From Relational to Hadoop

Part 1: Introduction to Hadoop

Gwen Shapira , Cloudera
and
Danil Zburivsky, Pythian

MYSQL WORLDWIDE CONFERENCE & EXPO

PERCONA LIVE

Tutorial Logistics

Got VM?

- Grab a USB
- USB contains:
  - Cloudera QuickStart VM
  - Slides
  - Exercises and Solutions
  - Data

Plan of Action

I talk about Hadoop.

You multi-task between:

- Listening to me talk about Hadoop
- Installing Virtualbox and Running the VM

Using VirtualBox

1. Install and open VirtualBox on your computer
2. Under the menu "File", select "Import..."
3. Navigate to where you unpacked the .ovf file
4. and select it

5. You will find a “troubleshooting” file on the USB

Agenda - First 90 minutes

- Why use Hadoop for ETL?
- How HDFS Works?
- How MapReduce Works?
- What other tools we’ll use?

Use Hadoop for ETL?

ETL

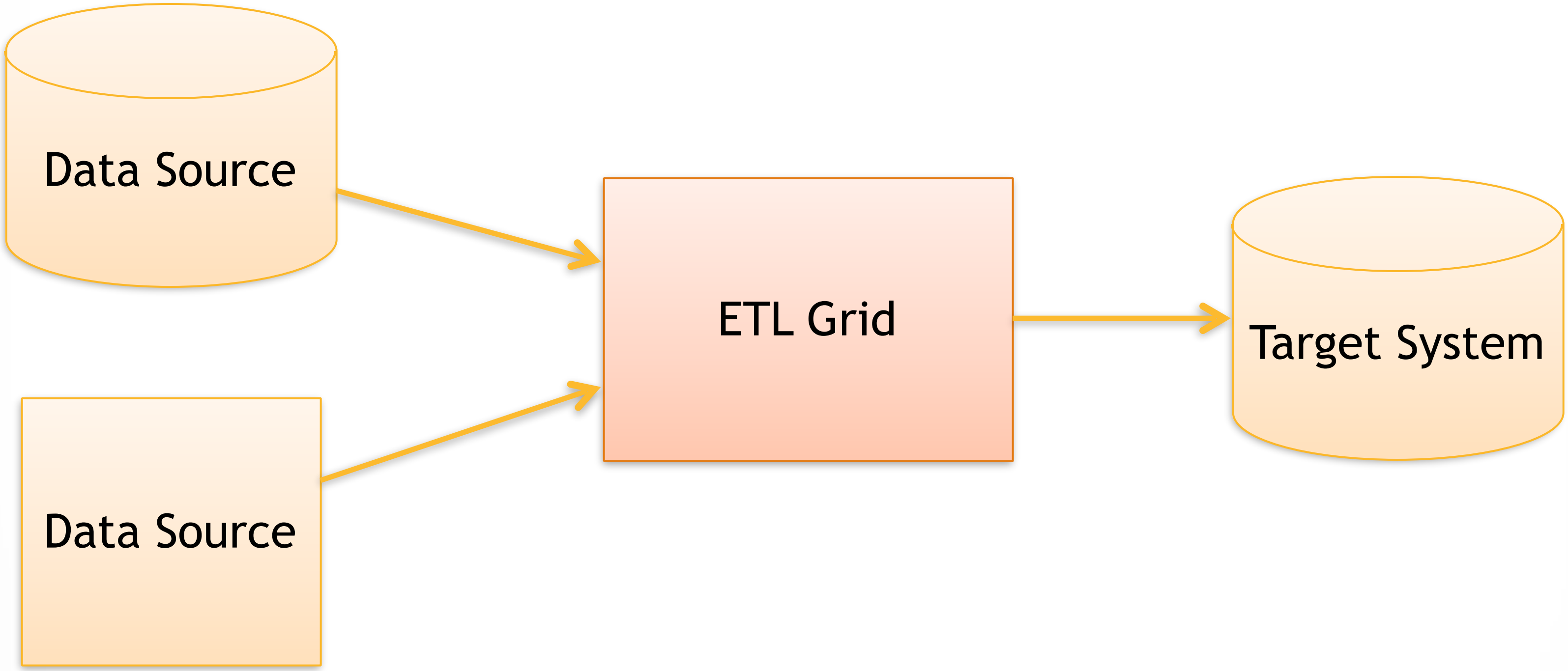
- Extracting data from outside sources
- Transforming it to fit operational needs
- Loading it into the end target

- (Wikipedia: http://en.wikipedia.org/wiki/Extract,_transform,_load)

Why ETL?

- Transform data before loading it to DB
  - XMLs, JSONs…
- Clean and standardize data
  - Units, names…
- Aggregate data, recommendations
- De-normalize for a DWH

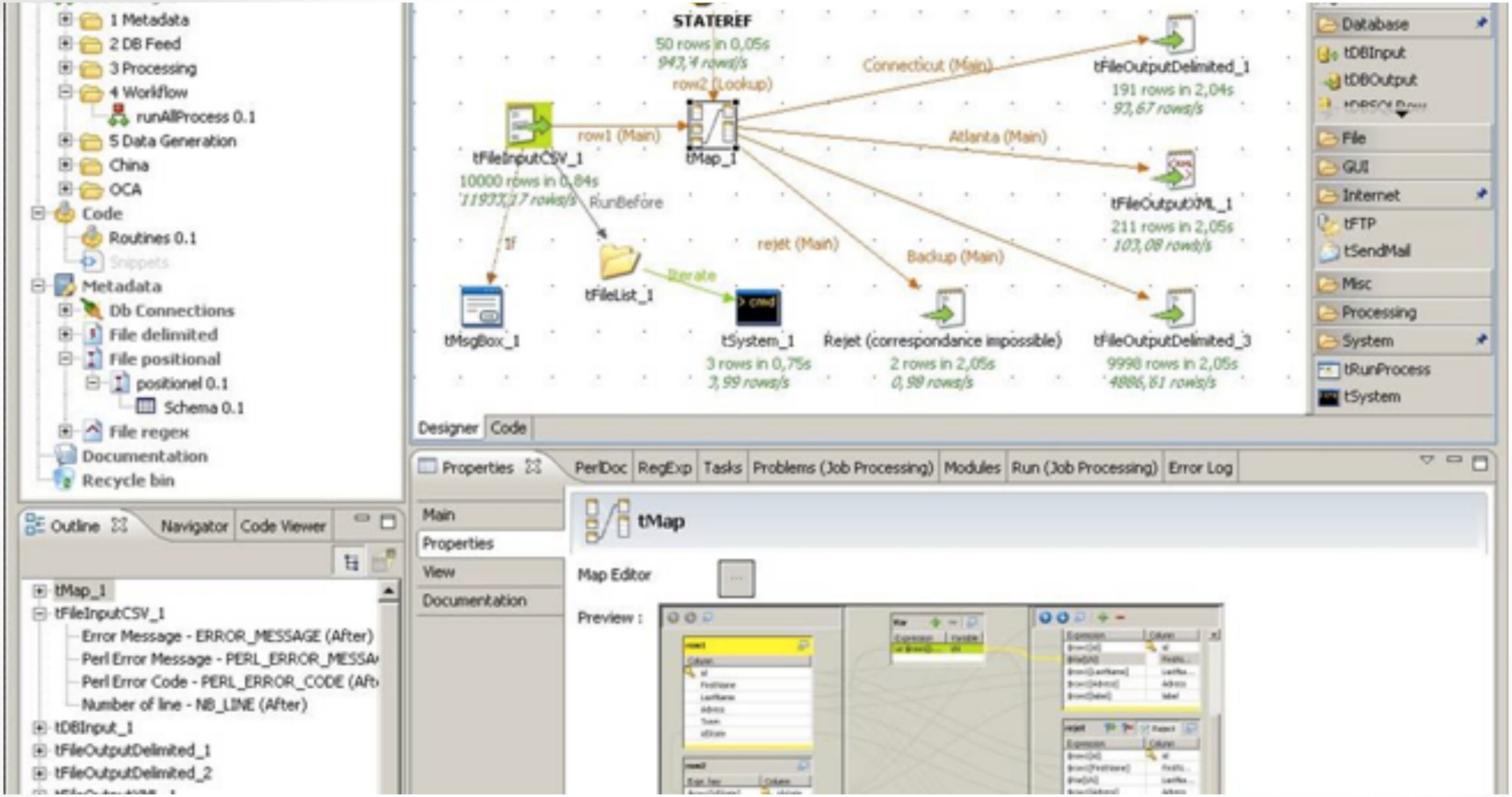
Some people do it like this:



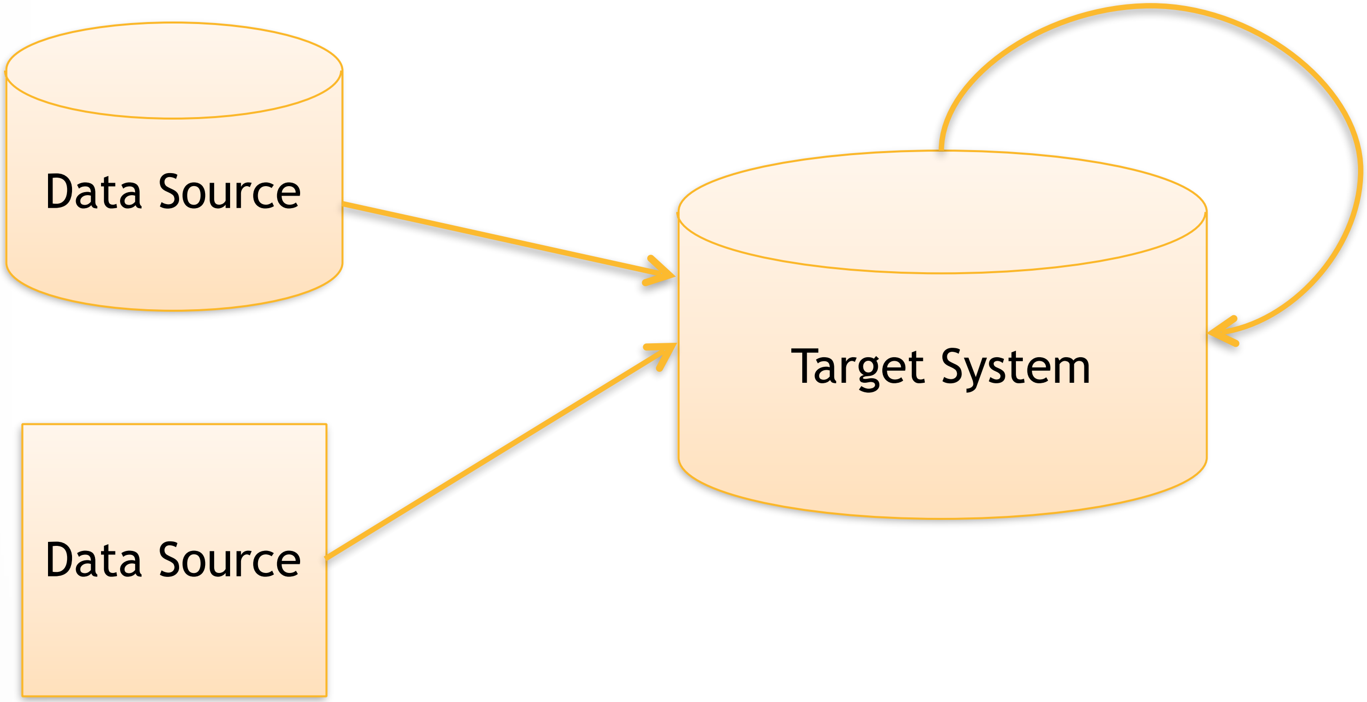
One Problem

8 hour processing pipeline can fail after 7.5 hours

OK, two


1 Metadata
2 DB Feed
3 Processing
4 Workflow
runAllProcess 0.1
5 Data Generation
China
OCA
Code
Routines 0.1
Snippets
Metadata
Db Connections
File delimited
File positional
positionel 0.1
Schema 0.1
File regex
Documentation
Recycle bin
Outline
Navigator
Code Viewer
tMap_1
tFileInputCSV_1
Error Message - ERROR_MESSAGE (After)
Perl Error Message - PERL_ERROR_MESSA
Perl Error Code - PERL_ERROR_CODE (Aft
Number of line - NB_LINE (After)
tDBInput_1
tFileOutputDelimited_1
tFileOutputDelimited_2
STATEREF
50 rows in 0,05s
943,4 rows/s
row2 (Lookup)
Connecticut (Main)
tFileOutputDelimited_1
191 rows in 2,04s
93,67 rows/s
row1 (Main)
tFileInputCSV_1
tMap_1
10000 rows in 0,84s
11933,17 rows/s
RunBefore
Atlanta (Main)
tFileOutputXML_1
211 rows in 2,05s
103,08 rows/s
If
rejet (Main)
Backup (Main)
Iterate
tFileList_1
tMsgBox_1
tSystem_1
3 rows in 0,75s
3,99 rows/s
Rejet (correspondance impossible)
2 rows in 2,05s
0,98 rows/s
tFileOutputDelimited_3
9998 rows in 2,05s
4886,61 rows/s
Database
tDBInput
tDBOutput
File
GUI
Internet
tFTP
tSendMail
Misc
Processing
System
tRunProcess
tSystem
Designer
Code
Properties
PerlDoc
RegExp
Tasks
Problems (Job Processing)
Modules
Run (Job Processing)
Error Log
Main
Properties
View
Documentation
tMap
Map Editor
Preview :

Or like this:



Few problems here too

- Unstructured data is a bigger challenge
- More contenders for CPU resources
  - Analysts
  - And even transactions
- Storage can get expensive
- Scaling DWH is a pain
- SQL is a bit limited

Use Hadoop!

- Handles unstructured data
- Scales horizontally
  - Storage
  - CPU
- Flexible and powerful
- Lots of tools

In this tutorial we'll show:

- Basics of using Hadoop
- Transform unstructured data
- Get RDBMS data to Hadoop
- Use Hadoop to join and aggregate
- Load the results back to an RDBMS

HDFS:

Distributed, fault-tolerant file system

Design Assumptions

- Failures are common
  - More scale == more failure
- Files are append-only
- Files are large (Gigabytes +)
- Access is large and sequential

Quick Disk Primer

- Disk does a seek for each I/O operation
- Seeks are expensive (~10ms)
- Throughput / IOPS tradeoff
  - 100 MB/s and 10 IOPS
  - 10MB/s and 100 IOPS
- Big I/Os mean better throughput

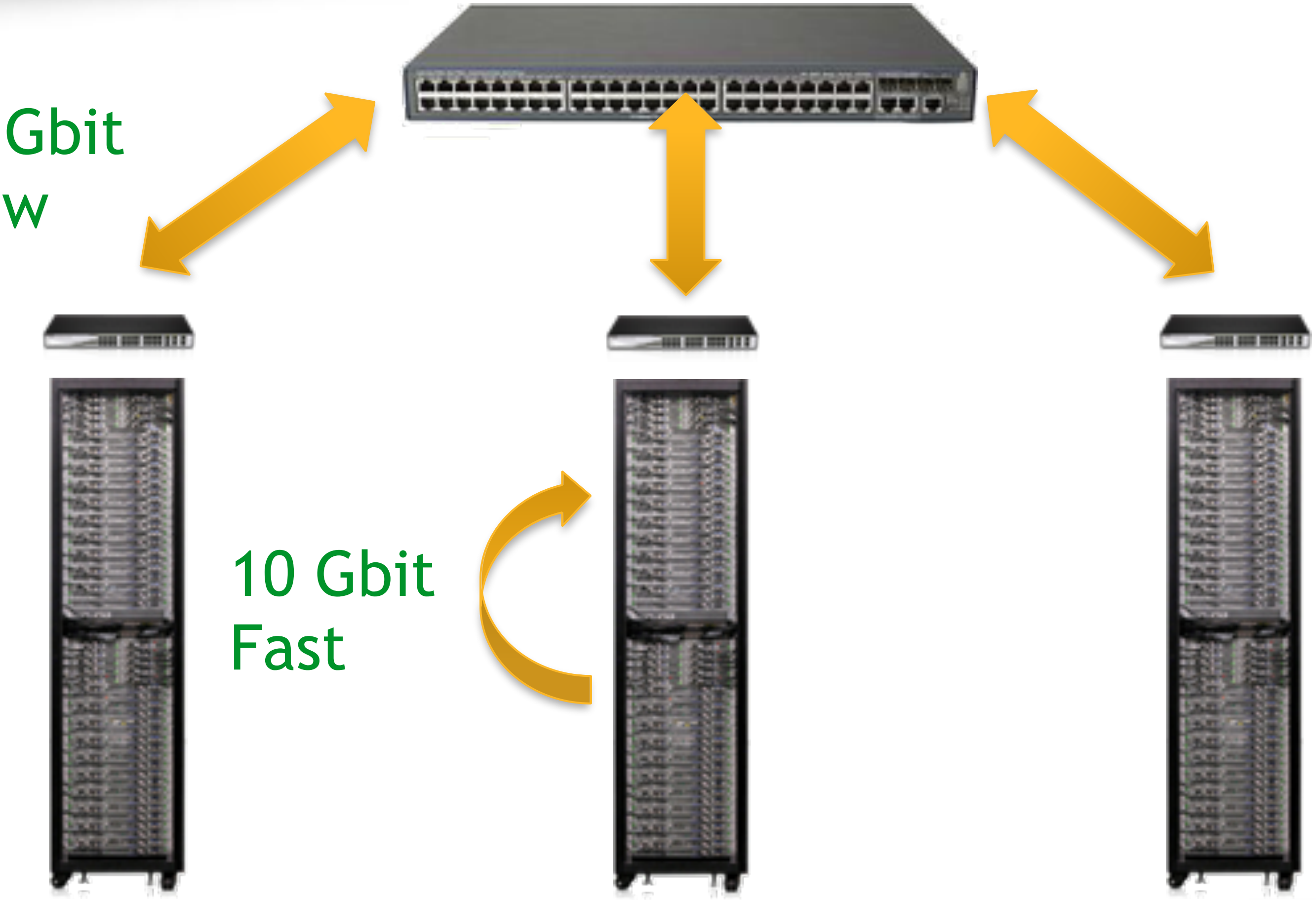
Quick Networking Primer

Core switch

Top-of-rack switch

Rack

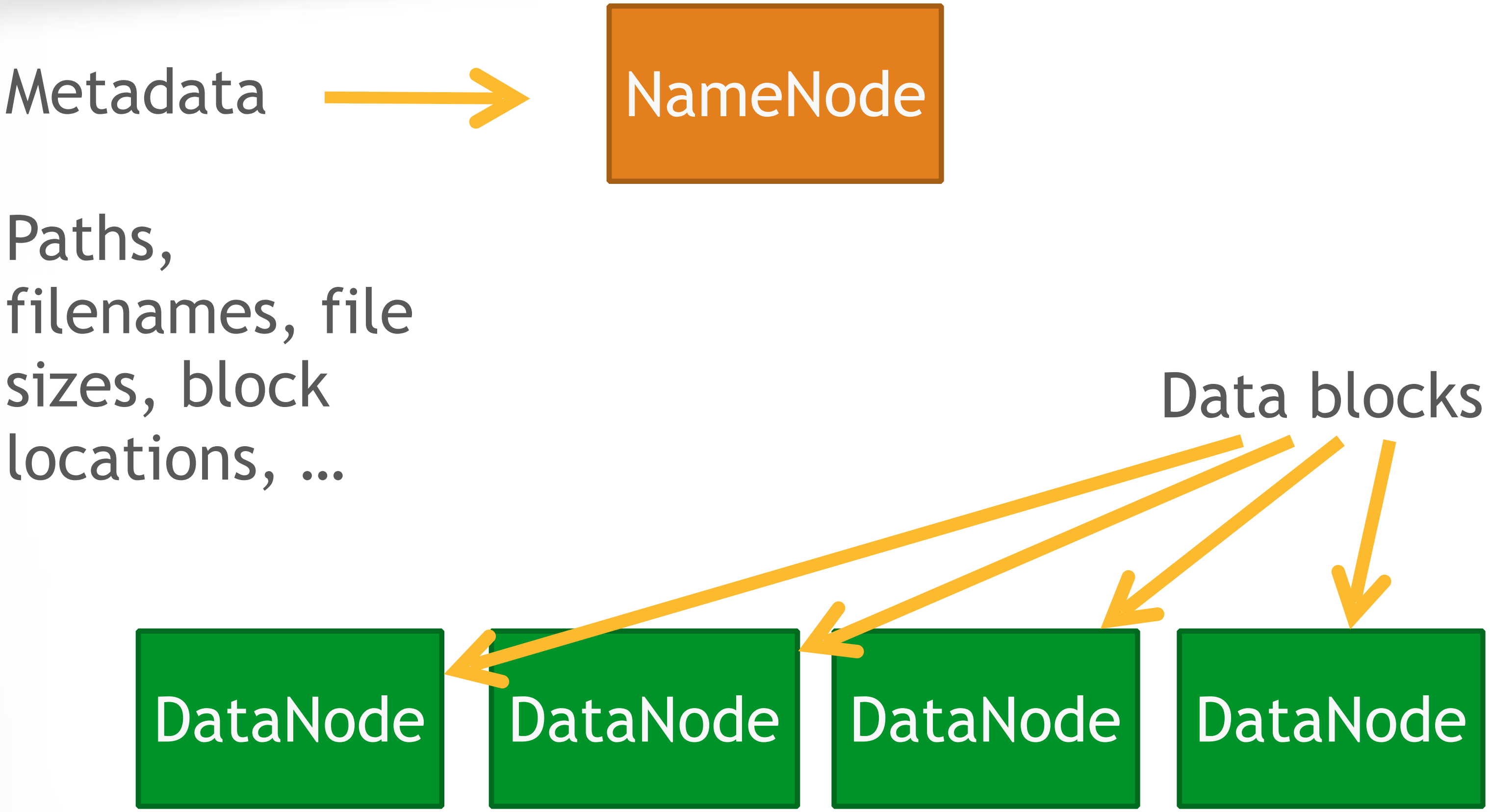
Quick Networking Primer



40 Gbit
Slow

10 Gbit
Fast

HDFS Architecture



Metadata

NameNode

Paths, filenames, file sizes, block locations, …

Data blocks

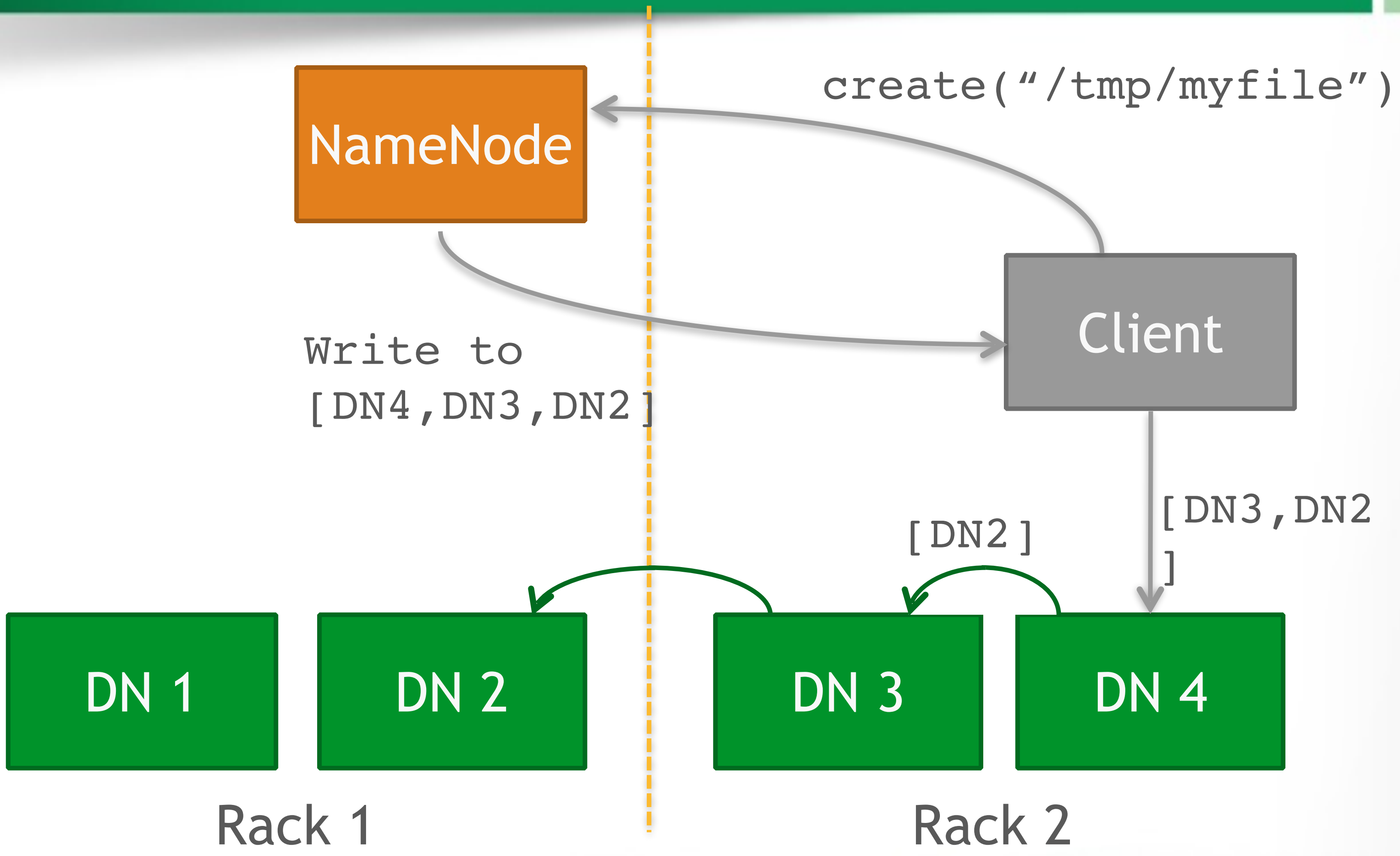
DataNode

DataNode

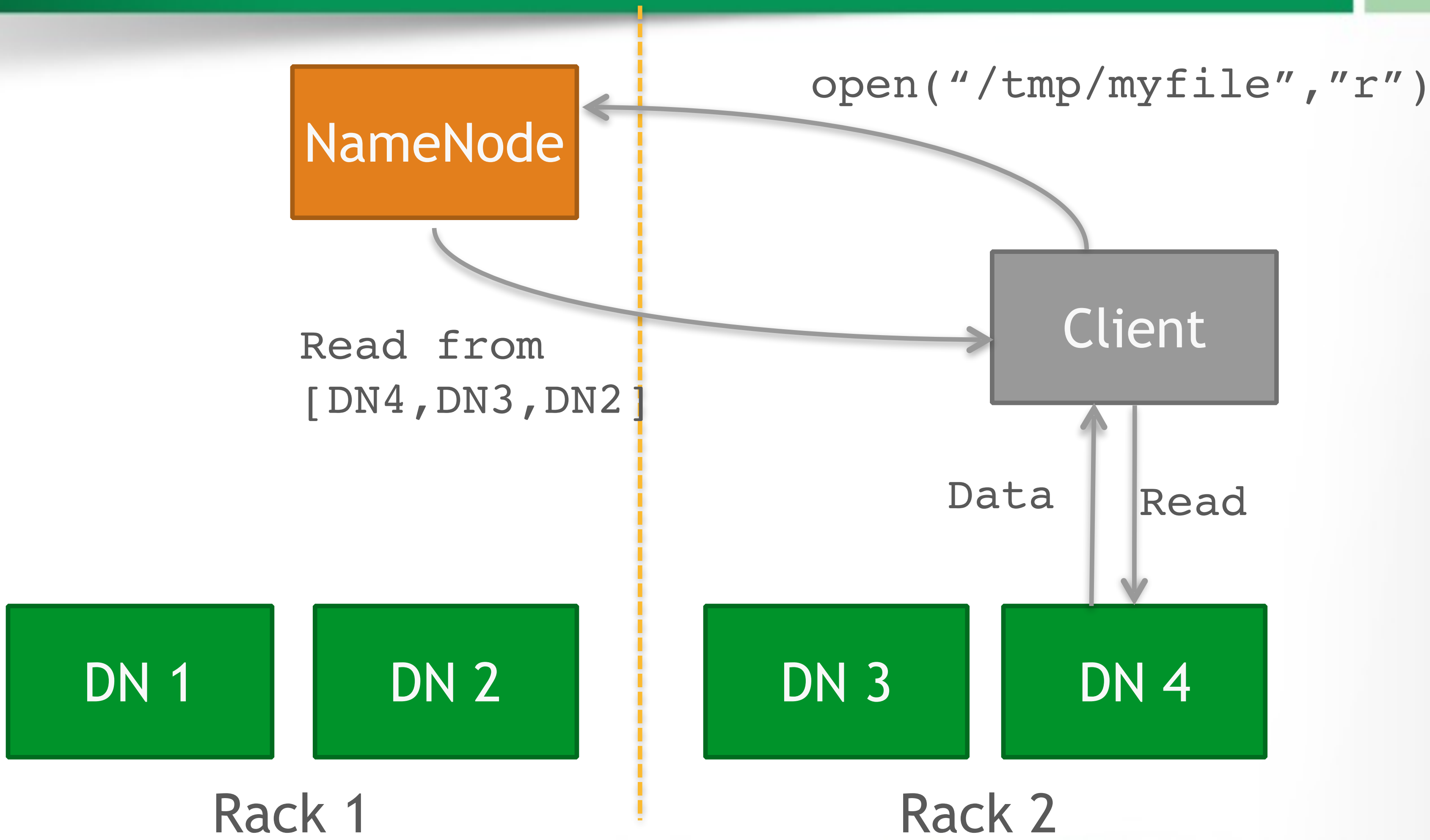
DataNode

DataNode

Writing to HDFS



Reading from HDFS



Using HDFS

- HDFS – command line interface
- hdfs dfs –mkdir
- hdfs dfs -ls
- hdfs dfs –put
- hdfs fsck

Practice Time

- Create HDFS directory:
  - /etl/earthquakes/landing
- Write earthquake data to directory
- Get first 10 lines of file
- How many blocks we have in the file?
- What is the replication factor?

Solution

-
  ```
  sudo –u hdfs hdfs dfs –mkdir /etl
  sudo –u hdfs hdfs dfs chown cloudera:cloudera /etl
  hdfs dfs –mkdir /etl/earthquakes/landing
  ```

-
  ```
  hdfs dfs –put ~/datasets/earthquakes.json /etl/earthquakes/landing
  ```

-
  ```
  hdfs dfs –cat /etl/earthquakes/landing/earthquakes.json | head -10
  ```

-
  ```
  hdfs fsck /etl/earthquakes/landing/earthquakes.json
  ```

MapReduce:

Programming and execution framework

Just implement two functions

- Map:
  - Operate on every element
  - Filter, transform
- Reduce:
  - Combine and aggregate results

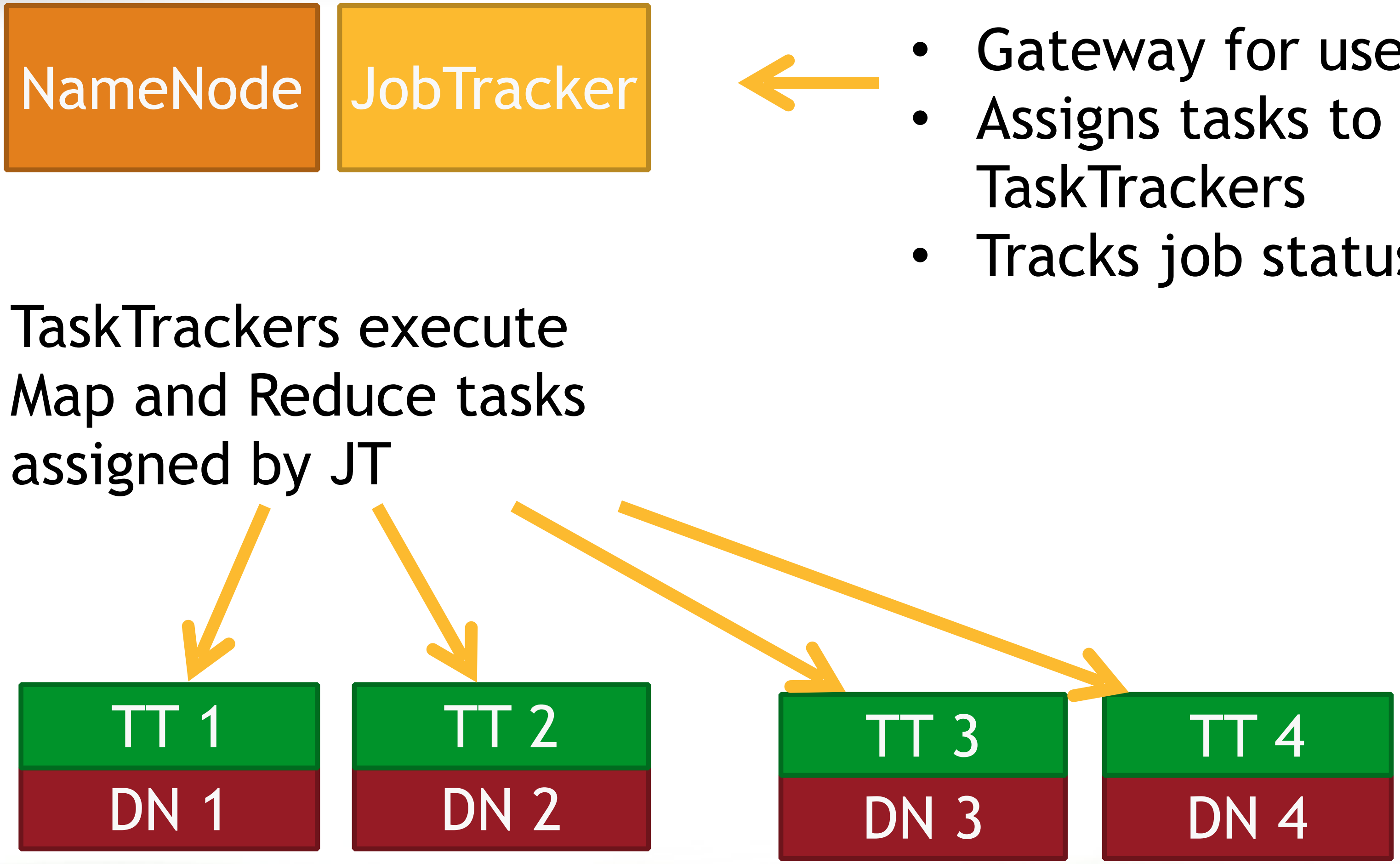
Good news

- You don’t need to know MR
- Many abstractions
- Alternative frameworks

Bad news

- You still need to know MR
- To understand how things work
- Still widely in use

Map Reduce Architecture



NameNode

JobTracker

- Gateway for users
- Assigns tasks to TaskTrackers
- Tracks job status

TaskTrackers execute Map and Reduce tasks assigned by JT

TT 1

DN 1

TT 2

DN 2

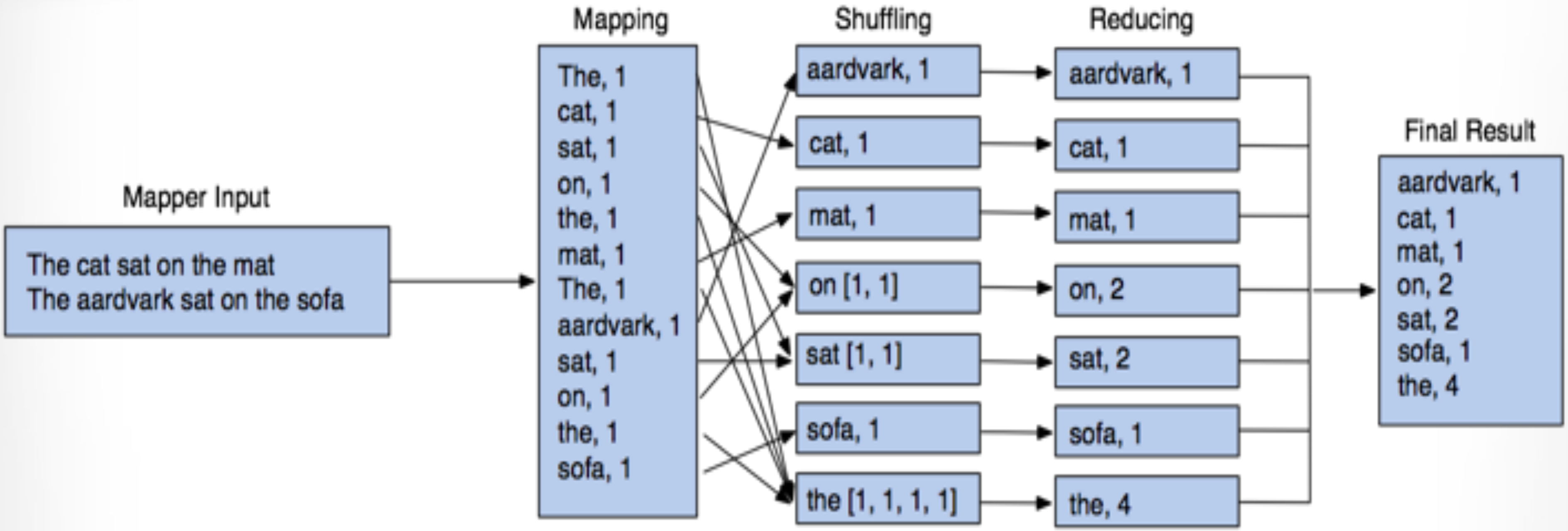
TT 3

DN 3

TT 4

DN 4

WordCount



Mapper Input

The cat sat on the mat
The aardvark sat on the sofa

Mapping

The, 1
cat, 1
sat, 1
on, 1
the, 1
mat, 1
The, 1
aardvark, 1
sat, 1
on, 1
the, 1
sofa, 1

Shuffling

aardvark, 1
cat, 1
mat, 1
on [1, 1]
sat [1, 1]
sofa, 1
the [1, 1, 1, 1]

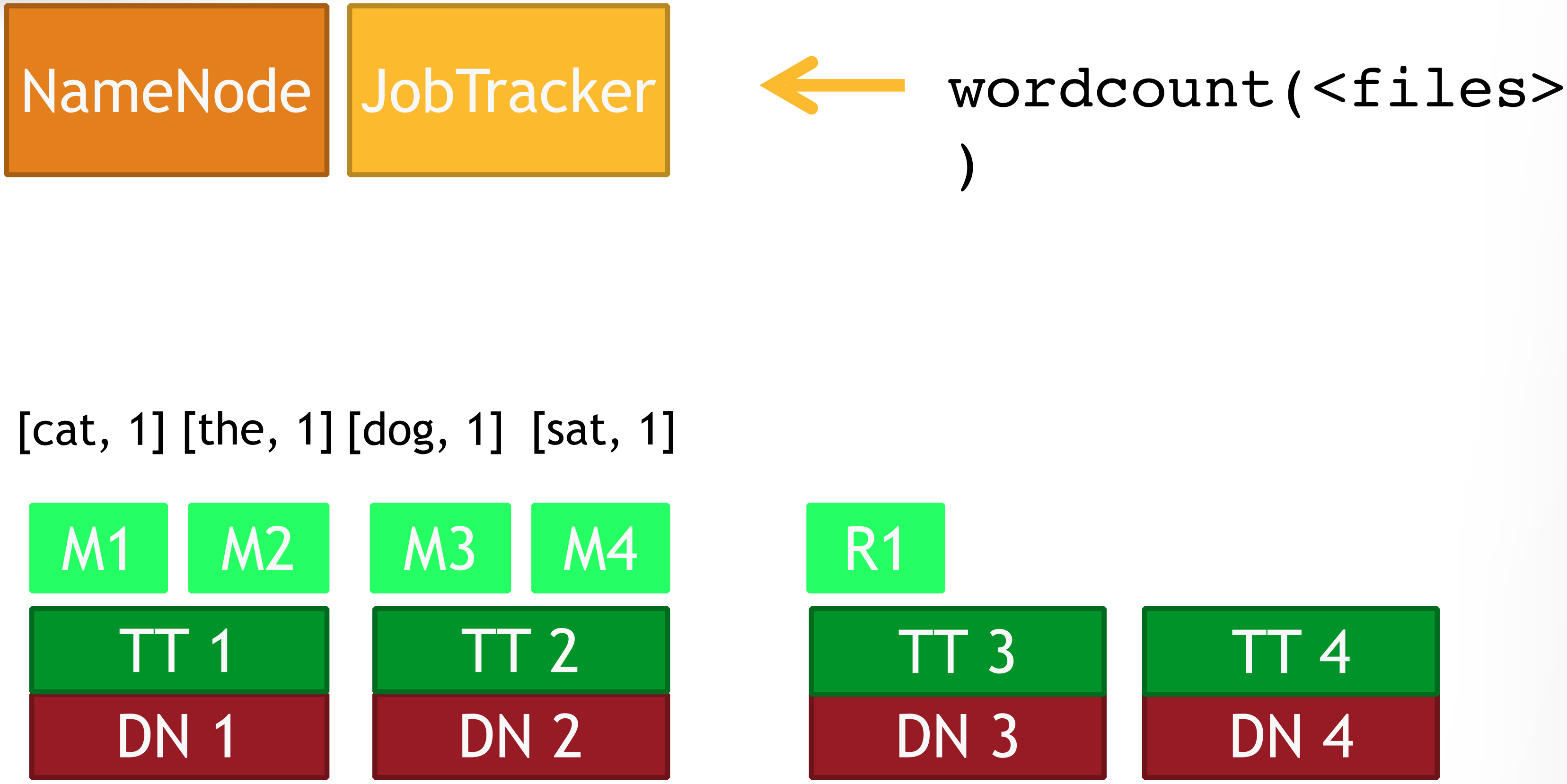
Reducing

aardvark, 1
cat, 1
mat, 1
on, 2
sat, 2
sofa, 1
the, 4

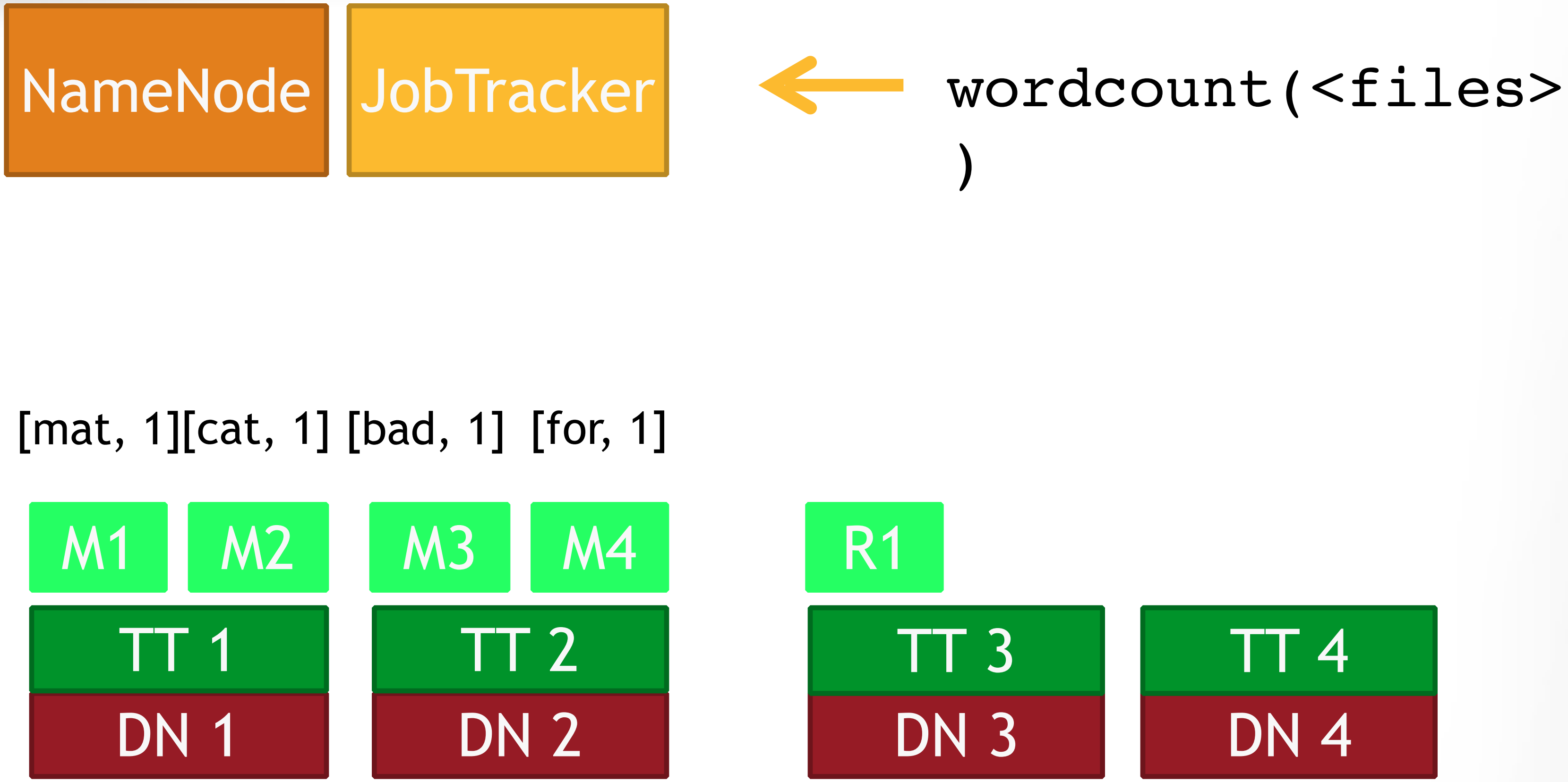
Final Result

aardvark, 1
cat, 1
mat, 1
on, 2
sat, 2
sofa, 1
the, 4

Running MapReduce



Running MapReduce



NameNode

JobTracker

wordcount(<files>
)

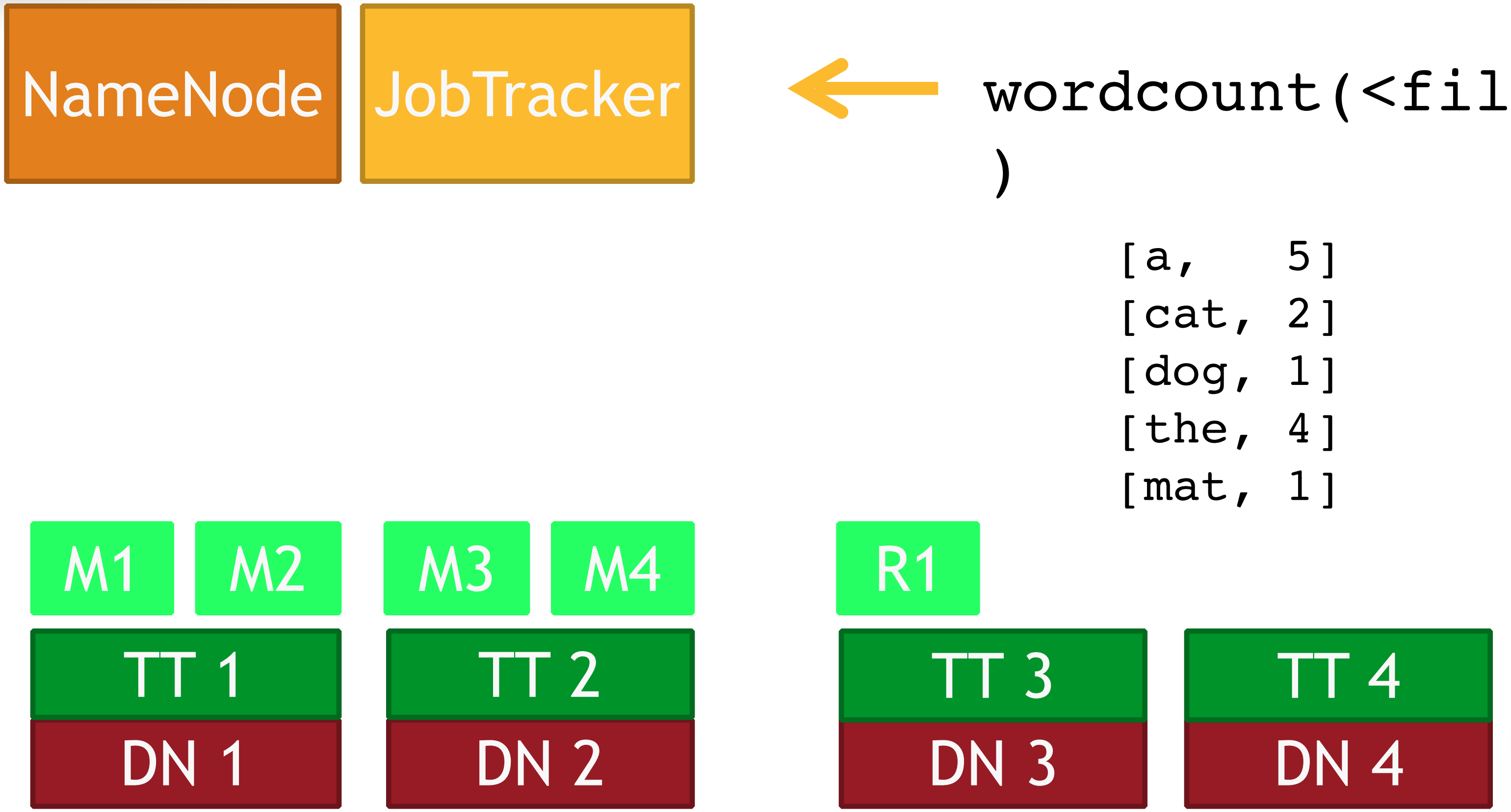
[mat, 1][cat, 1] [bad, 1] [for, 1]

M1 M2 M3 M4 R1

TT 1 TT 2 TT 3 TT 4

DN 1 DN 2 DN 3 DN 4

MapReduce



NameNode JobTracker

M1 M2 M3 M4 R1

TT 1 TT 2 TT 3 TT 4

DN 1 DN 2 DN 3 DN 4

```
wordcount(<files>
)
```

```
[a,   5]
[cat, 2]
[dog, 1]
[the, 4]
[mat, 1]
```

Using MapReduce

- Submit a JAR
- Specify the class that contains the mapper and reducer
- hadoop jar <jar> <class> <parameters>

Practice Time

- Load “Works of Shakespeare” or other file to HDFS
- Count words
- Generate 100M of random data.
- Run “terasort”.

Solution

-
  ```
  hdfs dfs -put ~/pl_tutorial/datasets/shakespeare.txt /user/cloudera
  ```

-
  ```
  hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar wordcount /user/cloudera/shakespeare.txt /user/cloudera/cnt
  ```

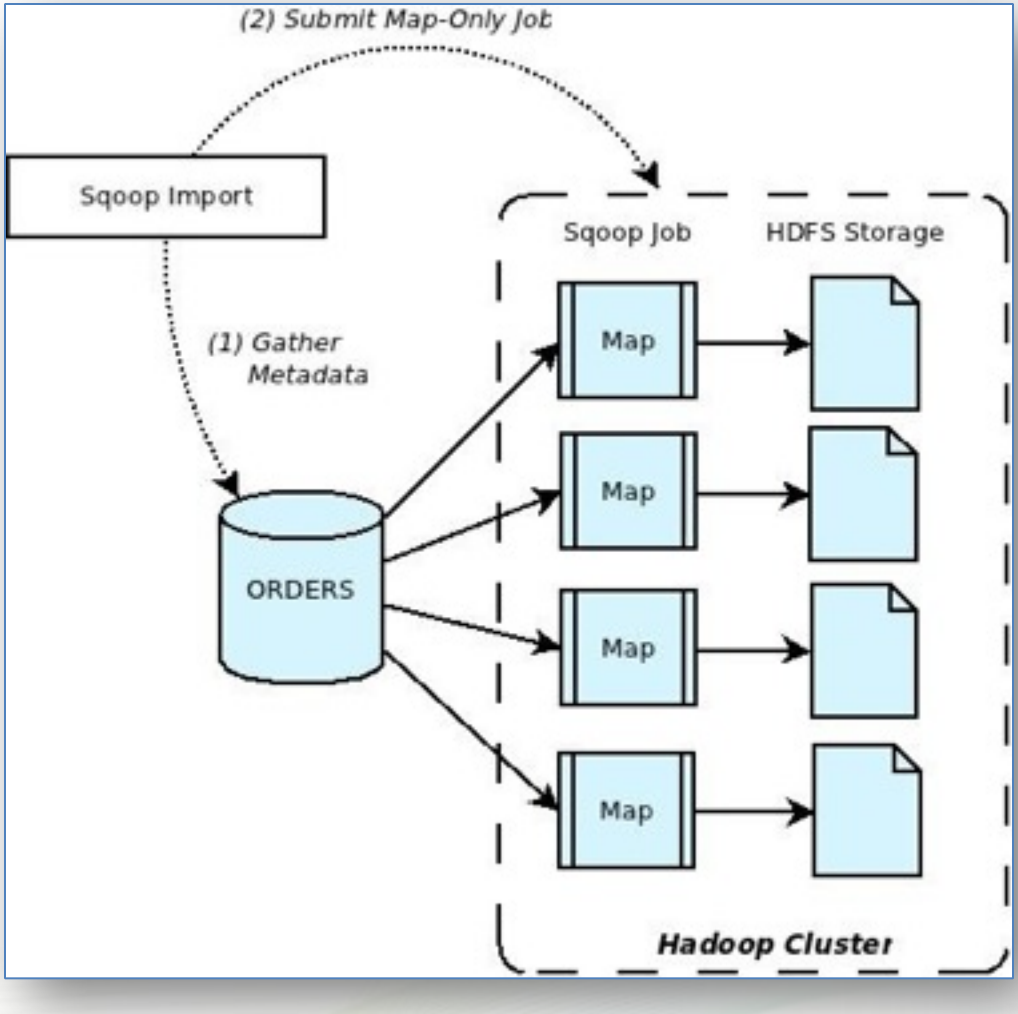
-
  ```
  hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar teragen  1000000 /user/cloudera/terasort-in
  ```

-
  ```
  hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar terasort /user/cloudera/terasort-in /user/cloudera/terasort-out
  ```

Ecosystem Tools

Sqoop

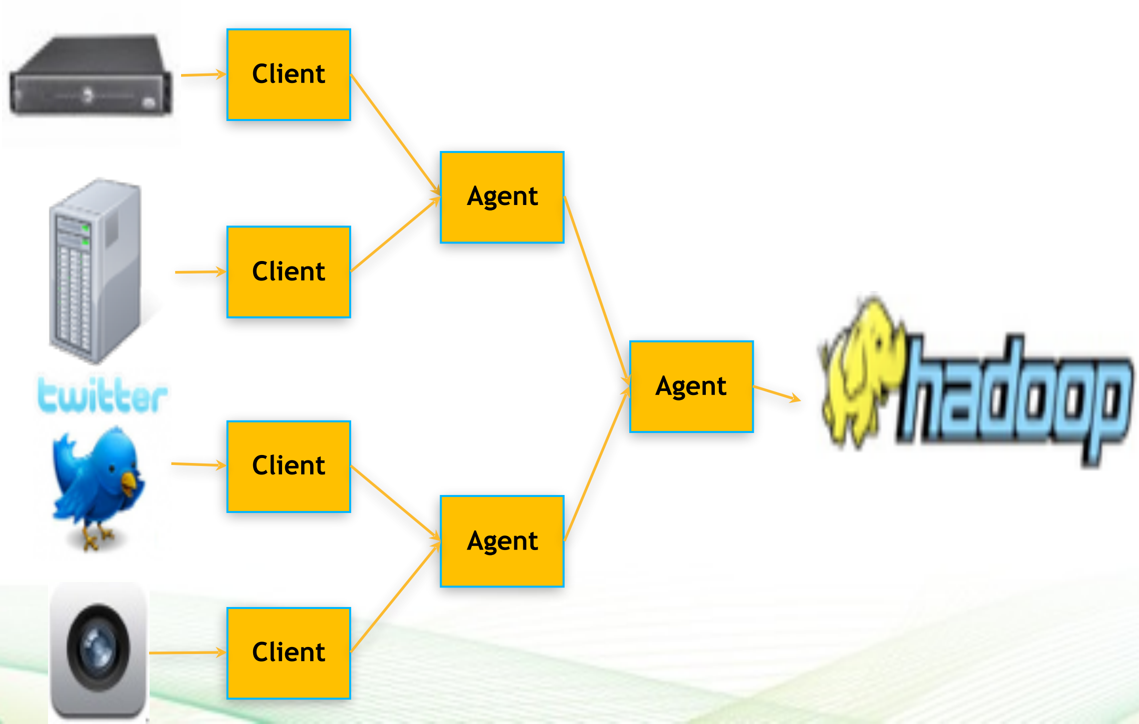
Transfers data between Hadoop and almost any SQL database with a JDBC driver



Flume

Streaming data collection

And aggregation for:
JMS queues, HTTP APIs, Log4J, Syslog, etc.



Hive

Translate SQL to MapReduce

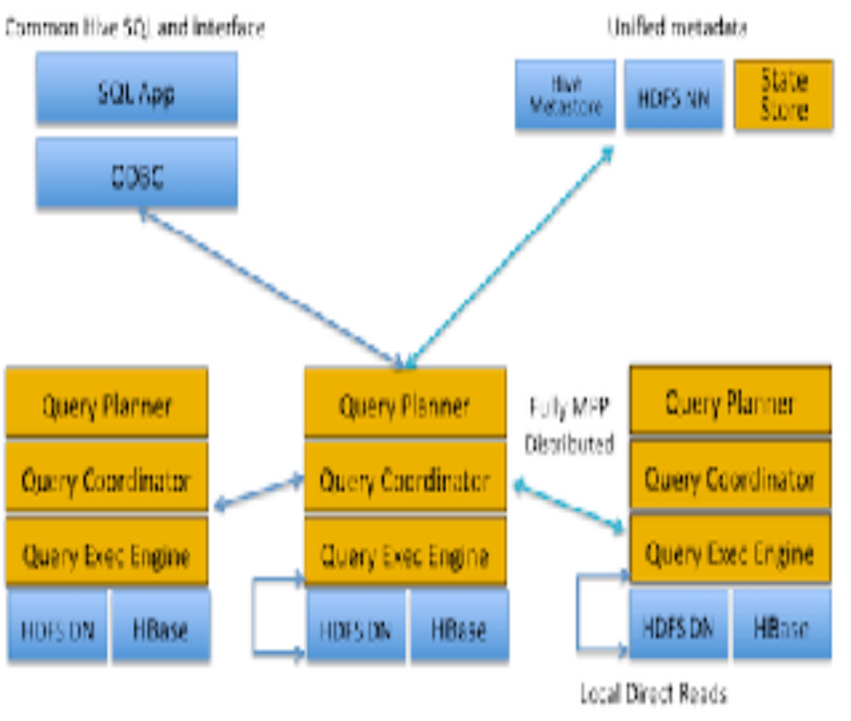
Select word,count(*) from shakespear
Group by word

Impala

Modern MPP database built on top of HDFS

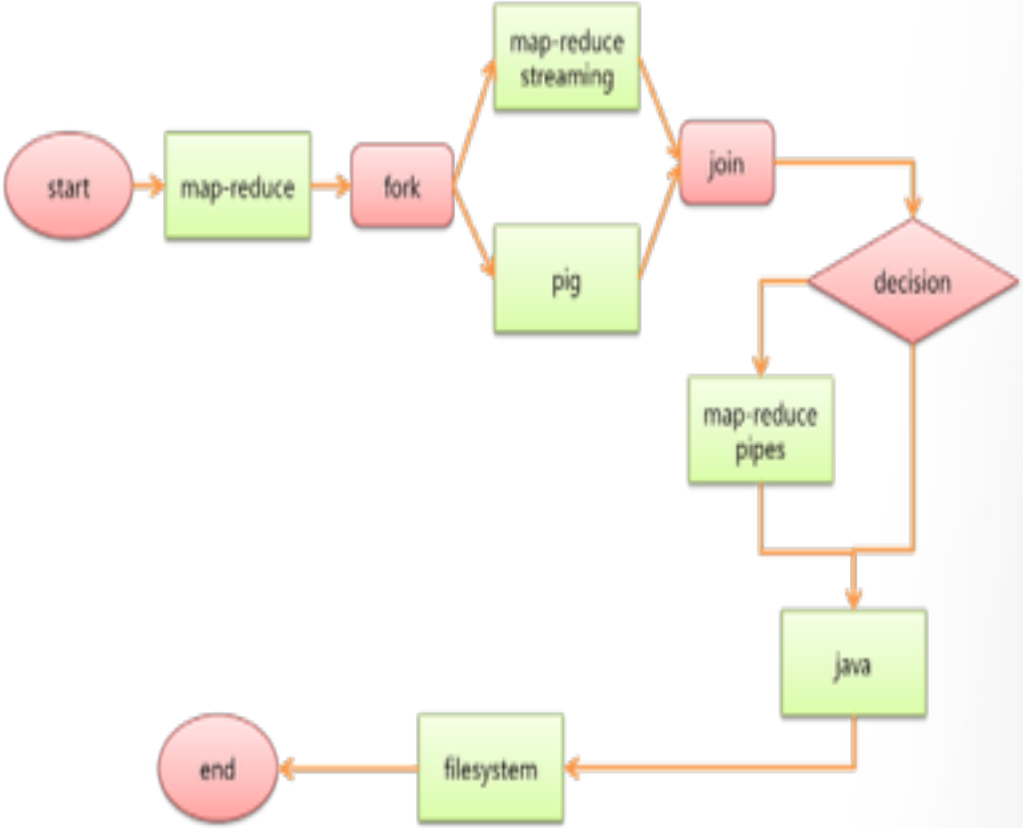
Really fast! Written in C++

10-100x faster than Hive



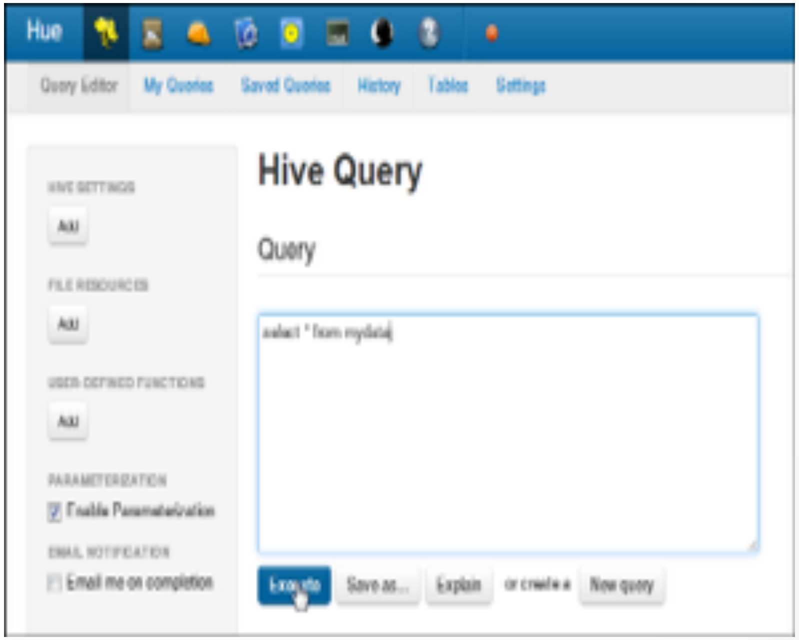
Oozie

A workflow engine and scheduler built specifically for large-scale job orchestration on a Hadoop cluster



Hue

- Hue is an open source web-based application for making it easier to use Apache Hadoop.
- Hue features
  - File Browser for HDFS
  - Job Designer/Browser for MapReduce
  - Query editors
  - Oozie



Hive Query

Query

```
select * from mydata
```

PARAMETERIZATION

☑ Enable Parameterization

EMAIL NOTIFICATION

☐ Email me on completion

Practice Time

- Login to Hue
- Play around
- We will dive into Sqoop, Hive and Oozie in the next hour