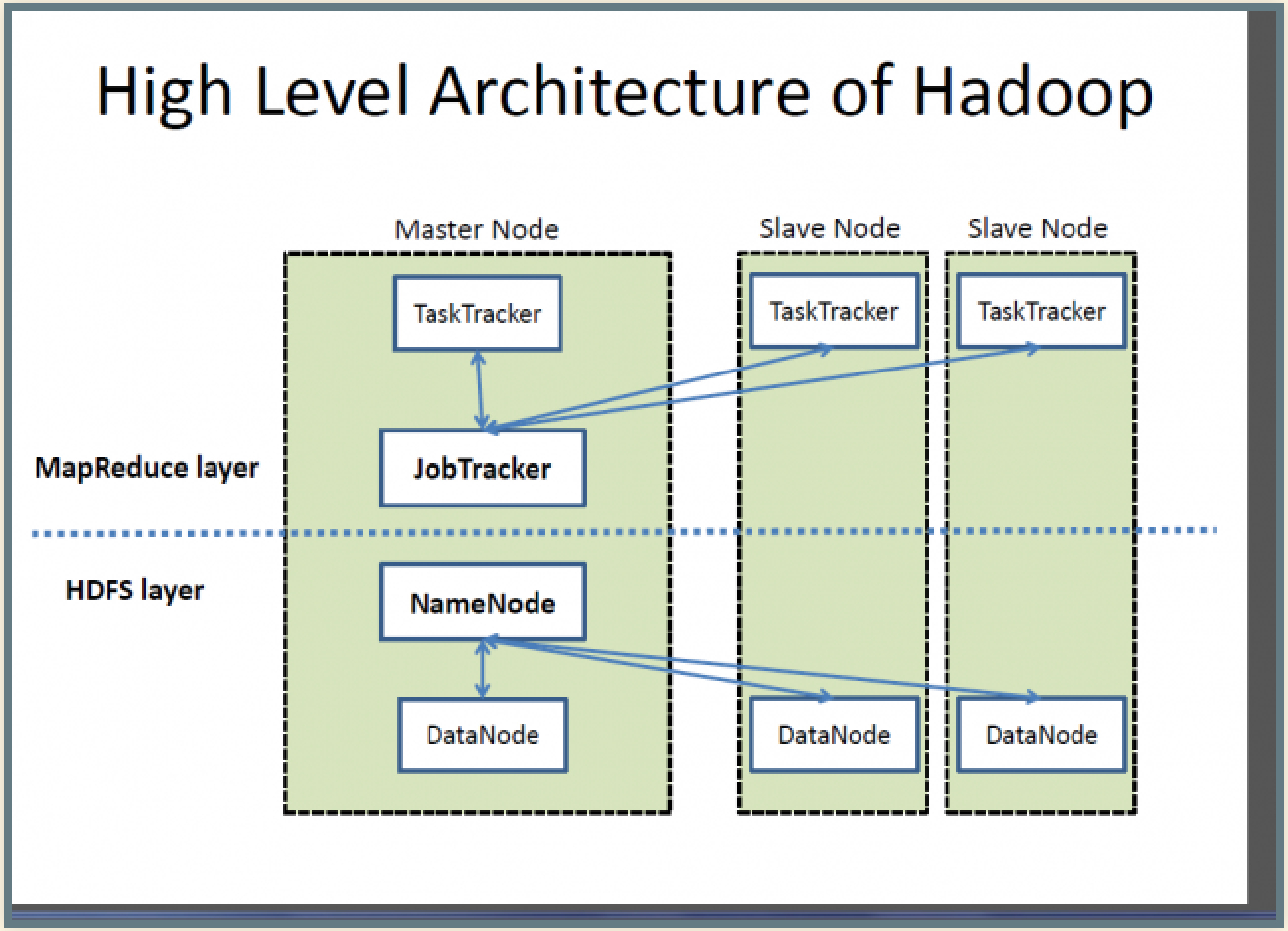
presto

1. BIG DATA

HIVE
OOZIE
STORM
hadoop
FLUME
APACHE
HBASE
Hortonworks
cloudera
Spark
sqoop
Cassandra

1.1. HADOOP

ARCHITECTURE



High Level Architecture of Hadoop

Master Node

Slave Node

Slave Node

TaskTracker

TaskTracker

TaskTracker

MapReduce layer

JobTracker

HDFS layer

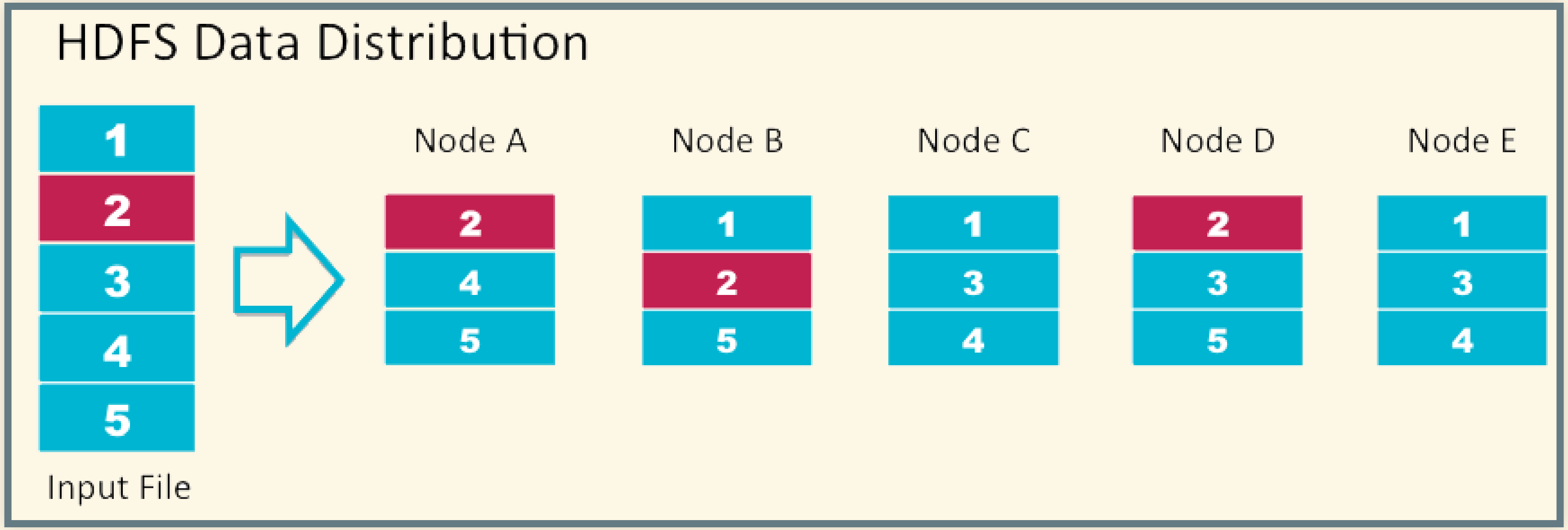
NameNode

DataNode

DataNode

DataNode

HDFS

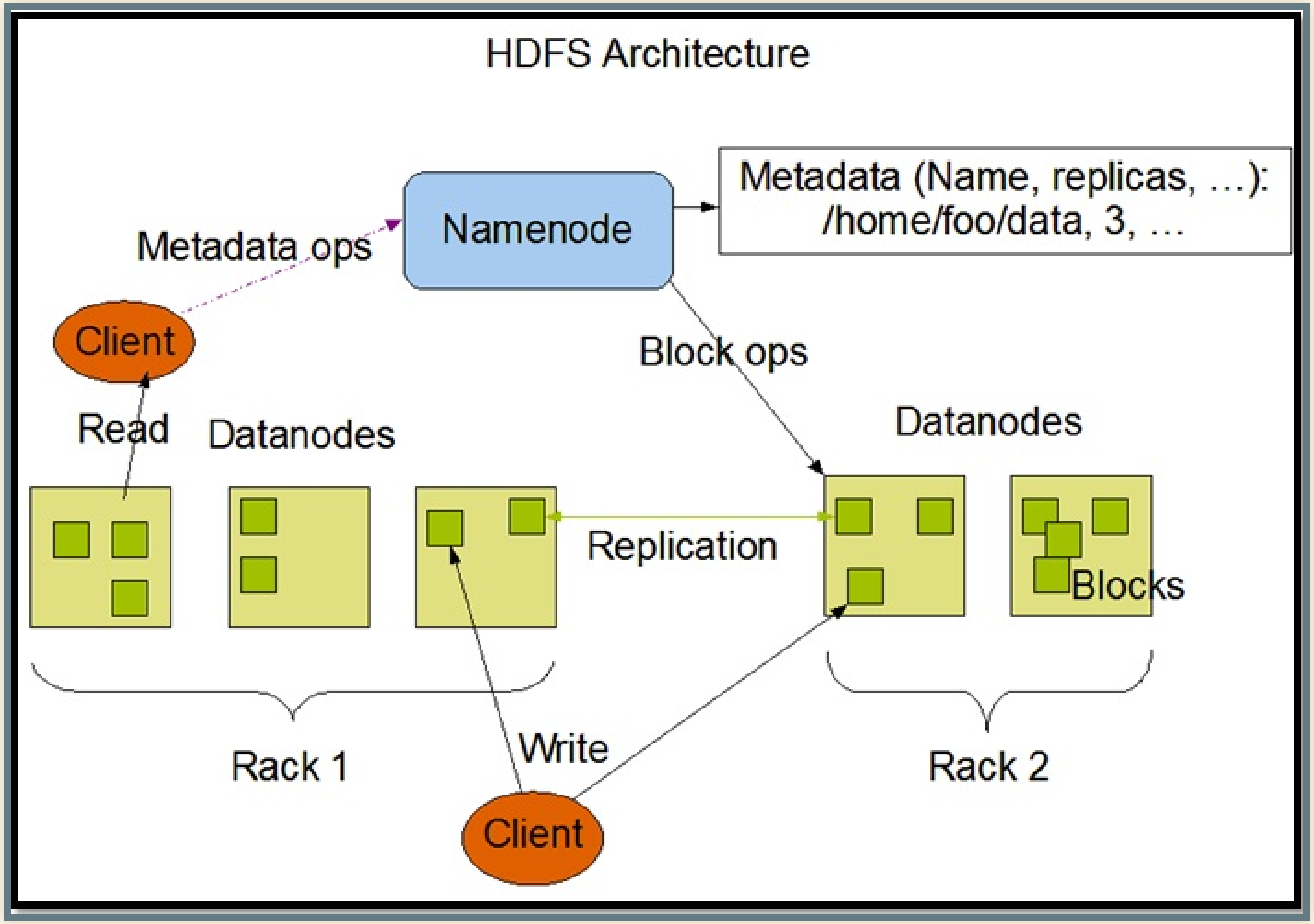


HDFS Data Distribution

1
2
3
4
5

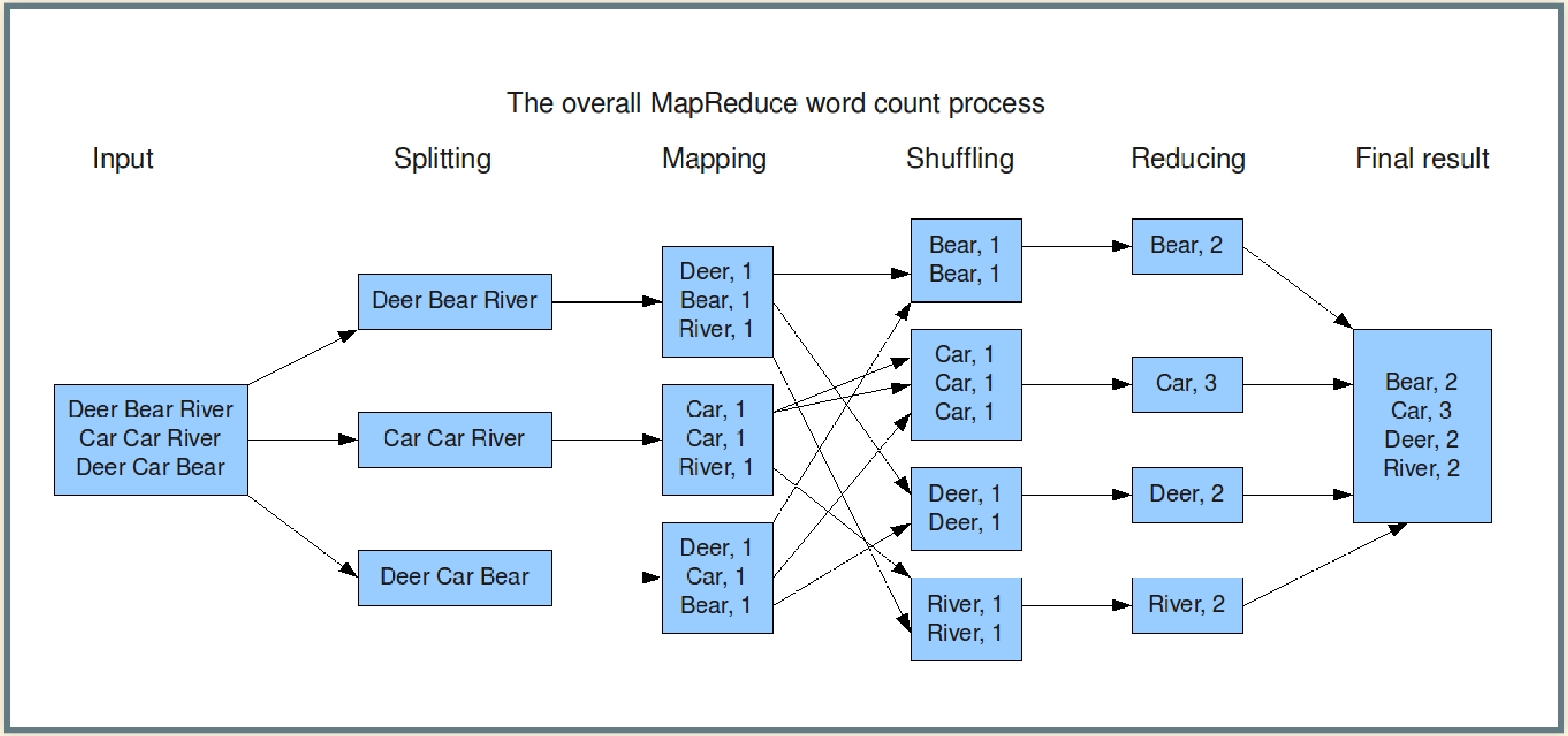
Input File

Node A	Node B	Node C	Node D	Node E
2	1	1	2	1
4	2	3	3	3
5	5	4	5	4



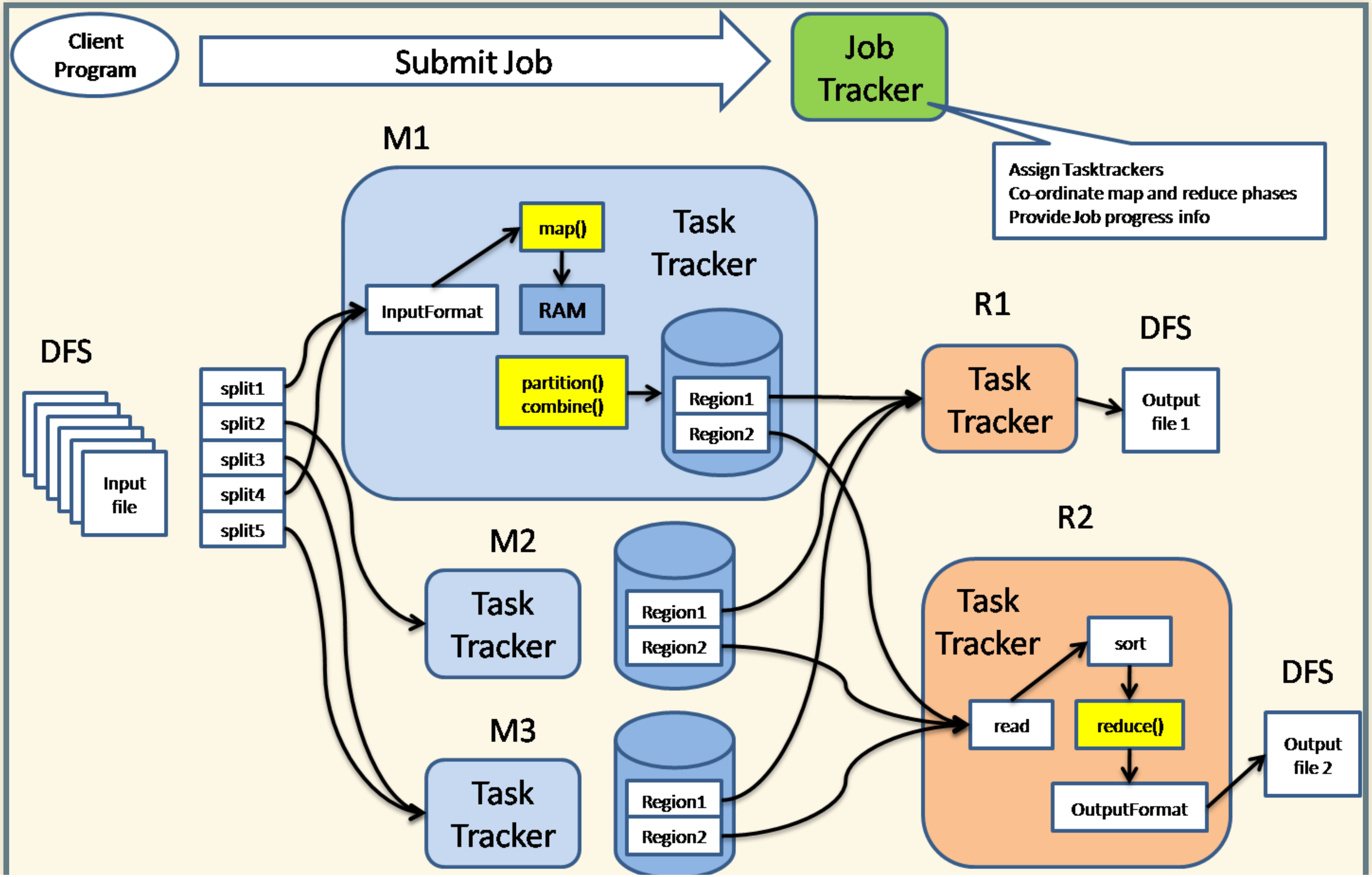
- NN (Namenode) is SPOF (Single Point of Failure)
- Normally HA (High availability) through a standby
- Zookeeper for coordination, bookeeping

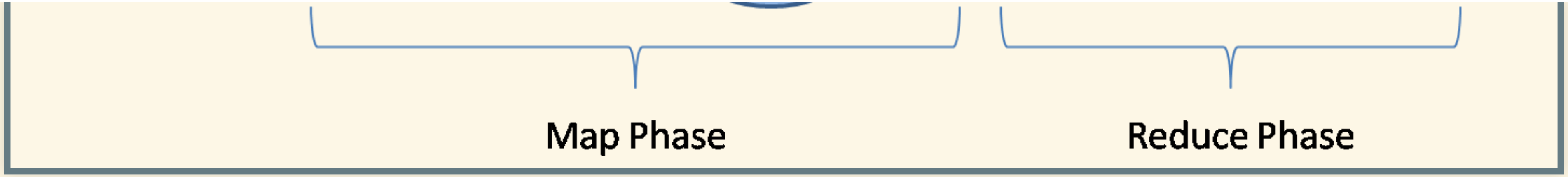
WORD COUNT



The overall MapReduce word count process

MAPREDUCE ARCHITECTURE



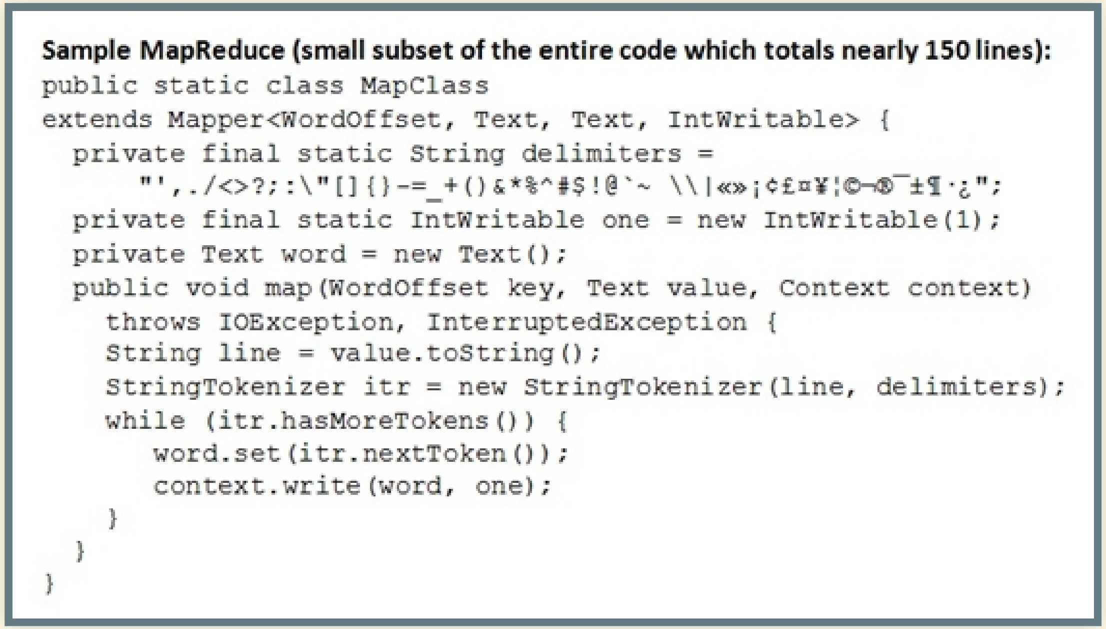


Map Phase

Reduce Phase

QUERYING

- Initially Java

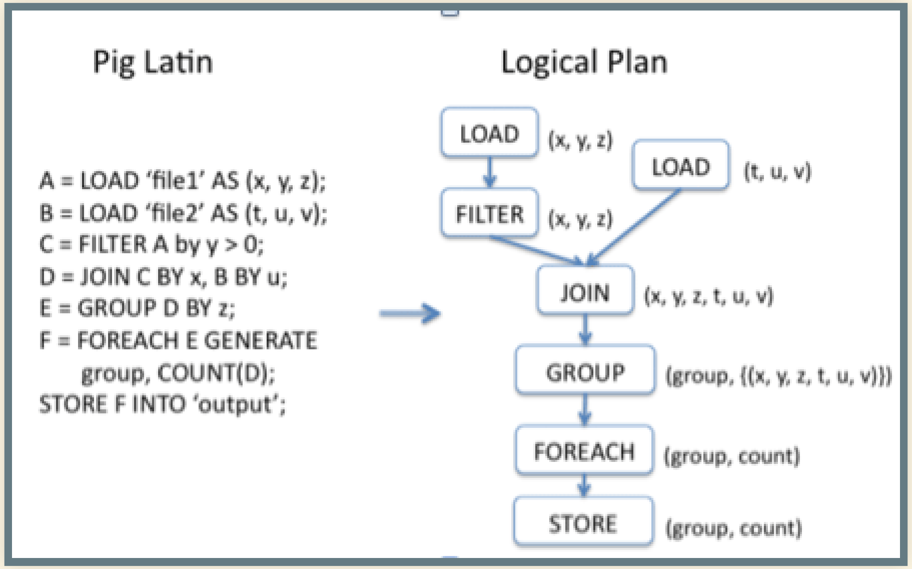


Sample MapReduce (small subset of the entire code which totals nearly 150 lines):

```
public static class MapClass
extends Mapper<WordOffset, Text, Text, IntWritable> {
  private final static String delimiters =
      "',./<>?;:\"[]{}-=_+()&*%^#$!@`~ \\|«»¡¢£¤¥¦©¬®¯±¶·¿";
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();
  public void map(WordOffset key, Text value, Context context)
    throws IOException, InterruptedException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line, delimiters);
    while (itr.hasMoreTokens()) {
       word.set(itr.nextToken());
       context.write(word, one);
    }
  }
}
```

QUERYING

- Apache Pig



Pig Latin

```
A = LOAD ‘file1’ AS (x, y, z);
B = LOAD ‘file2’ AS (t, u, v);
C = FILTER A by y > 0;
D = JOIN C BY x, B BY u;
E = GROUP D BY z;
F = FOREACH E GENERATE
      group, COUNT(D);
STORE F INTO ‘output’;
```

- Apache Hive

```
SELECT pv_users.gender,
count(DISTINCT pv_users.userid),
count(*), sum(DISTINCT
pv_users.userid) FROM pv_users GROUP
BY pv_users.gender;
```

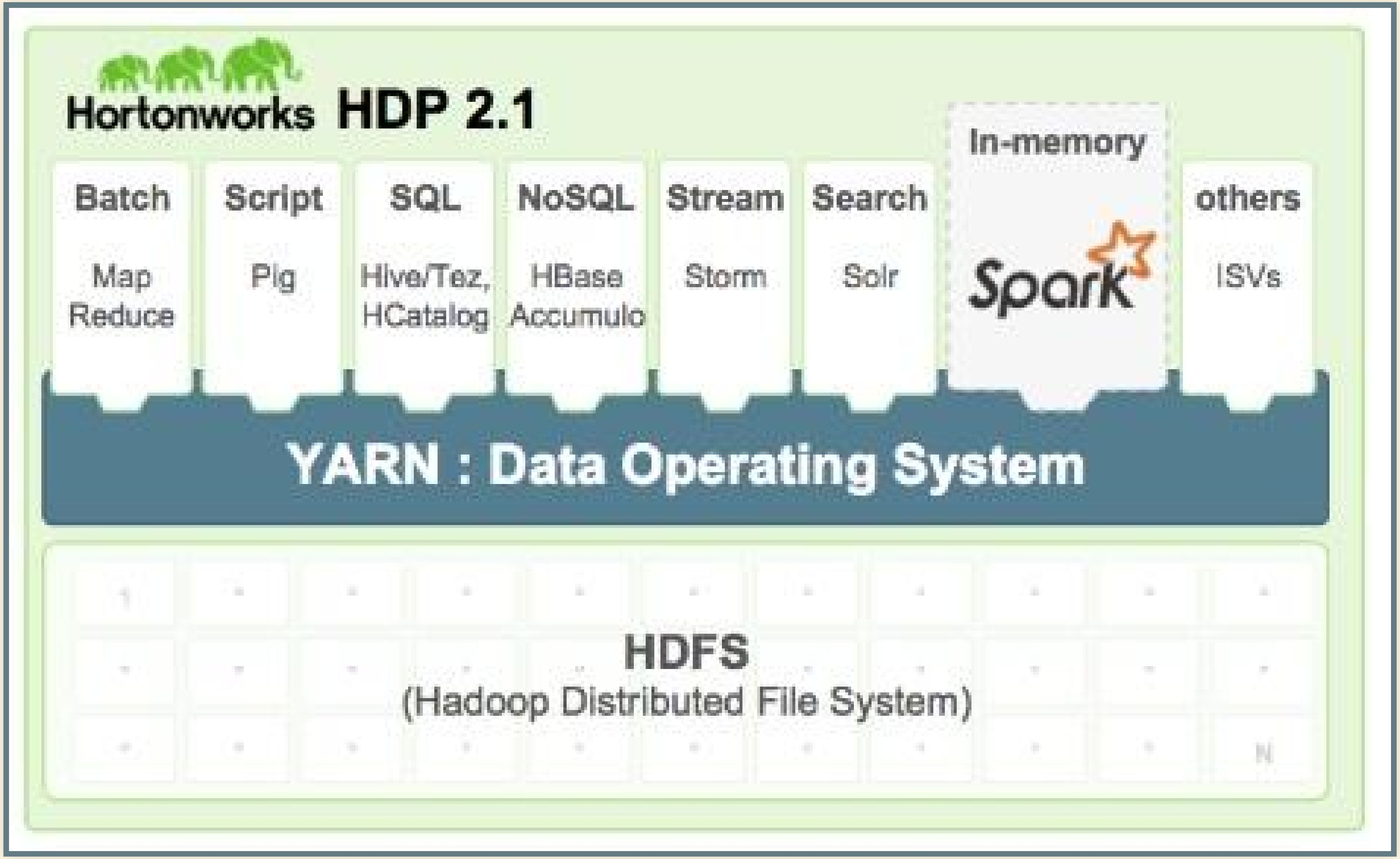
SQL

- Everyone knows
- Easy for analyists
- But there are actually alot of ways of querying HDFS
    - Cascading, Scalding, Cascalog, etc.

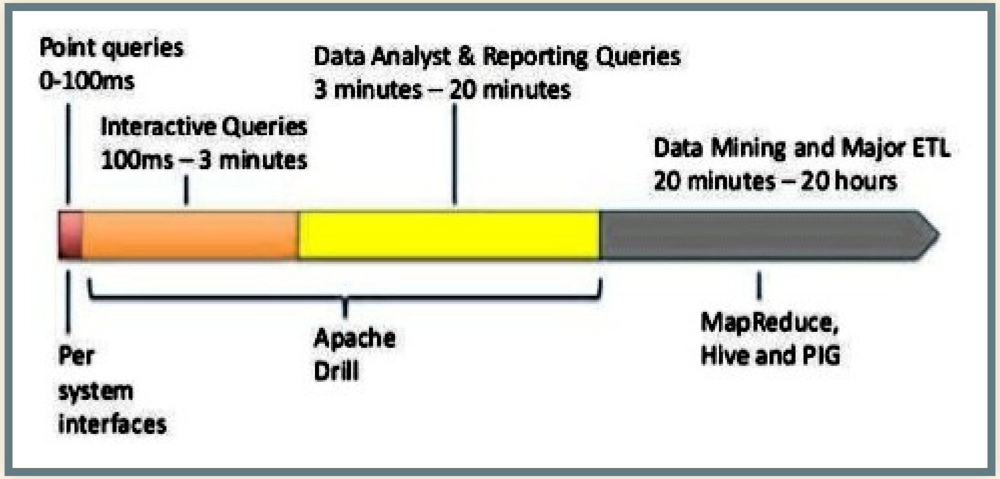
```
class WordCountJob(args : Args) extends Job(args) {
  TypedPipe.from(TextLine(args("input")))
    .flatMap { line => line.split("""\s+""") }
    .groupBy { word => word }
    .size
    .write(TypedTsv(args("output")))
}
```

```
(?- (stdout)
    (<- [?word ?count]
        (sentence :> ?line)
    (tokenise :< ?line :> ?word)
    (c/count :> ?count)))
```

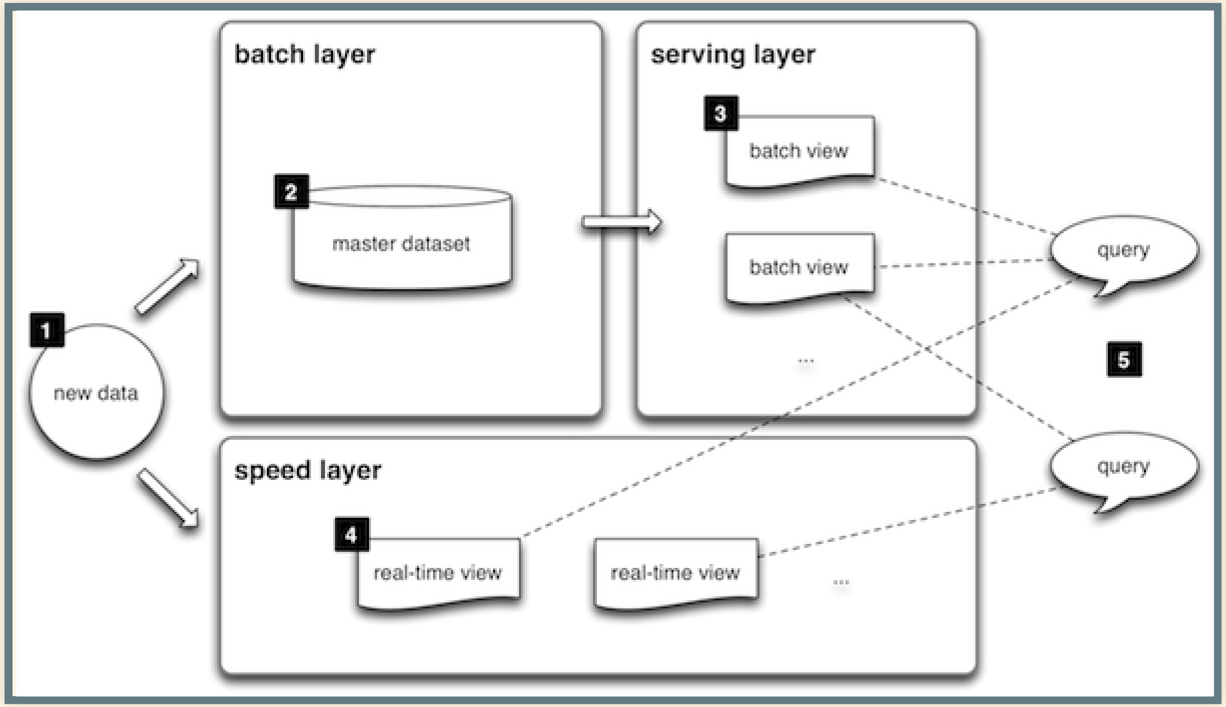
HDP


Hortonworks HDP 2.1
Batch
Map Reduce
Script
Pig
SQL
Hive/Tez, HCatalog
NoSQL
HBase Accumulo
Stream
Storm
Search
Solr
In-memory
Spark
others
ISVs
YARN : Data Operating System
1
HDFS
(Hadoop Distributed File System)
N

BUSINESS NEEDS



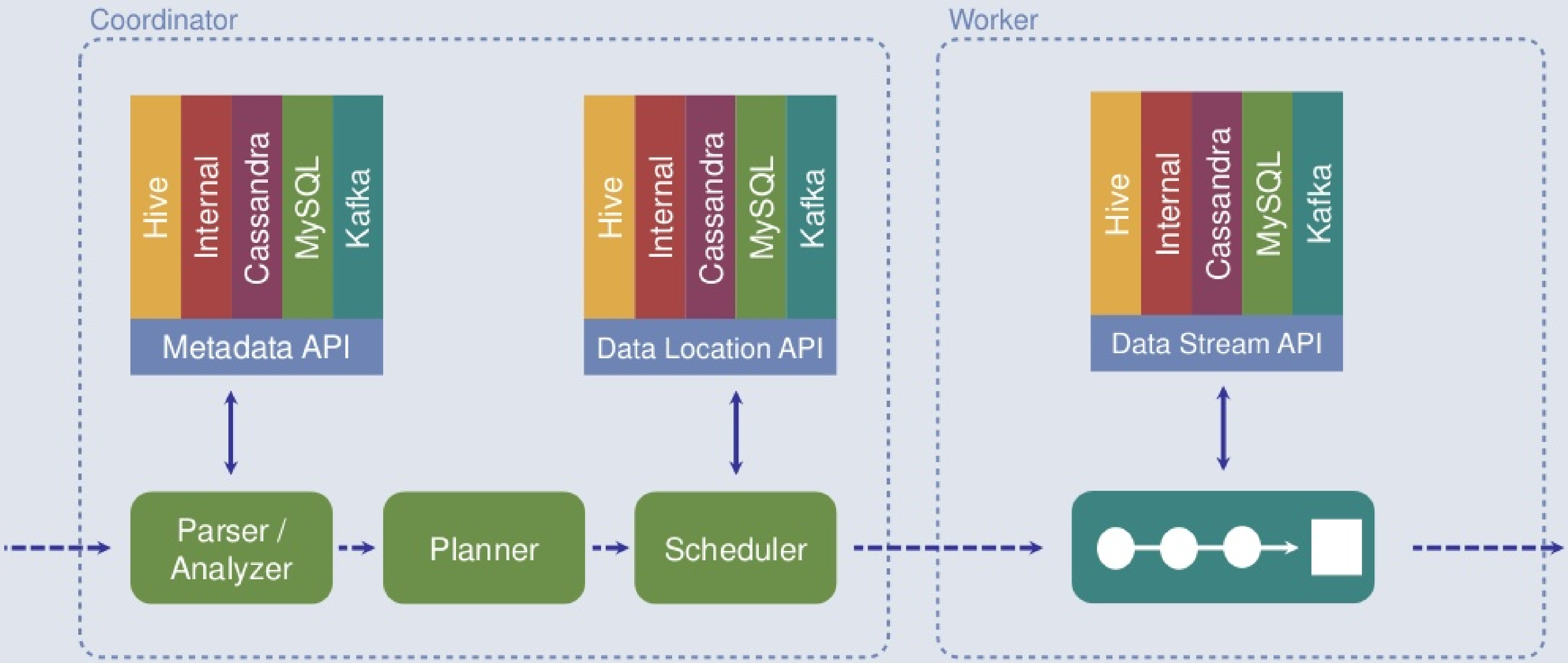
LAMBDA ARCHITECTURE



2. PRESTO

ARCHITECTURE

Connectors



Coordinator

Hive Internal Cassandra MySQL Kafka

Metadata API

Hive Internal Cassandra MySQL Kafka

Data Location API

Parser / Analyzer

Planner

Scheduler

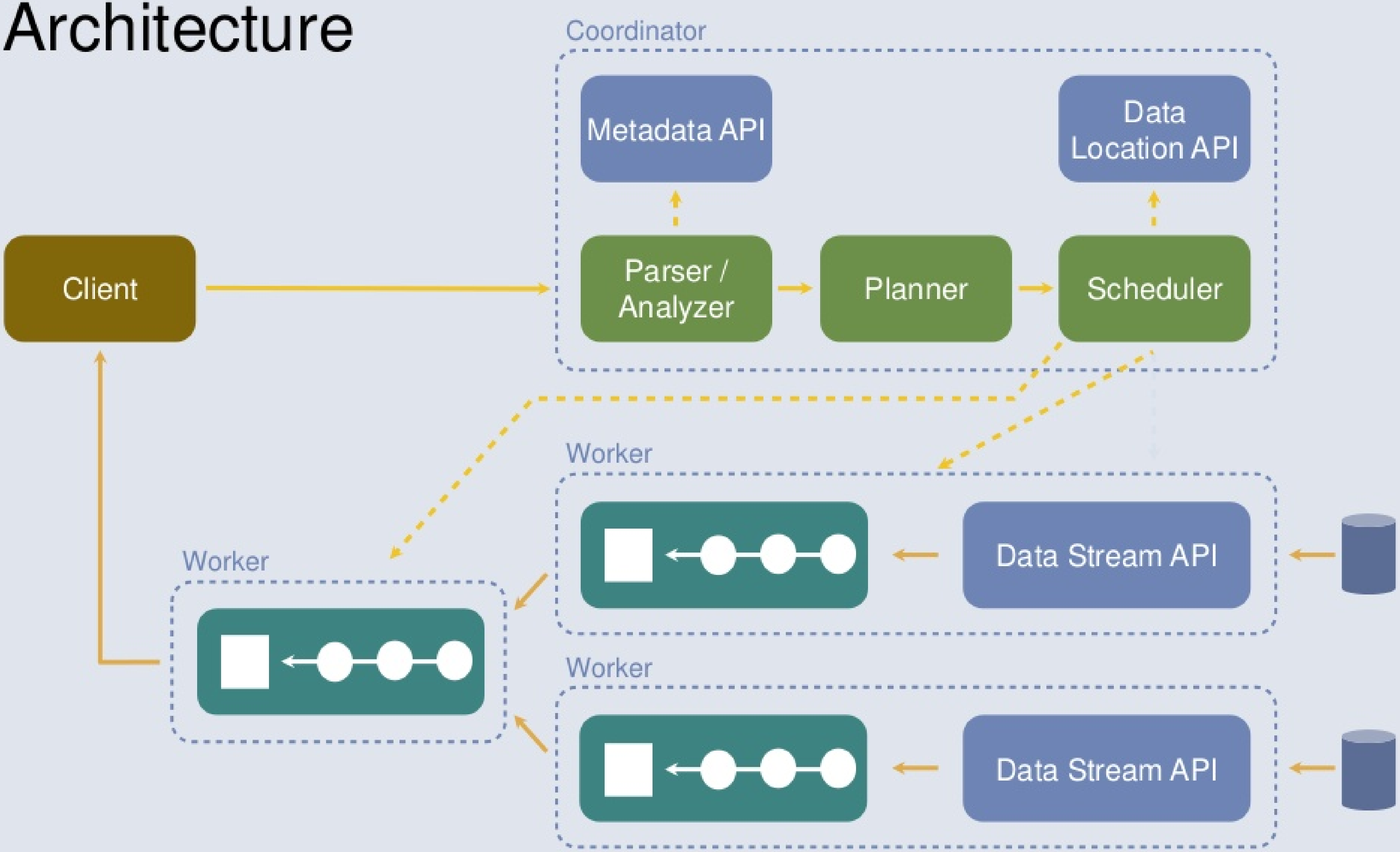
Worker

Hive Internal Cassandra MySQL Kafka

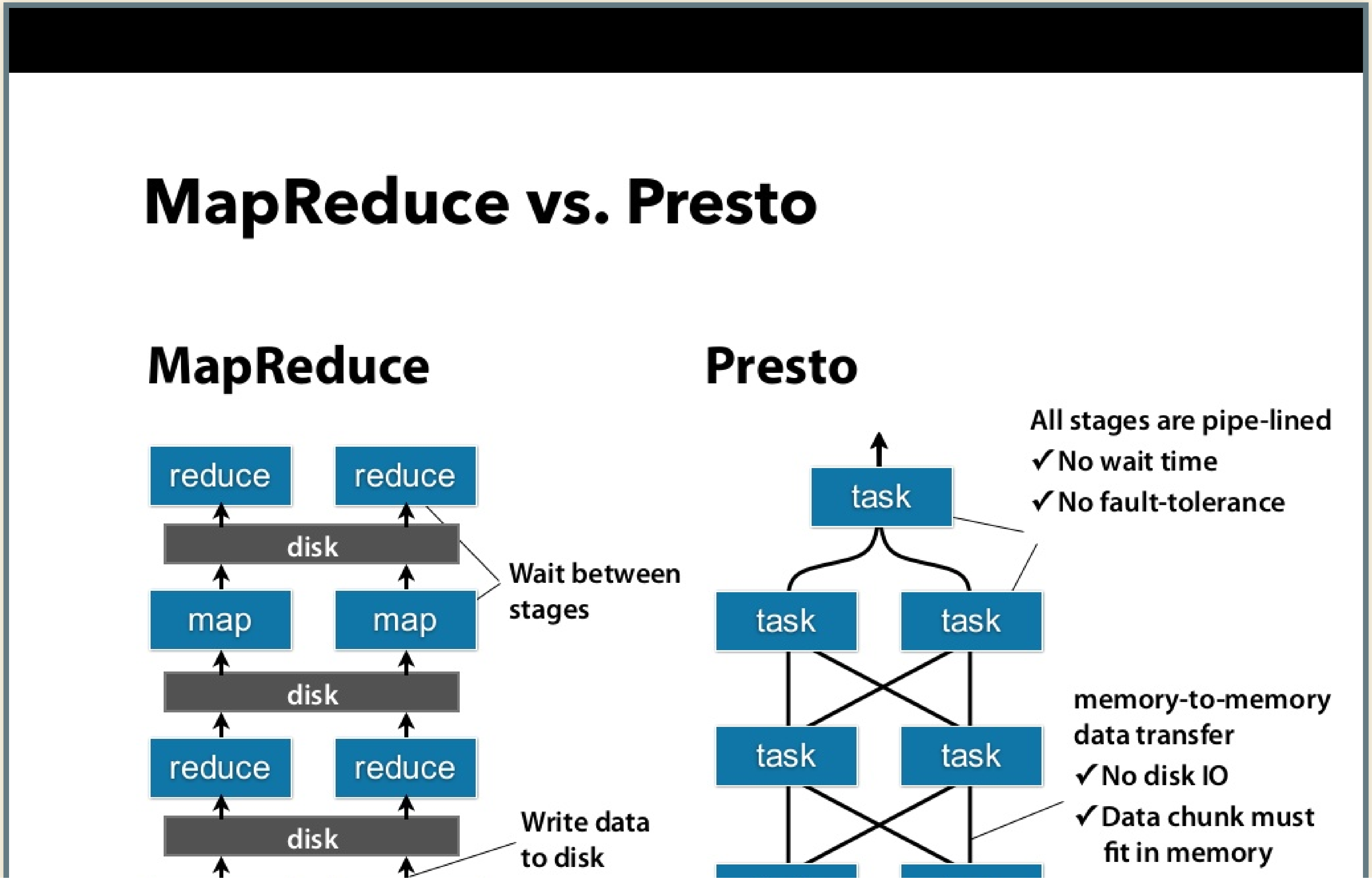
Data Stream API

ARCHITECTURE

Architecture



MR VS PRESTO



MapReduce vs. Presto

MapReduce

Wait between stages

Write data to disk

Presto

All stages are pipe-lined
✓No wait time
✓No fault-tolerance

memory-to-memory data transfer
✓No disk IO
✓Data chunk must fit in memory

map
map
task
task

DETAILS

- Data must fit in memory
- Uses connectors to several backends
    - Cassandra, Hive, JMX, Kafka, Mysql, Postgres
- Nodes are stateless

SUPPORT

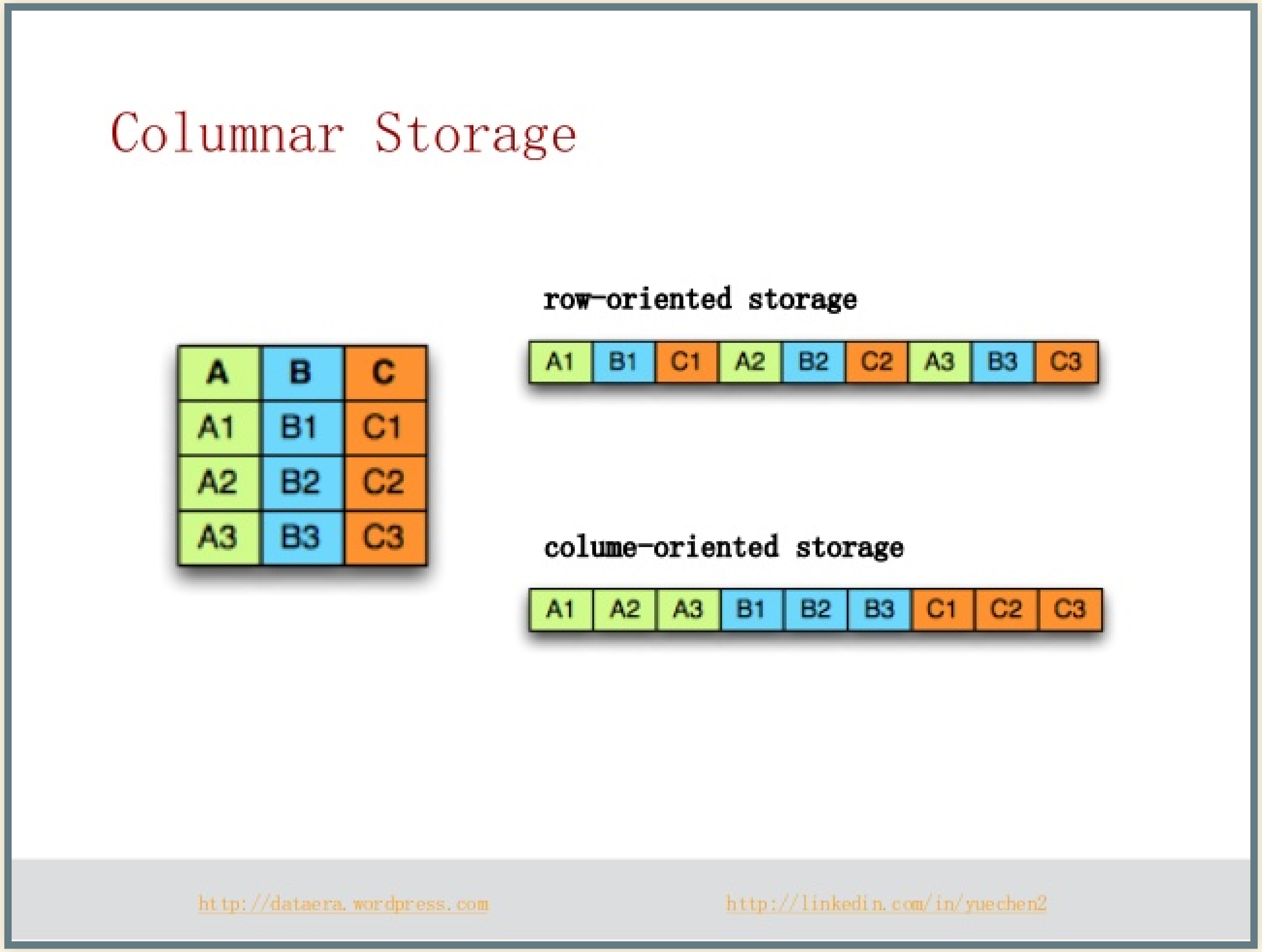
- Ansi SQL
- Approximation functions
- JSON functions
- PrestoML

TRICKS

BEST ONES

- **Vectorized Reader**: read based on column vectors
- **Predicate Pushdown**: use statistics/logic to skip data
- **Lazy Load**: postpone loading until needed
- **Lazy materialization**: postpone decoding until needed

2.1 COLUMNAR STORE



Columnar Storage

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

row-oriented storage

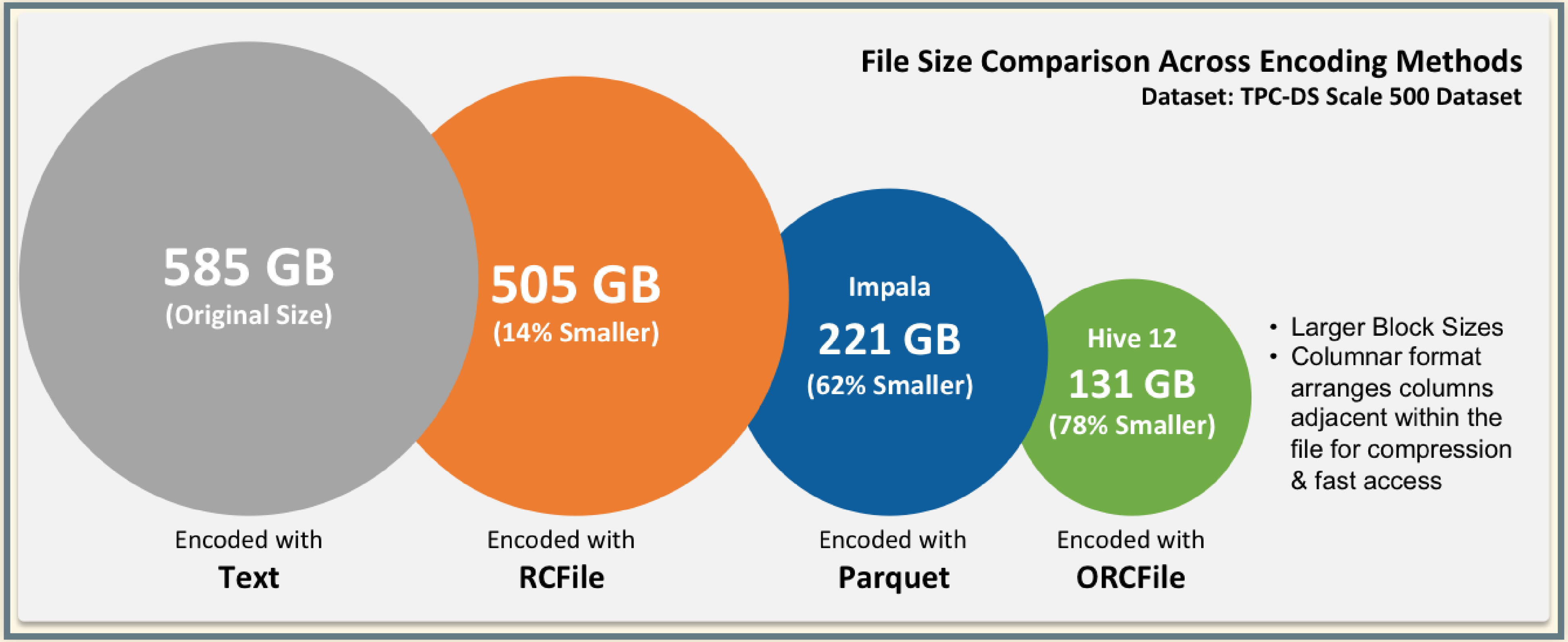
A1	B1	C1	A2	B2	C2	A3	B3	C3

colume-oriented storage

A1	A2	A3	B1	B2	B3	C1	C2	C3

http://dataera.wordpress.com http://linkedin.com/in/yuechen2

2.2 FILE SIZE COMPARISON

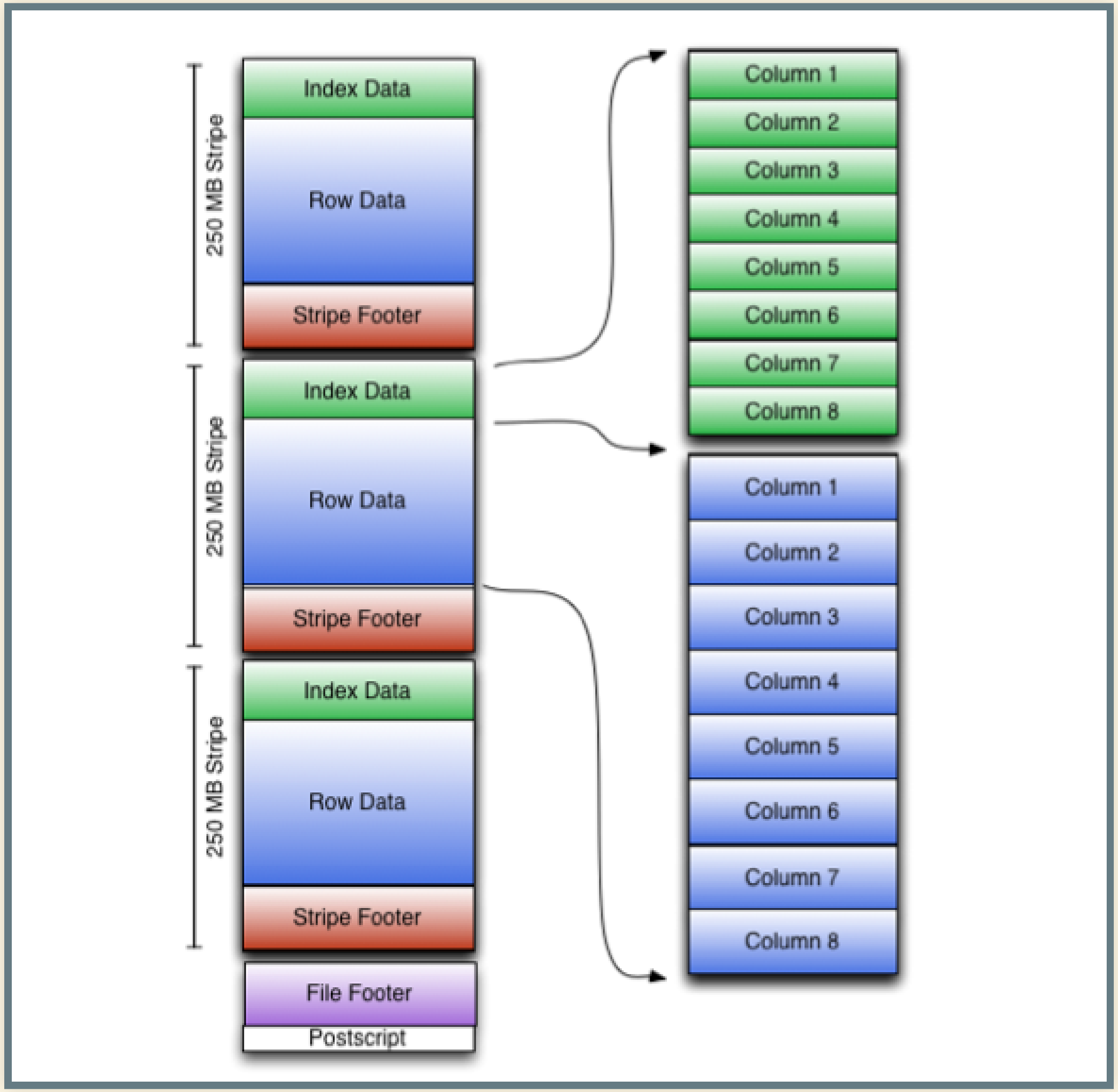


File Size Comparison Across Encoding Methods

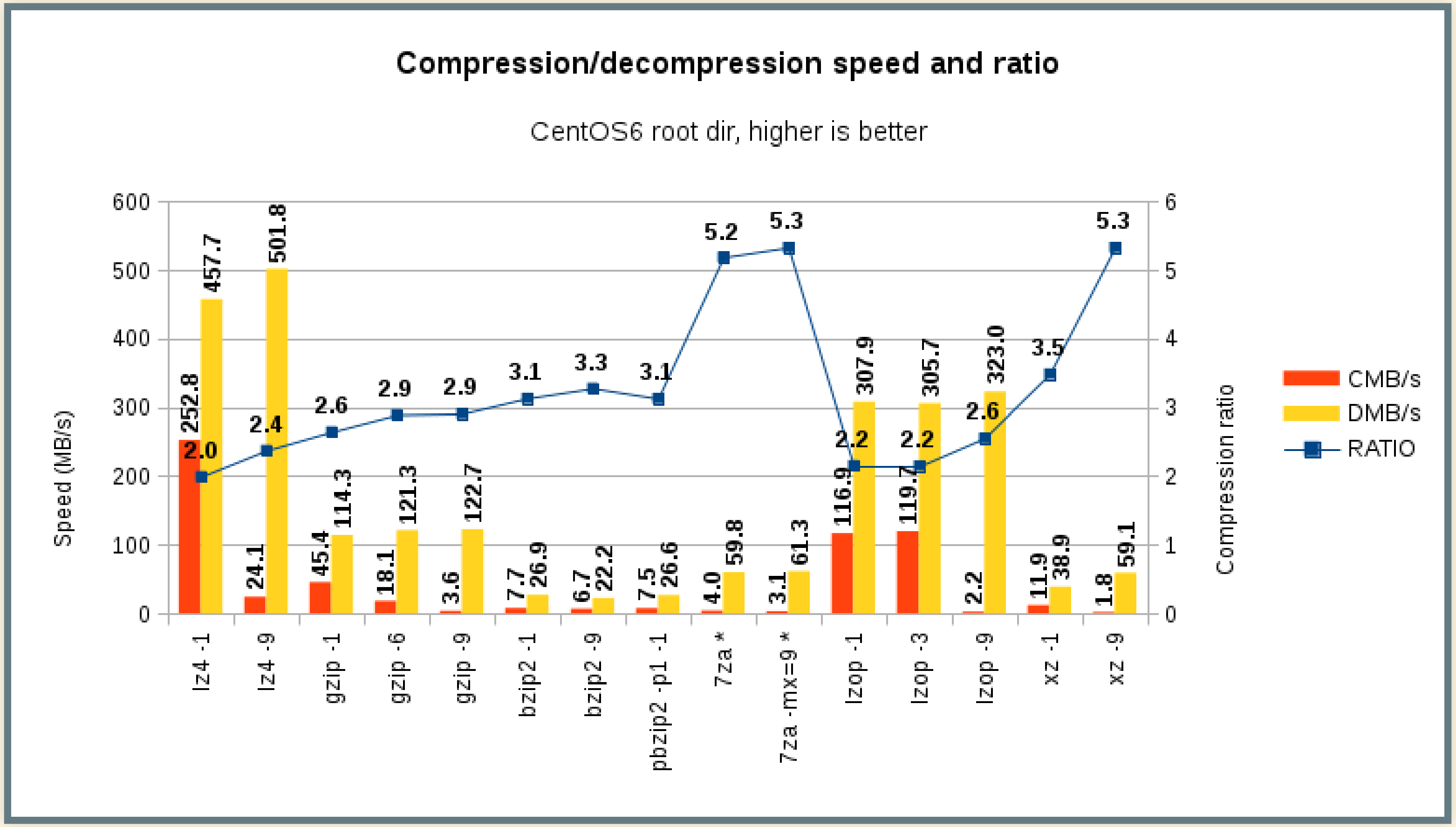
Dataset: TPC-DS Scale 500 Dataset

- Larger Block Sizes
- Columnar format arranges columns adjacent within the file for compression & fast access

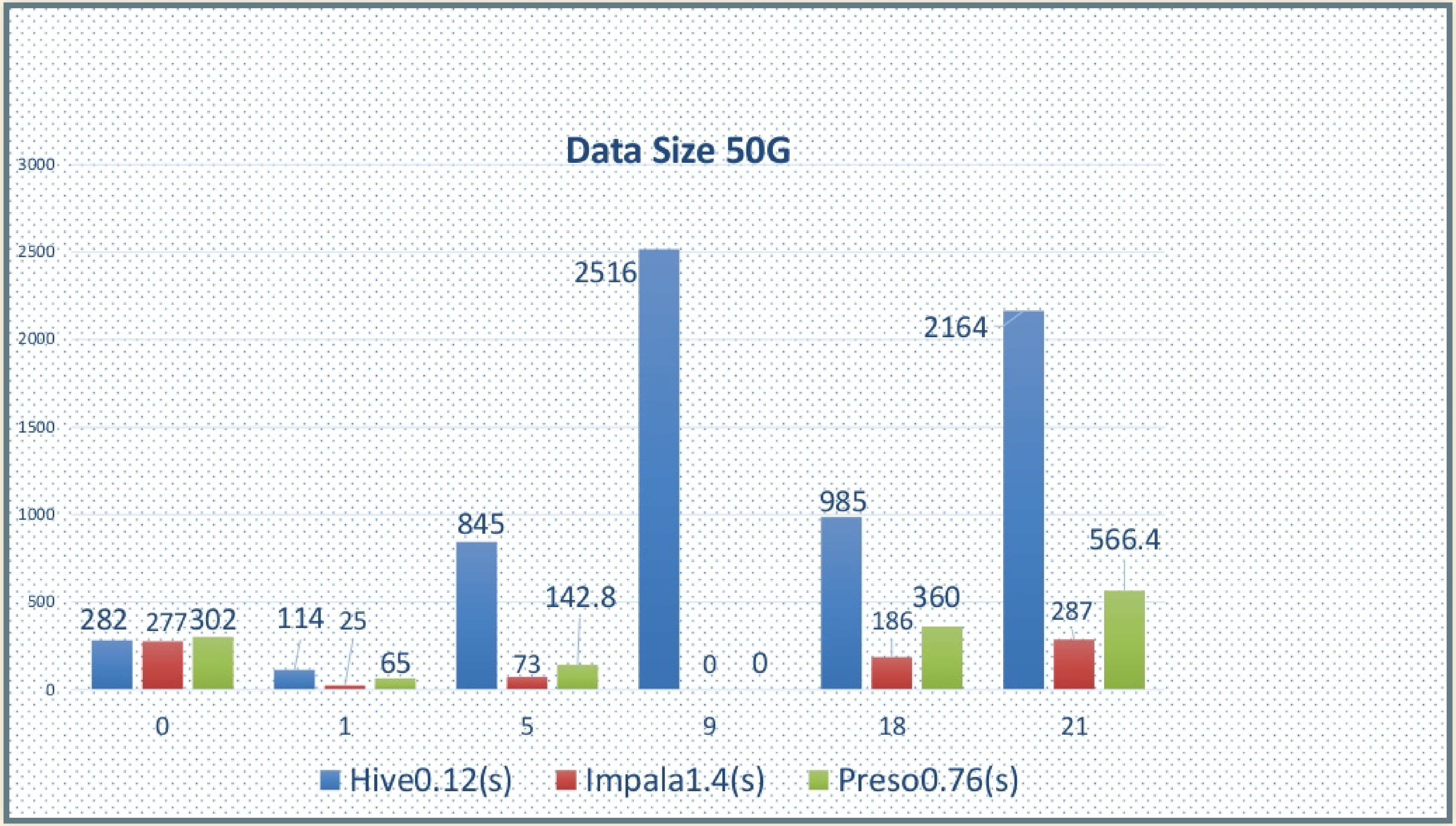
FILE LAYOUT


250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
File Footer
Postscript
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8

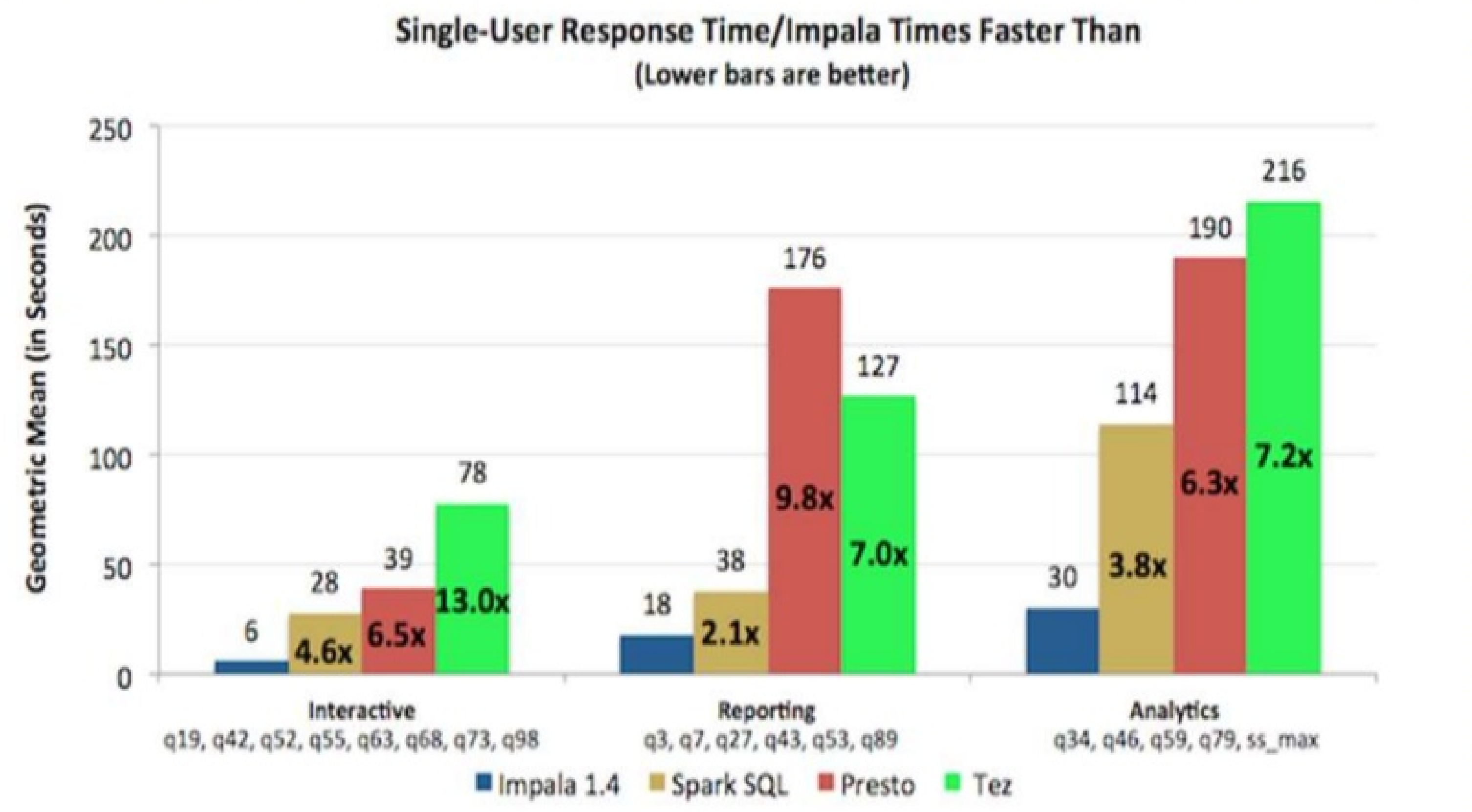
COMPRESSION



BENCHMARKS


Data Size 50G
3000
2500
2000
1500
1000
500
0
282
277
302
114
25
65
845
73
142.8
2516
0
0
985
186
360
2164
287
566.4
0
1
5
9
18
21
Hive0.12(s)
Impala1.4(s)
Preso0.76(s)

BENCHMARKS


Single-User Response Time/Impala Times Faster Than
(Lower bars are better)
Geometric Mean (in Seconds)
250
200
150
100
50
0
6
28
4.6x
39
6.5x
78
13.0x
18
38
2.1x
176
9.8x
127
7.0x
30
114
3.8x
190
6.3x
216
7.2x
Interactive
q19, q42, q52, q55, q63, q68, q73, q98
Reporting
q3, q7, q27, q43, q53, q89
Analytics
q34, q46, q59, q79, ss_max
Impala 1.4
Spark SQL
Presto
Tez

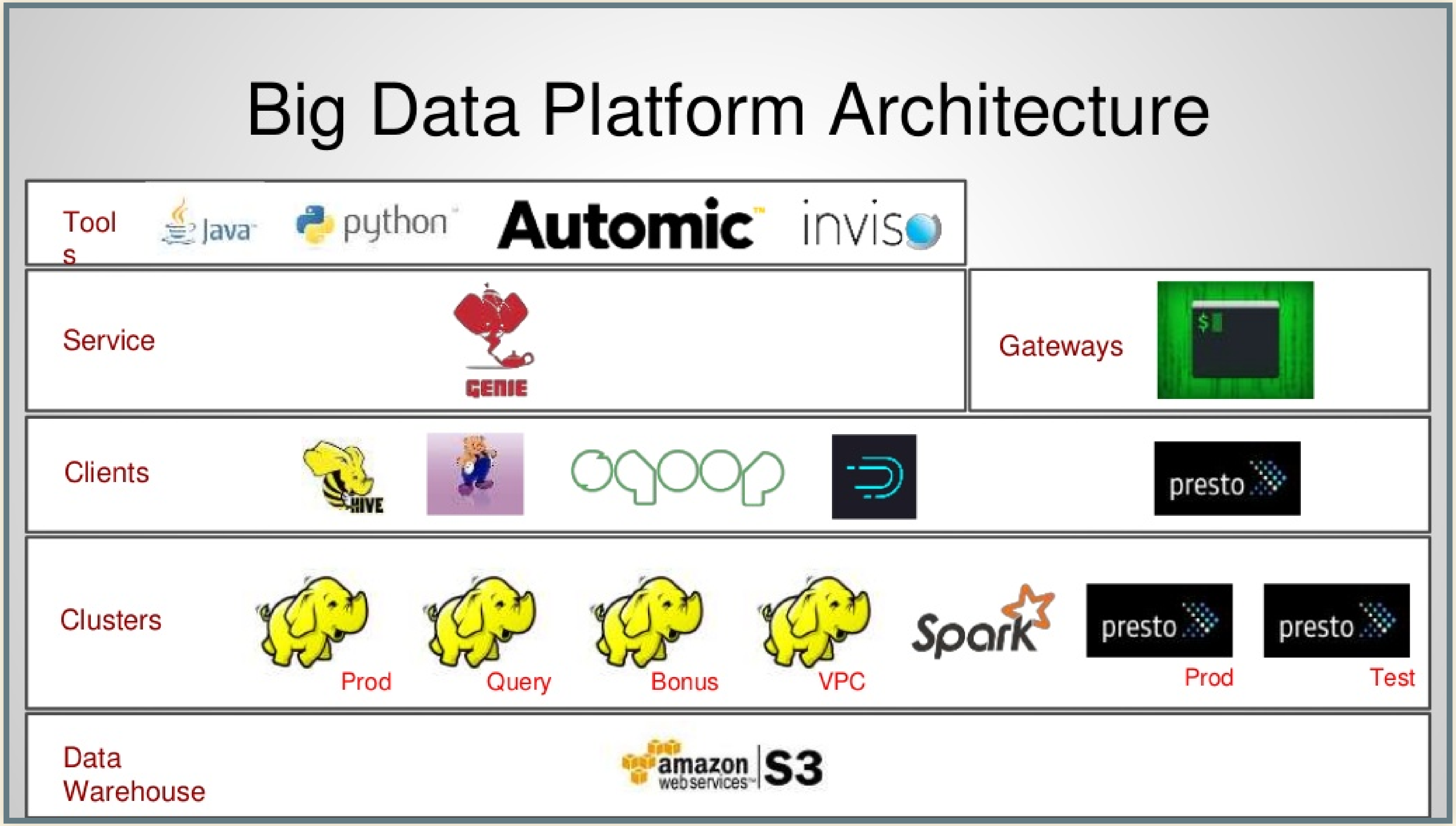
OTHERS

- Fast inner loops (related to CPU cache)
- sun.misc.Unsafe (non-gc memory access)
- Pipelining, streaming

NOTES

- SQL parser is being used in other projects (ex: crate.io)
- Airpal - UI for presto
- There's also Impala, Apache Drill, Apache Tajo, Redshift, Spark, etc
- PaaS - Presto as a service called Qubole

NETFLIX



Big Data Platform Architecture

Tools

Service

Gateways

Clients

Clusters

Prod

Query

Bonus

VPC

Prod

Test

Data Warehouse

http://www.slideshare.net/treasure-data/2015-0311td-techtalkinternalsofprestoservice?qid=702c79ef-0632-476b-abb0-0aaff121cf00&v=default&b=&from_search=12