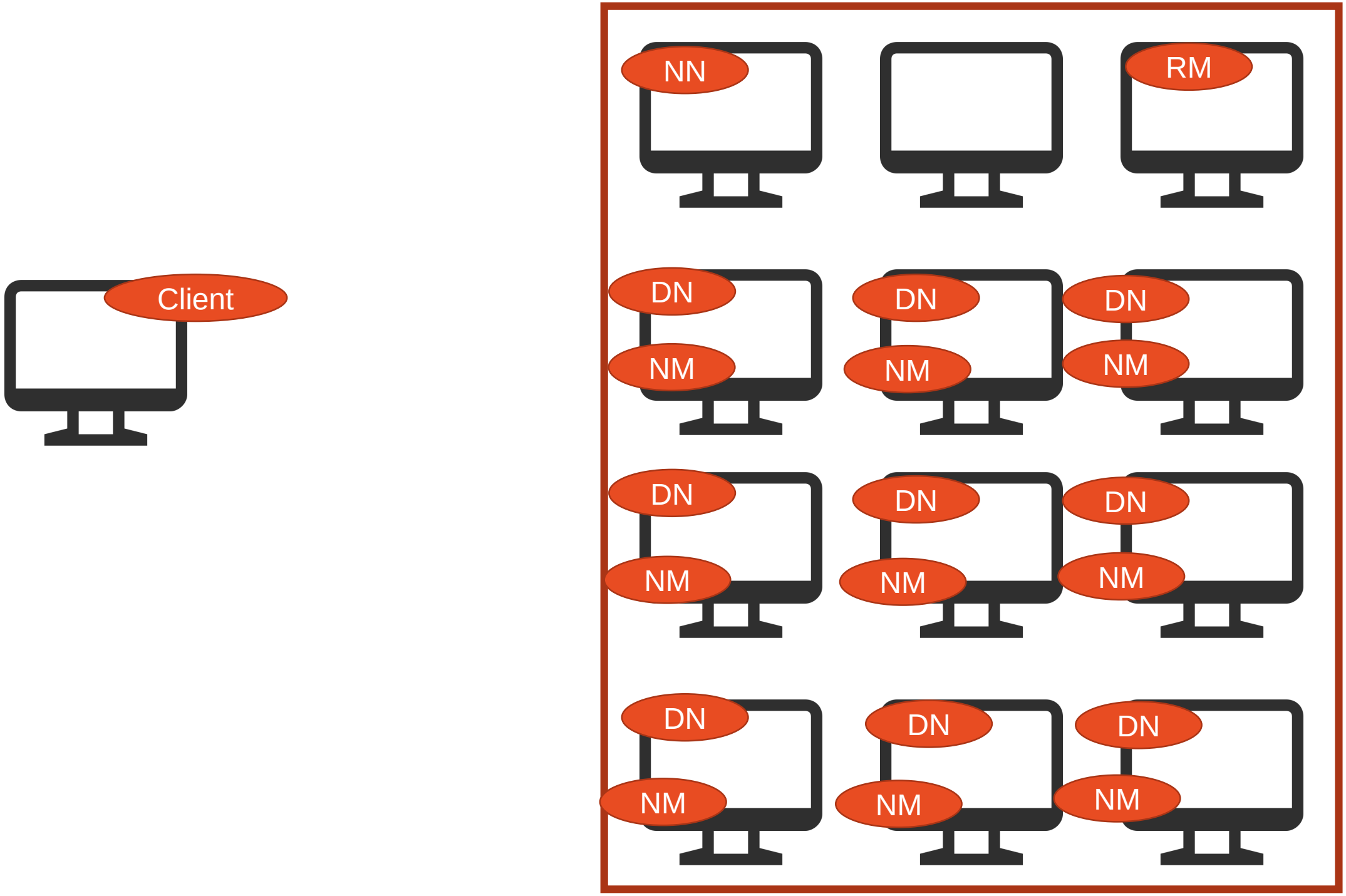

Analytics
MapReduce
Processing
Y A R N
Storage
H D F S
Ingest
FsShell

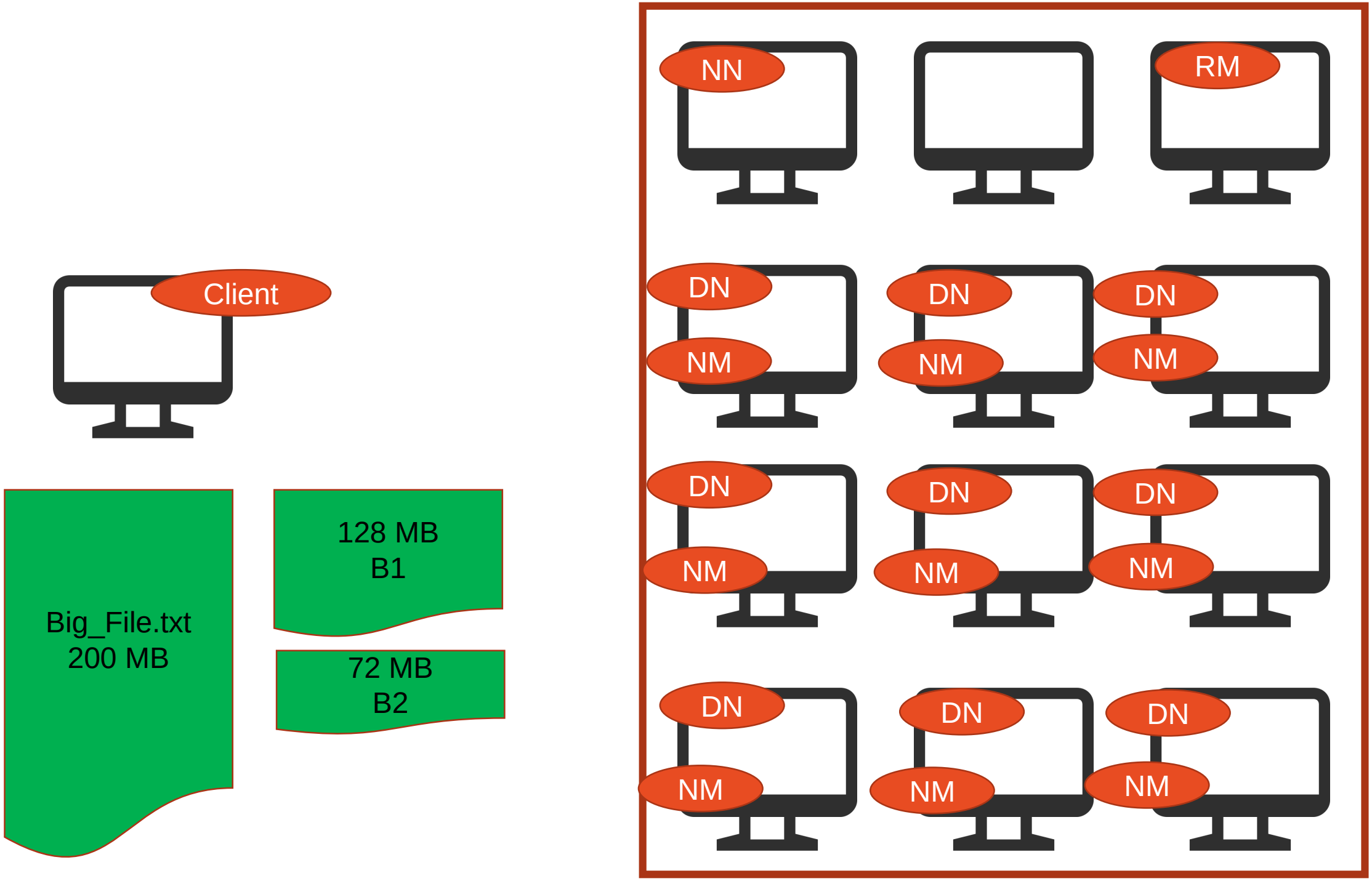
Hadoop Ecosystem



Hadoop Setup – Pseudo Distributed Mode




Client
NN
RM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM


Client
Big_File.txt
200 MB
128 MB
B1
72 MB
B2
NN
RM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM
DN
NM

RDD Examples

```
$ hadoop fs -mkdir orders_data

$ hadoop fs -put /home/cloudera/Downloads/orders orders_data/

# Count records in the orders dataset

$ pyspark

>>> sc.setLogLevel("ERROR")

>>> ordersRDD = sc.textFile("/user/cloudera/orders_data")

>>> ordersRDD.count()
```

RDD Examples

```
# Get distinct order_status from the orders dataset

>>> ordersRDD.first()

>>> x = ordersRDD.first()

>>> x.split(',')

>>> x.split(',')[3]

>>> ordersRDD.map(lambda x: x.split(',')[3]).distinct().collect()
```

RDD Examples

```
# Get count by order_status

>>> from operator import add

>>> ordersRDD = sc.textFile("/user/cloudera/orders_data")

>>> ordersRDD.map(lambda x: (x.split(',')[3], 1)).reduceByKey(add).collect()
```

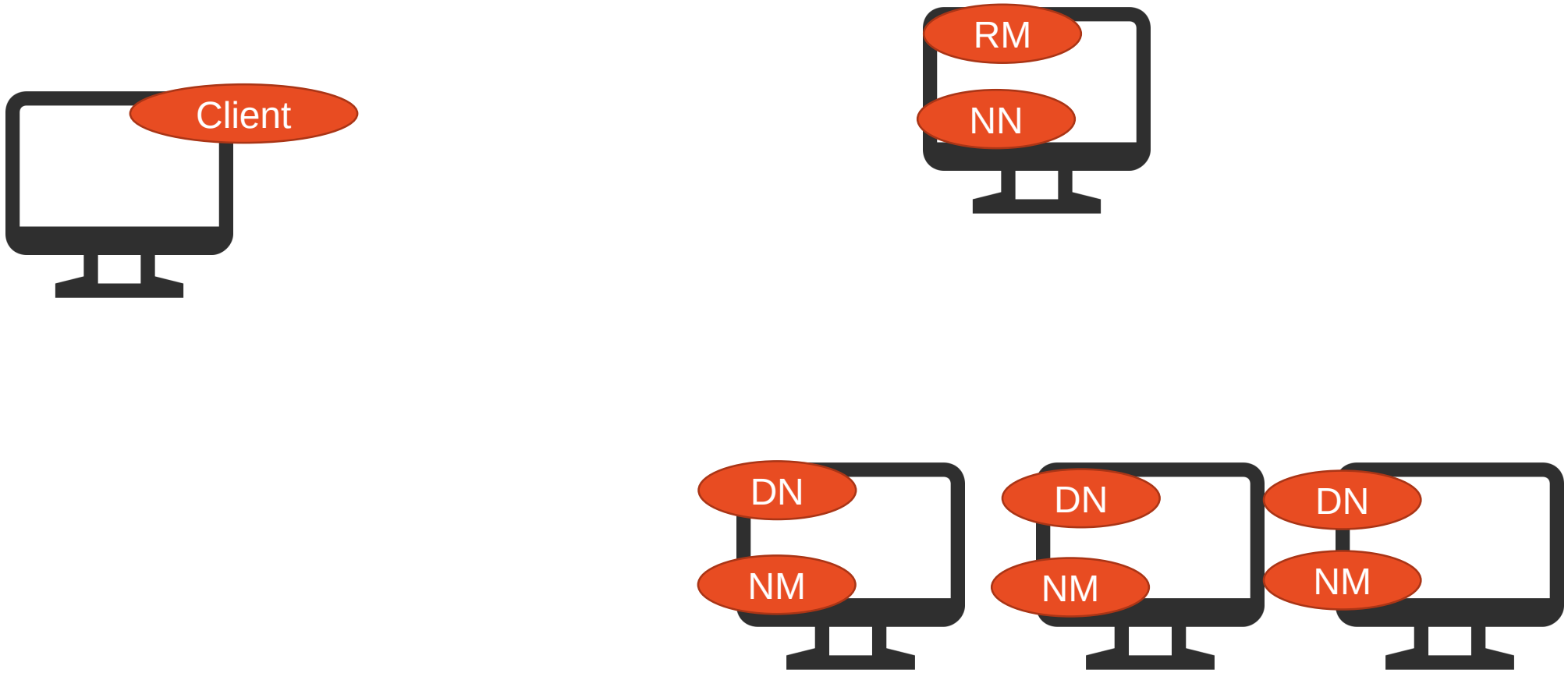
RDD Examples

```
# Get count of CLOSED and COMPLETED orders

>>> ordersRDD.filter(lambda x: (x.split(',')[3] == 'CLOSED' or x.split(',')[3]
== 'COMPLETE')).count()
```

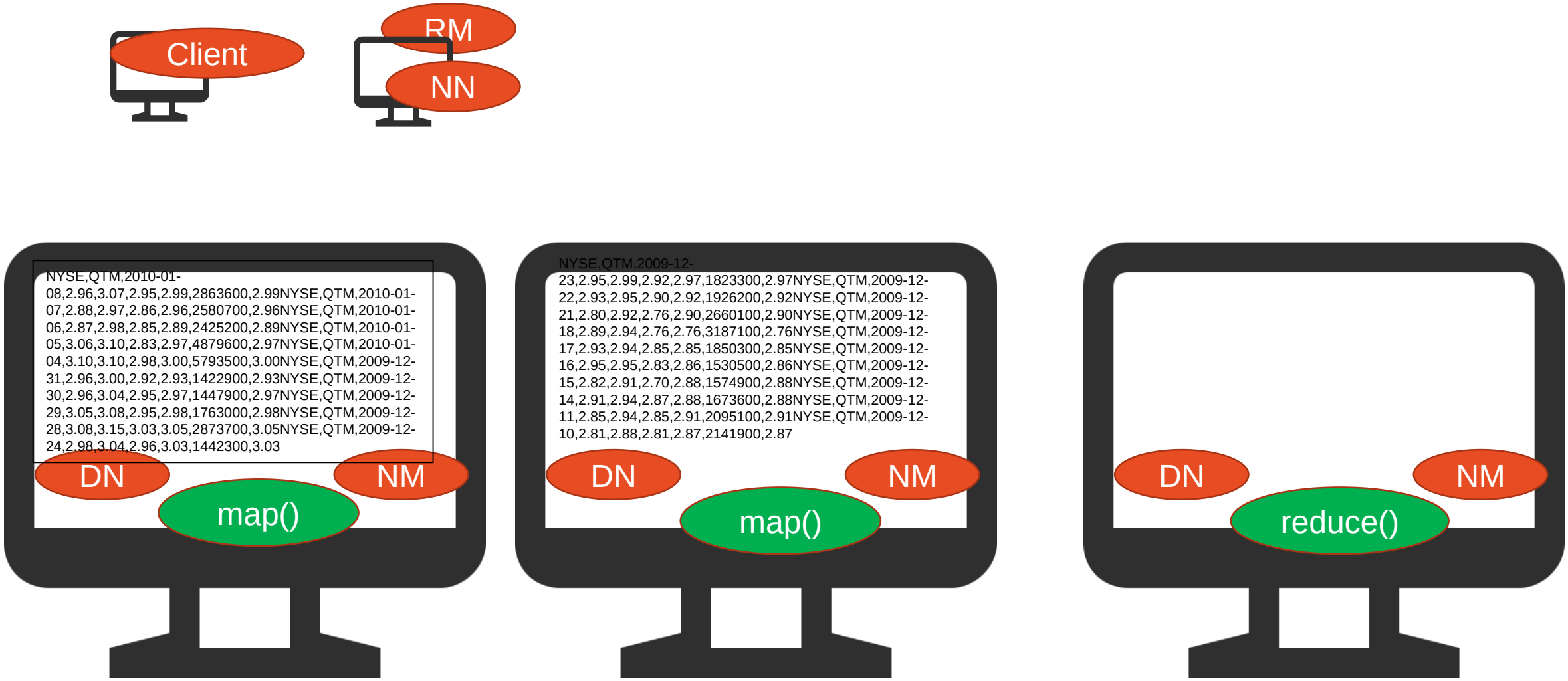
RDD Examples

```
# Get distinct order_status from the orders dataset

>>> ordersRDD.first()

>>> x = ordersRDD.first()

>>> x.split(',')

>>> x.split(',')[3]

>>> ordersRDD.map(lambda x: x.split(',')[3]).distinct().collect()
```

A small Hadoop cluster – 4 nodes



A small Hadoop cluster – 4 nodes



Spark on Hadoop (without YARN)