Developer Training for Spark and Hadoop

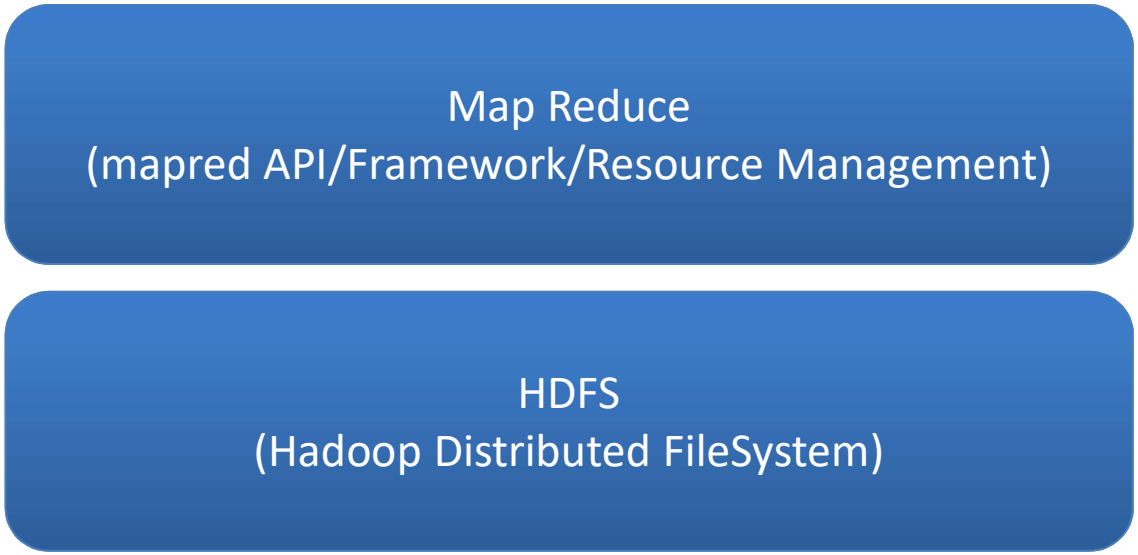
申建忠

Frank_shen@uuu.com.tw

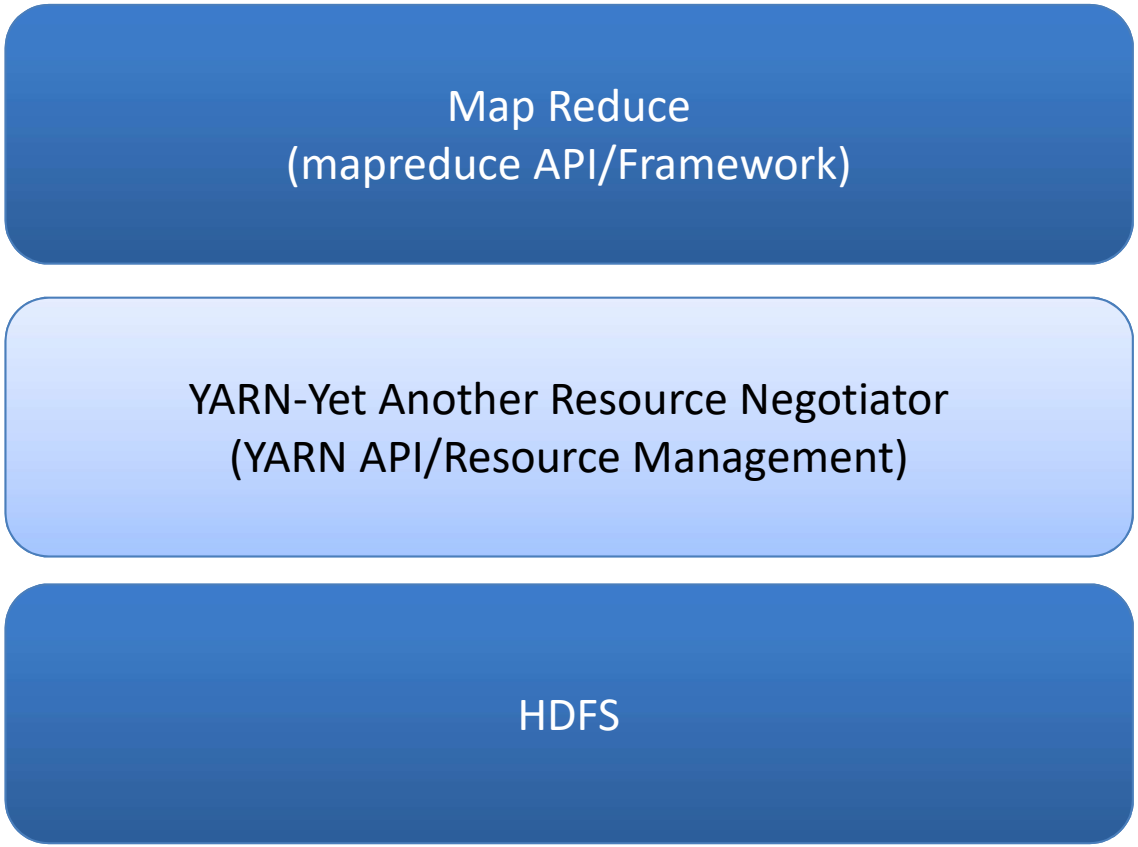
Hadoop介紹

- 創作者 : Doug Cutting
- 開發語言 : Java
- 靈感啟發 : 2004 Google所發表的2篇論文
  - MapReduce -> MapReduce
  - GFS -> HDFS
- 2006 Apache Top Project
  - MapReduce : Max utilization of CPU
  - HDFS : Max utilization of Disk

Core Hadoop(MR v1)

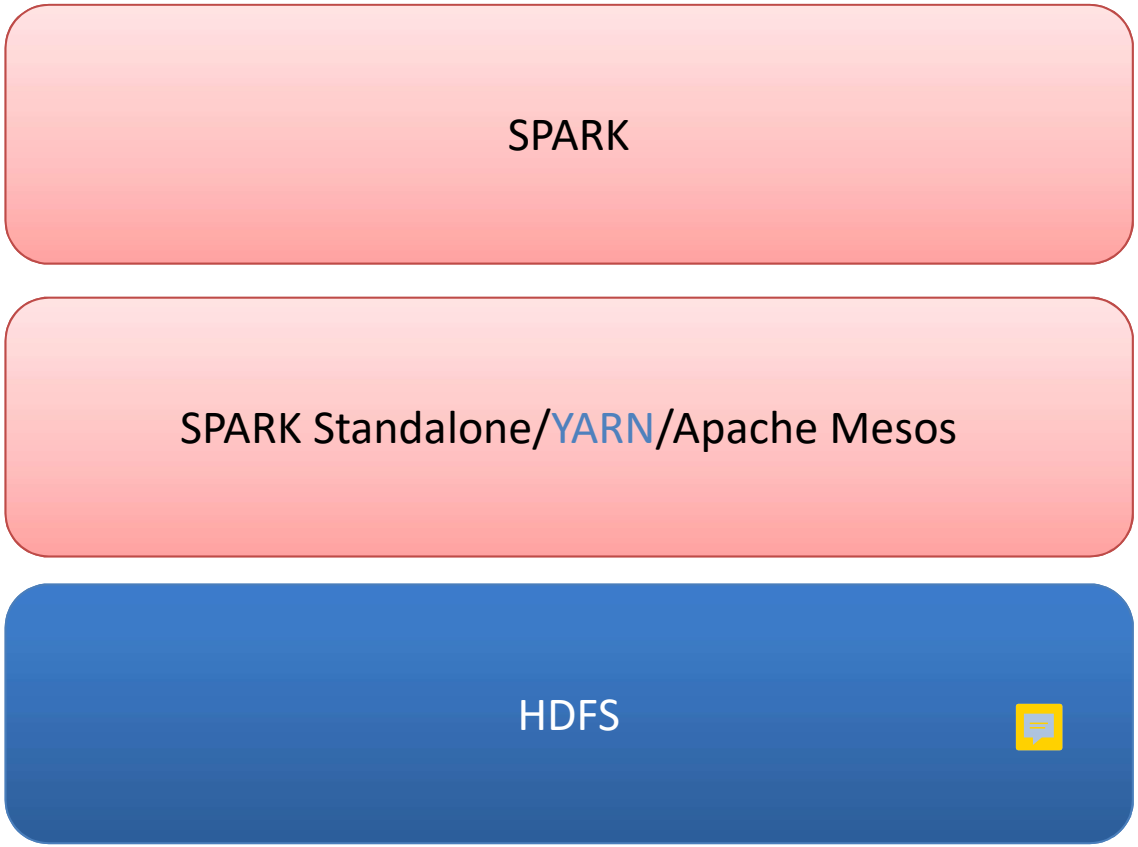


Core Hadoop(MR v2)

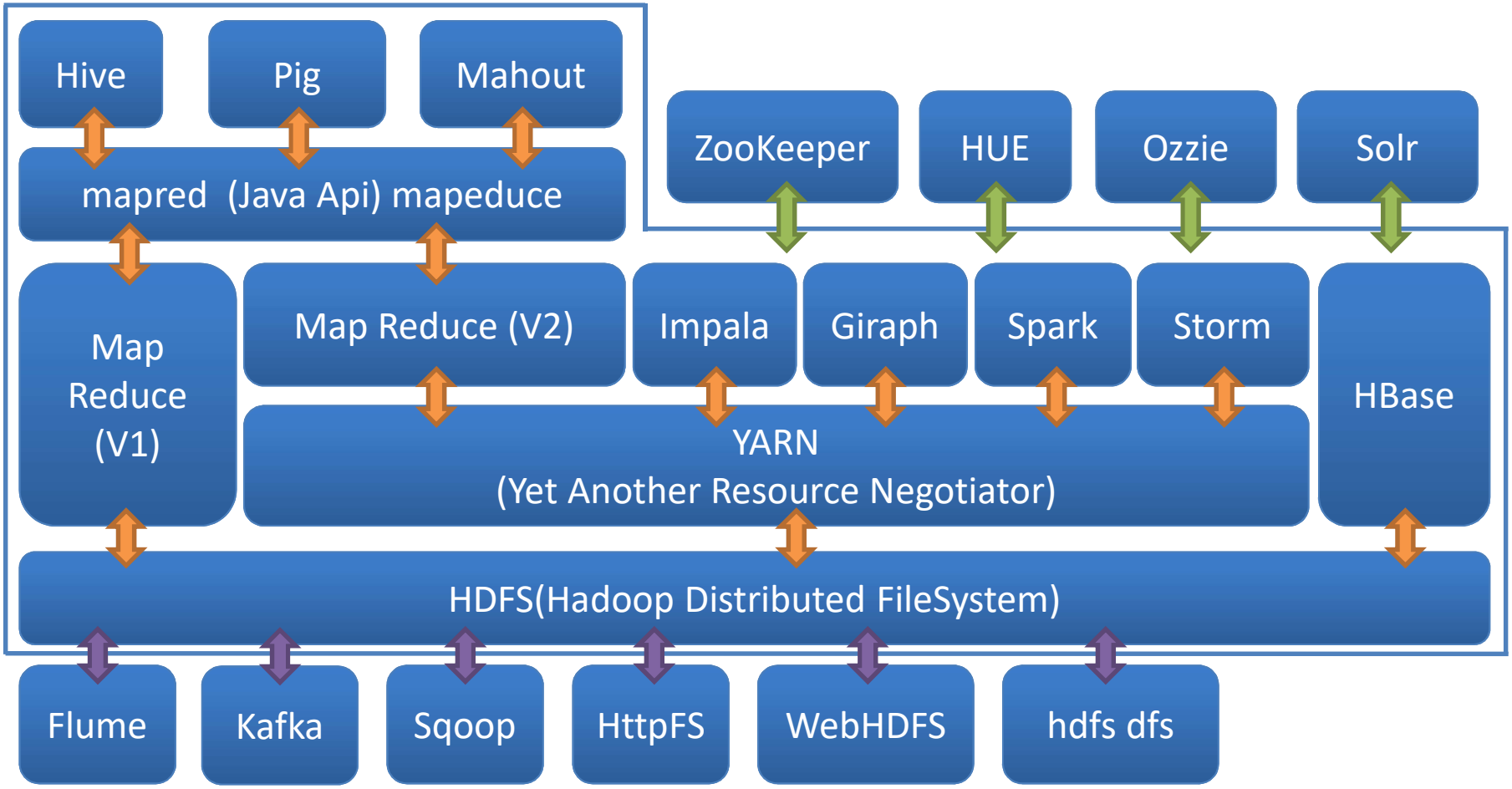


HDFS+YARN+SPARK

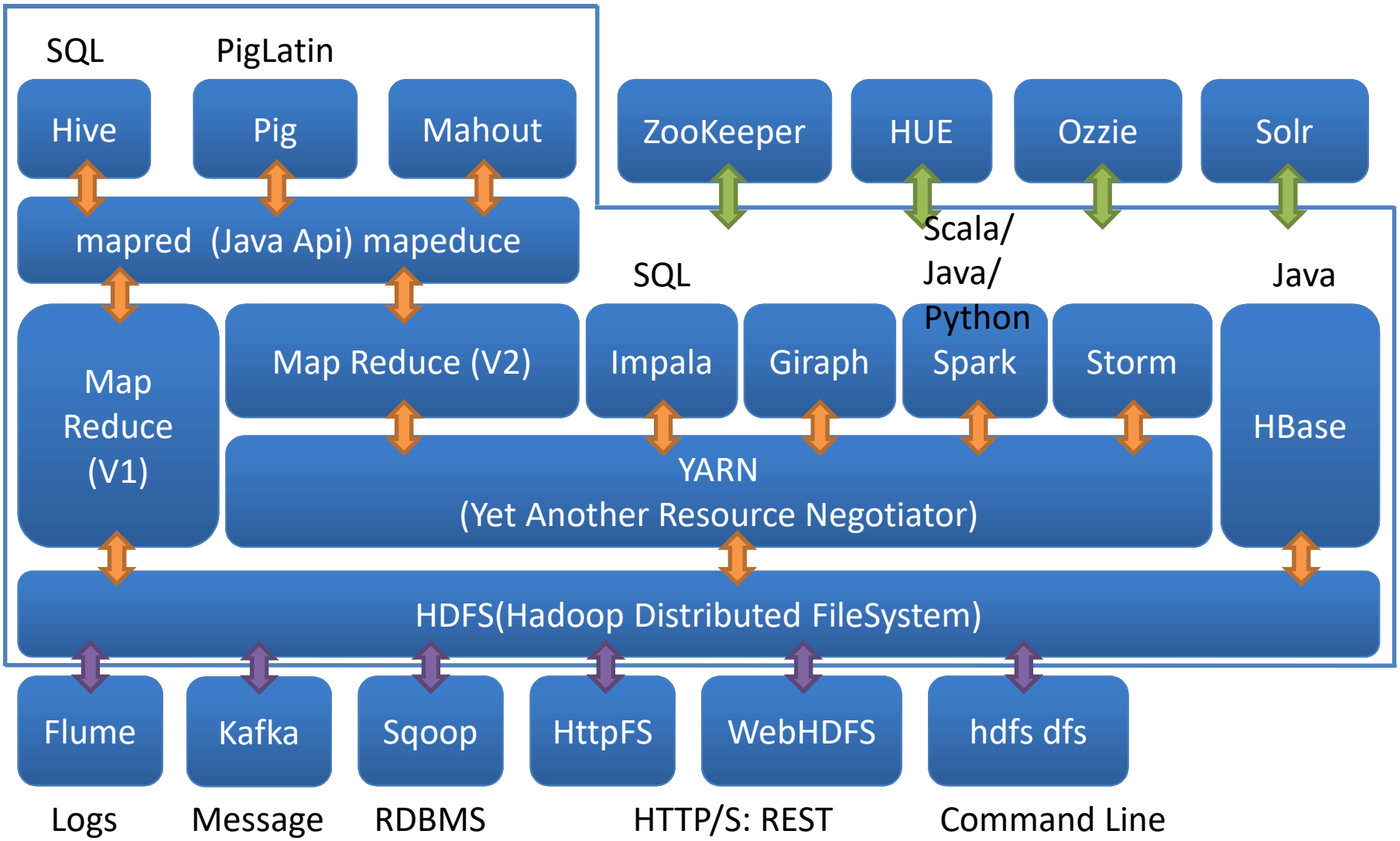
HDFS+SPARK



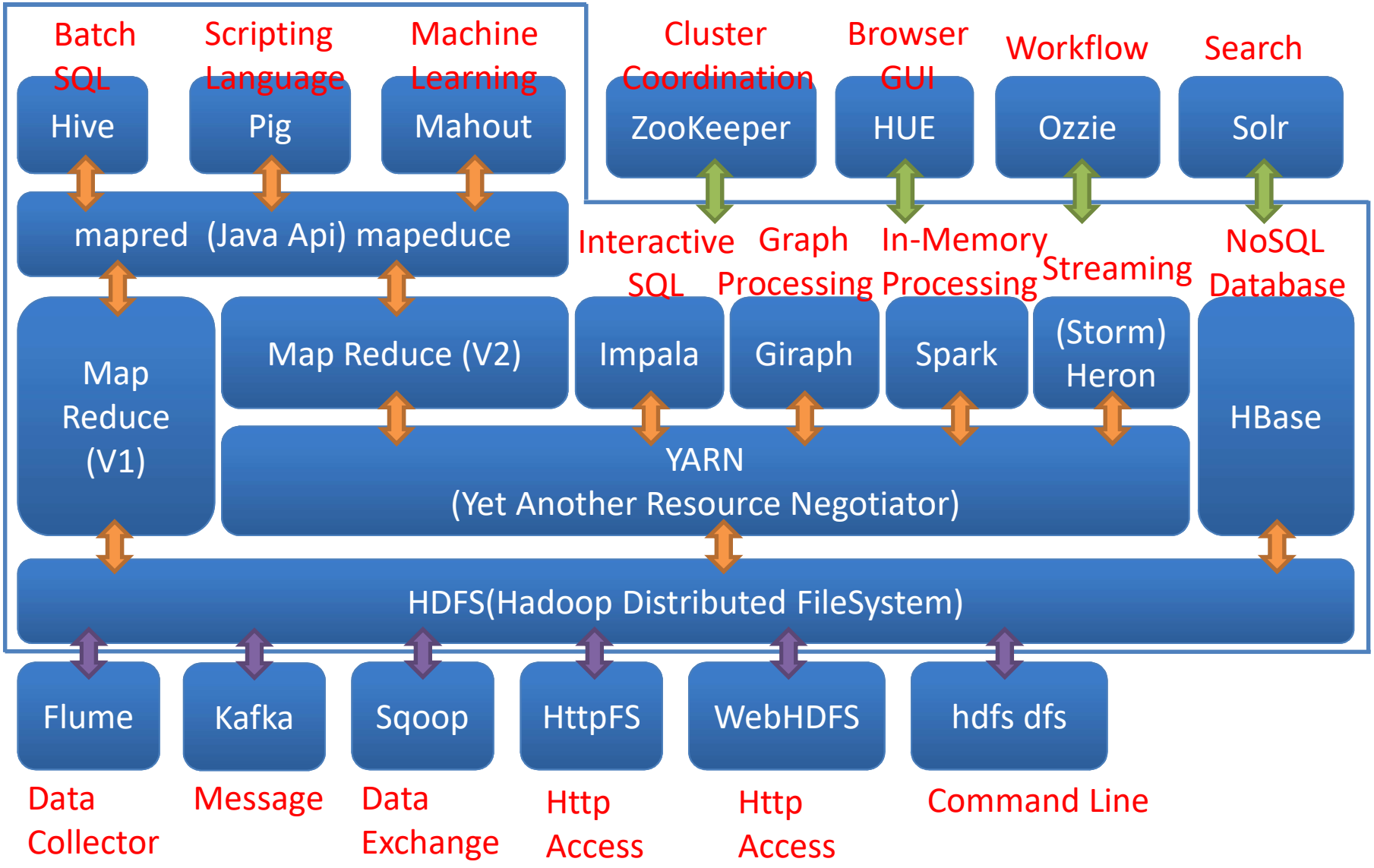
Hadoop Ecosystem



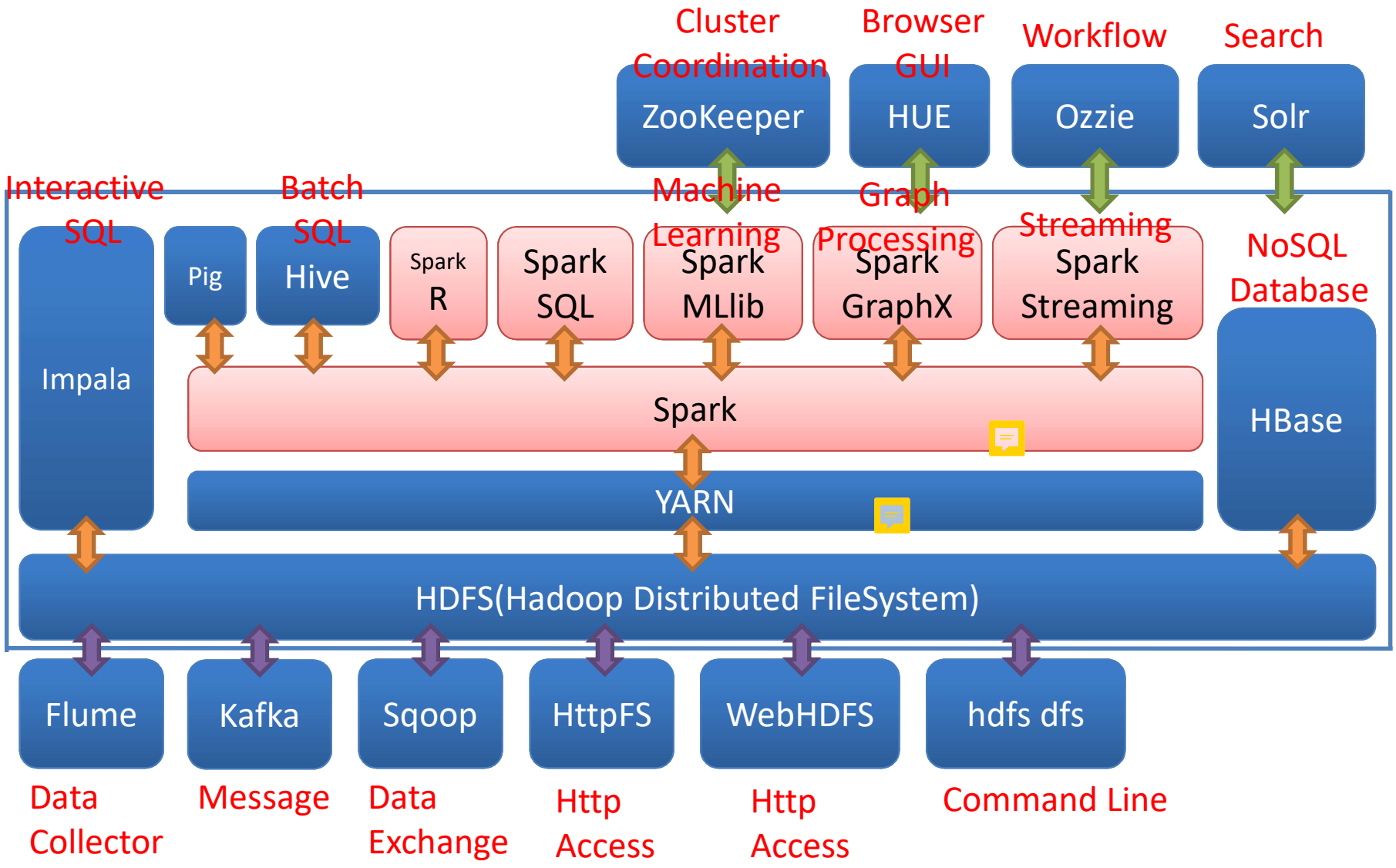
Hadoop Ecosystem

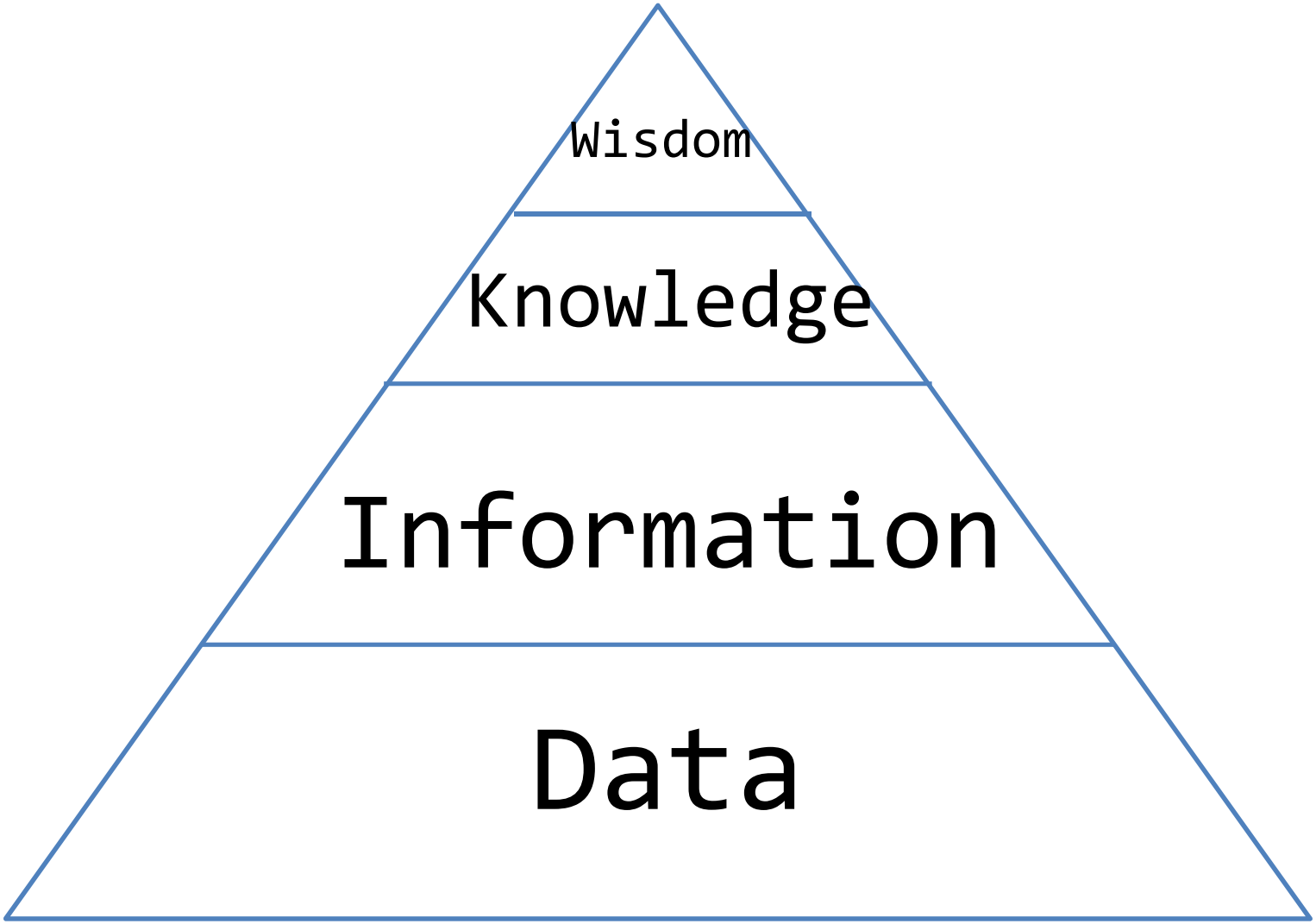


Hadoop Ecosystem



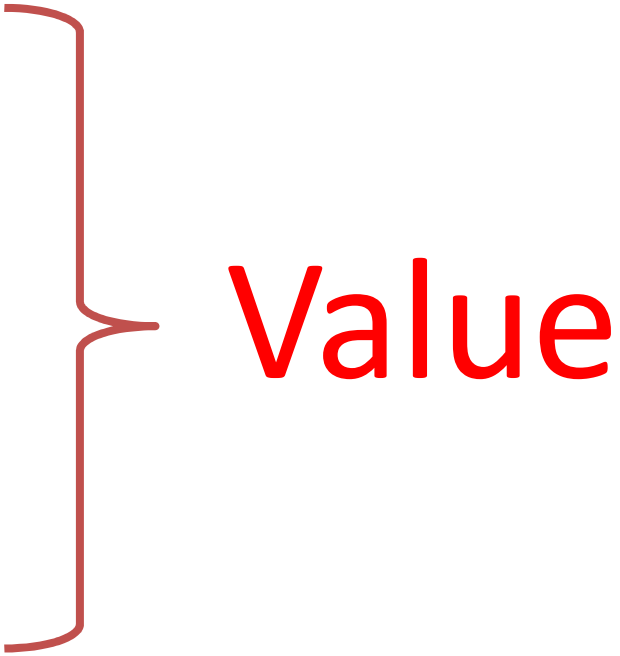
Hadoop Ecosystem+SPARK




Wisdom
Knowledge
Information
Data

Big Data

- Volume
  - Data amount
- Variety
  - Data type
- Velocity
  - Data processing Speed
  - Generated



How big is BIG

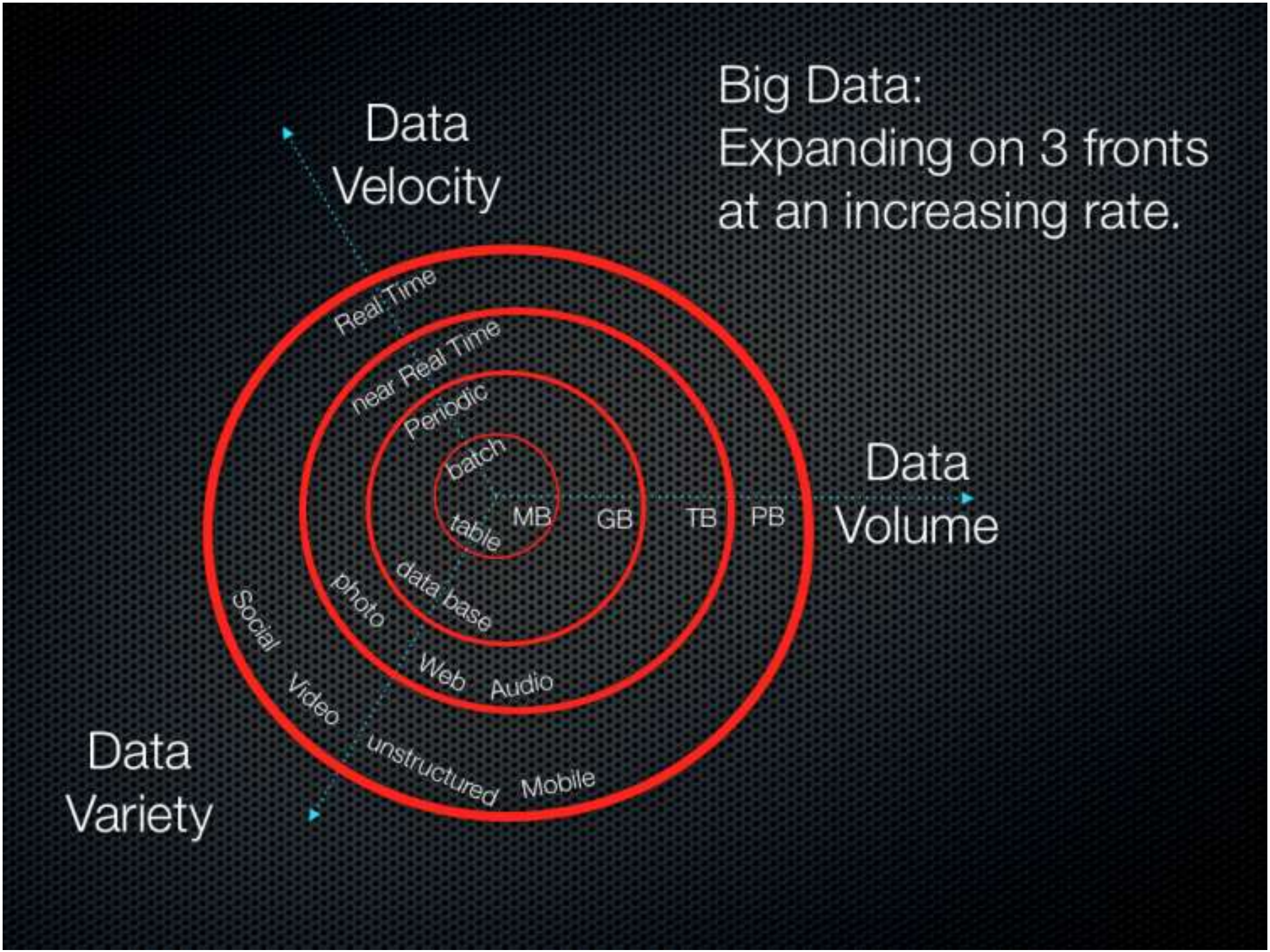
- 大數據（或Megadata），或稱巨量資料、海量資料、大資料，指的是所涉及的資料量規模巨大到無法透過人工，在合理時間內達到擷取、管理、處理、並整理成為人類所能解讀的資訊。
- 在總資料量相同的情況下，與個別分析獨立的小型資料集（data set）相比，將各個小型資料集合併後進行分析可得出許多額外的資訊和資料關聯性，可用來察覺商業趨勢、判定研究品質、避免疾病擴散、打擊犯罪或測定即時交通路況等；這樣的用途正是大型資料集盛行的原因。

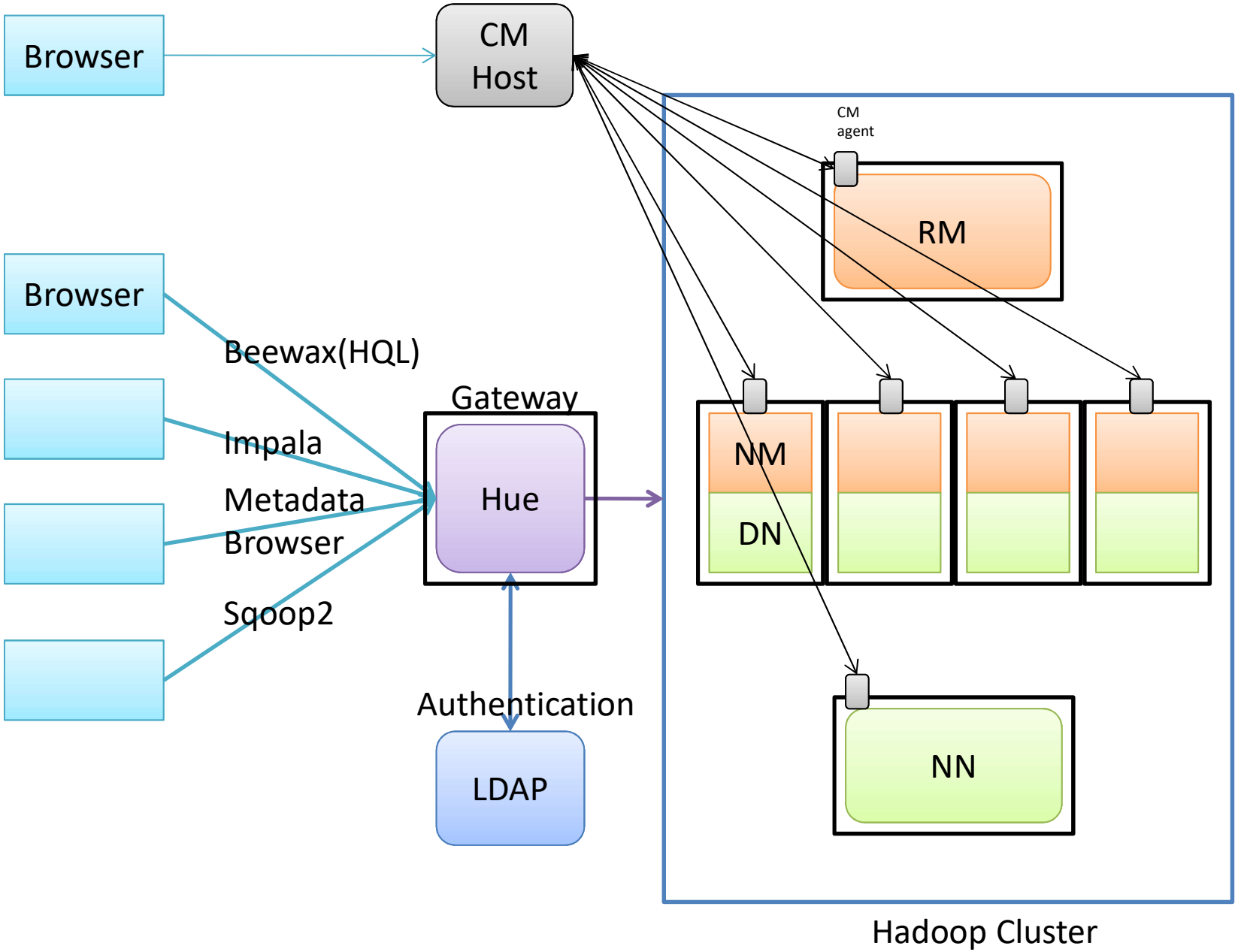
How many is Variety

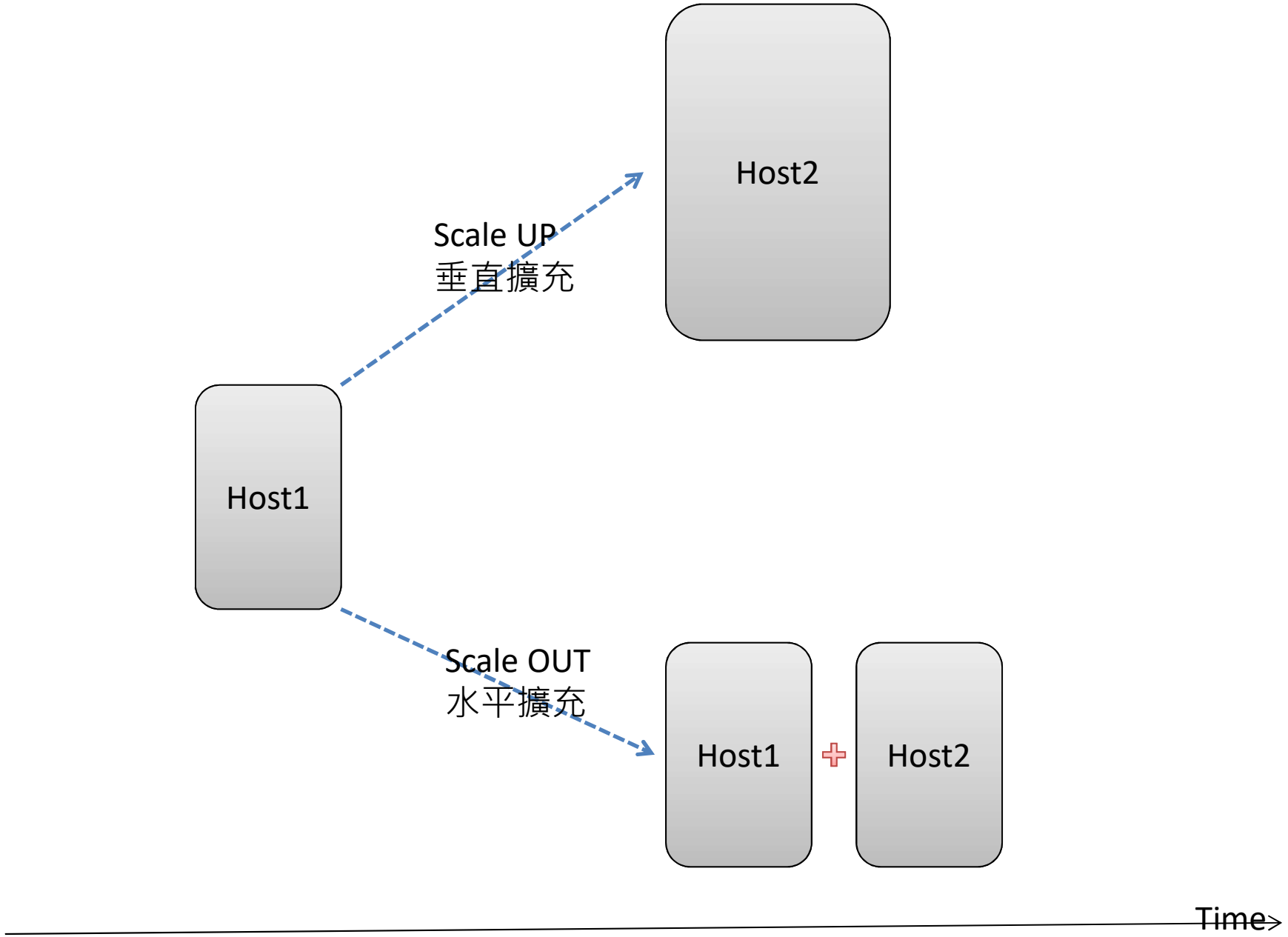
- 大數據的來源種類包羅萬象，十分多樣化，如果一定要把資料分類的話，最簡單的方法是分兩類，結構化與非結構化。早期的非結構化資料主要是文字，隨著網路的發展，又擴展到電子郵件、網頁、社交媒體、視訊，音樂、圖片等等，這些非結構化的資料造成儲存（storage）、探勘（mining）、分析（analyzing）上的困難。當然還有原本就已經儲存在傳統資料庫的資料。
- 資料集大小增長的部分原因來自於資訊持續從各種來源被廣泛收集，這些來源包括搭載感測裝置的行動裝置、高空感測科技（遙感）、軟體記錄、相機、麥克風、無線射頻辨識（RFID）和無線感測網路。
- 自1980年代起，現代科技可儲存資料的容量每40個月即增加一倍；截至2012年，全世界每天產生2.5 EB（2.5×10^{18}）的資料。

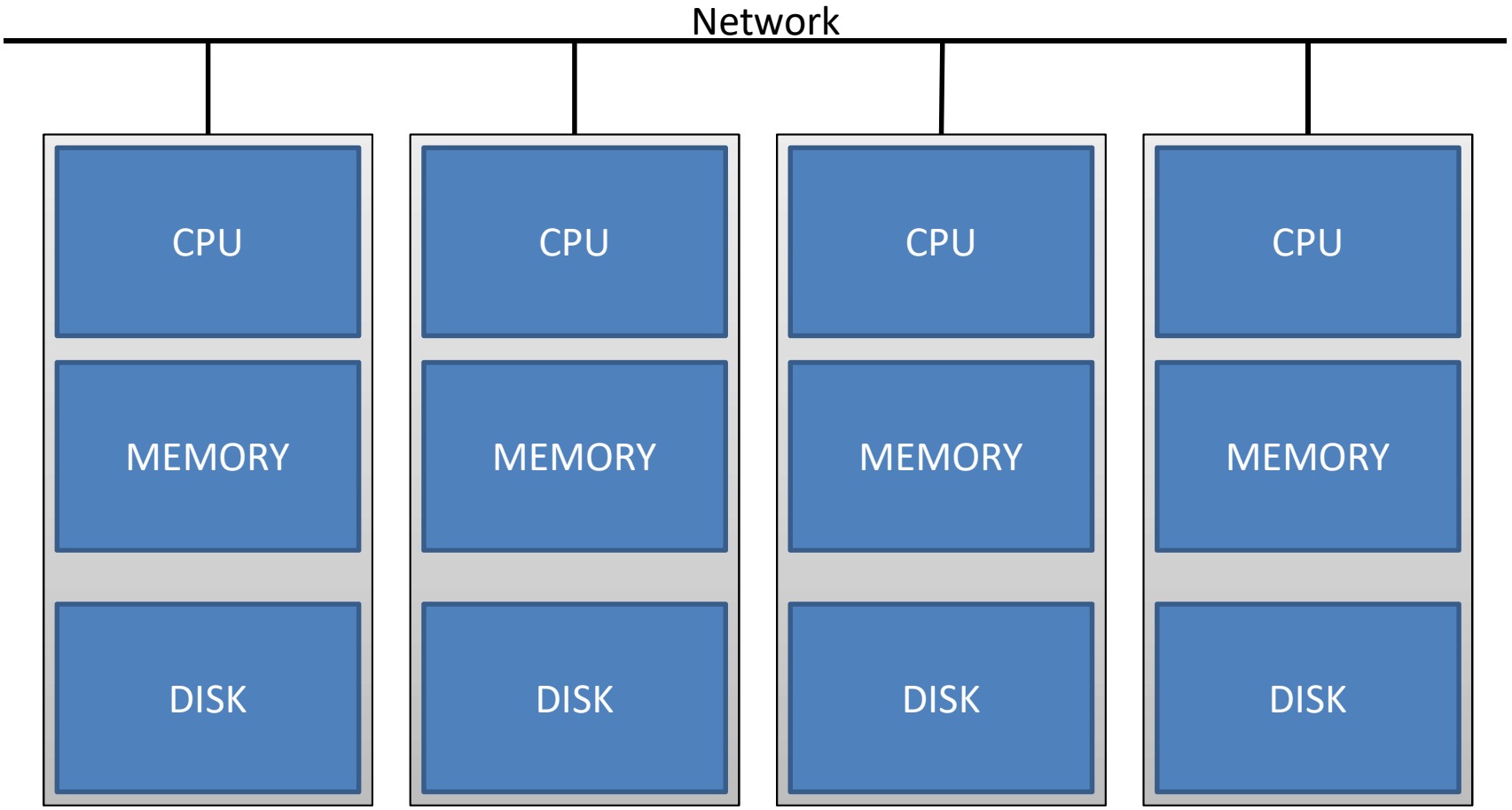
How fast is Velocity

- 資料的傳輸流動（data streaming）是連續且快速的，隨著越來越多的機器、網路使用者，社群網站、搜尋結果每秒都在成長，每天都在輸出更多的內容。公司跟機構要處理龐大的資訊大潮向他們襲來，而回應、反應這些資料的速度也成為他們最大的挑戰，許多資料要能即時得到結果才能發揮最大的價值，因此也有人認為 Velocity 是「時效性」。

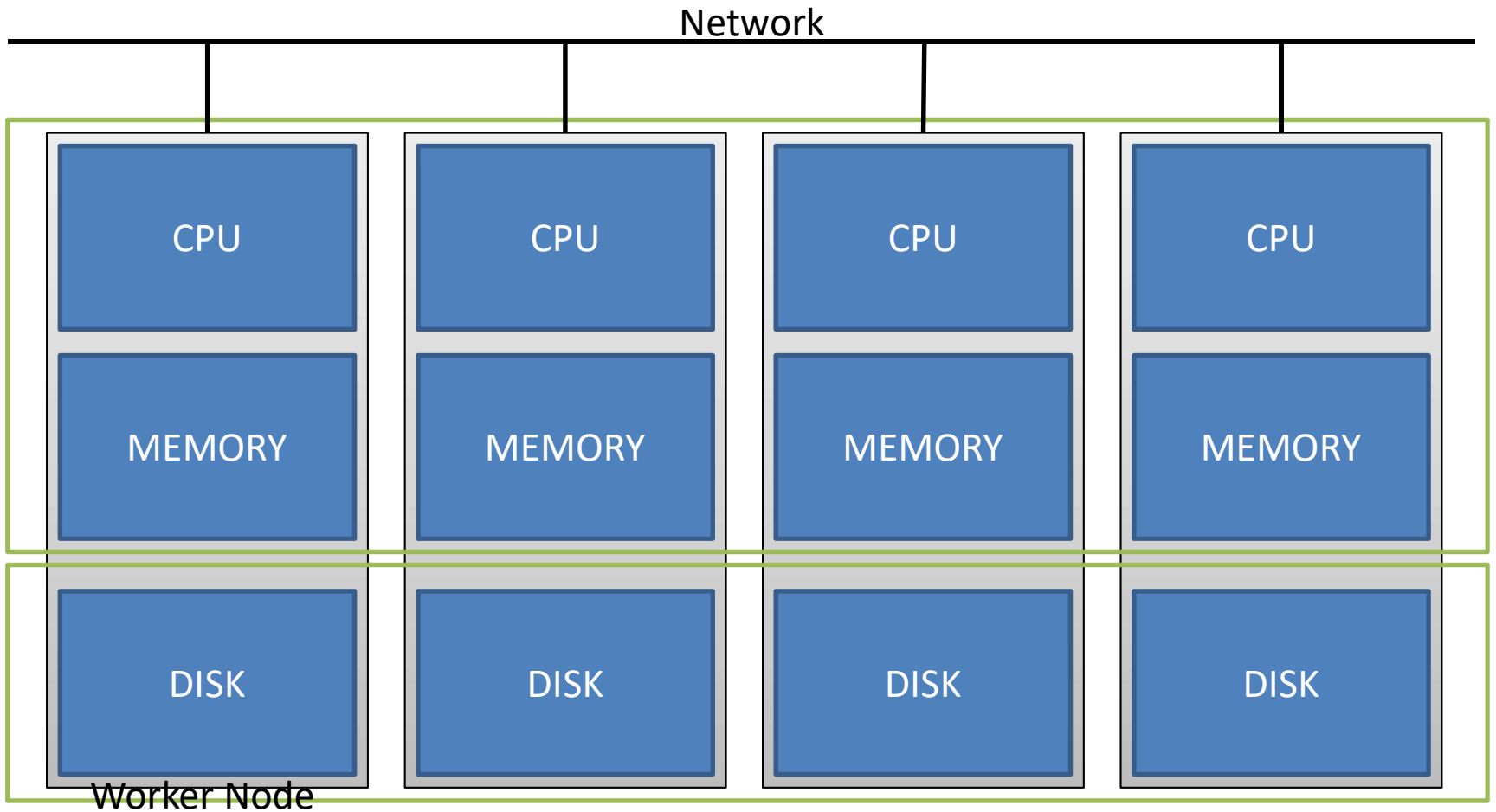

Big Data:
Expanding on 3 fronts
at an increasing rate.
Data
Velocity
Real Time
near Real Time
Periodic
batch
table
MB
GB
TB
PB
data base
photo
Social
Web
Audio
Video
unstructured
Mobile
Data
Volume
Data
Variety


Browser
CM Host
CM agent
RM
Browser
Beewax(HQL)
Gateway
Impala
Hue
NM
DN
Metadata
Browser
Sqoop2
Authentication
LDAP
NN
Hadoop Cluster


Host2
Scale UP
垂直擴充
Host1
Scale OUT
水平擴充
Host1
Host2
Time

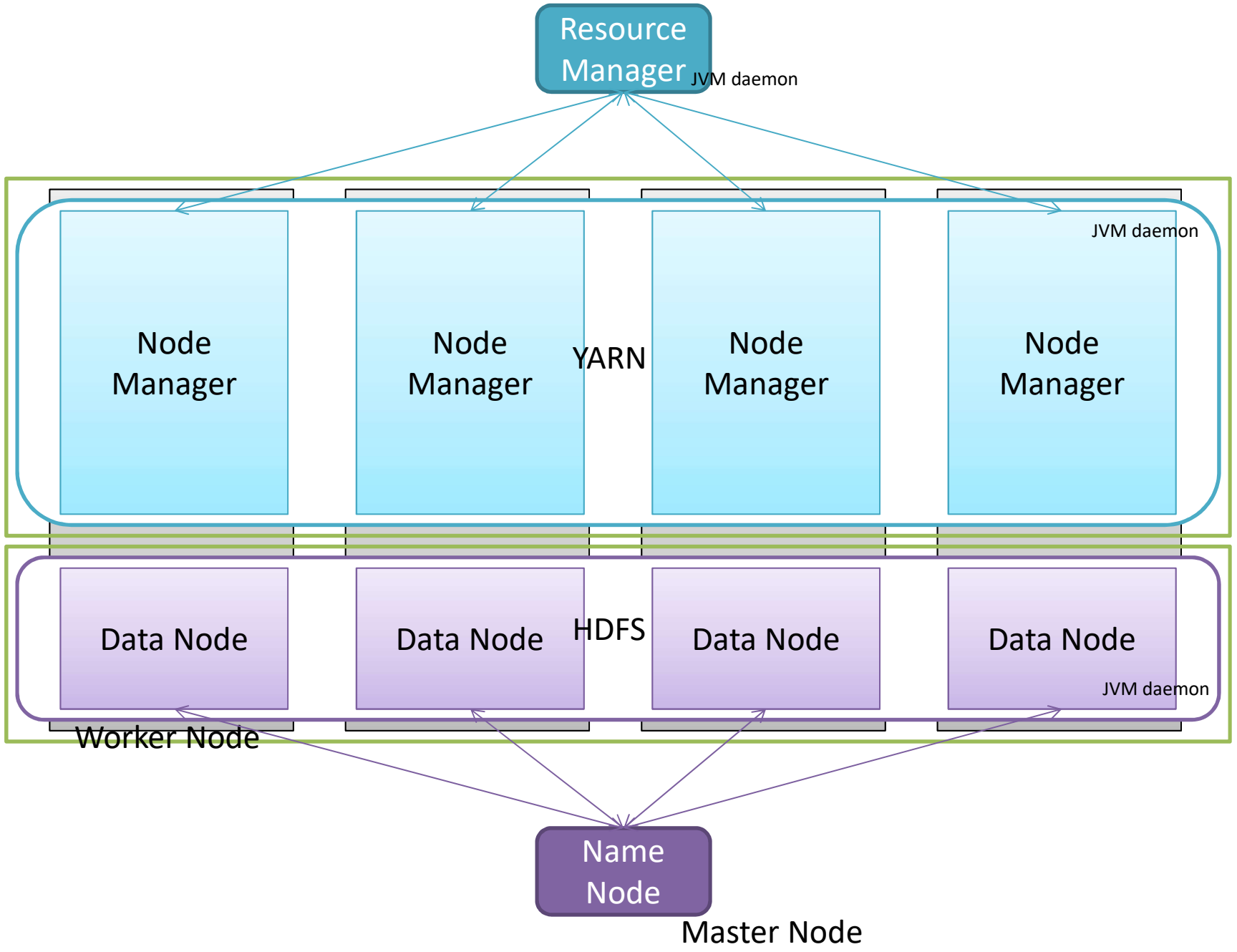


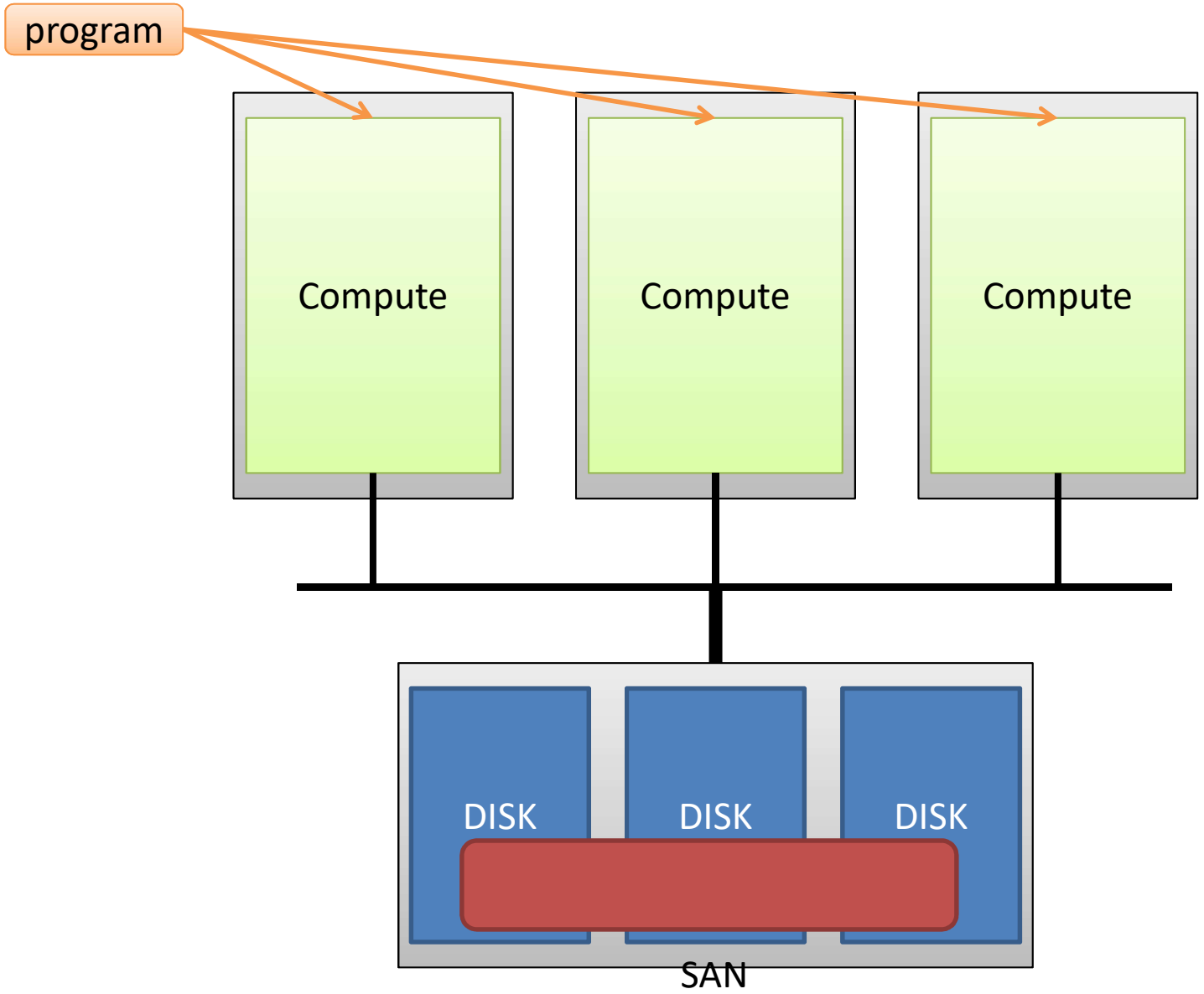
Share Nothing Cluster

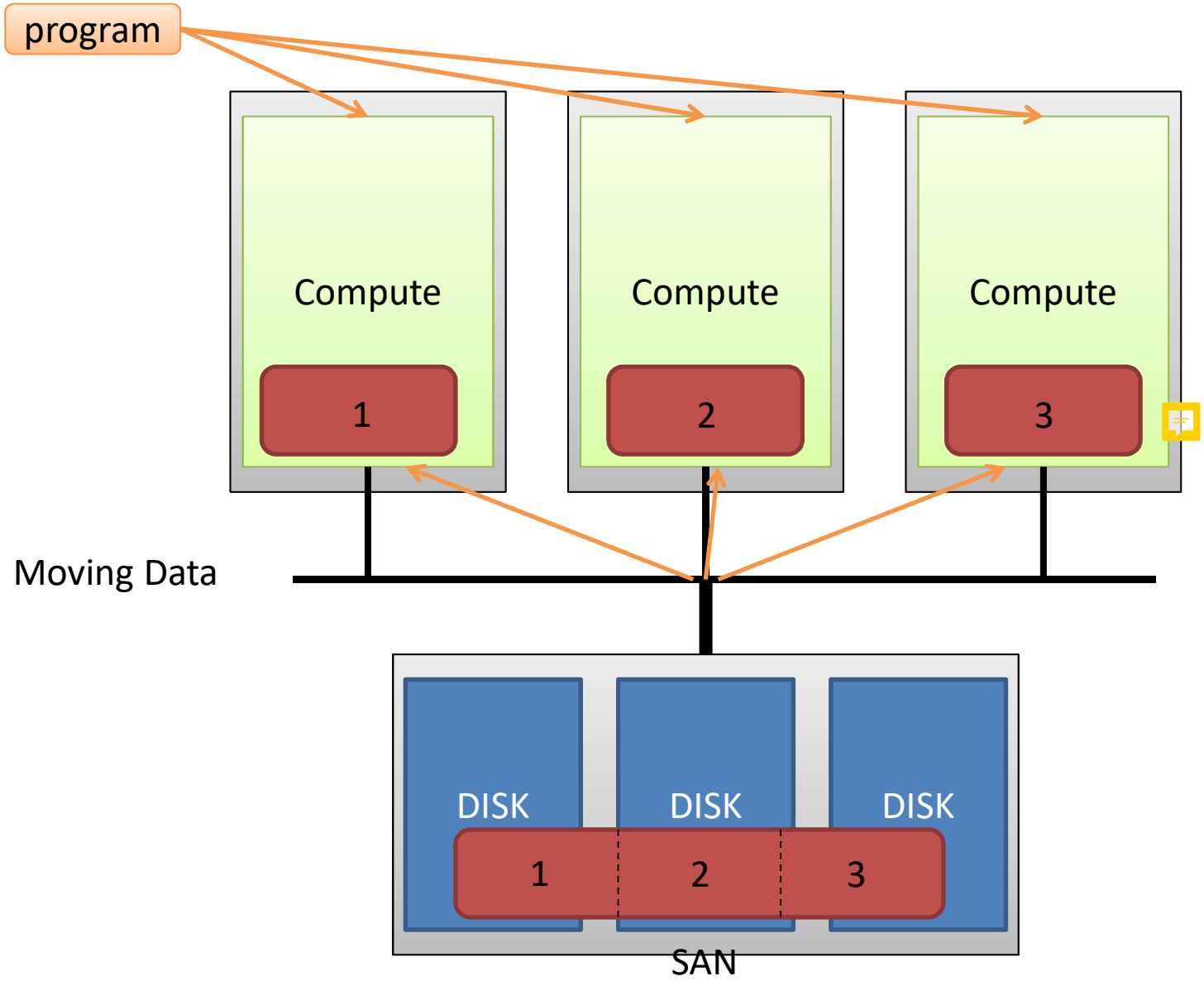

Network
CPU
CPU
CPU
CPU
MEMORY
MEMORY
MEMORY
MEMORY
DISK
DISK
DISK
DISK
Worker Node

Hadoop components

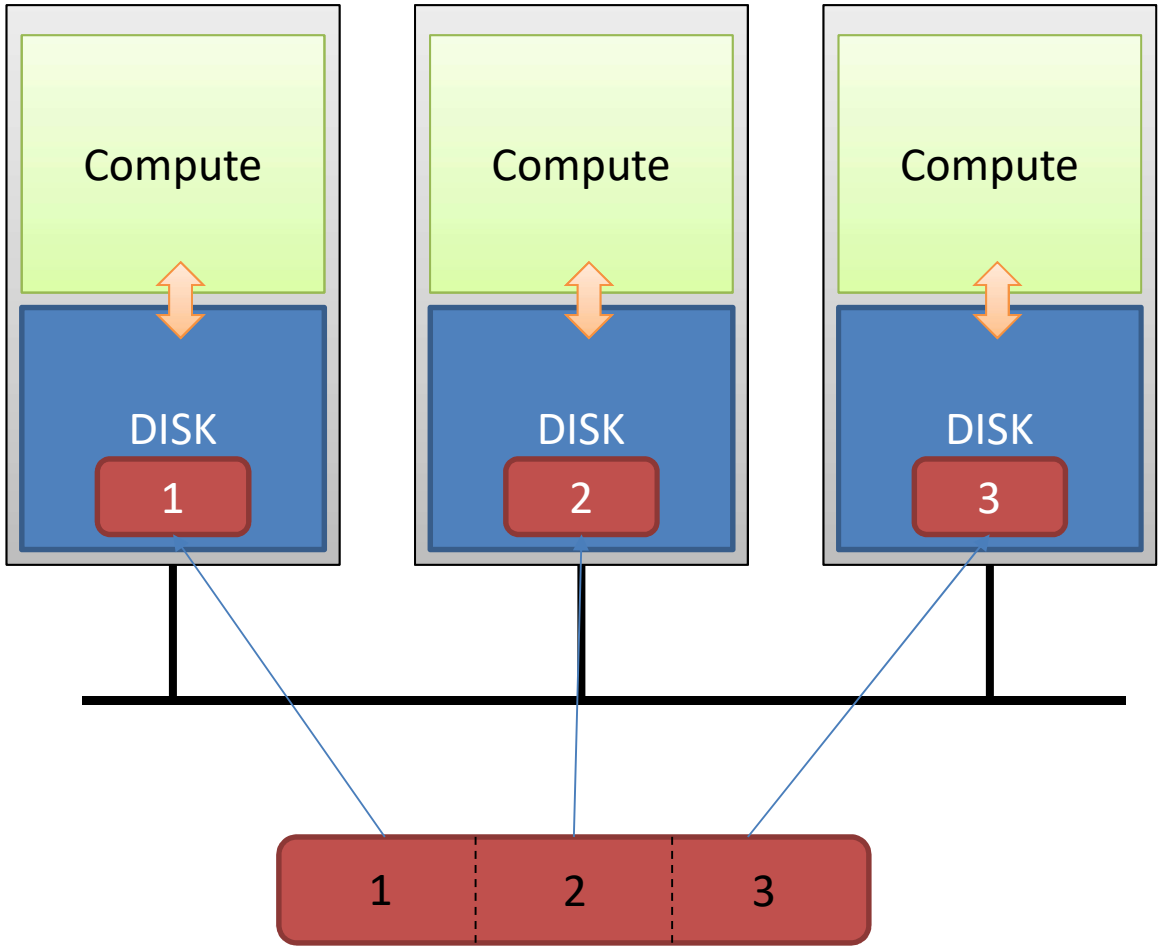
- Master Node
  - HDFS : Name Node
  - YARN : Resource Manager/Job History
- Worker Node
  - HDFS : Data Node
  - YRAN : Node Manager

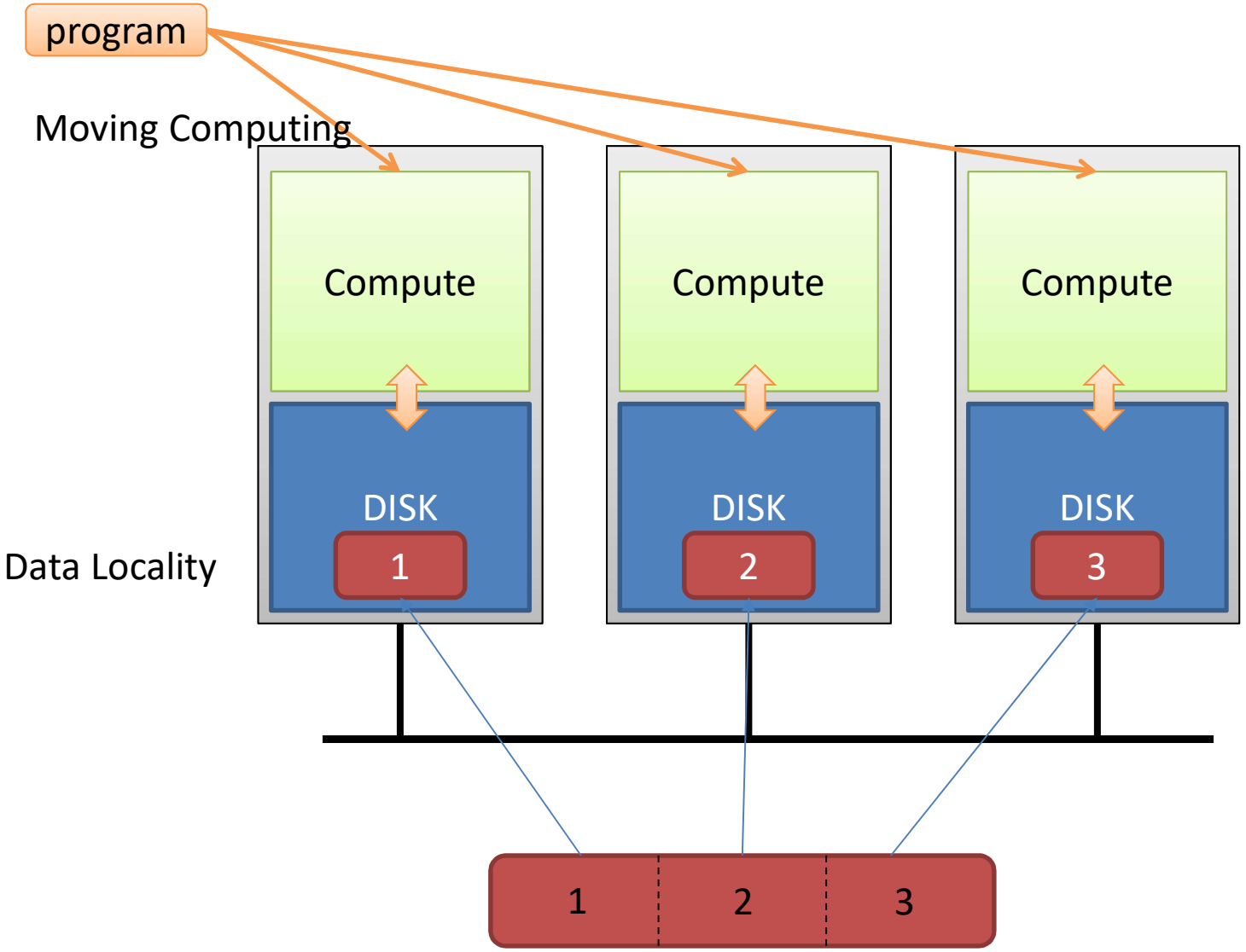

Resource Manager
JVM daemon
JVM daemon
Node Manager
Node Manager
YARN
Node Manager
Node Manager
Data Node
Data Node
HDFS
Data Node
Data Node
JVM daemon
Worker Node
Name Node
Master Node


program
Compute
Compute
Compute
DISK
DISK
DISK
SAN


program
Compute
1
Compute
2
Compute
3
Moving Data
DISK
DISK
DISK
1
2
3
SAN

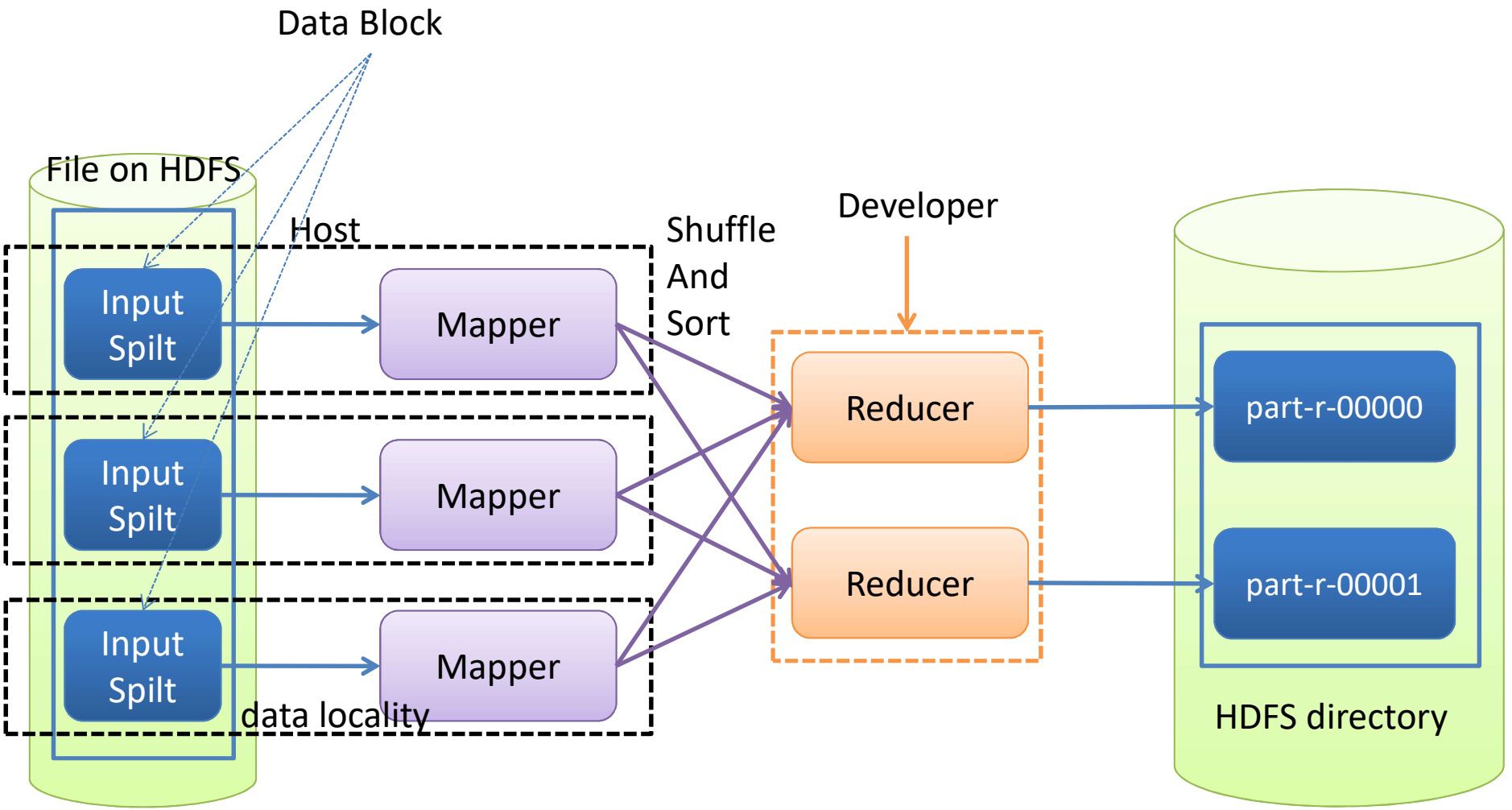
Data Locality

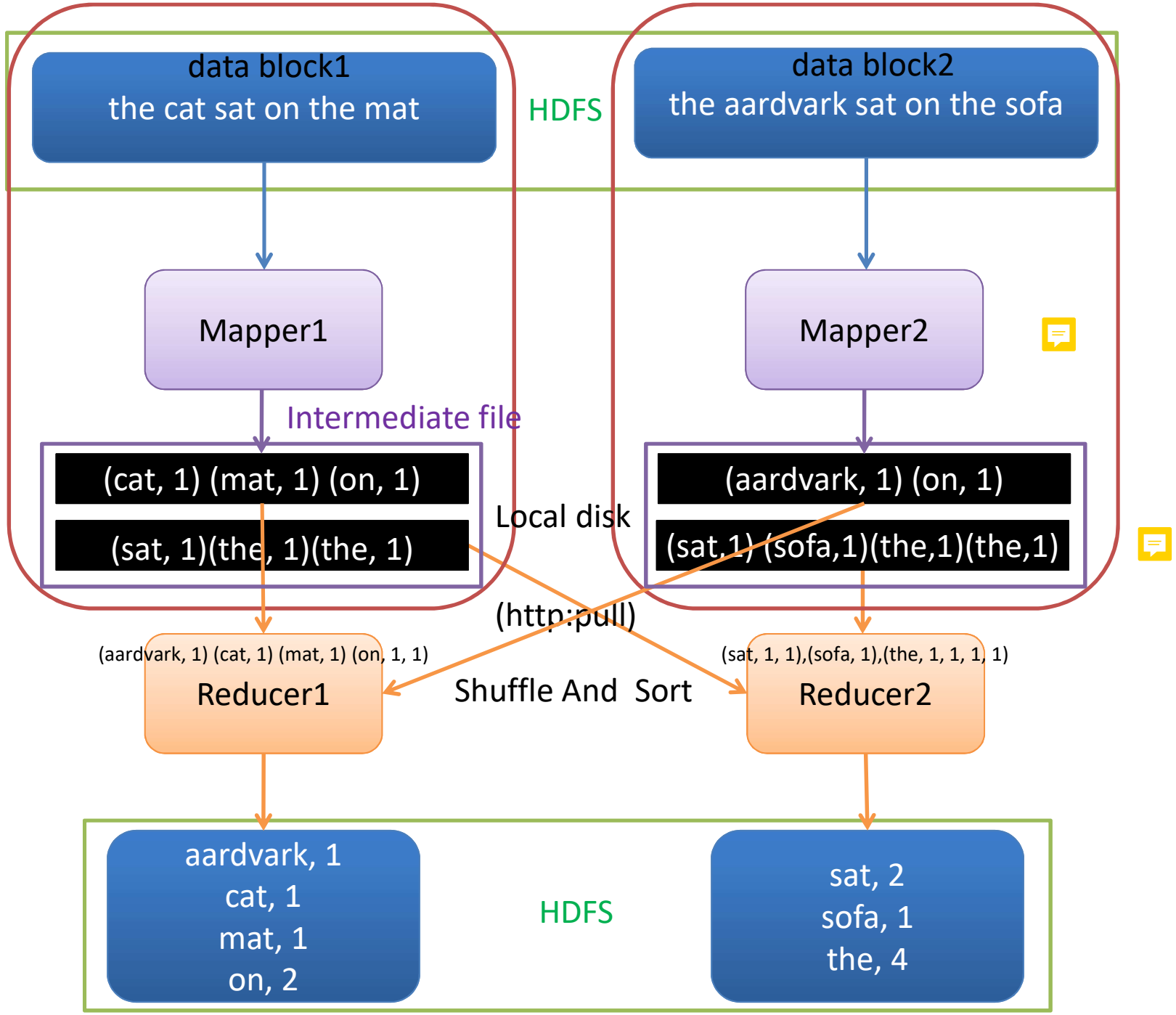



program
Moving Computing
Compute
Compute
Compute
DISK
DISK
DISK
1
2
3
Data Locality
1
2
3

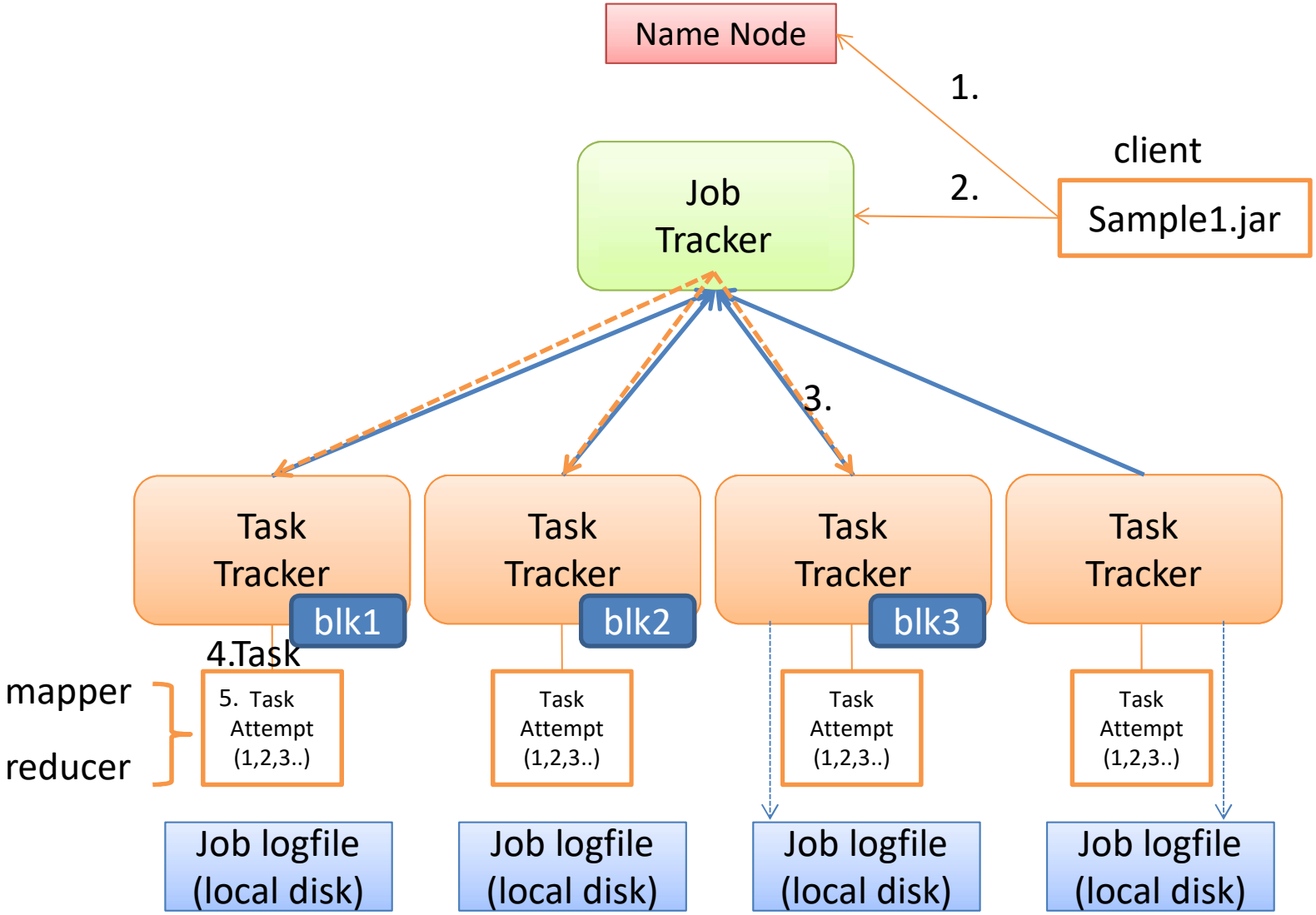
HDFS

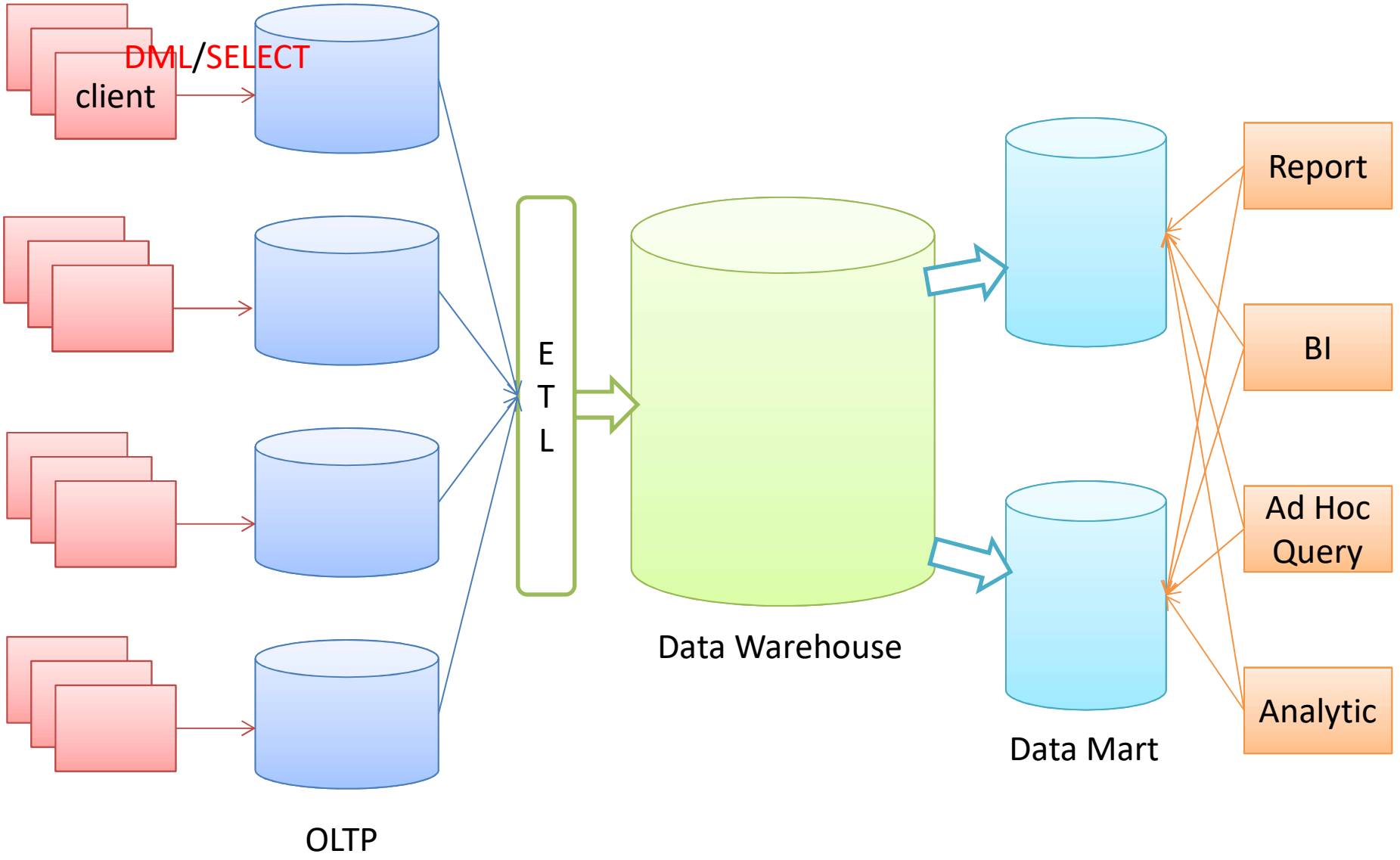
- $ hdfs dfs -put f30M.txt /user/training/
  - hdfs file : f30M.txt (1 hdfs block : 30M)
- $ hdfs dfs -put f100M.txt /user/training/
  - hdfs file : f100M.txt (1 hdfs block : 100M)
- $ hdfs dfs -put f300M.txt /user/training/
  - hdfs file : f300M.txt (3 hdfs block : 128/128/44M)


Data Block
File on HDFS
Host
Input Spilt
Input Spilt
Input Spilt
Mapper
Mapper
Mapper
data locality
Shuffle And Sort
Developer
Reducer
Reducer
part-r-00000
part-r-00001
HDFS directory

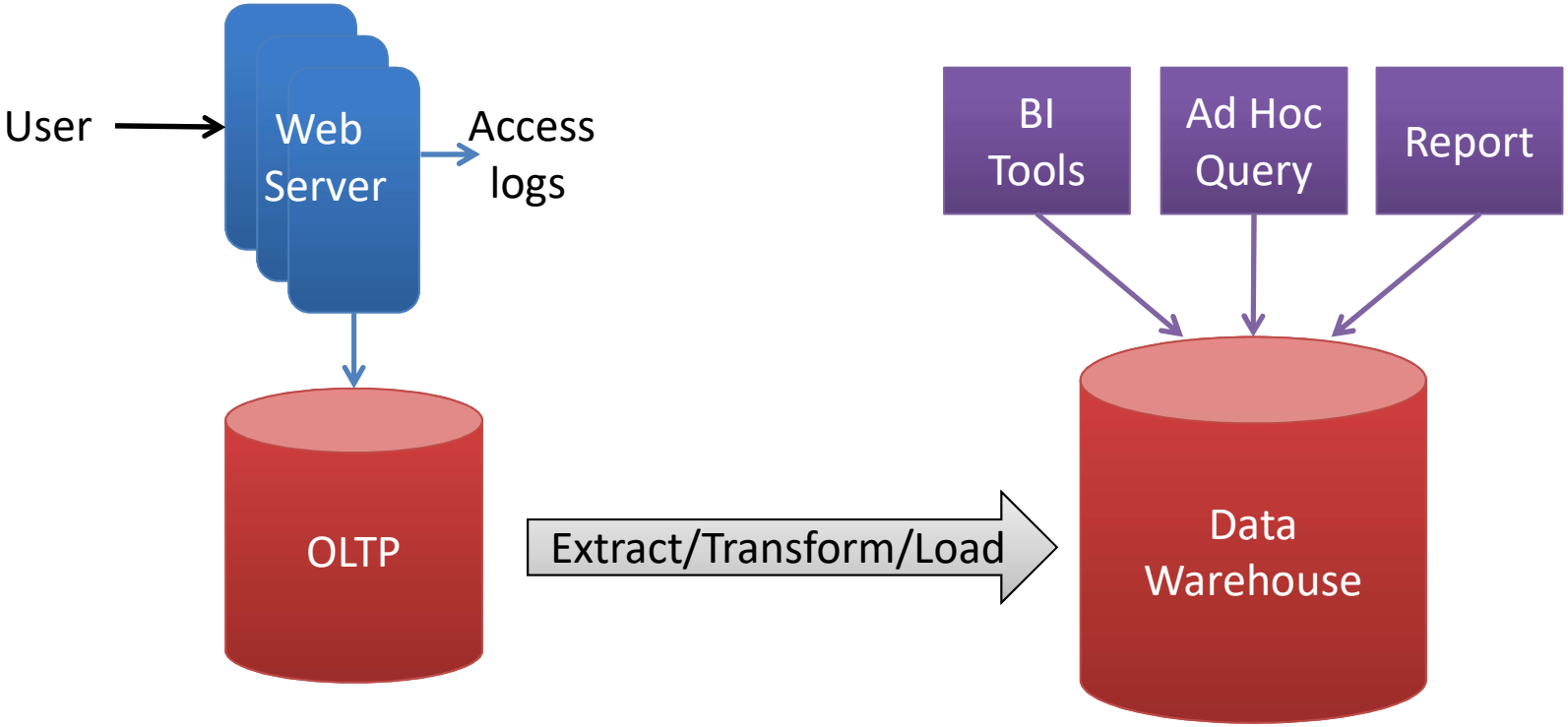

data block1
the cat sat on the mat
HDFS
data block2
the aardvark sat on the sofa
Mapper1
Mapper2
Intermediate file
(cat, 1) (mat, 1) (on, 1)
(sat, 1)(the, 1)(the, 1)
Local disk
(aardvark, 1) (on, 1)
(sat,1) (sofa,1)(the,1)(the,1)
(http:pull)
(aardvark, 1) (cat, 1) (mat, 1) (on, 1, 1)
Reducer1
Shuffle And Sort
(sat, 1, 1),(sofa, 1),(the, 1, 1, 1, 1)
Reducer2
aardvark, 1
cat, 1
mat, 1
on, 2
HDFS
sat, 2
sofa, 1
the, 4

Map Reduce v1

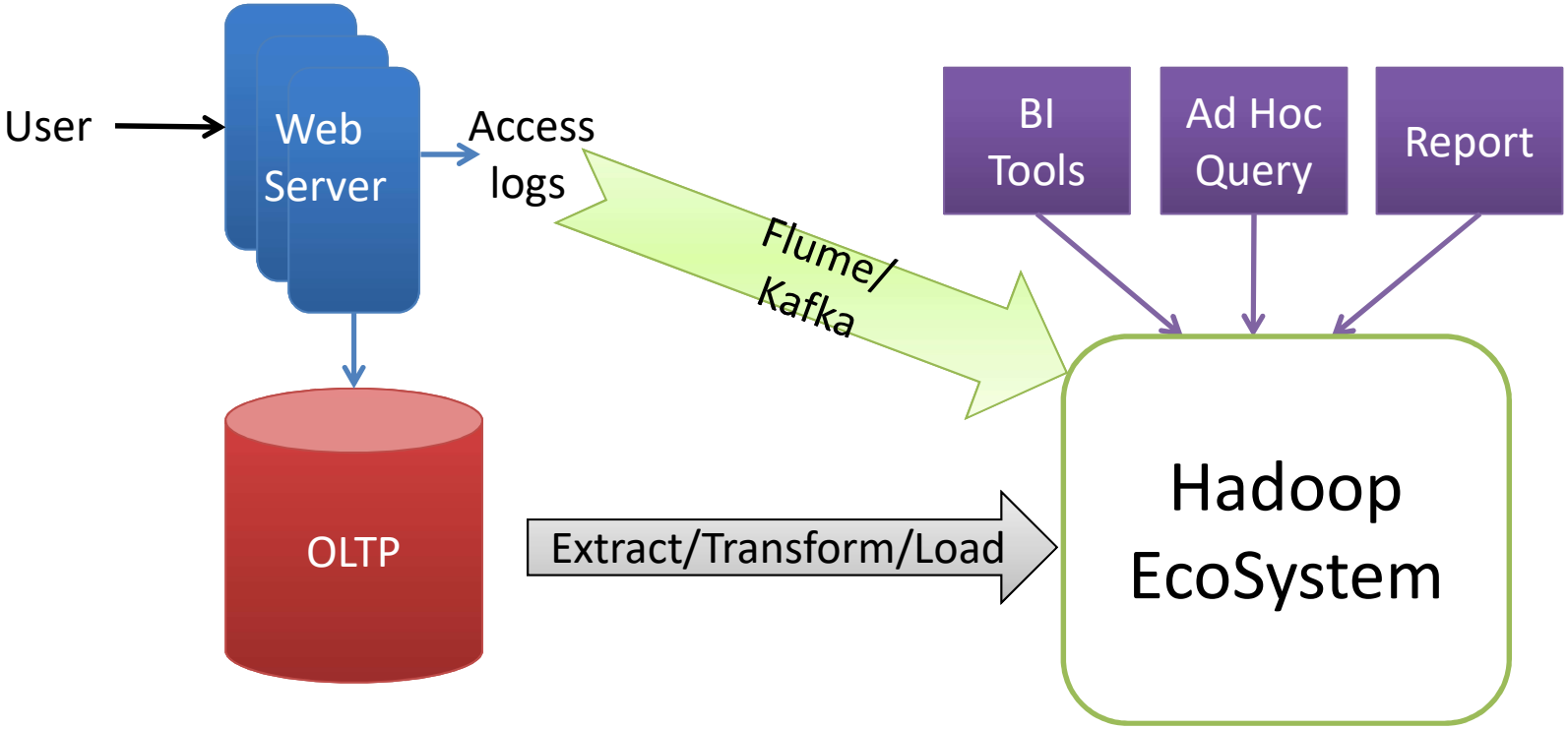



client
DML/SELECT
E
T
L
Data Warehouse
Report
BI
Ad Hoc
Query
Analytic
Data Mart
OLTP

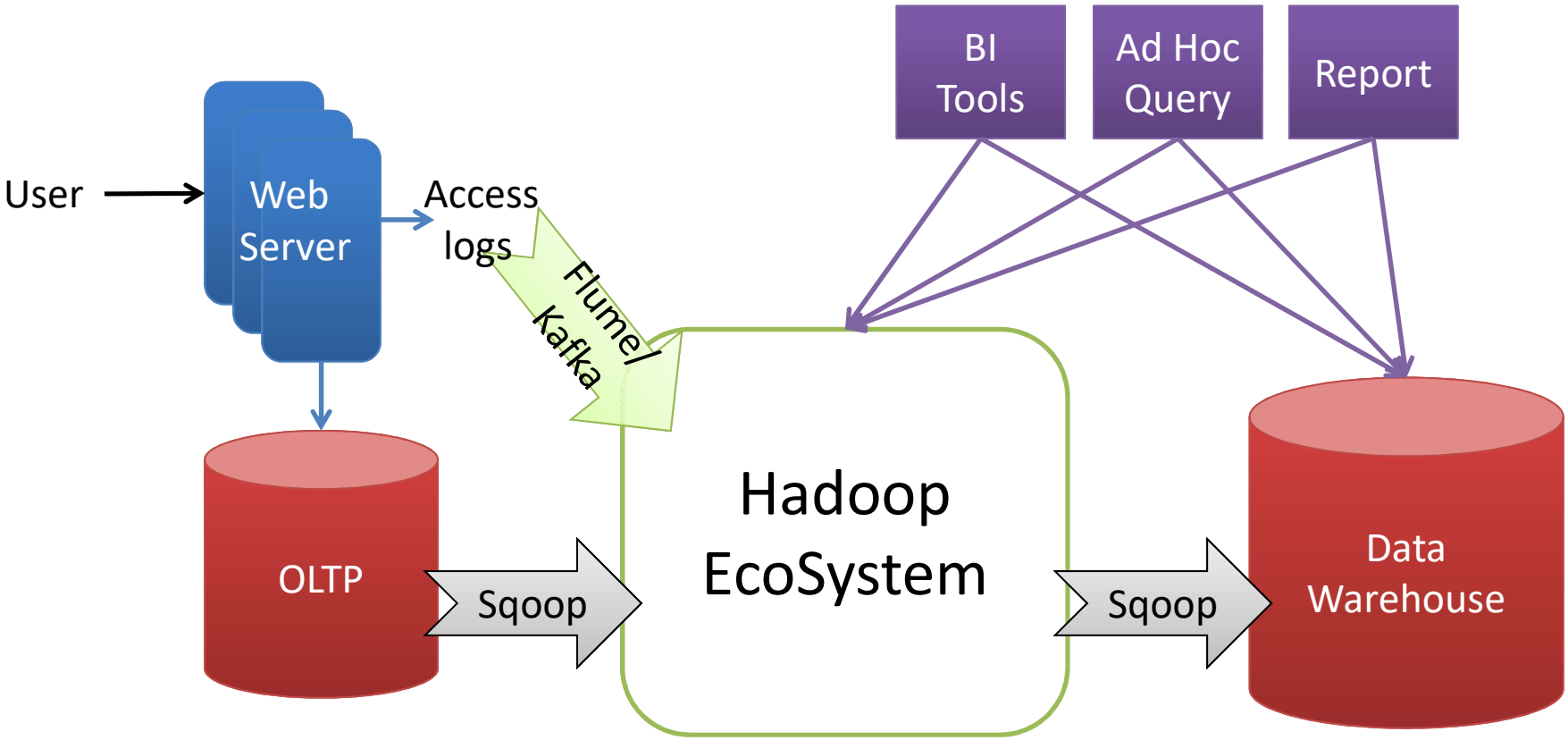
Data Warehouse based on RDBMS



Data Warehouse based on Hadoop



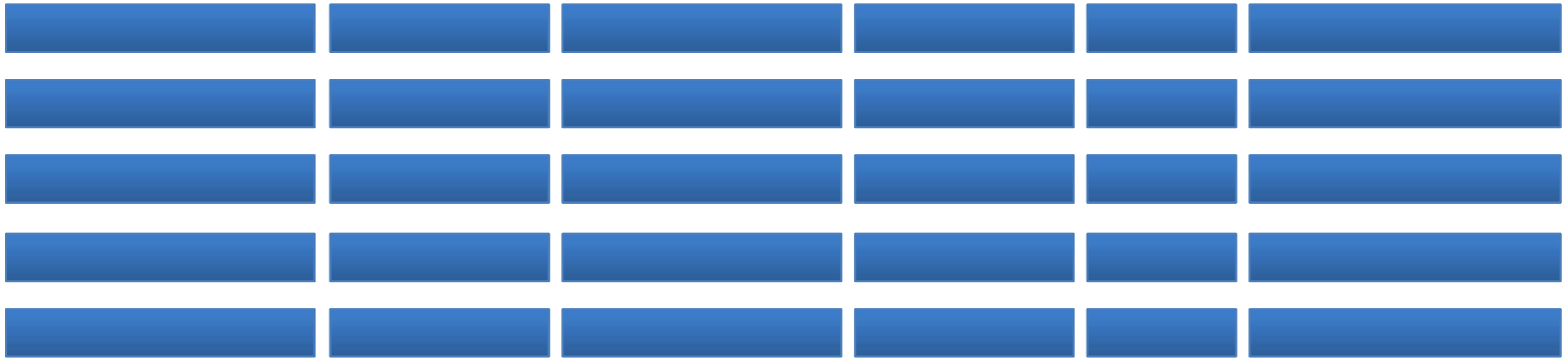
Integrated RDBMS with Hadoop



SQL on Hadoop

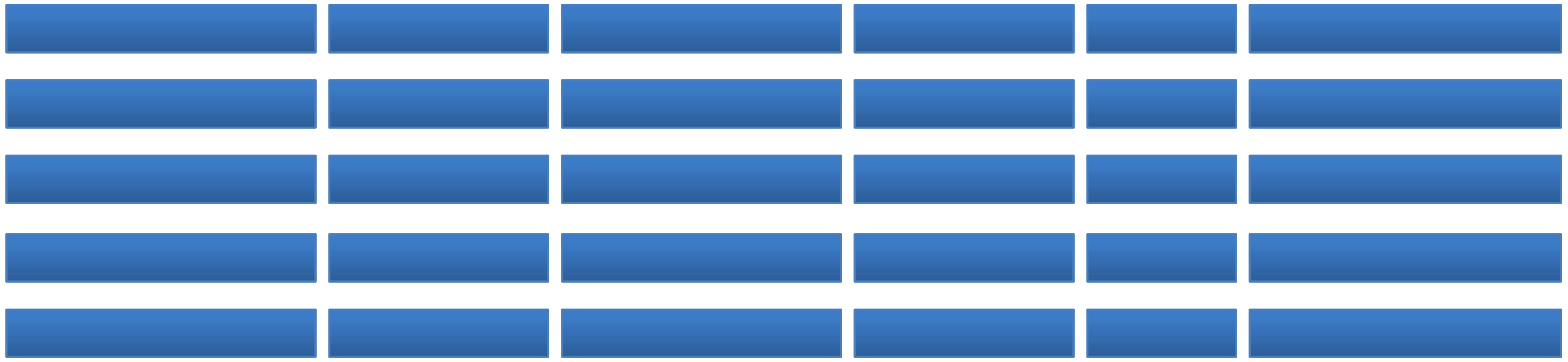
- Batch SQL
  - Hive(Facebook)
    - Based on MapReduce
    - Based on Tez(Hortonworks)
- Interactive SQL
  - Spark SQL(Hive based on Spark)
  - Impala(Cloudera)(Google Dremel)
  - Apache Drill(Google Dremel)
  - Presto(Facebook)
  - Pinot(LinkedIn:OLAP)
  - Kylin(Ebay China:OLAP)
- Operational SQL
  - Apache Phoenix(SQL on Hbase)

Schema on Write

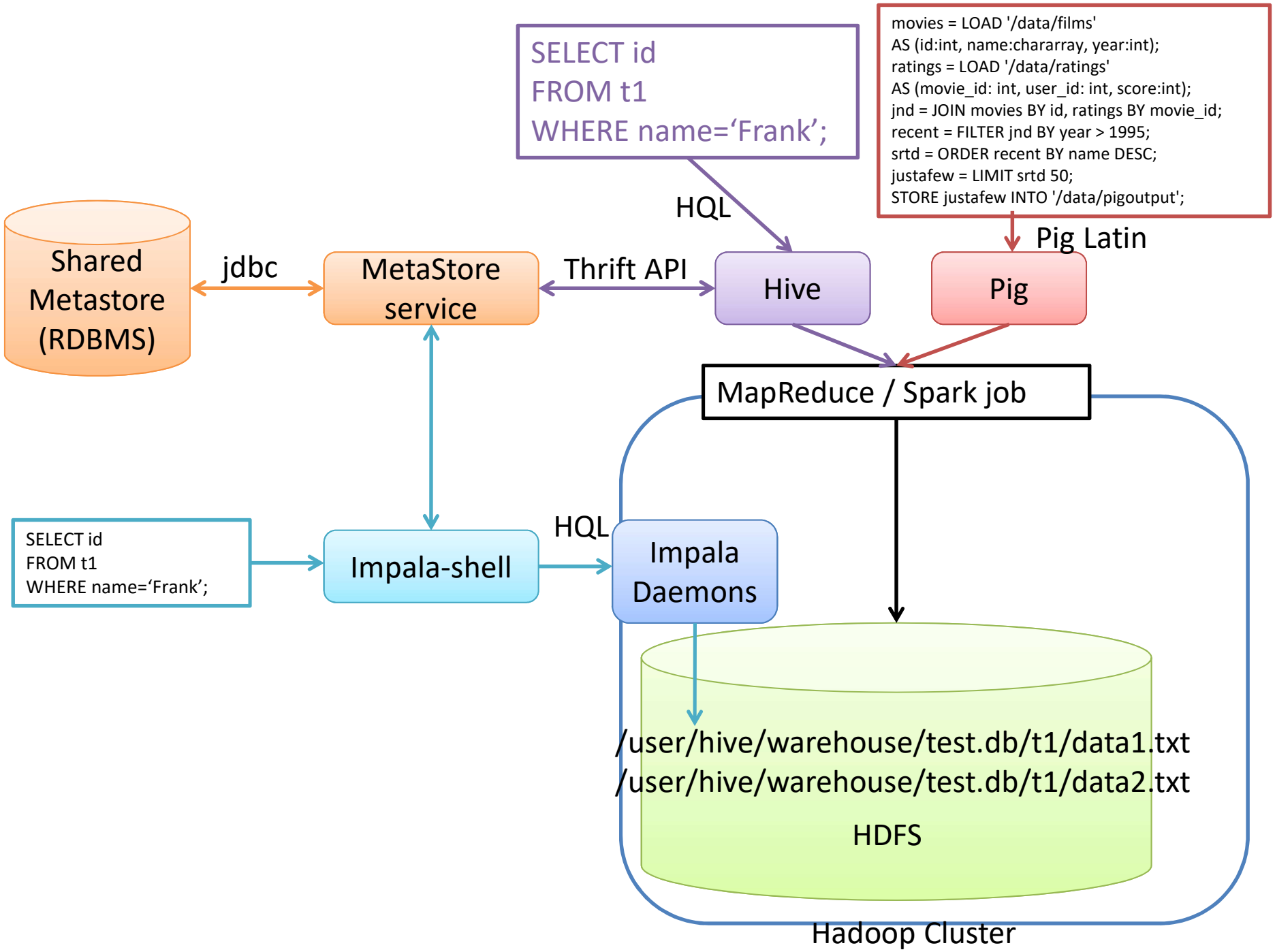


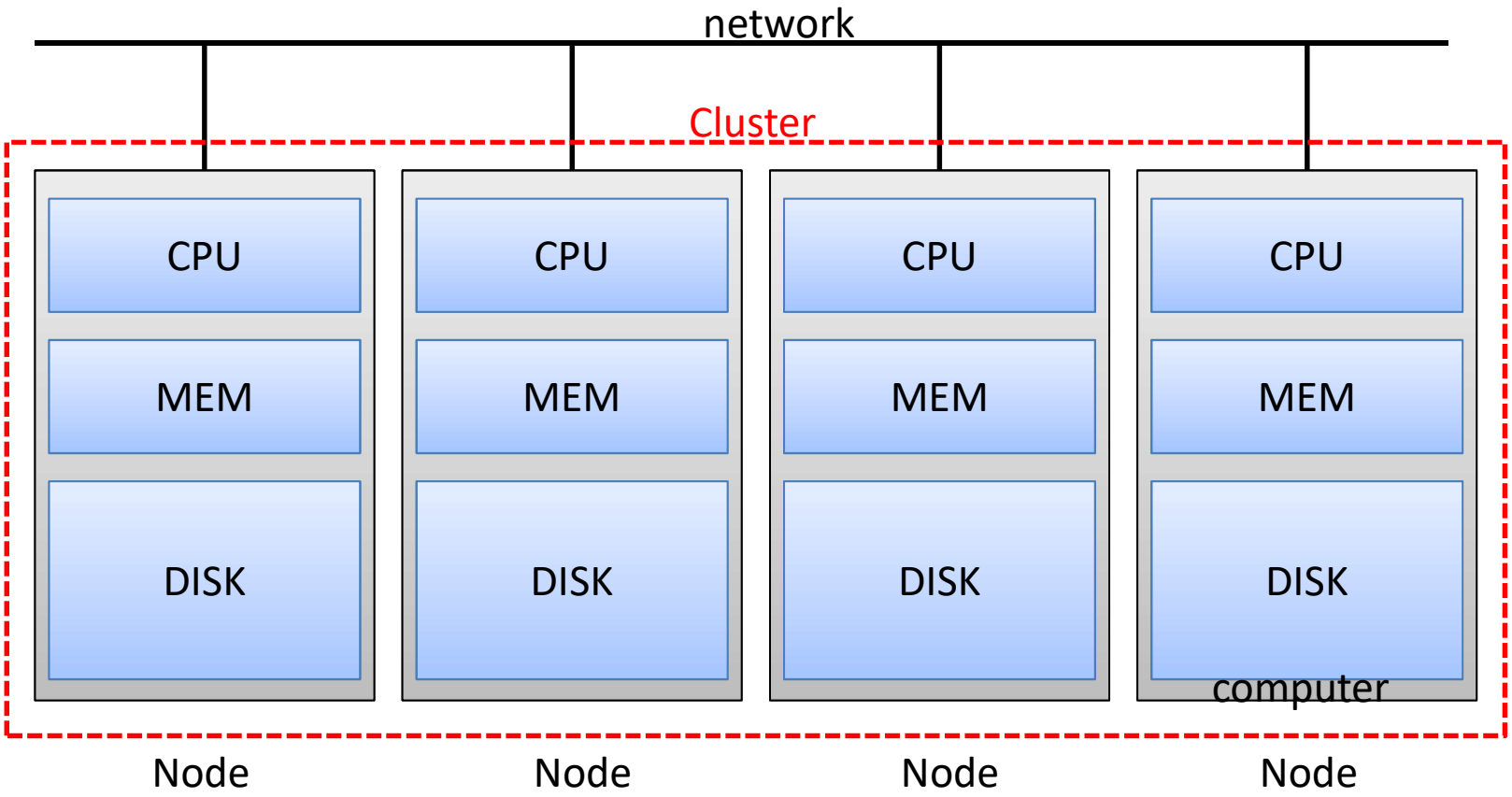
Empid	Ename	Salary	Hire_date

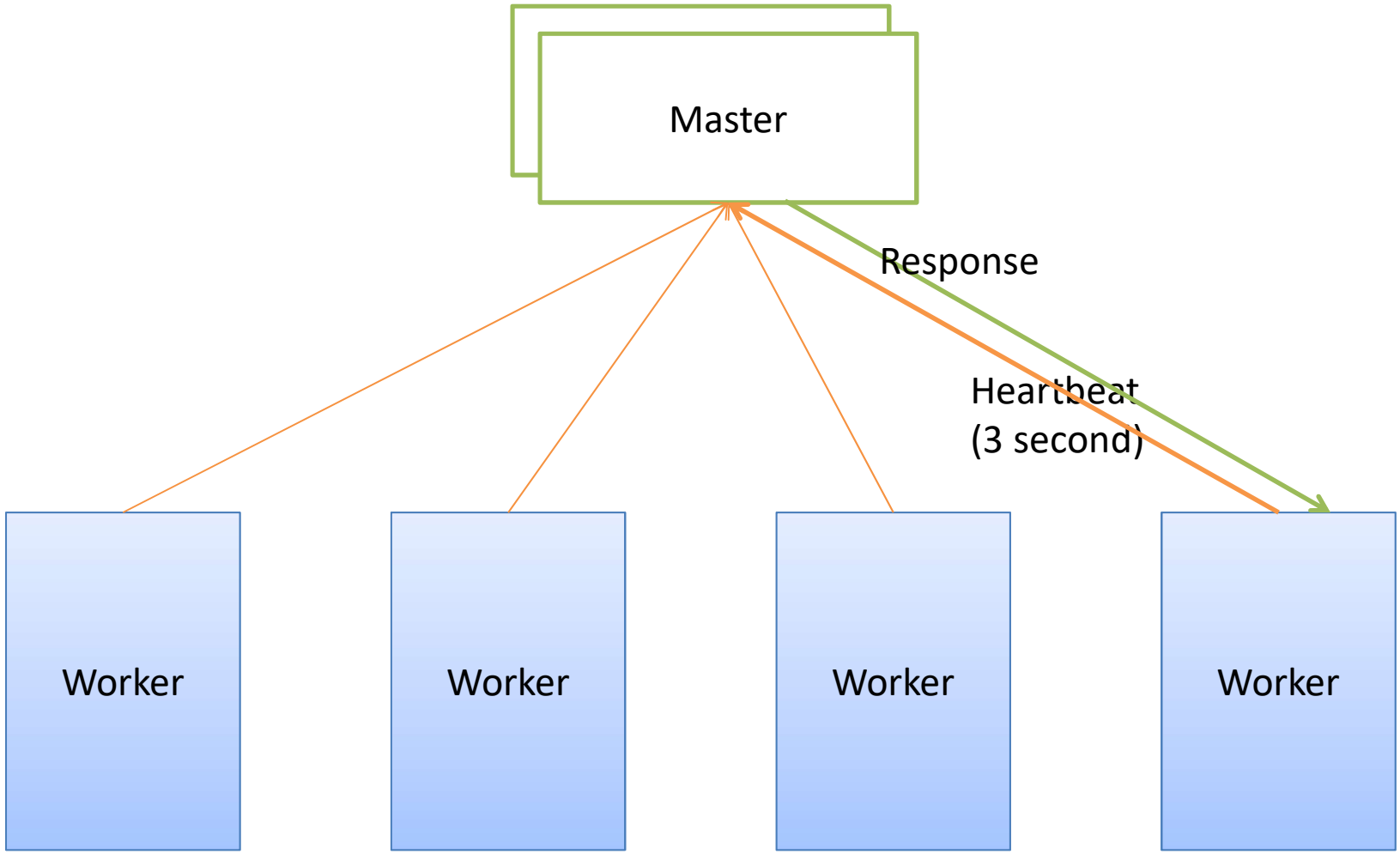
Schema on Read

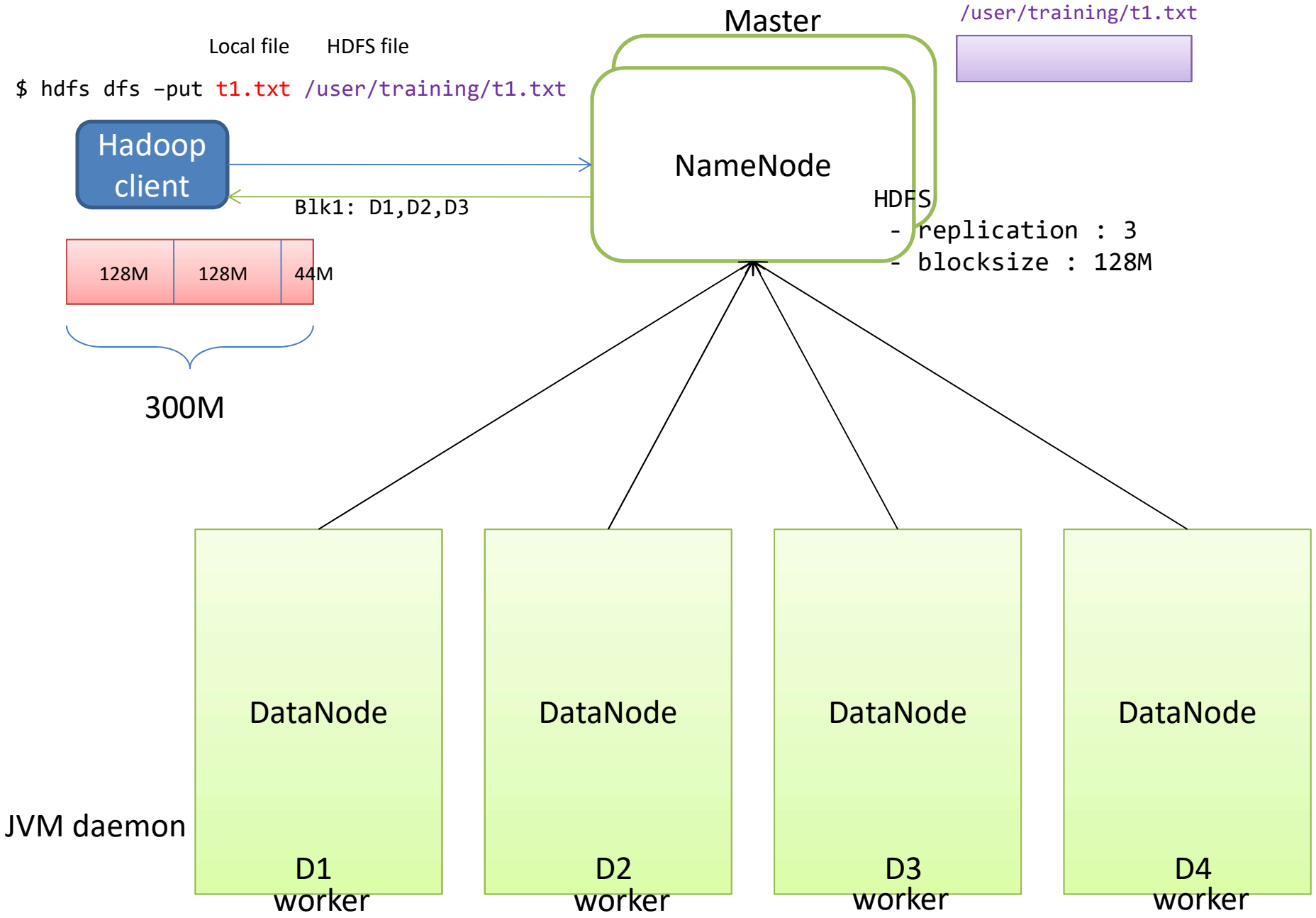


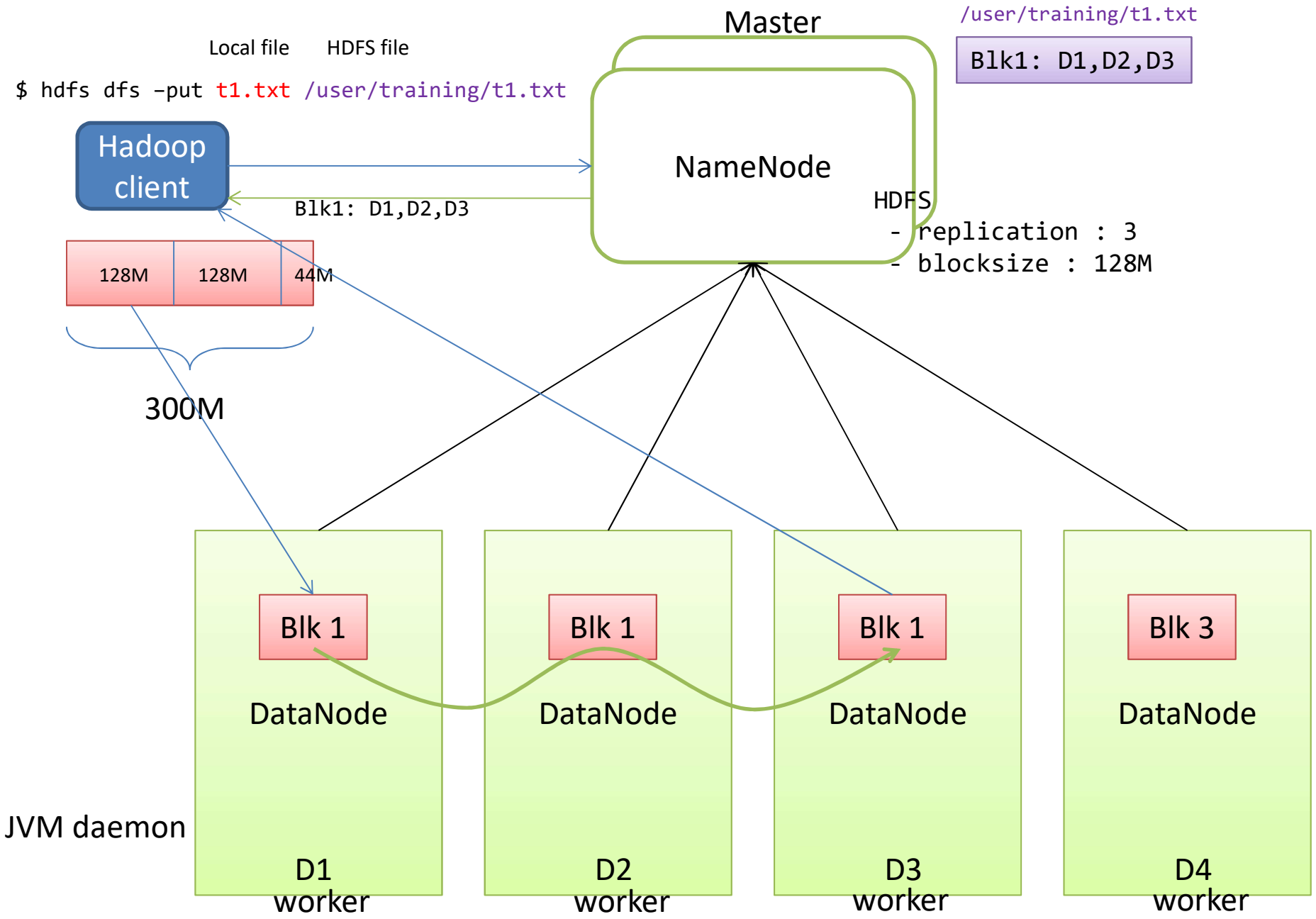
Empid	Ename	Job	Salary	Dept	Hire_date

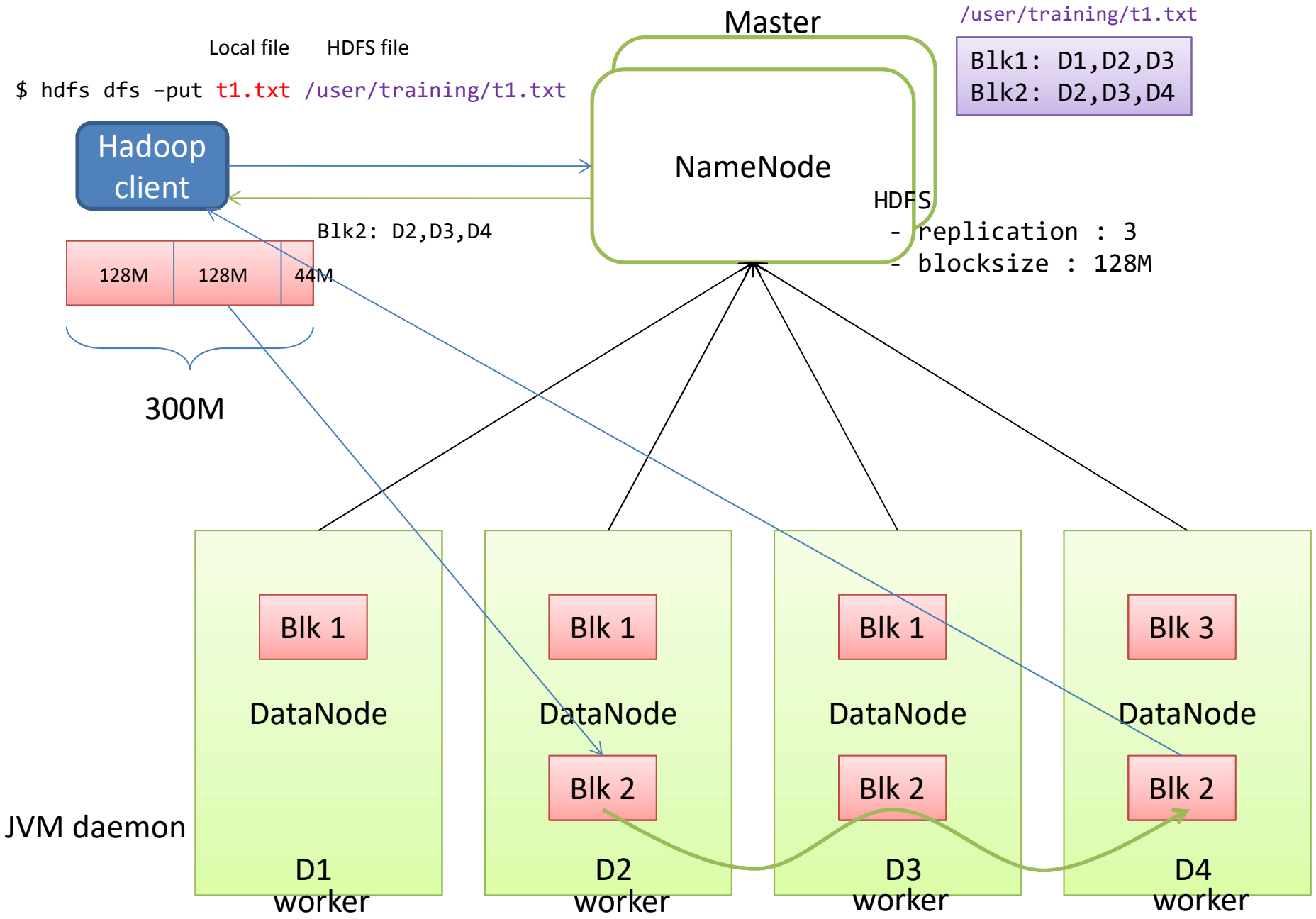

SELECT id
FROM t1
WHERE name='Frank';
movies = LOAD '/data/films'
AS (id:int, name:chararray, year:int);
ratings = LOAD '/data/ratings'
AS (movie_id: int, user_id: int, score:int);
jnd = JOIN movies BY id, ratings BY movie_id;
recent = FILTER jnd BY year > 1995;
srtd = ORDER recent BY name DESC;
justafew = LIMIT srtd 50;
STORE justafew INTO '/data/pigoutput';
HQL
Pig Latin
Shared Metastore (RDBMS)
jdbc
MetaStore service
Thrift API
Hive
Pig
MapReduce / Spark job
SELECT id
FROM t1
WHERE name='Frank';
Impala-shell
HQL
Impala Daemons
/user/hive/warehouse/test.db/t1/data1.txt
/user/hive/warehouse/test.db/t1/data2.txt
HDFS
Hadoop Cluster

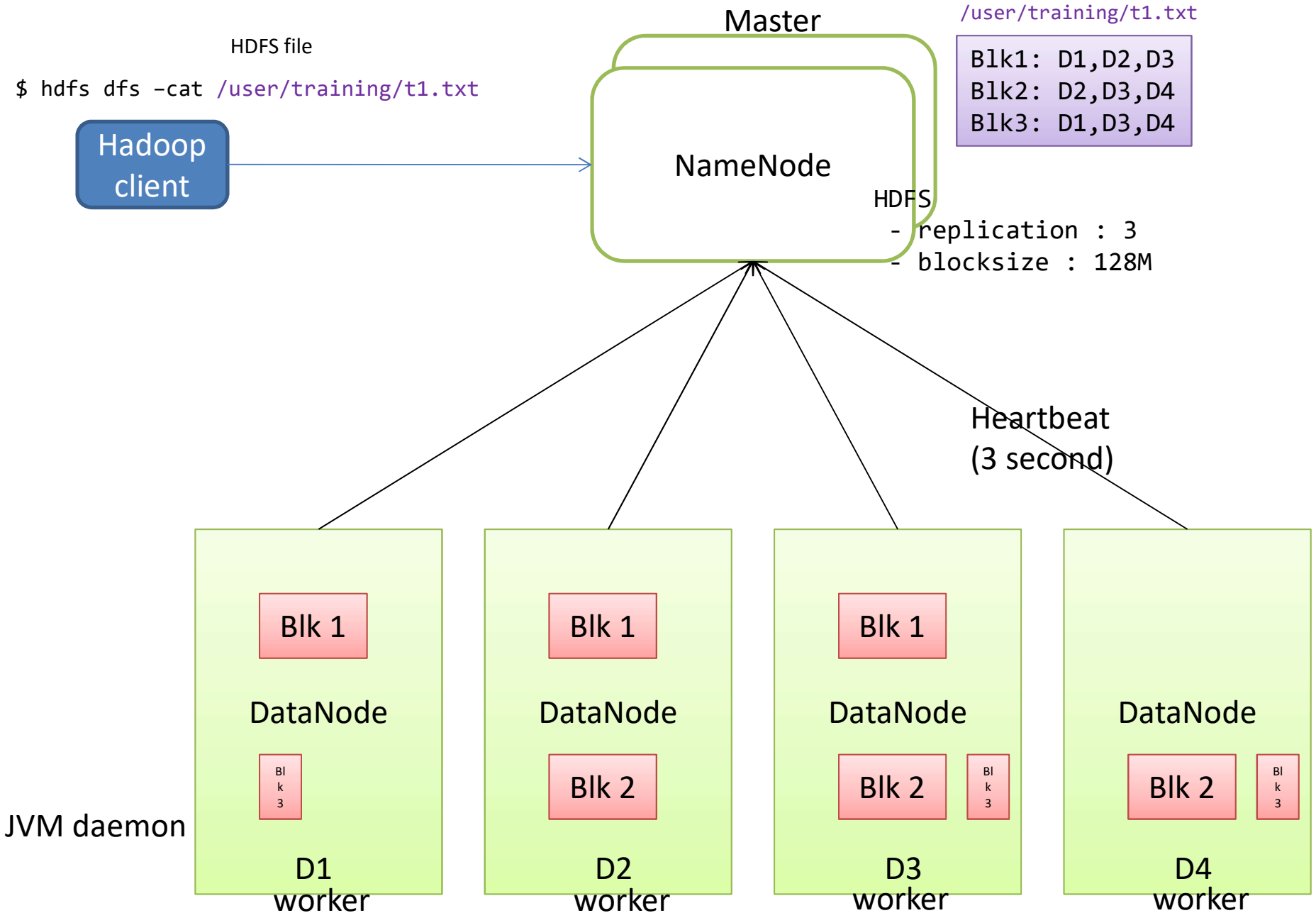

network
Cluster
CPU
MEM
DISK
CPU
MEM
DISK
CPU
MEM
DISK
CPU
MEM
DISK
computer
Node
Node
Node
Node

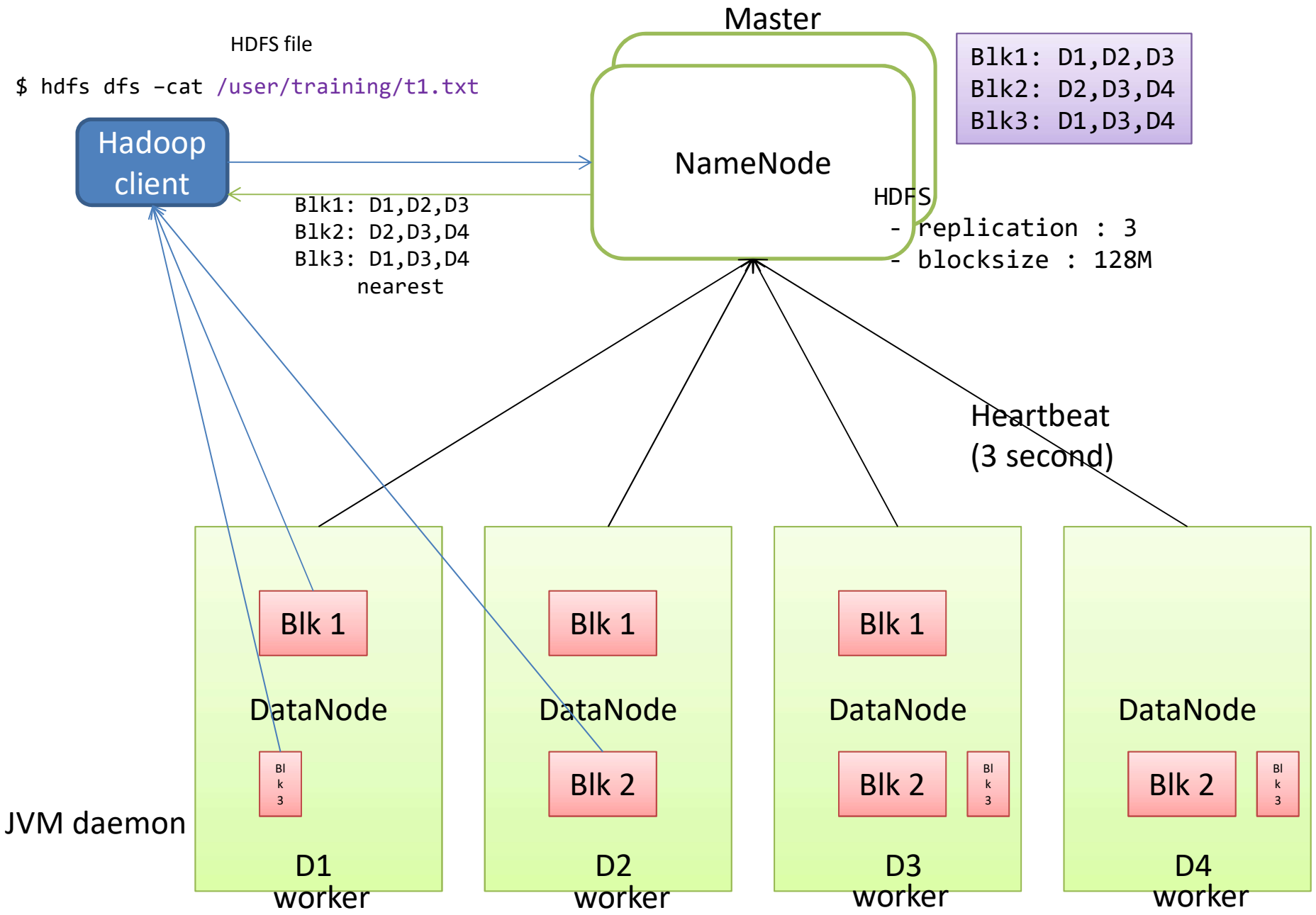

Master
Response
Heartbeat
(3 second)
Worker
Worker
Worker
Worker

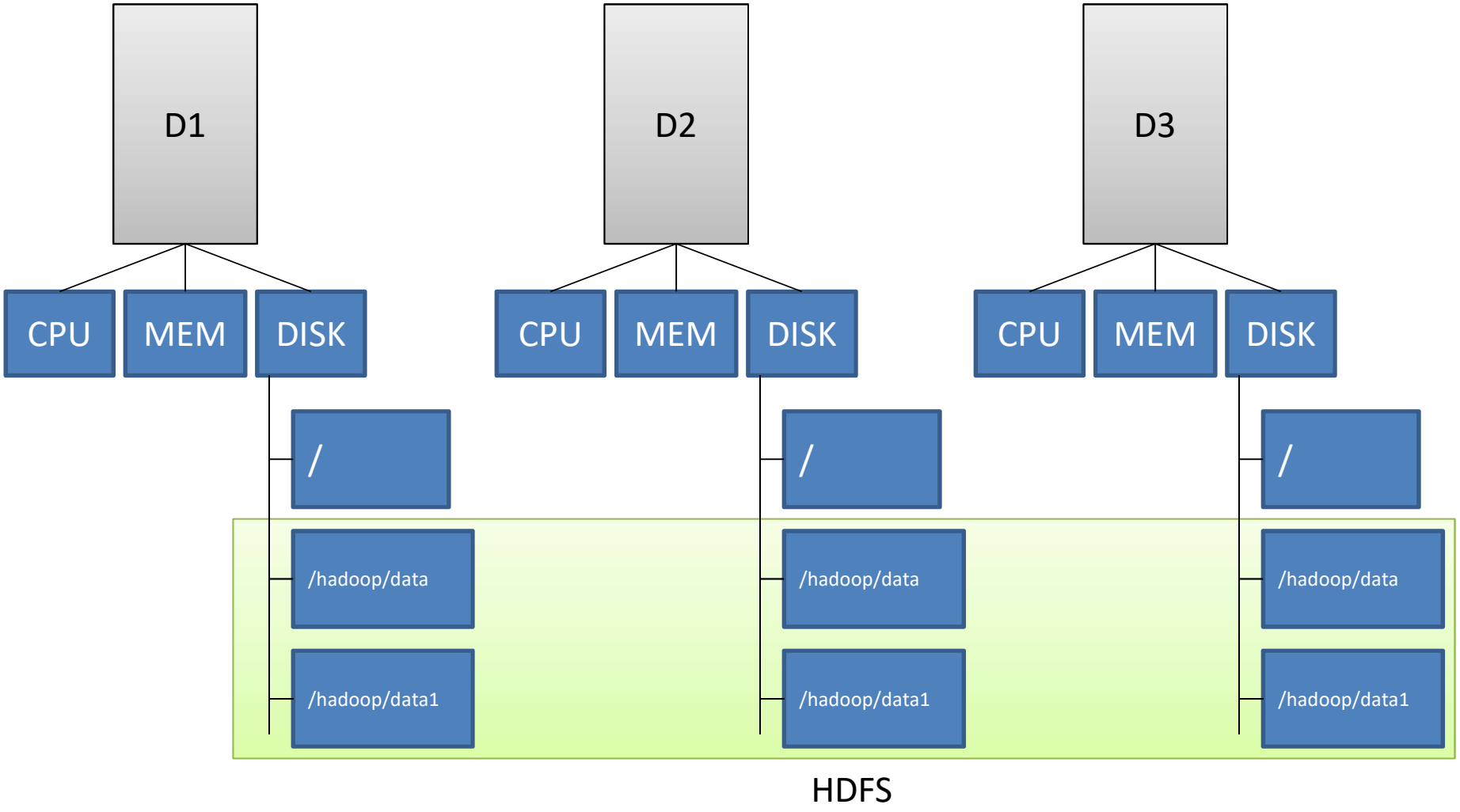

Master
/user/training/t1.txt
Local file
HDFS file
$ hdfs dfs –put t1.txt /user/training/t1.txt
Hadoop client
NameNode
Blk1: D1,D2,D3
HDFS
- replication : 3
- blocksize : 128M
128M
128M
44M
300M
DataNode
DataNode
DataNode
DataNode
JVM daemon
D1
worker
D2
worker
D3
worker
D4
worker

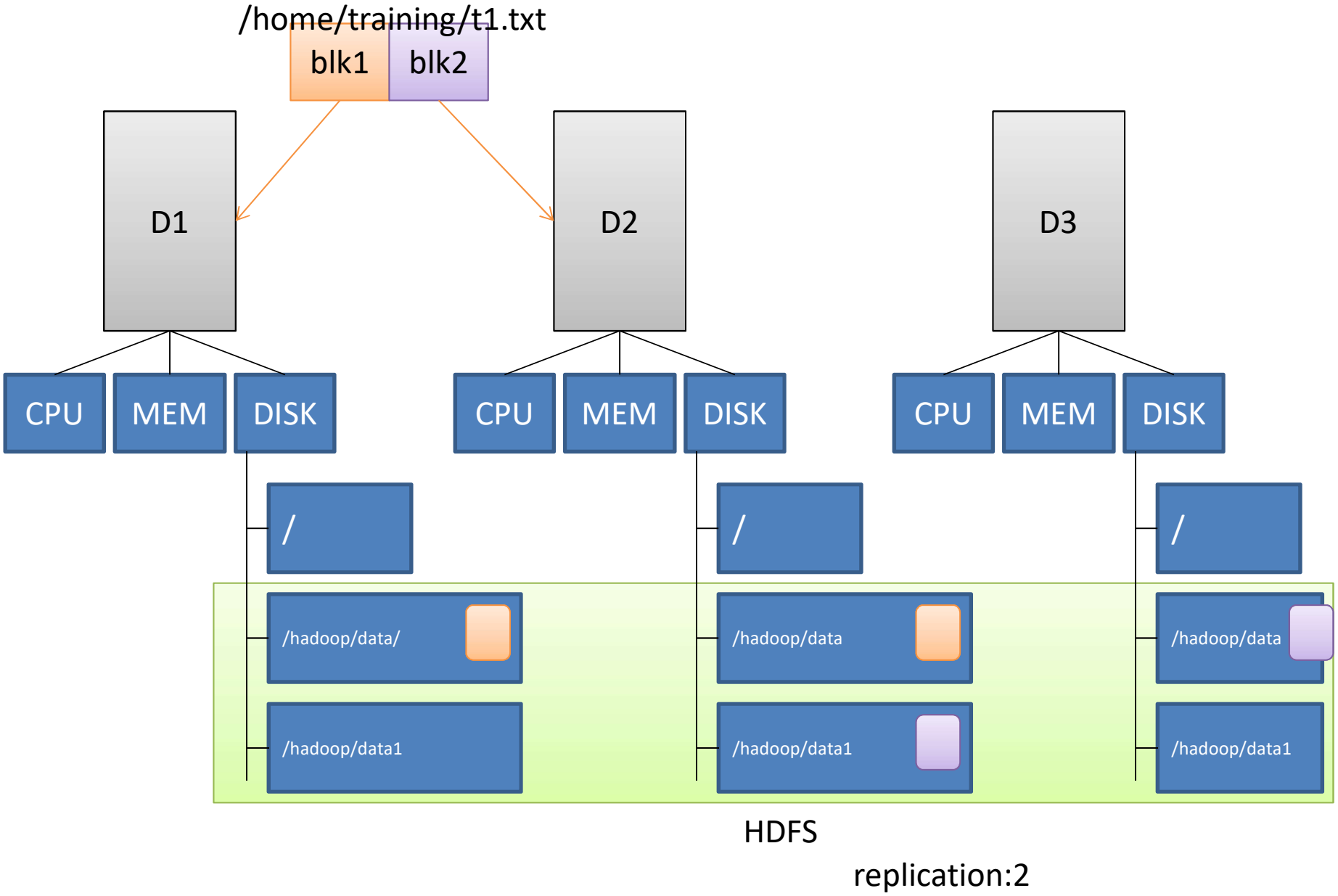

Master
/user/training/t1.txt
Blk1: D1,D2,D3
Local file
HDFS file
$ hdfs dfs –put t1.txt /user/training/t1.txt
Hadoop client
NameNode
Blk1: D1,D2,D3
HDFS
- replication : 3
- blocksize : 128M
128M
128M
44M
300M
Blk 1
Blk 1
Blk 1
Blk 3
DataNode
DataNode
DataNode
DataNode
JVM daemon
D1 worker
D2 worker
D3 worker
D4 worker


Master
/user/training/t1.txt
Blk1: D1,D2,D3
Blk2: D2,D3,D4
Local file
HDFS file
$ hdfs dfs –put t1.txt /user/training/t1.txt
Hadoop client
NameNode
HDFS
- replication : 3
- blocksize : 128M
Blk2: D2,D3,D4
128M
128M
44M
300M
Blk 1
DataNode
Blk 1
DataNode
Blk 1
DataNode
Blk 3
DataNode
Blk 2
Blk 2
Blk 2
JVM daemon
D1
worker
D2
worker
D3
worker
D4
worker


Master
/user/training/t1.txt
Blk1: D1,D2,D3
Blk2: D2,D3,D4
Blk3: D1,D3,D4
HDFS file
$ hdfs dfs –cat /user/training/t1.txt
Hadoop client
NameNode
HDFS
- replication : 3
- blocksize : 128M
Heartbeat
(3 second)
Blk 1
DataNode
Blk 2
Blk 3
JVM daemon
D1
worker
D2
worker
D3
worker
D4
worker


Master
HDFS file
$ hdfs dfs –cat /user/training/t1.txt
Hadoop client
NameNode
Blk1: D1,D2,D3
Blk2: D2,D3,D4
Blk3: D1,D3,D4
HDFS
- replication : 3
- blocksize : 128M
Blk1: D1,D2,D3
Blk2: D2,D3,D4
Blk3: D1,D3,D4
nearest
Heartbeat
(3 second)
Blk 1
DataNode
Blk 2
Blk 3
JVM daemon
D1
worker
D2
worker
D3
worker
D4
worker


D1
D2
D3
CPU
MEM
DISK
CPU
MEM
DISK
CPU
MEM
DISK
/
/
/
/hadoop/data
/hadoop/data1
/hadoop/data
/hadoop/data1
/hadoop/data
/hadoop/data1
HDFS


/home/training/t1.txt
blk1
blk2
D1
D2
D3
CPU
MEM
DISK
CPU
MEM
DISK
CPU
MEM
DISK
/
/
/
/hadoop/data/
/hadoop/data
/hadoop/data
/hadoop/data1
/hadoop/data1
/hadoop/data1
HDFS
replication:2

DataNode

- Store block
- Block Report after DataNode startup
- Heart Beat(3 seconds) to NameNode

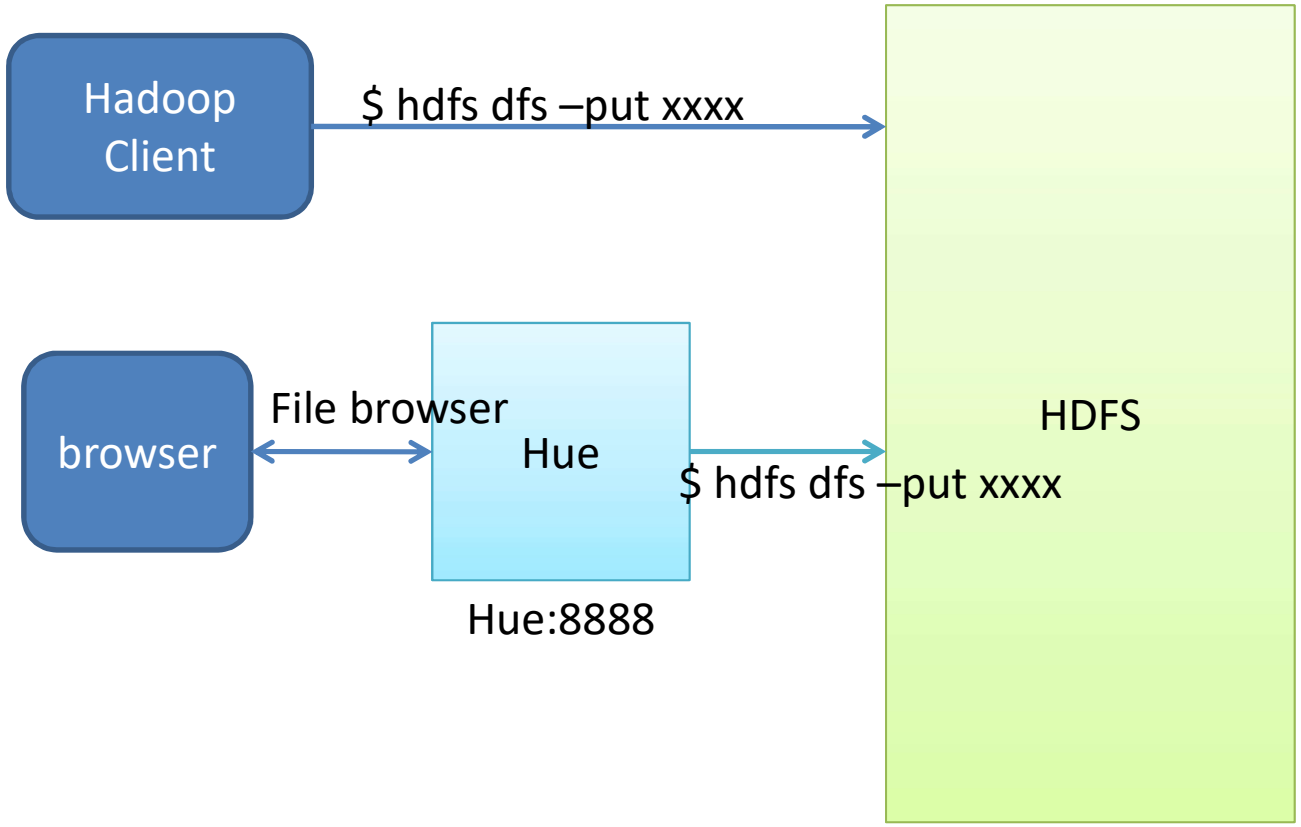
```
[cloudera@quickstart ~]$ cat /etc/hadoop/conf/hdfs-site.xml|grep fs.datanode.data.dir -B 1 -A 2
  <property>
     <name>dfs.datanode.data.dir</name>
     <value>/var/lib/hadoop-hdfs/cache/${user.name}/dfs/data</value>
  </property>
[cloudera@quickstart ~]$ sudo ls -l /var/lib/hadoop-hdfs/cache/hdfs/dfs/data/current/
total 8
drwx------ 4 hdfs hadoop 4096 Oct  1 06:23 BP-989008105-127.0.0.1-1433846136903
-rw-r--r-- 1 hdfs hadoop  229 Oct  1 06:23 VERSION
[cloudera@quickstart ~]$ sudo ls -l /var/lib/hadoop-hdfs/cache/hdfs/dfs/data/current/BP-989008105-127
.0.0.1-1433846136903/current/finalized/subdir0/subdir0 |more
total 220432
-rw-r--r-- 1 hdfs hadoop    11175 Jun  9 03:39 blk_1073741825
-rw-r--r-- 1 hdfs hadoop       95 Jun  9 03:39 blk_1073741825_1001.meta
-rw-r--r-- 1 hdfs hadoop    48741 Jun  9 03:39 blk_1073741826
-rw-r--r-- 1 hdfs hadoop      391 Jun  9 03:39 blk_1073741826_1002.meta
-rw-r--r-- 1 hdfs hadoop    16046 Jun  9 03:39 blk_1073741827
-rw-r--r-- 1 hdfs hadoop      135 Jun  9 03:39 blk_1073741827_1003.meta
-rw-r--r-- 1 hdfs hadoop  6377448 Jun  9 03:39 blk_1073741828
-rw-r--r-- 1 hdfs hadoop    49831 Jun  9 03:39 blk_1073741828_1004.meta
-rw-r--r-- 1 hdfs hadoop    12312 Jun  9 03:39 blk_1073741829
-rw-r--r-- 1 hdfs hadoop      107 Jun  9 03:39 blk_1073741829_1005.meta
```

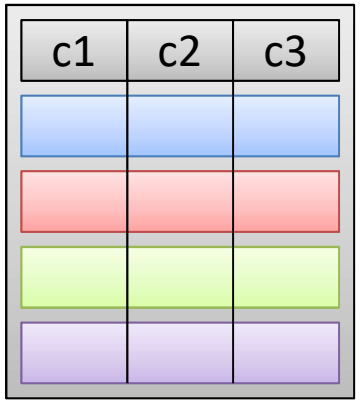
Block Replication policy

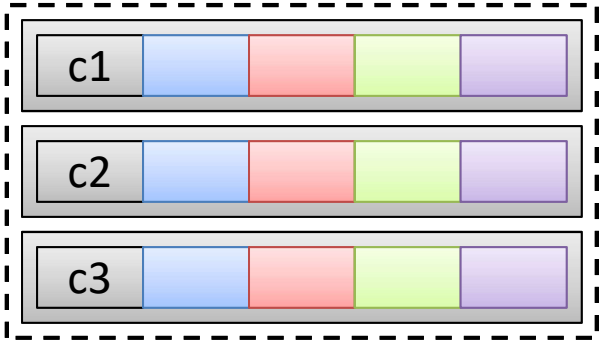
- First Replication
  - 放在上傳文件的DN。如果不是上傳者不是DN，則隨機選擇Disk不太滿且CPU不太忙的DN。
- Secondary Replication
  - 放在與第一個Replication不同的機架上的DN。
- Third Replcation
  - 放在與第二個Replication相同機架的不同DN。

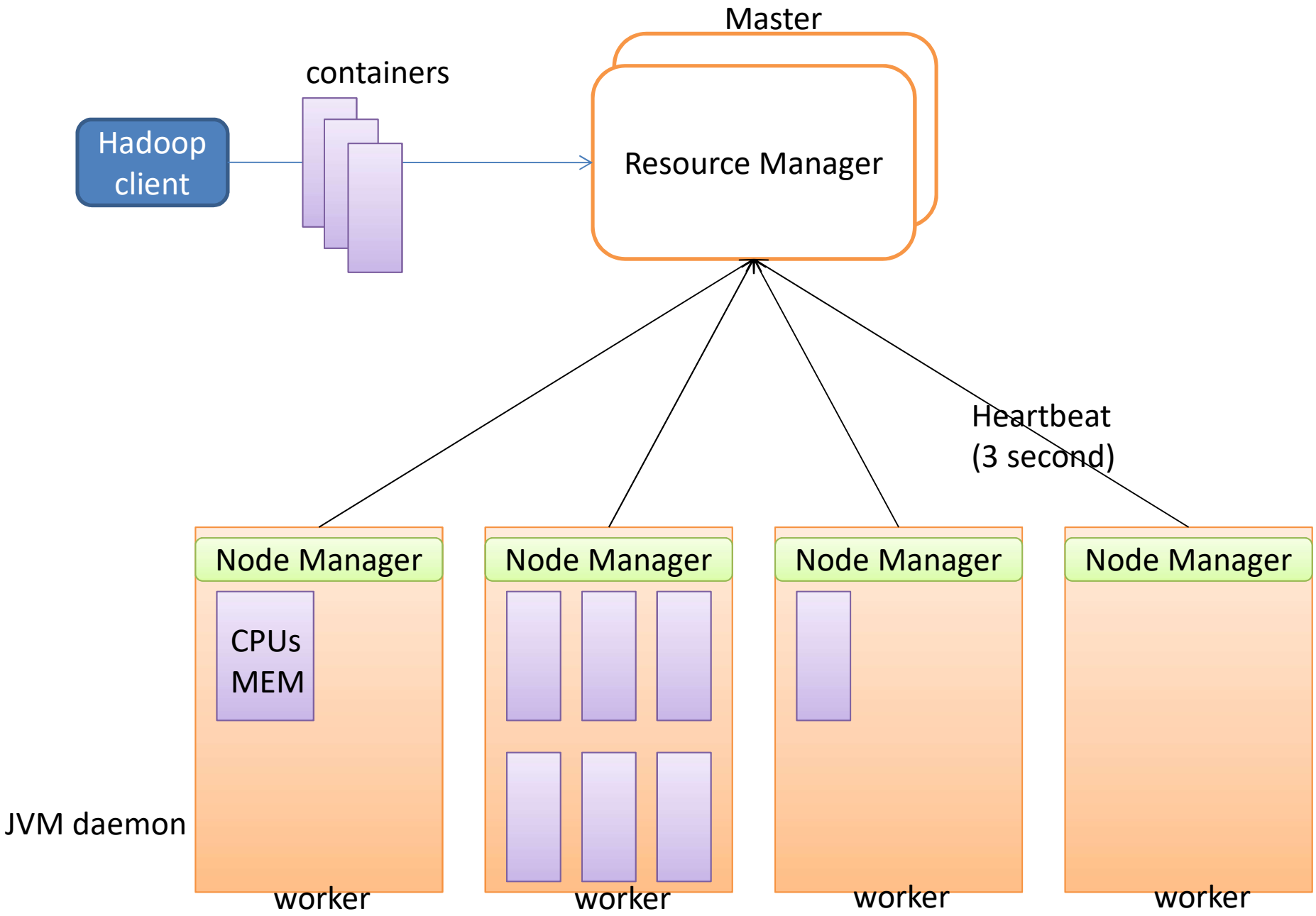
Block size and File size

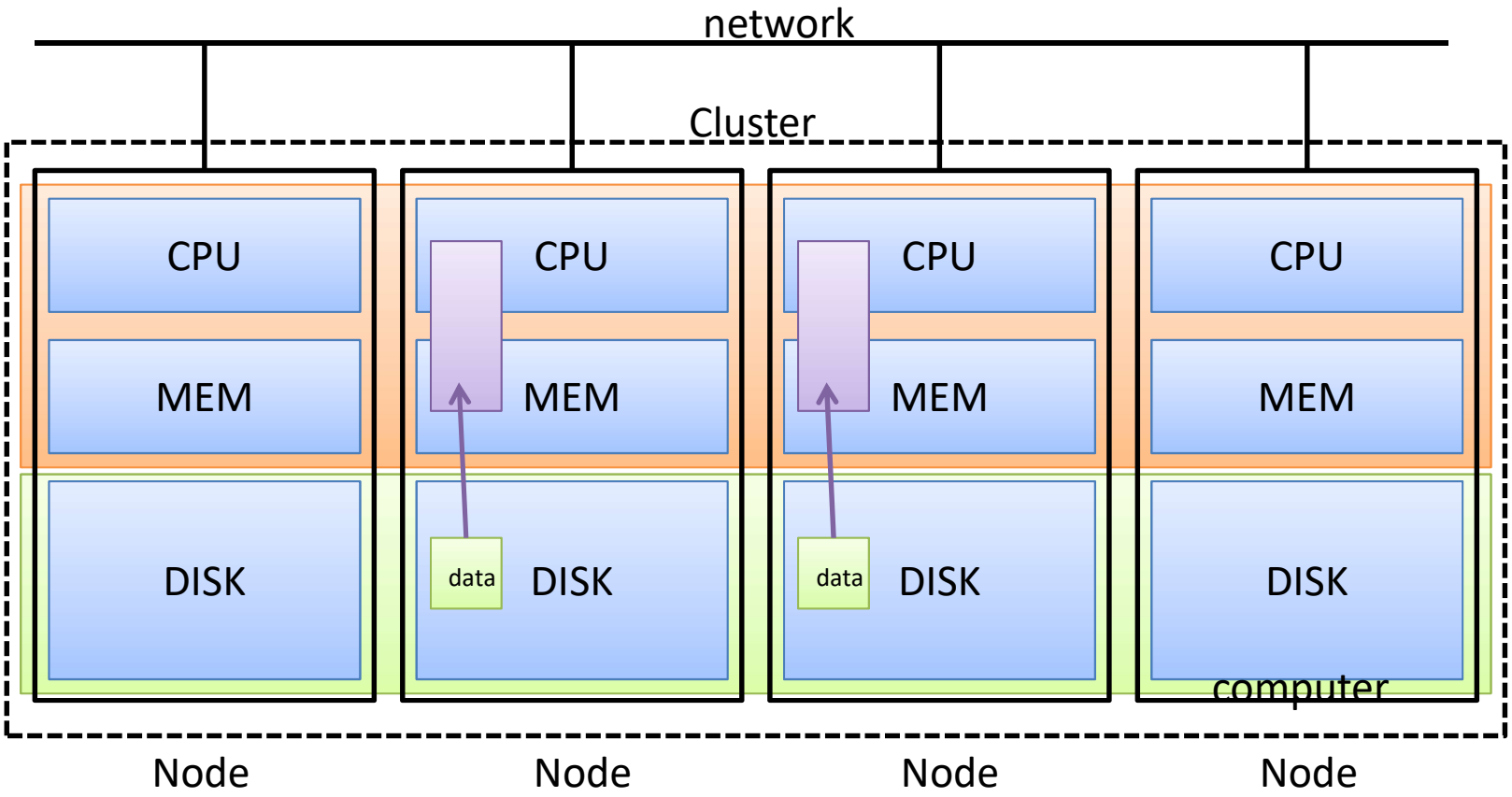
- 假設目前一個Block Size為128M，如果上傳文件小於128M，則該文件只由一個Block組成，但該Block以實際文件大小決定。
- 但每個file所佔的metadata資料空間還是一樣。所以為節省NameNode的記憶體，HDFS file應該為大量的大檔案，而不是極大量的小檔案所組成。


Hadoop Client
$ hdfs dfs –put xxxx
browser
File browser
Hue
Hue:8888
$ hdfs dfs –put xxxx
HDFS

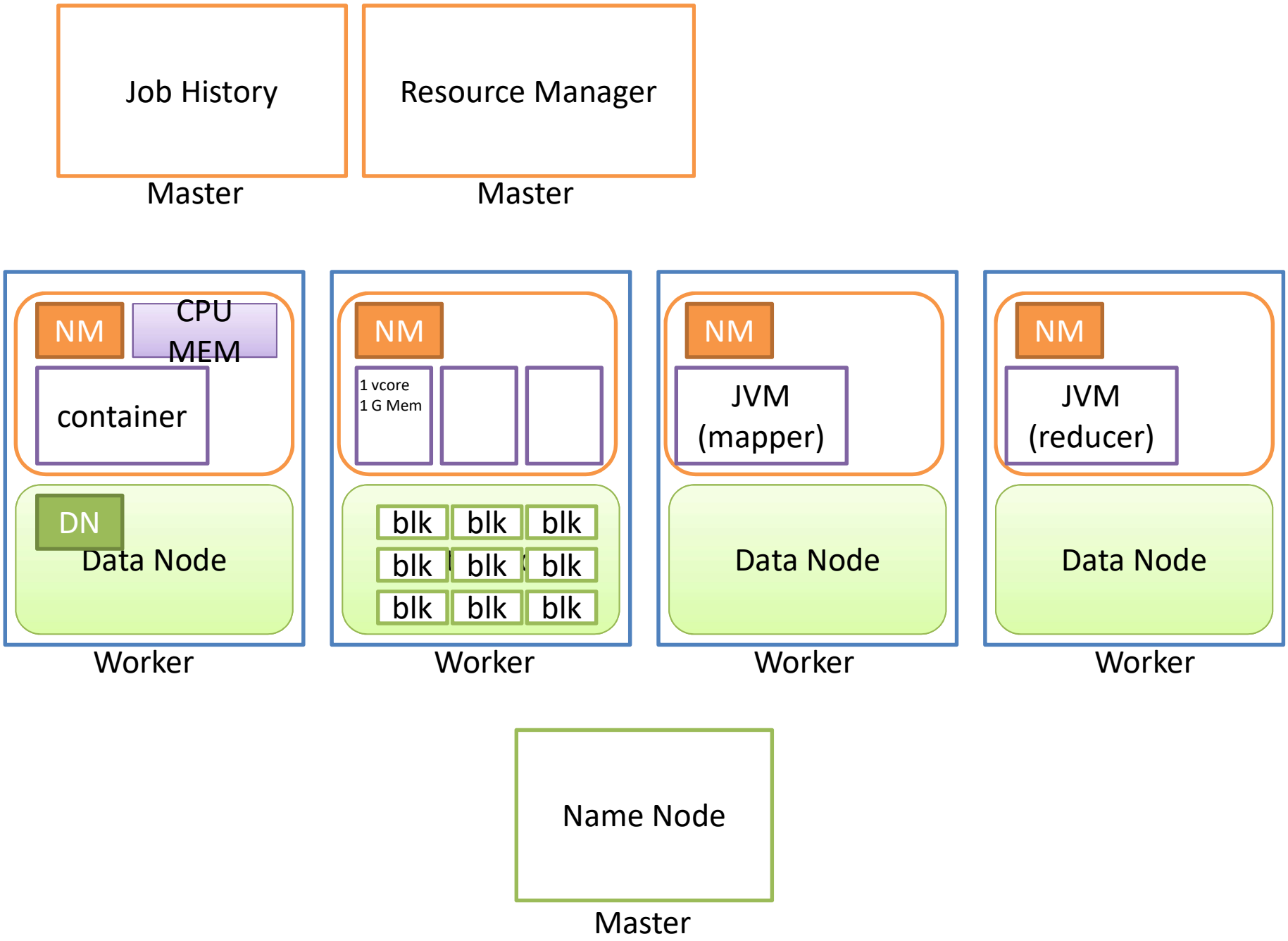

c1
c2
c3

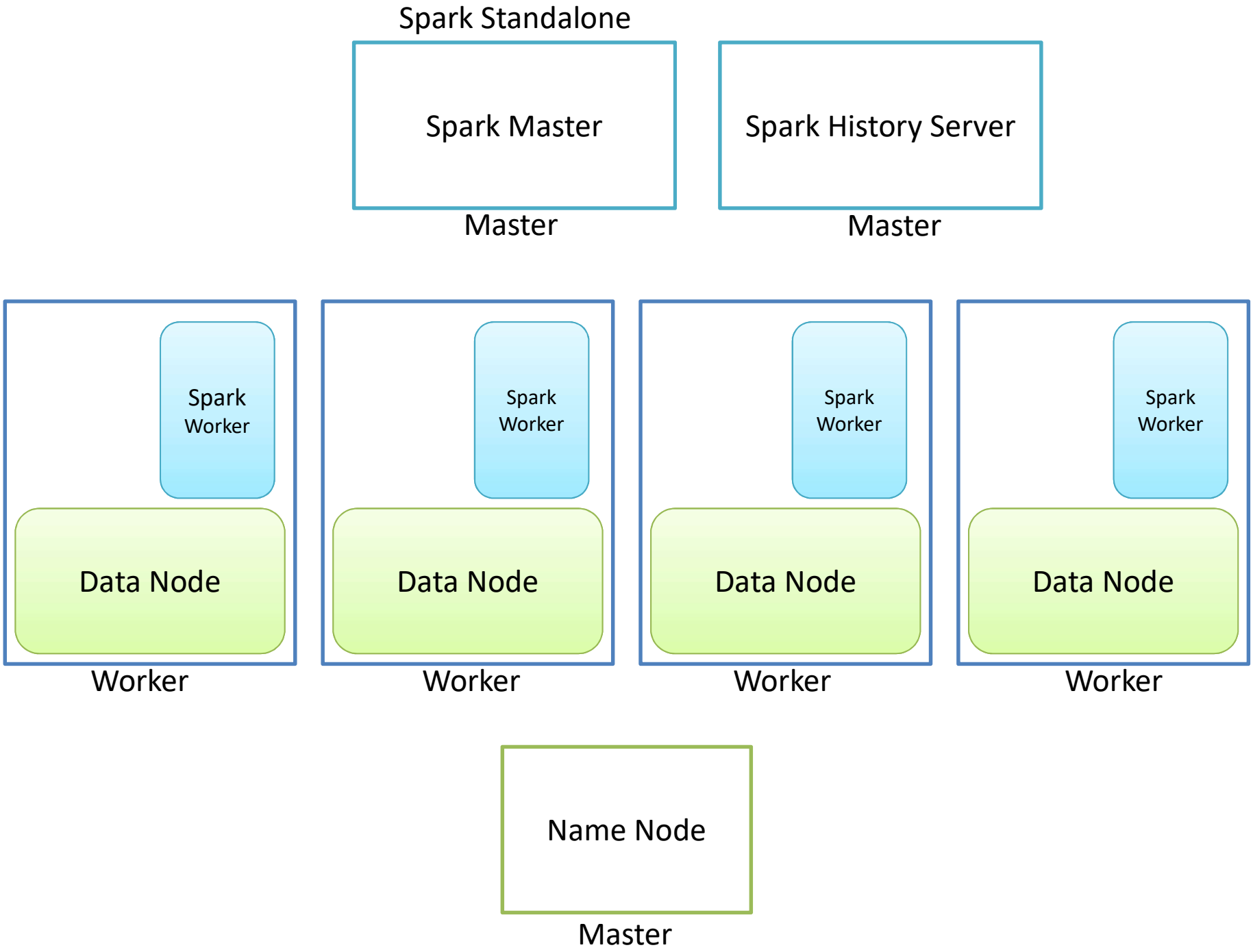

c1
c2
c3

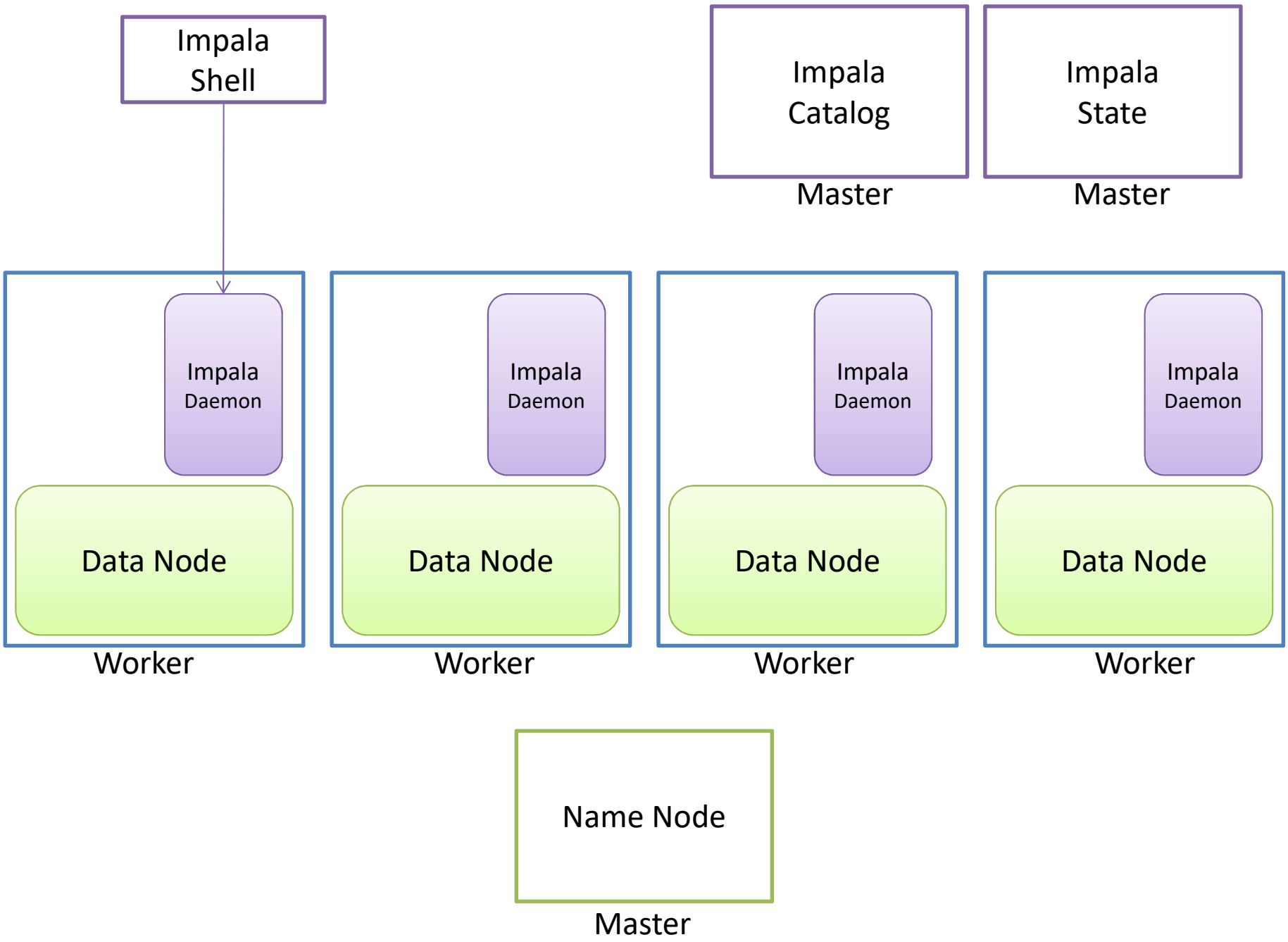

Master
containers
Hadoop client
Resource Manager
Heartbeat
(3 second)
Node Manager
Node Manager
Node Manager
Node Manager
CPUs
MEM
JVM daemon
worker
worker
worker
worker

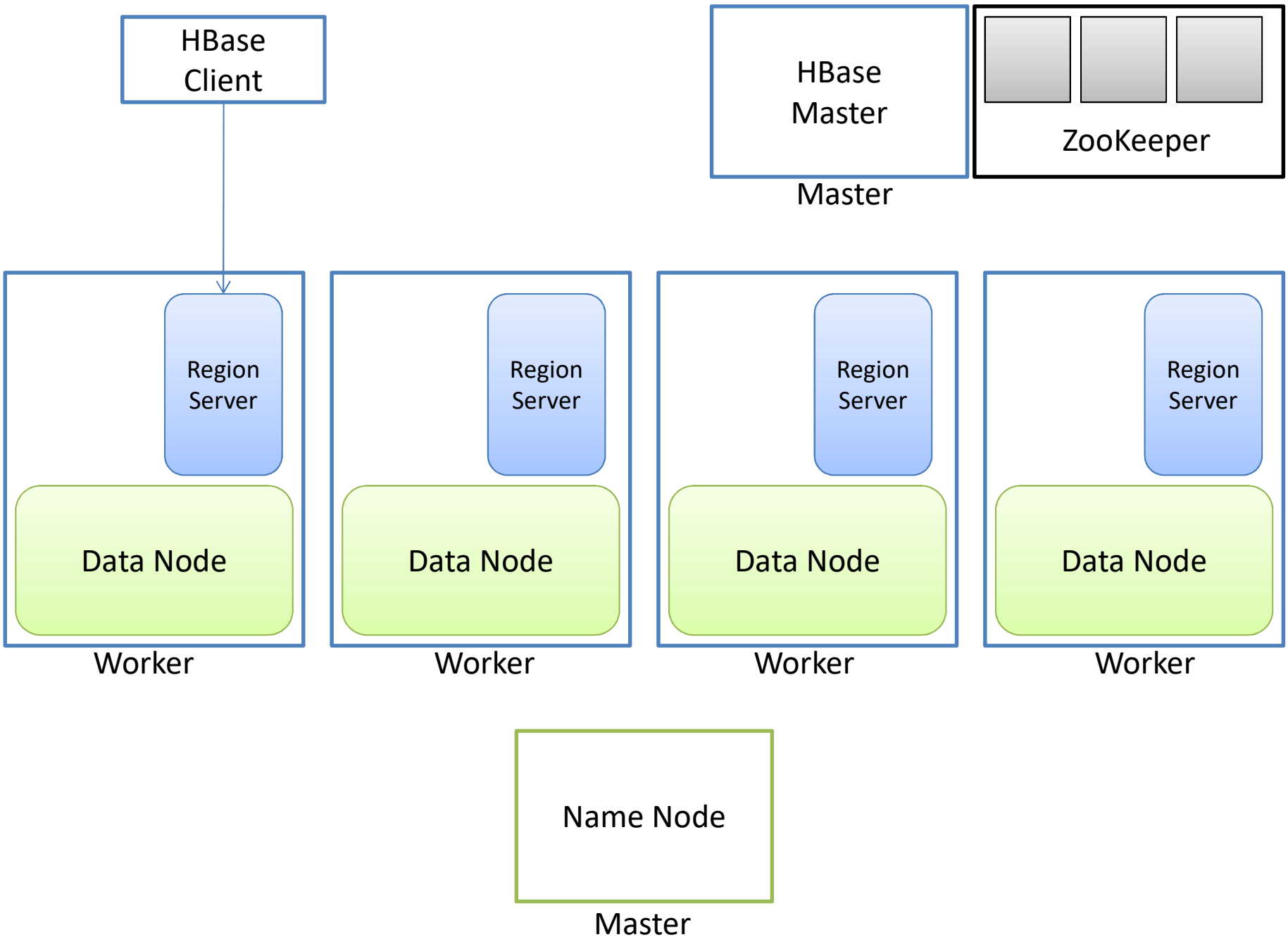


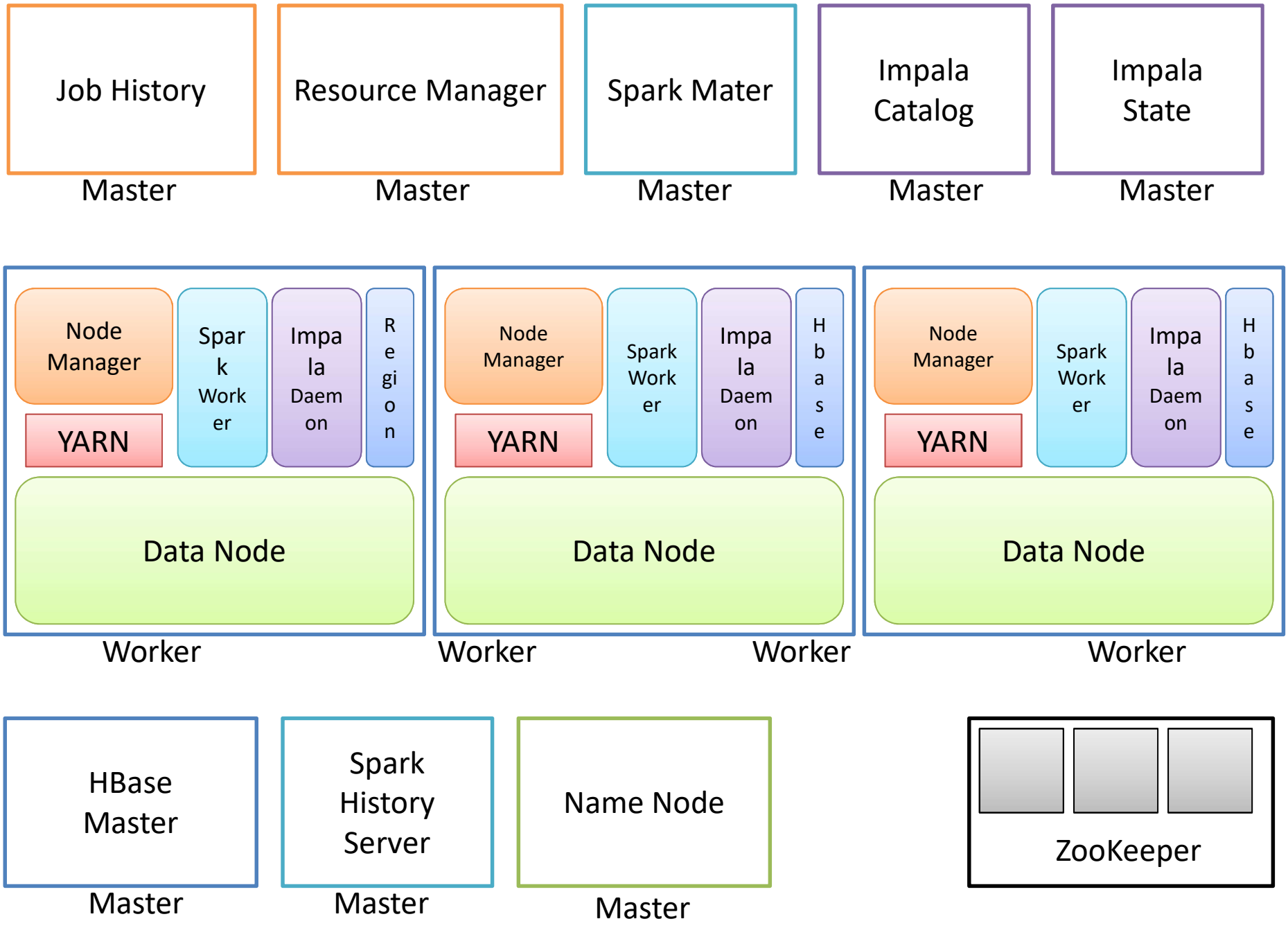
Data Locality

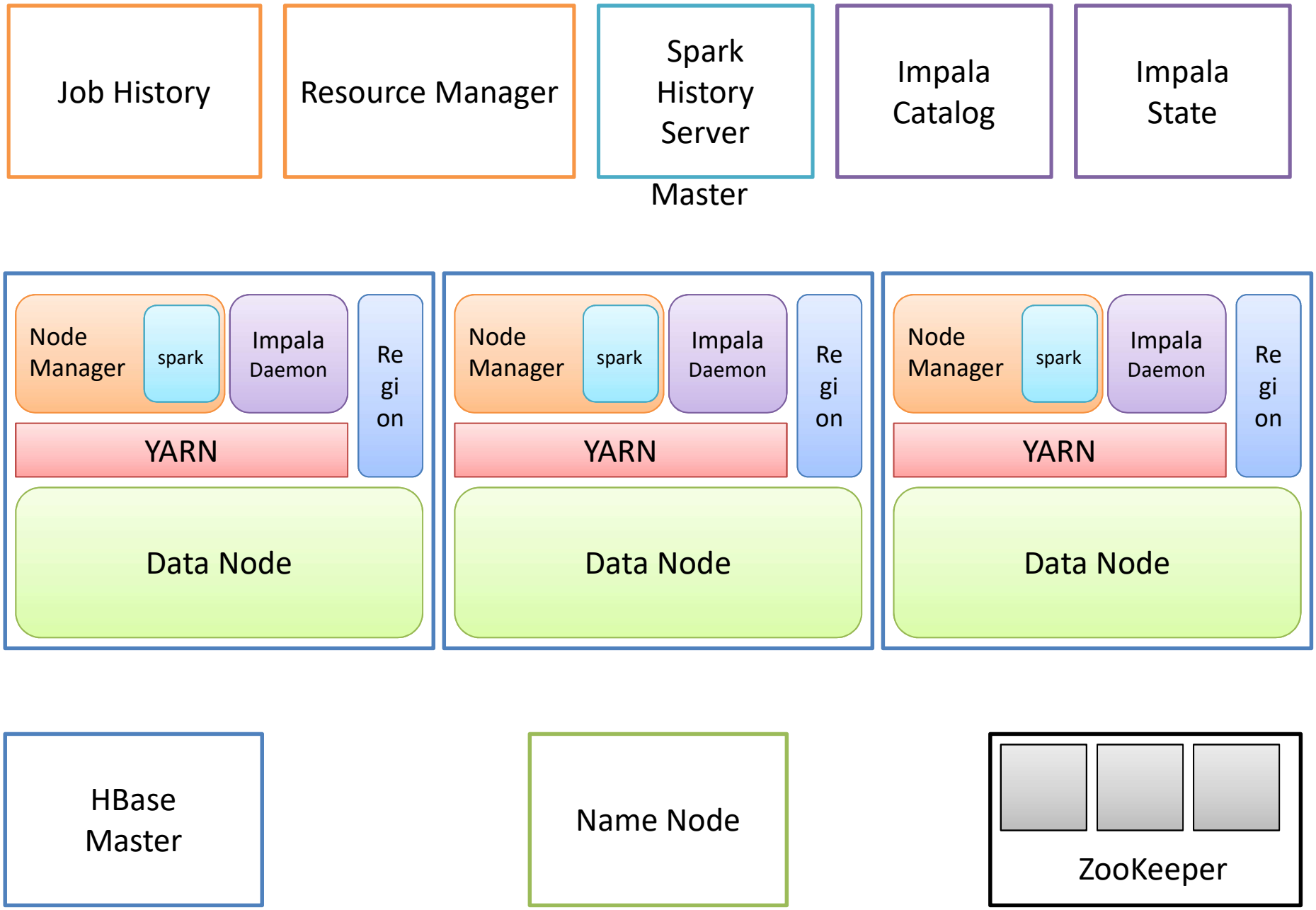

Job History
Resource Manager
Master
Master
NM
CPU
MEM
container
DN
Data Node
Worker
NM
1 vcore
1 G Mem
blk
blk
blk
blk
blk
blk
blk
blk
blk
Worker
NM
JVM
(mapper)
Data Node
Worker
NM
JVM
(reducer)
Data Node
Worker
Name Node
Master

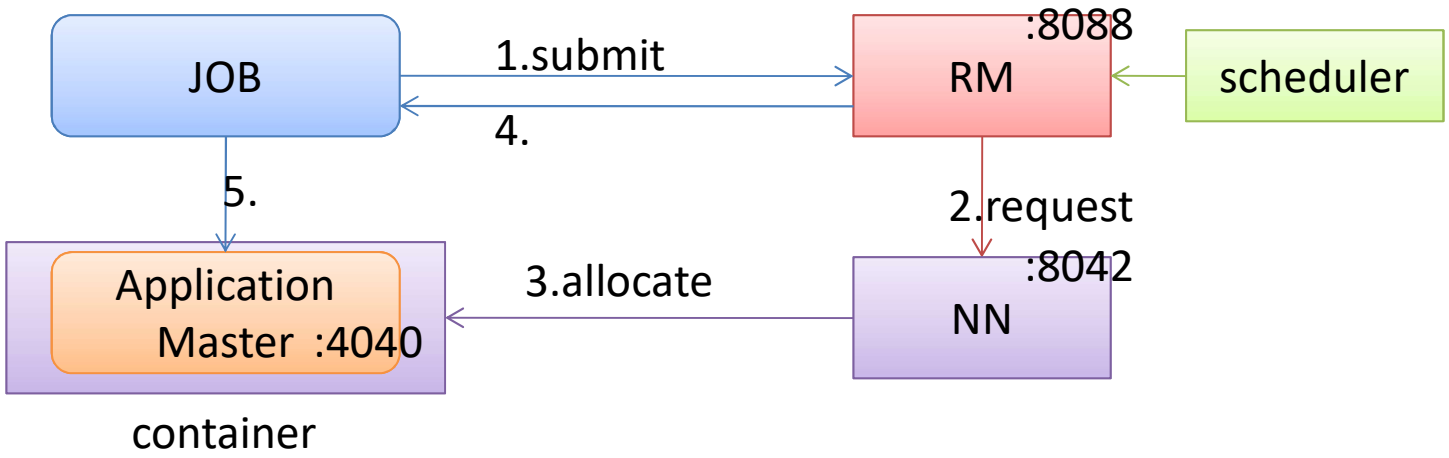

Spark Standalone
Spark Master
Master
Spark History Server
Master
Spark Worker
Data Node
Worker
Spark Worker
Data Node
Worker
Spark Worker
Data Node
Worker
Spark Worker
Data Node
Worker
Name Node
Master

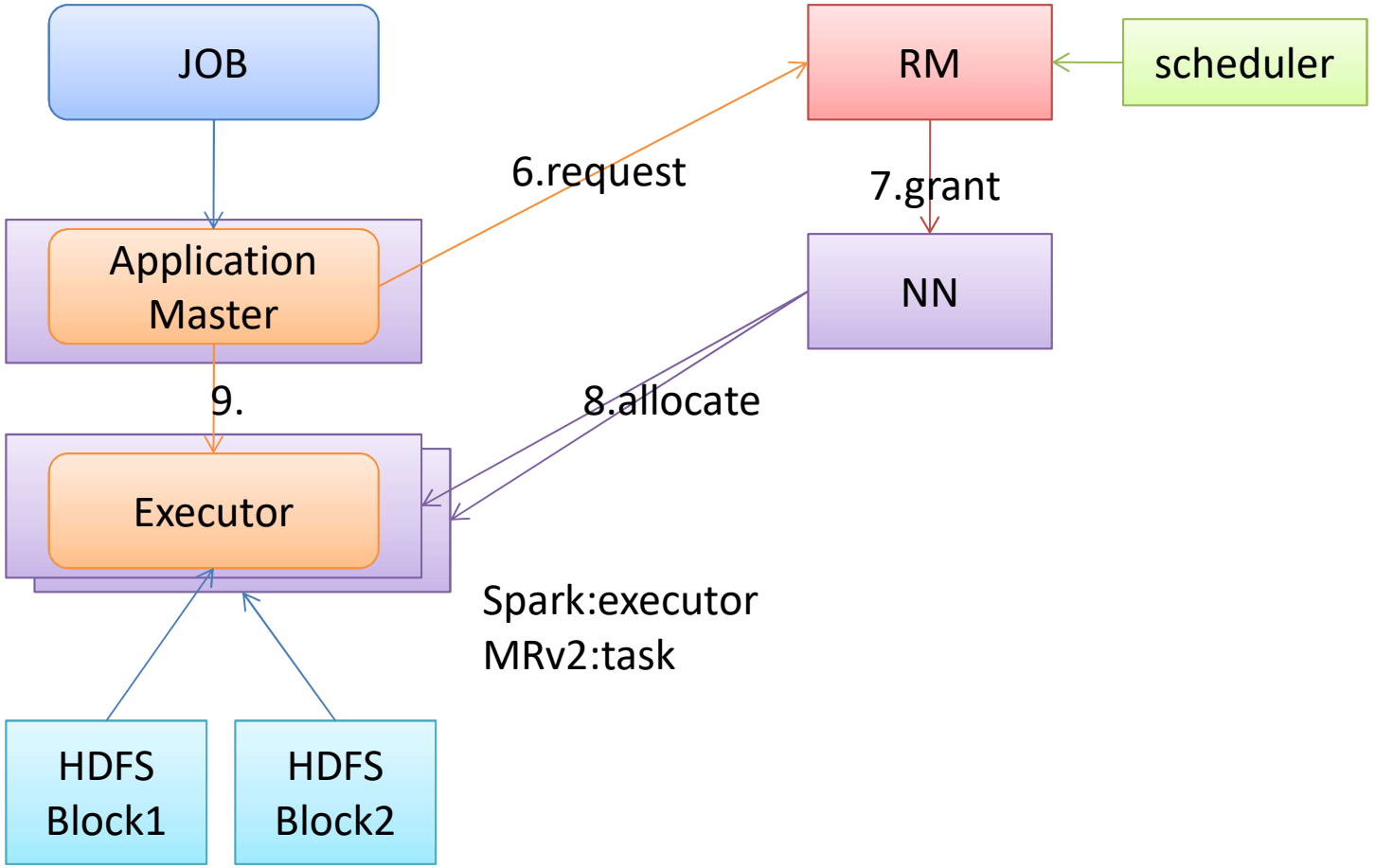

Impala
Shell
Impala
Catalog
Master
Impala
State
Master
Impala
Daemon
Data Node
Worker
Impala
Daemon
Data Node
Worker
Impala
Daemon
Data Node
Worker
Impala
Daemon
Data Node
Worker
Name Node
Master

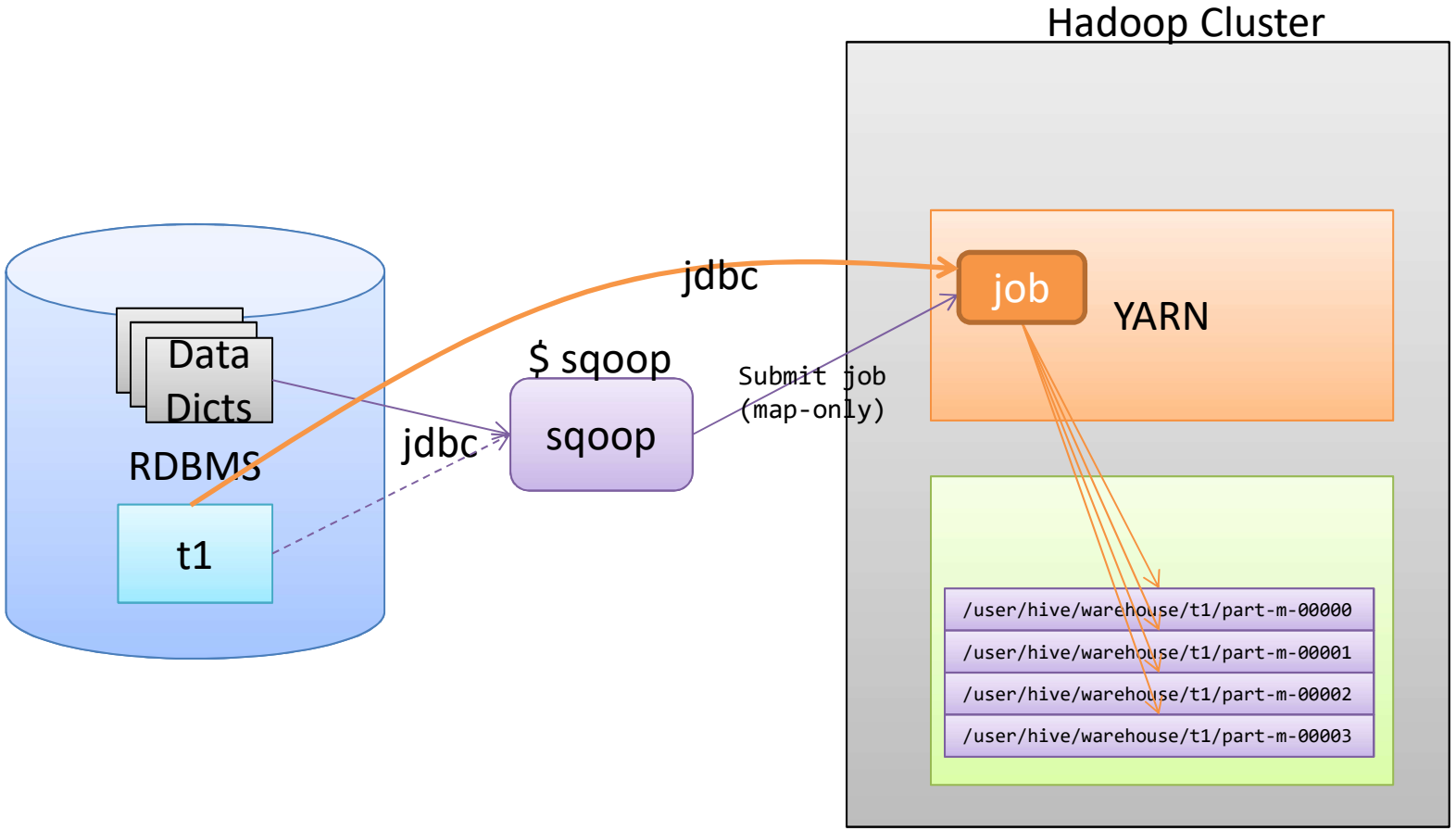

HBase
Client
HBase
Master
ZooKeeper
Master
Region
Server
Data Node
Worker
Region
Server
Data Node
Worker
Region
Server
Data Node
Worker
Region
Server
Data Node
Worker
Name Node
Master

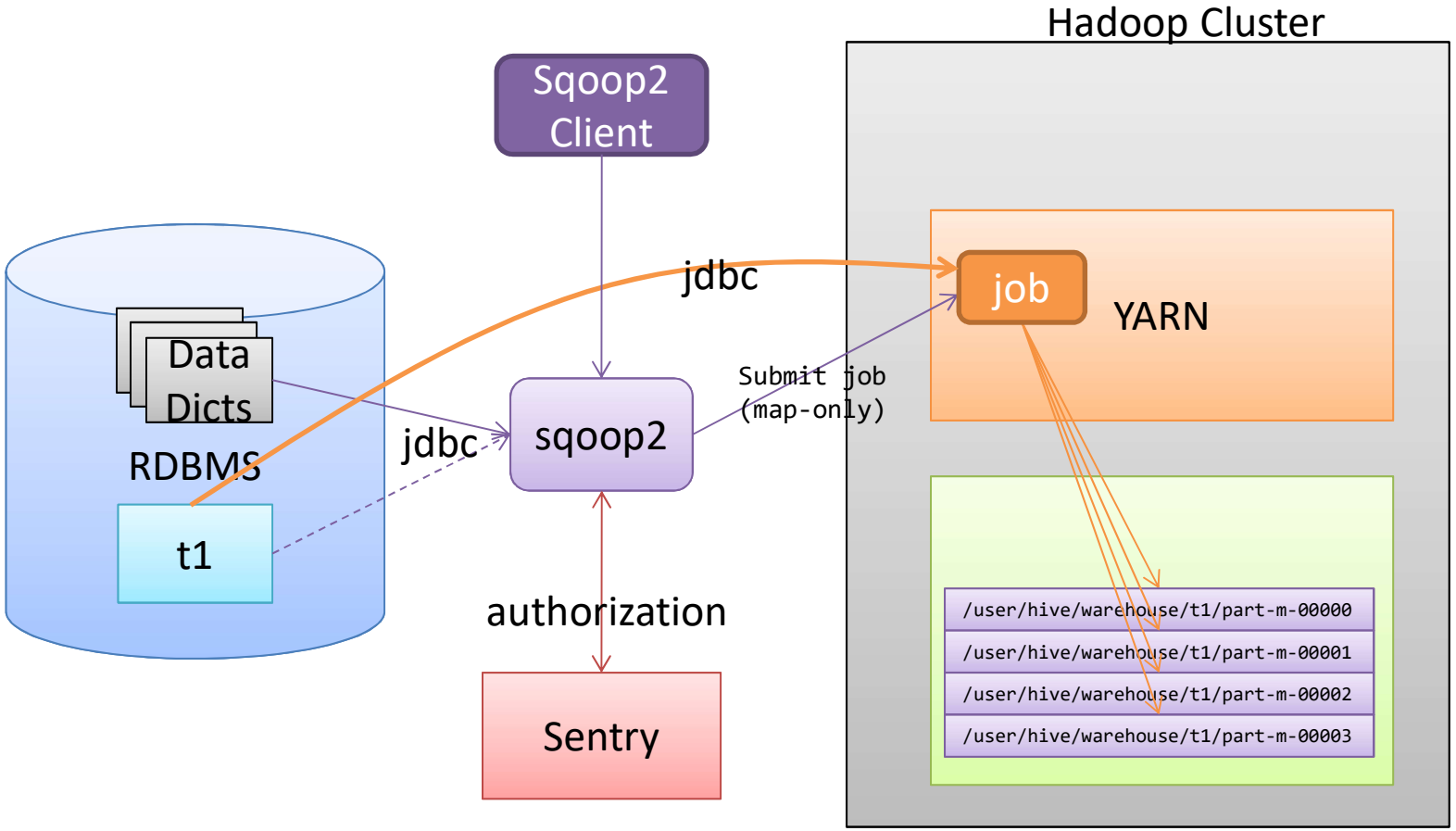

Job History
Master
Resource Manager
Master
Spark Mater
Master
Impala Catalog
Master
Impala State
Master
Node Manager
YARN
Spar k Work er
Impa la Daem on
R e gi o n
Data Node
Worker
Node Manager
YARN
Spark Work er
Impa la Daem on
H b a s e
Data Node
Worker
Worker
Node Manager
YARN
Spark Work er
Impa la Daem on
H b a s e
Data Node
Worker
HBase Master
Master
Spark History Server
Master
Name Node
Master
ZooKeeper


Job History
Resource Manager
Spark History Server
Impala Catalog
Impala State
Master
Node Manager
spark
Impala Daemon
Re gi on
YARN
Data Node
Node Manager
spark
Impala Daemon
Re gi on
YARN
Data Node
Node Manager
spark
Impala Daemon
Re gi on
YARN
Data Node
HBase Master
Name Node
ZooKeeper


JOB
1.submit
4.
RM
:8088
scheduler
5.
2.request
:8042
Application
Master :4040
3.allocate
NN
container


JOB
RM
scheduler
6.request
7.grant
Application Master
NN
9.
8.allocate
Executor
Spark:executor
MRv2:task
HDFS Block1
HDFS Block2


Hadoop Cluster
Data
Dicts
RDBMS
t1
jdbc
$ sqoop
sqoop
jdbc
Submit job
(map-only)
job
YARN
/user/hive/warehouse/t1/part-m-00000
/user/hive/warehouse/t1/part-m-00001
/user/hive/warehouse/t1/part-m-00002
/user/hive/warehouse/t1/part-m-00003


Hadoop Cluster
Sqoop2
Client
jdbc
job
YARN
Data
Dicts
RDBMS
t1
jdbc
sqoop2
Submit job
(map-only)
authorization
Sentry
/user/hive/warehouse/t1/part-m-00000
/user/hive/warehouse/t1/part-m-00001
/user/hive/warehouse/t1/part-m-00002
/user/hive/warehouse/t1/part-m-00003

```
$ sqoop import \
--connect jdbc:mysql://localthost/loudacre \
--username dbuser --password pw \
--table accounts
--where "state='CA'" \
--columns "id,first_name,last_name,state" \
```

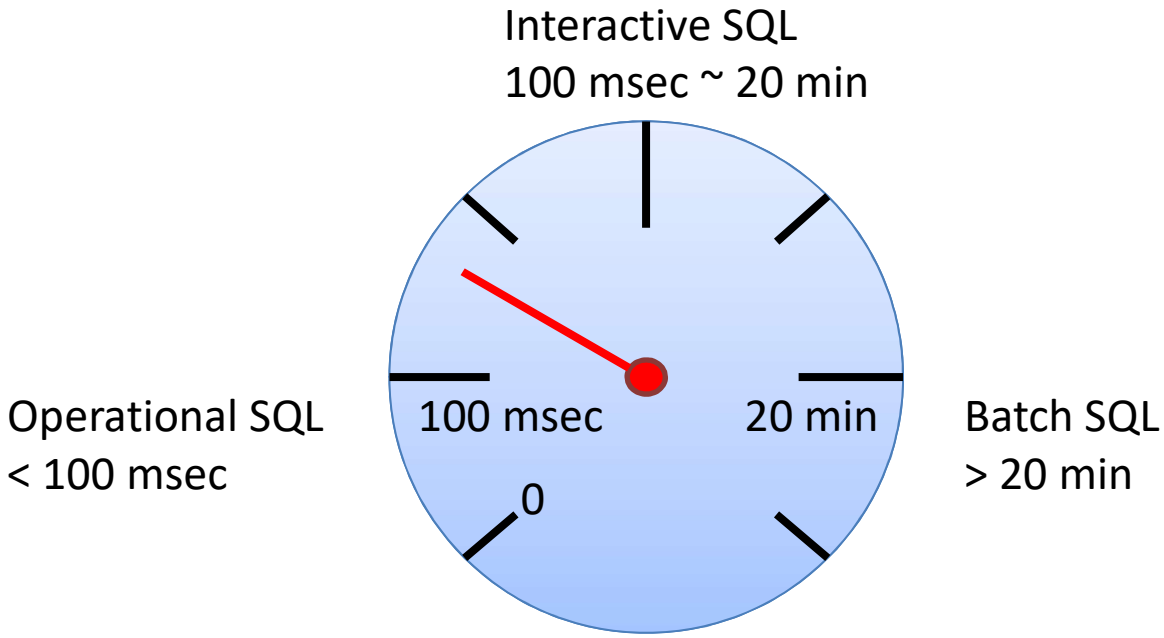
```
SELECT id,first_name,last_name,state
FROM accounts
WHERE state='CA'
```

```
$ sqoop import \
--connect jdbc:mysql://localthost/loudacre \
--username dbuser --password pw \
--target-dir /data/loudacre/payable
--split-by account.id
--query " SELECT
id,first_name,last_name,state
FROM accounts
WHERE \$CONDITION AND state='CA'"
```

```
$ sqoop import \
--connect jdbc:mysql://dbhost/loudacre \
--username dbuser --password pw \
--target-dir /data/loudacre/payable \
--split-by accounts.id \
--query 'SELECT accounts.id, first_name, last_name,
bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE $CONDITIONS AND
bill_amount >= 40'
```

```
$ sqoop import \
--connect jdbc:mysql://dbhost/loudacre \
--username dbuser --password pw \
--target-dir /data/loudacre/payable \
--split-by accounts.id \
--query "SELECT accounts.id, first_name, last_name,
bill_amount FROM accounts JOIN invoices ON
(accounts.id = invoices.cust_id) WHERE \$CONDITIONS AND
bill_amount >= 40"
```

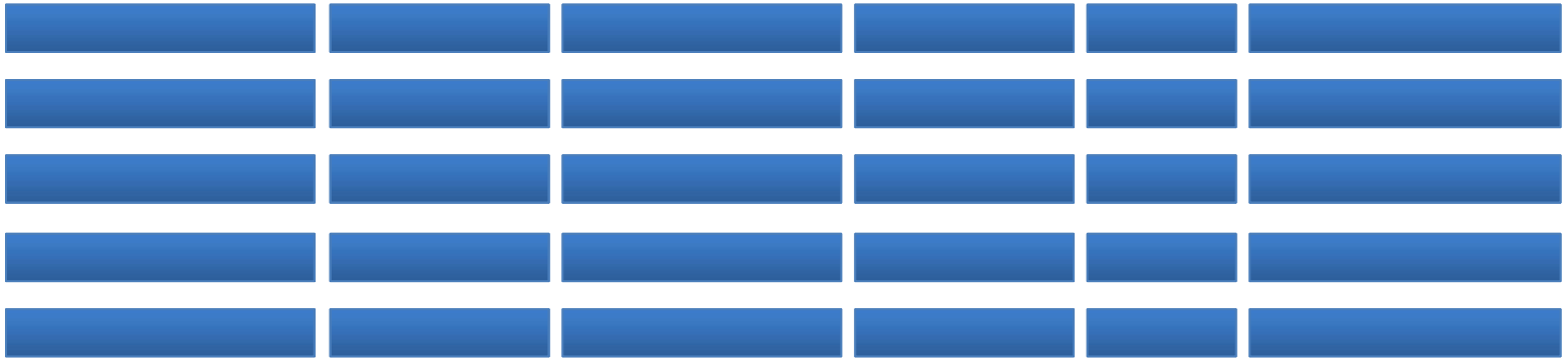
SQL on Hadoop



SQL on Hadoop

- Batch SQL
  - Hive
    - Based on MapReduce
    - Based on Tez(Hortonworks)
- Interactive SQL
  - Spark SQL(Hive on Spark)
  - Impala(Cloudera)(Google Dremel)
  - Apache Drill(Google Dremel)
  - Presto(Facebook) (Google Dremel)
- OLAP
  - Pinot(LinkedIn)
  - Kylin(Ebay)
- Operational SQL
  - Apache Phoenix(SQL on HBase)

Schema on Write

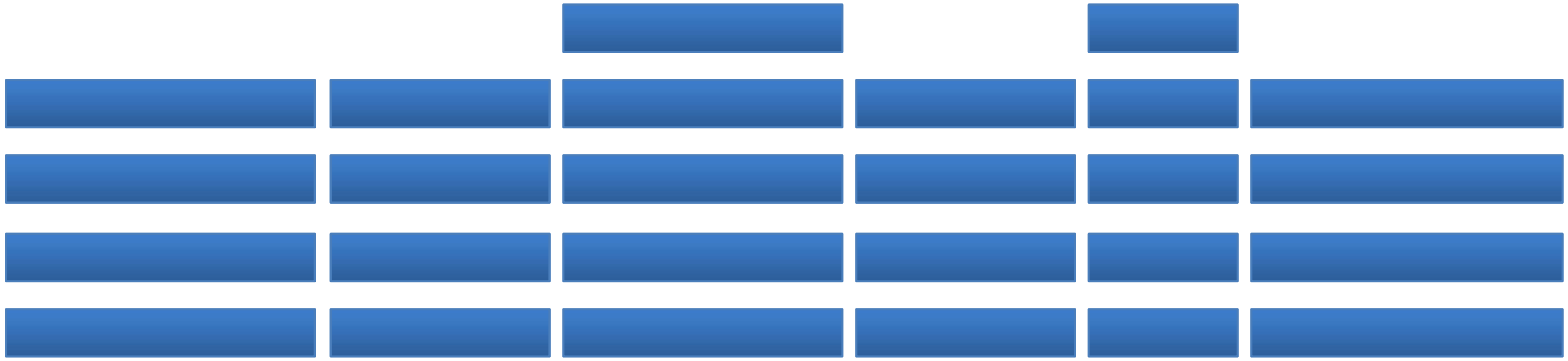


2.Load data into table

1.Create table

Empid	Ename	Salary	Hire_date

Schema on Write



2.Load data into table

1.Create table

Empid	Ename	Salary	Hire_date

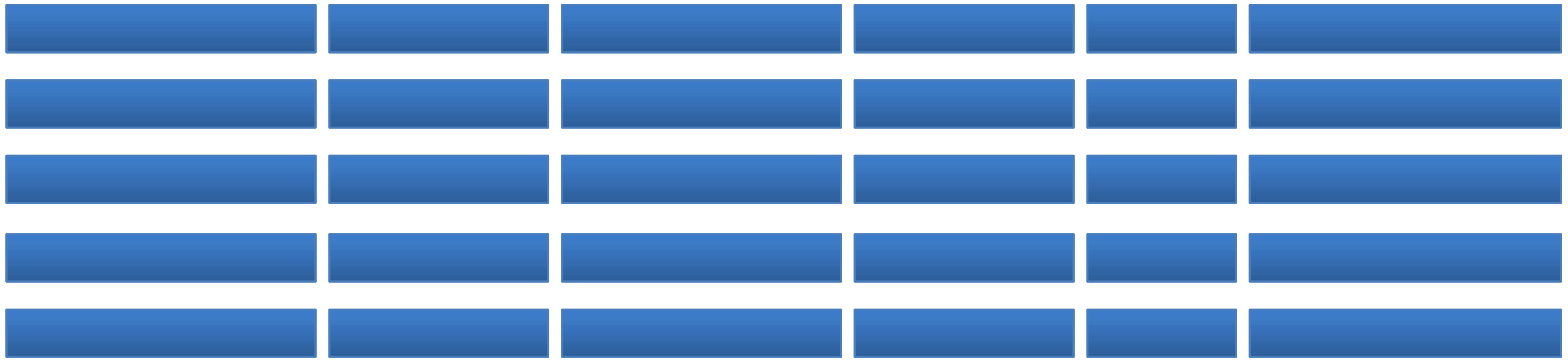
Schema on Write

2.Load data into table

1.Create table

Empid	Ename	Salary	Hire_date

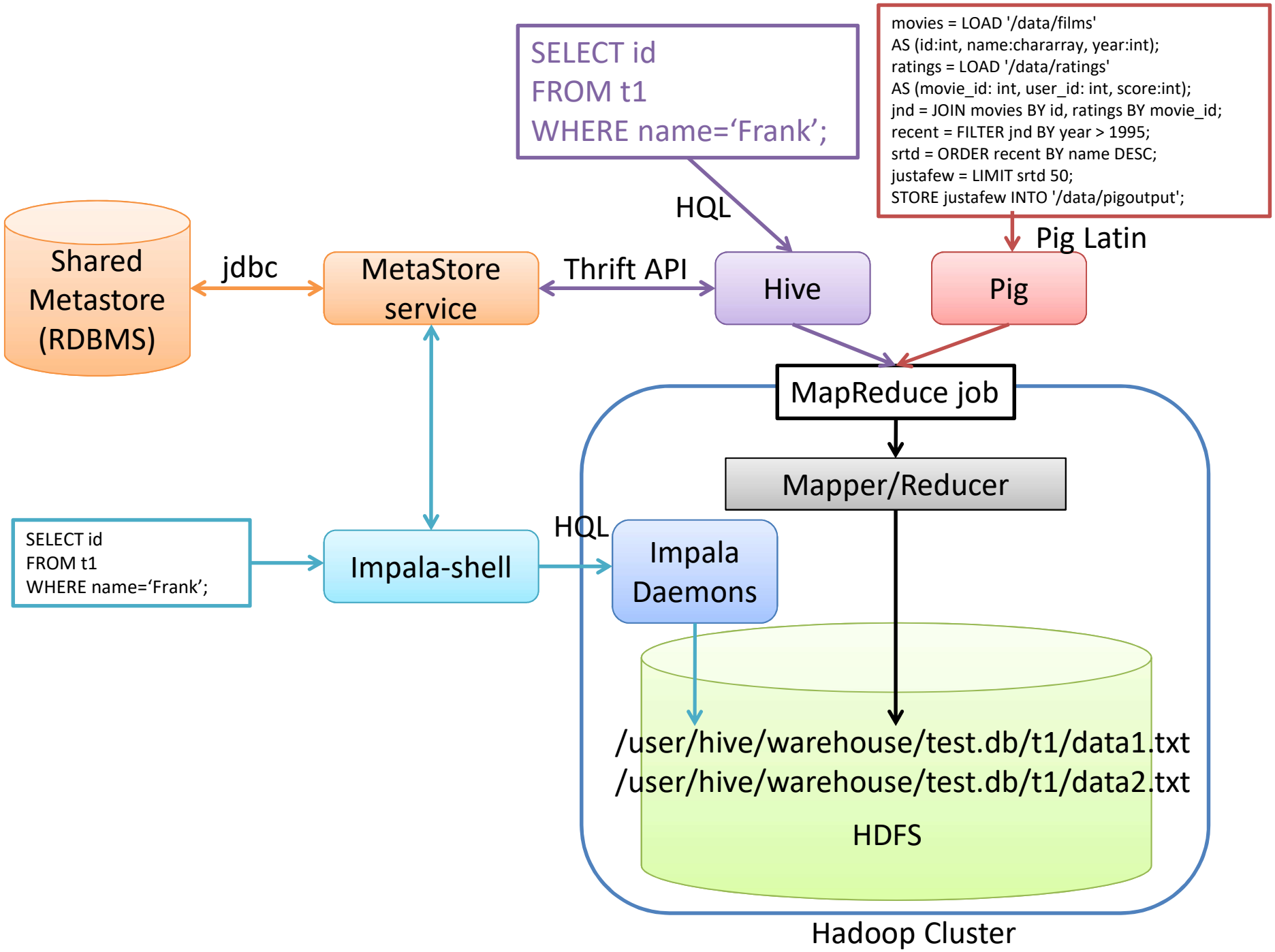
Schema on Read

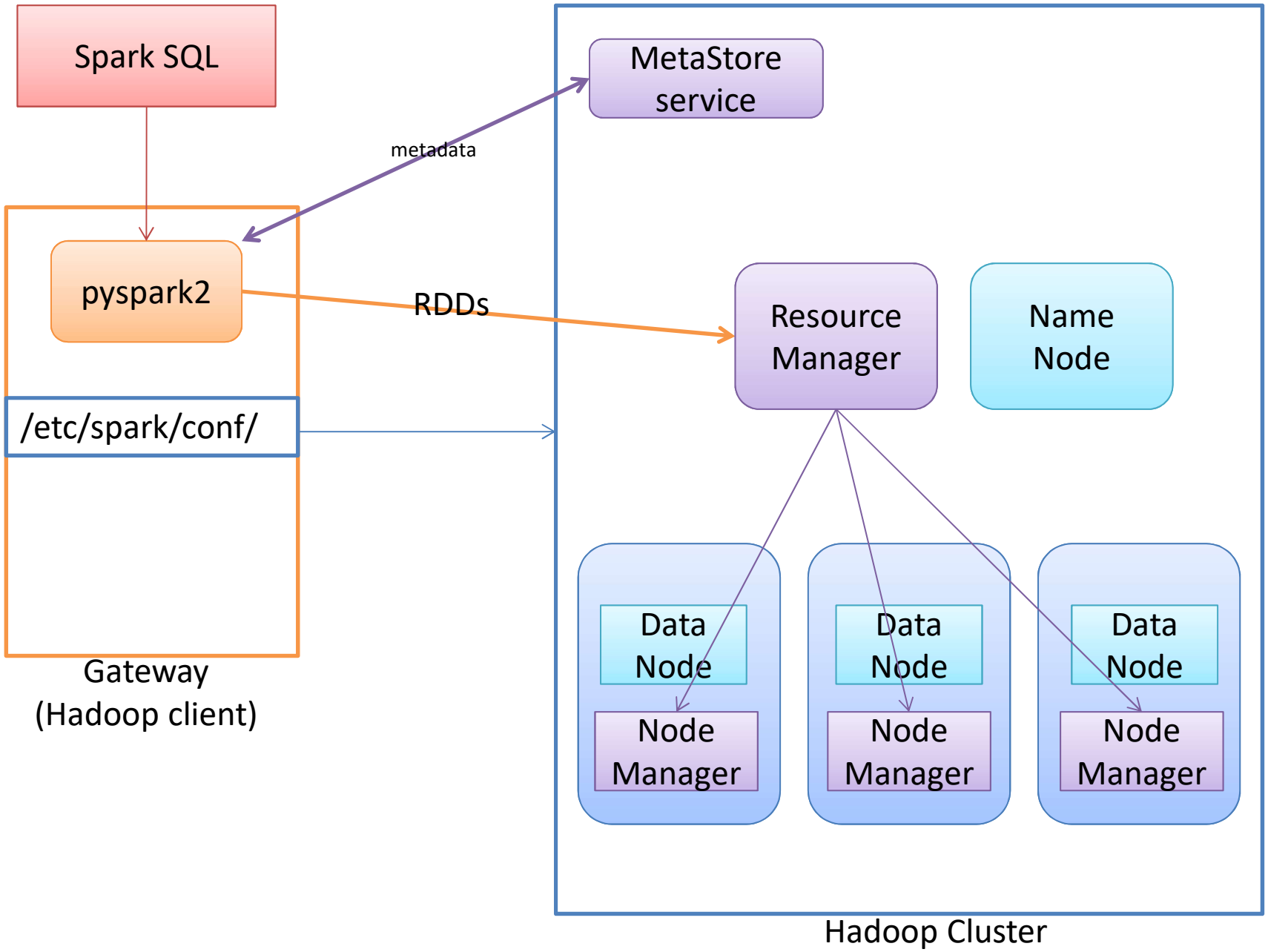


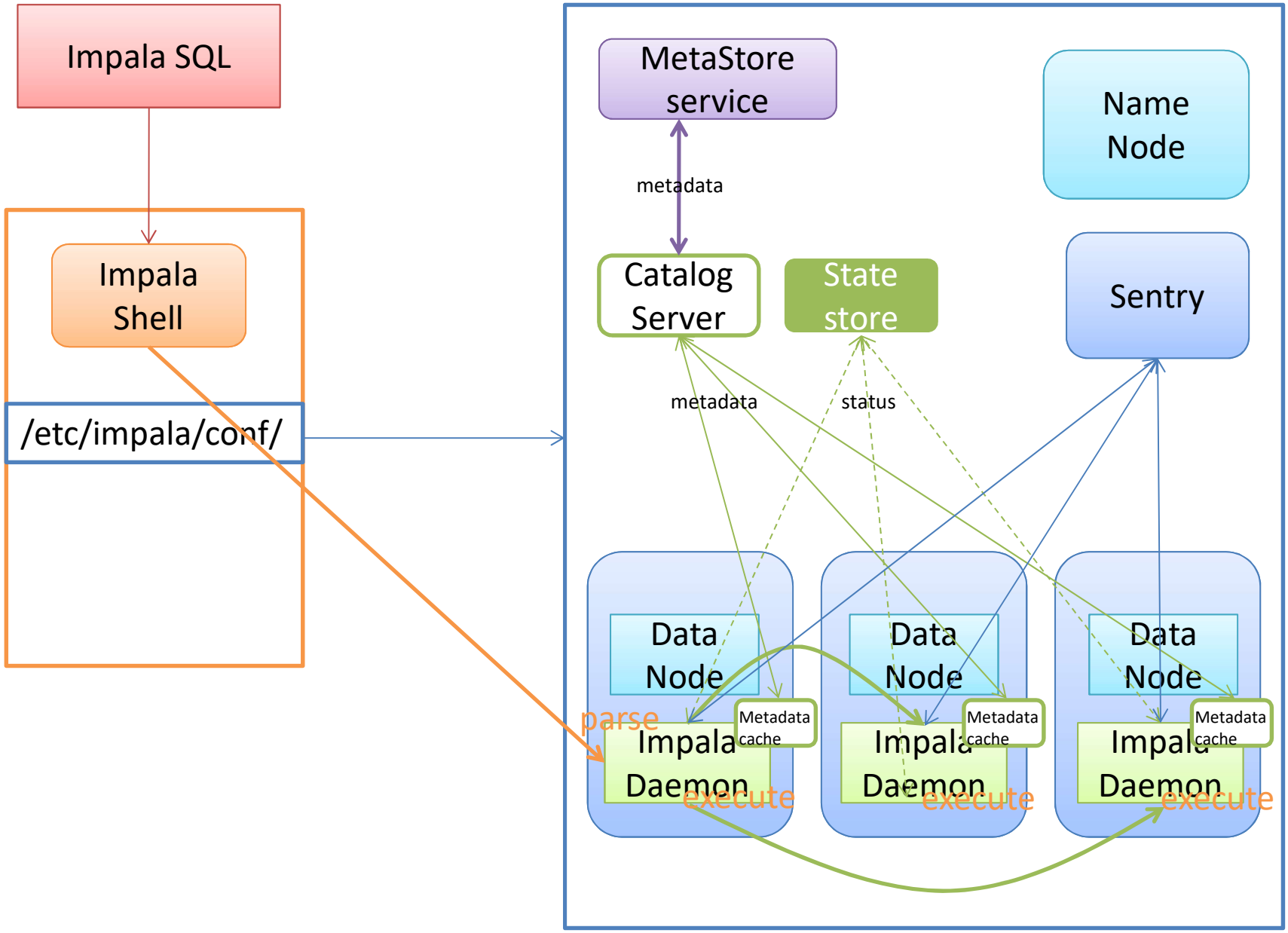
1.Load data into HDFS

Schema on Read

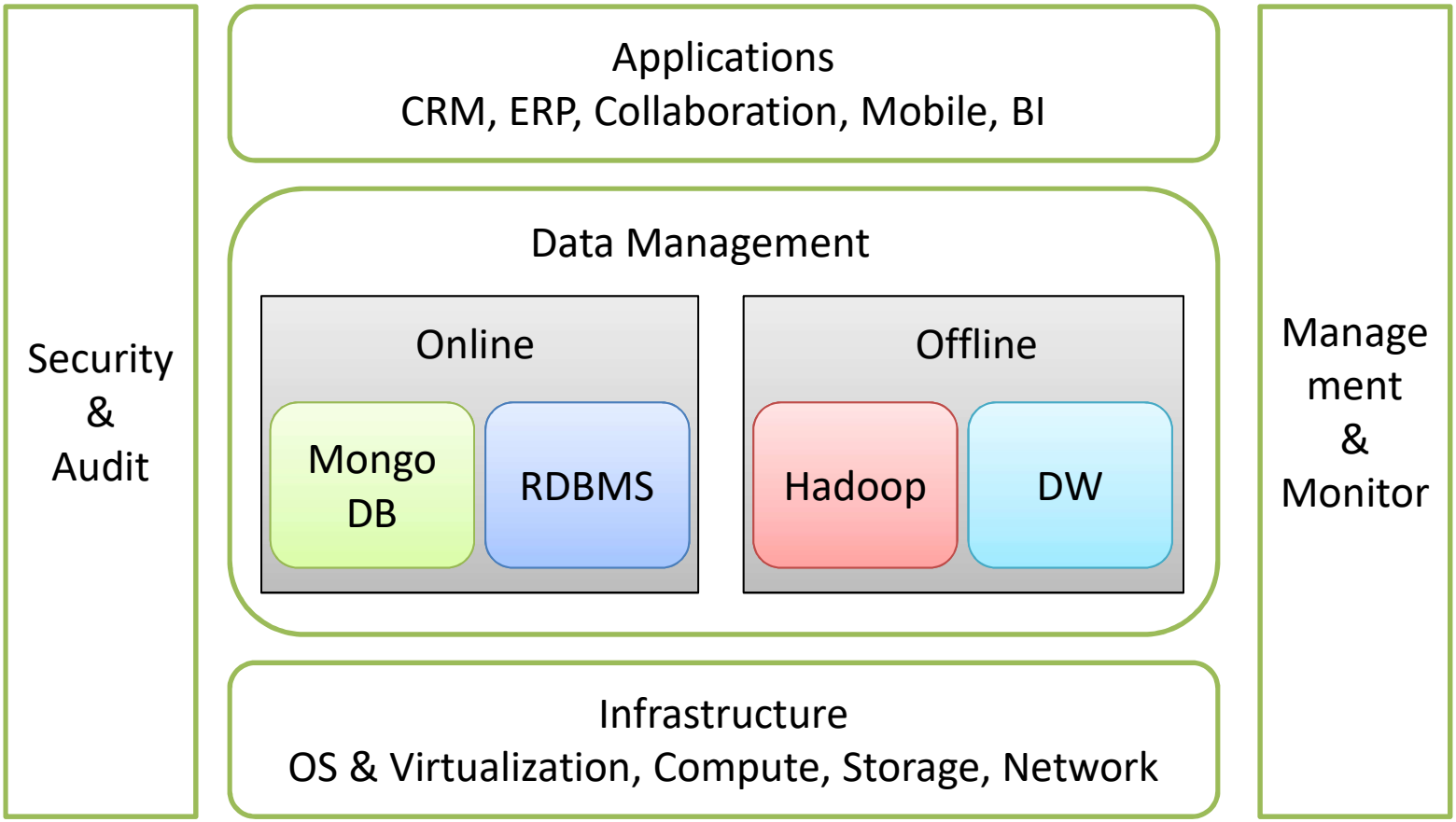
2. Create Hive table mapping HDFS file


SELECT id
FROM t1
WHERE name='Frank';
movies = LOAD '/data/films'
AS (id:int, name:chararray, year:int);
ratings = LOAD '/data/ratings'
AS (movie_id: int, user_id: int, score:int);
jnd = JOIN movies BY id, ratings BY movie_id;
recent = FILTER jnd BY year > 1995;
srtd = ORDER recent BY name DESC;
justafew = LIMIT srtd 50;
STORE justafew INTO '/data/pigoutput';
HQL
Pig Latin
Shared Metastore (RDBMS)
jdbc
MetaStore service
Thrift API
Hive
Pig
MapReduce job
Mapper/Reducer
SELECT id
FROM t1
WHERE name='Frank';
Impala-shell
HQL
Impala Daemons
/user/hive/warehouse/test.db/t1/data1.txt
/user/hive/warehouse/test.db/t1/data2.txt
HDFS
Hadoop Cluster


Spark SQL
MetaStore service
metadata
pyspark2
RDDs
Resource Manager
Name Node
/etc/spark/conf/
Data Node
Node Manager
Data Node
Node Manager
Data Node
Node Manager
Gateway
(Hadoop client)
Hadoop Cluster


Impala SQL
Impala Shell
/etc/impala/conf/
MetaStore service
metadata
Name Node
Catalog Server
State store
Sentry
metadata
status
Data Node
Metadata cache
Impala Daemon
parse
execute
Data Node
Metadata cache
Impala Daemon
execute
Data Node
Metadata cache
Impala Daemon
execute

Enterprise Big Data Stack

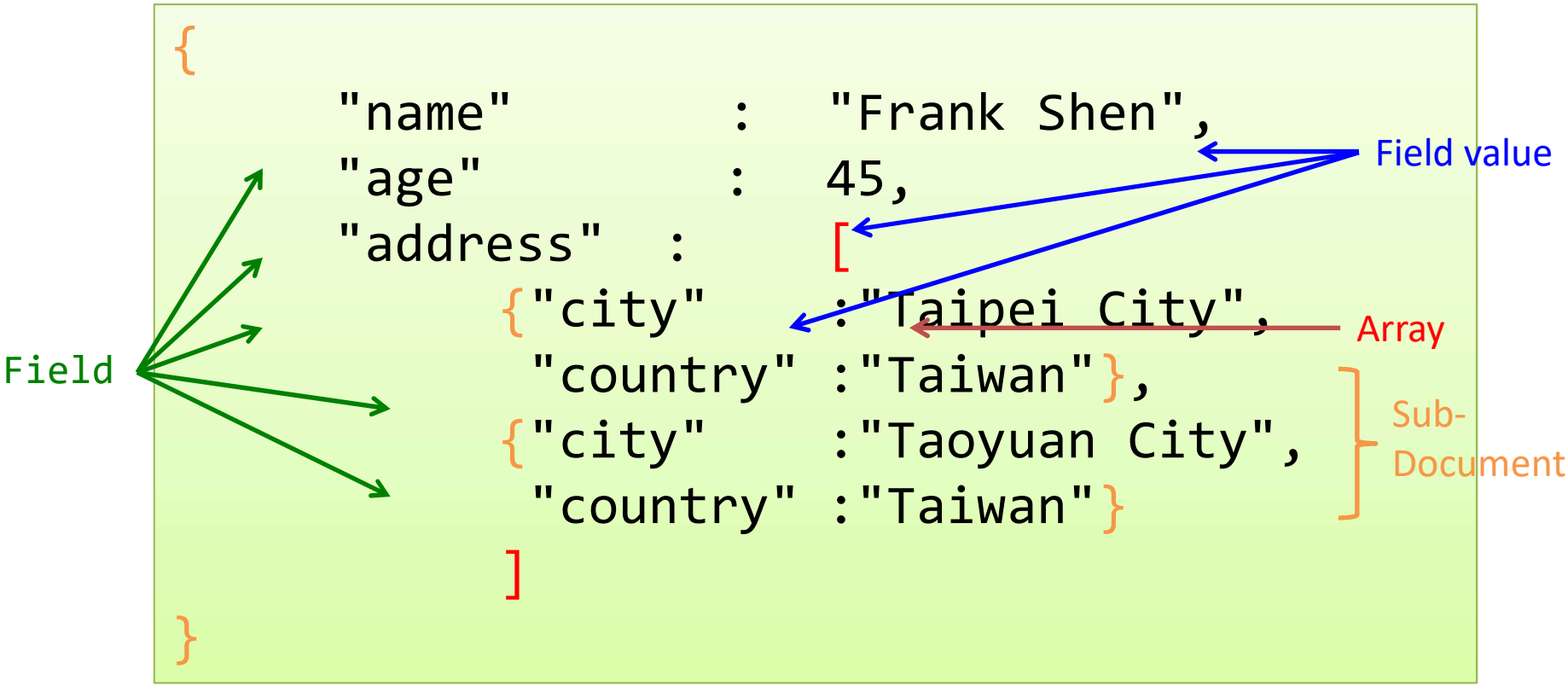


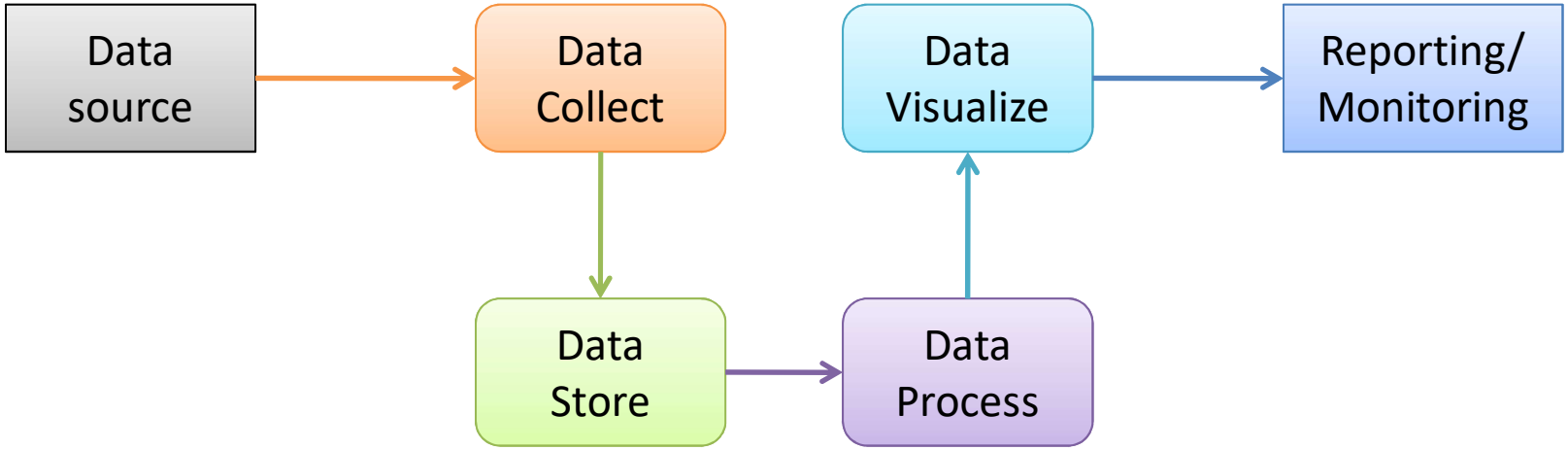
Document是類JSON格式

```
{
      "name"          :   "Frank Shen",
      "age"           :   45,
      "address"   :       [
            {"city"     :       "Taipei City",
             "country" :       "Taiwan"},
            {"city"     :       "Taoyuan City",
             "country" :       "Taiwan"}
            ]
}
```

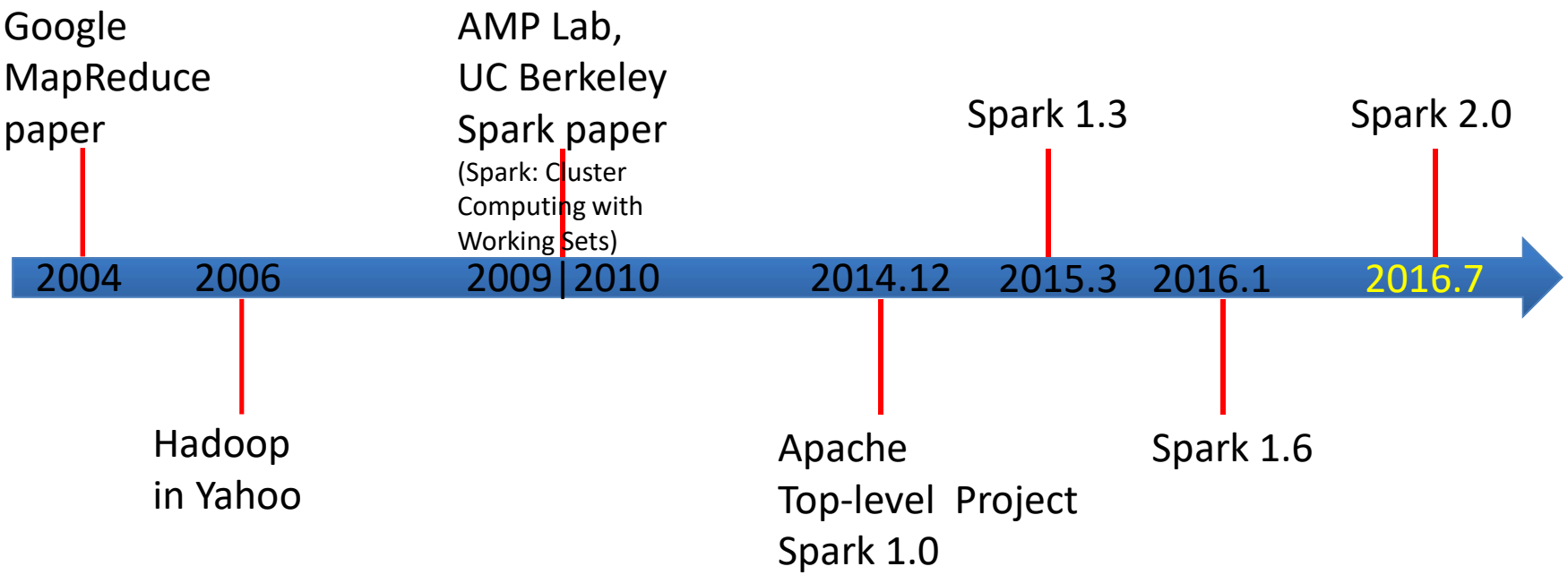
```
{
            "name"        : “Wilson Chang",
            "age"         : 38
}
```

Document結構


{
"name" : "Frank Shen",
"age" : 45,
"address" : [
{"city" :"Taipei City",
"country" :"Taiwan"},
{"city" :"Taoyuan City",
"country" :"Taiwan"}
]
}
Field
Field value
Array
Sub-
Document


Data source
Data Collect
Data Visualize
Reporting/ Monitoring
Data Store
Data Process

Brief History of Spark



What and Why Spark

- Fast and expressive cluster computing system interoperable with Apache Hadoop
- Improves efficiency through:
  - In-memory computing primitives
  - General computation graphs
- Improves usability through:
  - Rich APIs in Scala,Java,Python
  - Interactive shell

Up to 100x faster
(2 ~ 10x on Disks)

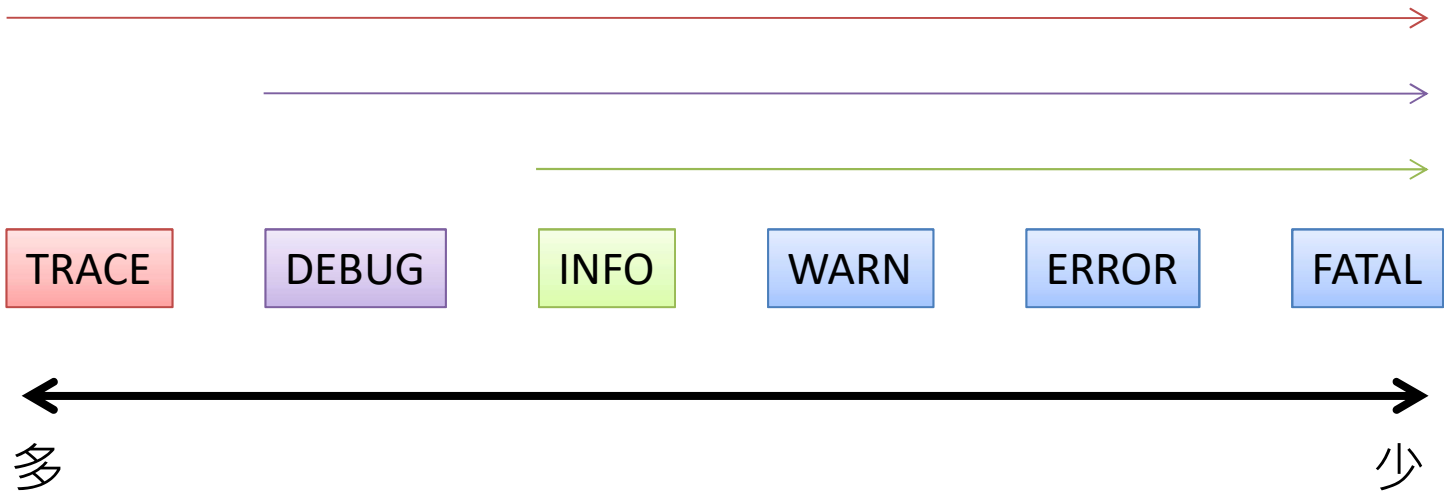
Often 5x less on code

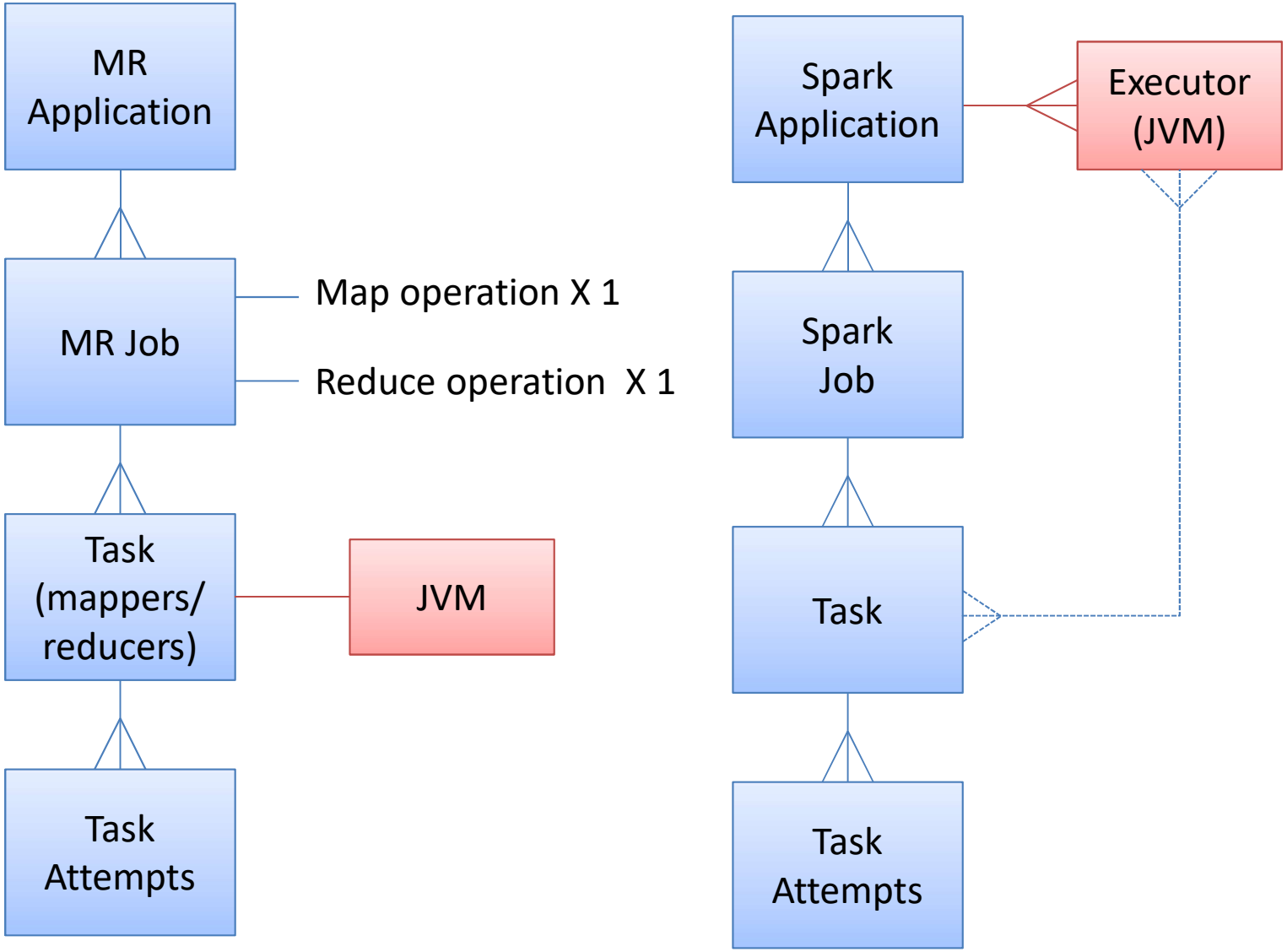
Java

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException
{
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Spark

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" "))
                  .map(lambda word: (word, 1))
                  .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```


TRACE
DEBUG
INFO
WARN
ERROR
FATAL
多
少


MR Application
MR Job
Map operation X 1
Reduce operation X 1
Task (mappers/ reducers)
JVM
Task Attempts
Spark Application
Executor (JVM)
Spark Job
Task
Task Attempts

RDD to DataFrame

```
pdata.map(lambda x: (x.dept, [x.age, 1])) \
  .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
  .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
  .collect()
```

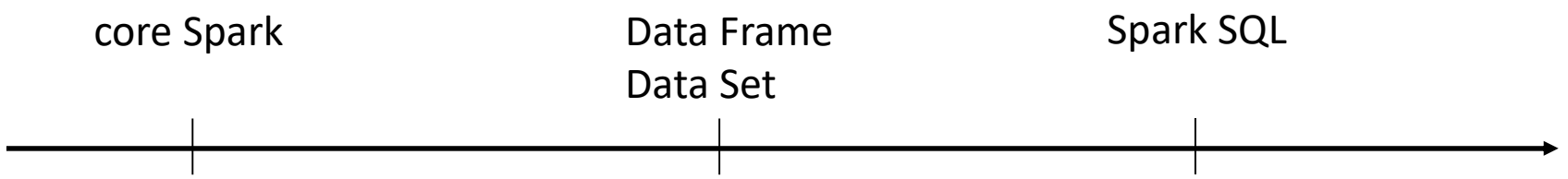
```
data.groupBy("dept").avg("age")
```


資料來源:databricks

SQL integrate with Spark

```
sqlcontext = HiveContext(sc)
results = sqlcontext.sql("SELECT country,count(*) FROM people
GROUP BY country")
names = results.map(lambda p: p.name)
```

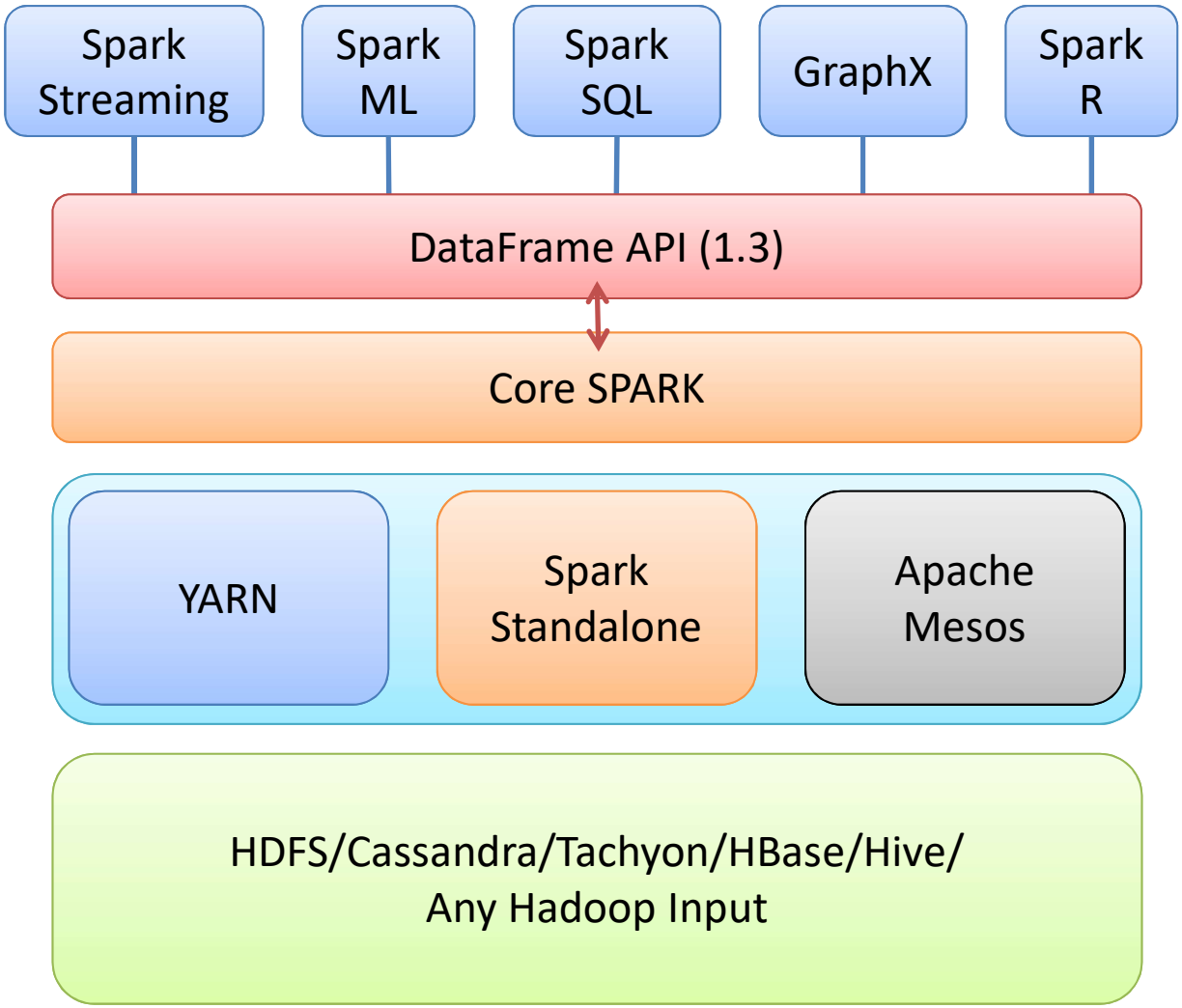
Spark Coding 演進



What is DataFrame

- Distributed collection of data grouped into named columns
  - Inspired by data frames in R and Python Pandas
  - Conceptually equivalent to a table in RDBMS
- Supported Data Formats and Sources
  - Structured files by Parquet、JSON、S3、HDFS、Cassandra
  - Tables in Hive、RDBMS
- Available in Python、Scala、Java and R

Dataframe is future of Spark(1.3)

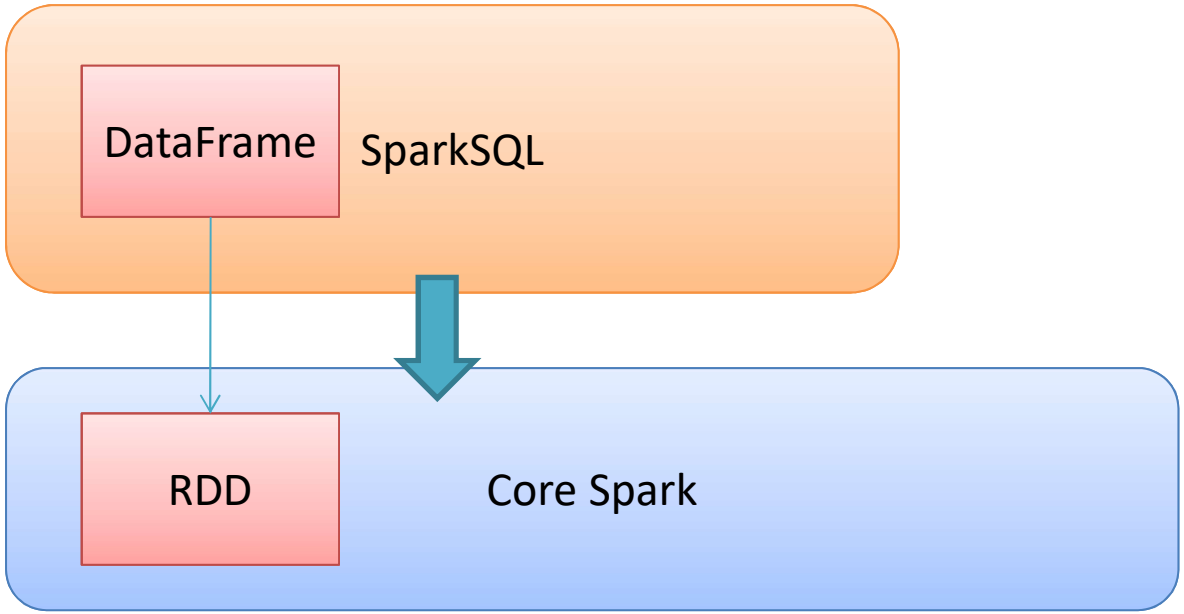


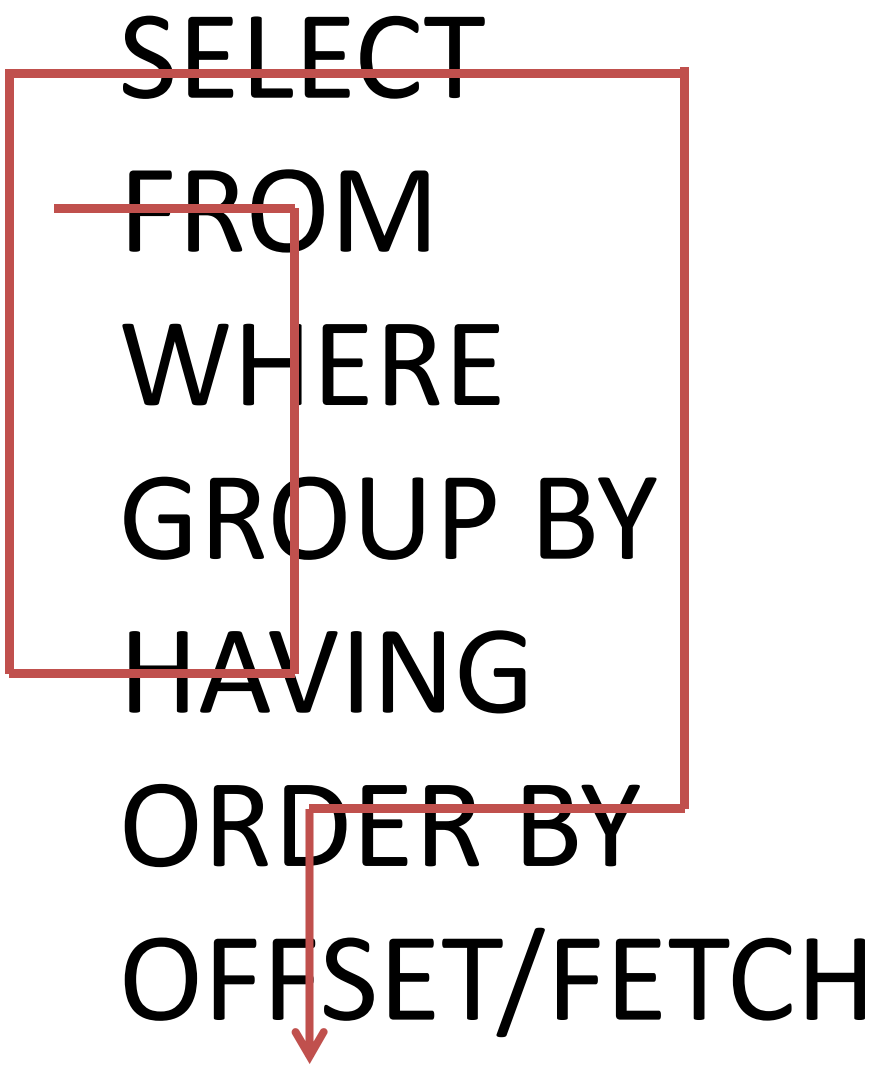
RDD to DataFrame

```
pdata.map(lambda x: (x.dept, [x.age, 1])) \
  .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
  .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
  .collect()
```

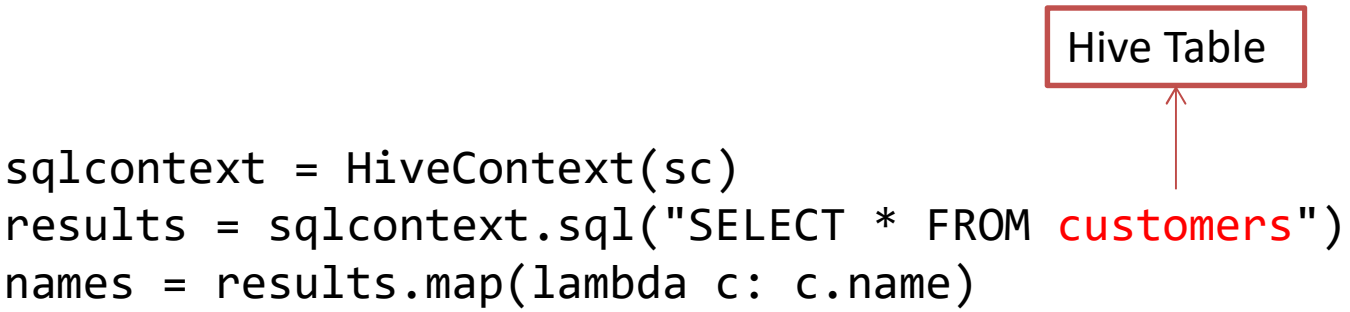
```
data.groupBy("dept").avg("age")
```


資料來源:databricks

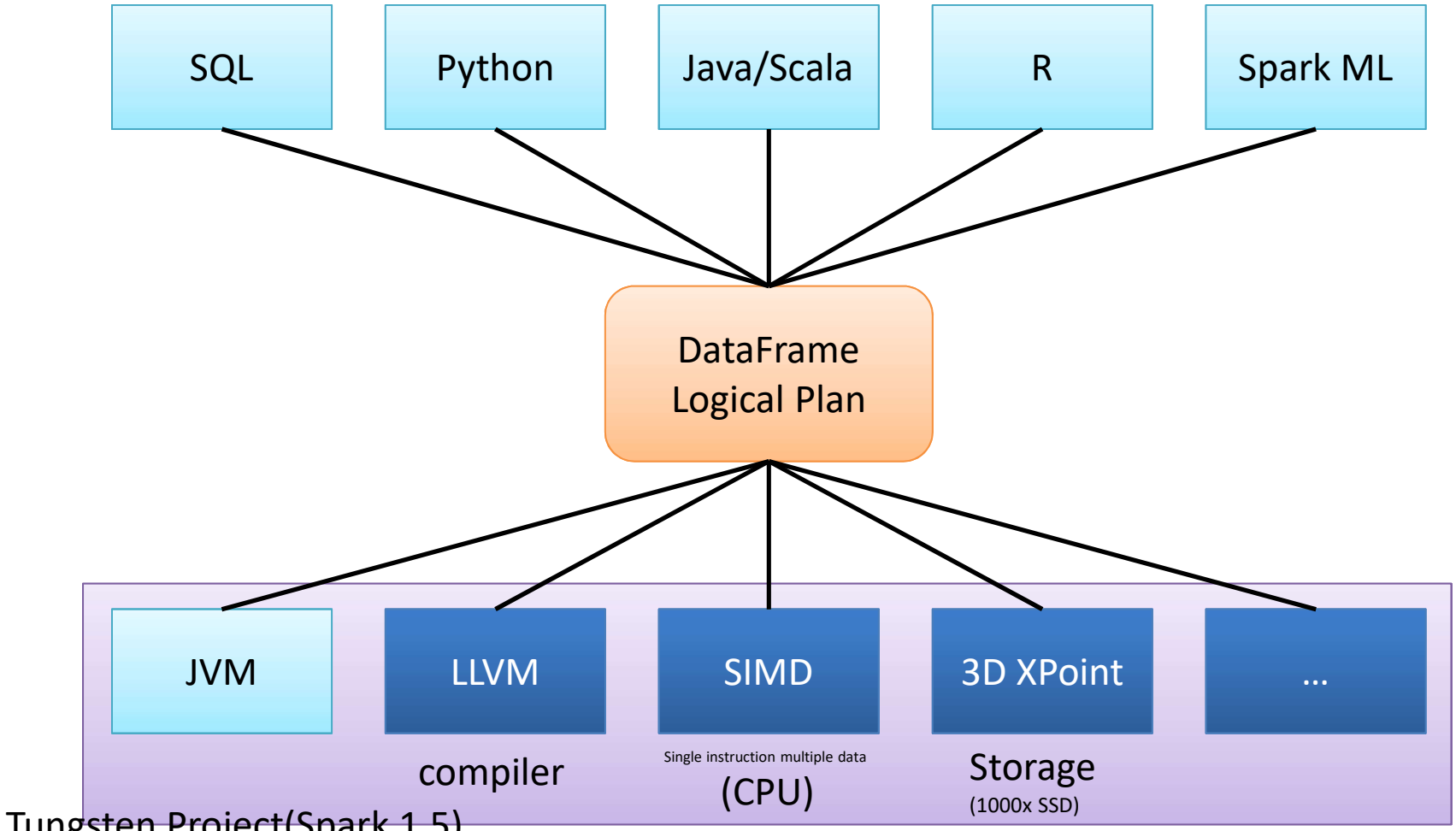

DataFrame
SparkSQL
RDD
Core Spark


SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
OFFSET/FETCH

SQL integrate with Spark



Unified API. One Engine. Automatically Optimized



Tungsten Project(Spark 1.5)

Hardware Trends

	2010	2015	
Storage	50+ MB/s (HDD)	500+ MB/s (SSD)	 10x
Network	1Gbps	10Gbps	 10x
CPU	~3GHz	~3GHz	

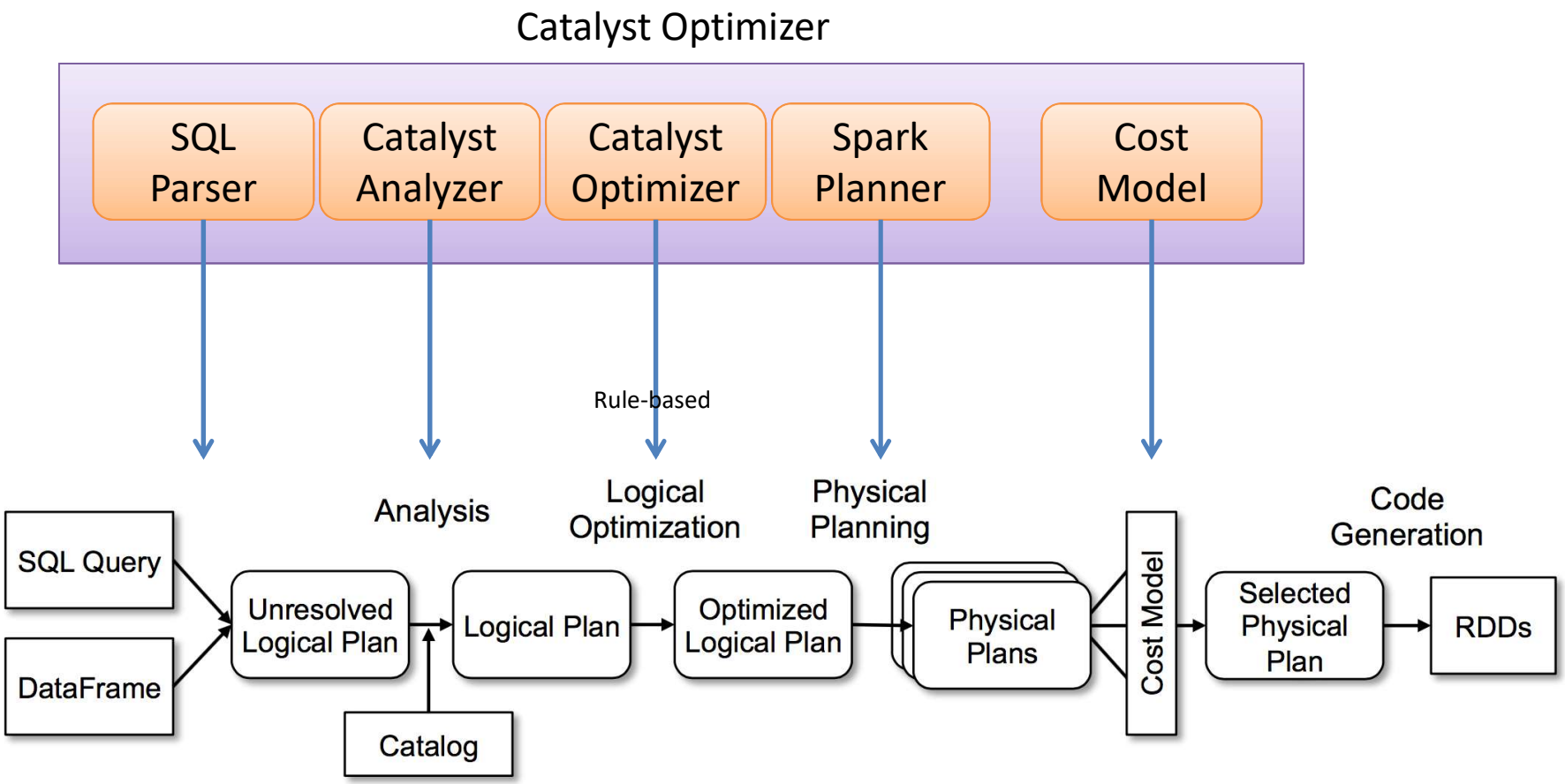
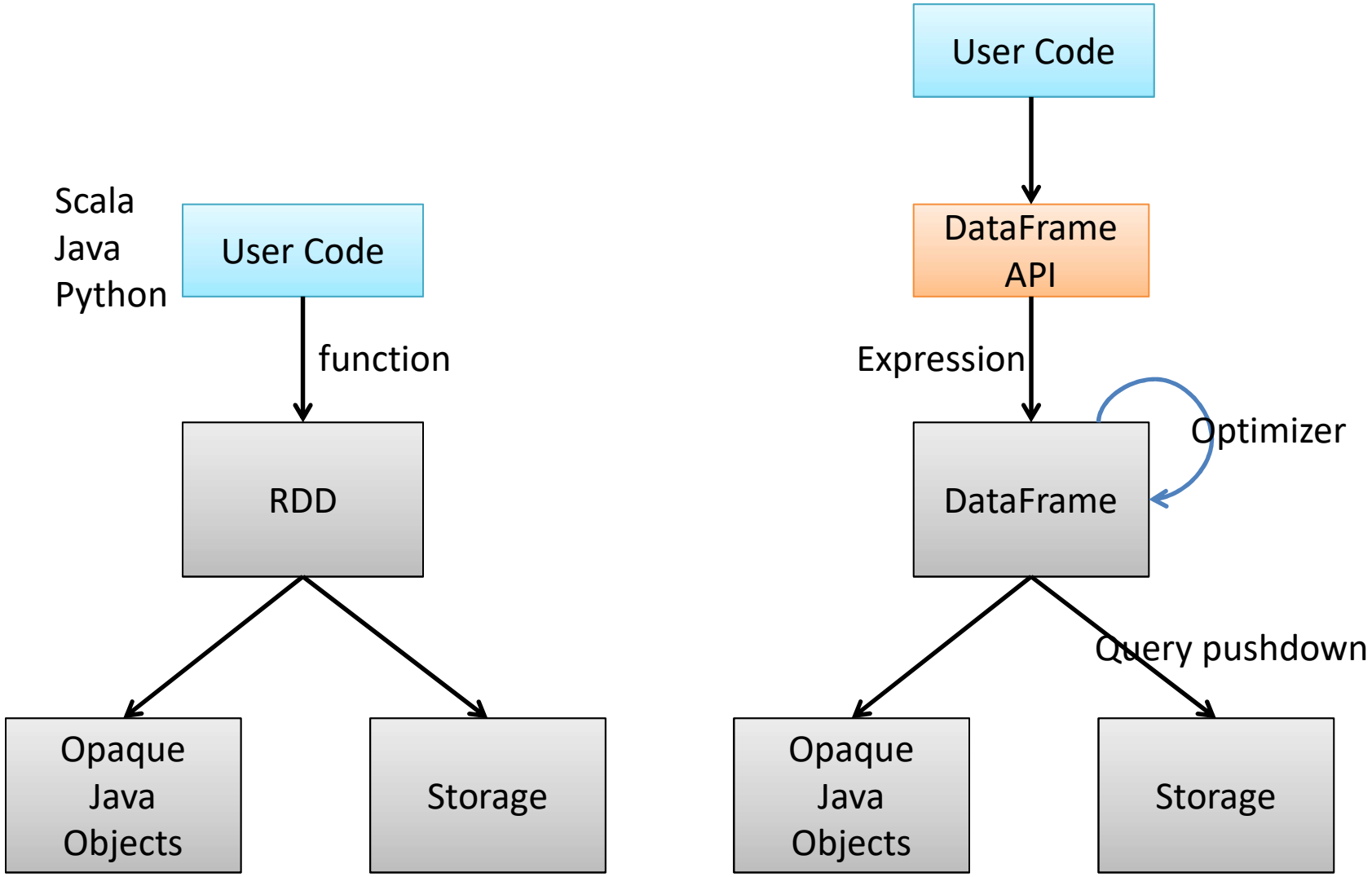


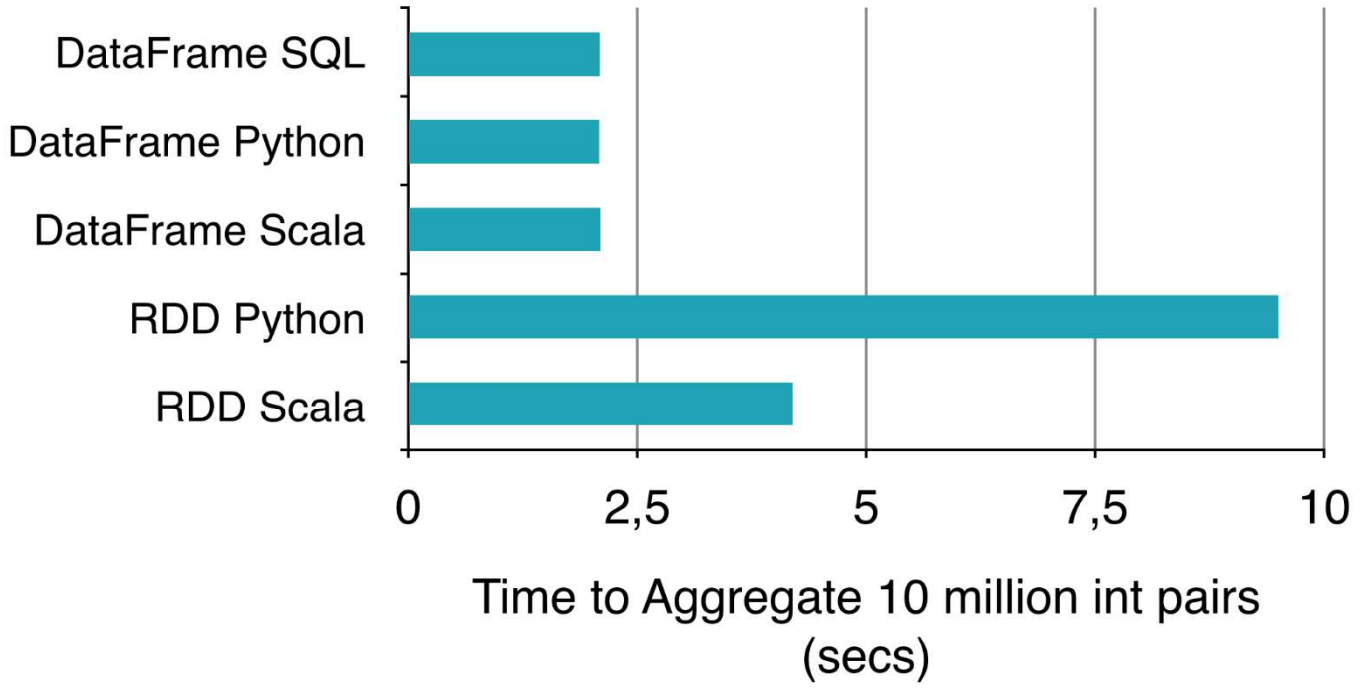
Figure 3: Phases of query planning in Spark SQL. Rounded rectangles represent Catalyst trees.

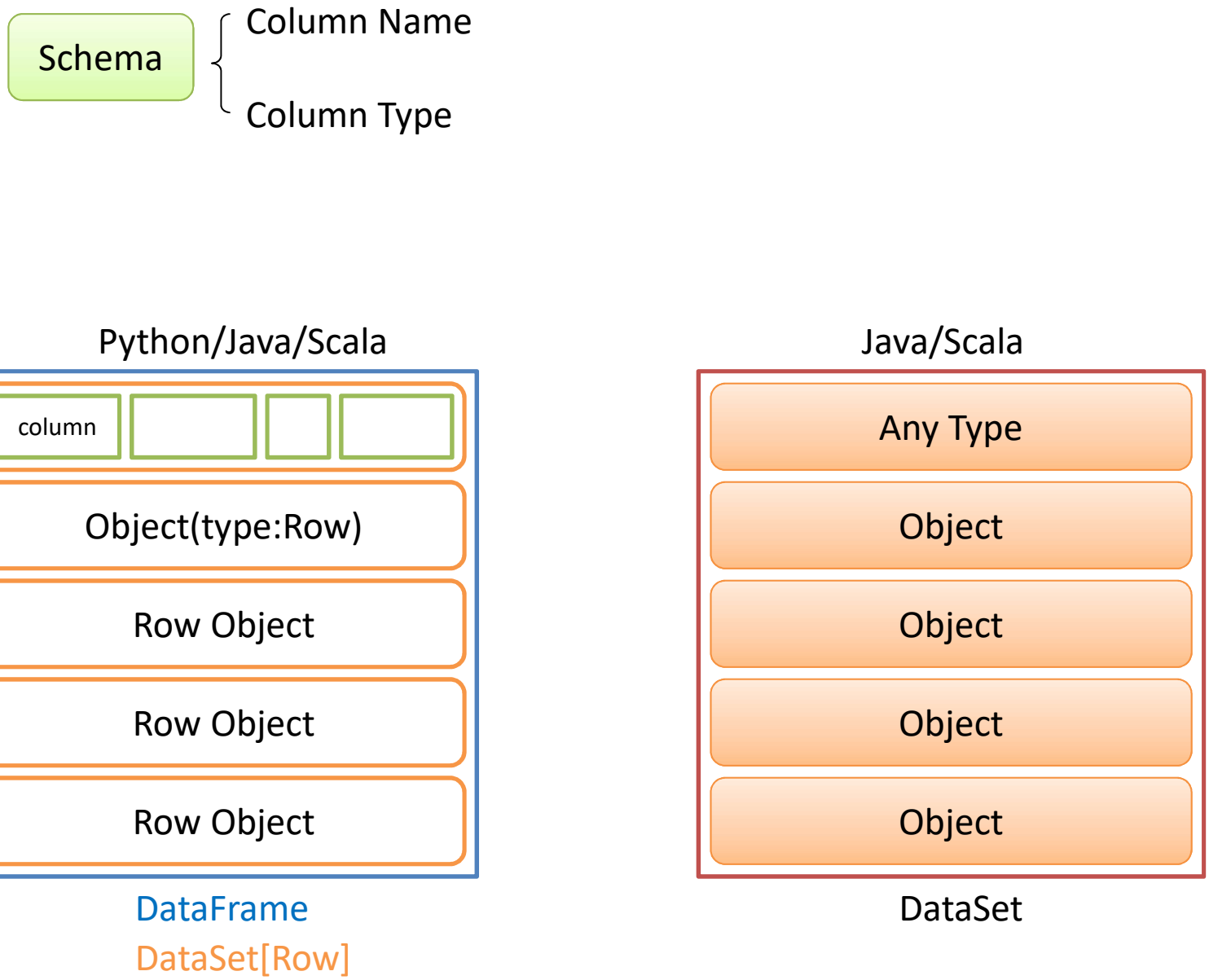
資料來源:Spark SQL: Relational Data Processing in Spark sigmod 2015

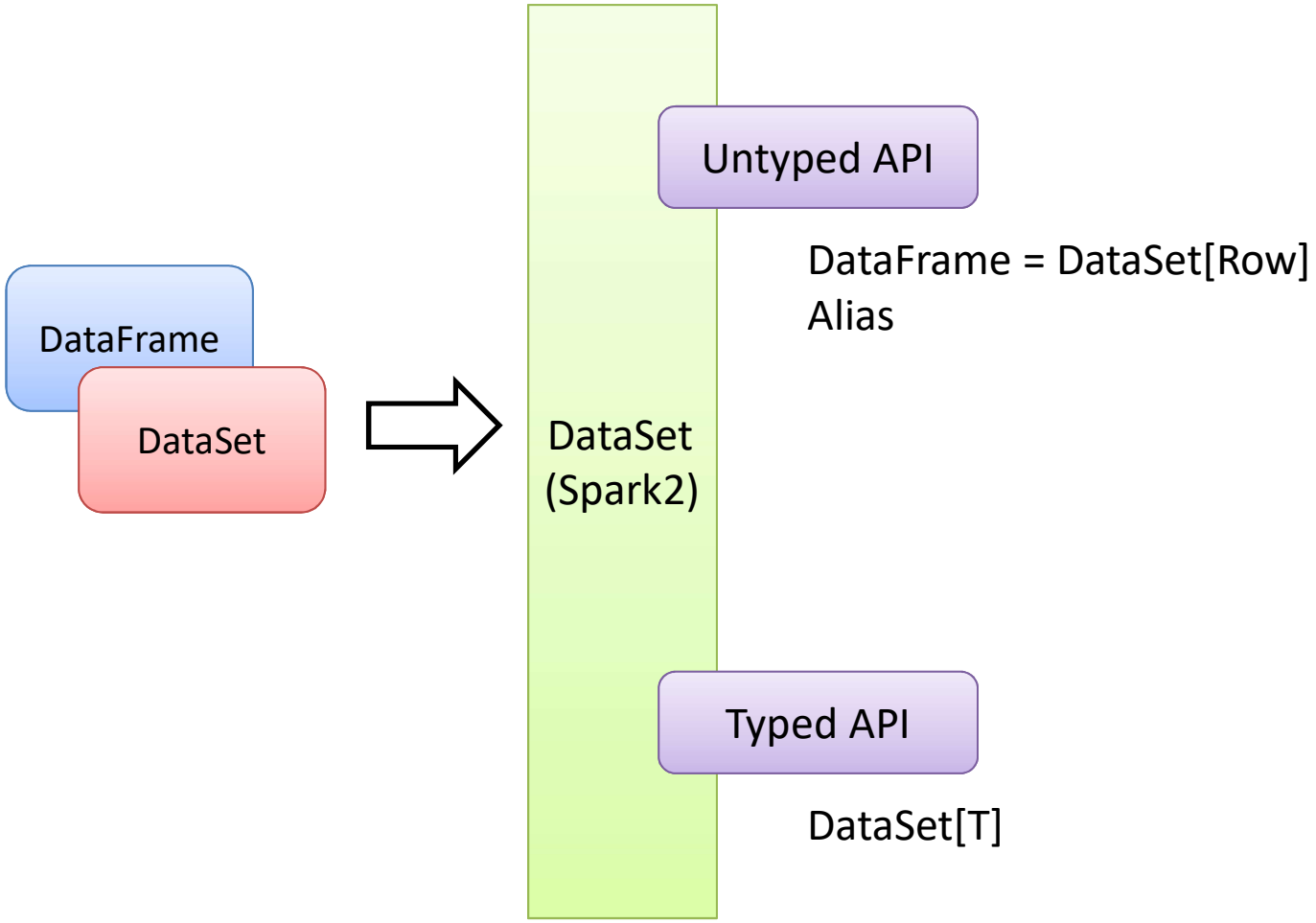

Scala
Java
Python
User Code
function
RDD
Opaque
Java
Objects
Storage
User Code
DataFrame
API
Expression
Optimizer
DataFrame
Query pushdown
Opaque
Java
Objects
Storage

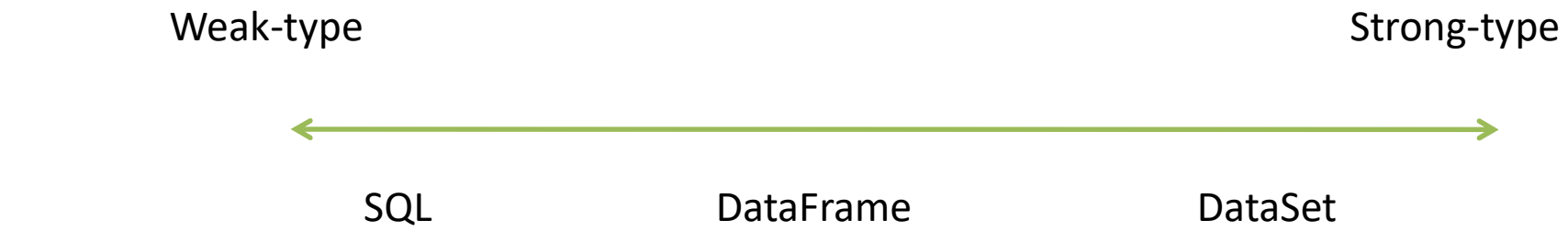
Benefit of Logical Plan

Physical Execution: Unified Across Languages




Schema
Column Name
Column Type
Python/Java/Scala
column
Object(type:Row)
Row Object
Row Object
Row Object
DataFrame
DataSet[Row]
Java/Scala
Any Type
Object
Object
Object
Object
DataSet


DataFrame
DataSet
DataSet
(Spark2)
Untyped API
DataFrame = DataSet[Row]
Alias
Typed API
DataSet[T]

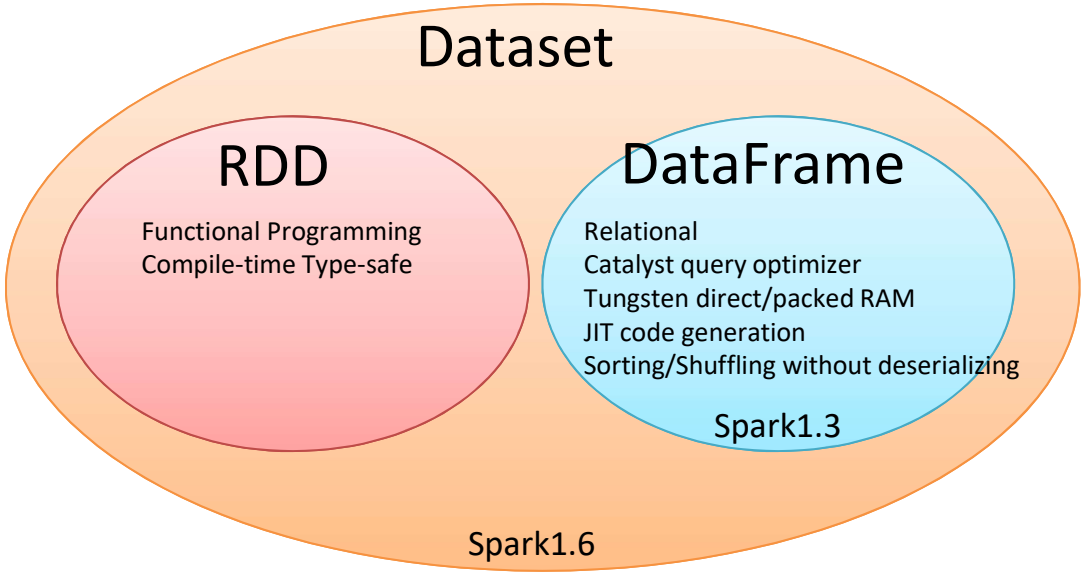


	SQL	DataFrame	DataSet
Syntax Errors	Runtime	Compile Time	Compile Time
Semantic Errors	Runtime	Runtime	Compile Time

DataSet

- Spark 1.6增加DataSet概念
  - 類似RDD
  - 既有Spark SQL的好處，又允許使用者自行定義物件與使用lambda函數
  - 提供編譯階段的型別檢查，與物件導向式API。
  - 可以自由地在DataSet、DataFrame、RDD間轉換。
- Spark 2.0開始，統一DataSet與DataFrame API。
  - DataFrame被定義為Dataset[ROW]

DataSet是什麼?


Dataset
RDD
Functional Programming
Compile-time Type-safe
DataFrame
Relational
Catalyst query optimizer
Tungsten direct/packed RAM
JIT code generation
Sorting/Shuffling without deserializing
Spark1.3
Spark1.6

RDD、DataFrame、DataSet之間的關係

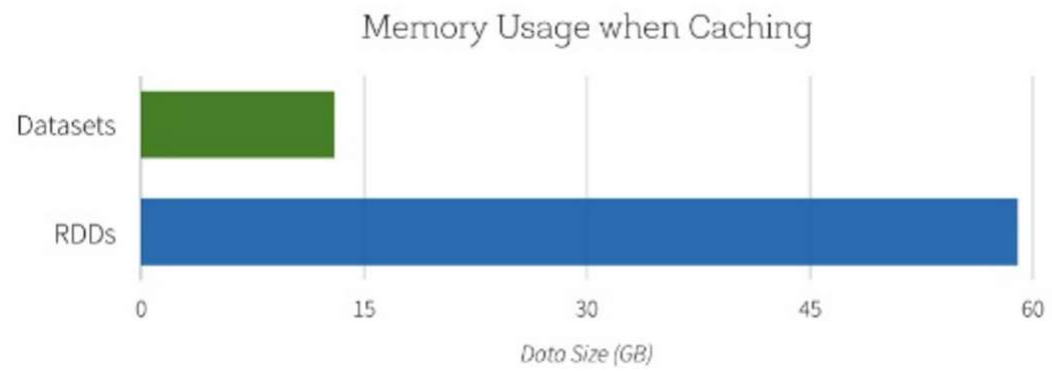
- DataSet是分散式的資料集合。
  - 提供RDD的優點(strong type、能夠使用lambda函數)與Spark SQL執行引擎的優點。
  - DataSet可以由JVM物件建構並使用轉換功能(map、flatMap、filter等等)。
  - DataSet API可以在Scala與Java API使用，但Python不支援靜態型態，所以不能使用。
- DataFrame是由DataSet組成的Row物件。
  - 概念等同關聯式資料庫的表格或R/Python的DataFrame。

Typed API與Untyped API

語言	主要的抽象
Scala	DataSet[T] & DataFrame(alias for Dataset[ROW])
Java	DataSet[T]
Python	DataFrame
R	DataFrame

因為Python與R並沒有編譯階段的型態安全，所以只能使用DataFrame

Space Efficiency



資料來源：A Tale of Three Apache Spark APIs:RDDs vs DataFrames and DataSets

何時該用RDD

- 希望能夠對資料集進行最基本的轉換、處理和控制
- 資料結構為非結構化，例如：media stream或text stream
- 希望使用函數程式設計(functional programming)而不是特定領域表示(domain specific expression)來處理資料
- 不想要使用綱要、行格式(columnar)、以及使用名字或欄位存取資料
- 不想得到利用DataFrame/DataSet操作結構化、半結構化資料所得到的程式優化與效能提升

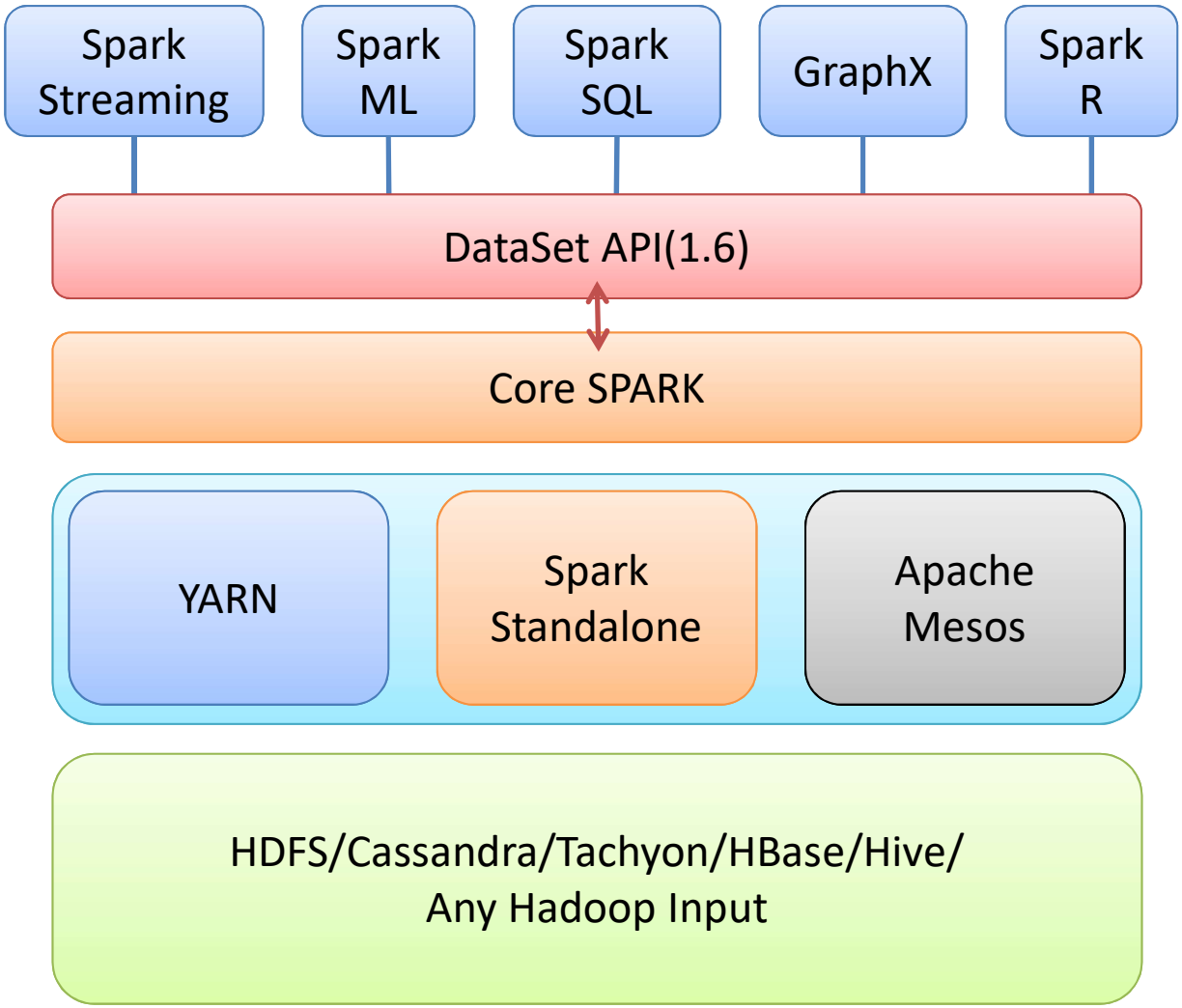
資料來源：A Tale of Three Apache Spark APIs:RDDs vs DataFrames and DataSets

何時該用DataFrame/DataSet

- 需要豐富的語義、高階抽象與特定領域專用的API
- 需要對半結構化資料進行高階處理如：filter、map、aggregation、average、sum、SQL查詢、columar存取或使用lambda函數
- 需要編譯階段的類型安全、typed JVM物件、使用Catalyst優化並利用Tungsten所產生的高效程式碼，則建議使用Dataset
- 需要在不同的Spark程式庫之間使用一致和簡化的API
- 使用R撰寫Spark程式碼，則建議使用DataFrame
- 使用Python撰寫Spark程式碼，一般建議使用DataFrame。但若要進行更精細的操作，則可以用RDD。

資料來源：A Tale of Three Apache Spark APIs:RDDs vs DataFrames and DataSets

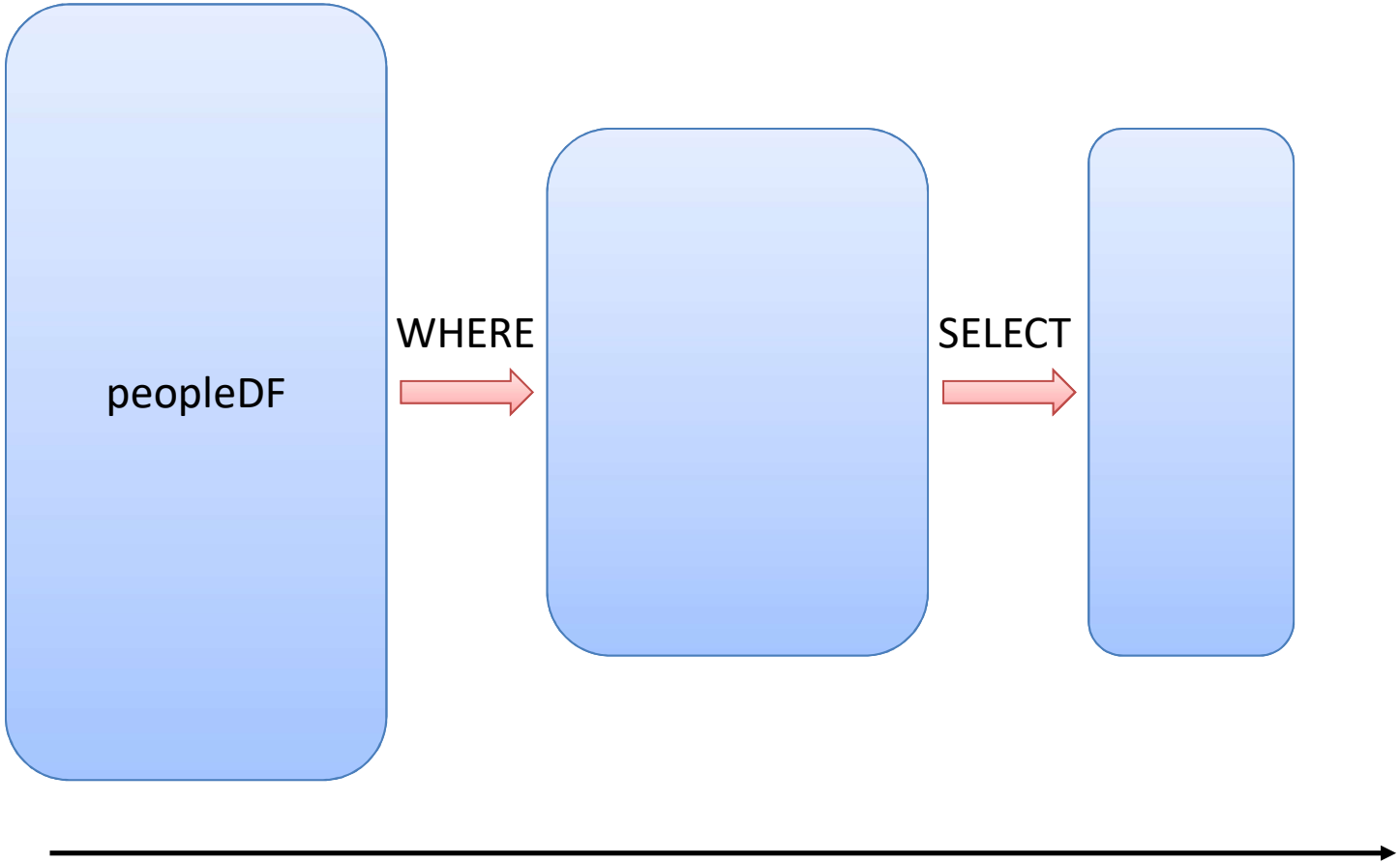
DataSet is future of Spark 1.6



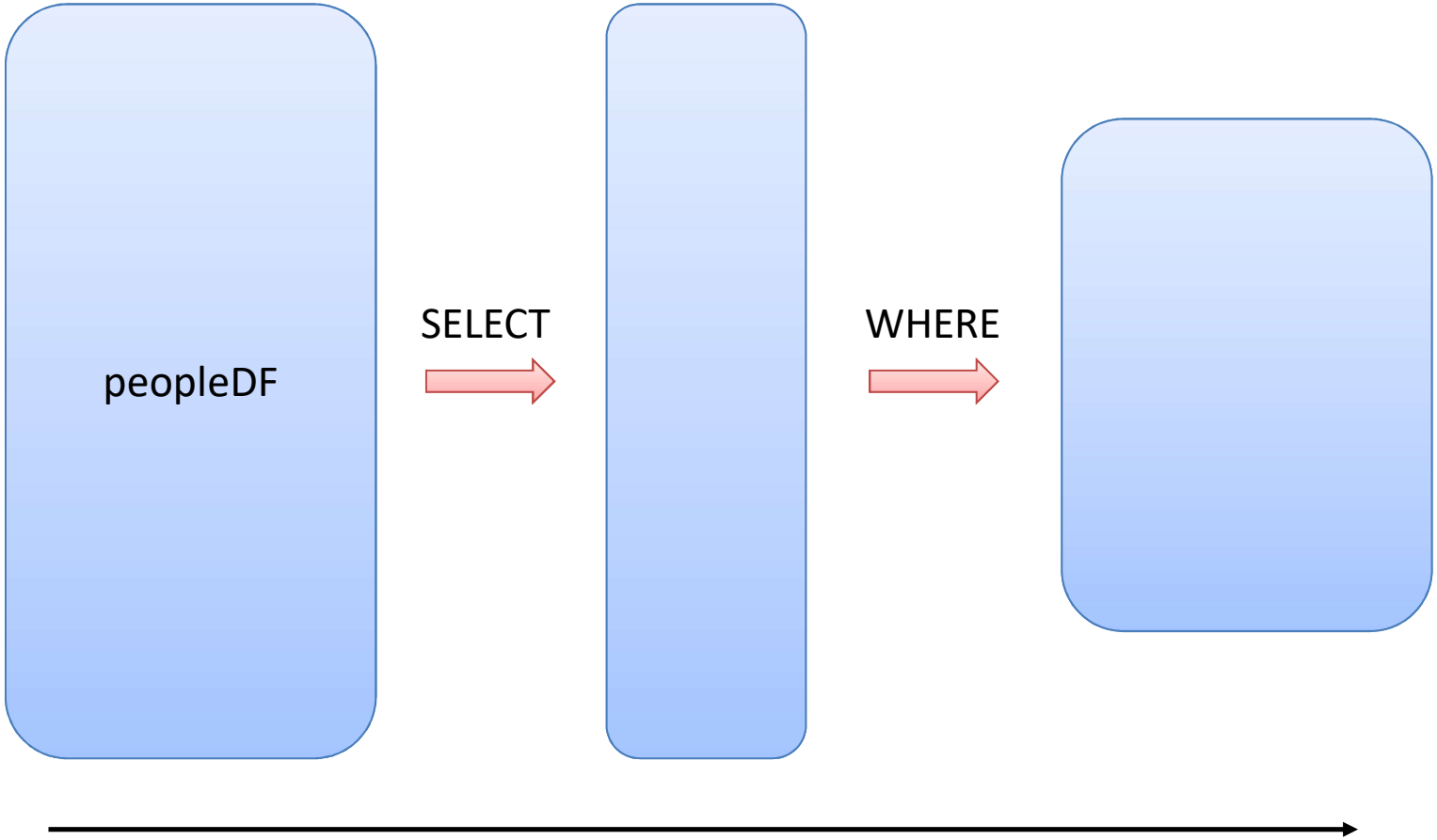
SPJ

- SELECTION
  - WHERE:過濾rows
- PROJECTION
  - SELECT:擷取columns
- JOIN
  - 結合兩個DataFrame

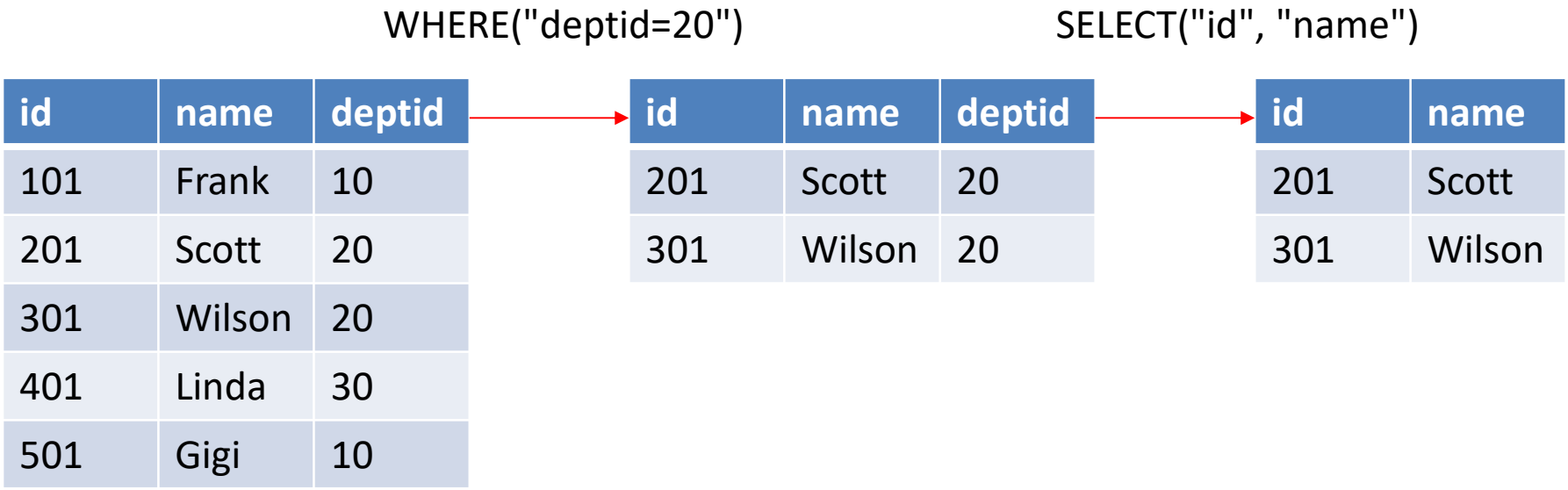
Transformation order



Transformation order



df.where(“deptid=20”).select(“id”,”name”)



id	name	deptid
101	Frank	10
201	Scott	20
301	Wilson	20
401	Linda	30
501	Gigi	10

id	name	deptid
201	Scott	20
301	Wilson	20

id	name
201	Scott
301	Wilson

df1.join(df2,join_conds,join_type)

- INNER JOIN
- OUTER JOIN
  - `LEFT OUTER JOIN`
  - `RIGHT OUTER JOIN`
  - `FULL OUTER JOIN`
- CROSS JOIN

JOIN_TYPEs

```
'inner', 'outer', 'full', 'fullouter', 'full_outer', 'leftouter', 'left',
'left_outer', 'rightouter', 'right', 'right_outer', 'leftsemi',
'left_semi', 'leftanti', 'left_anti', 'cross'
```

empDF,deptDF

```
>>> empDF.show()
+-----+------+---------+----+---------+----+----+------+
|EMPNO| ENAME|      JOB| MGR| HIREDATE| SAL|COMM|DEPTNO|
+-----+------+---------+----+---------+----+----+------+
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|
| 7901| FRANK|  TRAINER|7839|23-JAN-85|3300|null|    50|
+-----+------+---------+----+---------+----+----+------+

>>> deptDF.show()
+------+----------+--------+
|deptno|     dname|     loc|
+------+----------+--------+
|    10|ACCOUNTING|NEW YORK|
|    20|  RESEARCH|  DALLAS|
|    30|     SALES| CHICAGO|
|    40|OPERATIONS|  BOSTON|
+------+----------+--------+
```

INNER JOIN(equality join)

```
>>> empDF.join(deptDF,empDF.deptno==deptDF.deptno,'inner').show()
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
|empno|  name|      job| mgr| hiredate| sal|comm|deptno|deptno|     dname|     loc|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    10|ACCOUNTING|NEW YORK|
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    30|     SALES| CHICAGO|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    10|ACCOUNTING|NEW YORK|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    20|  RESEARCH|  DALLAS|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|    20|  RESEARCH|  DALLAS|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|    30|     SALES| CHICAGO|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|    30|     SALES| CHICAGO|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|    30|     SALES| CHICAGO|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|    30|     SALES| CHICAGO|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|    20| RESEARCH'|  DALLAS|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|    30|     SALES| CHICAGO|
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|    10|ACCOUNTING|NEW YORK|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
```

INNER JOIN(non-equality join)

```
>>> empDF.join(deptDF,empDF.deptno>deptDF.deptno,'inner').show()
+-----+------+--------+----+---------+----+----+------+------+----------+--------+
|empno|  name|     job| mgr| hiredate| sal|comm|deptno|deptno|     dname|     loc|
+-----+------+--------+----+---------+----+----+------+------+----------+--------+
| 7698| BLAKE| MANAGER|7839|01-MAY-81|2850|null|    30|    10|ACCOUNTING|NEW YORK|
| 7698| BLAKE| MANAGER|7839|01-MAY-81|2850|null|    30|    20|  RESEARCH|  DALLAS|
| 7566| JONES| MANAGER|7839|02-APR-81|2975|null|    20|    10|ACCOUNTING|NEW YORK|
| 7788| SCOTT| ANALYST|7566|19-APR-87|3000|null|    20|    10|ACCOUNTING|NEW YORK|
| 7902|  FORD| ANALYST|7566|03-DEC-81|3000|null|    20|    10|ACCOUNTING|NEW YORK|
| 7369| SMITH|   CLERK|7902|17-DEC-80| 800|null|    20|    10|ACCOUNTING|NEW YORK|
| 7499| ALLEN|SALESMAN|7698|20-FEB-81|1600| 300|    30|    10|ACCOUNTING|NEW YORK|
| 7499| ALLEN|SALESMAN|7698|20-FEB-81|1600| 300|    30|    20|  RESEARCH|  DALLAS|
| 7521|  WARD|SALESMAN|7698|22-FEB-81|1250| 500|    30|    10|ACCOUNTING|NEW YORK|
| 7521|  WARD|SALESMAN|7698|22-FEB-81|1250| 500|    30|    20|  RESEARCH|  DALLAS|
| 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|    30|    10|ACCOUNTING|NEW YORK|
| 7654|MARTIN|SALESMAN|7698|28-SEP-81|1250|1400|    30|    20|  RESEARCH|  DALLAS|
| 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|   0|    30|    10|ACCOUNTING|NEW YORK|
| 7844|TURNER|SALESMAN|7698|08-SEP-81|1500|   0|    30|    20|  RESEARCH|  DALLAS|
| 7876| ADAMS|   CLERK|7788|23-MAY-87|1100|null|    20|    10|ACCOUNTING|NEW YORK|
| 7900| JAMES|   CLERK|7698|03-DEC-81| 950|null|    30|    10|ACCOUNTING|NEW YORK|
| 7900| JAMES|   CLERK|7698|03-DEC-81| 950|null|    30|    20|  RESEARCH|  DALLAS|
+-----+------+--------+----+---------+----+----+------+------+----------+--------+
```

LEFT OUTER JOIN

```
>>> empDF.join(deptDF,empDF.deptno==deptDF.deptno,'left_outer').show()
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
|empno|  name|      job| mgr| hiredate| sal|comm|deptno|deptno|     dname|     loc|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    10|ACCOUNTING|NEW YORK|
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    30|     SALES| CHICAGO|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    10|ACCOUNTING|NEW YORK|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    20| RESEARCH'|  DALLAS|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    20| RESEARCH'|  DALLAS|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|    20| RESEARCH'|  DALLAS|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|    20| RESEARCH'|  DALLAS|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|    30|     SALES| CHICAGO|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|    30|     SALES| CHICAGO|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|    30|     SALES| CHICAGO|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|    30|     SALES| CHICAGO|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|    20| RESEARCH'|  DALLAS|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|    30|     SALES| CHICAGO|
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|    10|ACCOUNTING|NEW YORK|
| 7901| FRANK|  TRAINER|7839|23-JAN-85|3300|null|    50|  null|      null|    null|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
```

RIGHT OUTER JOIN

```
>>> empDF.join(deptDF,empDF.deptno==deptDF.deptno,'right_outer').show()
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
|empno|  name|      job| mgr| hiredate| sal|comm|deptno|deptno|     dname|     loc|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|    10|ACCOUNTING|NEW YORK|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    10|ACCOUNTING|NEW YORK|
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    10|ACCOUNTING|NEW YORK|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|    20|  RESEARCH|  DALLAS|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|    20|  RESEARCH|  DALLAS|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    20|  RESEARCH|  DALLAS|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|    30|     SALES| CHICAGO|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|    30|     SALES| CHICAGO|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|    30|     SALES| CHICAGO|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|    30|     SALES| CHICAGO|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|    30|     SALES| CHICAGO|
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    30|     SALES| CHICAGO|
| null|  null|     null|null|     null|null|null|  null|    40|OPERATIONS|  BOSTON|
+-----+------+---------+----+---------+----+----+------+------+----------+--------+
```

FULL OUTER JOIN

```
>>> empDF.join(deptDF,empDF.deptno==deptDF.deptno,'full_outer').show()
+-----+------+---------+----+---------+----+----+------+------+----------+---------+
|empno|  name|      job| mgr| hiredate| sal|comm|deptno|deptno|     dname|      loc|
+-----+------+---------+----+---------+----+----+------+------+----------+---------+
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    30|     SALES| CHICAGO|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|    30|     SALES| CHICAGO|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|    30|     SALES| CHICAGO|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|    30|     SALES| CHICAGO|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|    30|     SALES| CHICAGO|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|    30|     SALES| CHICAGO|
| null|  null|     null|null|     null|null|null|  null|    40|OPERATIONS|  BOSTON|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    20|  RESEARCH|  DALLAS|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|    20|  RESEARCH|  DALLAS|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|    20|  RESEARCH|  DALLAS|
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    10|ACCOUNTING|NEW YORK|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    10|ACCOUNTING|NEW YORK|
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|    10|ACCOUNTING|NEW YORK|
| 7901| FRANK|  TRAINER|7839|23-JAN-85|3300|null|    50|  null|      null|    null|
+-----+------+---------+----+---------+----+----+------+------+----------+---------+
```

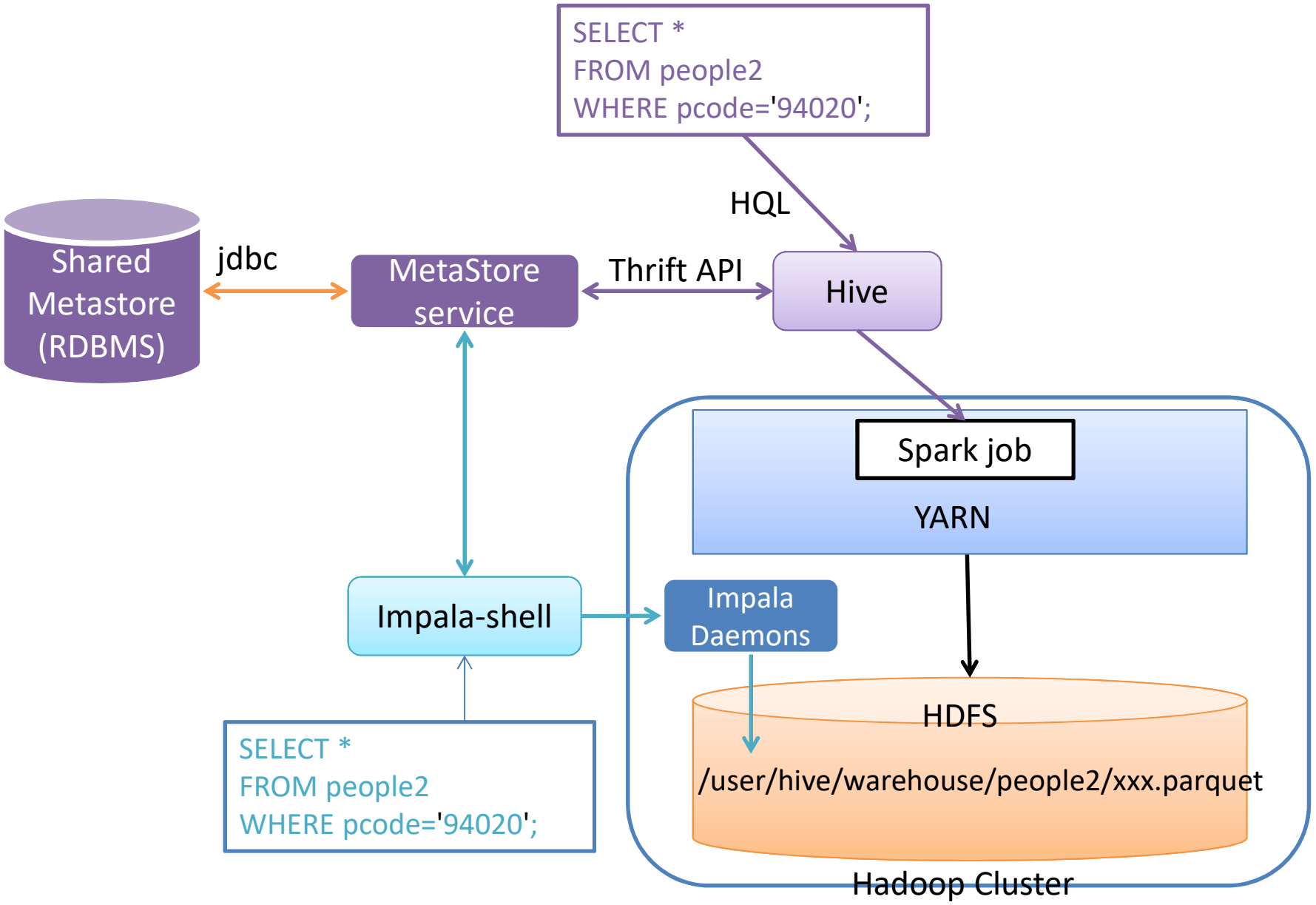
CROSS JOIN

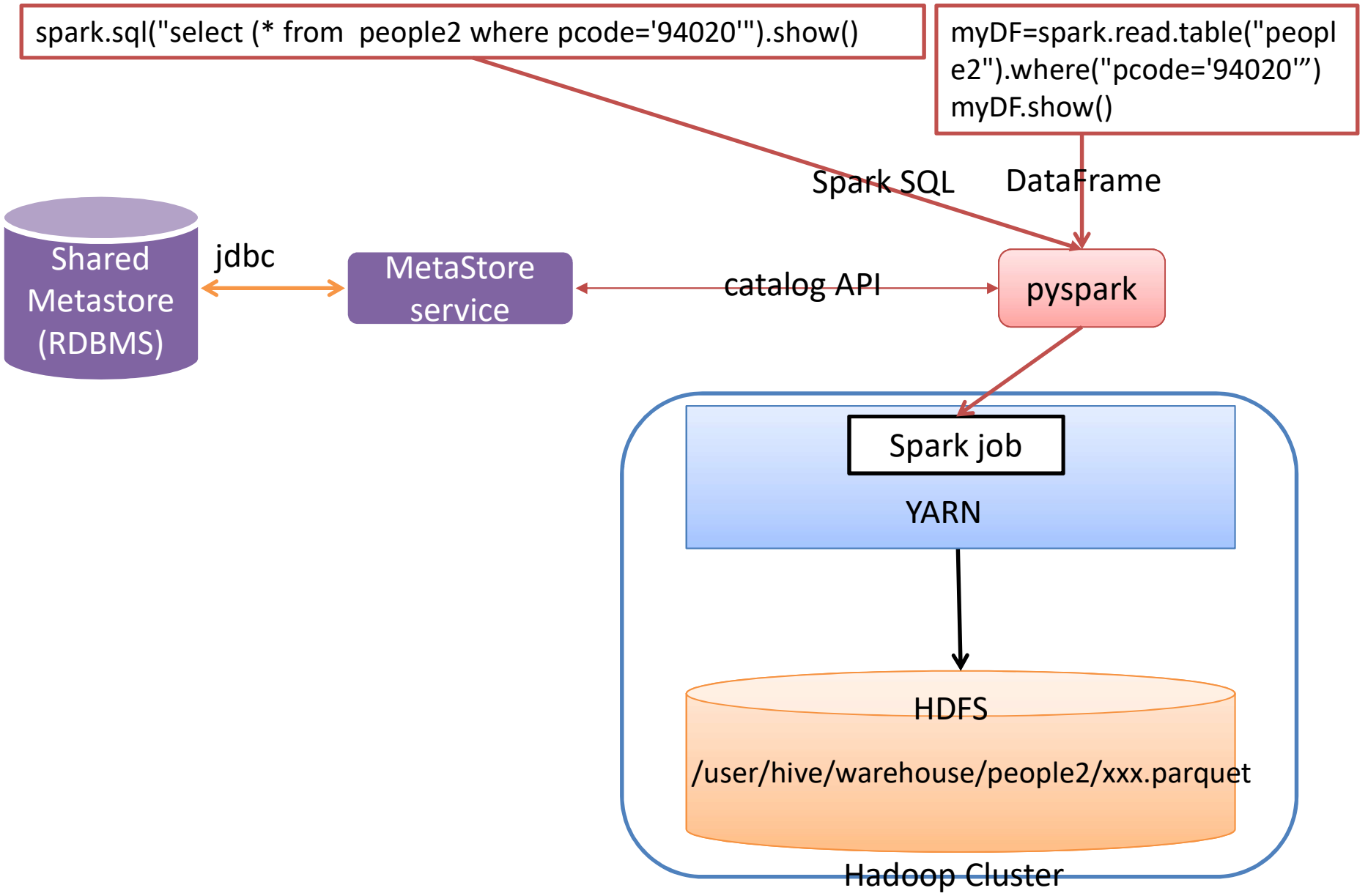
```
>>> empDF.crossJoin(deptDF).show())
+-----+-----+---------+----+---------+----+----+------+------+----------+--------+
|empno| name|      job| mgr| hiredate| sal|comm|deptno|deptno|     dname|     loc|
+-----+-----+---------+----+---------+----+----+------+------+----------+--------+
| 7839| KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    10|ACCOUNTING|NEW YORK|
| 7839| KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    20|  RESEARCH|  DALLAS|
| 7839| KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    30|     SALES| CHICAGO|
| 7839| KING|PRESIDENT|null|17-NOV-81|5000|null|    10|    40|OPERATIONS|  BOSTON|
| 7698|BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    10|ACCOUNTING|NEW YORK|
| 7698|BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    20|  RESEARCH|  DALLAS|
| 7698|BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    30|     SALES| CHICAGO|
| 7698|BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|    40|OPERATIONS|  BOSTON|
| 7782|CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    10|ACCOUNTING|NEW YORK|
| 7782|CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    20|  RESEARCH|  DALLAS|
| 7782|CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    30|     SALES| CHICAGO|
| 7782|CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|    40|OPERATIONS|  BOSTON|
| 7566|JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    10|ACCOUNTING|NEW YORK|
| 7566|JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    20|  RESEARCH|  DALLAS|
| 7566|JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    30|     SALES| CHICAGO|
| 7566|JONES|  MANAGER|7839|02-APR-81|2975|null|    20|    40|OPERATIONS|  BOSTON|
| 7788|SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    10|ACCOUNTING|NEW YORK|
| 7788|SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    20|  RESEARCH|  DALLAS|
| 7788|SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    30|     SALES| CHICAGO|
| 7788|SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|    40|OPERATIONS|  BOSTON|
+-----+-----+---------+----+---------+----+----+------+------+----------+--------+
only showing top 20 rows
```

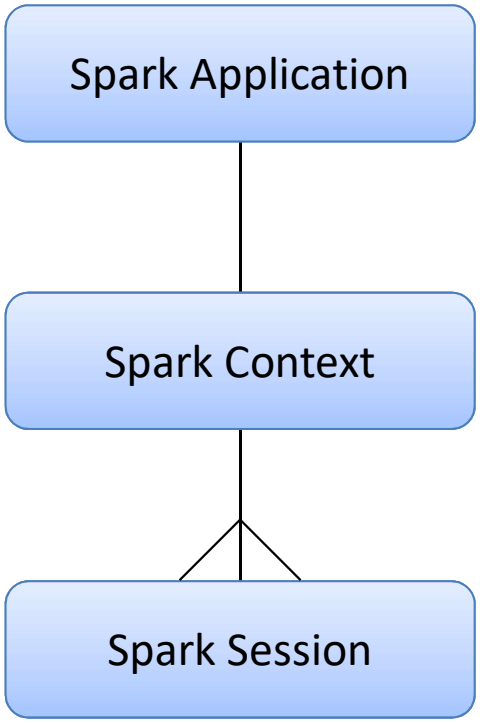
LEFF-SEMI JOIN

```
>>> empDF.join(deptDF,empDF.deptno==deptDF.deptno,'left_semi').show()
+-----+------+---------+----+---------+----+----+------+
|empno|  name|      job| mgr| hiredate| sal|comm|deptno|
+-----+------+---------+----+---------+----+----+------+
| 7839|  KING|PRESIDENT|null|17-NOV-81|5000|null|    10|
| 7698| BLAKE|  MANAGER|7839|01-MAY-81|2850|null|    30|
| 7782| CLARK|  MANAGER|7839|09-JUN-81|2450|null|    10|
| 7566| JONES|  MANAGER|7839|02-APR-81|2975|null|    20|
| 7788| SCOTT|  ANALYST|7566|19-APR-87|3000|null|    20|
| 7902|  FORD|  ANALYST|7566|03-DEC-81|3000|null|    20|
| 7369| SMITH|    CLERK|7902|17-DEC-80| 800|null|    20|
| 7499| ALLEN| SALESMAN|7698|20-FEB-81|1600| 300|    30|
| 7521|  WARD| SALESMAN|7698|22-FEB-81|1250| 500|    30|
| 7654|MARTIN| SALESMAN|7698|28-SEP-81|1250|1400|    30|
| 7844|TURNER| SALESMAN|7698|08-SEP-81|1500|   0|    30|
| 7876| ADAMS|    CLERK|7788|23-MAY-87|1100|null|    20|
| 7900| JAMES|    CLERK|7698|03-DEC-81| 950|null|    30|
| 7934|MILLER|    CLERK|7782|23-JAN-82|1300|null|    10|
+-----+------+---------+----+---------+----+----+------+
```

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT


SELECT *
FROM people2
WHERE pcode='94020';
HQL
Shared Metastore (RDBMS)
jdbc
MetaStore service
Thrift API
Hive
Spark job
YARN
Impala-shell
Impala Daemons
HDFS
/user/hive/warehouse/people2/xxx.parquet
SELECT *
FROM people2
WHERE pcode='94020';
Hadoop Cluster


spark.sql("select (* from people2 where pcode='94020'").show()
myDF=spark.read.table("people2").where("pcode='94020'")
myDF.show()
Spark SQL
DataFrame
Shared Metastore (RDBMS)
jdbc
MetaStore service
catalog API
pyspark
Spark job
YARN
HDFS
/user/hive/warehouse/people2/xxx.parquet
Hadoop Cluster


Spark Application
Spark Context
Spark Session

Application Deployment mode

- --master yarn
  - --deploy-mode cluster
  - --deploy-mode client
- --master meson
- --master spark
- --master local

RDD(Resilient Distributed Dataset)

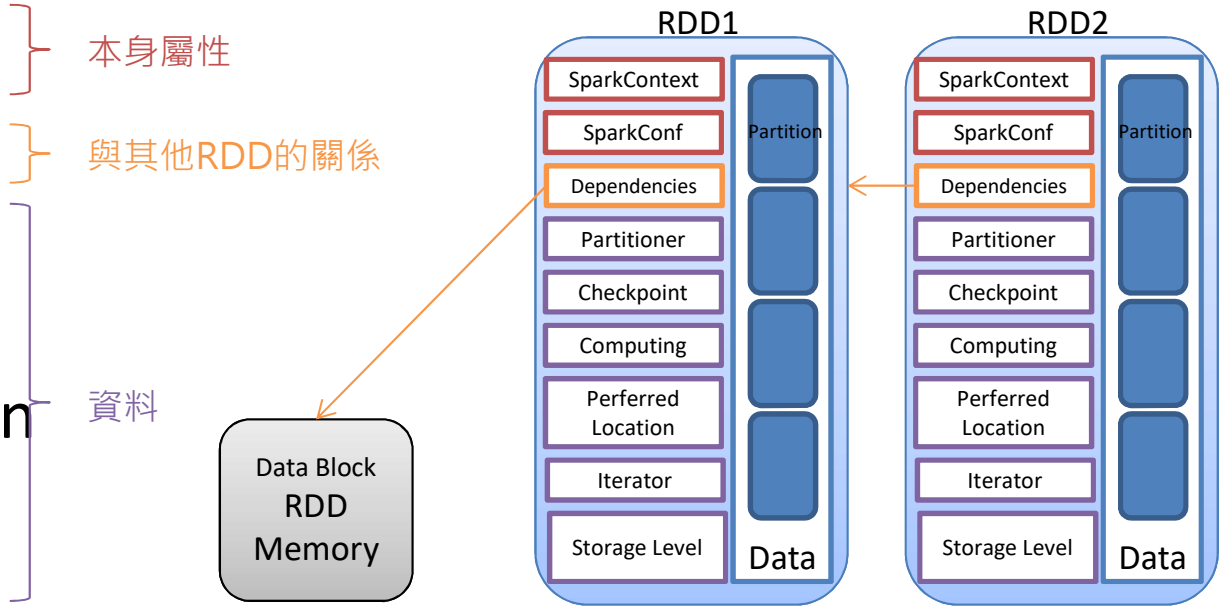
- 定義
  - 由可容錯元件所組成的集合，這個集合可以被平行操作。
    - RDDs are fault-tolerate, parallel data structures.
  - 是一個唯讀、被分區的紀錄集合。
    - An RDD is a read-only, partitioned collection of records.

”Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing” 2012

- Resilient
  - 有彈性的，可彈回。
  - 表示發生錯誤後，透過lineage可以重新產生RDD。
- Distributed
  - RDD由partition組成，而partition被分散到多個節點上執行。
- Dataset
  - 不是真正的“集合”，而是Spark處理資料的邏輯。
  - 資料放在partition裡。

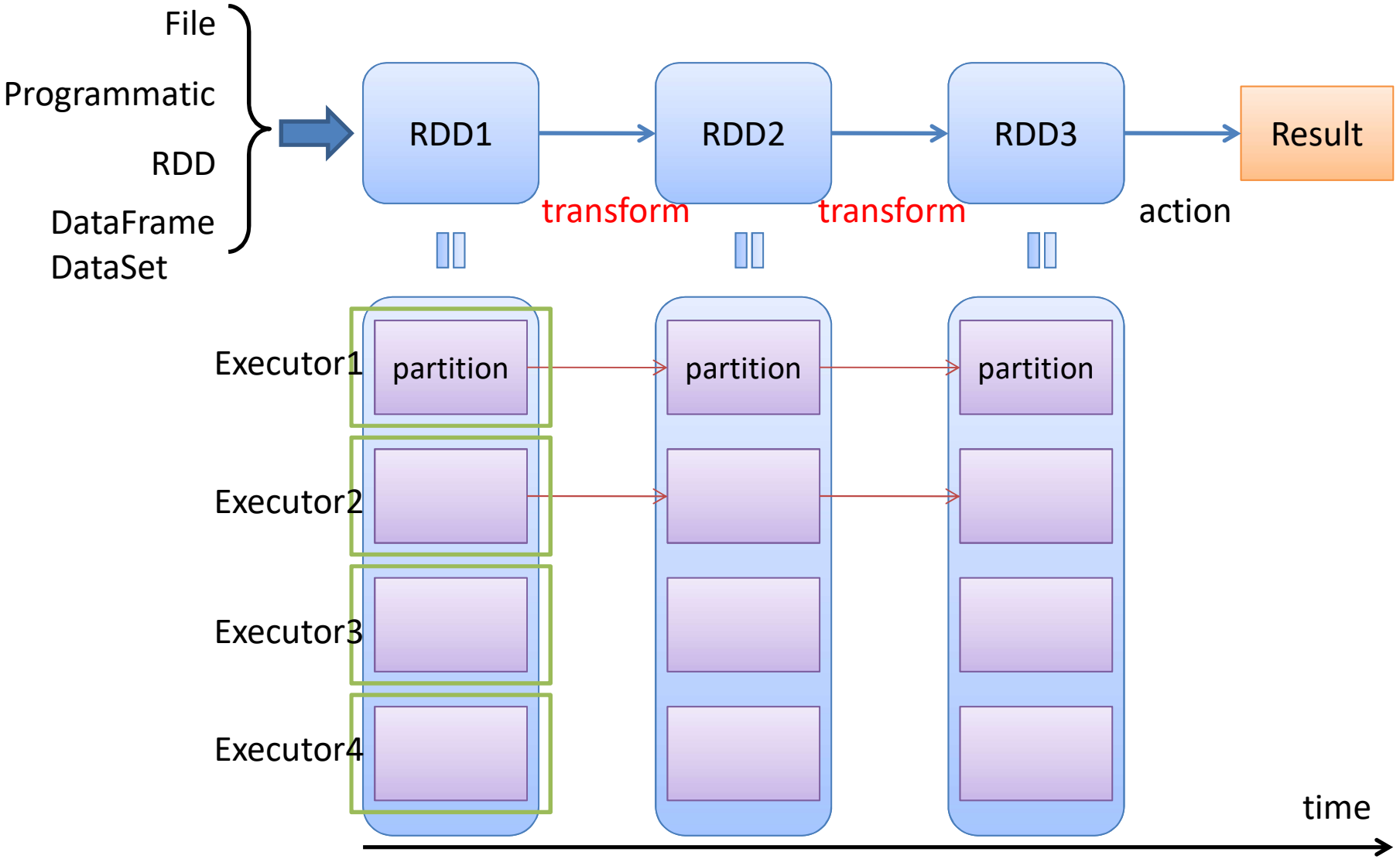
RDD的內部組成元件

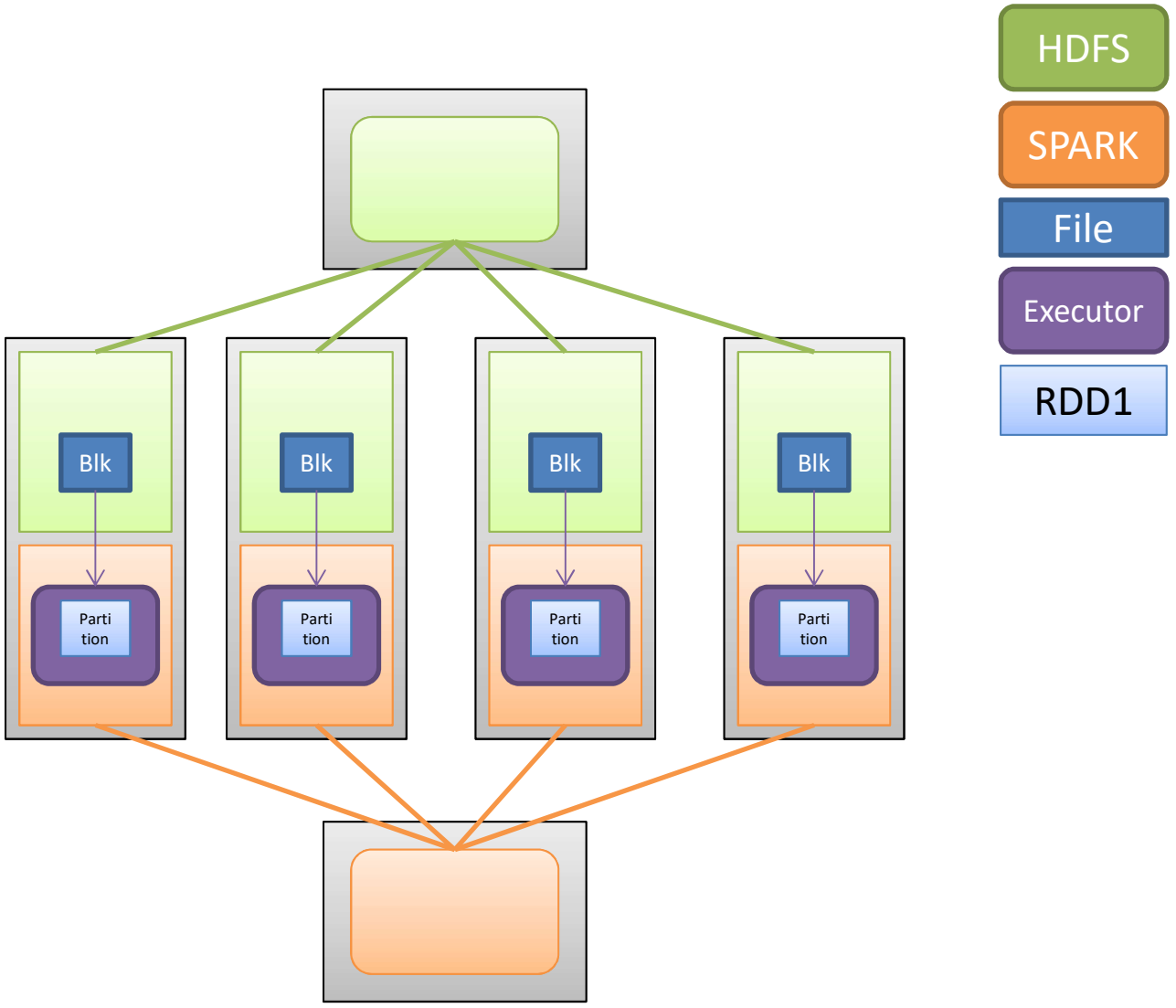
- SparkContext
- SparkConf
- Dependencies
- Partitioner
- Checkpoint
- Computing
- Preferred Location
- Iterator
- Storage Level
- Data



lineage

Resilient Distributed Dataset



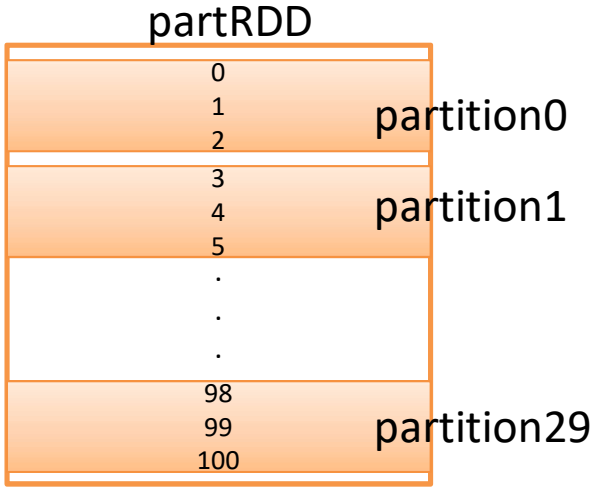

HDFS
SPARK
File
Executor
RDD1
Blk
Blk
Blk
Blk
Parti tion
Parti tion
Parti tion
Parti tion

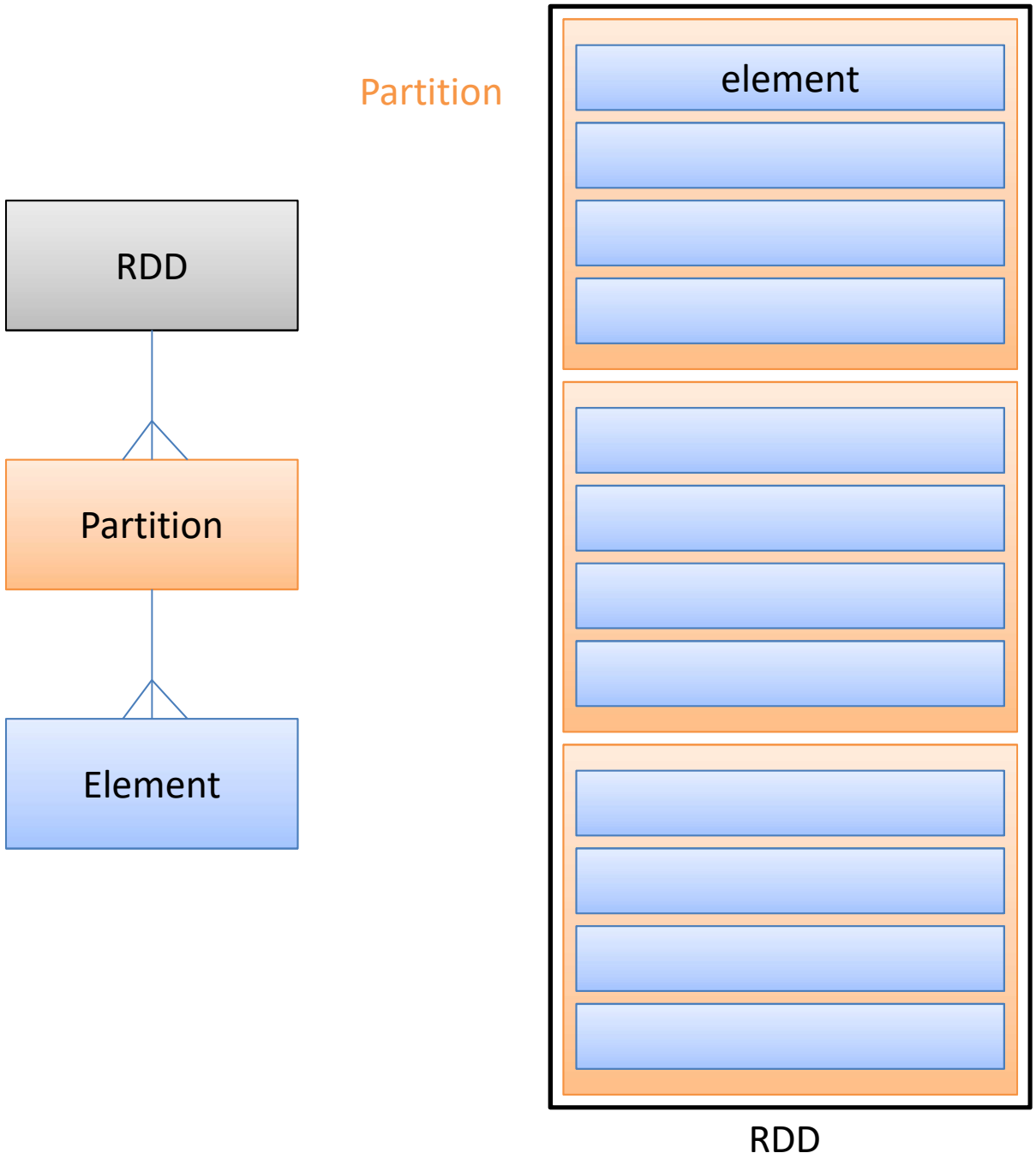
Partition

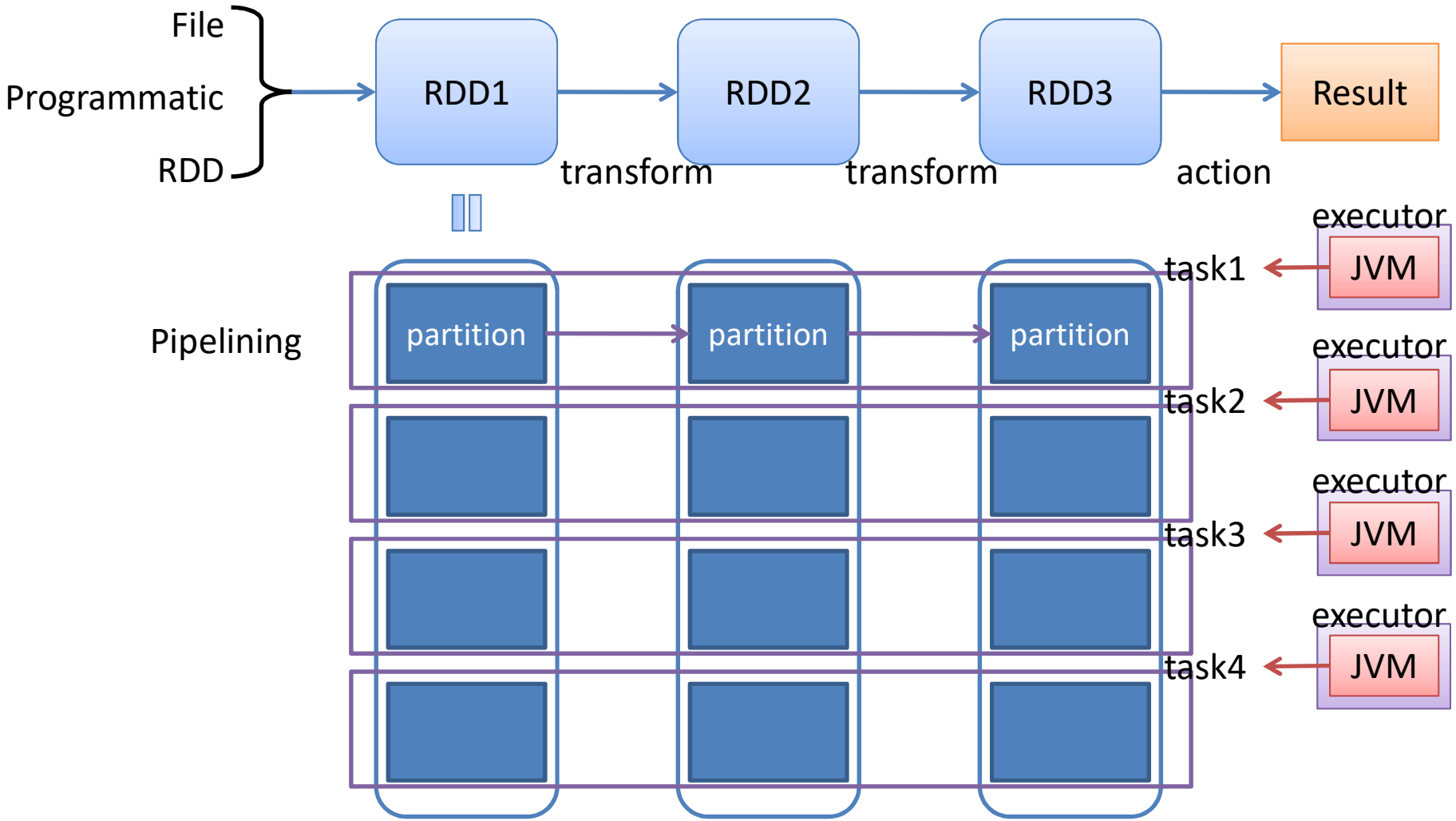
- 產生RDD時，明確指定
  sc.textFile(“inputPath”,minPartitions)
- 若沒有明確指定partition數量，則由RDD生成方式決定預設partition數量
  - 由collection方式產生：
    - 為此application所分配到的CPU數量(Executors)
  - 由HDFS檔案產生：
    - 檔案的HDFS Block數量，在Cluster架構預設值為2
  - 由RDD產生
    - 繼承父系RDD的數量

How Many Partitions

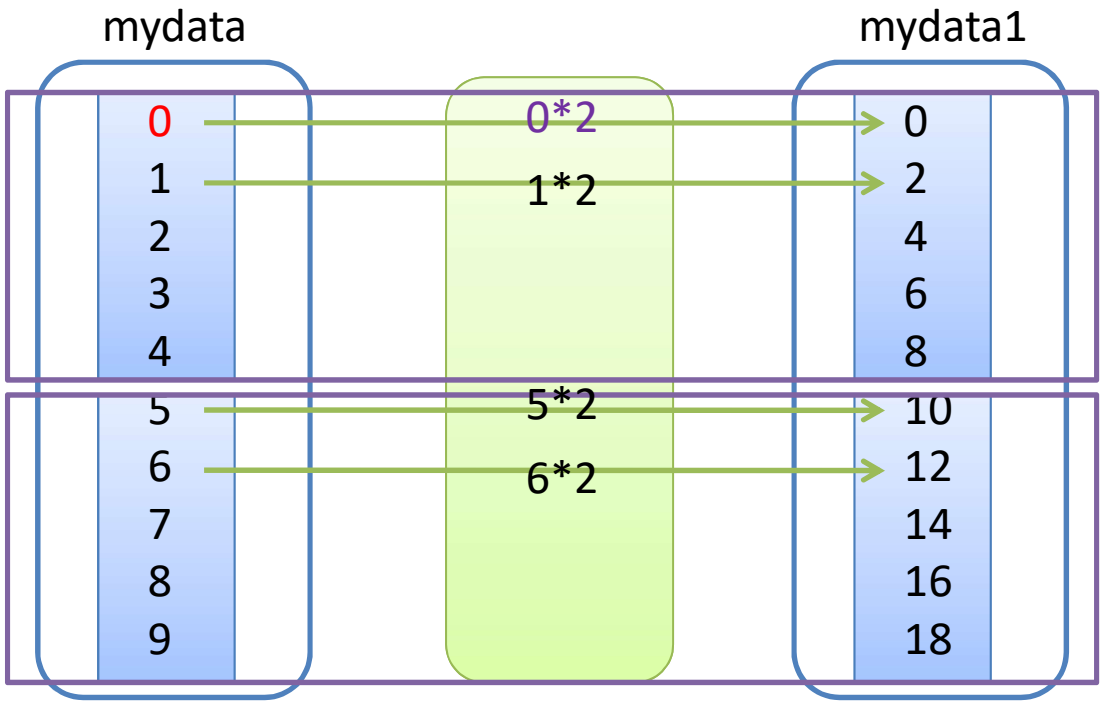
```
>>> range(101)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100]
>>> partRDD = sc.parallelize(range(101),30)
>>> partRDD.getNumPartitions()
30
```



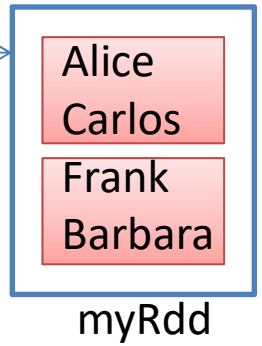

RDD
Partition
Element
Partition
element
RDD


File
Programmatic
RDD
RDD1
RDD2
RDD3
Result
transform
transform
action
Pipelining
partition
partition
partition
executor
JVM
task1
executor
JVM
task2
executor
JVM
task3
executor
JVM
task4

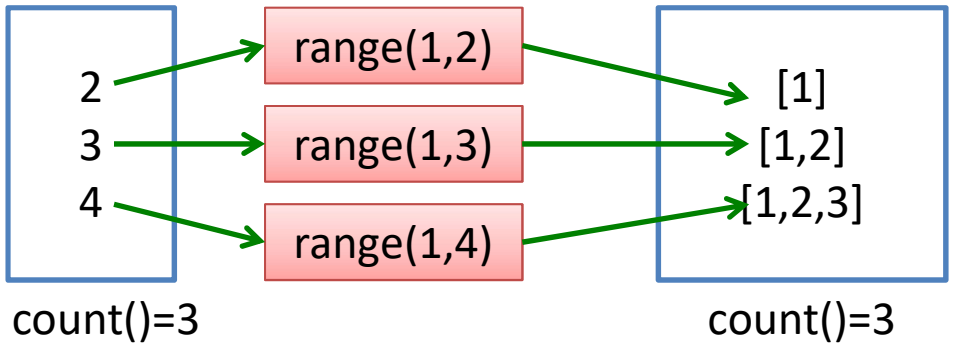
```
In [1]: mydata = sc.parallelize(range(10))

In [2]: mydata.getNumPartitions()
Out[2]: 2

In [3]: mydata1 = mydata.map(lambda x: x*2)

In [4]: mydata1.collect()

Out[4]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```



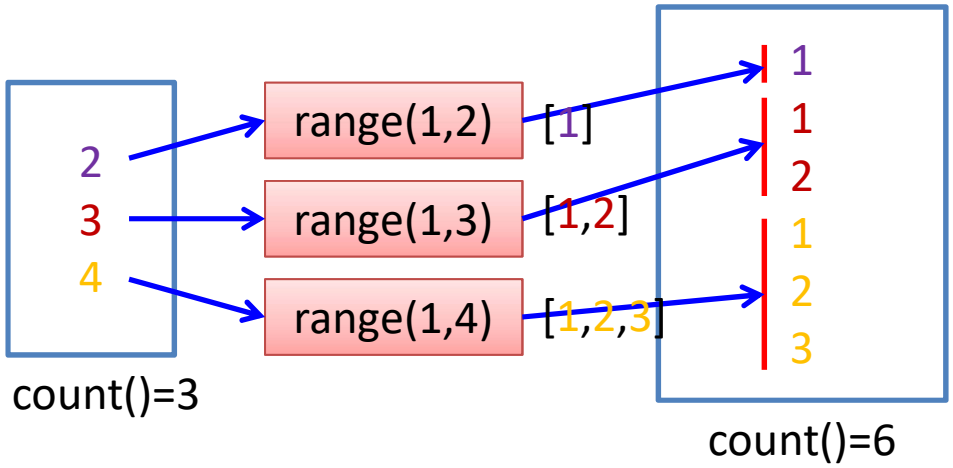
```
> myData = ["Alice","Carlos","Frank","Barbara"]

> myRdd = sc.parallelize(myData)

> myRdd
ParallelCollectionRDD[15] at parallelize at
PythonRDD.scala:376

> myRdd.take(2)
['Alice', 'Carlos']

> myRdd.count()
4

> myRdd.getNumPartitions()
2

> myRdd.collect()
['Alice', 'Carlos', 'Frank', 'Barbara']

> myRdd.sortBy(lambda x: x[0]).collect()
['Alice', 'Barbara', 'Carlos', 'Frank']
```



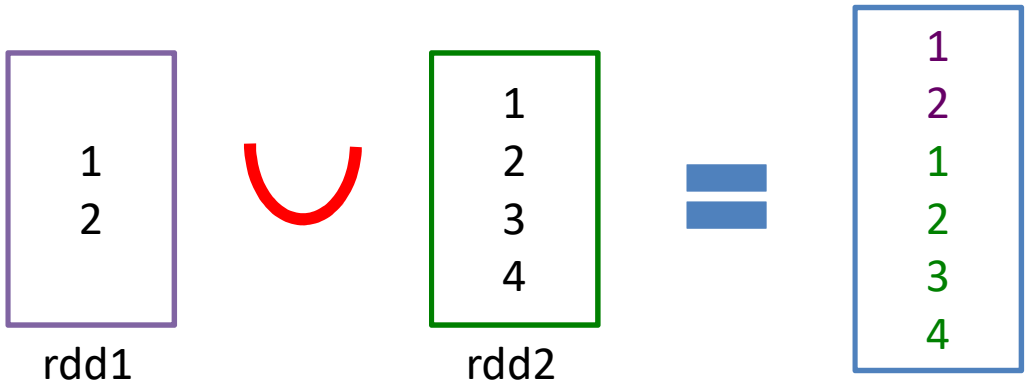
```
In [1]: sc.parallelize([2,3,4]).map(lambda x:range(1,x)).collect()
Out[1]: [[1], [1, 2], [1, 2, 3]]
```



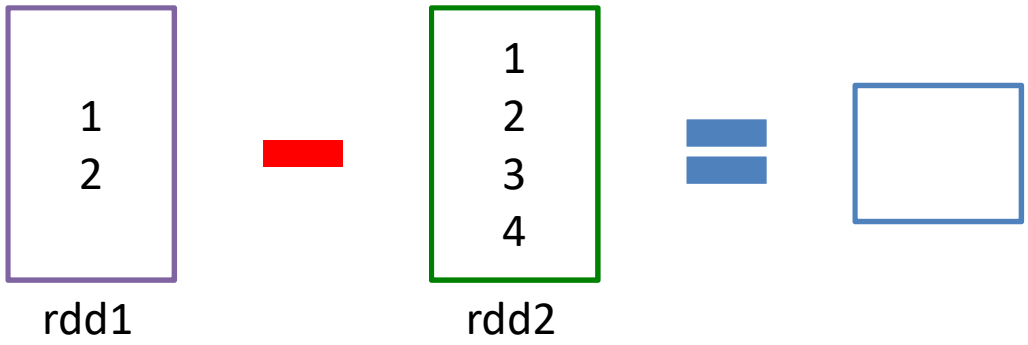

```
In [2]: sc.parallelize([2,3,4]).flatmap(lambda x:range(1,x)).collect()
Out[2]: [1, 1, 2, 1, 2, 3]
```



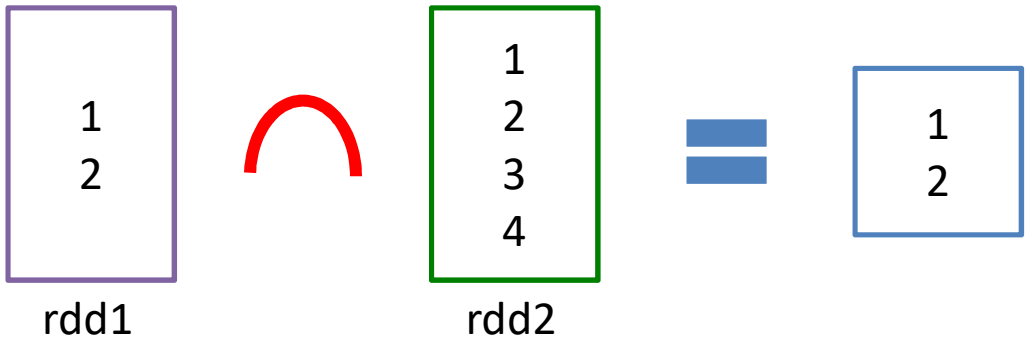
```
In [1]: rdd1 = sc.parallelize(range(1,3))
In [2]: rdd2 = sc.parallelize(range(1,5))
In [3]: rdd1.union(rdd2).collect()
Out[3]: [1, 2, 1, 2, 3, 4]
```



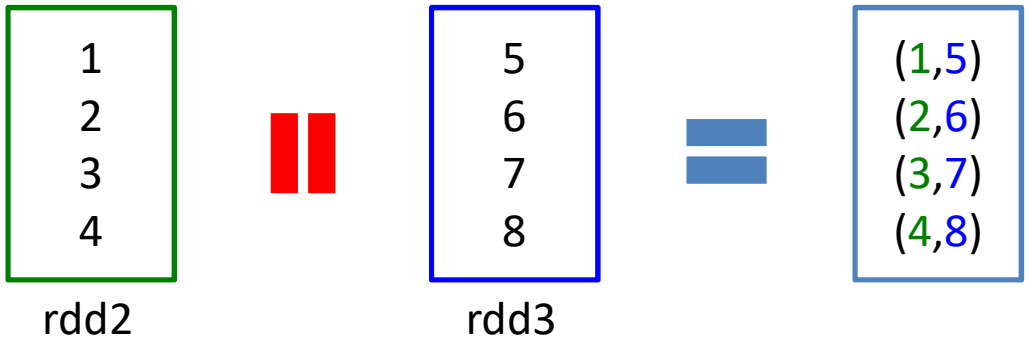

```
In [4]: rdd1.subtract(rdd2).collect()
Out[4]: []
In [5]: rdd2.subtract(rdd1).collect()
Out[5]: [3,4]
```



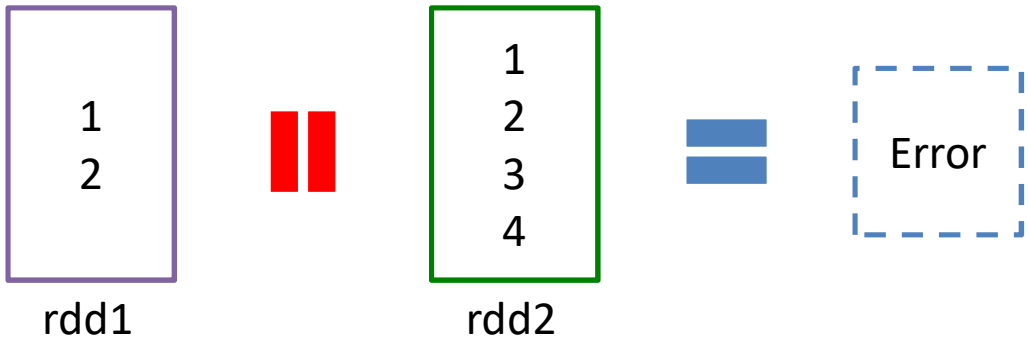
```
In [6]: rdd1.intersection(rdd2).collect()
Out[6]: [1, 2, 1, 2, 3, 4]
```



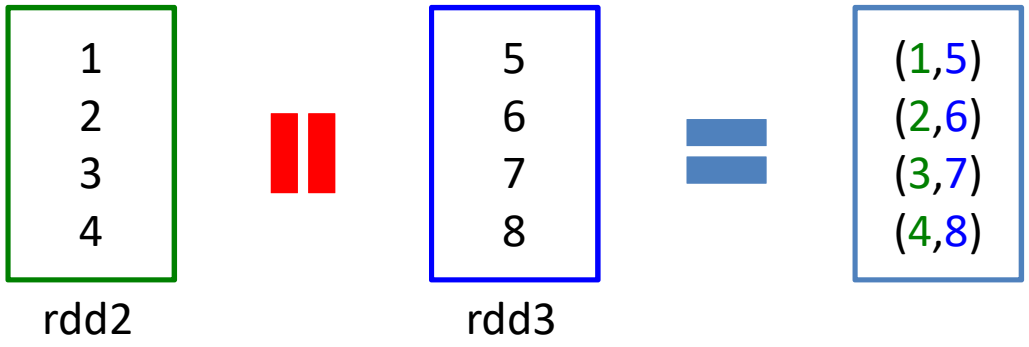

```
In [7]: rdd3=sc.parallelize(range(5,9))
In [8]: rdd2.zip(rdd3).collect()
Out[8]: [(1, 5), (2, 6), (3, 7), (4, 8)]
```



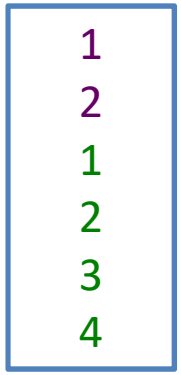
```
In [9]: rdd1.zip(rdd2).collect()
```




```
In [10]: rdd3=sc.parallelize(range(5,9))
In [11]: rdd2.zip(rdd3).collect()
Out[11]: [(1, 5), (2, 6), (3, 7), (4, 8)]
```

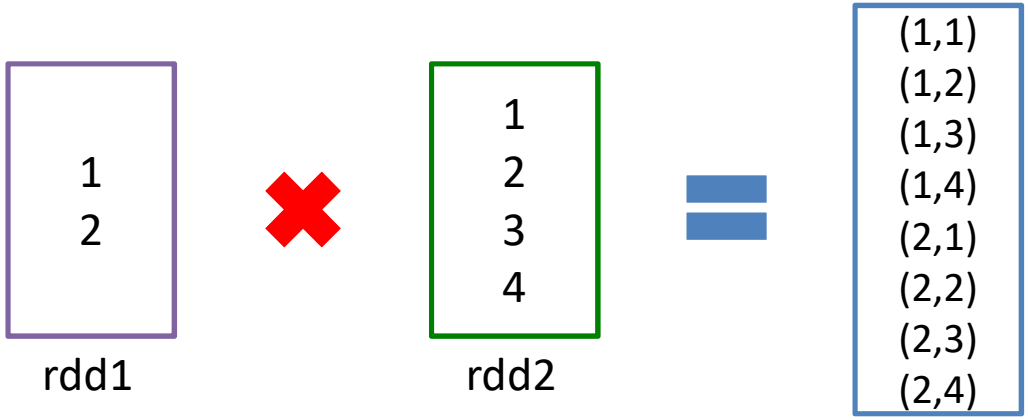


```
In [12]: rdd4 = rdd1.union(rdd2)
In [12]: rdd4.collect()
Out[12]: [1, 2, 1, 2, 3, 4]
```



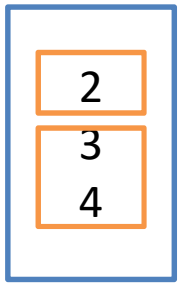
rdd4

```
In [13]: rdd4.distinct.collect()
Out[13]: [1,2,3,4]

In [14]: rdd1.cartesian(rdd2).collect()
Out[14]: [(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (2, 4)]
```



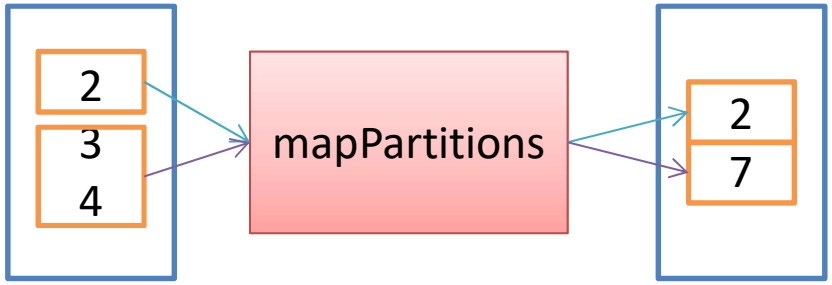
rdd1 rdd2

```
In [15]: sc.parallelize([2,3,4]).getNumPartitions()
Out[15]: 2
In [16]: sc.parallelize([2,3,4]).glom().collect()
Out[16]: [[2], [3, 4]]
```

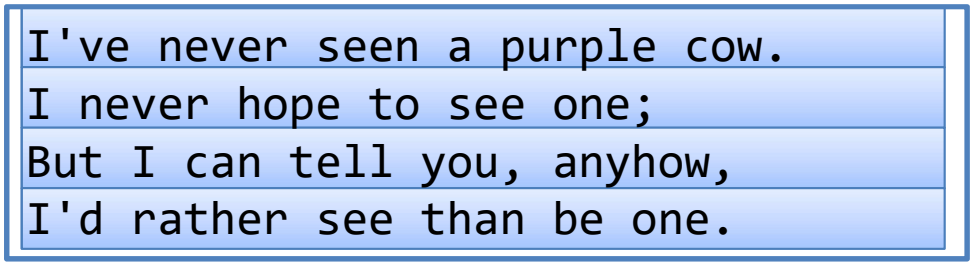



```
In [17]: def f(iteral):
   ....:     yield sum(iteral)
   ....:
```

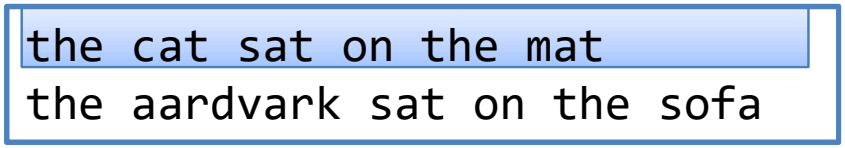
```
In [18]: sc.parallelize([2,3,4]).mapPartitions(f).collect()
Out[18]: [2, 7]
```



```
$ hdfs dfs -cat purplecow.txt
I've never seen a purple cow.
I never hope to see one;
But I can tell you, anyhow,
I'd rather see than be one.

$ pyspark2

In [1]: mydata = sc.textFile('purplecow.txt')
```

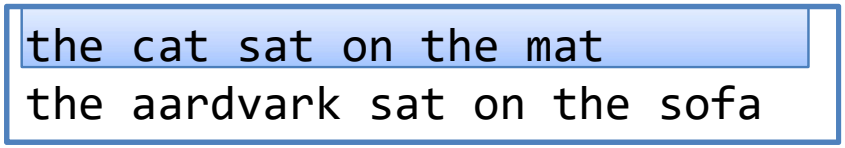



```
In [2]: mydata.collect()
Out[2]:
[u"I've never seen a purple cow.",
 u'I never hope to see one;',
 u'But I can tell you, anyhow,',
 u"I'd rather see than be one."]
```



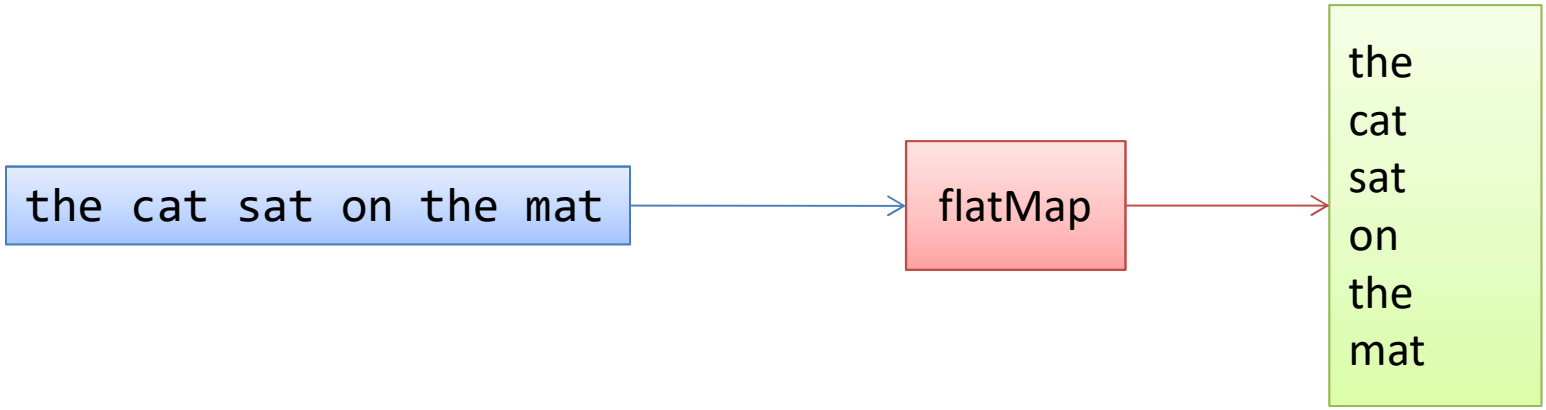
mydata

```
In [4]: mydata2 = mydata.map(lambda line: line.split(' '))
```

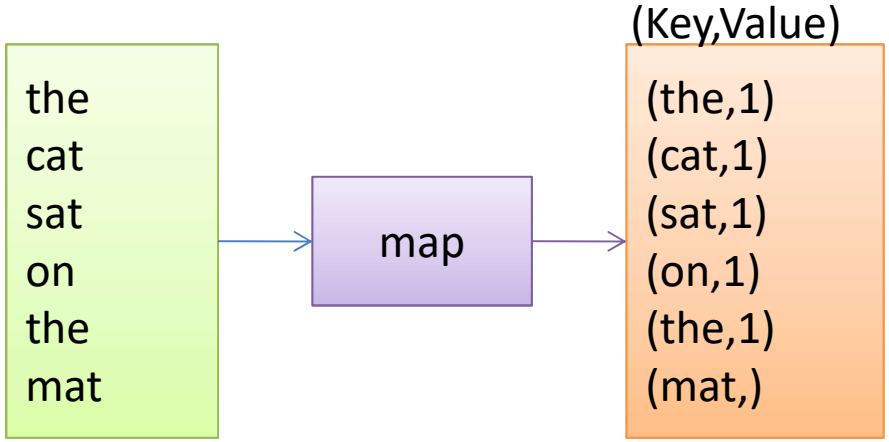


mydata

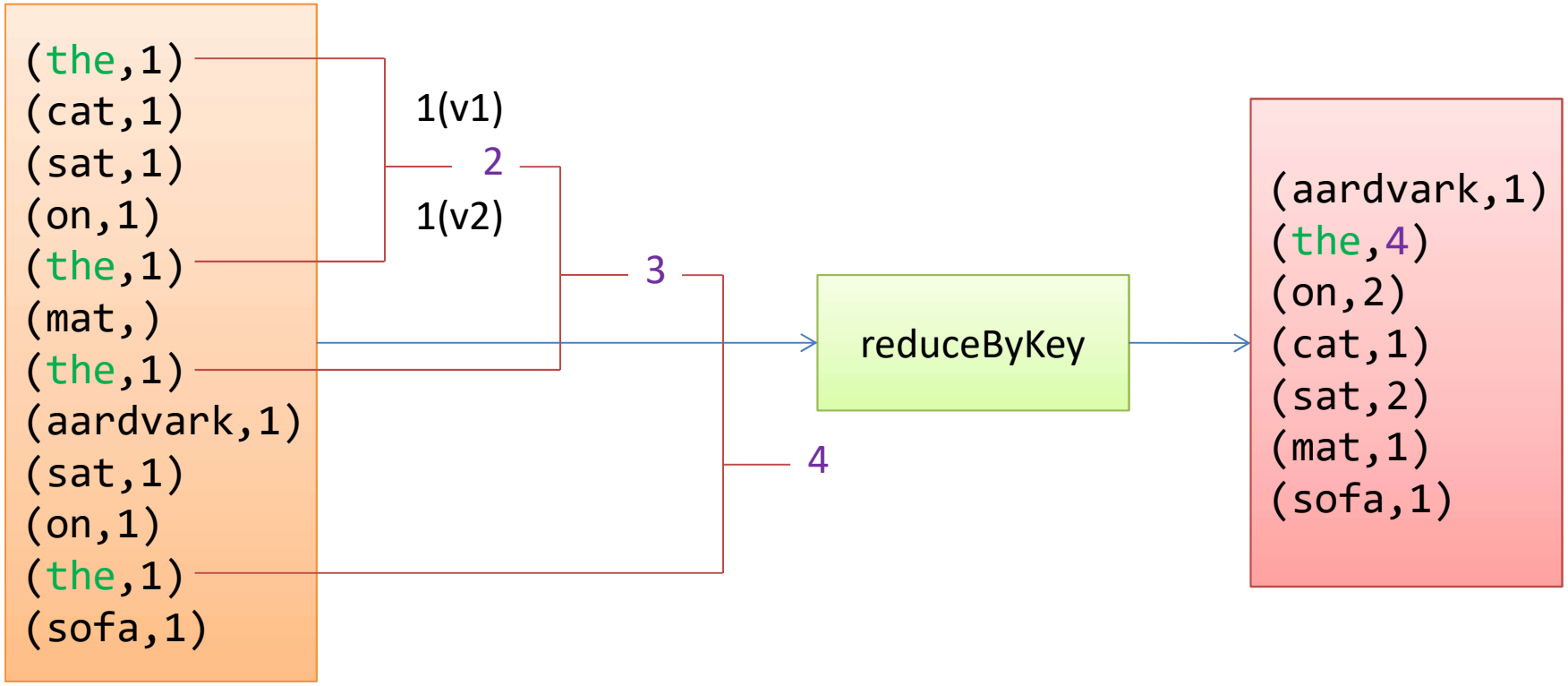
```
In [4]: mydata2 = mydata.flatMap(lambda line: line.split(' '))
```



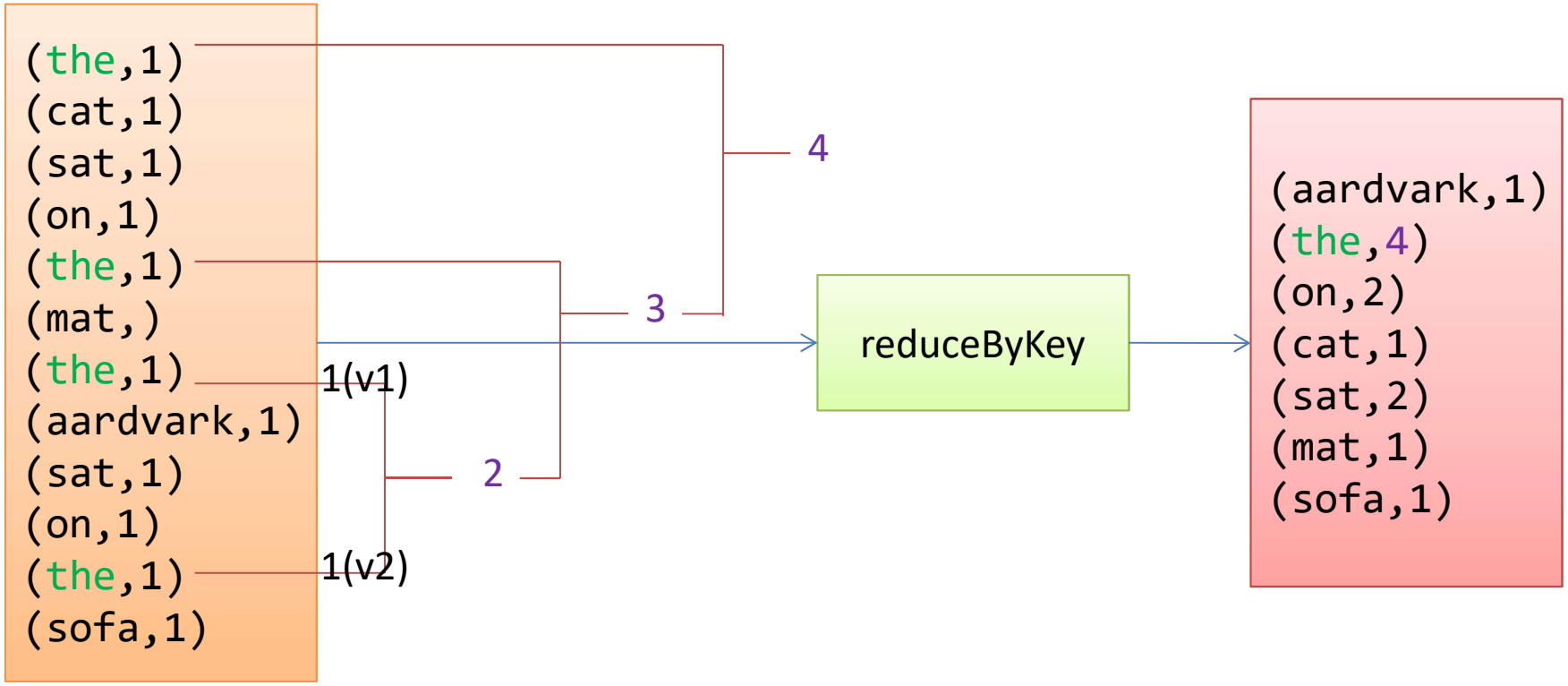

```
In [7]: mydata3 = mydata2.map(lambda word: (word,1))
```



```
In [10]: mydata4 = mydata3.reduceByKey(lambda v1,v2: v1+v2)
```



```
In [10]: mydata4 = mydata3.reduceByKey(lambda v1,v2: v1+v2)
```

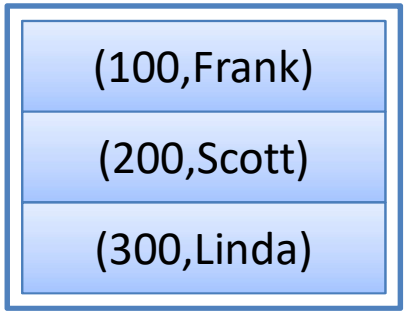


user001\tFred Flintstone\n
user090\tBugs Bunny\n
user111\tHarry Potter\n

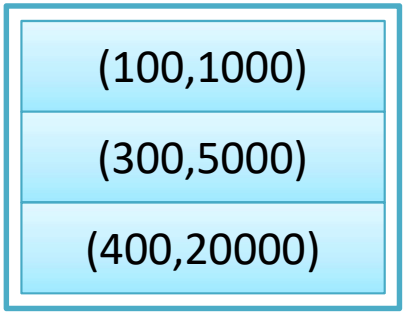
```
> users = sc.textFile(file) \
  .map(lambda line: line.split('\t')) \
  .map(lambda fields: (fields[0],fields[1]))
```

user001\tFred Flintstone
[user001,Fred Flintstone]
(user001,Fred Flintstone)

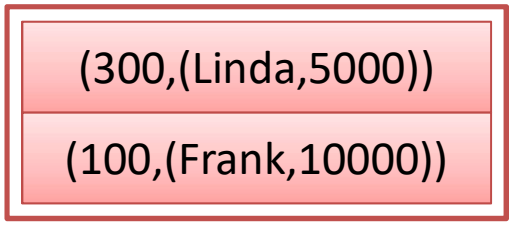
rdd1m



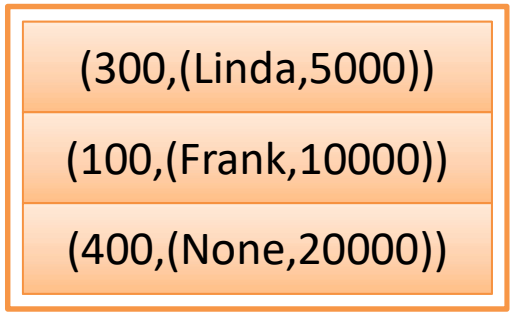
rdd2m



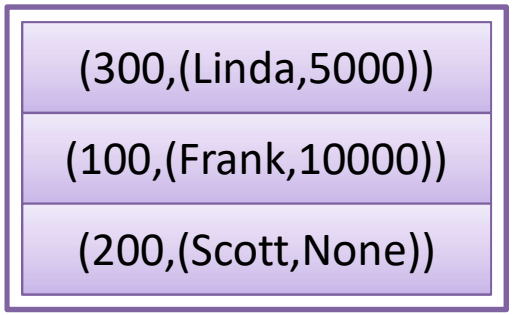
rdd1m.join(rdd2m)



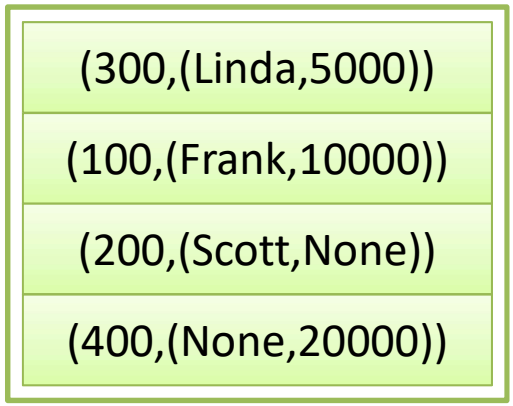
rdd1m.rightOuterJoin(rdd2m)



rdd1m.leftOuterJoin(rdd2m)



Rdd1m.fulltOuterJoin(rdd2m)



0 1 2 3 4 5

```
56.38.234.188 - 99788 "GET /KBDOC-00157.html HTTP/1.0"
```

```
> sc.textFile(logfile).keyBy(lambda line: line.split(' ')[2])
```

```
(99788,56.38.234.188 - 99788 "GET /KBDOC-00157.html HTTP/1.0")
```

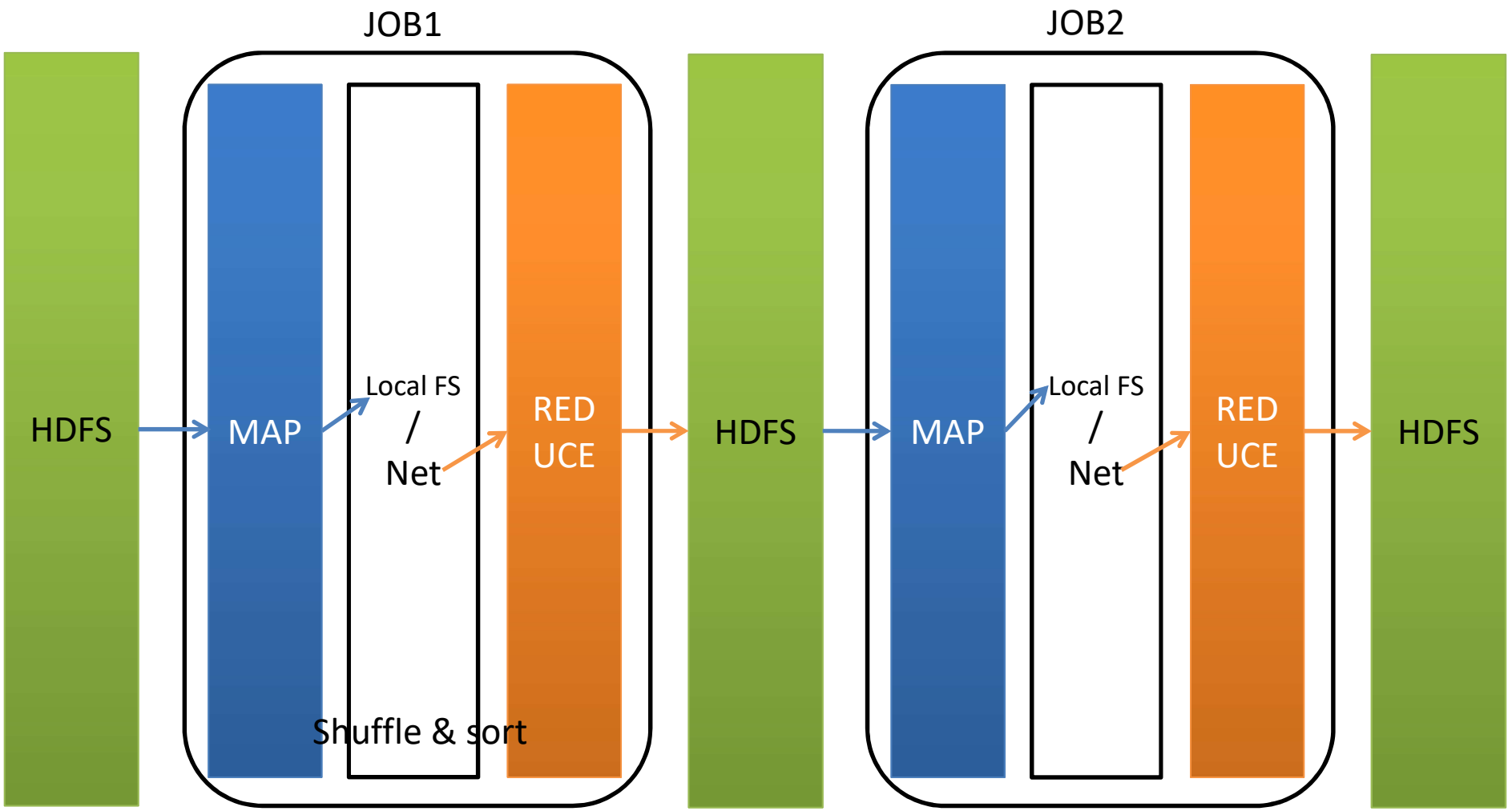
0 1 2

```
00210 43.005895 -71.013202
```

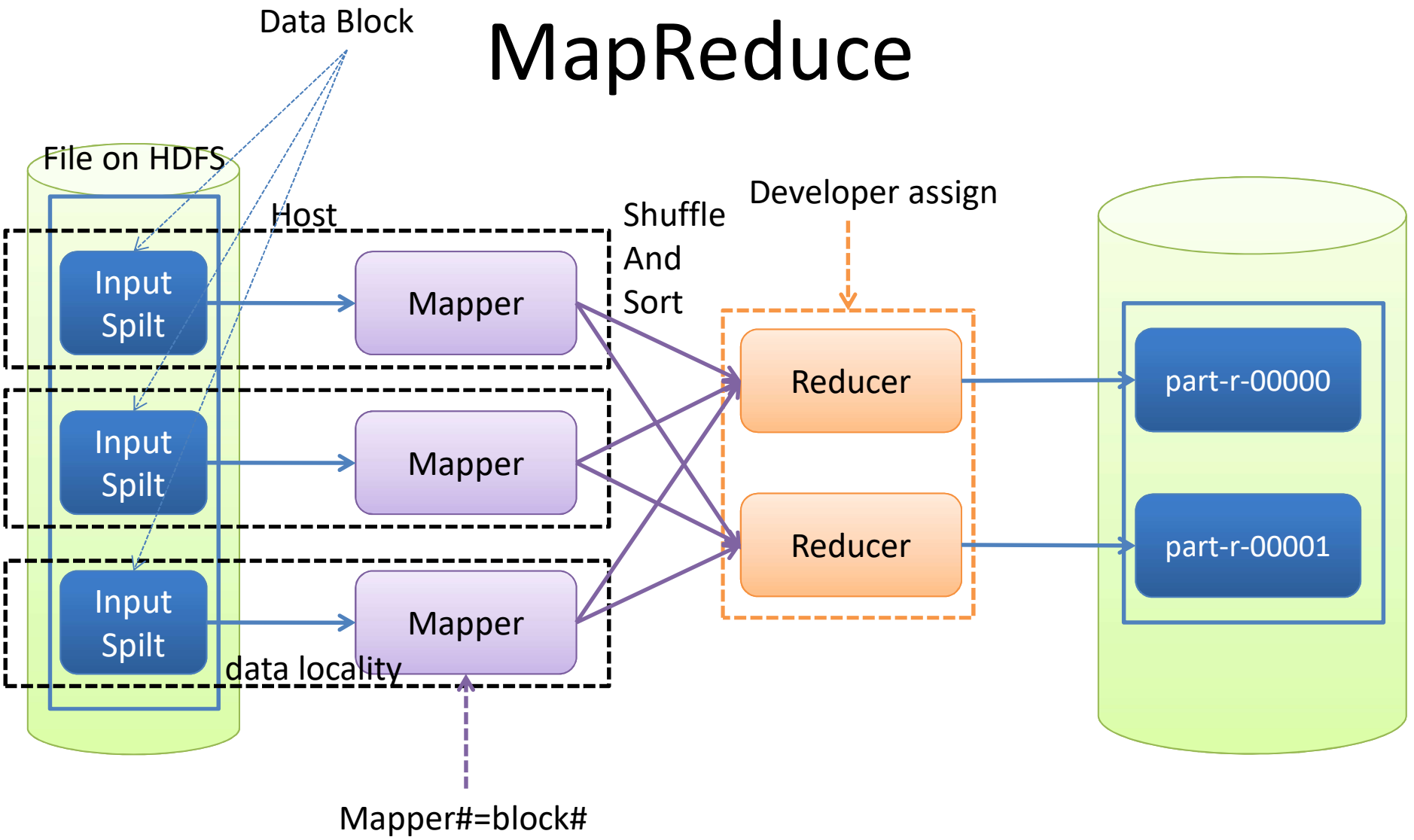
```
> sc.textFile(file).map(lambda line: line.split(' ').\
   map(lambda fields: (fields[0],(fields[1],fields[2])))
```

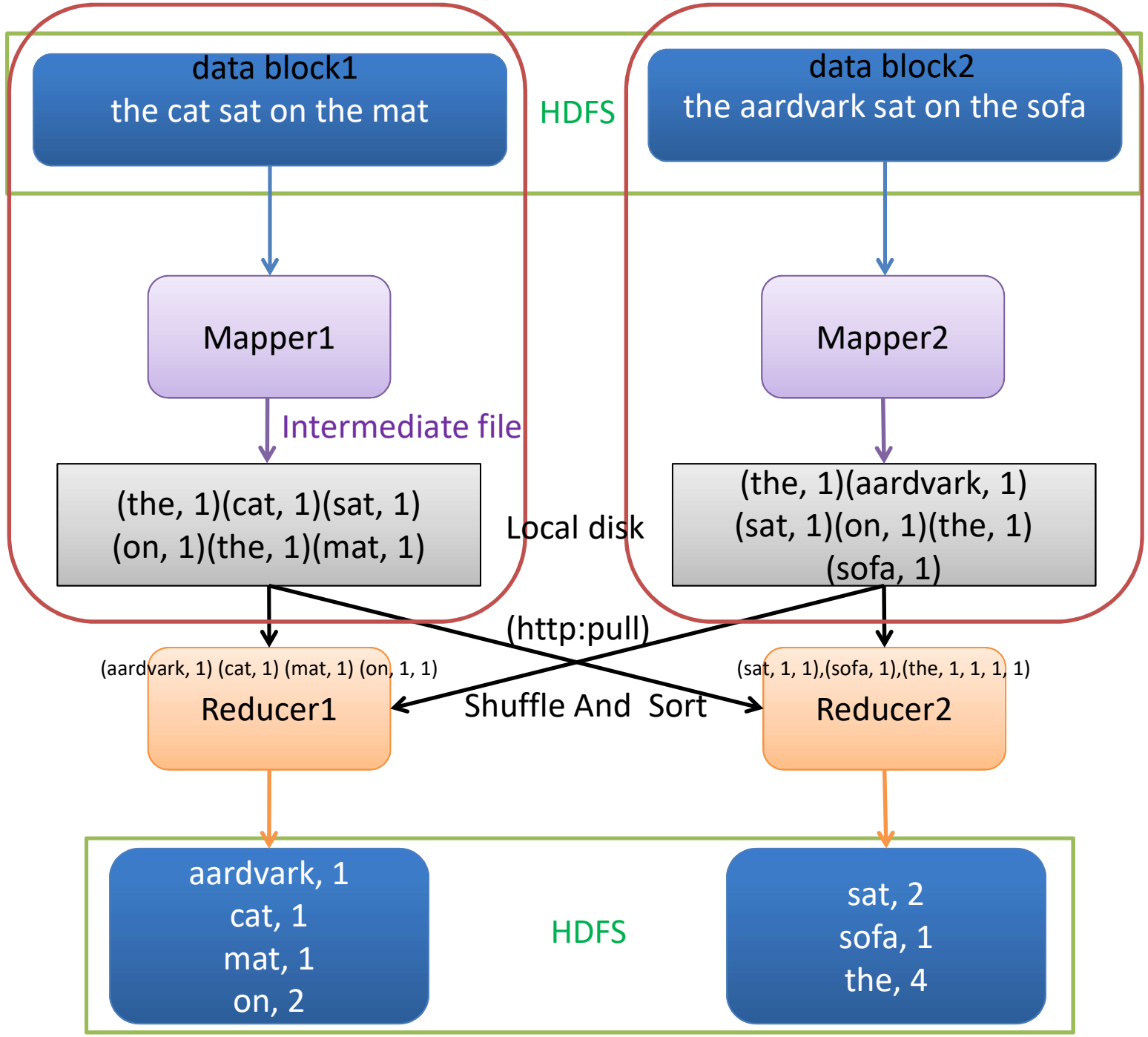
```
(00210,(43.005895,-71.013202))
```

MapReduce

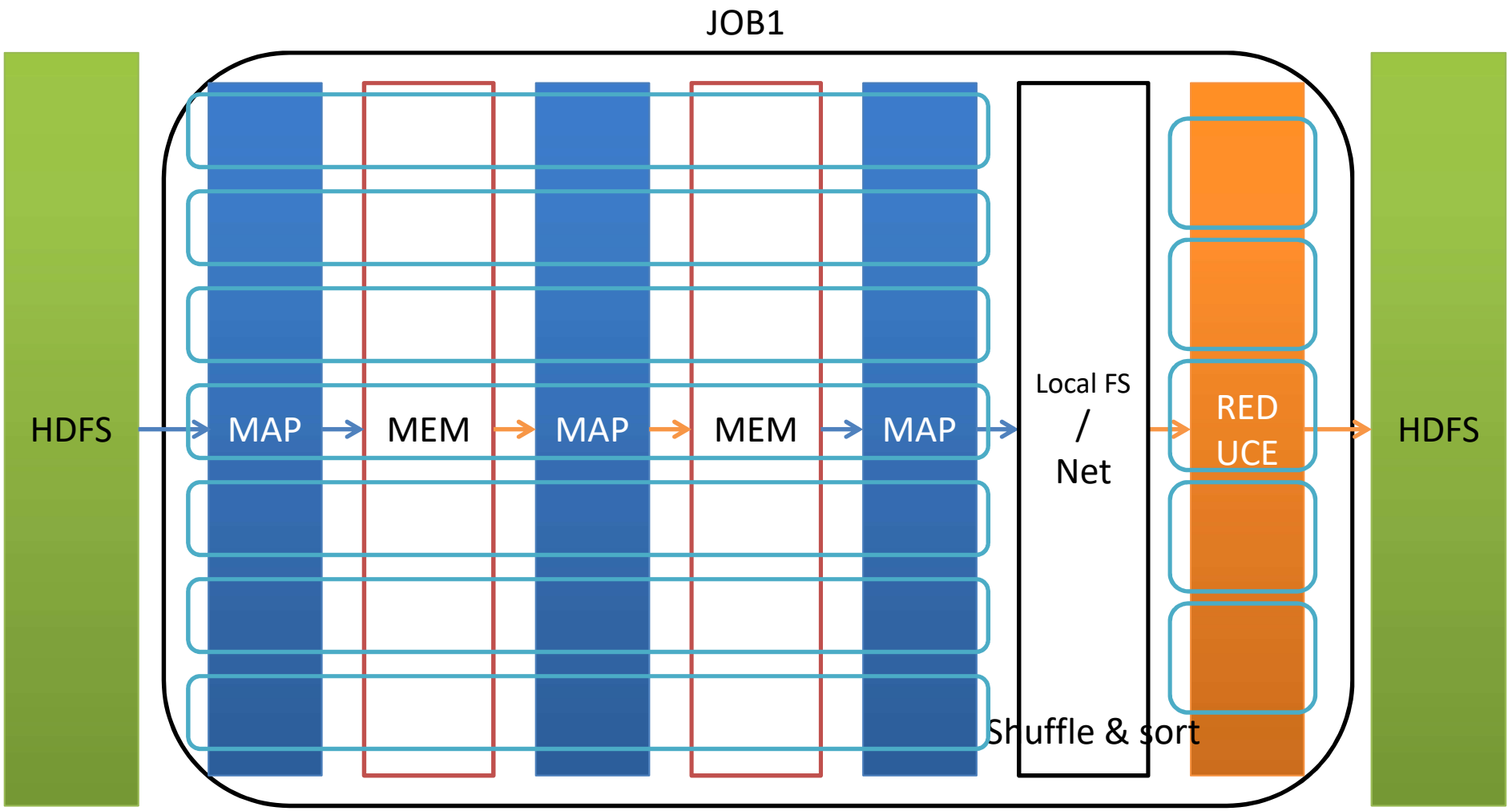

JOB1
JOB2
HDFS
MAP
Local FS
/
Net
RED
UCE
HDFS
MAP
Local FS
/
Net
RED
UCE
HDFS
Shuffle & sort

MapReduce


Data Block
File on HDFS
Host
Input Spilt
Input Spilt
Input Spilt
Mapper
Mapper
Mapper
data locality
Mapper#=block#
Shuffle And Sort
Developer assign
Reducer
Reducer
part-r-00000
part-r-00001

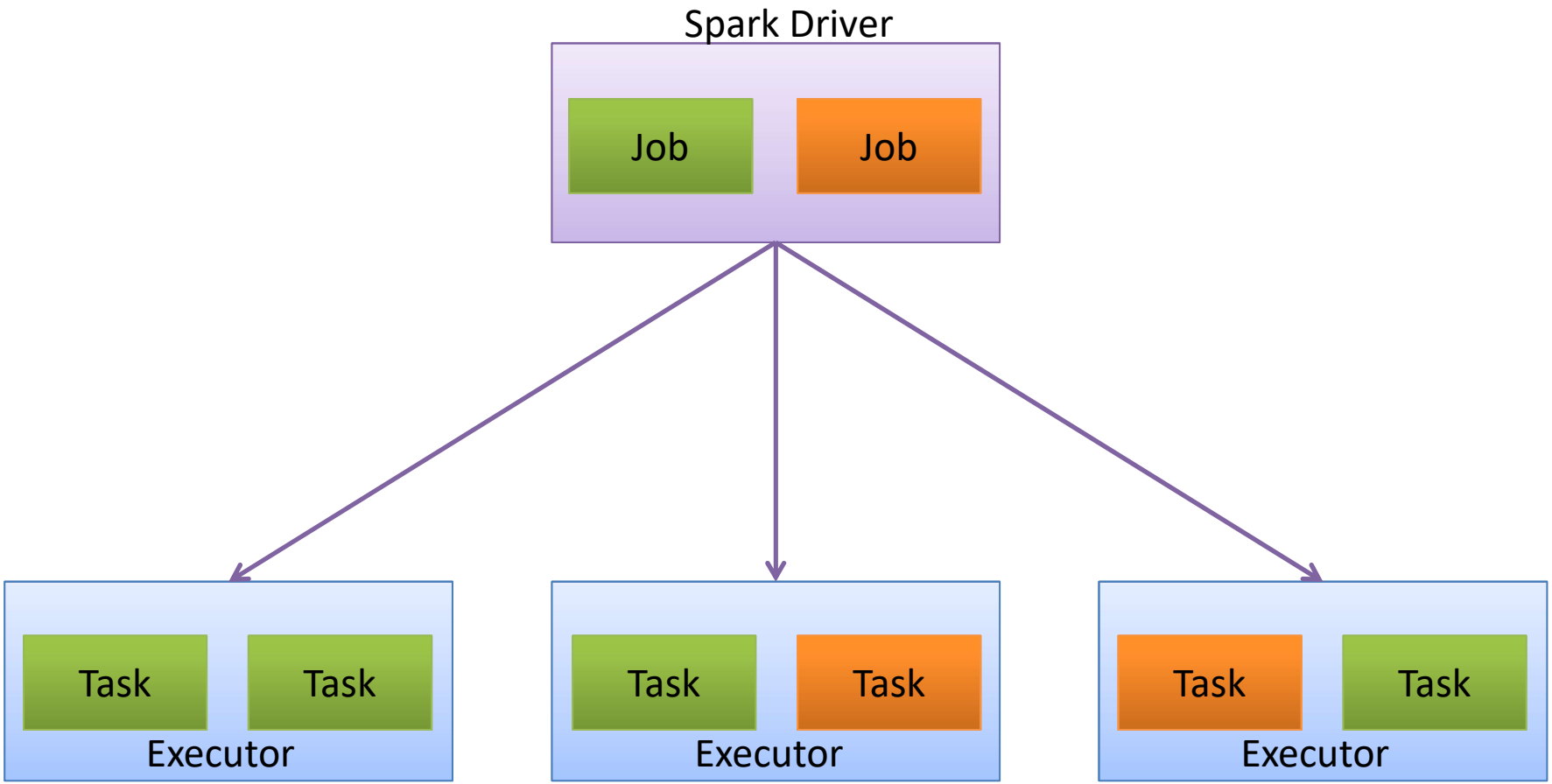

data block1
the cat sat on the mat
HDFS
data block2
the aardvark sat on the sofa
Mapper1
Mapper2
Intermediate file
(the, 1)(cat, 1)(sat, 1)
(on, 1)(the, 1)(mat, 1)
Local disk
(the, 1)(aardvark, 1)
(sat, 1)(on, 1)(the, 1)
(sofa, 1)
(http:pull)
(aardvark, 1) (cat, 1) (mat, 1) (on, 1, 1)
(sat, 1, 1),(sofa, 1),(the, 1, 1, 1, 1)
Reducer1
Shuffle And Sort
Reducer2
aardvark, 1
cat, 1
mat, 1
on, 2
HDFS
sat, 2
sofa, 1
the, 4

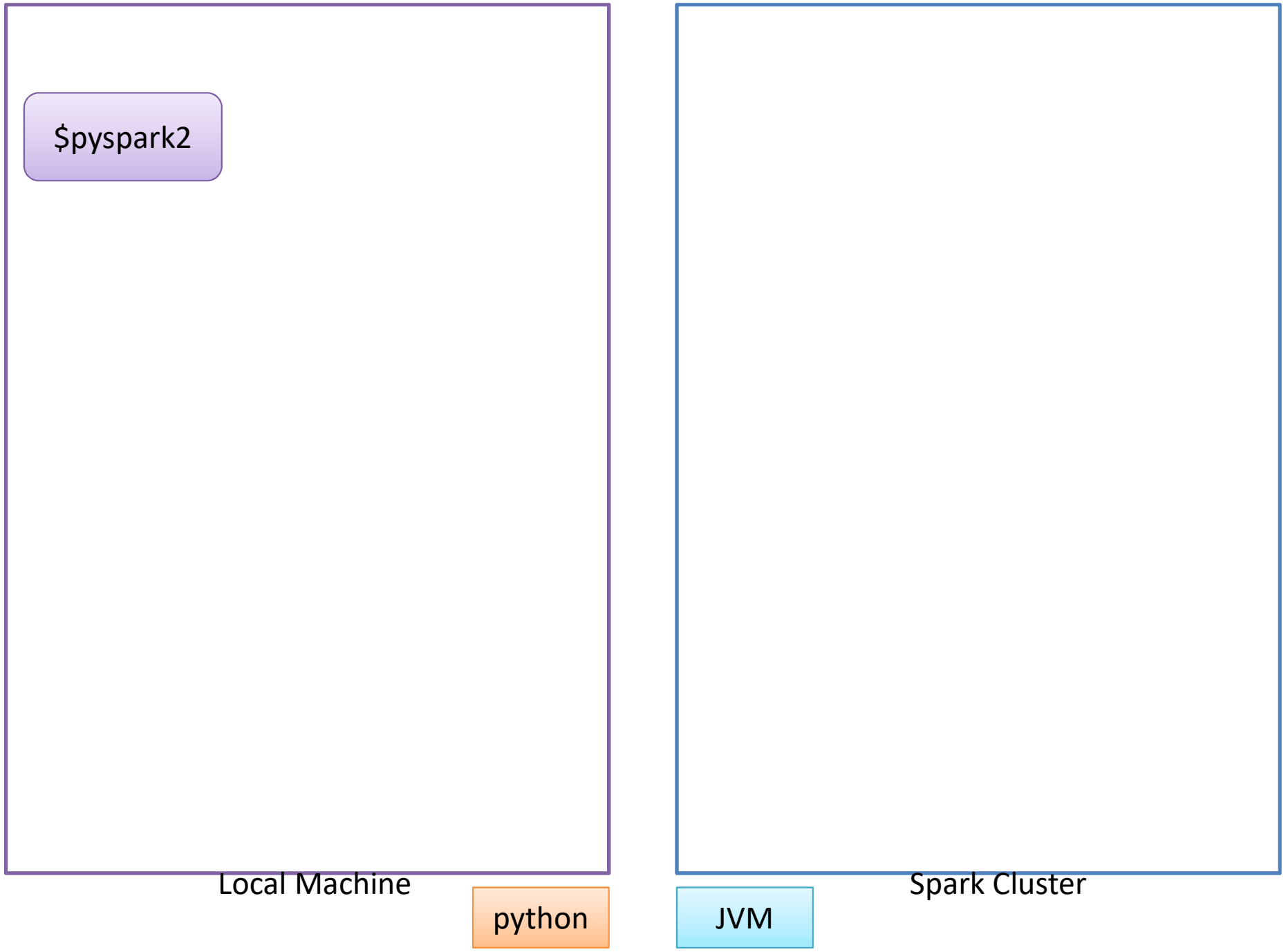
Spark

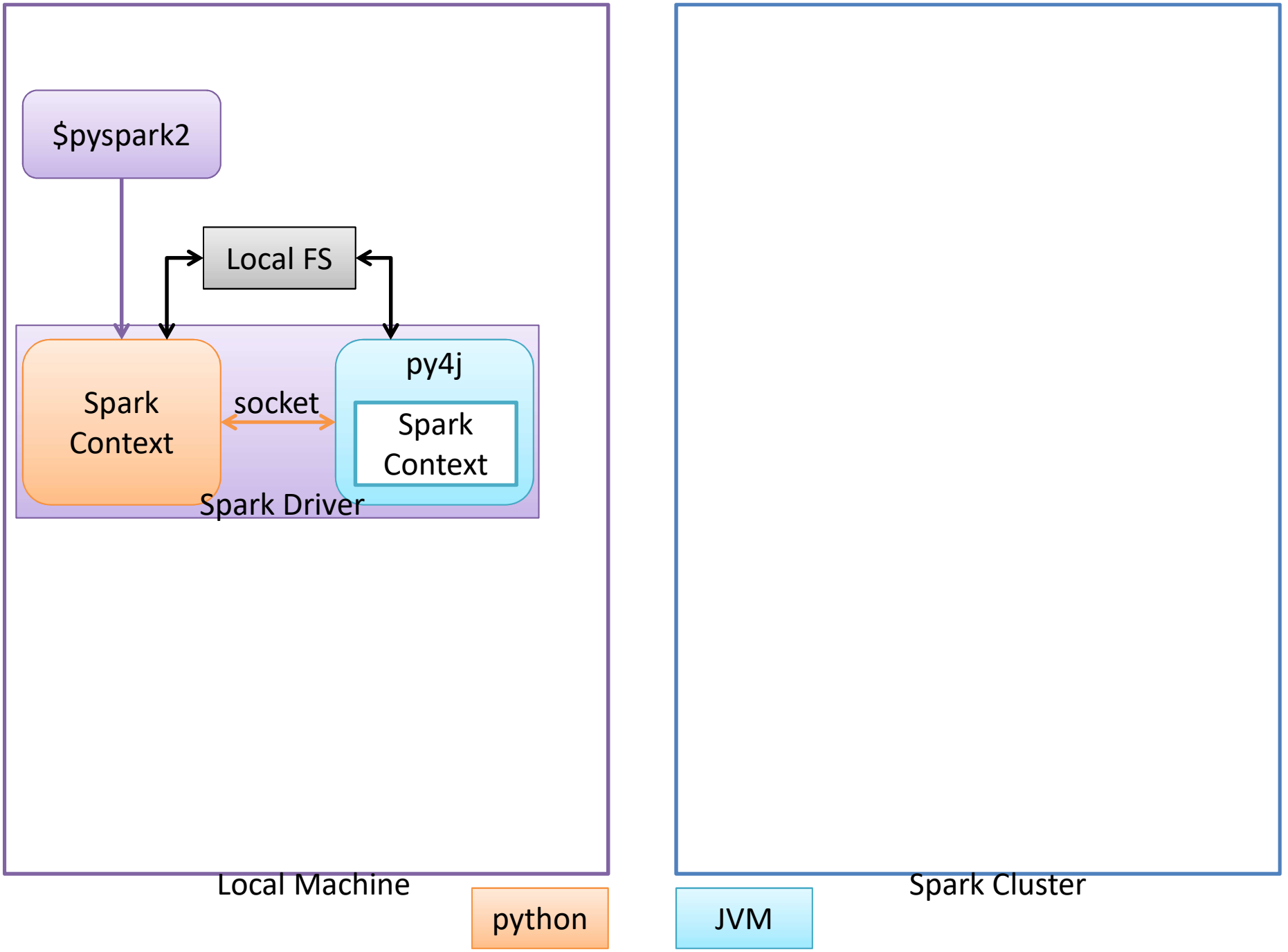

JOB1
HDFS
MAP
MEM
MAP
MEM
MAP
Local FS
/
Net
RED
UCE
HDFS
Shuffle & sort

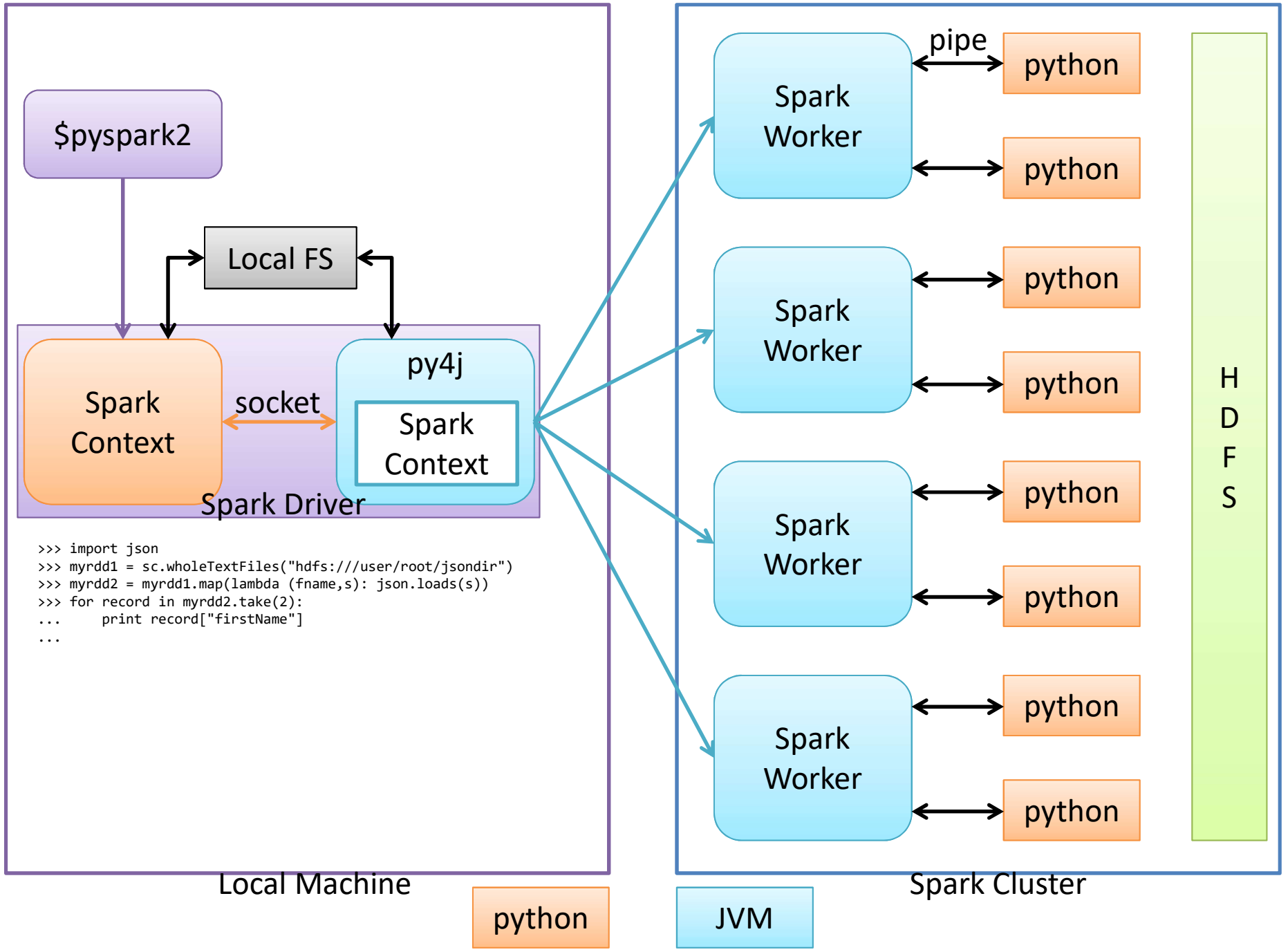
Spark Application

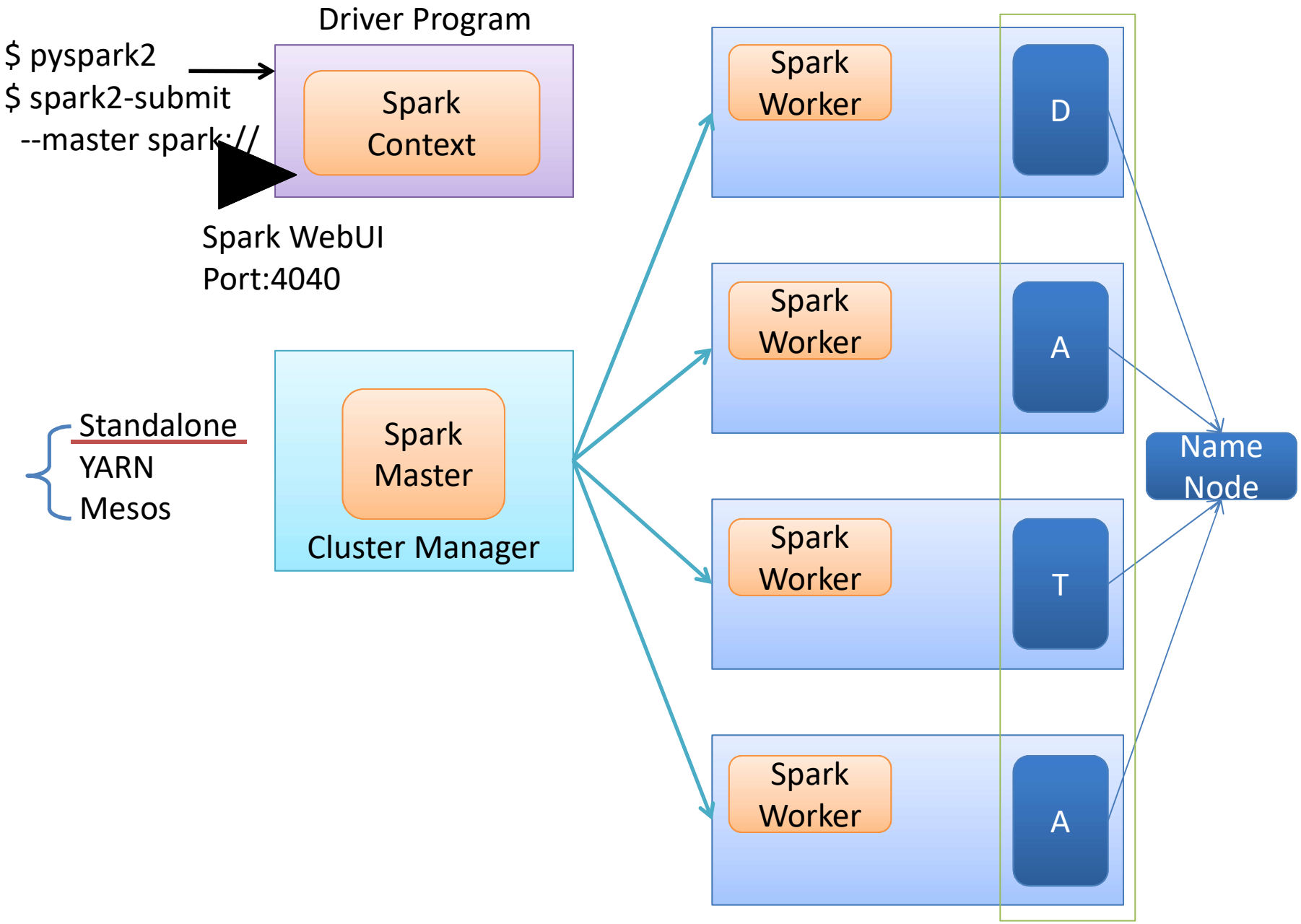
- 只能有一個sparkContext
- 可以一個或多個sparkSession

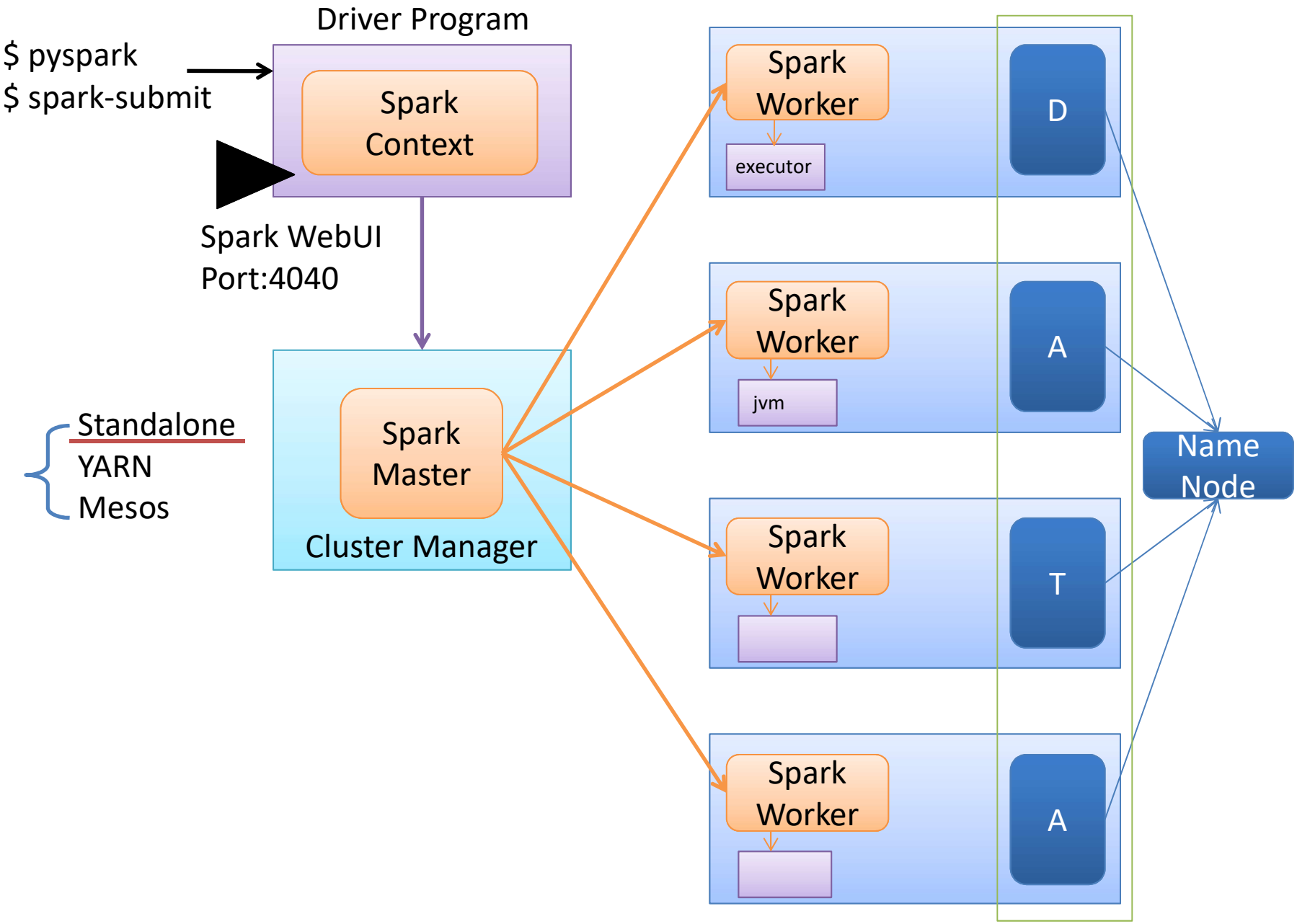

Spark Driver
Job
Job
Task
Task
Executor
Task
Task
Executor
Task
Task
Executor

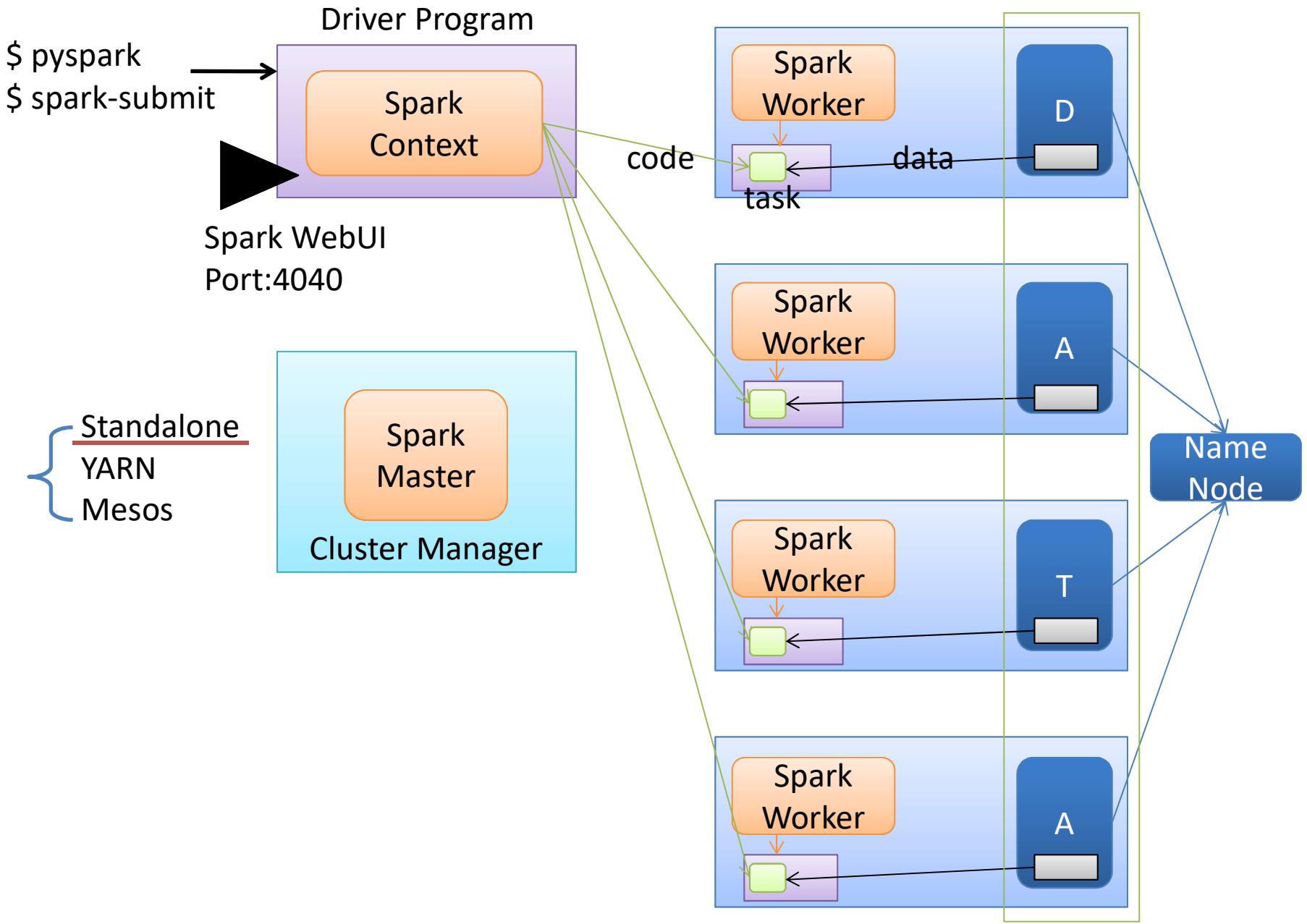

$pyspark2
Local Machine
python
JVM
Spark Cluster

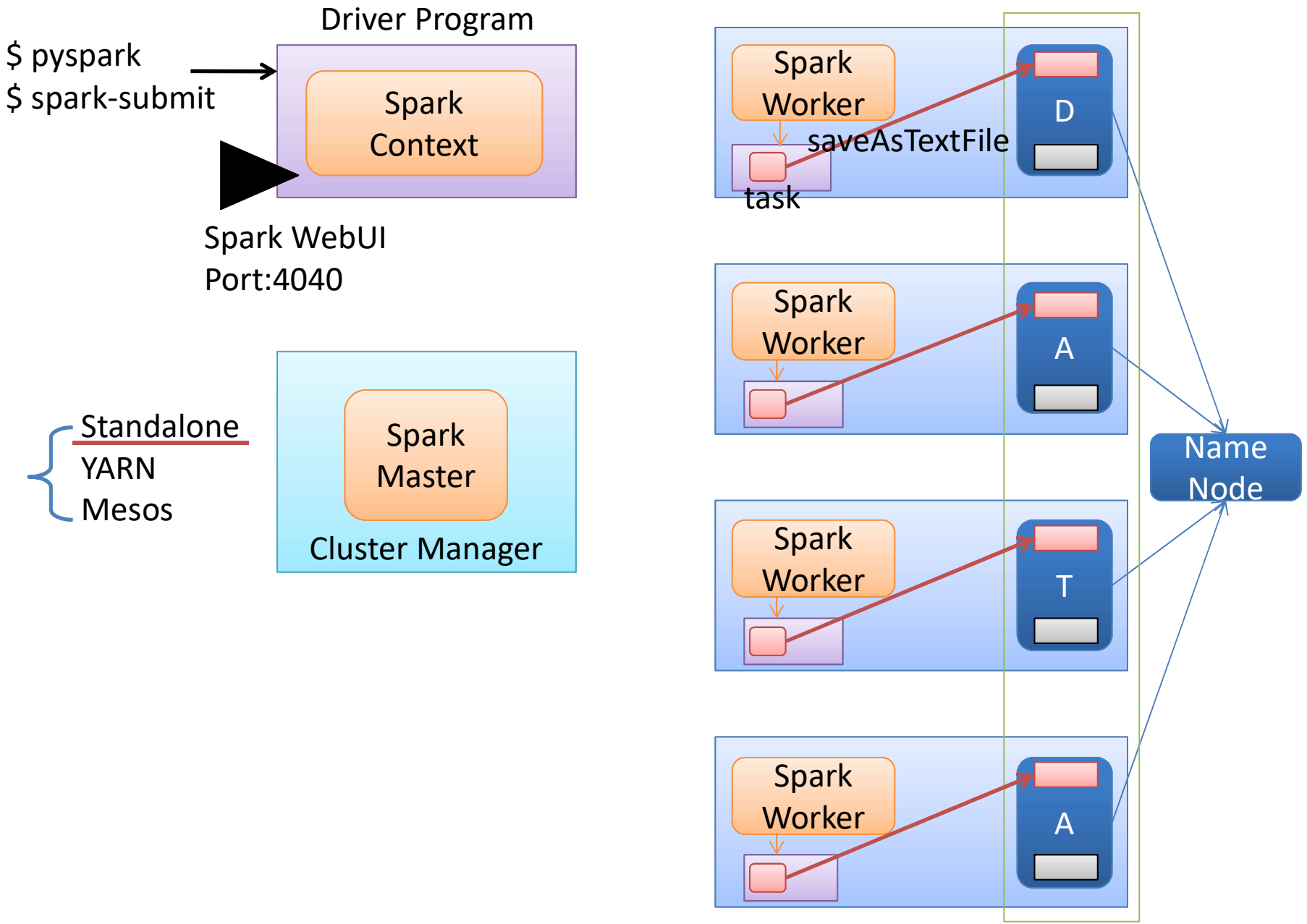

$pyspark2
Local FS
Spark Context
socket
py4j
Spark Context
Spark Driver
Local Machine
Spark Cluster
python
JVM

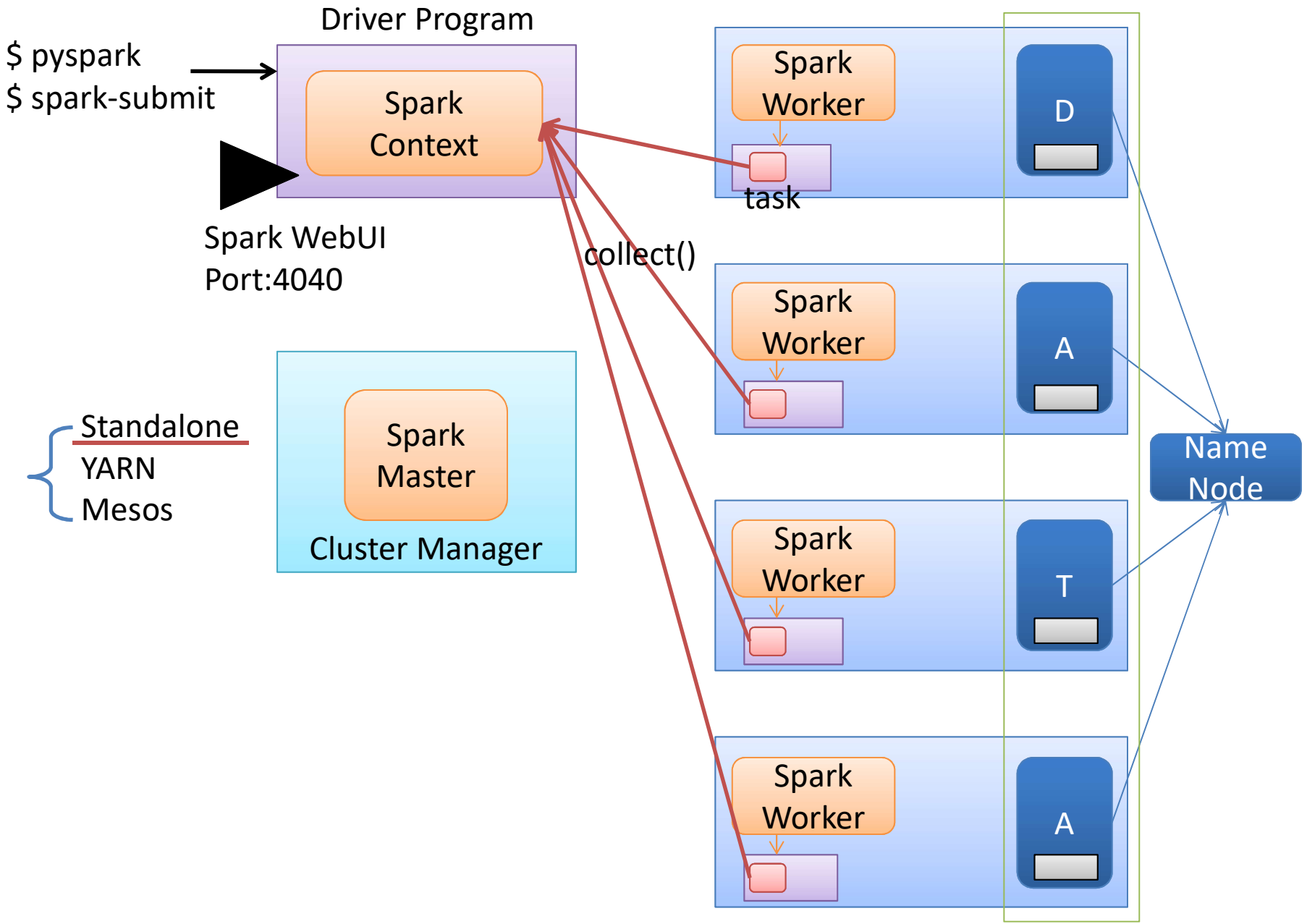

$pyspark2
Local FS
Spark Context
socket
py4j
Spark Context
Spark Driver
>>> import json
>>> myrdd1 = sc.wholeTextFiles("hdfs:///user/root/jsondir")
>>> myrdd2 = myrdd1.map(lambda (fname,s): json.loads(s))
>>> for record in myrdd2.take(2):
...   print record["firstName"]
...
Local Machine
python
pipe
Spark Worker
Spark Worker
Spark Worker
Spark Worker
python
python
python
python
python
python
python
python
H D F S
Spark Cluster
JVM

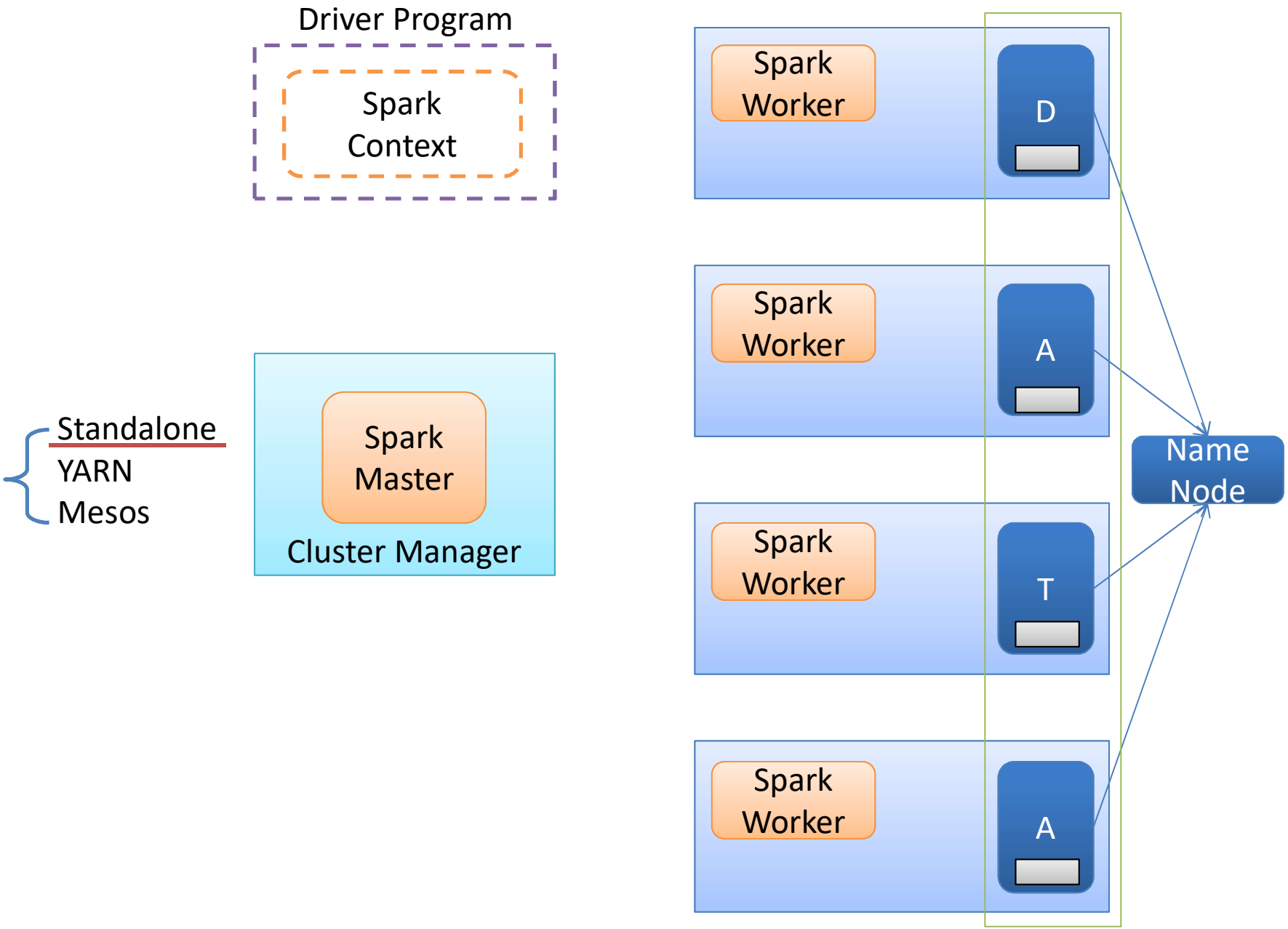

Driver Program
$ pyspark2
$ spark2-submit
--master spark://
Spark Context
Spark WebUI
Port:4040
Standalone
YARN
Mesos
Spark Master
Cluster Manager
Spark Worker
D
Spark Worker
A
Spark Worker
T
Spark Worker
A
Name Node

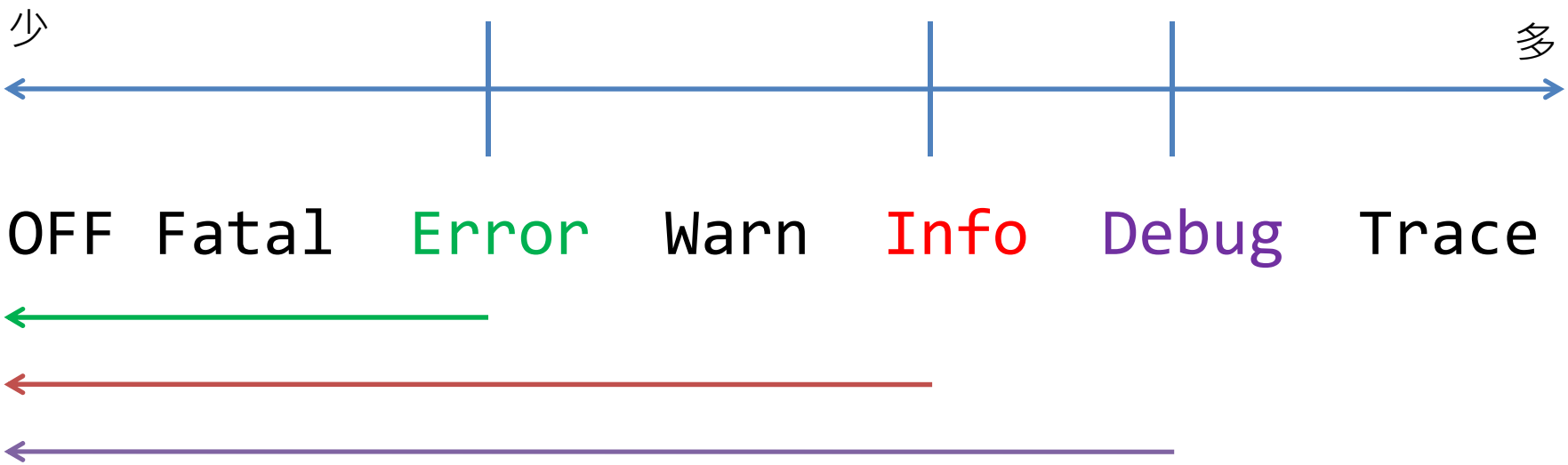

Driver Program
$ pyspark
$ spark-submit
Spark Context
Spark WebUI
Port:4040
Standalone
YARN
Mesos
Spark Master
Cluster Manager
Spark Worker
executor
D
Spark Worker
jvm
A
Spark Worker
T
Spark Worker
A
Name Node

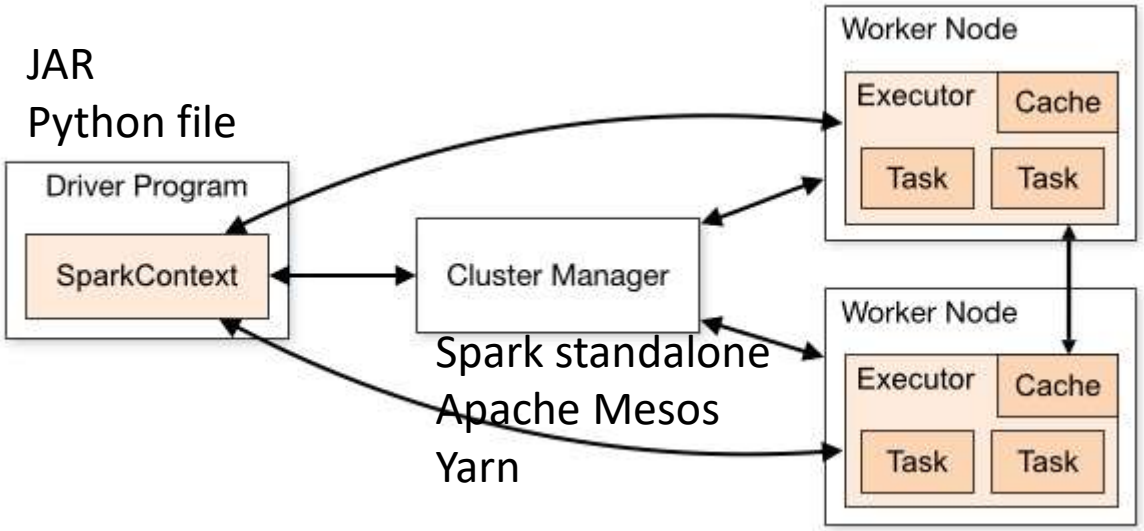

Driver Program
$ pyspark
$ spark-submit
Spark Context
Spark WebUI
Port:4040
code
data
task
Spark Worker
D
Spark Worker
A
Spark Worker
T
Spark Worker
A
Name Node
Standalone
YARN
Mesos
Spark Master
Cluster Manager

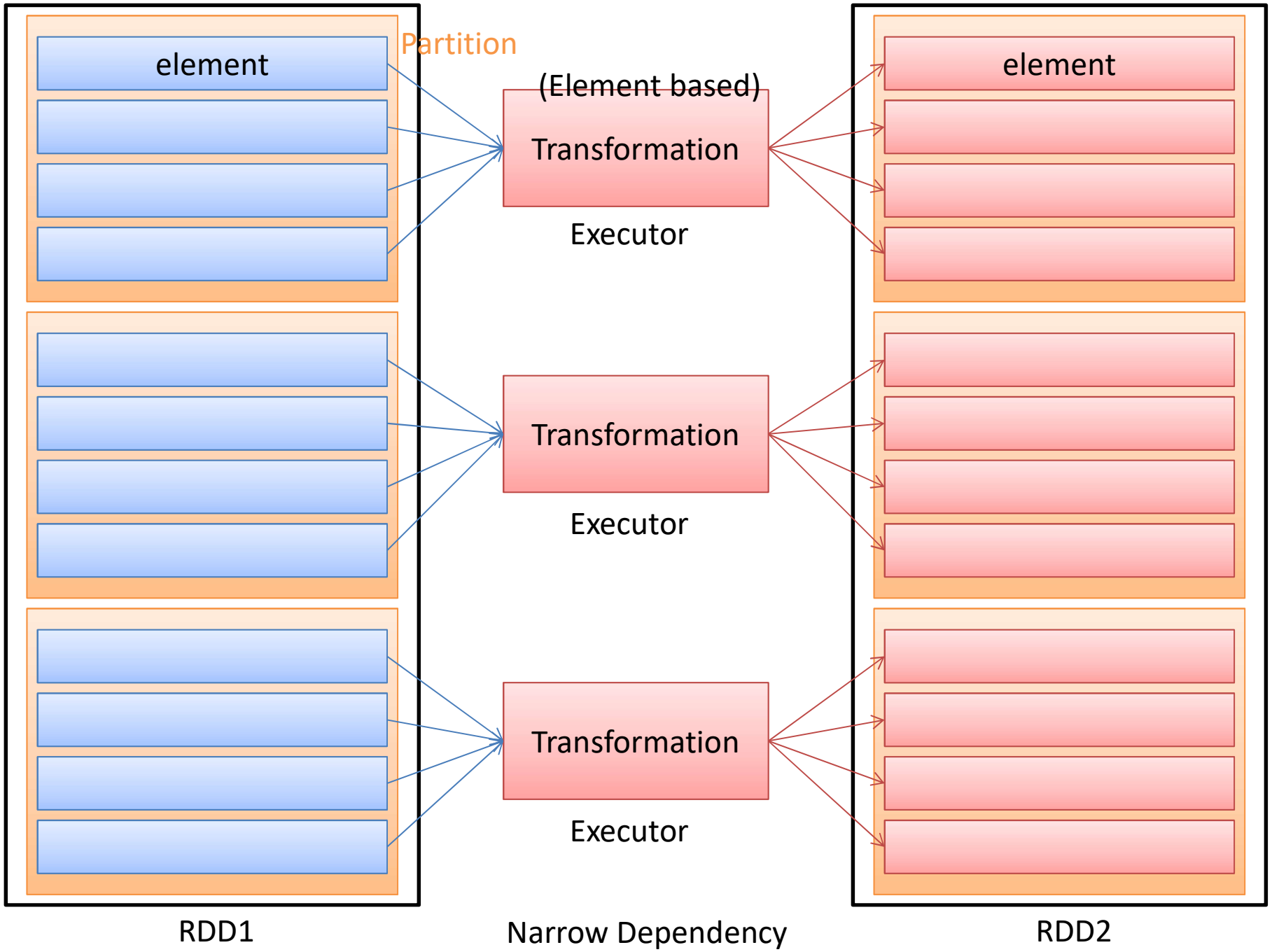

Driver Program
$ pyspark
$ spark-submit
Spark Context
Spark WebUI
Port:4040
Standalone
YARN
Mesos
Spark Master
Cluster Manager
Spark Worker
saveAsTextFile
task
D
Spark Worker
A
Spark Worker
T
Spark Worker
A
Name Node

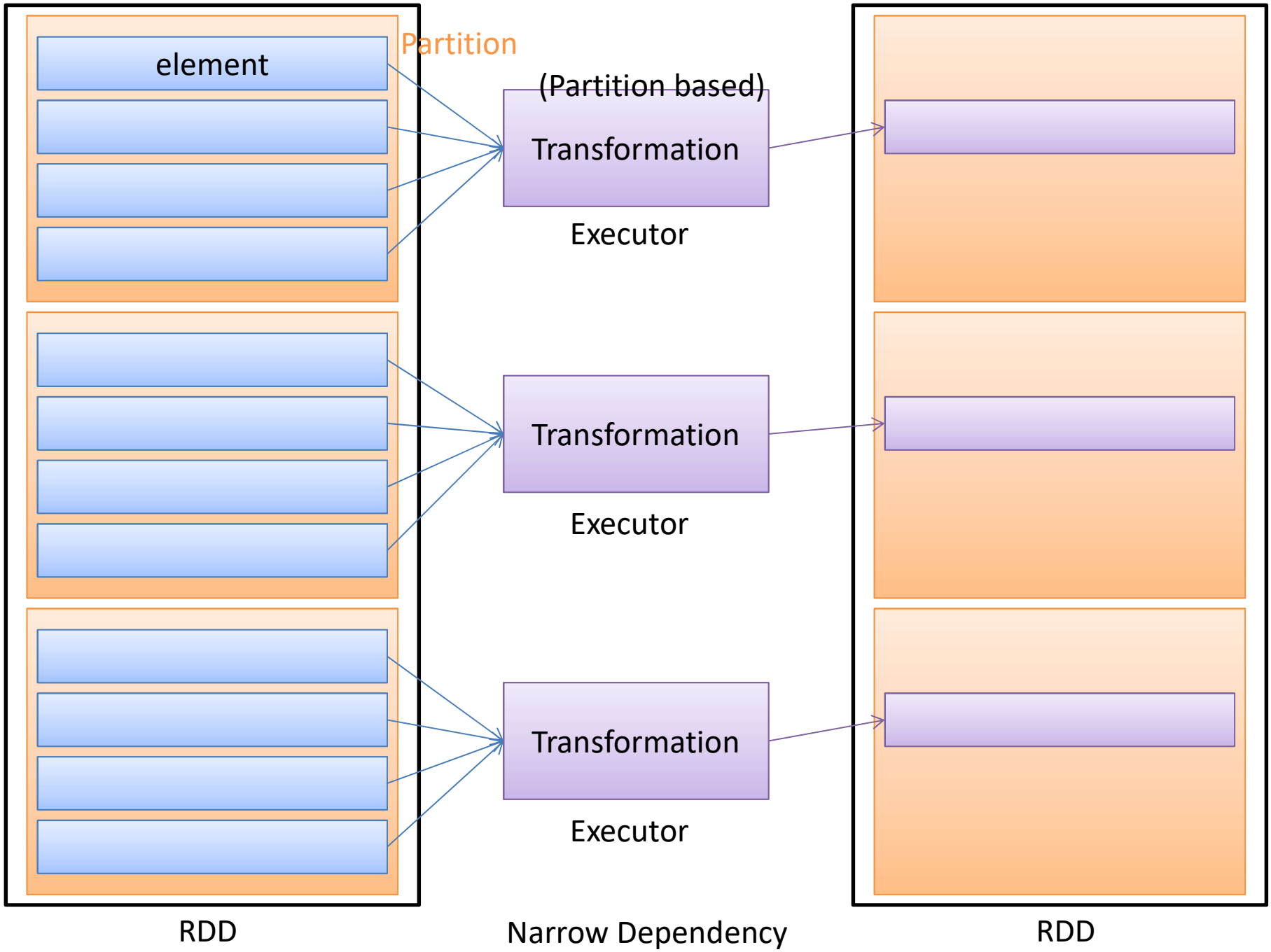

Driver Program
$ pyspark
$ spark-submit
Spark Context
Spark WebUI
Port:4040
collect()
Spark Worker
task
D
Spark Worker
A
Spark Worker
T
Spark Worker
A
Name Node
Standalone
YARN
Mesos
Spark Master
Cluster Manager

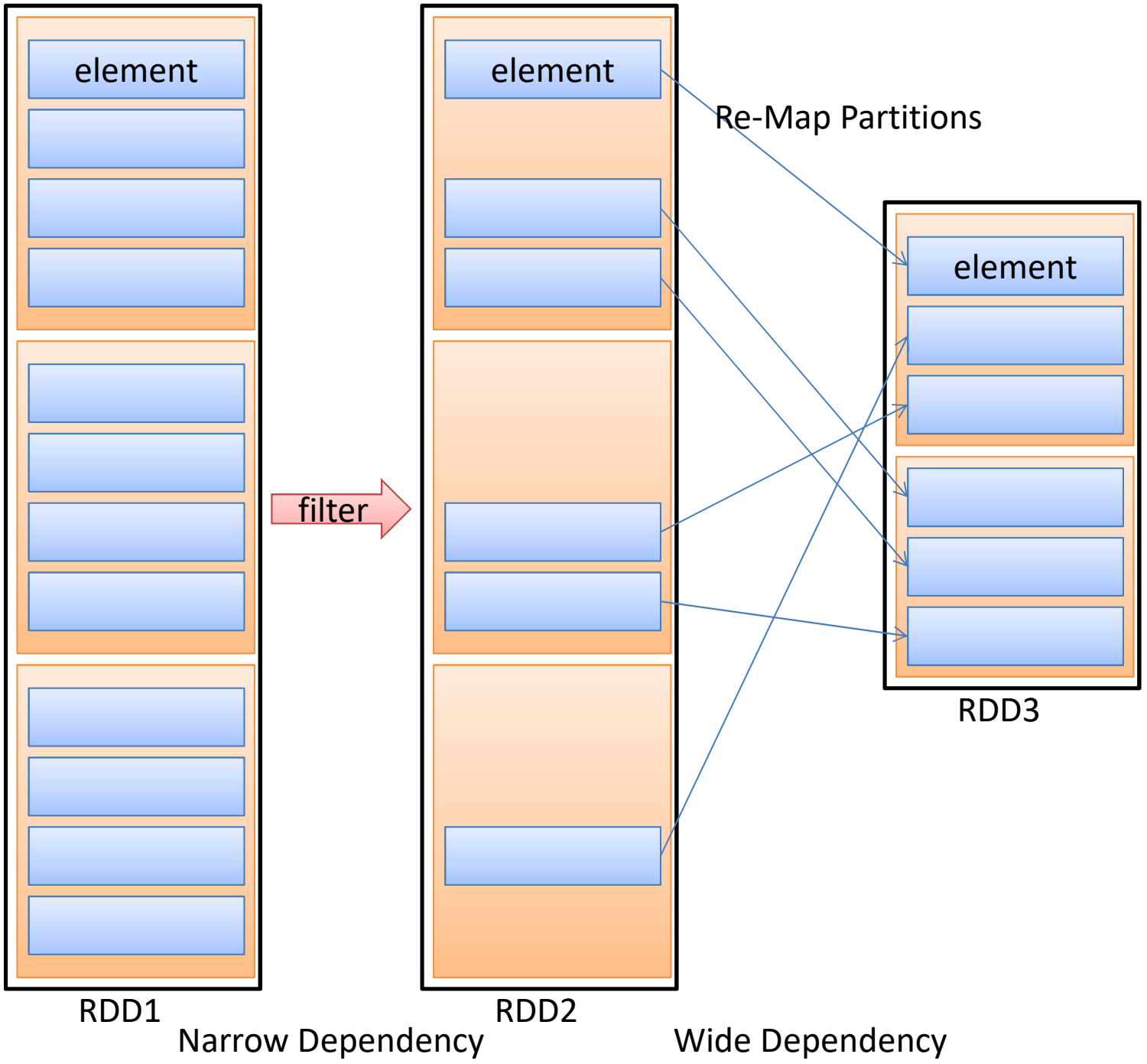

Driver Program
Spark Context
Standalone
YARN
Mesos
Spark Master
Cluster Manager
Spark Worker
D
Spark Worker
A
Spark Worker
T
Spark Worker
A
Name Node

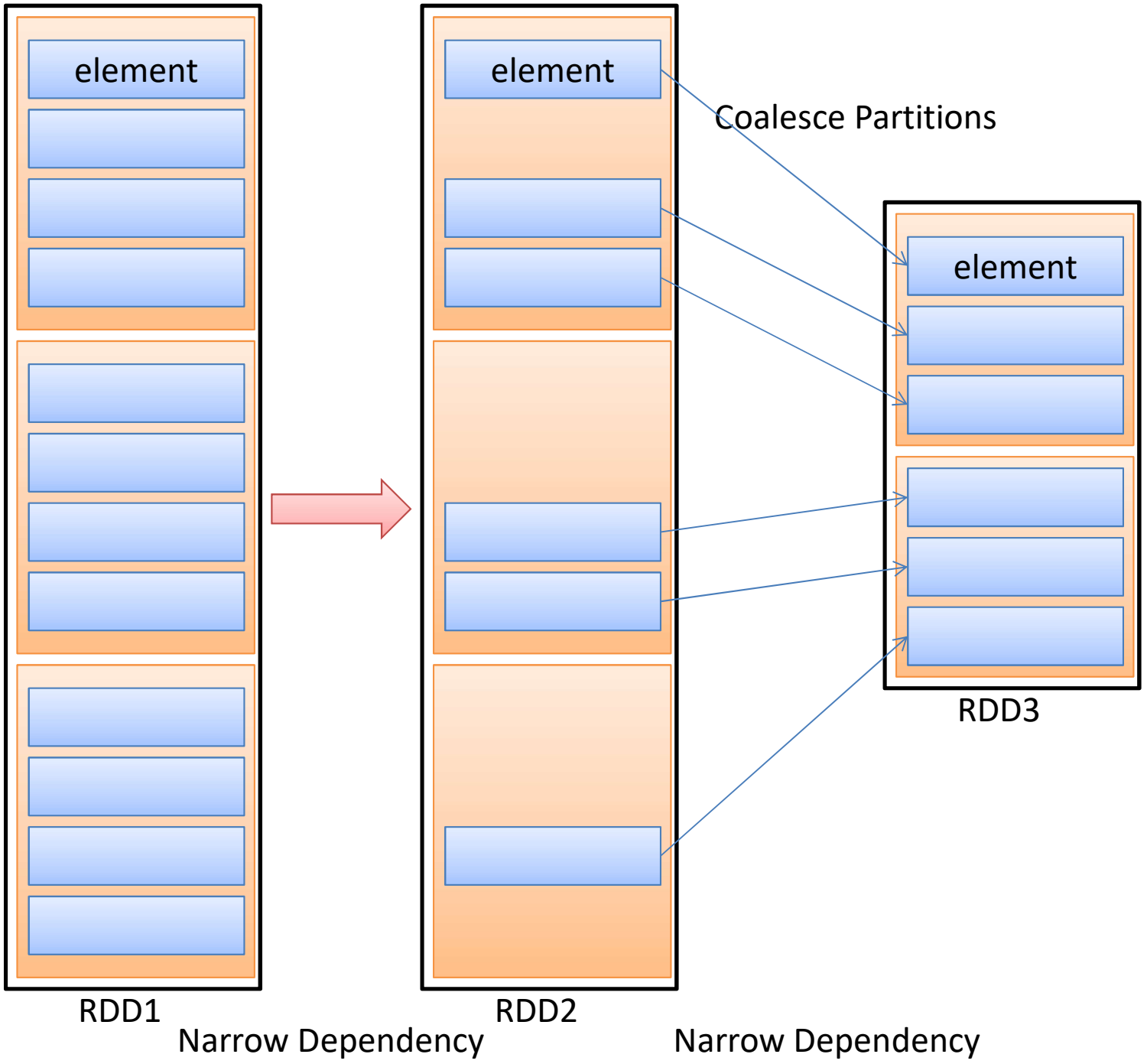

少
多
OFF Fatal Error Warn Info Debug Trace

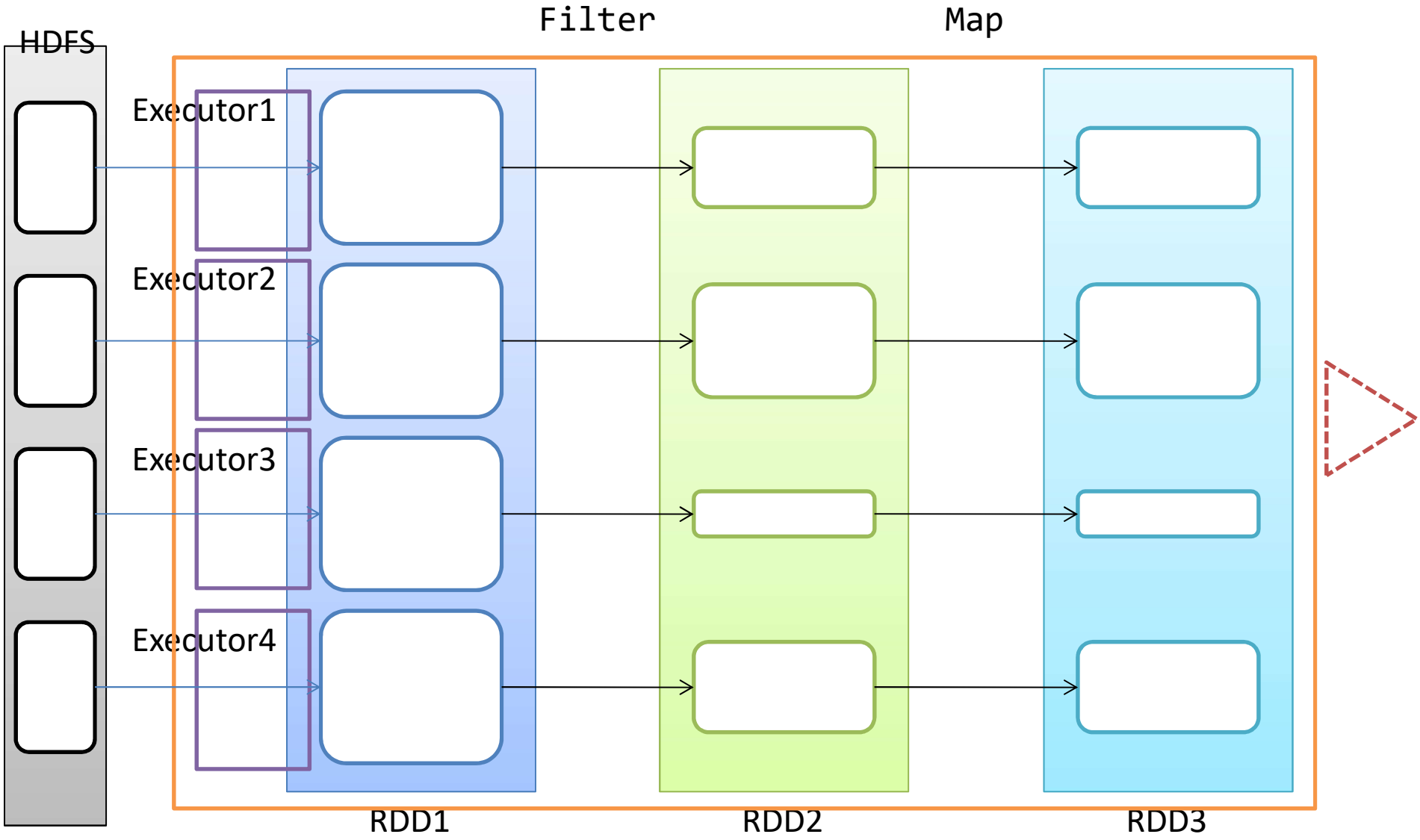

JAR
Python file
Driver Program
SparkContext
Cluster Manager
Spark standalone
Apache Mesos
Yarn
Worker Node
Executor
Cache
Task
Task
Worker Node
Executor
Cache
Task
Task

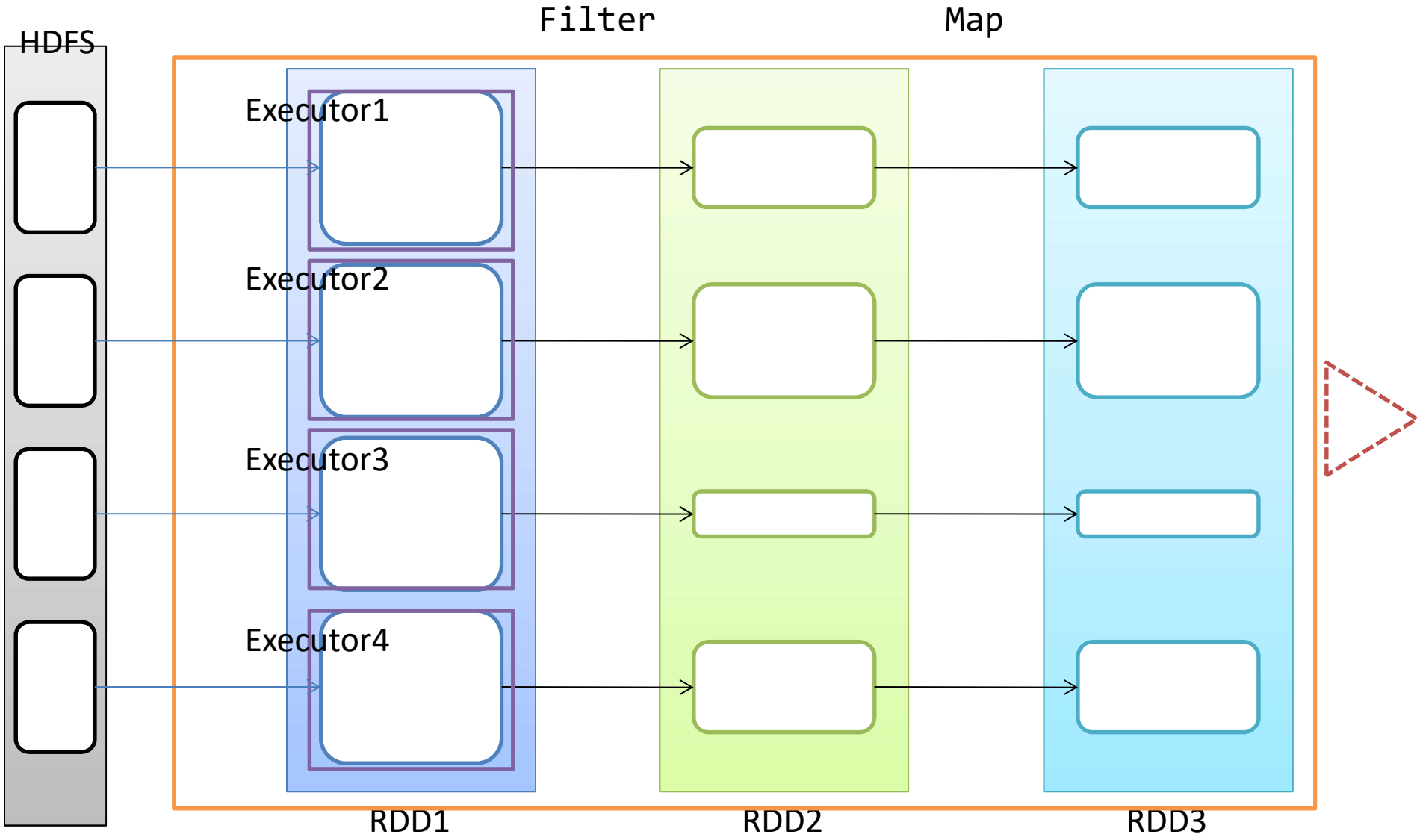

element
Partition
(Element based)
Transformation
Executor
element
Transformation
Executor
Transformation
Executor
RDD1
Narrow Dependency
RDD2

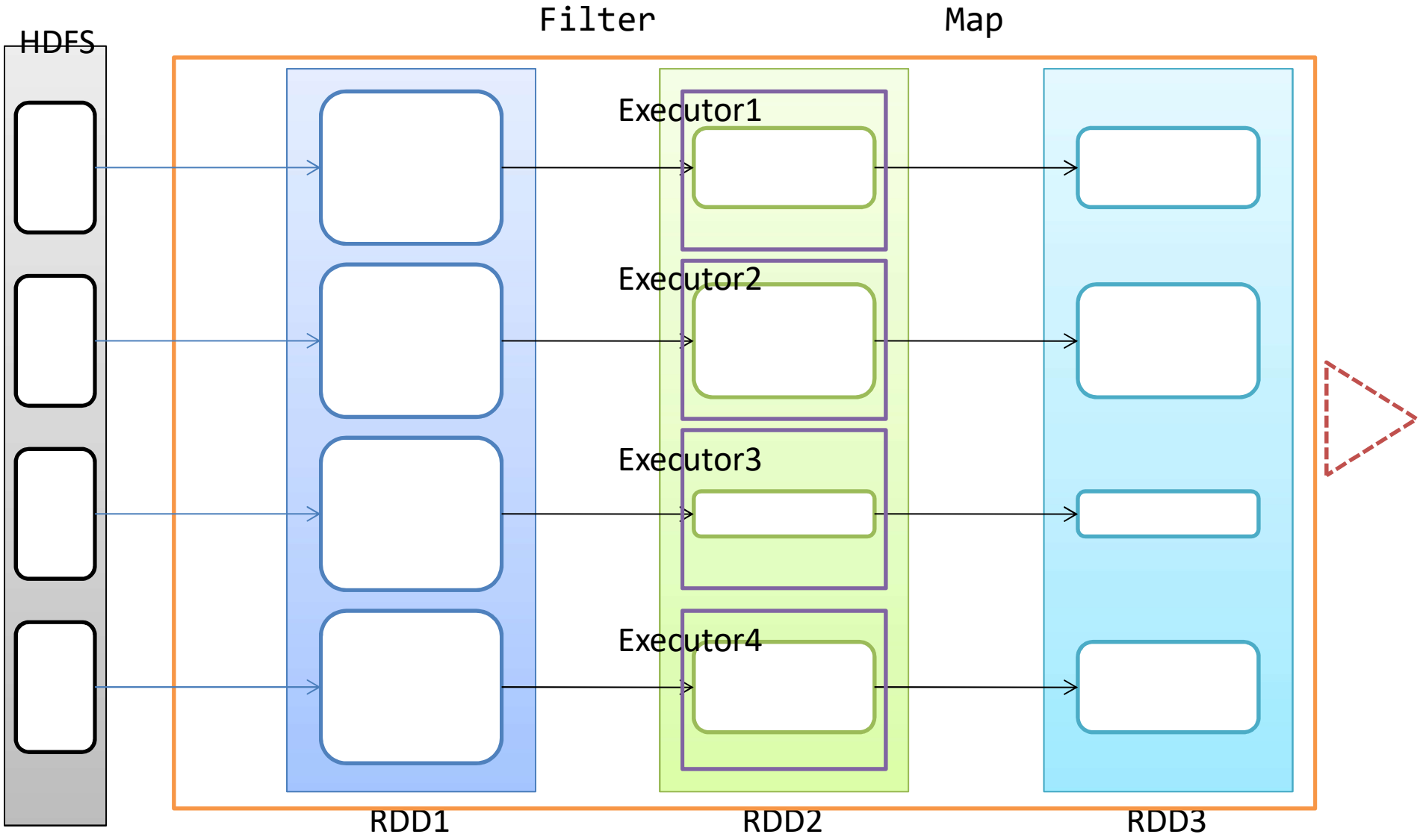

element
Partition
(Partition based)
Transformation
Executor
Transformation
Executor
Transformation
Executor
RDD
Narrow Dependency
RDD

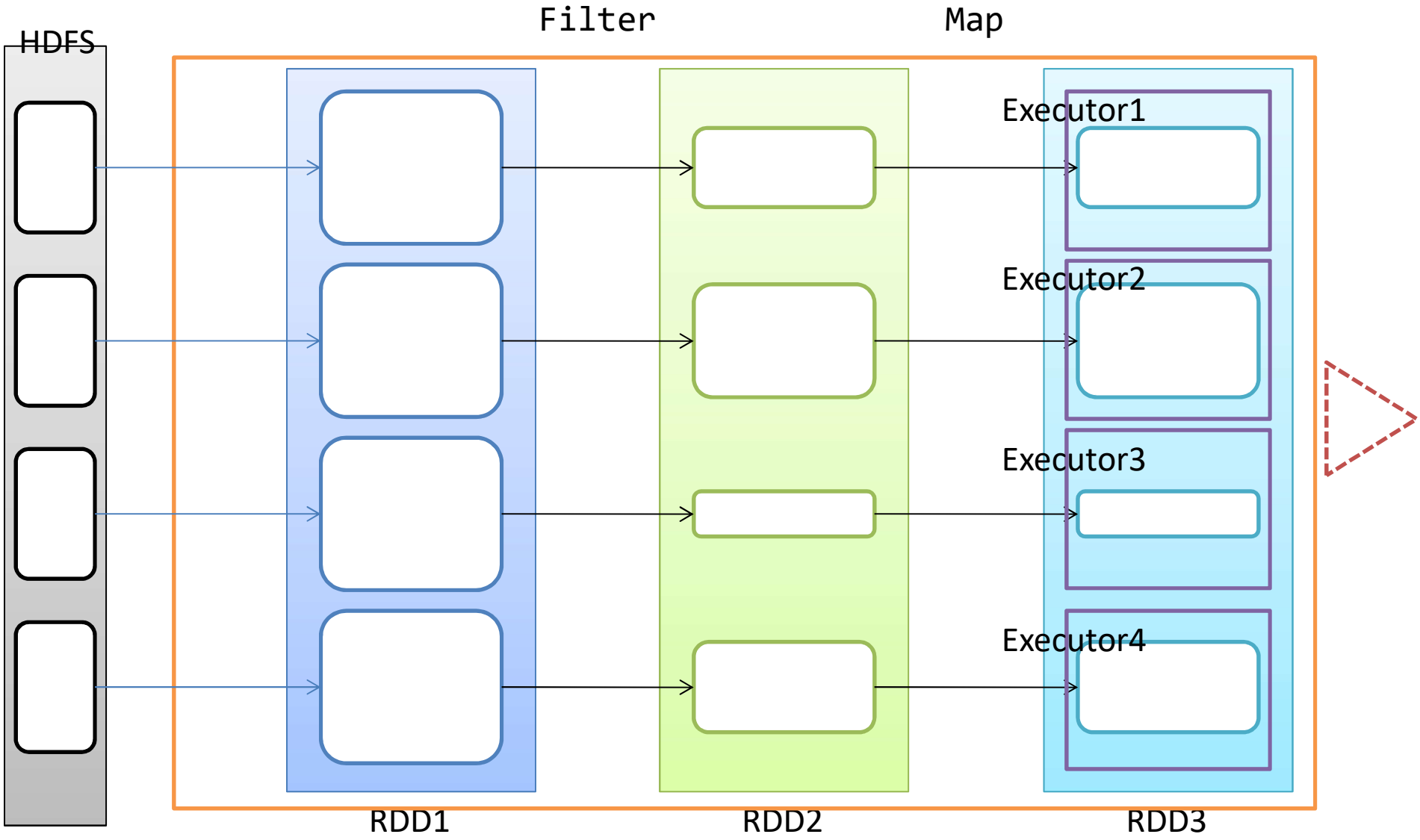

element
element
Re-Map Partitions
element
filter
RDD1
RDD2
RDD3
Narrow Dependency
Wide Dependency

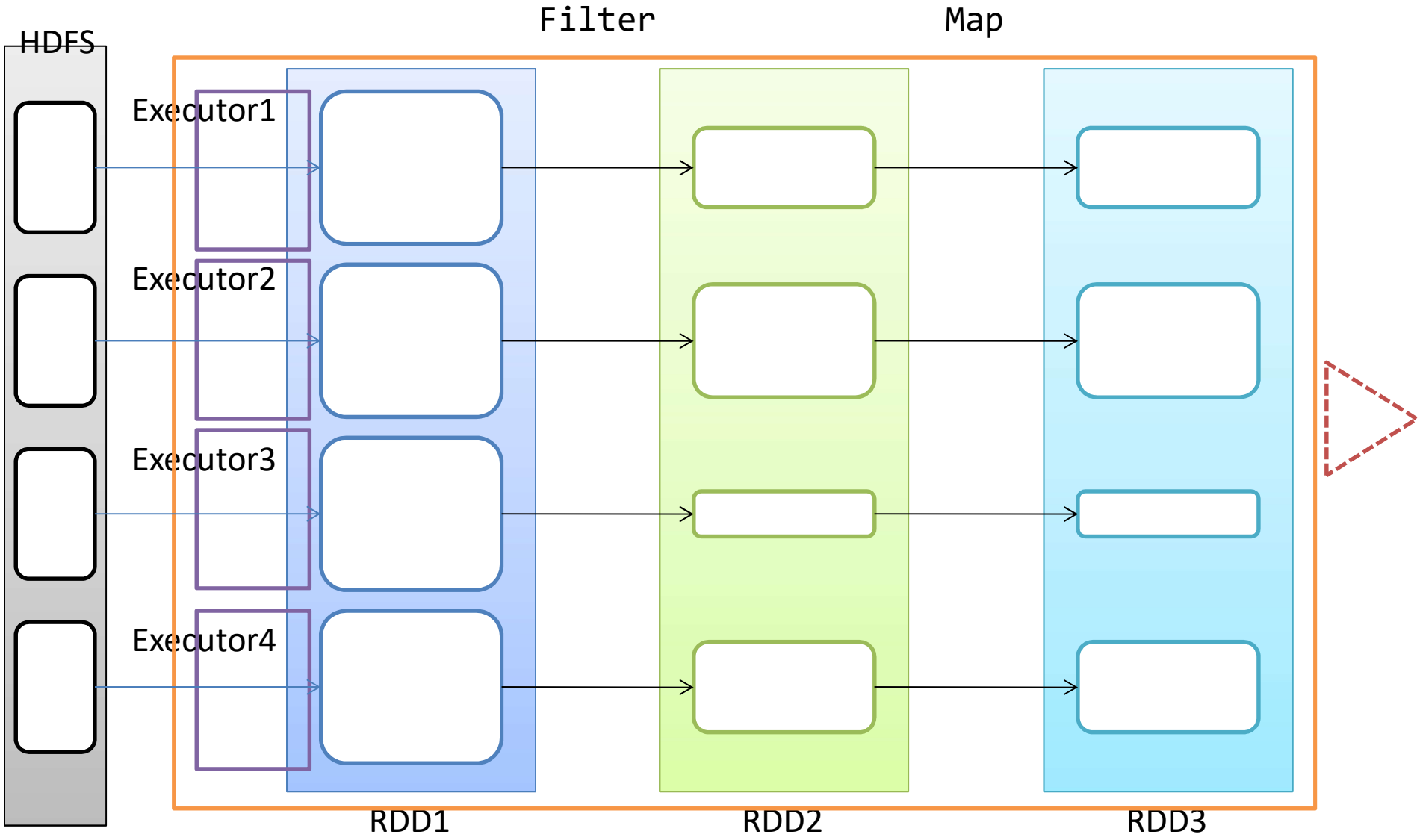

element
element
Coalesce Partitions
element
RDD1
RDD2
RDD3
Narrow Dependency
Narrow Dependency

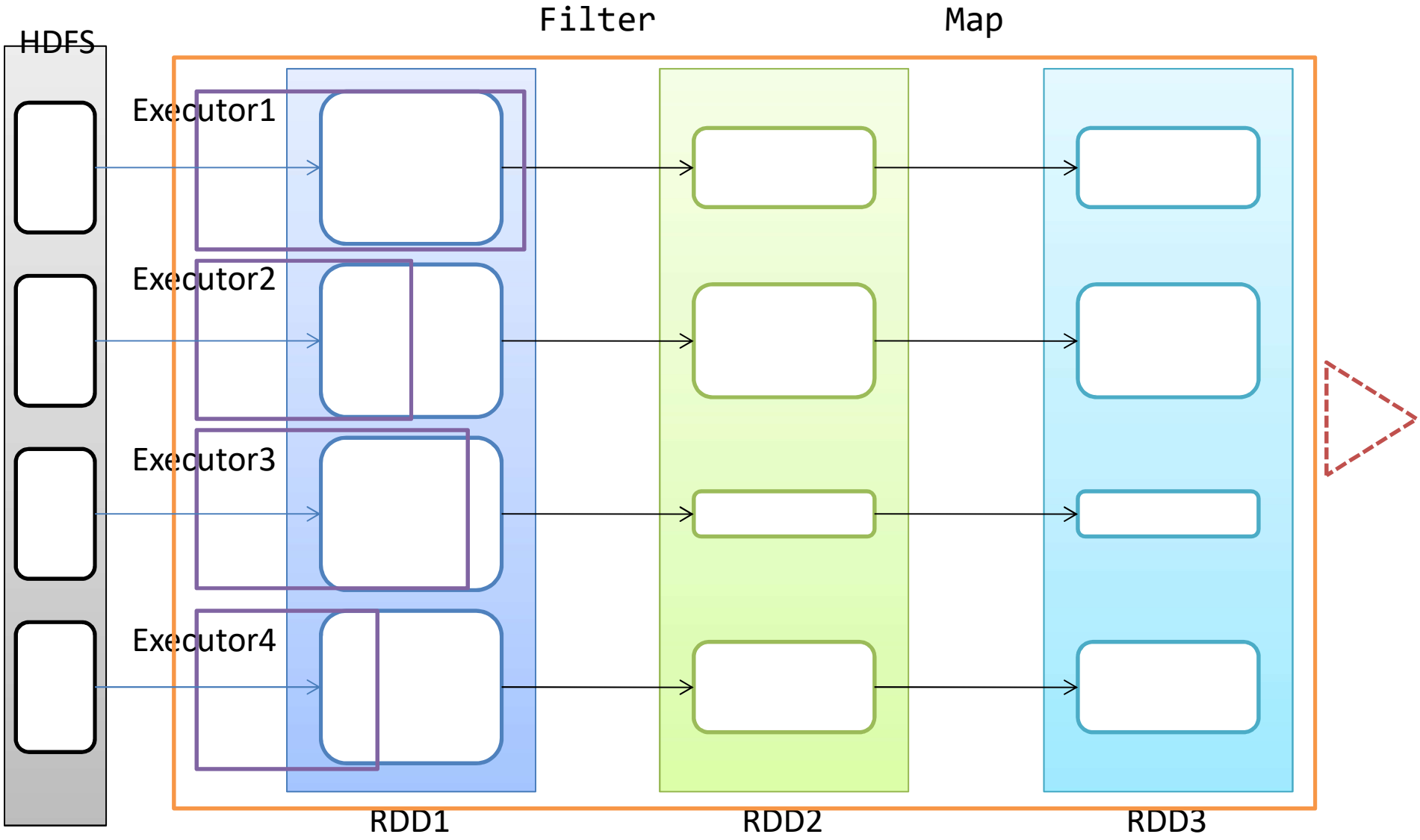

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

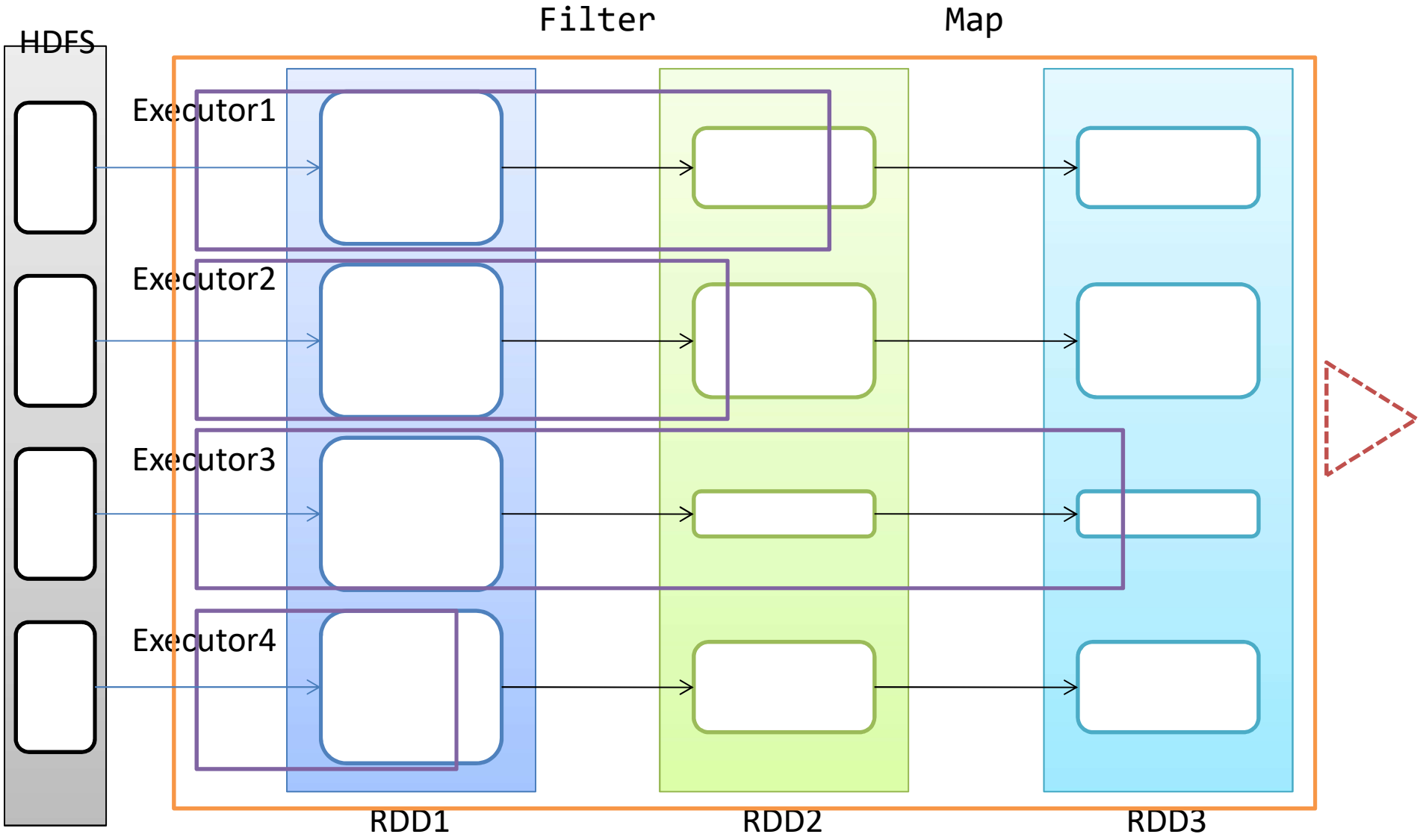

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

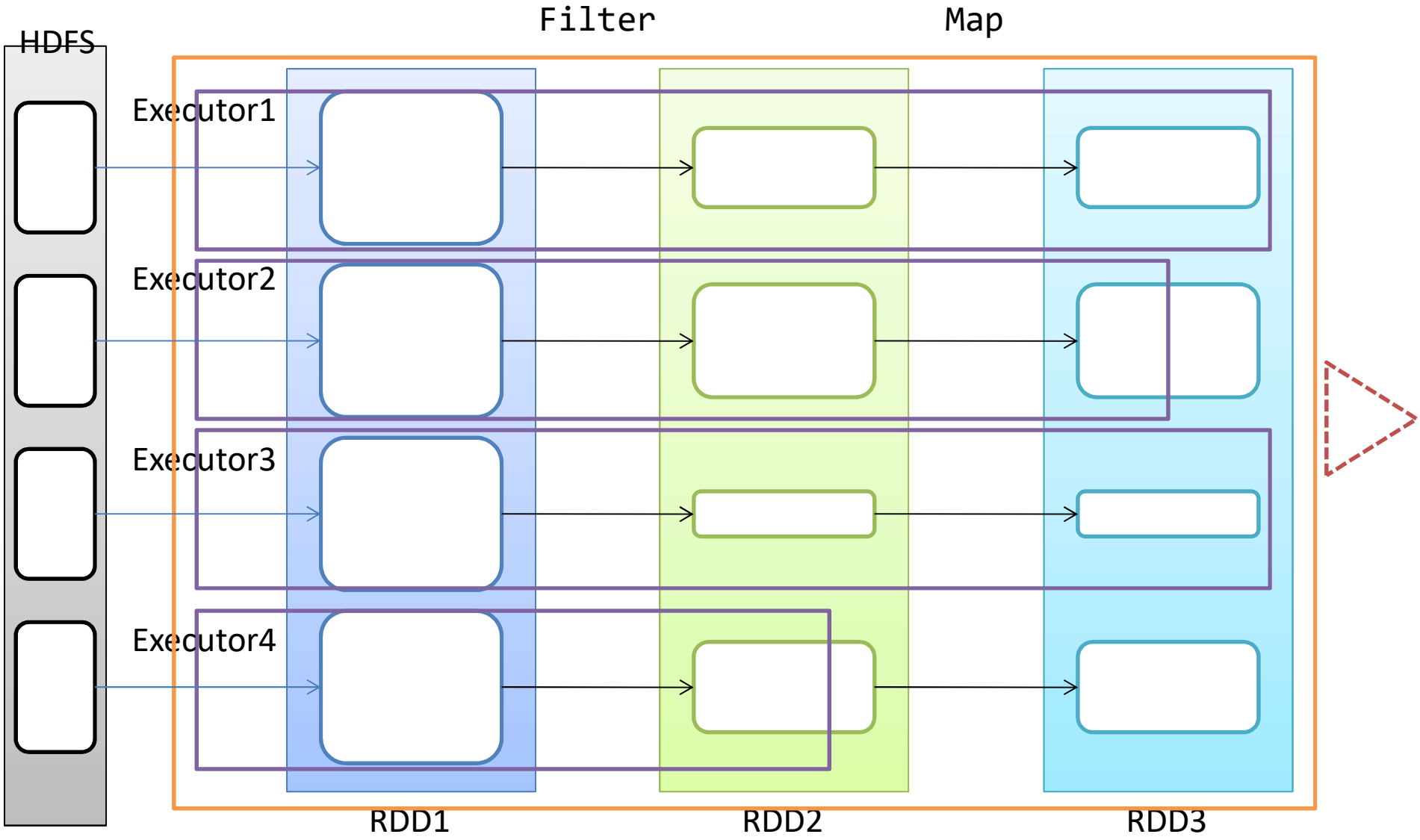

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

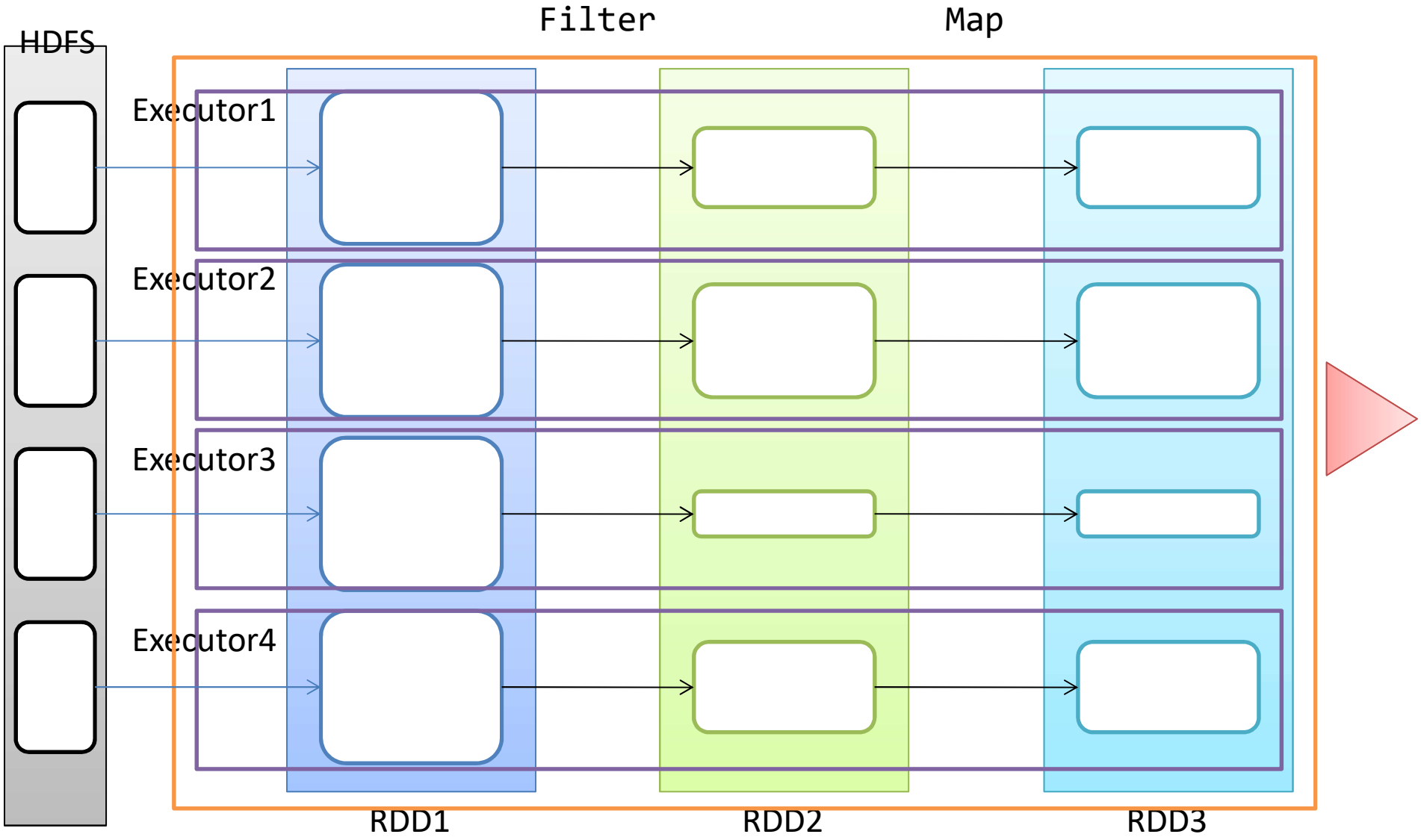

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

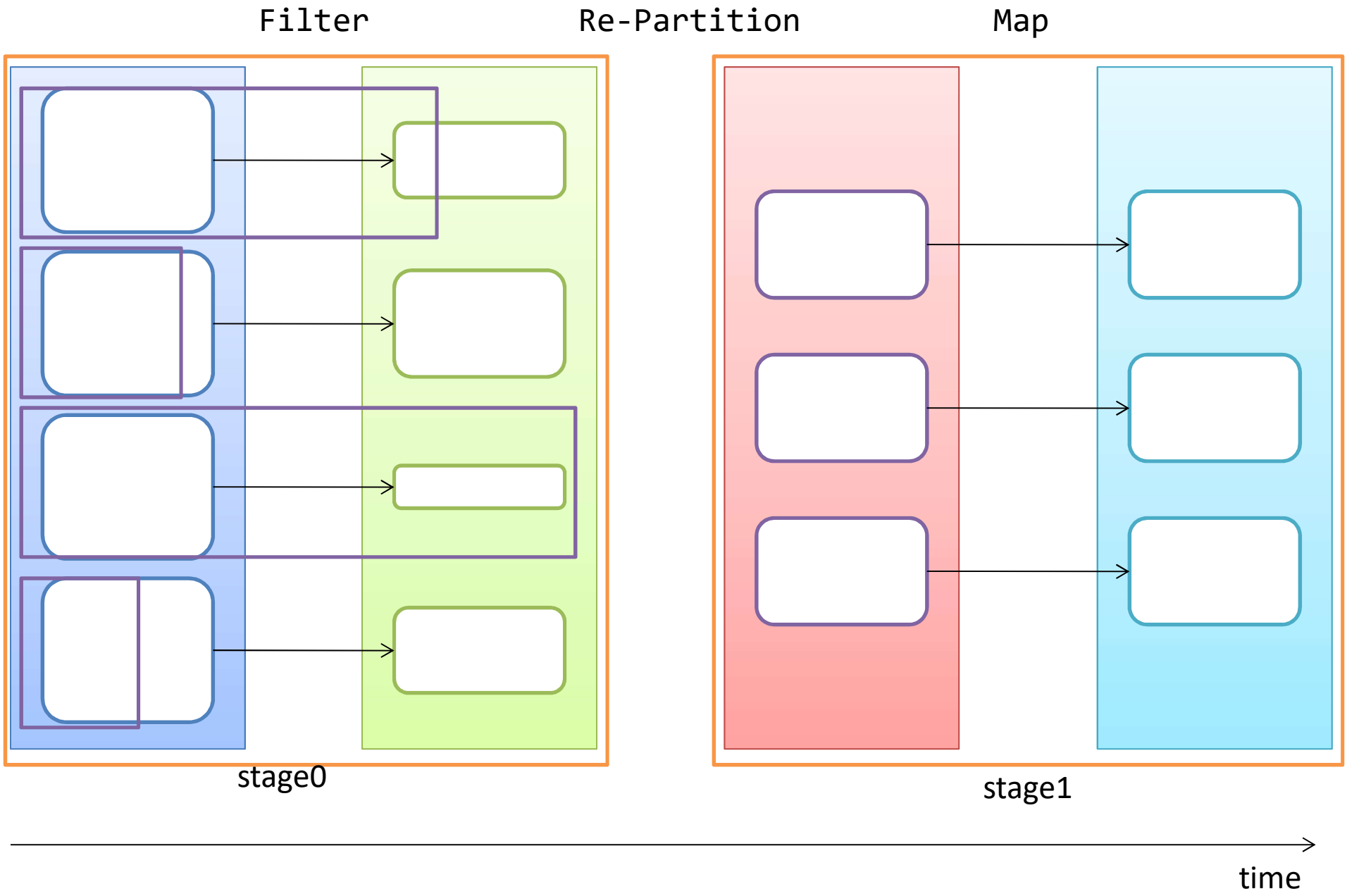

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

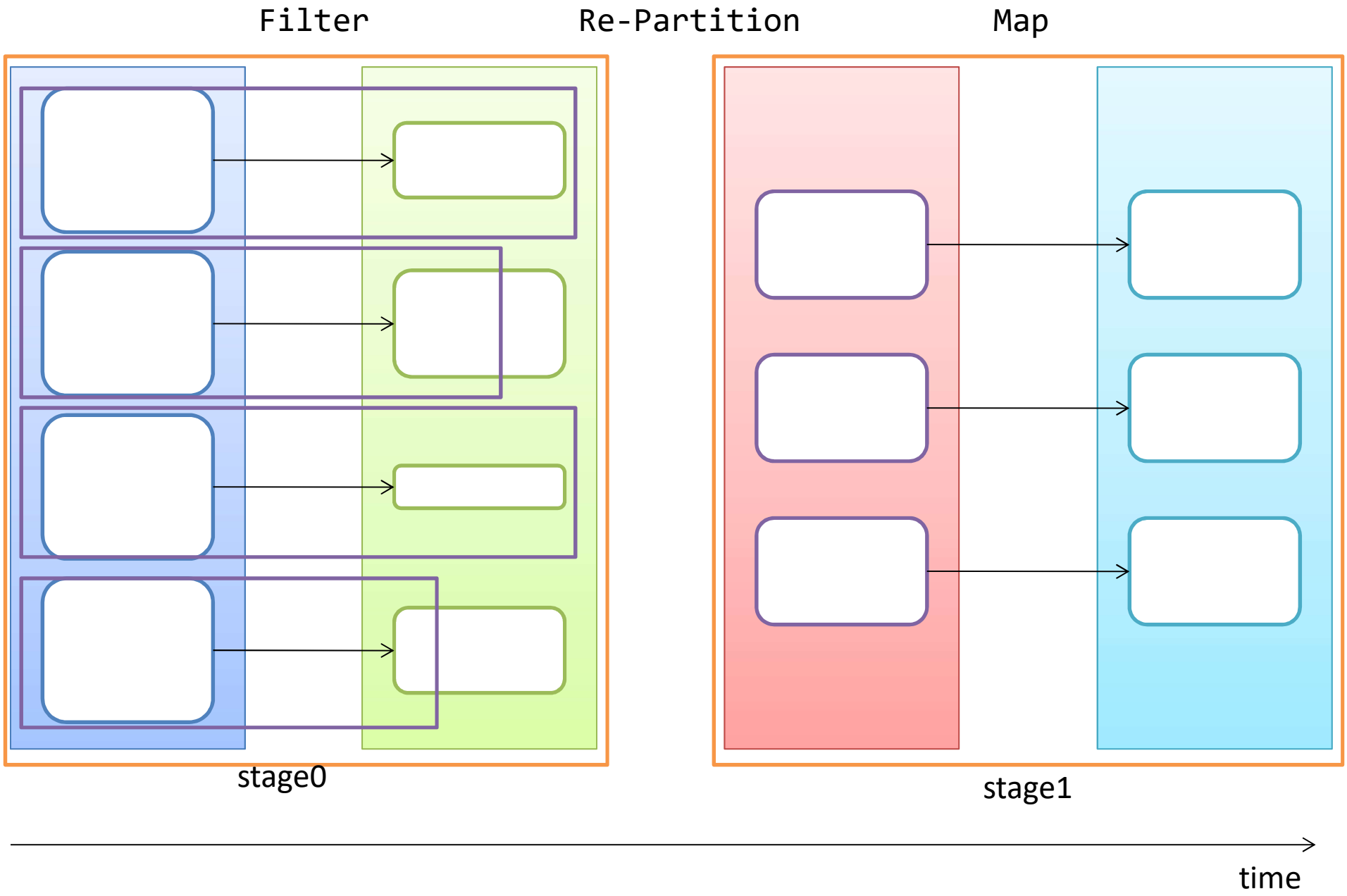

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

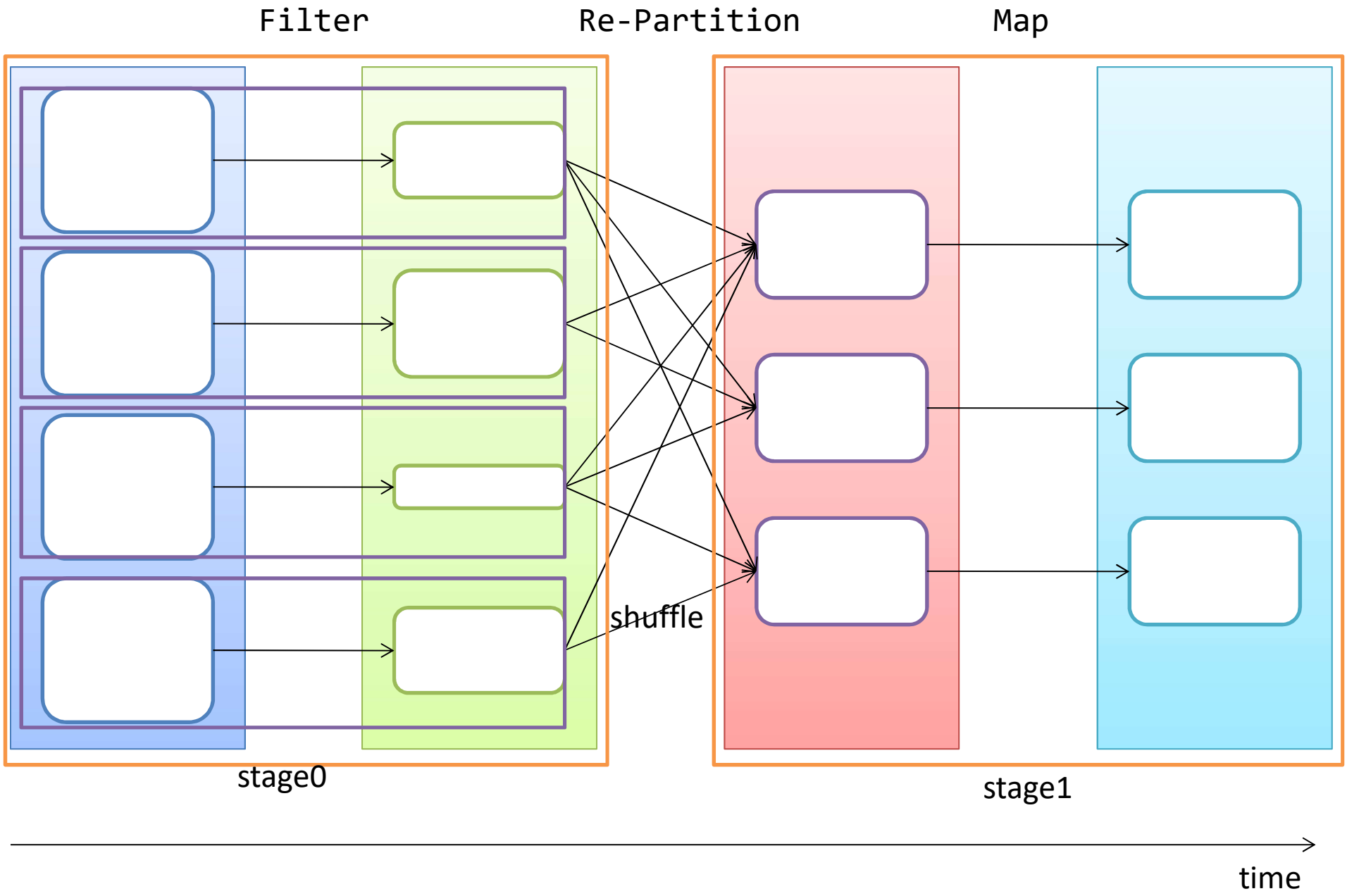

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3

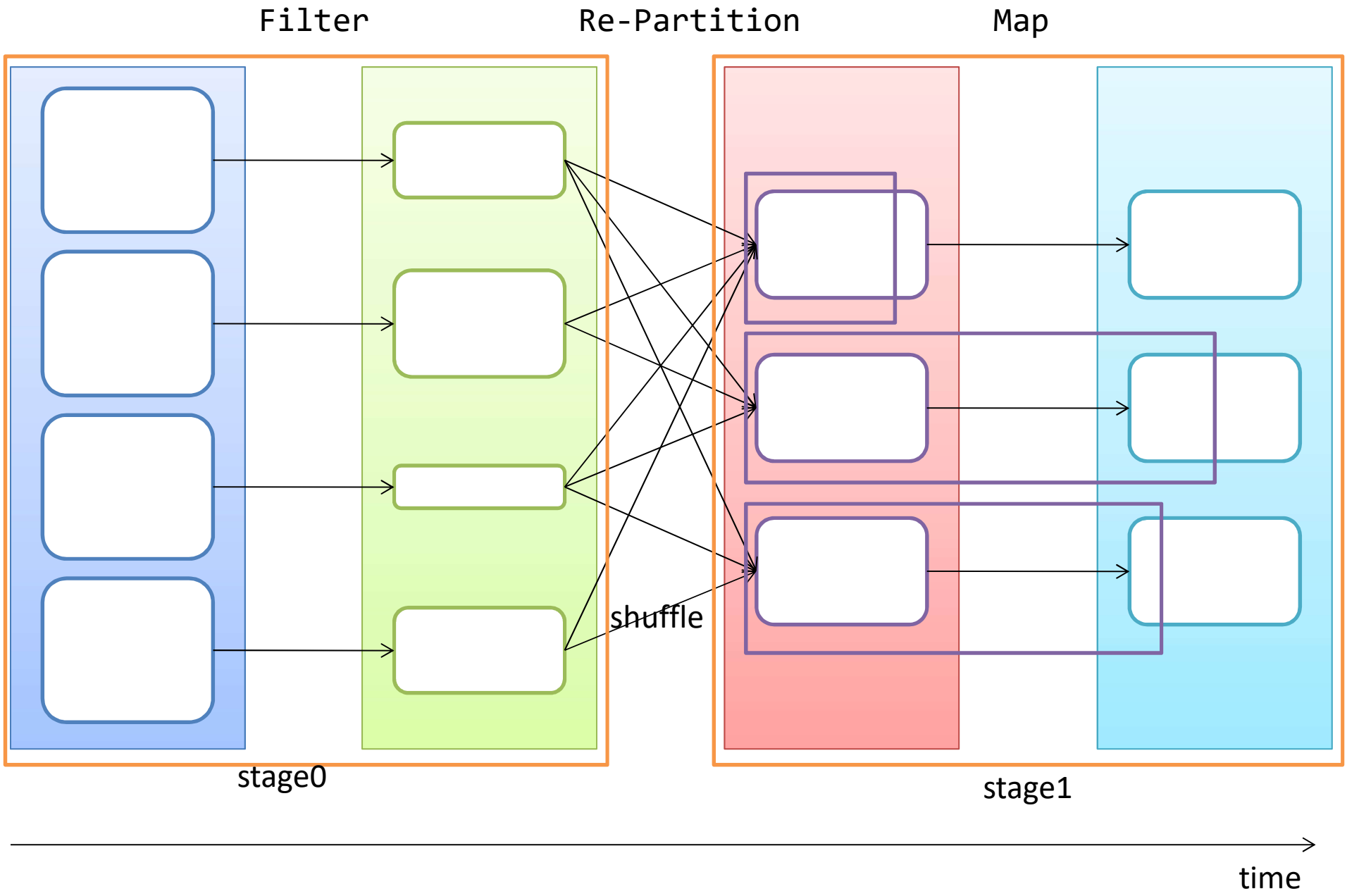

HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3


HDFS
Filter
Map
Executor1
Executor2
Executor3
Executor4
RDD1
RDD2
RDD3


Filter
Re-Partition
Map
stage0
stage1
time

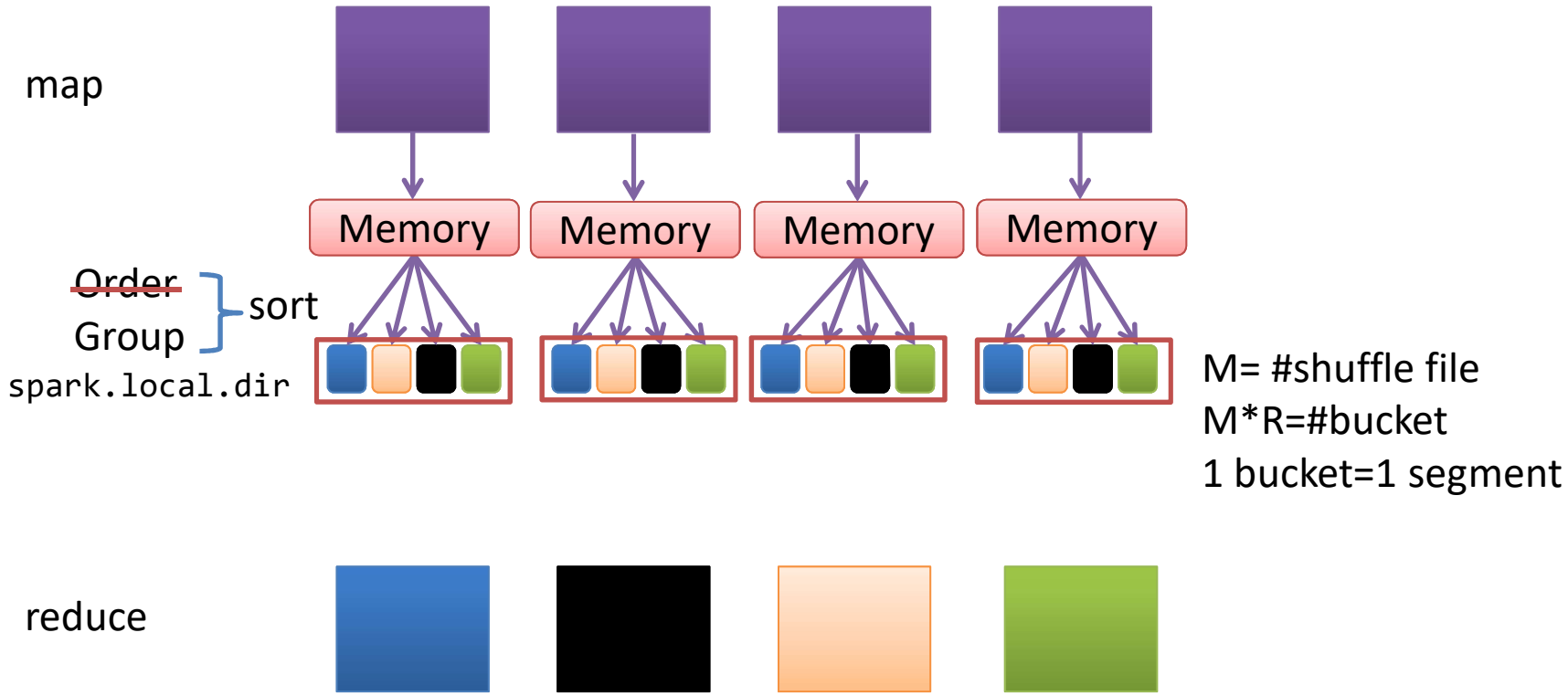

Filter
Re-Partition
Map
stage0
stage1
time


Filter
Re-Partition
Map
shuffle
stage0
stage1
time

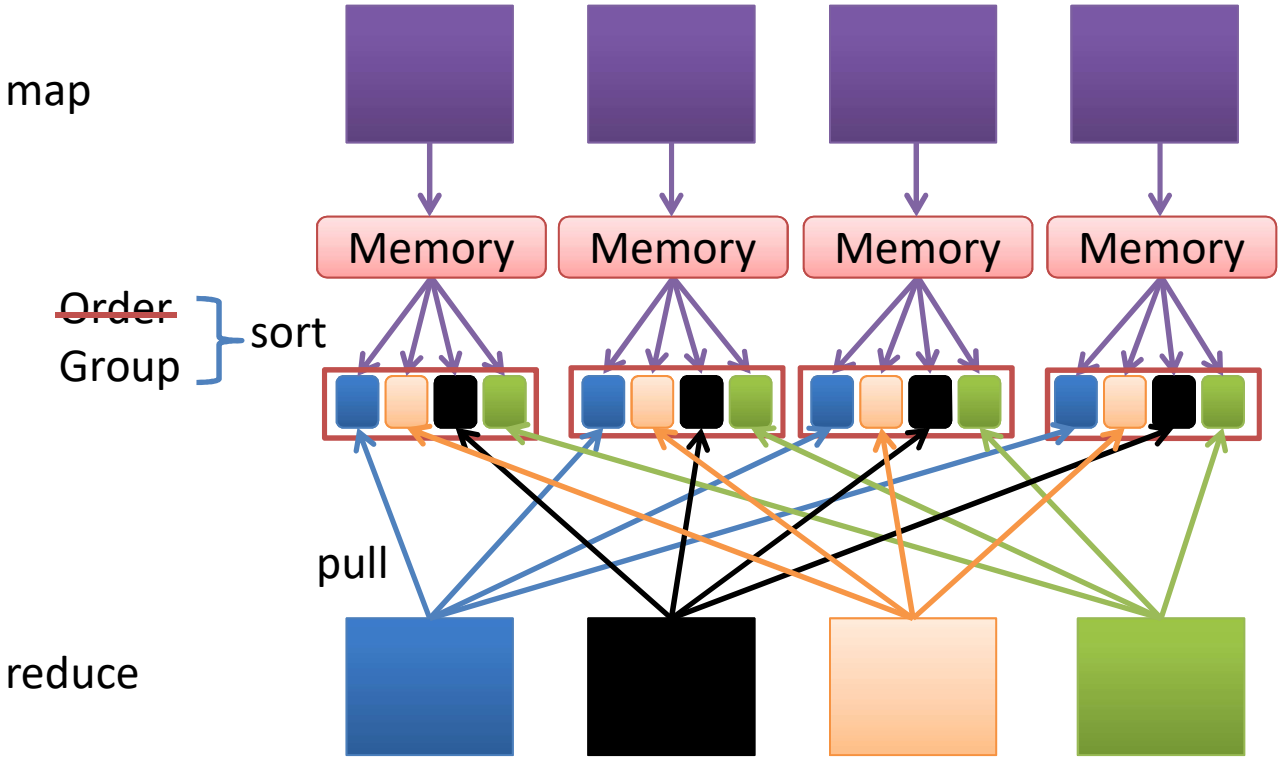

Filter
Re-Partition
Map
shuffle
stage0
stage1
time

Spark 1.1 Sort-based shuffle

Core=4 => 4 Tasks running



Spark 1.1 Sort-based shuffle



- Application
- Driver Program
- Cluster Manager
- Worker Node
- Executor
- Job
- Stage
- Task
- Partition

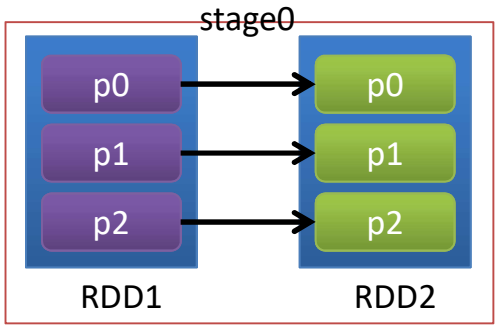
Job

- 在一個Spark Application中，以Action操作為邊界劃分為多個Job。
- 一個job中，可包含多個Transformation操作與一個Action操作。
- 一個job可由多個stage組成

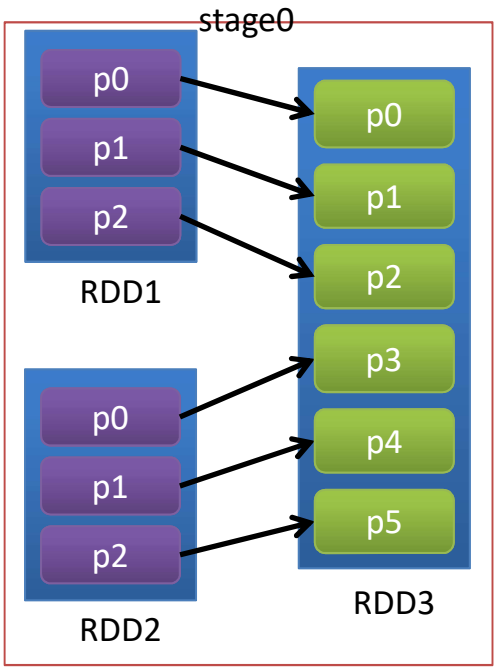
Dependency(相依)

- Narrow Dependency(窄相依)
  - 父RDD的一個partition最多被一個子RDD partition所依賴
- Wide Dependency(寬相依)
  - 父RDD的一個partition被多個子RDD partition所依賴，此時會依據元素的Key值進行混洗。
  - mapPartitions、repartitionAndSortWithinPartitions、sortBy、repartition、~~coalescce~~、ByKey operations(groupByKey、reduceByKey)、join operations(cogroup、join)
- 寬相依是Job切分stage的參考點
- 窄相依之間可以進行pipeline操作

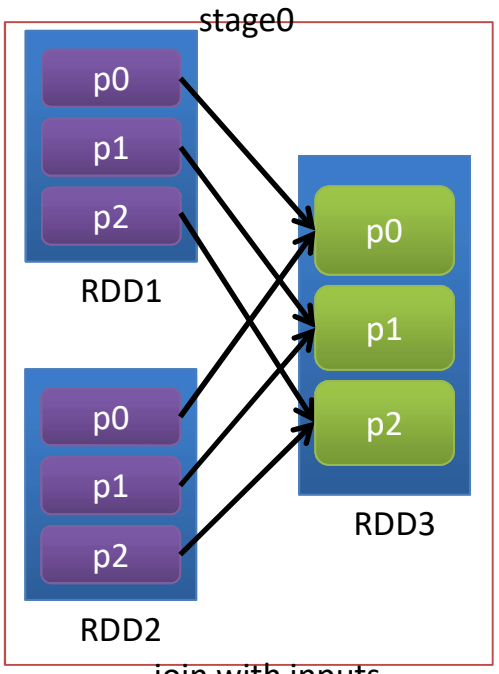
Narrow Dependency(窄相依)



map、filter

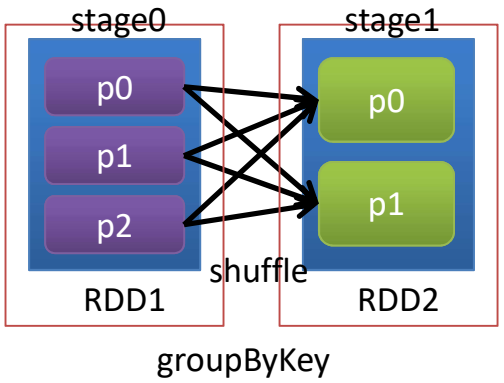


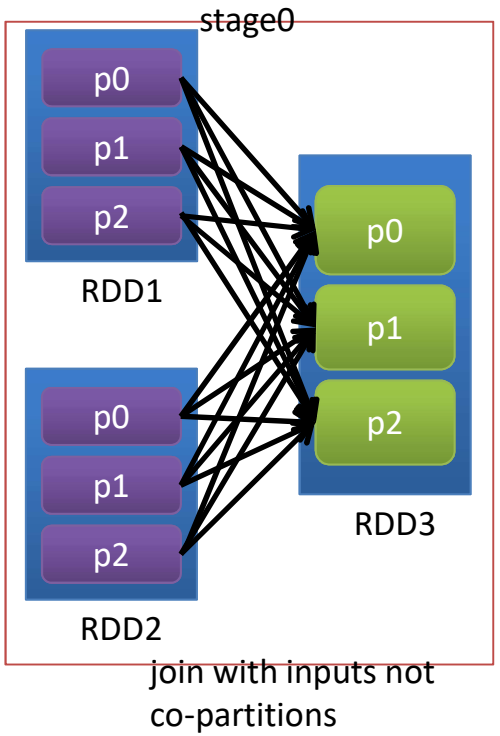
union



join with inputs
co-partitions

Wide Dependency(宽相依)



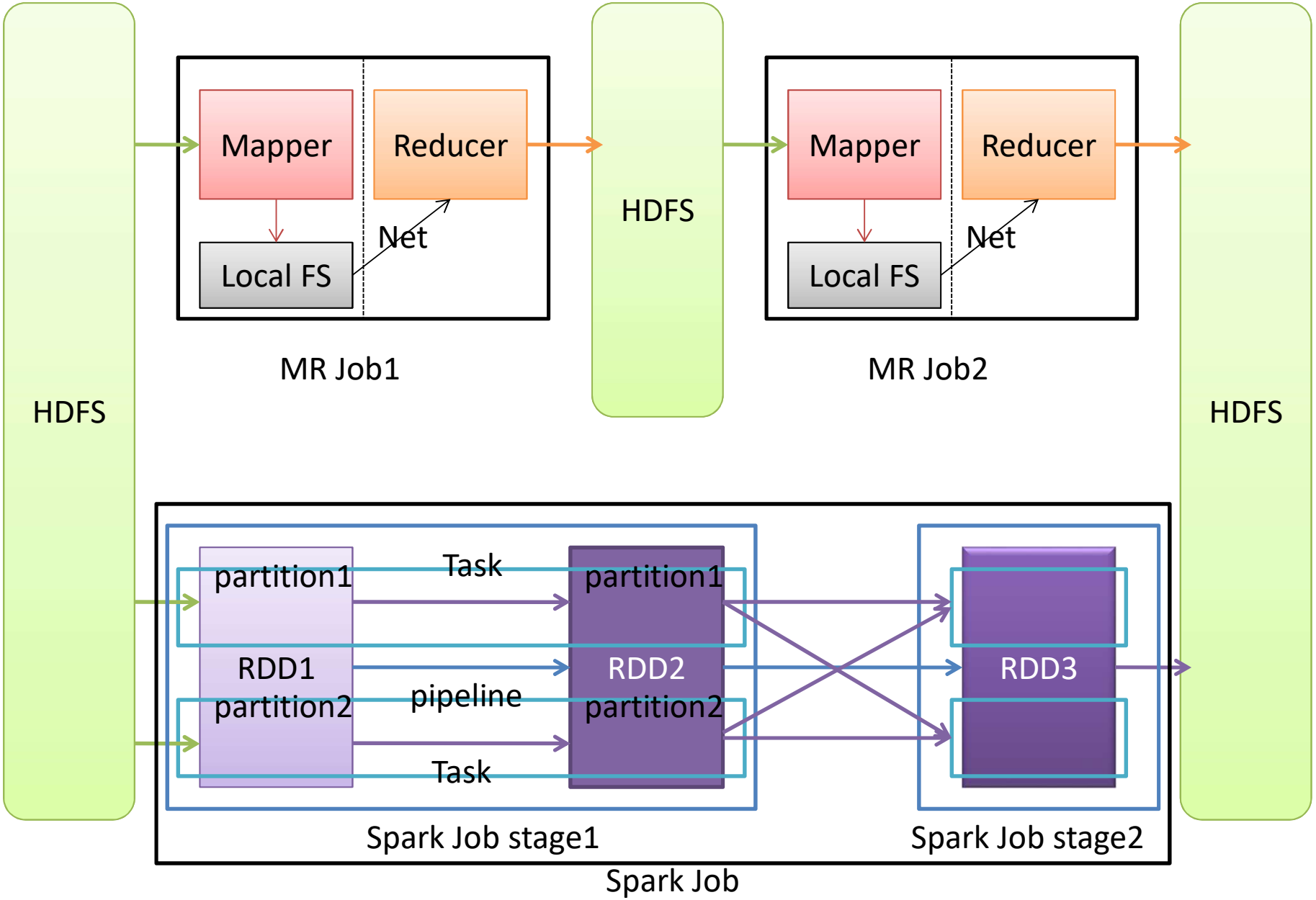


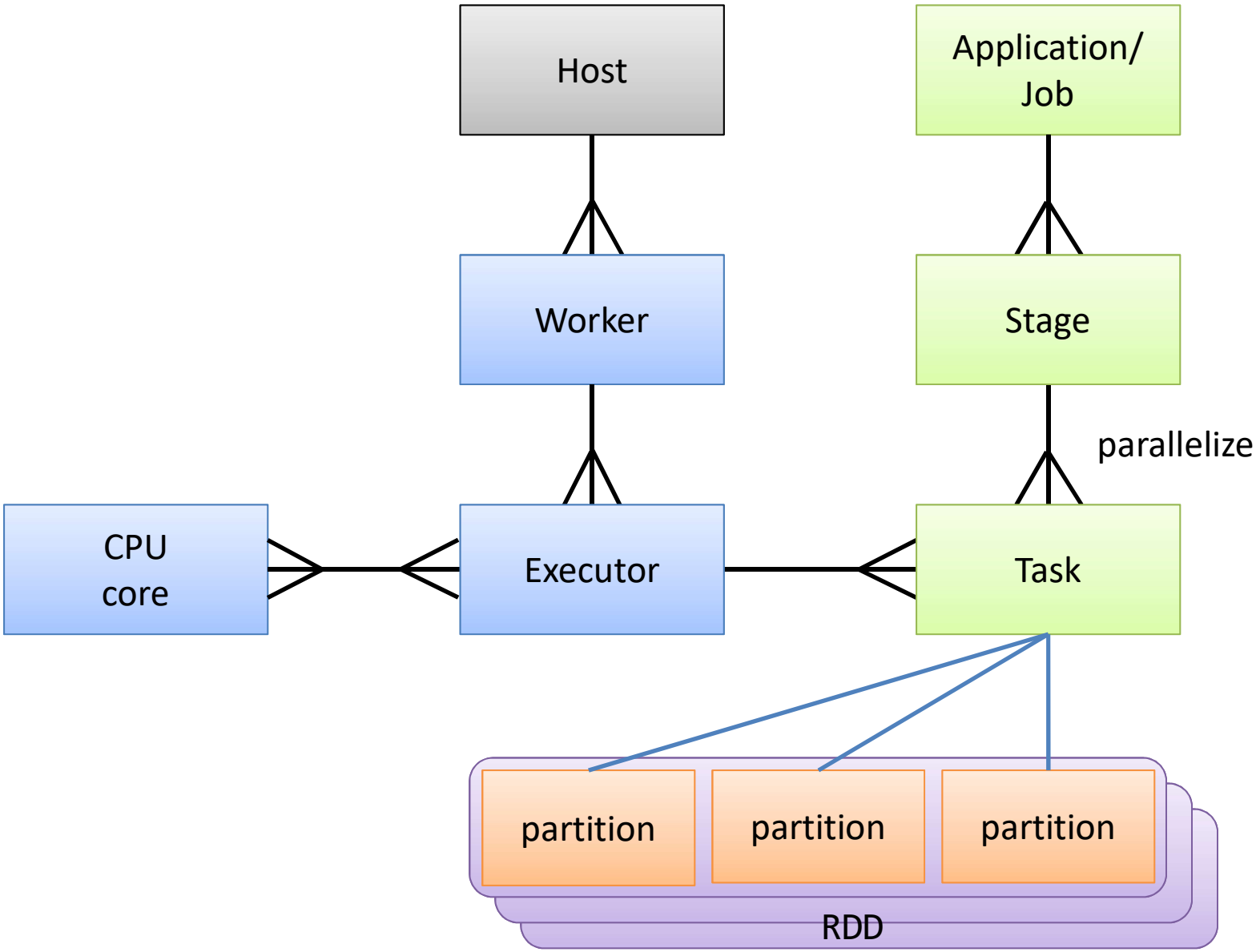
Stage

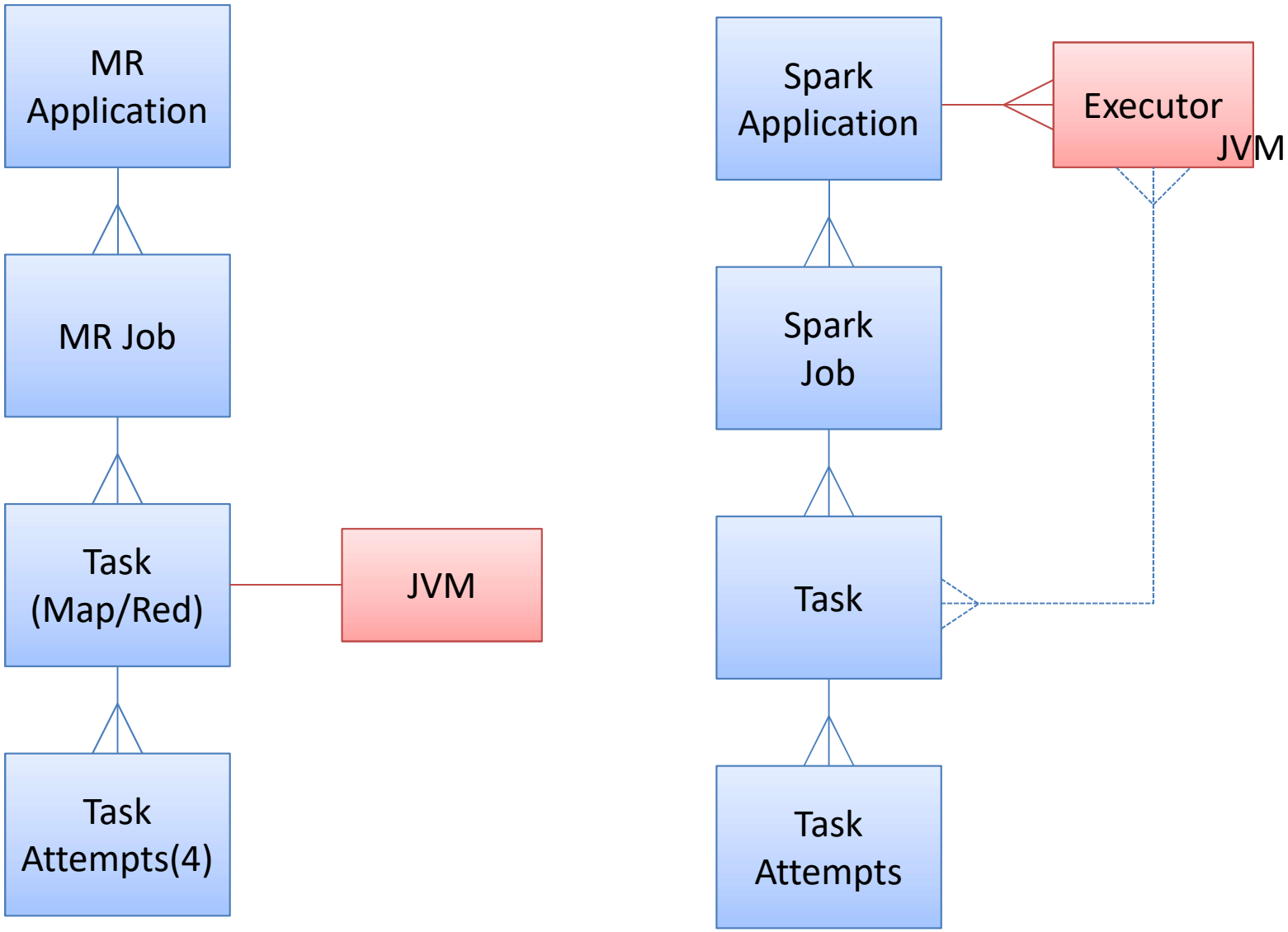
- 一個Spark job中以需要shuffle操作的轉換為邊界劃分出不同階段(stage)
- 每個階段包含一組管線化的窄相依操作
- 不同階段之間不能平行地同時進行，因為位在後面的階段需要位在前面的階段的執行結果。

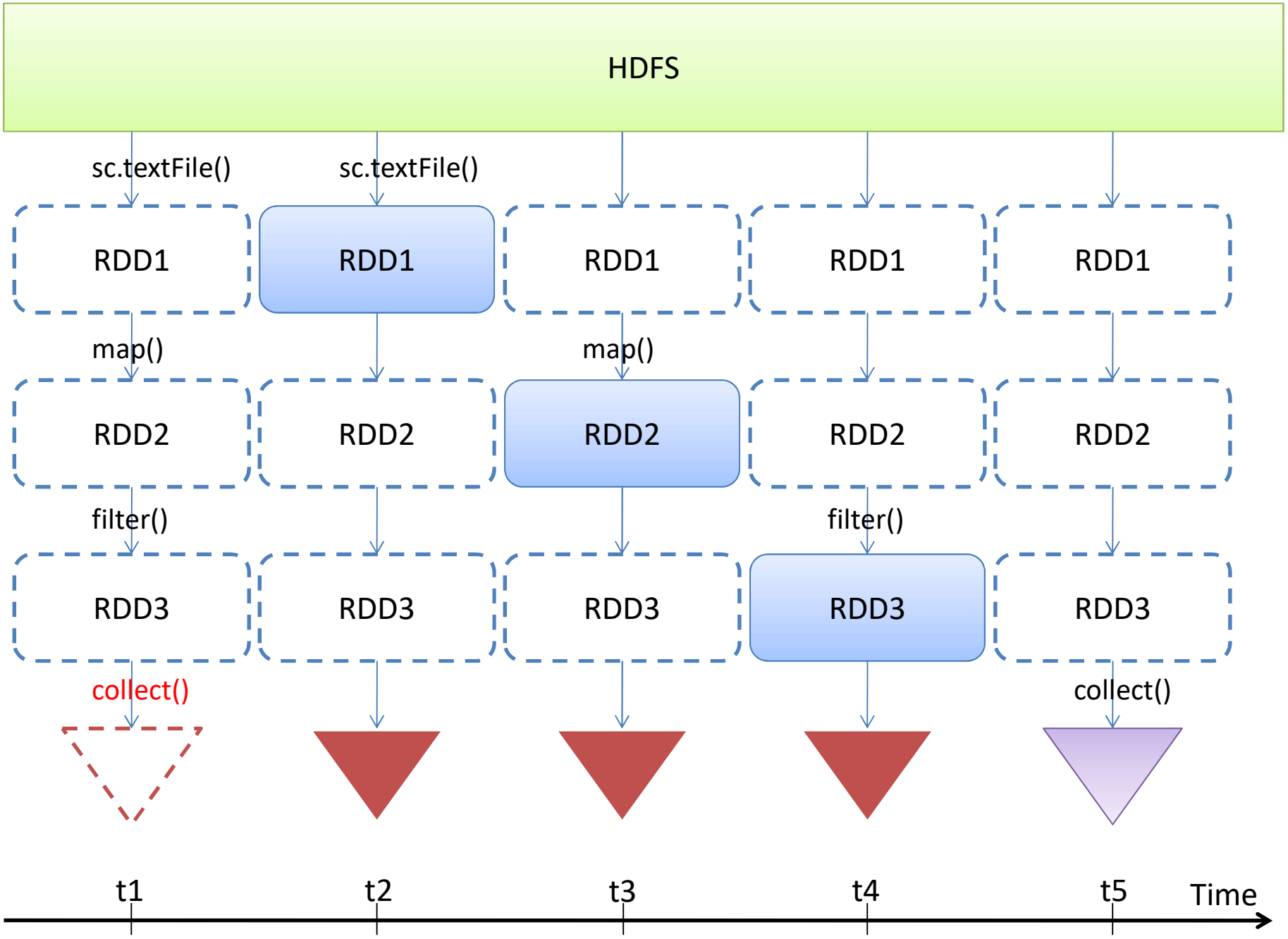
Task

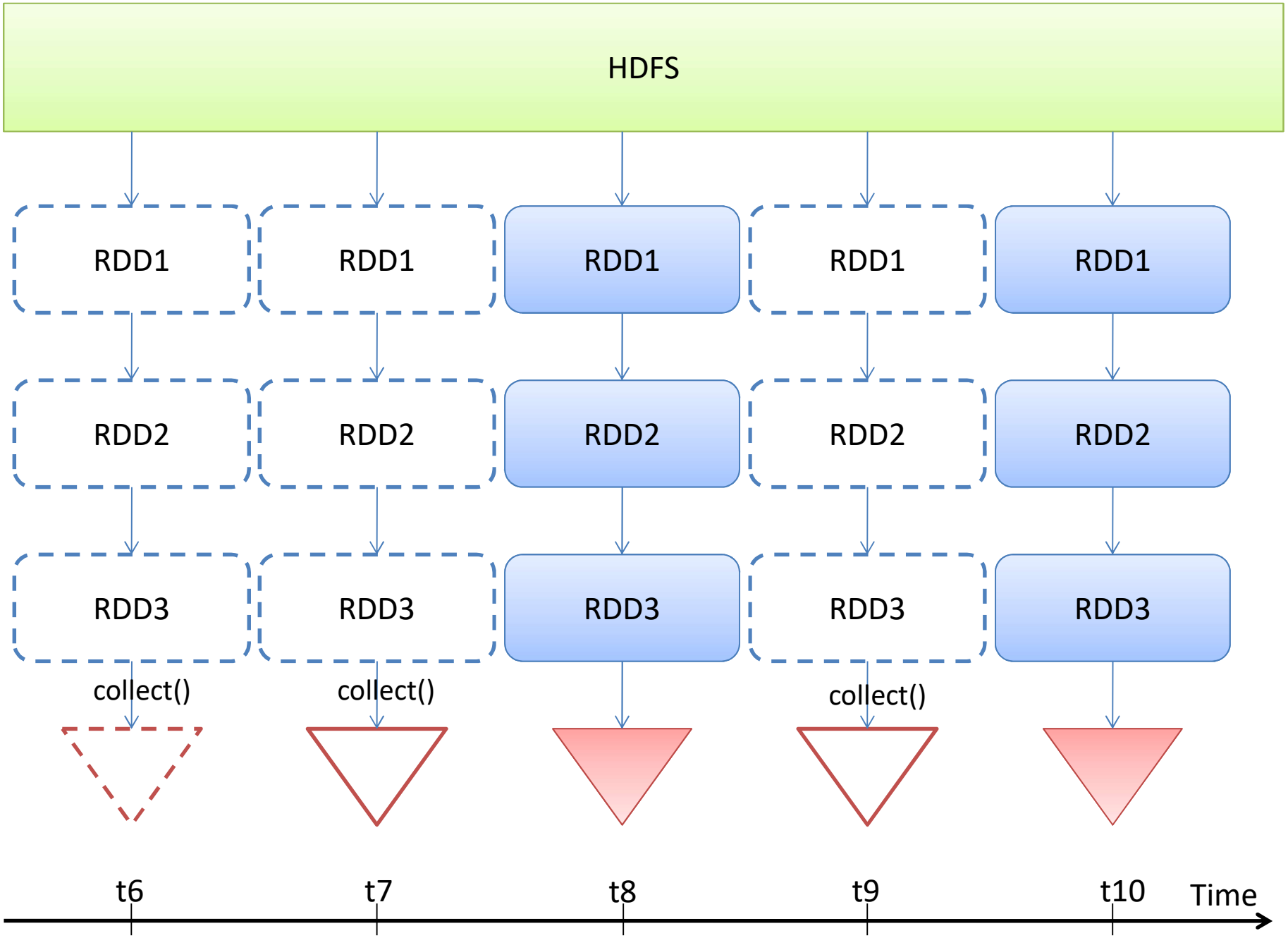
- 在一個stage的RDDs可以序列地管道化進行轉換操作。
- 每個RDD可以切分成多個Partition，每個Partition對應一個Task。所以一個stage可有多個Task組成(taskset)。這些task的執行邏輯完全相同，只是對應不同的partition。
- 一個stage的總task數，由此stage的最後一個RDD的partition數決定。
- 一個Executor可以平行地執行多少個task，由executor被配置的cpu數/每個task所需cpu數決定(spark.executor.cores/spark.task.cpus)
- Task分為ShuffleMapTask/ResultTask，位在最後一個stage的task為ResultTask，其他階段的task皆屬於ShuffleMapTask。

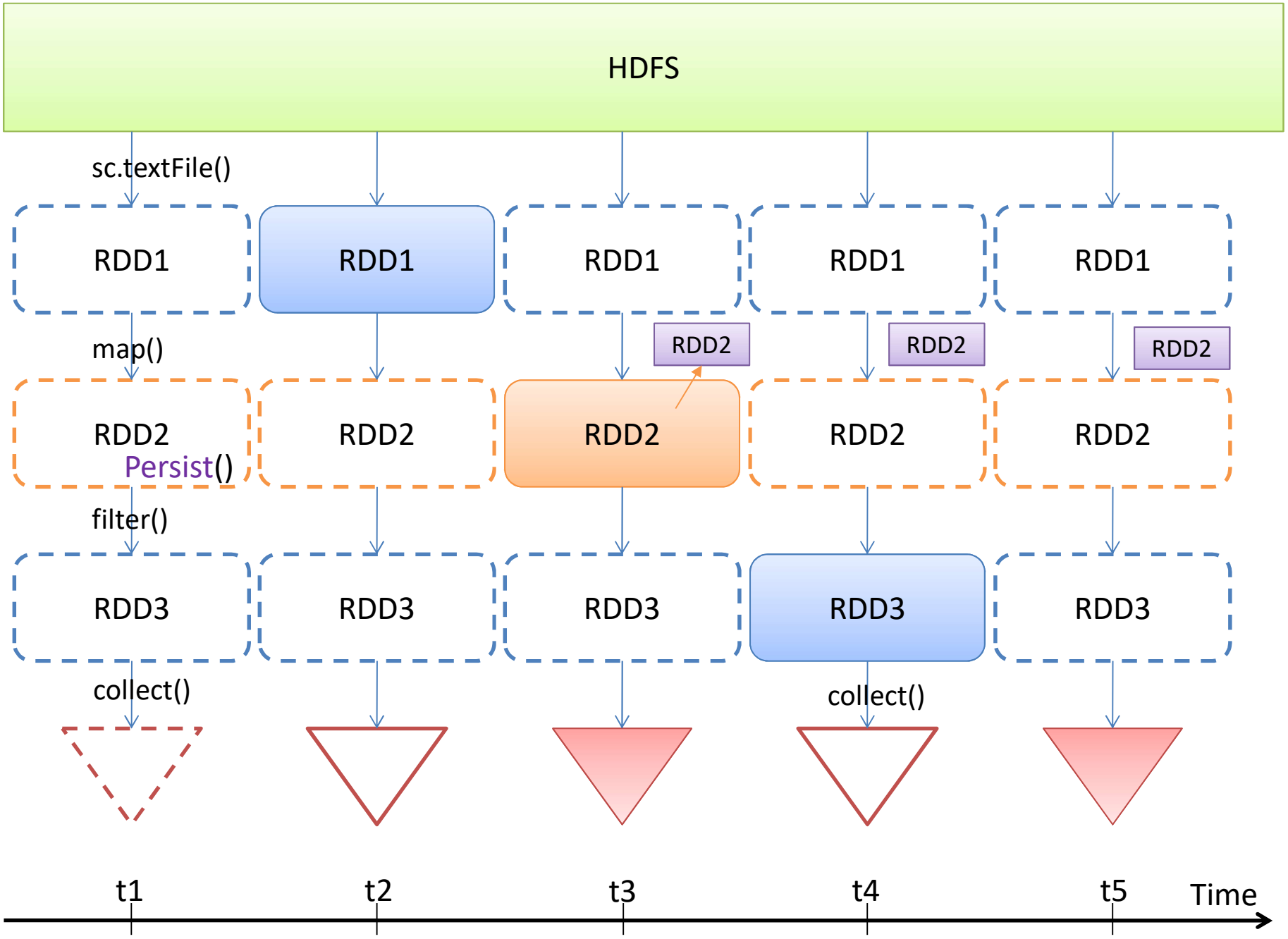

HDFS
Mapper
Reducer
Local FS
Net
MR Job1
HDFS
Mapper
Reducer
Local FS
Net
MR Job2
HDFS
partition1
RDD1
partition2
Task
pipeline
Task
partition1
RDD2
partition2
RDD3
Spark Job stage1
Spark Job stage2
Spark Job

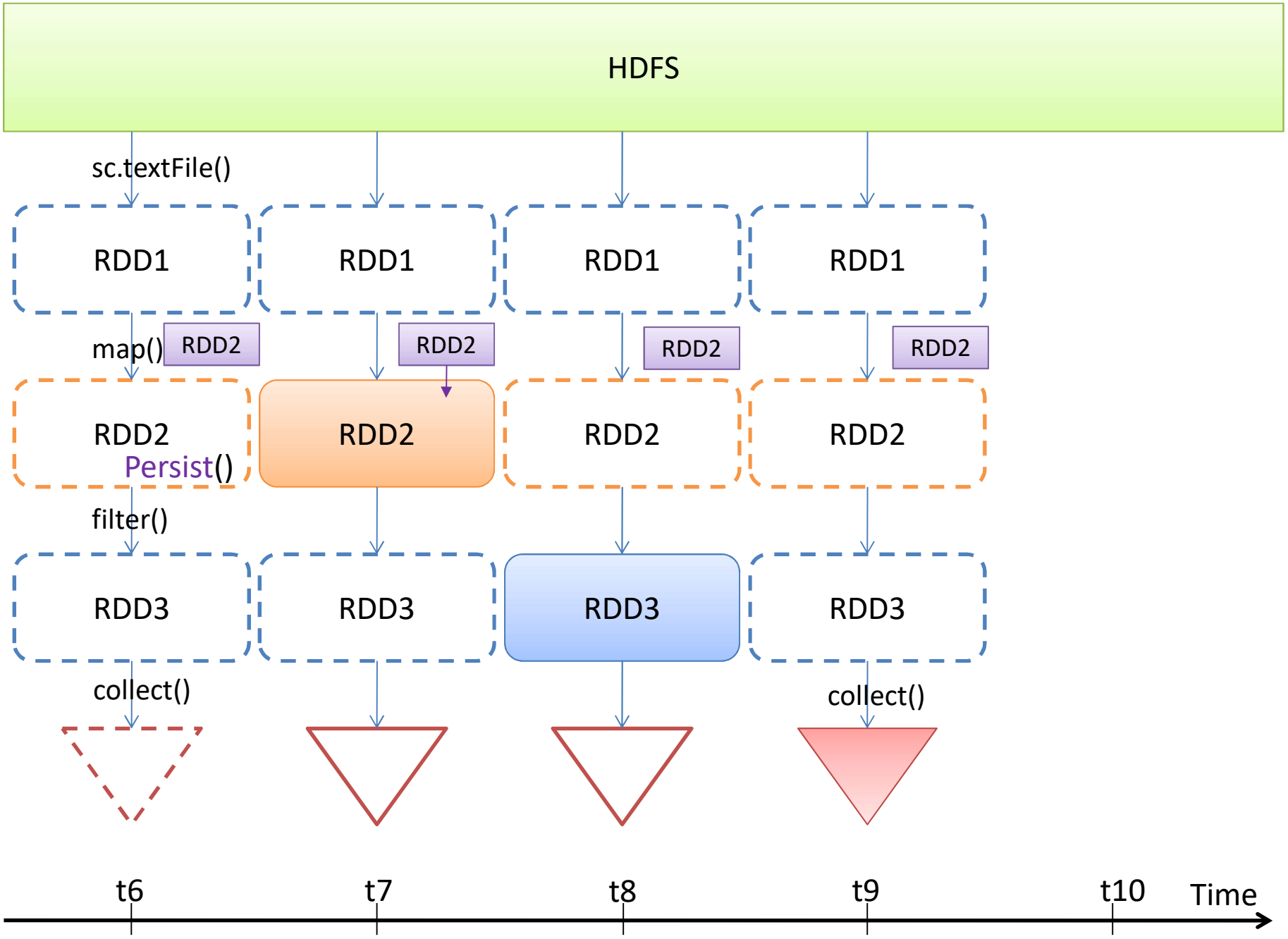

Host
Application/
Job
Worker
Stage
parallelize
CPU
core
Executor
Task
partition
partition
partition
RDD

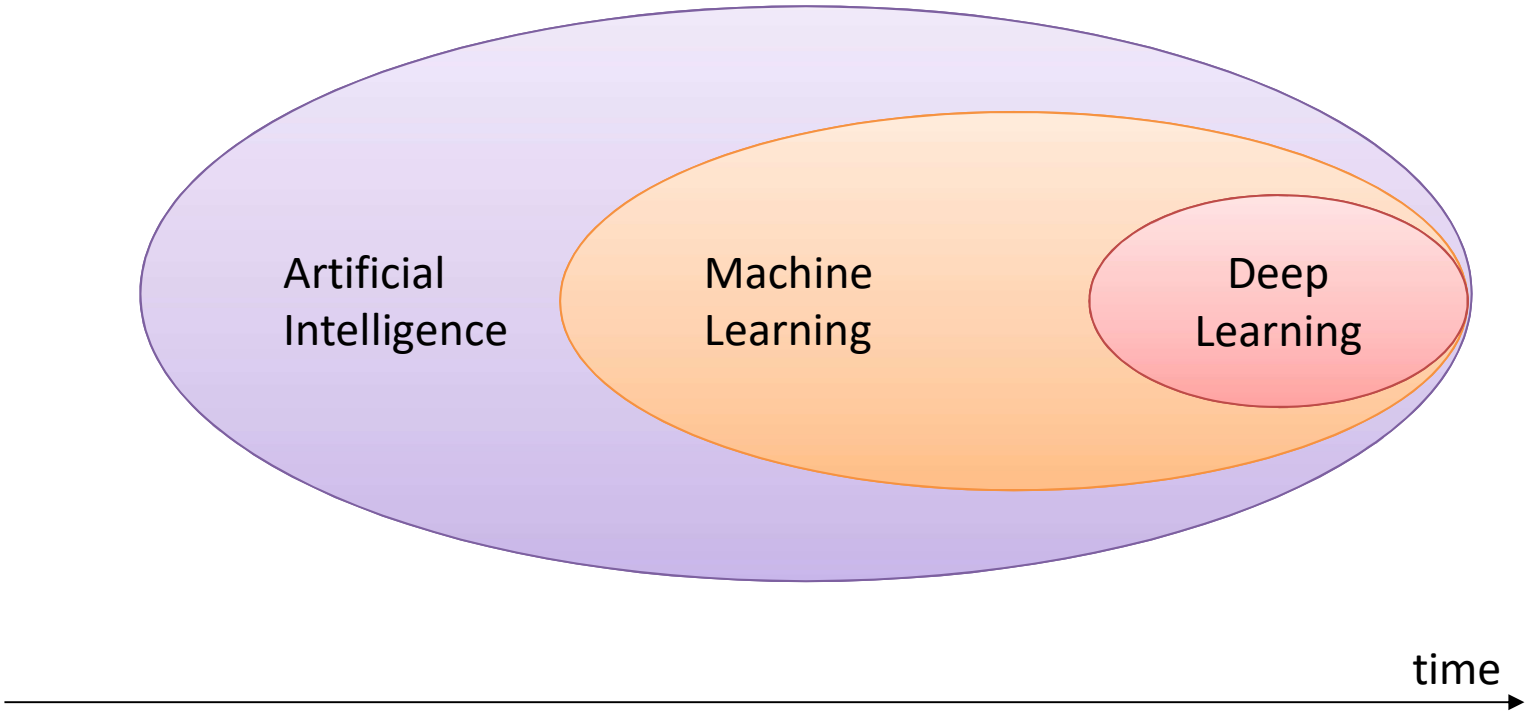

MR Application
MR Job
Task (Map/Red)
JVM
Task Attempts(4)
Spark Application
Executor
JVM
Spark Job
Task
Task Attempts

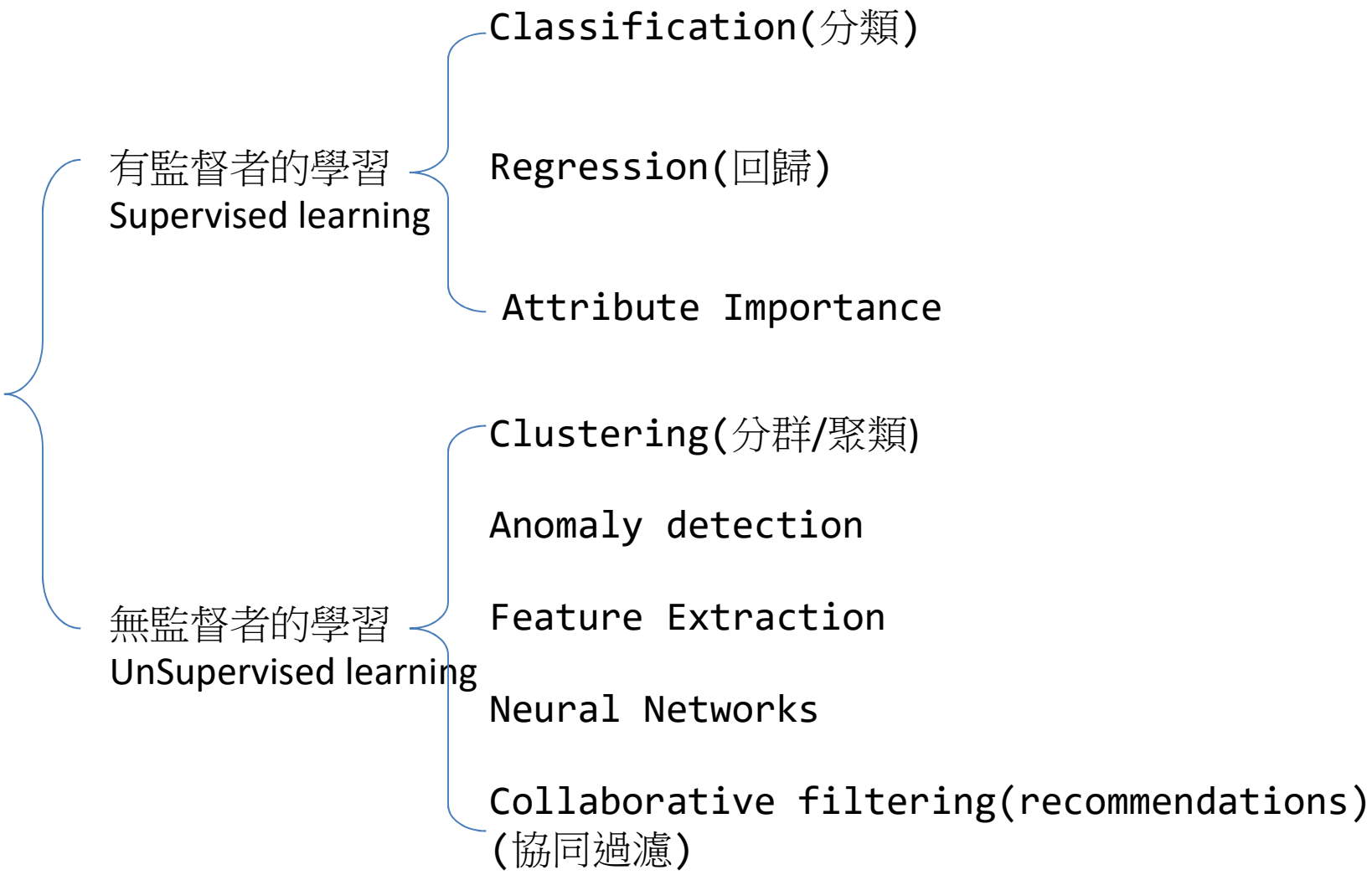

HDFS
sc.textFile()
sc.textFile()
RDD1
RDD1
RDD1
RDD1
RDD1
map()
map()
RDD2
RDD2
RDD2
RDD2
RDD2
filter()
filter()
RDD3
RDD3
RDD3
RDD3
RDD3
collect()
collect()
t1
t2
t3
t4
t5
Time


HDFS
RDD1
RDD1
RDD1
RDD1
RDD1
RDD2
RDD2
RDD2
RDD2
RDD2
RDD3
RDD3
RDD3
RDD3
RDD3
collect()
collect()
collect()
t6
t7
t8
t9
t10
Time

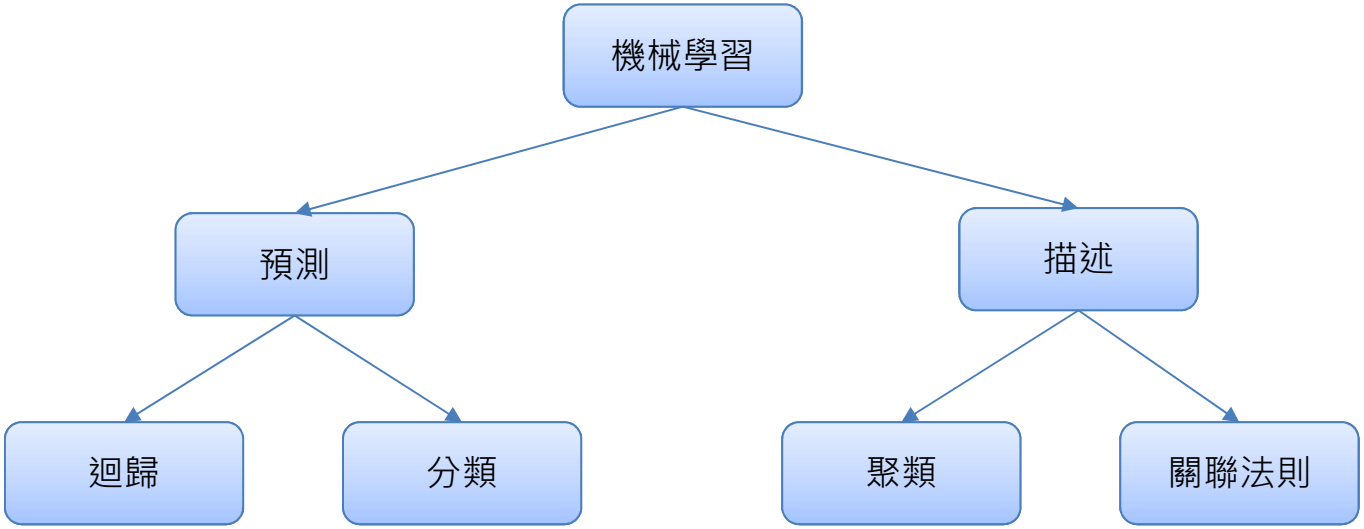

HDFS
sc.textFile()
RDD1
RDD1
RDD1
RDD1
RDD1
map()
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
Persist()
filter()
RDD3
RDD3
RDD3
RDD3
RDD3
collect()
collect()
t1
t2
t3
t4
t5
Time


HDFS
sc.textFile()
RDD1
RDD1
RDD1
RDD1
map()
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
RDD2
Persist()
filter()
RDD3
RDD3
RDD3
RDD3
collect()
collect()
t6
t7
t8
t9
t10
Time


Artificial
Intelligence
Machine
Learning
Deep
Learning
time


有監督者的學習
Supervised learning
Classification(分類)
Regression(回歸)
Attribute Importance
無監督者的學習
UnSupervised learning
Clustering(分群/聚類)
Anomaly detection
Feature Extraction
Neural Networks
Collaborative filtering(recommendations)
(協同過濾)

選擇適當的機械學習方法

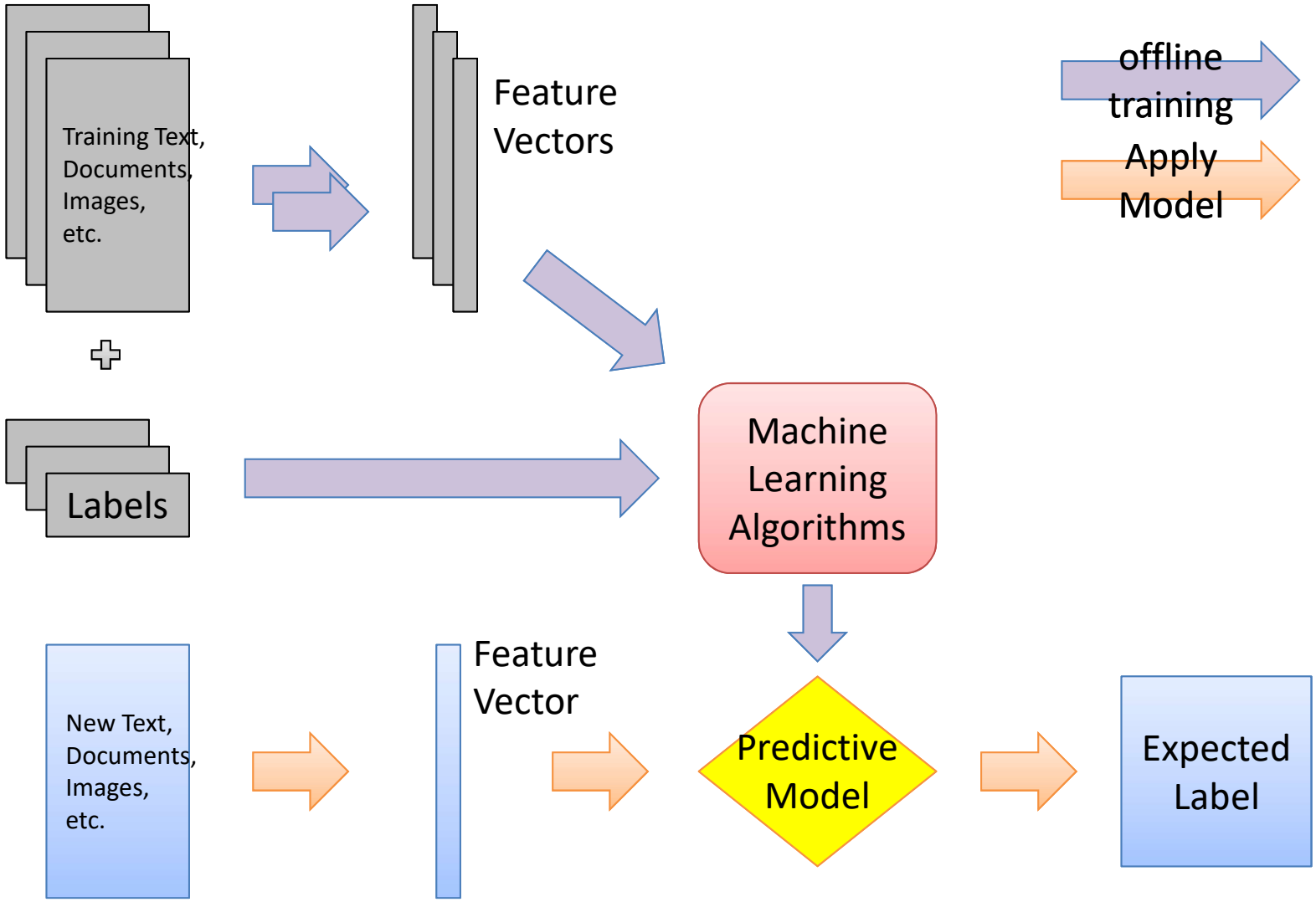


監督式學習

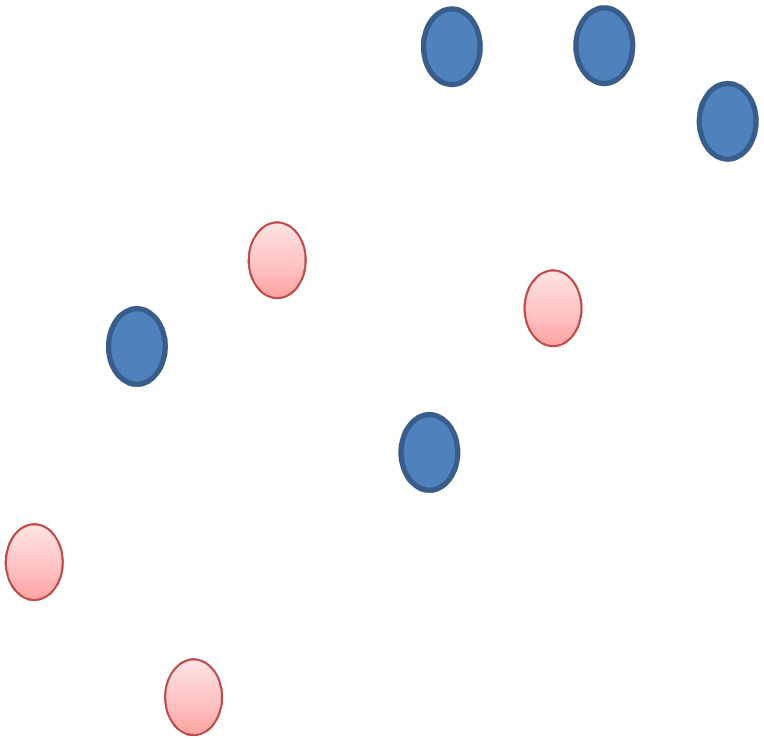
- 從給定的訓練資料集中學習出一個函式，當新的資料到來時，可以根據這個函式預測結果。監督學習的訓練集要求是包括輸入和輸出，也可以說是特徵和目標。訓練集中的目標是由人標註的。
  - 迴歸(Regression)-輸出為數值型資料
    - 線性回歸(Linear Regression)
    - 多項式迴歸(Polynomial Regression)
  - 分類(Classification)-輸出為類別型資料
    - 邏輯回歸(Logistic Regression)
    - 決策樹(Decision Tree)、隨機森林(Random Forest)
    - k-最近鄰居(k-Nearest Neighbors，k-NN)
    - 支持向量機(SVM)、相關向量機(RVM)
    - 樸素貝葉斯(Naïve Bayes)
    - 神經網絡(Neural Network)
    - 感知器(Perceptron)

資料來源：維基百科

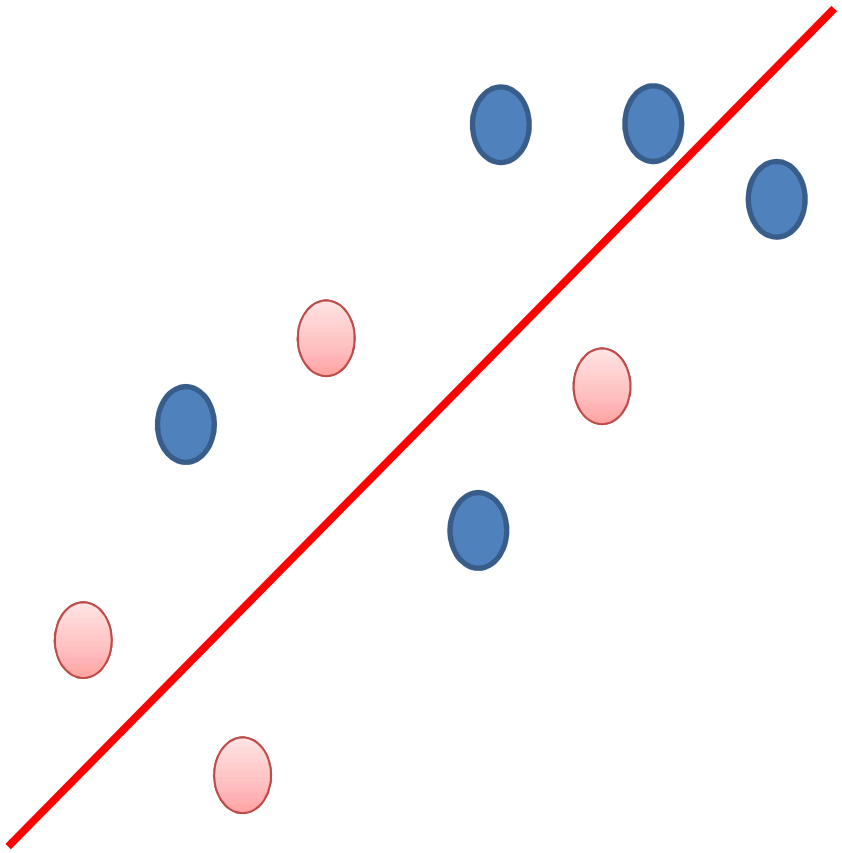
監督式學習流程


Training Text,
Documents,
Images,
etc.
Feature
Vectors
offline
training
Apply
Model
Labels
Machine
Learning
Algorithms
New Text,
Documents,
Images,
etc.
Feature
Vector
Predictive
Model
Expected
Label

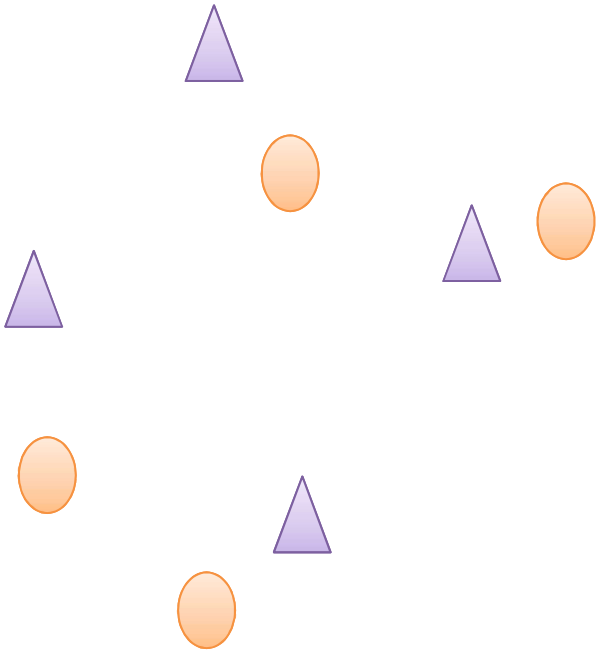
線性迴歸



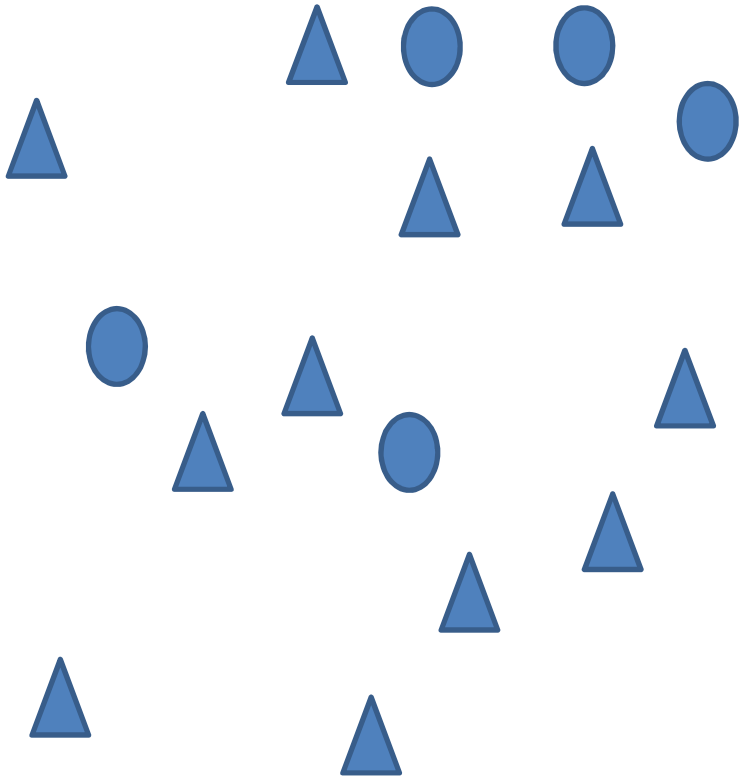
線性迴歸



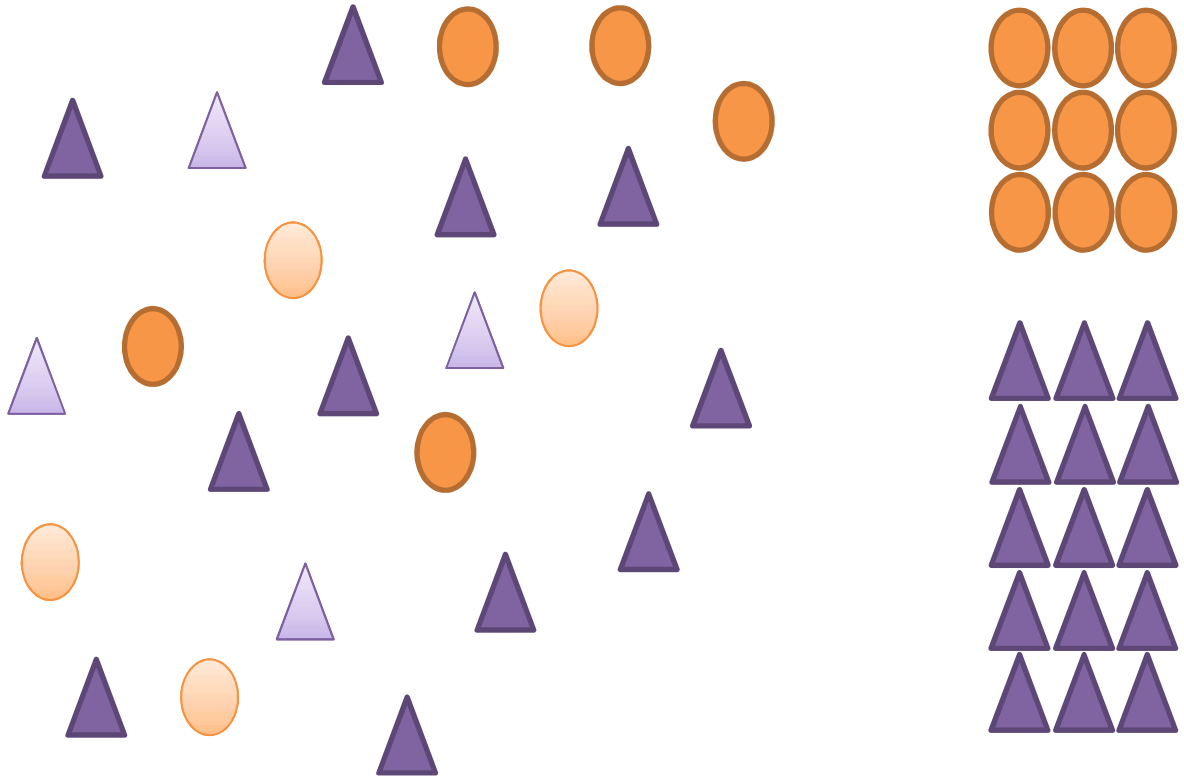
分類



分類



分類

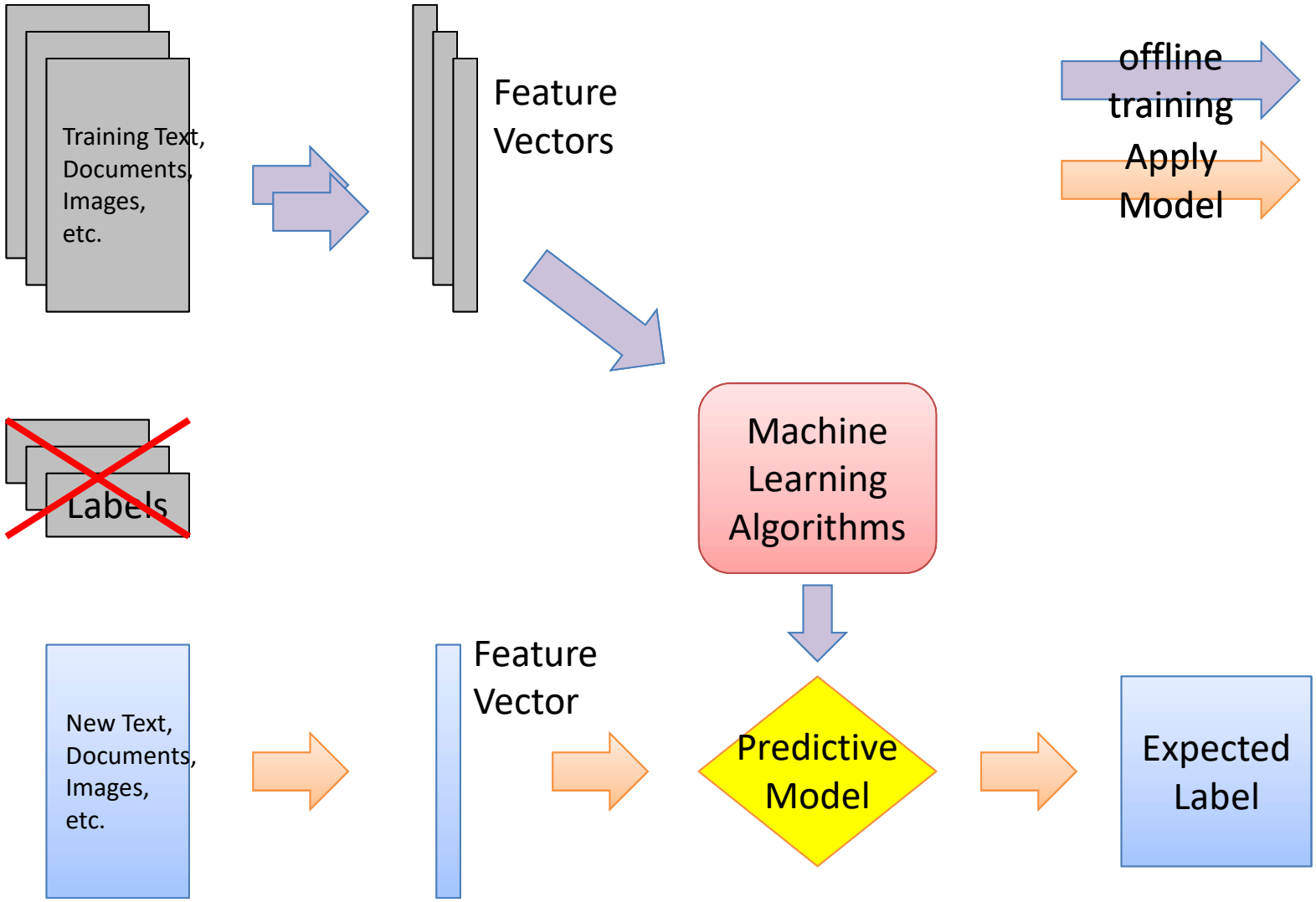


無監督式學習

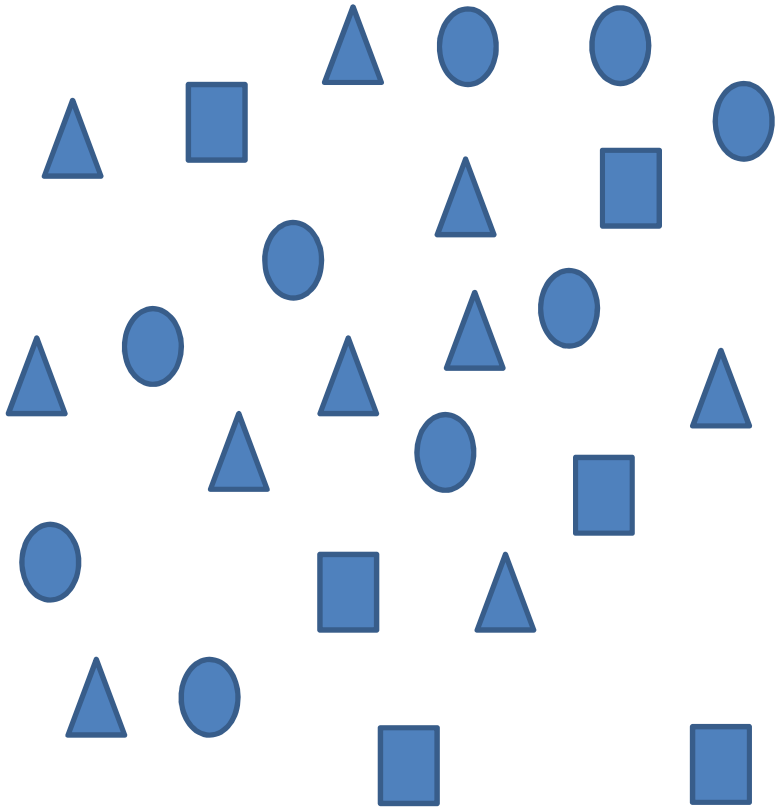
- 與監督學習相比，訓練集沒有人為標註的結果。
  - 聚類/分群(Clustering)
    - **hierarchical clustering**
    - k-平均數(k-means)
    - **Density-based spatial clustering of applications with noise (DBSCAN)**
  - 異常偵測(Anomaly Detection)
  - 主要成分分析(Principal Component Analysis，PCA)

資料來源：維基百科

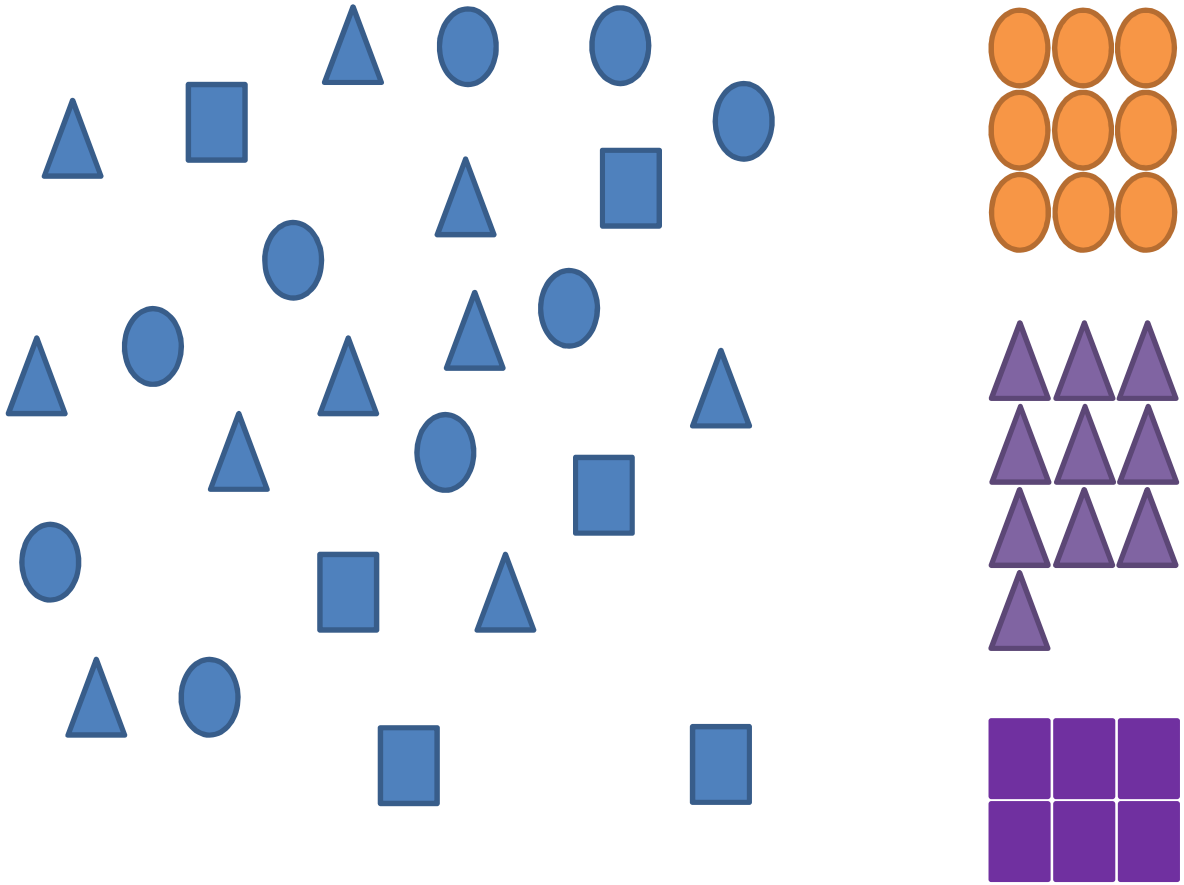
無監督式學習流程


Training Text, Documents, Images, etc.
Feature Vectors
offline training
Apply Model
Labels
Machine Learning Algorithms
New Text, Documents, Images, etc.
Feature Vector
Predictive Model
Expected Label

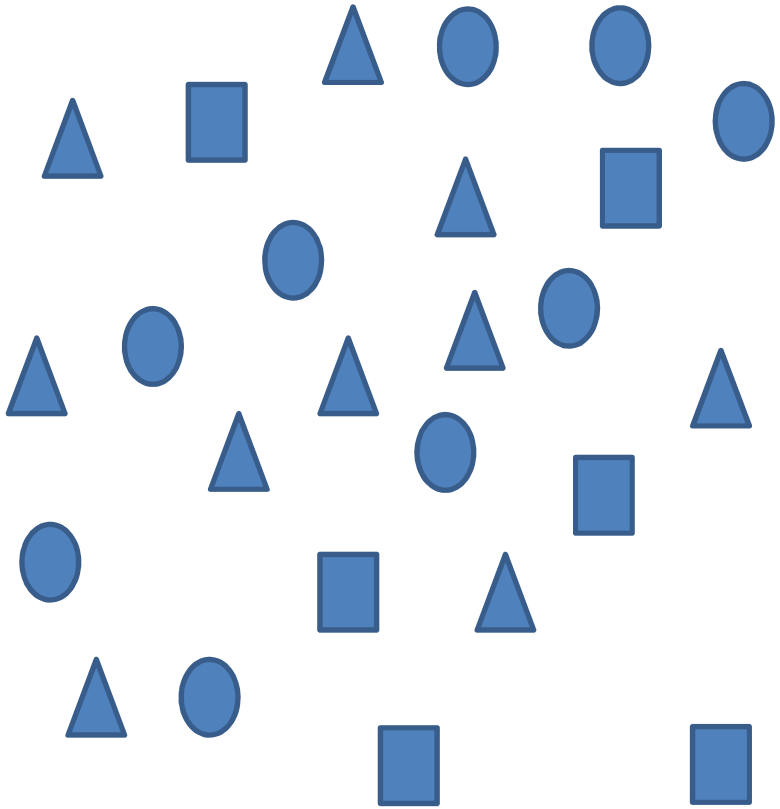
分群



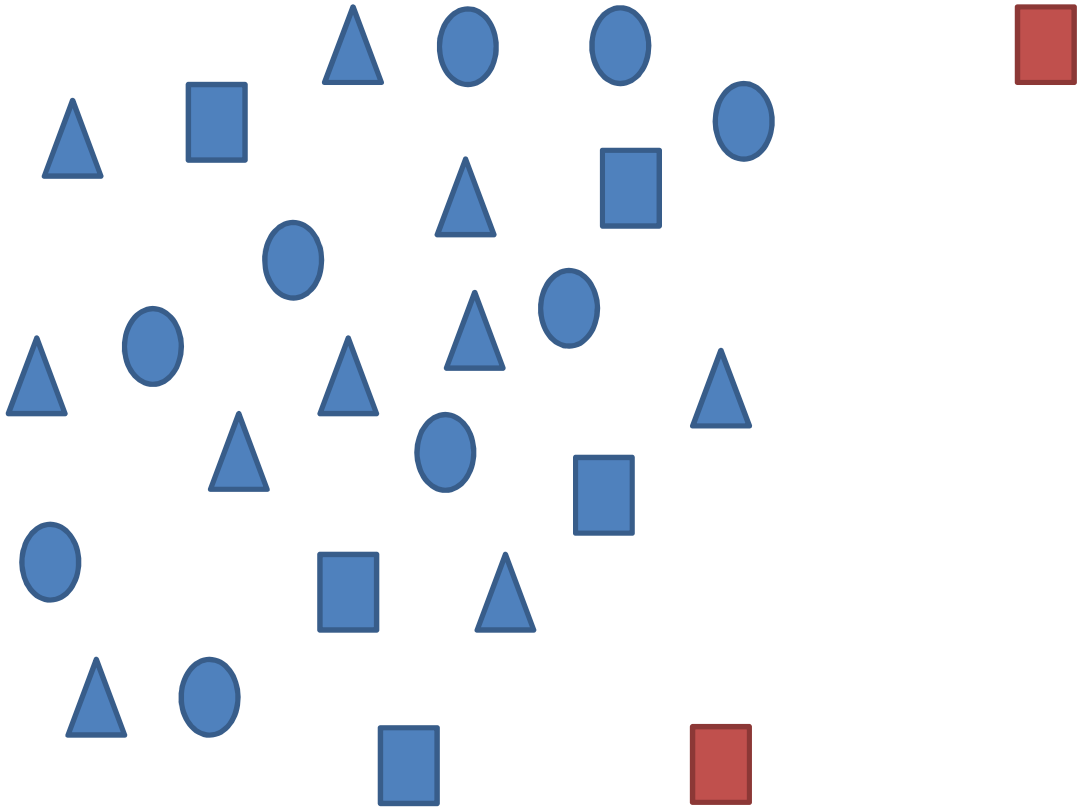
分群

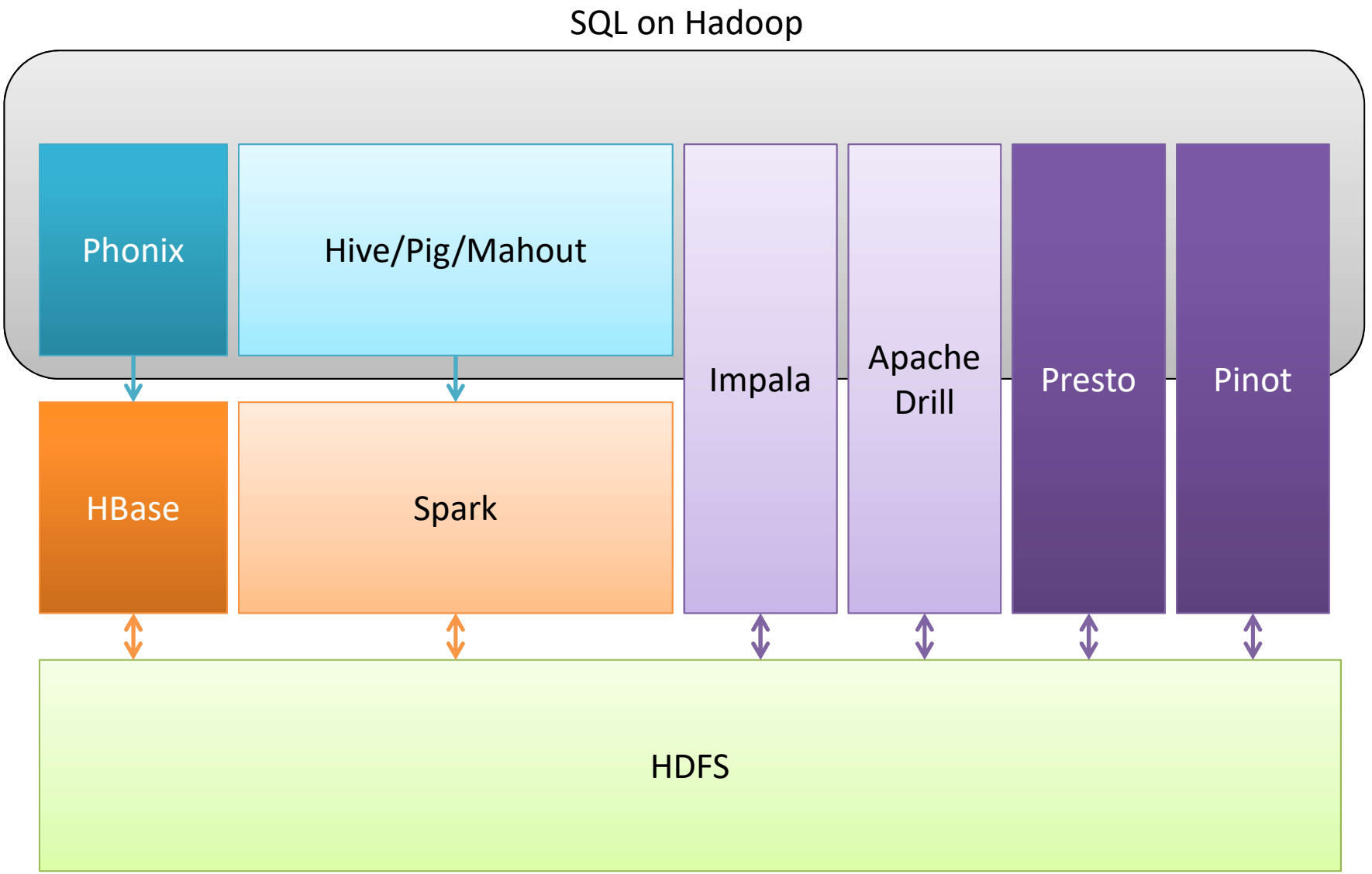


異常偵測

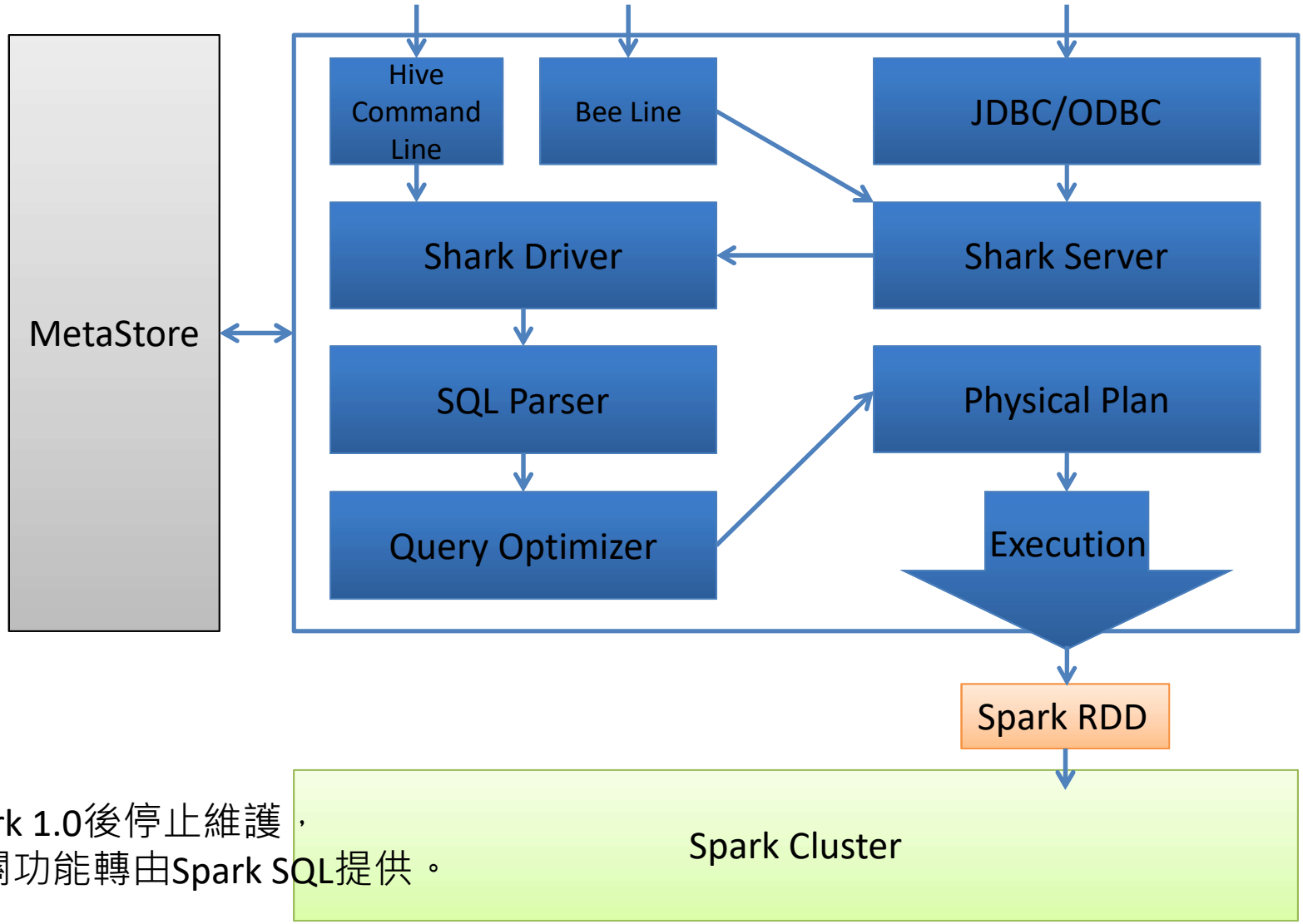


異常偵測



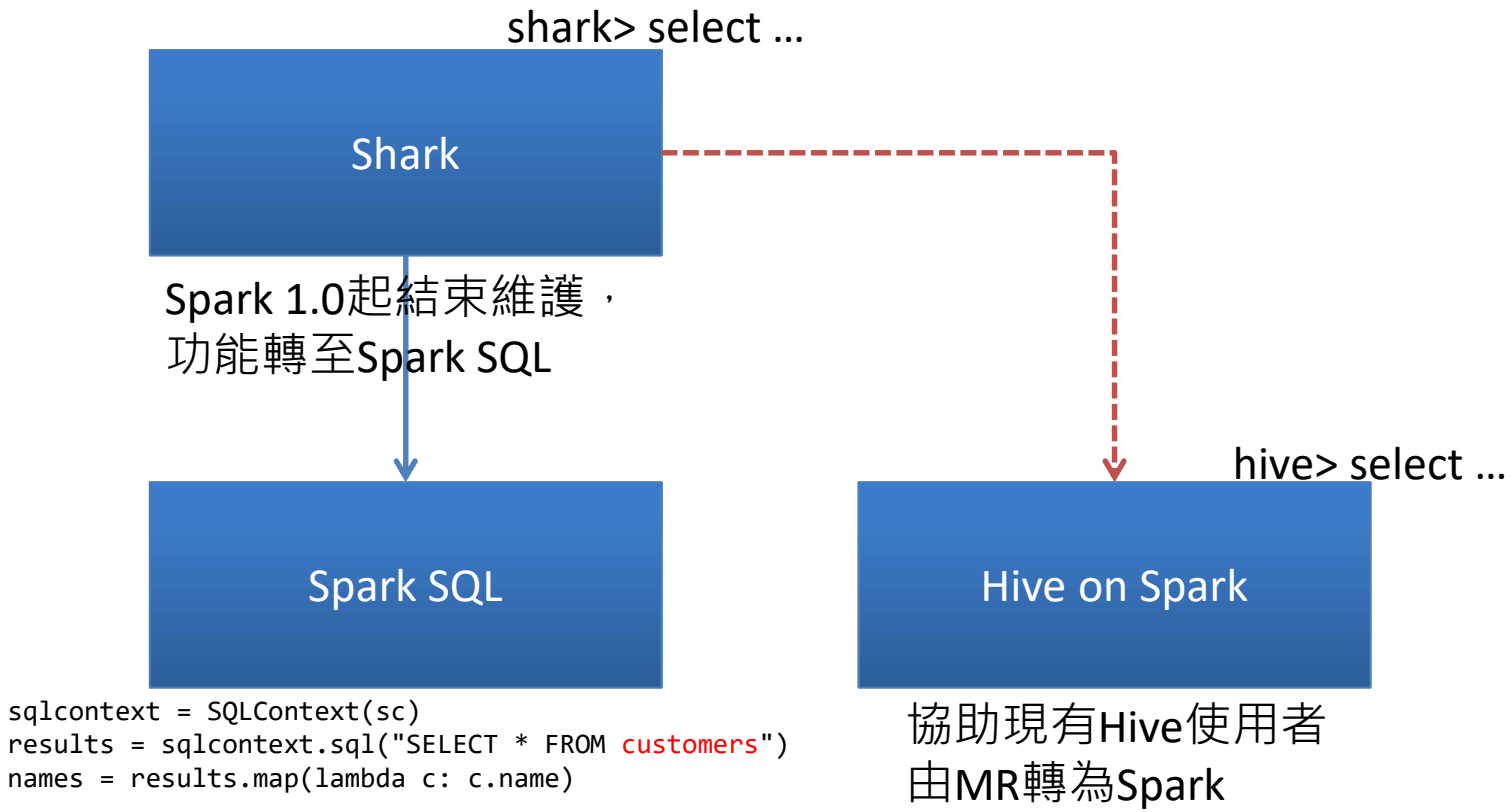

SQL on Hadoop
Phonix
Hive/Pig/Mahout
Impala
Apache Drill
Presto
Pinot
HBase
Spark
HDFS

Shark

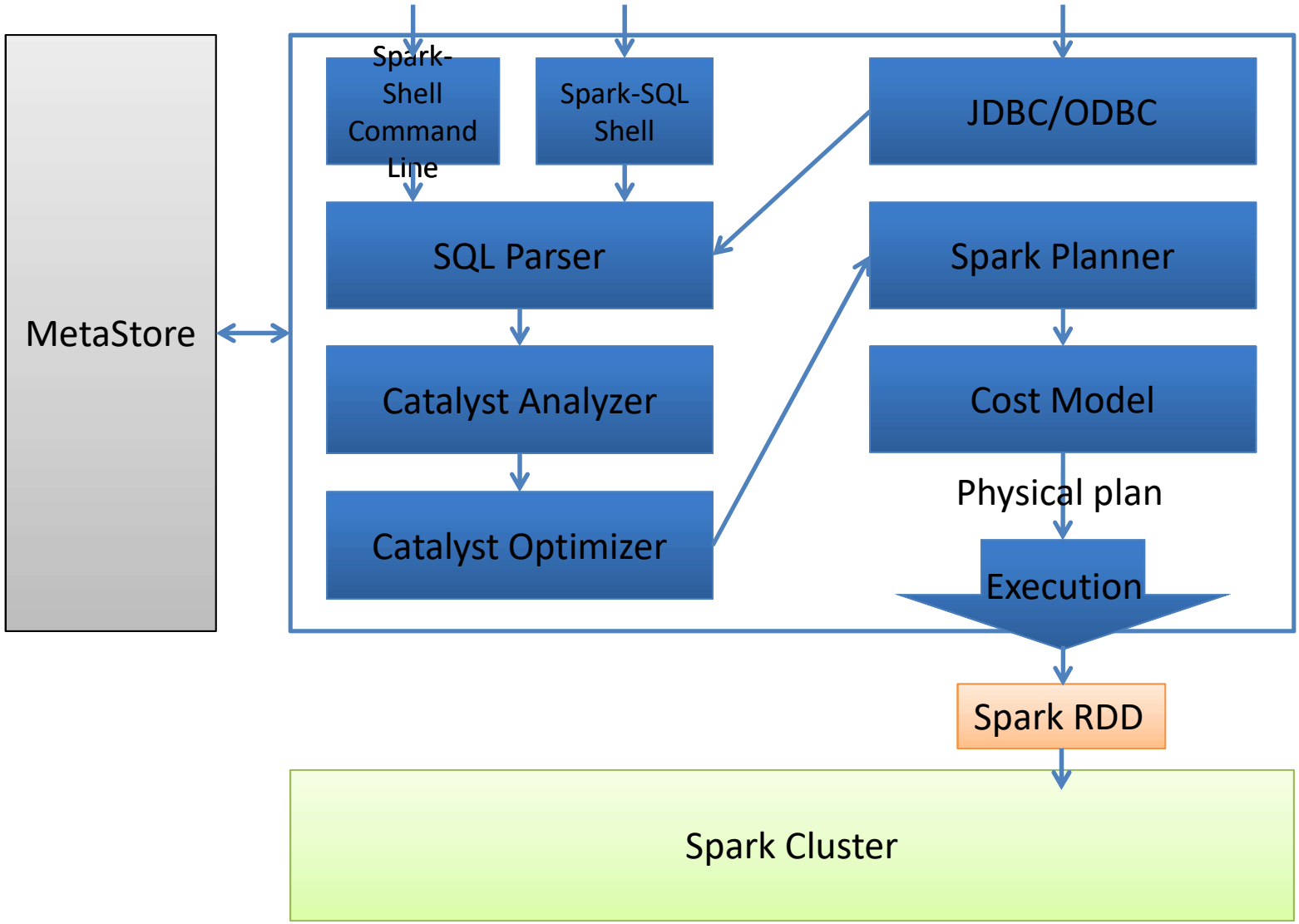


Spark 1.0後停止維護，
相關功能轉由Spark SQL提供。

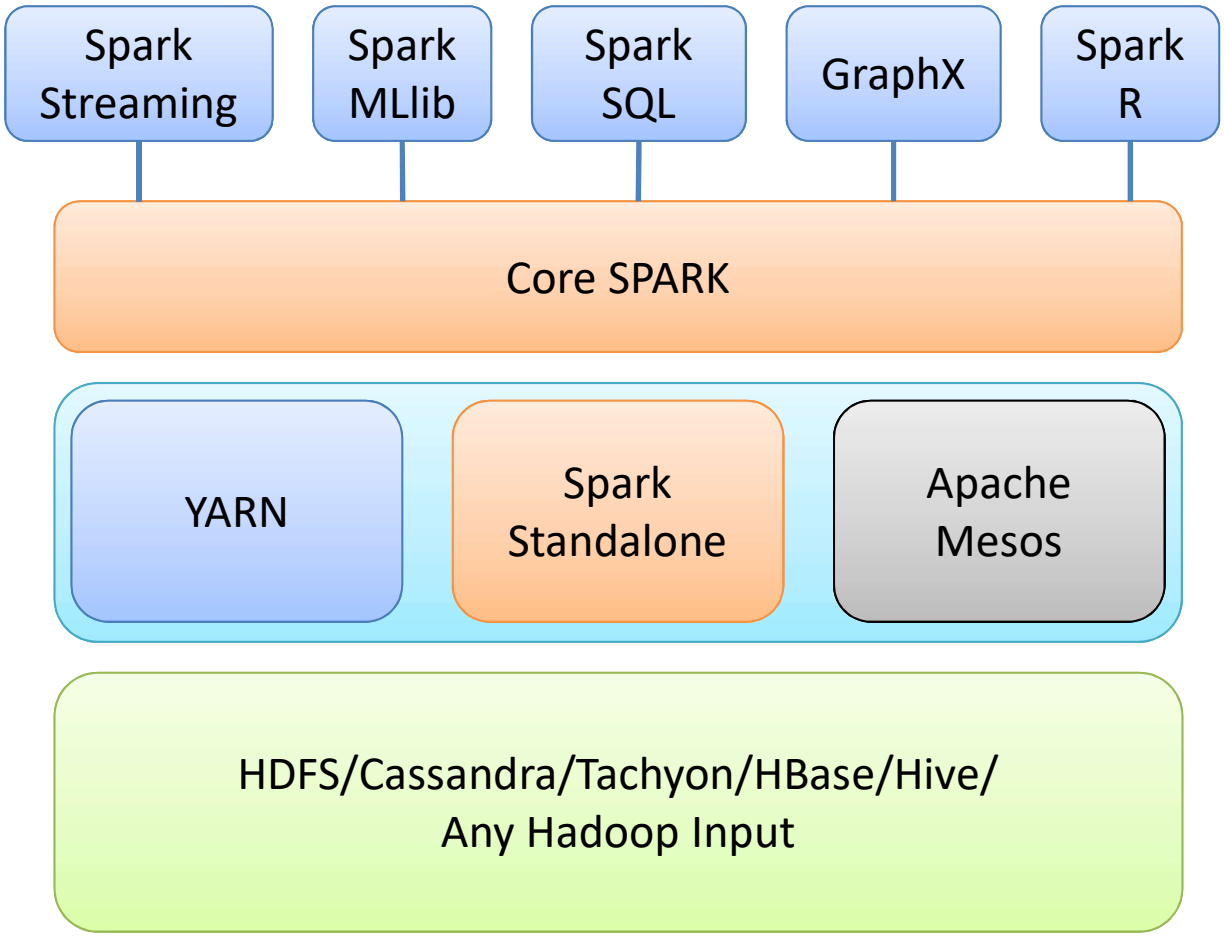
SQL on Spark


shark> select …
Shark
Spark 1.0起結束維護，
功能轉至Spark SQL
hive> select …
Spark SQL
Hive on Spark
sqlcontext = SQLContext(sc)
results = sqlcontext.sql("SELECT * FROM customers")
names = results.map(lambda c: c.name)
協助現有Hive使用者
由MR轉為Spark

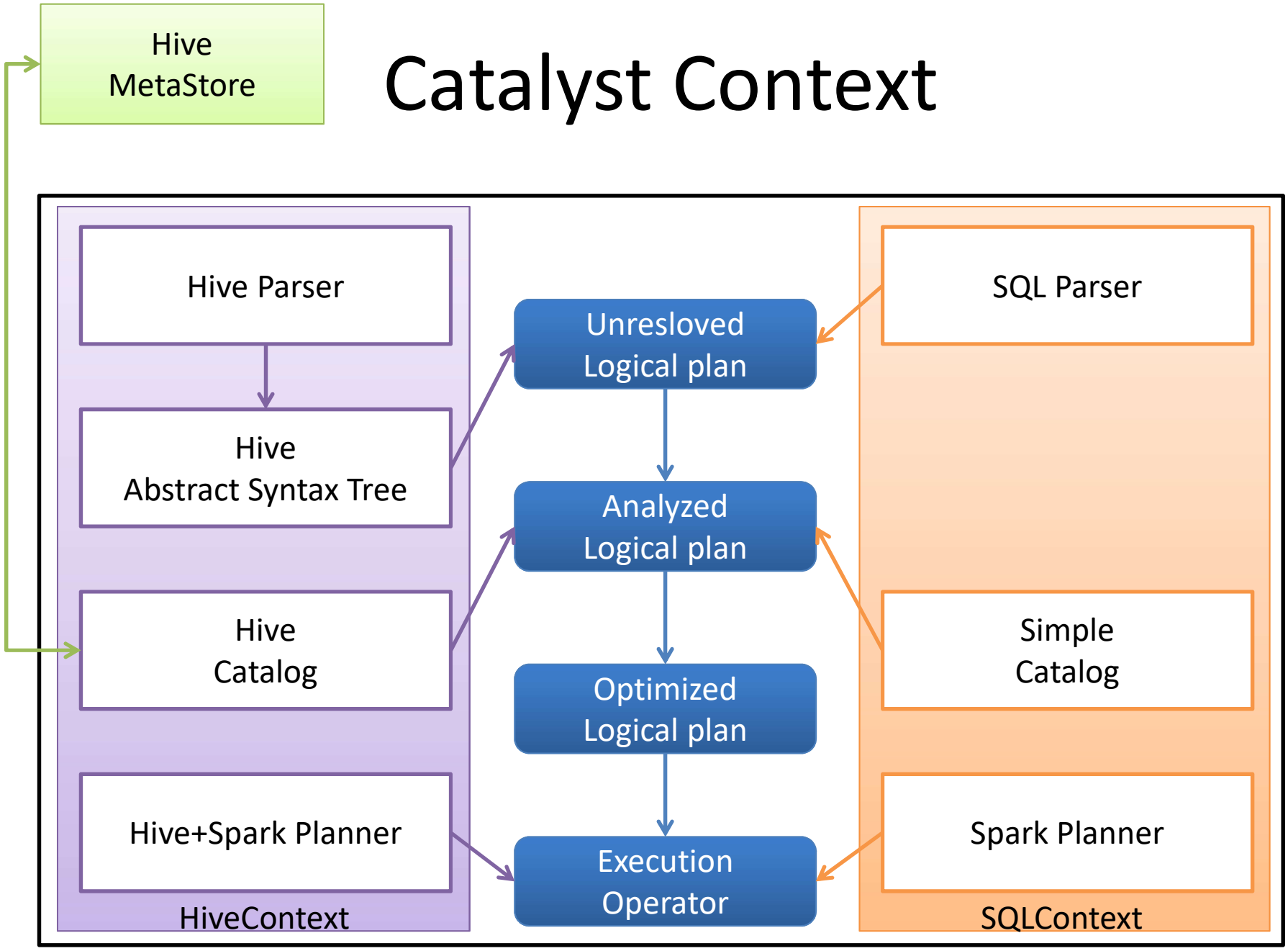
SparkSQL

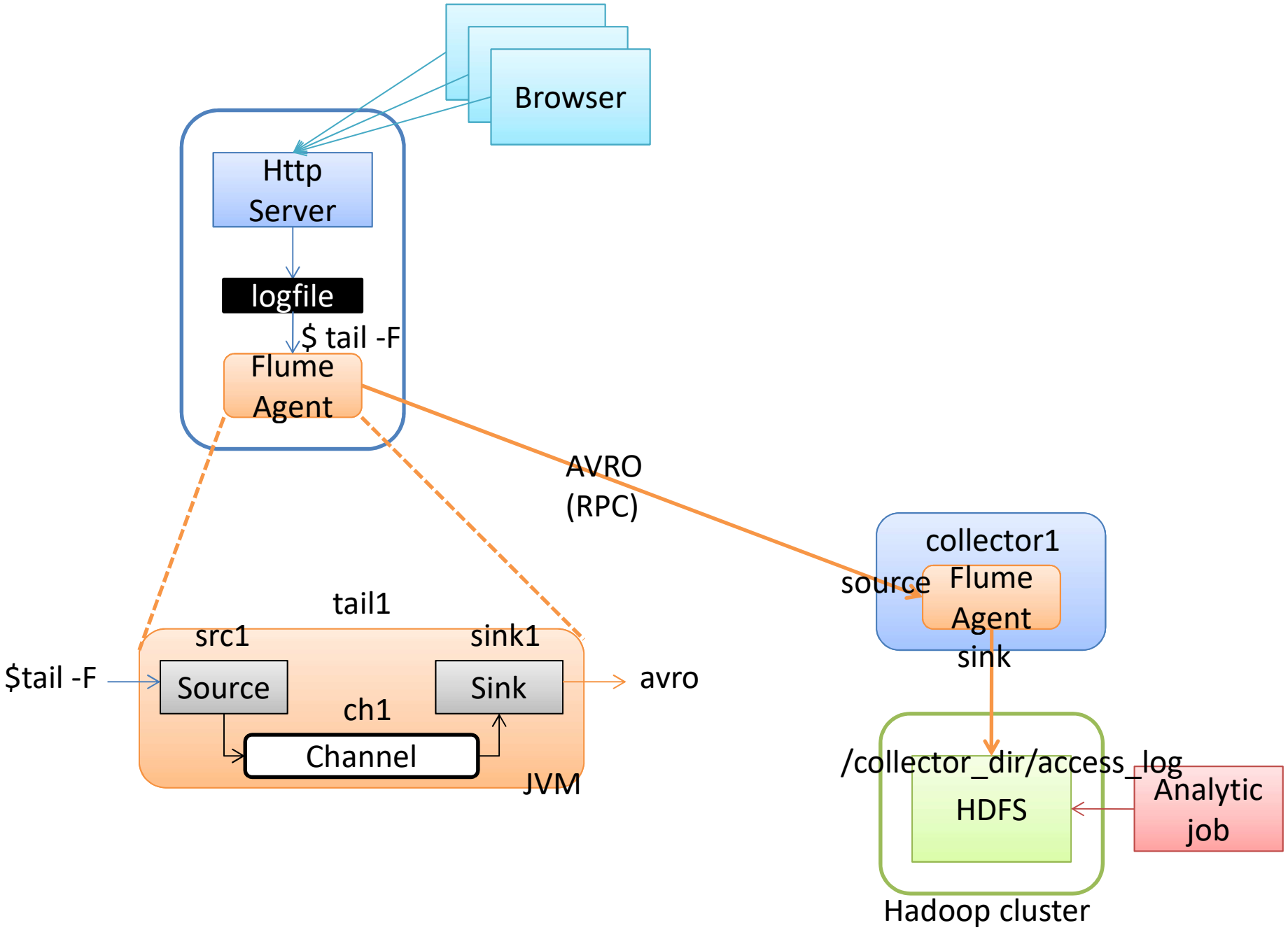

MetaStore
Spark-Shell Command Line
Spark-SQL Shell
JDBC/ODBC
SQL Parser
Spark Planner
Catalyst Analyzer
Cost Model
Catalyst Optimizer
Physical plan
Execution
Spark RDD
Spark Cluster

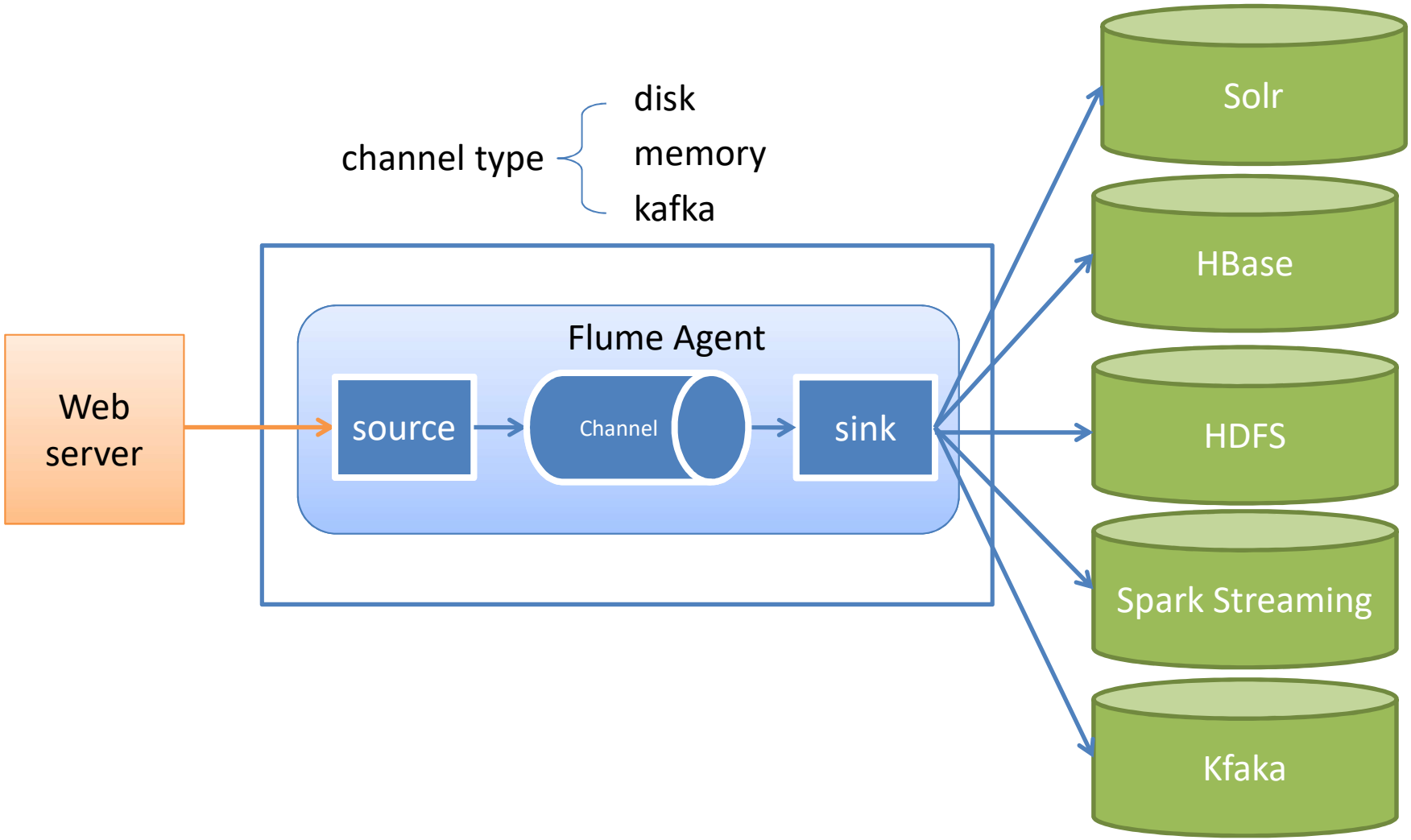
SPARK


Spark Streaming
Spark MLlib
Spark SQL
GraphX
Spark R
Core SPARK
YARN
Spark Standalone
Apache Mesos
HDFS/Cassandra/Tachyon/HBase/Hive/
Any Hadoop Input

Catalyst Context




Browser
Http Server
logfile
$ tail -F
Flume Agent
AVRO (RPC)
collector1
source
Flume Agent
sink
tail1
src1
sink1
$tail -F
Source
Sink
avro
ch1
Channel
JVM
/collector_dir/access_log
HDFS
Analytic job
Hadoop cluster


channel type
disk
memory
kafka
Web server
Flume Agent
source
Channel
sink
Solr
HBase
HDFS
Spark Streaming
Kfaka

Spark Streaming

- Spark Streaming是core Spark API的延伸，對即時的資料流提供縮放性、高吞吐量、容錯的串流處理。
- Spark Streaming可以處理來自多種資料源的串流資料
  - Kafka、Flume、Kinesis、TCP Socket

- Spark Streaming接收到即時的資料串流輸入，將串流資料切分為批次資料，然後交由Spark引擎處理產生最終結果。

資料來源： http://spark.apache.org/docs/latest/streaming-programming-guide.html

DStream(Discretized Stream)

- DStream用來表示一個連續的資料串流
- DStream可以看作是RDD的順序
- 建立Dstream有兩種方式:
  - 由資料源接受資料串流
    - Kafka、Flume、Kinesis
  - 原本的Dstream經過Spark Streaming的高階操作而生成
    - Scala、Java、Python

StreamingContext()

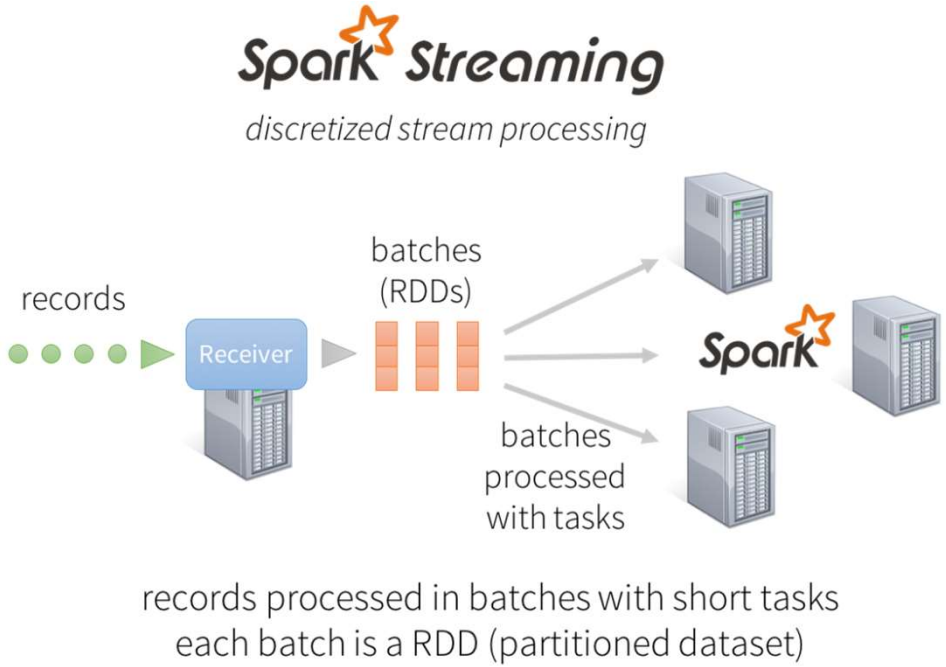
- StreamingContext是Spark Streaming的入口點
- StreamingContext用來與Spark Cluster通信，以及建立Dstream。
- 建立StreamingContext需要通過SparkContext

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

sc = SparkContext()
ssc = StreamingContext(sc,1)
```

Processing Time and Event Time

- Event Time
  - 資料生成時間，通常內含在資料本身。
    - 例如ETC記錄：車號ABC-123的車輛於12:01:50通過閘門1
    - 12:01:50即為Event Time
- Processing Time
  - 資料被處理的時間
    - 閘門1的資料於12:01:59秒送到Spark Streaming進行處理
    - 12:01:59則是Processing Time
- Spark Streaming基於micro-batch模式，將串流資料依照接受時間每幾秒切分為一個batch資料。所以Spark Streaming是基於Processing Time進行切分。

Spark Streaming架構

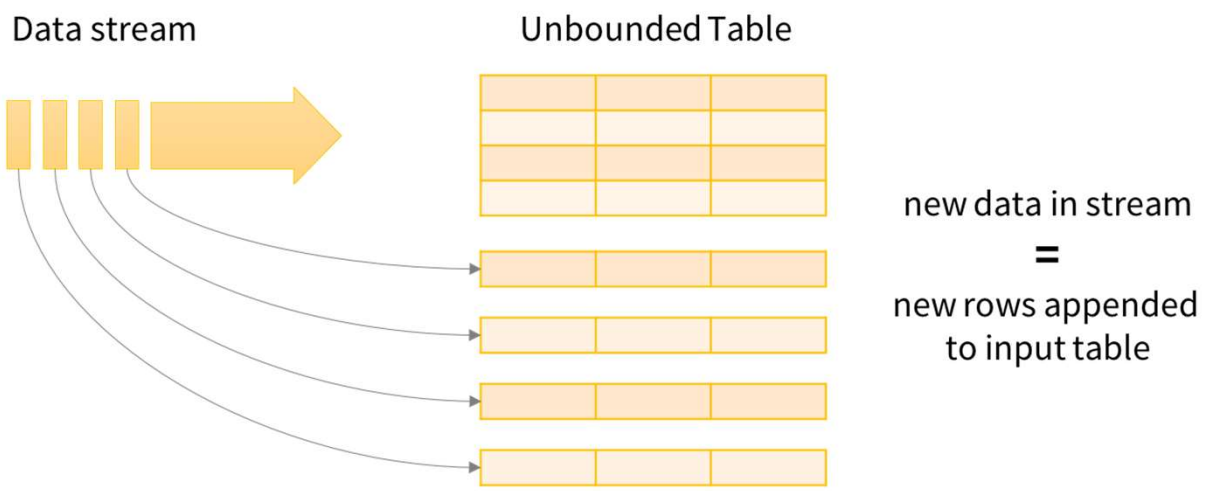


Structured Streaming

- 2016年於Spark 2.0開始引入
  - 可以區分processing time與event time
  - 使用relational execution engine提升效能
- Declarative(宣告式) API
  - Structured Streaming
- Imperative(命令式) API
  - Spark Streaming

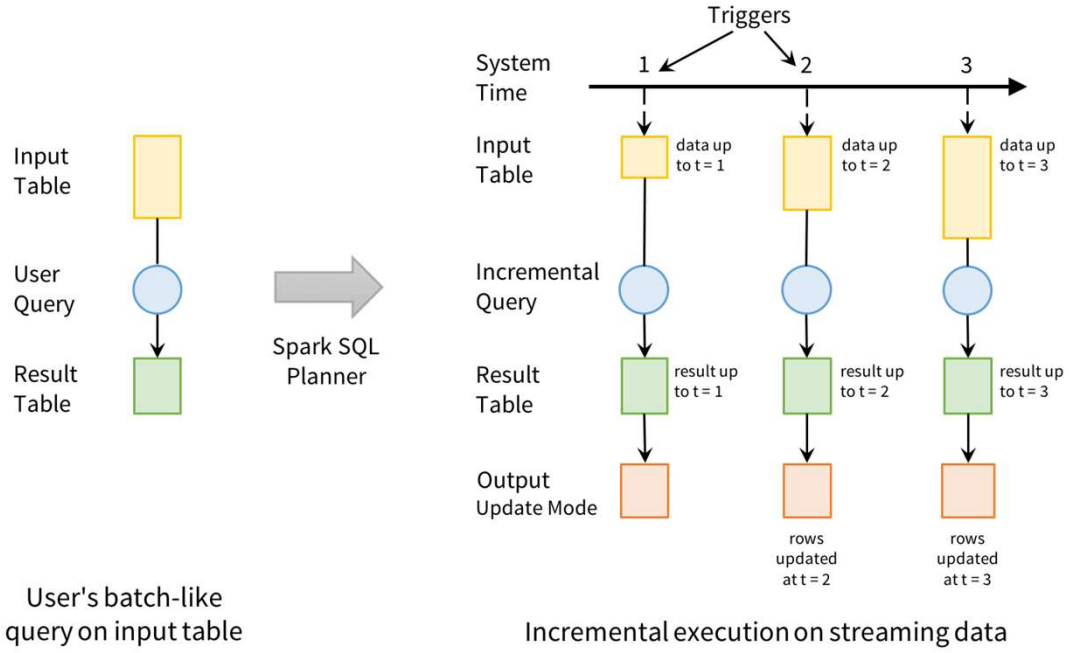
Structured Streaming的優點

- Incremental query model
  - 可以執行增量查詢。且程式碼與批次操作(RDD)API完全一樣(基於DataFrame與DataSet API)
- Support for end-to-end application
  - Structured Streming與內建的connector使得end-to-end程式非常容易撰寫，而且”correct by default”。
- Reuse Spark SQL engine
  - 通過DataFrame/DataSet API讓Structured Streaming可以直接利用Spark SQL之前所做的最佳化努力，如執行計畫優化、codegen、記憶體管理等。



Data stream as an unbounded Input Table

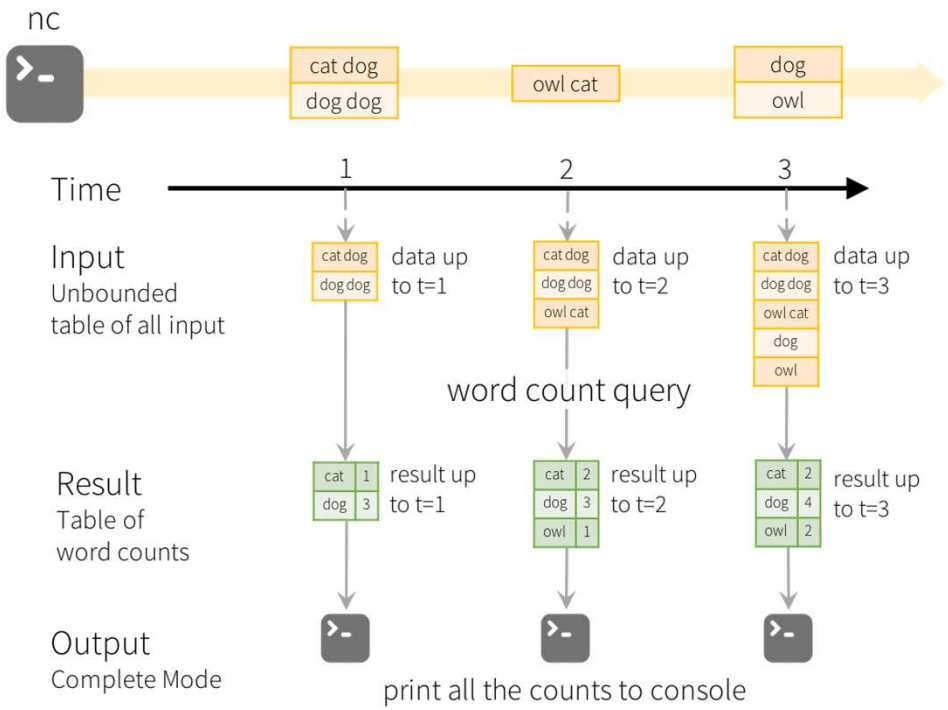
資料來源： https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



Structured Streaming Processing Model

Users express queries using a batch API; Spark incrementalizes them to run on streams

資料來源： https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html



Model of the Quick Example

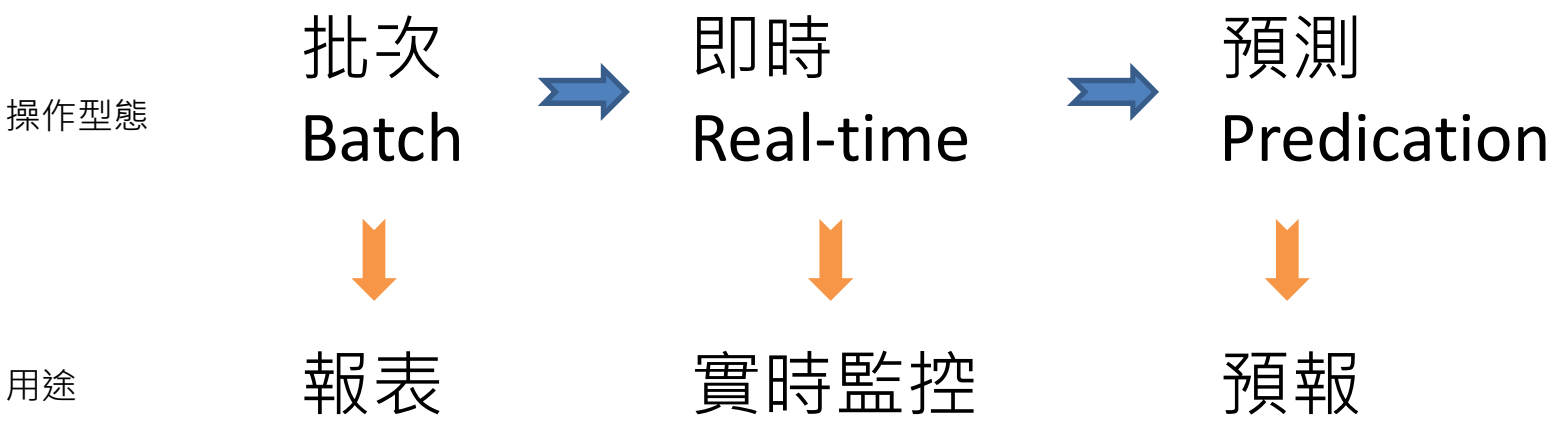
資料來源： https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

與其他串流引擎的比較

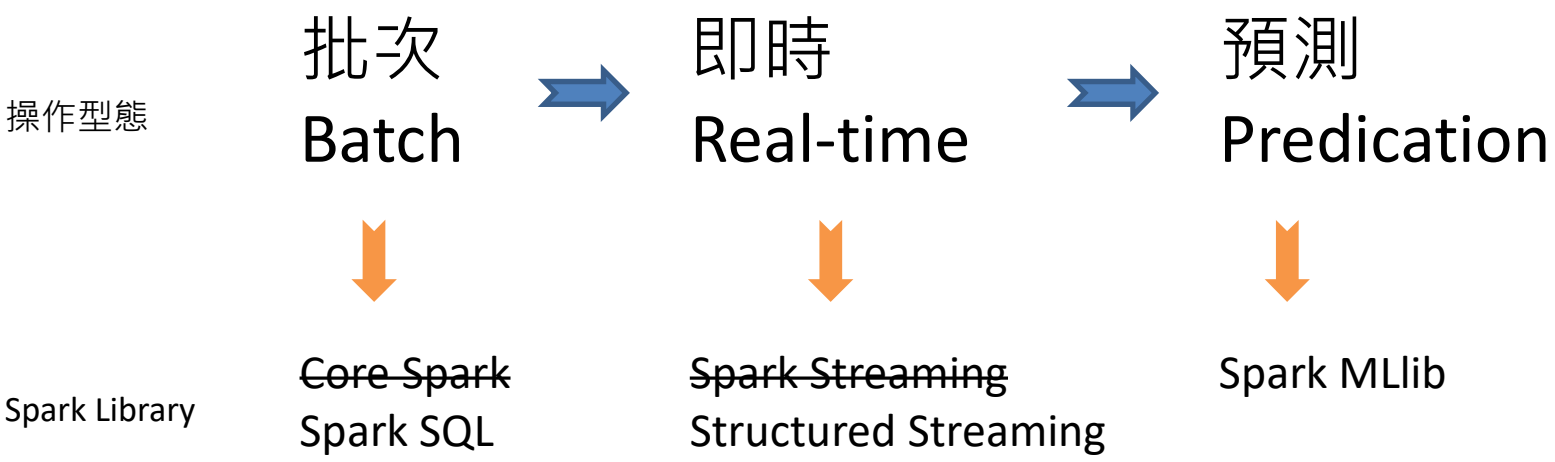
Property	Structured Streaming	Spark Streaming	Apache Storm	Apache Flink	Kafka Streams	Google Dataflow
Streaming API	incrementalize batch queries	integrates with batch	separate from batch	separate from batch	separate from batch	integrates with batch
Prefix Integrity Guarantee	✔	✔	✘	✘	✘	✘
Internal Processing	exactly once	exactly once	at least once	exactly once	at least once	exactly once
Transactional Sources/Sinks	✔	some	some	some	✘	✘
Interactive Queries	✔	✔	✘	✘	✘	✘
Joins with Static Data	✔	✔	✘	✘	✘	✘

資料來源： https://databricks.com/blog/2016/07/28/structured-streaming-in-apache-spark.html

資料用途的演化



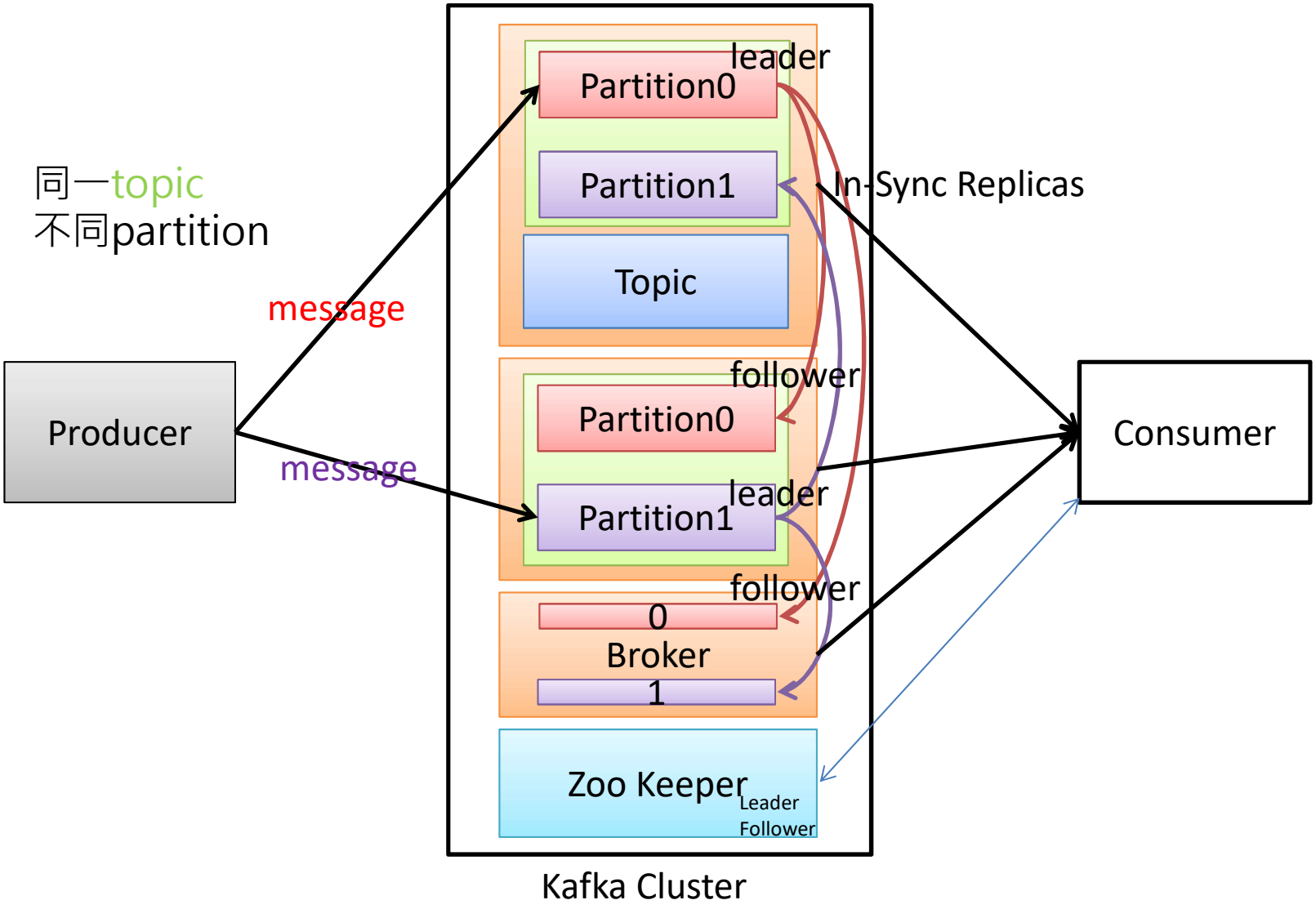
Spark程式庫與資料用途的對應



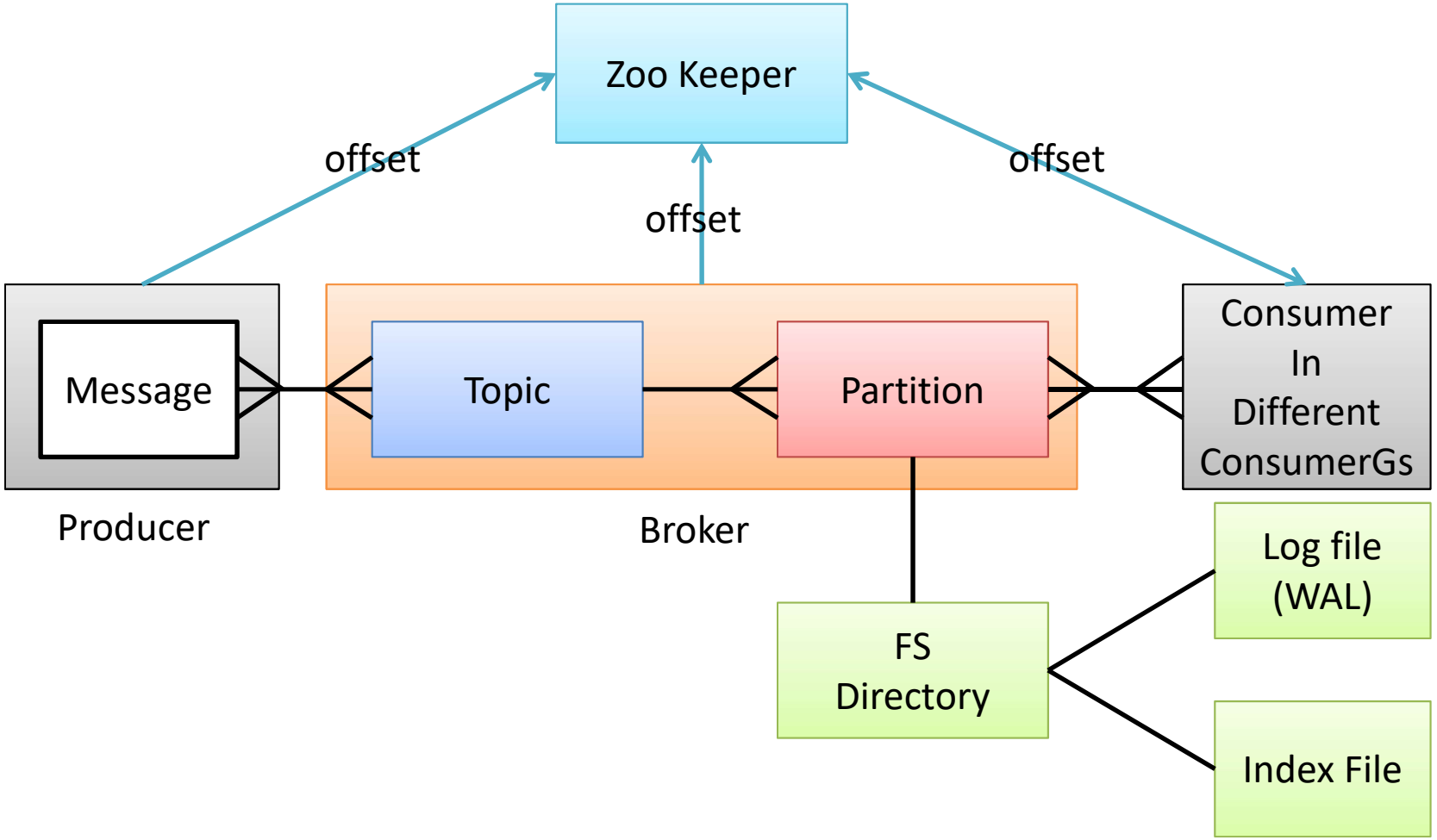
Structured Streaming

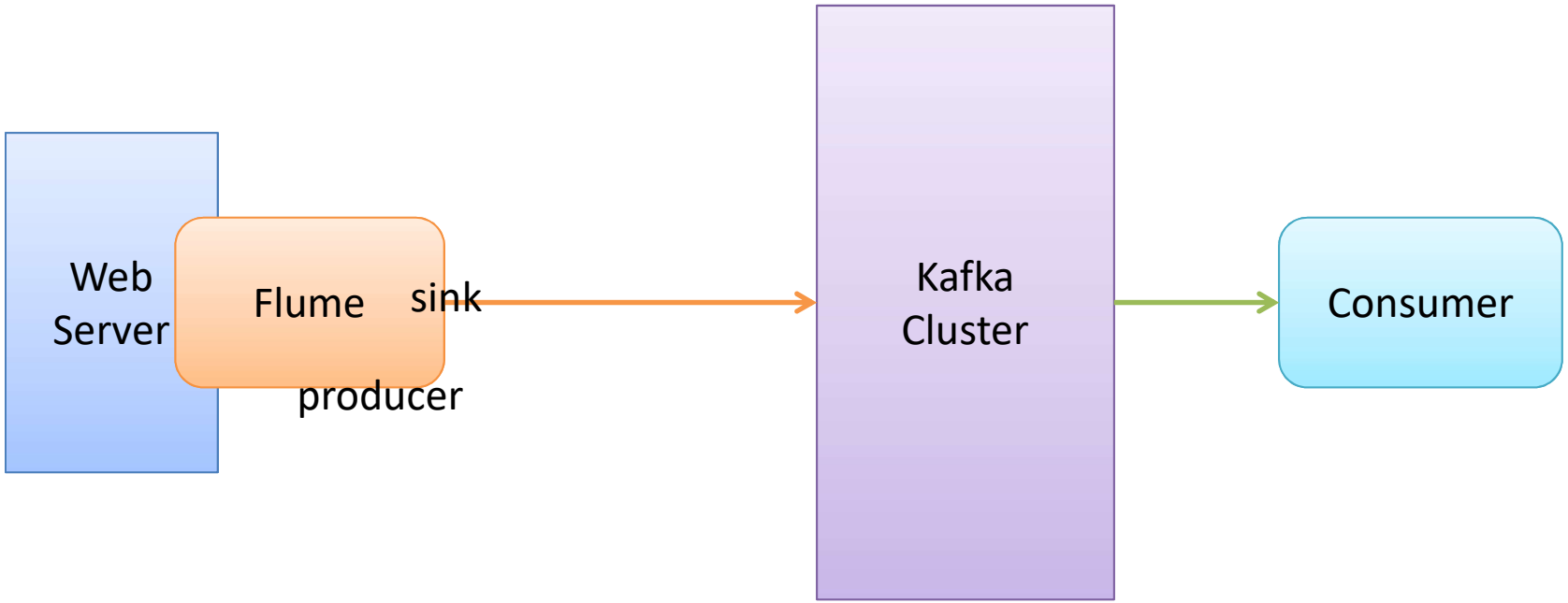
- Spark 2.0開始引入，用來處理stream資料。
  - 底層基於SparkSQL與Catalyst Optimizer
  - 讓使用者可以向使用靜態Dataset開發方式處理stream資料
- 未來將取代Spark Streaming用來處理stream資料

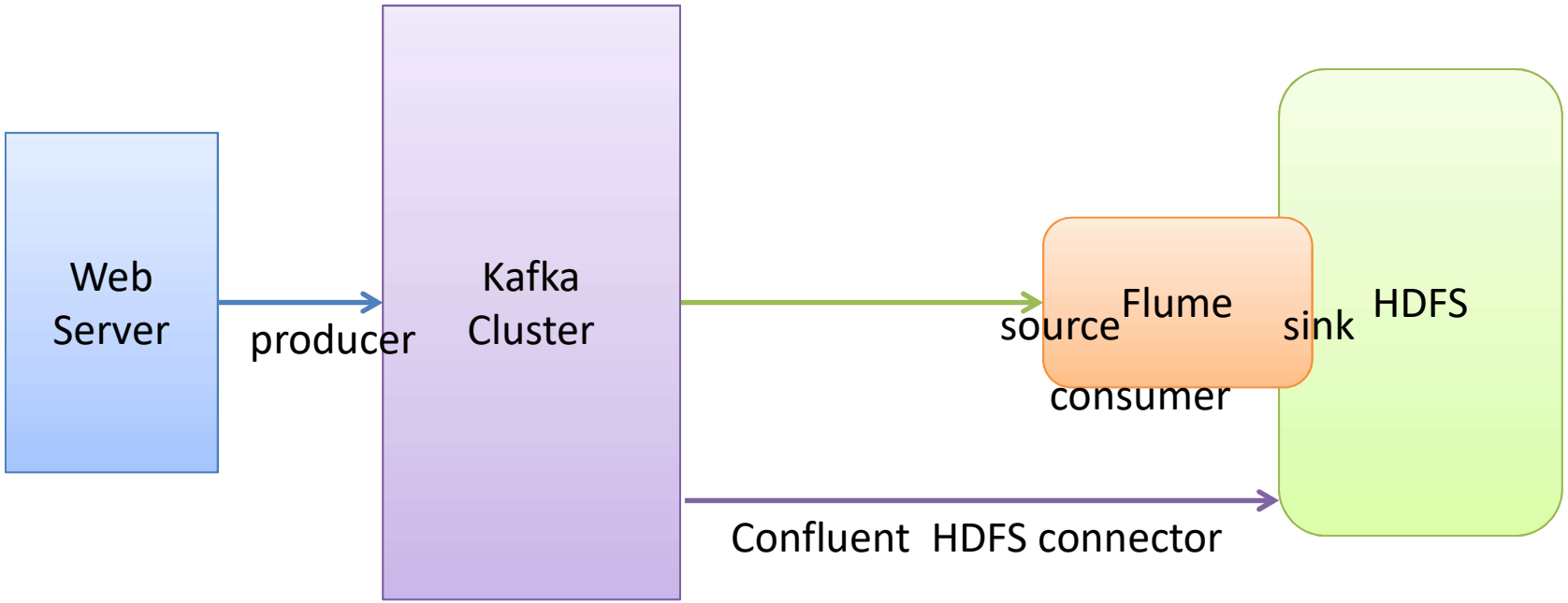
Kafka Architecture



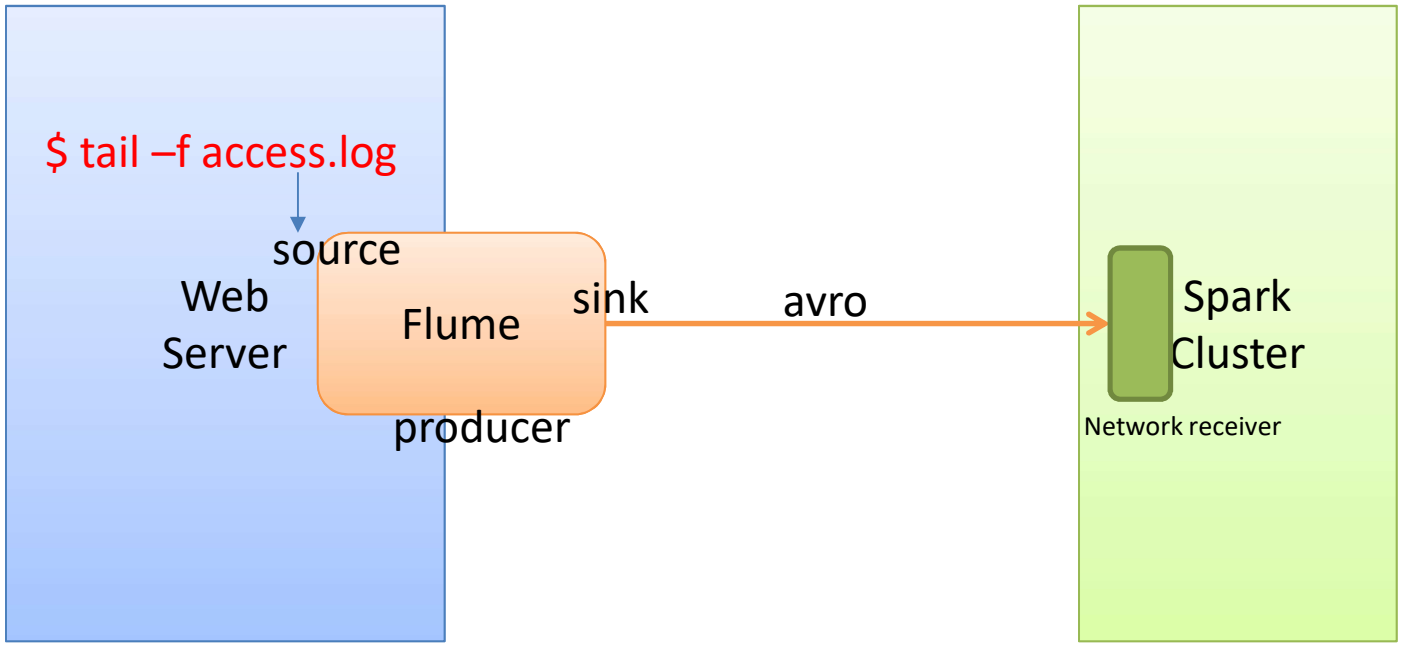
Kafka terminology



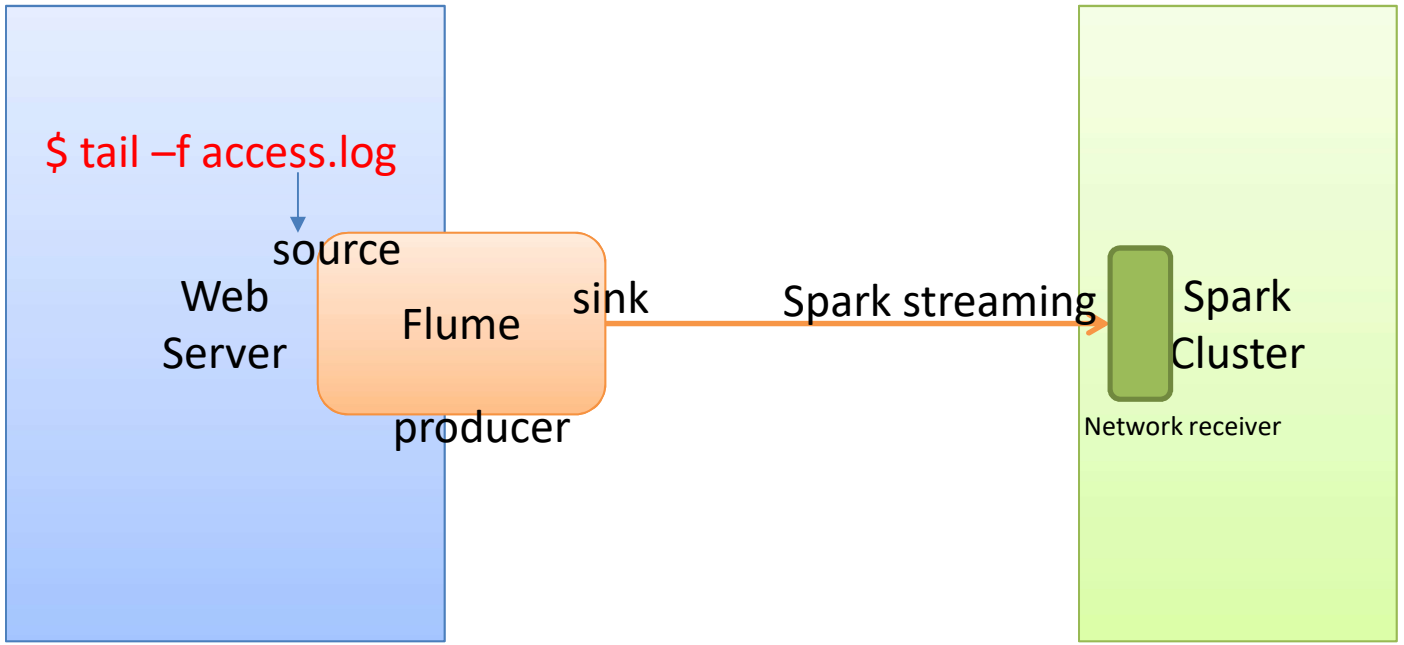

Web
Server
Flume
sink
producer
Kafka
Cluster
Consumer


Web
Server
producer
Kafka
Cluster
source
Flume
sink
HDFS
consumer
Confluent HDFS connector

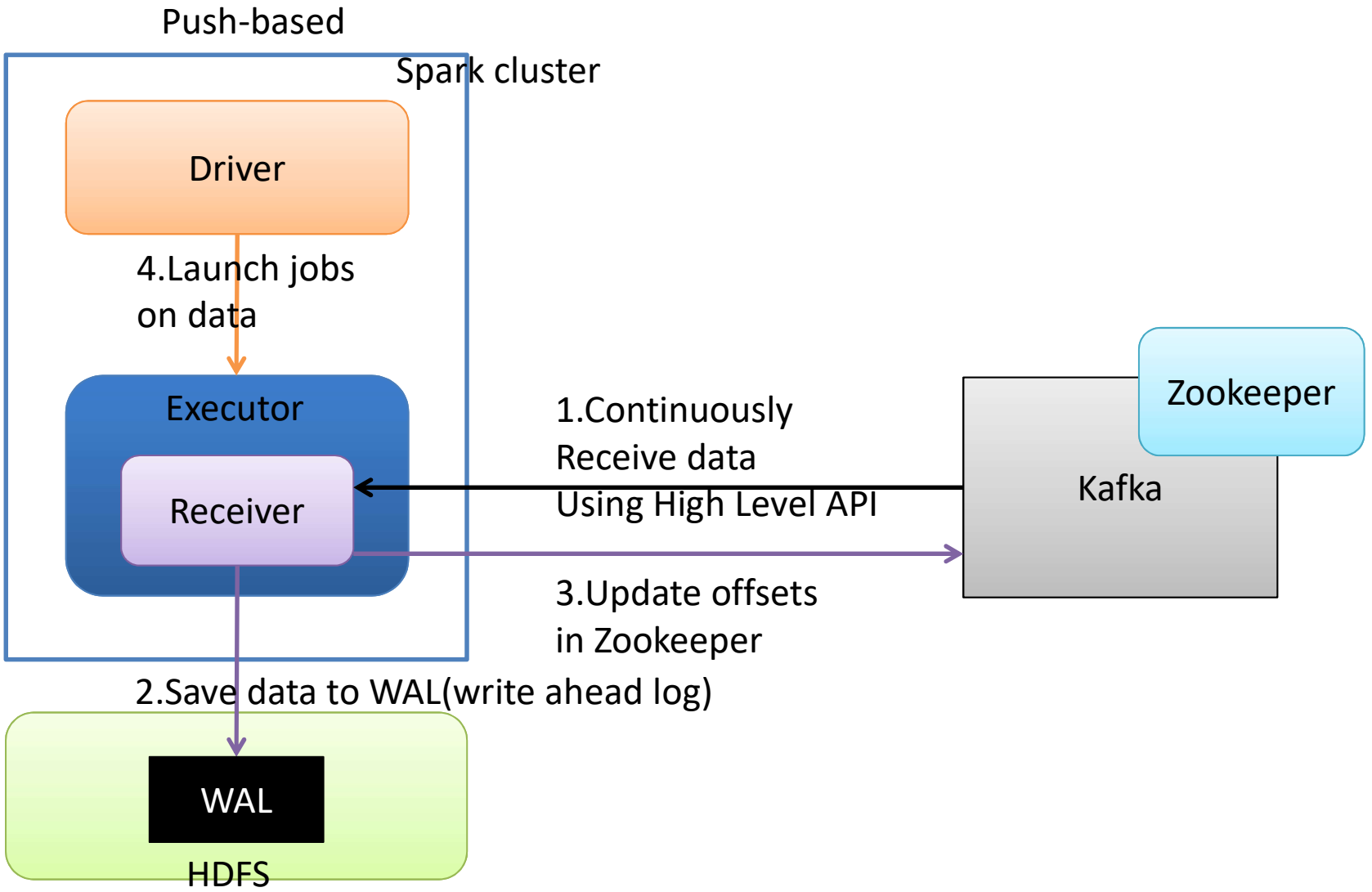
PUSH


$ tail –f access.log
source
Web
Server
Flume
sink
avro
producer
Spark
Cluster
Network receiver

PUSH


$ tail –f access.log
source
Web
Server
Flume
sink
producer
Spark streaming
Spark
Cluster
Network receiver

Kafka with Receiver and WAL



Kafka w/o Receiver and WAL