Agenda

a. Real time problem statement
b. History of data
c. Spark Architecture
d. How to play with Spark ?
e. Spark Advance Concepts

Problem statement

Netflix → IMDB

Qn.1

which Comedy Movie has the more ratings and return the ratings?

Million Movies

100 Million users

① Python / Java

② RDBMS (MySql, Oracle, Postgres ....)

Movies Table

ratings T2

Join

Result

1 TB → ++ 100TB

Request Time out

③ Hadoop (Map Reduce)

↓ Time

Hadoop? = A Distributed System

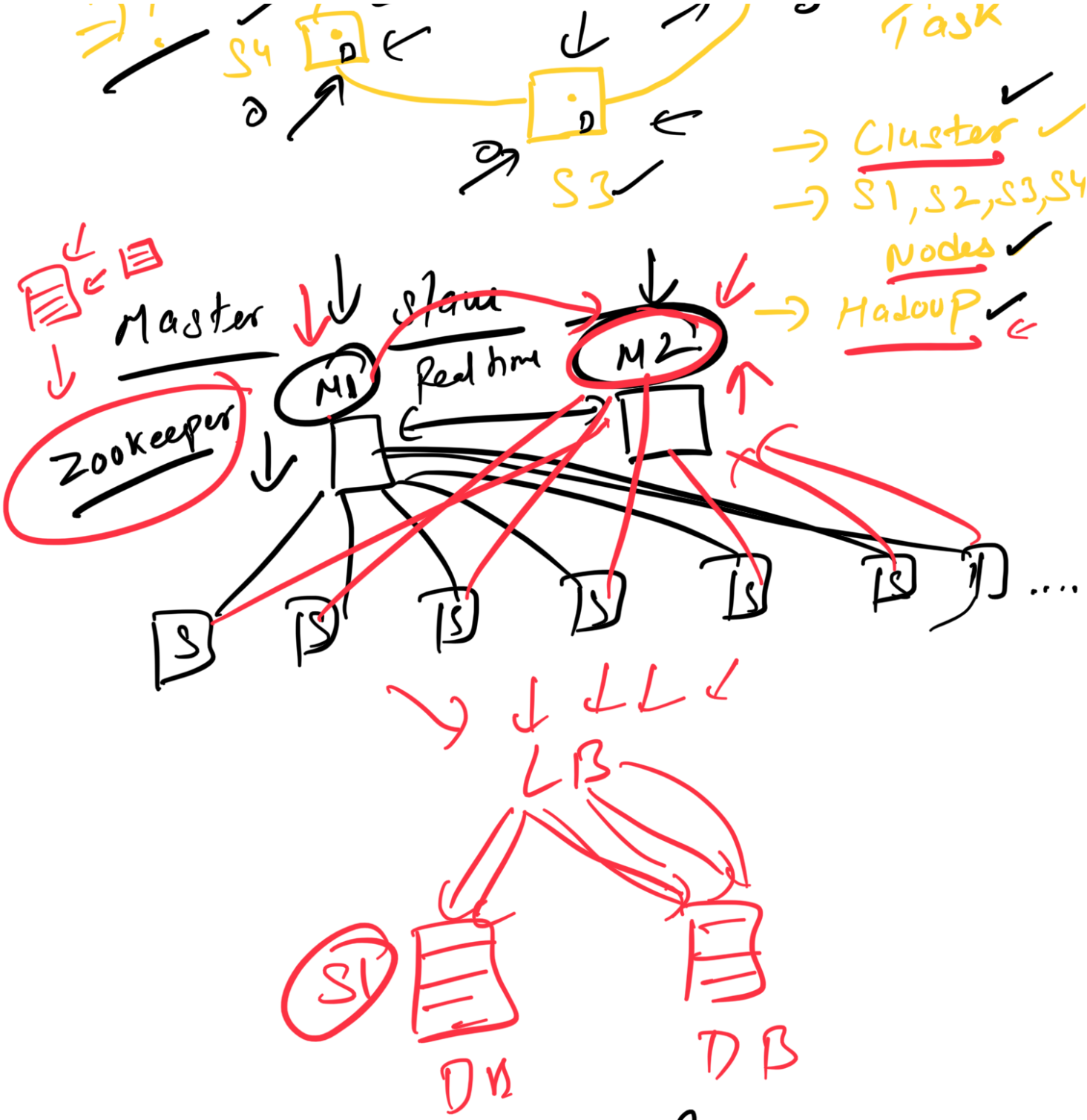
Server 4230 M/C

Cluster

Master

N/W

S2

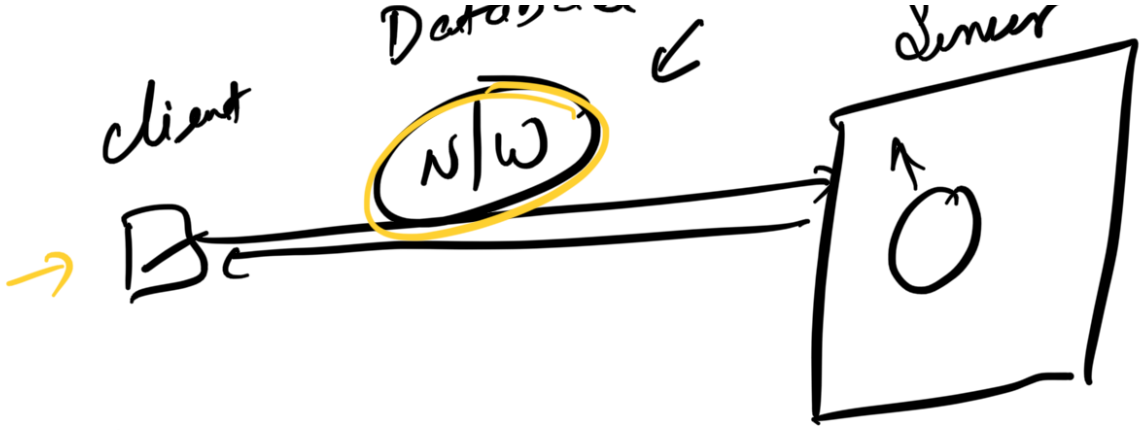


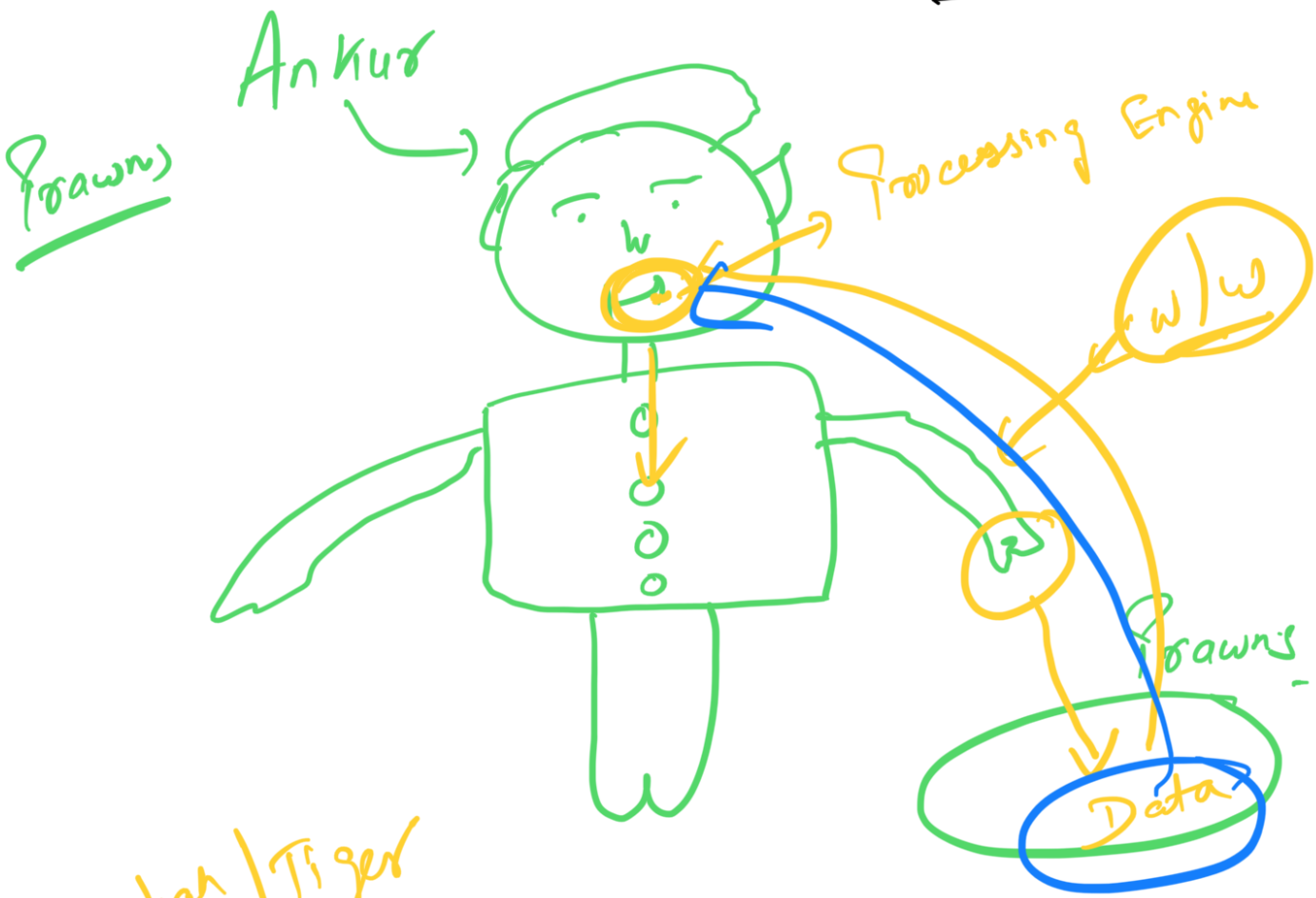
why Hadoop?

1950 → KB, Punch cards

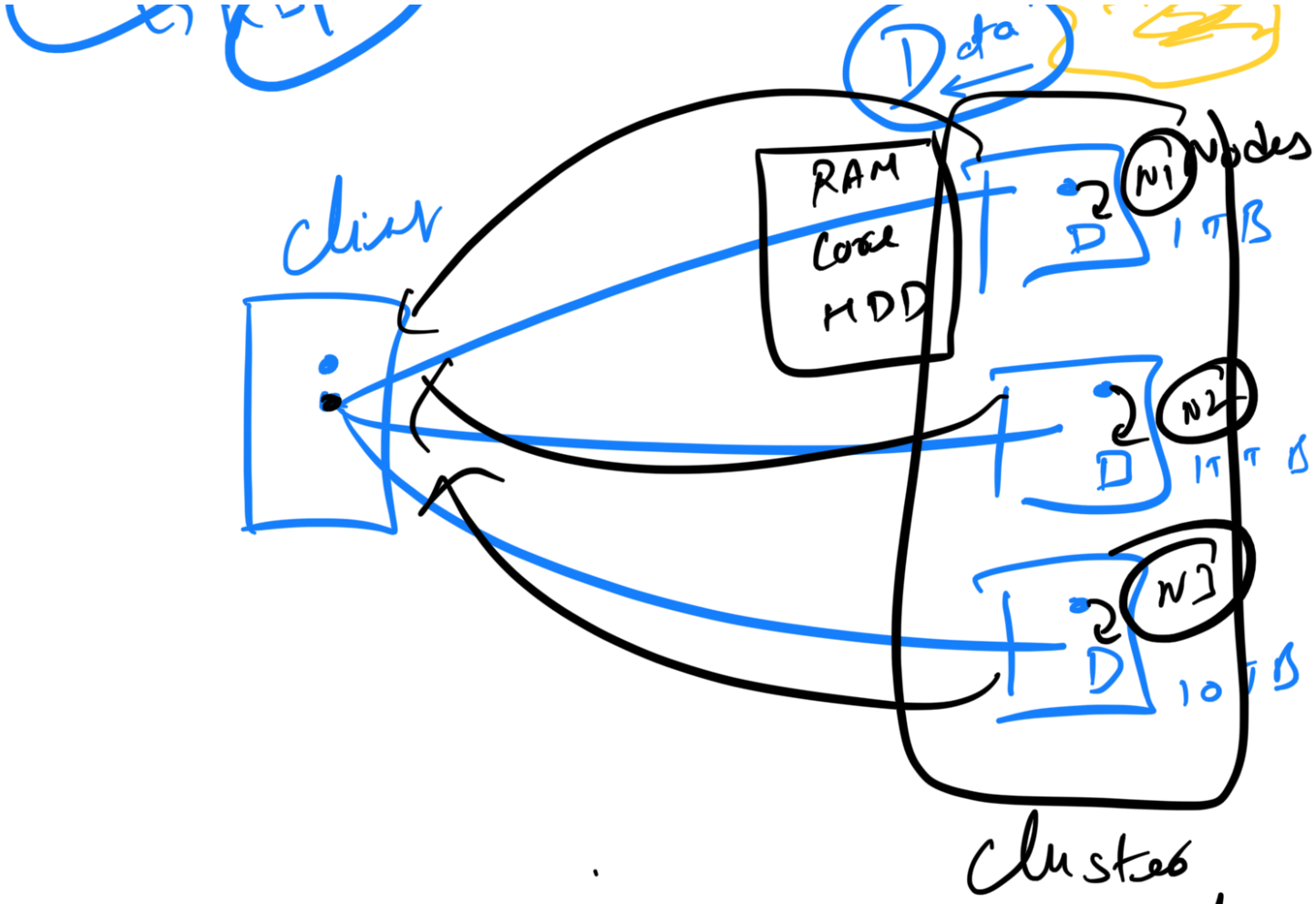
1970 → MB, IBM DB

2000 → GB,

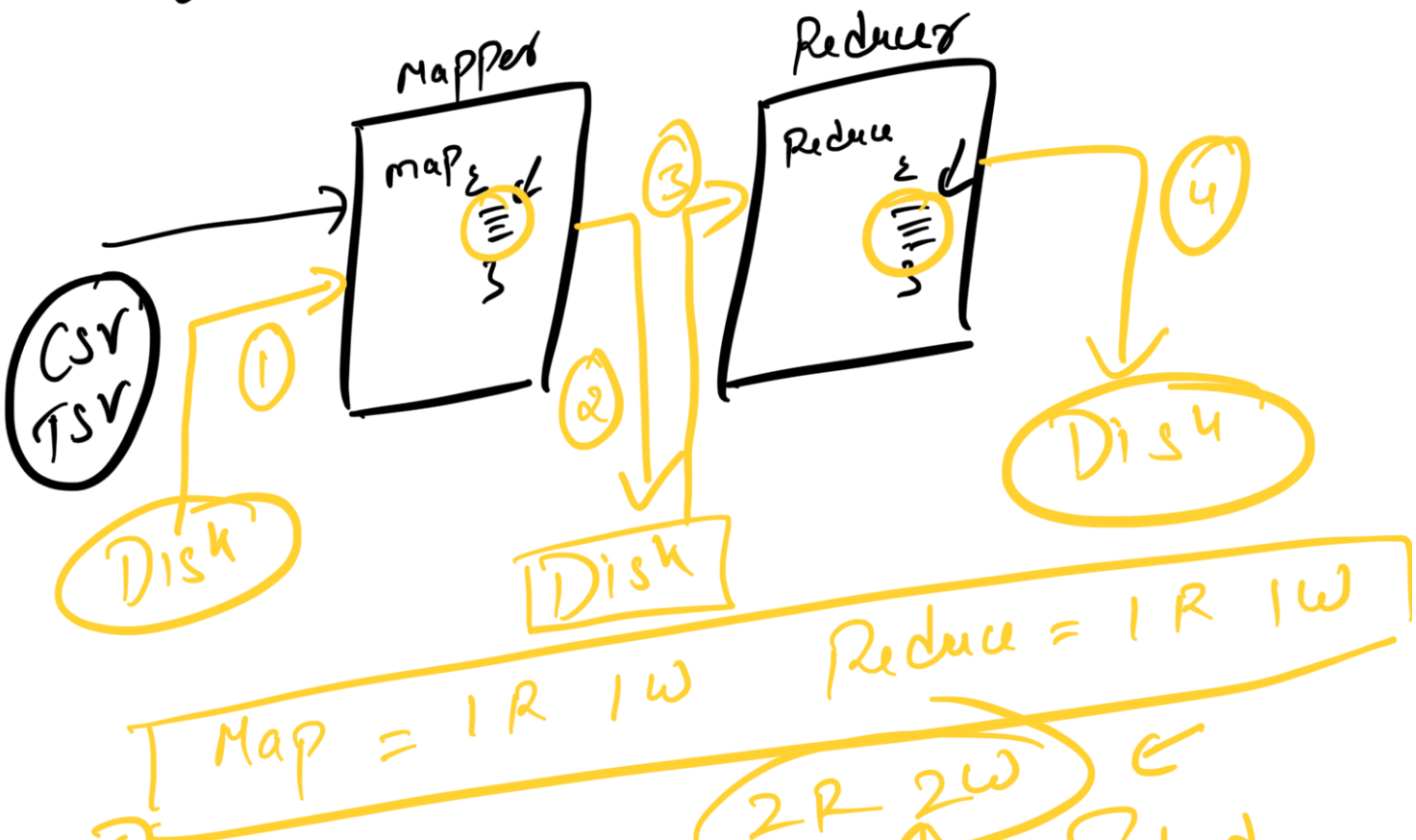

client
N/W
Server


Ankur
Prawns
Processing Engine
N/W
Prawns
Data


Cheetah/Tiger
P.E
Java
Scala
N/W
Deer



Hadoop ⇒ MapReduce (Java / Python)

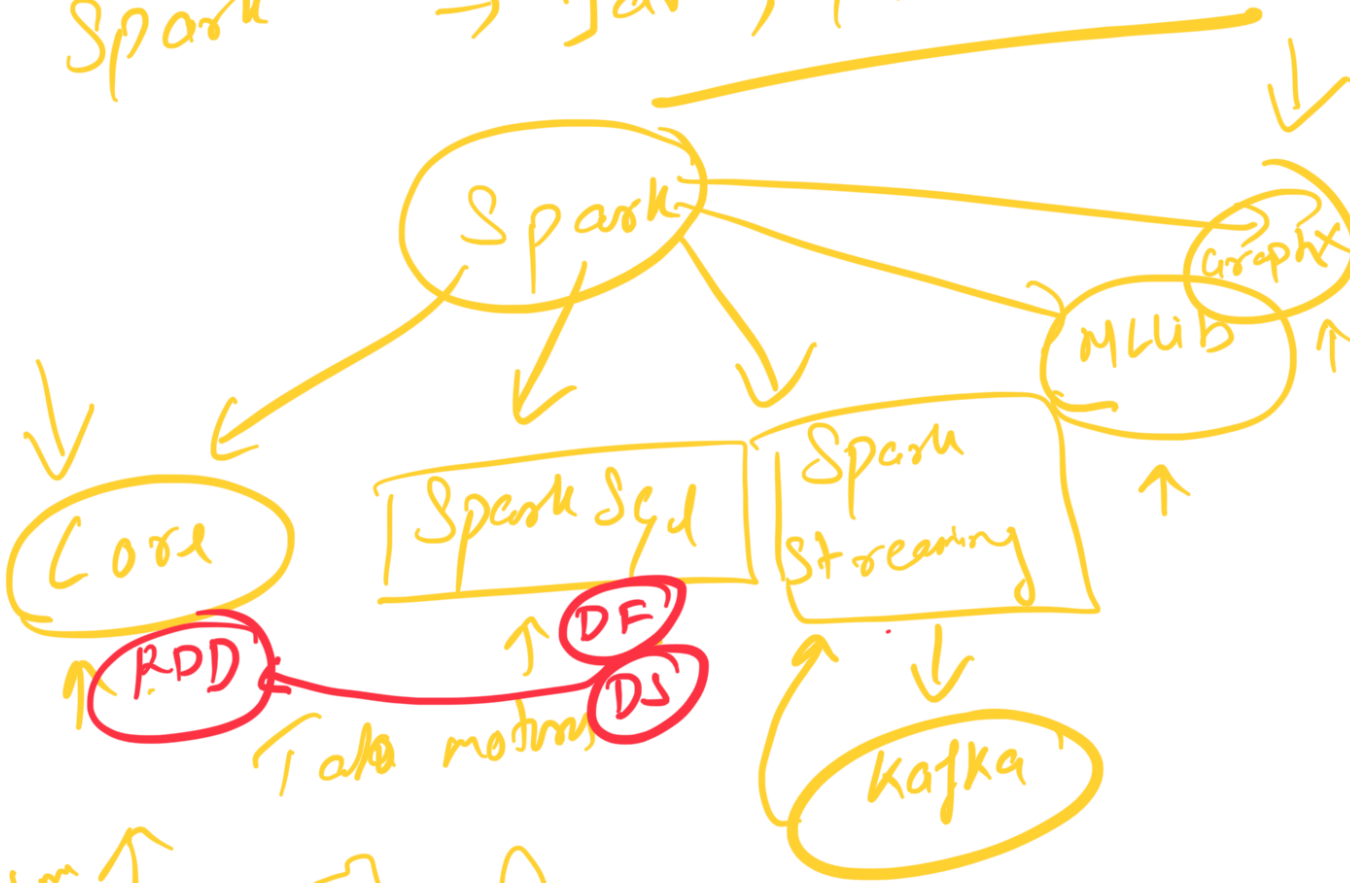


2009
Matai Zehania
UC
Spark
IR, IW
Sothmes
Apache Spark
2011
Databricks
$
2013
Aws Azm GCP
Apache Spark
Databricks
spark
SLA
2-4days
1 week
2 week
opensource
Error
1day
10days
100
IT
P0
P1
P2
Linux
Redhat
Apache Spark
MapReduce

↳ Open Source analytics engine for large scale data processing and doing as min 100 times faster as compared to map Reduce.

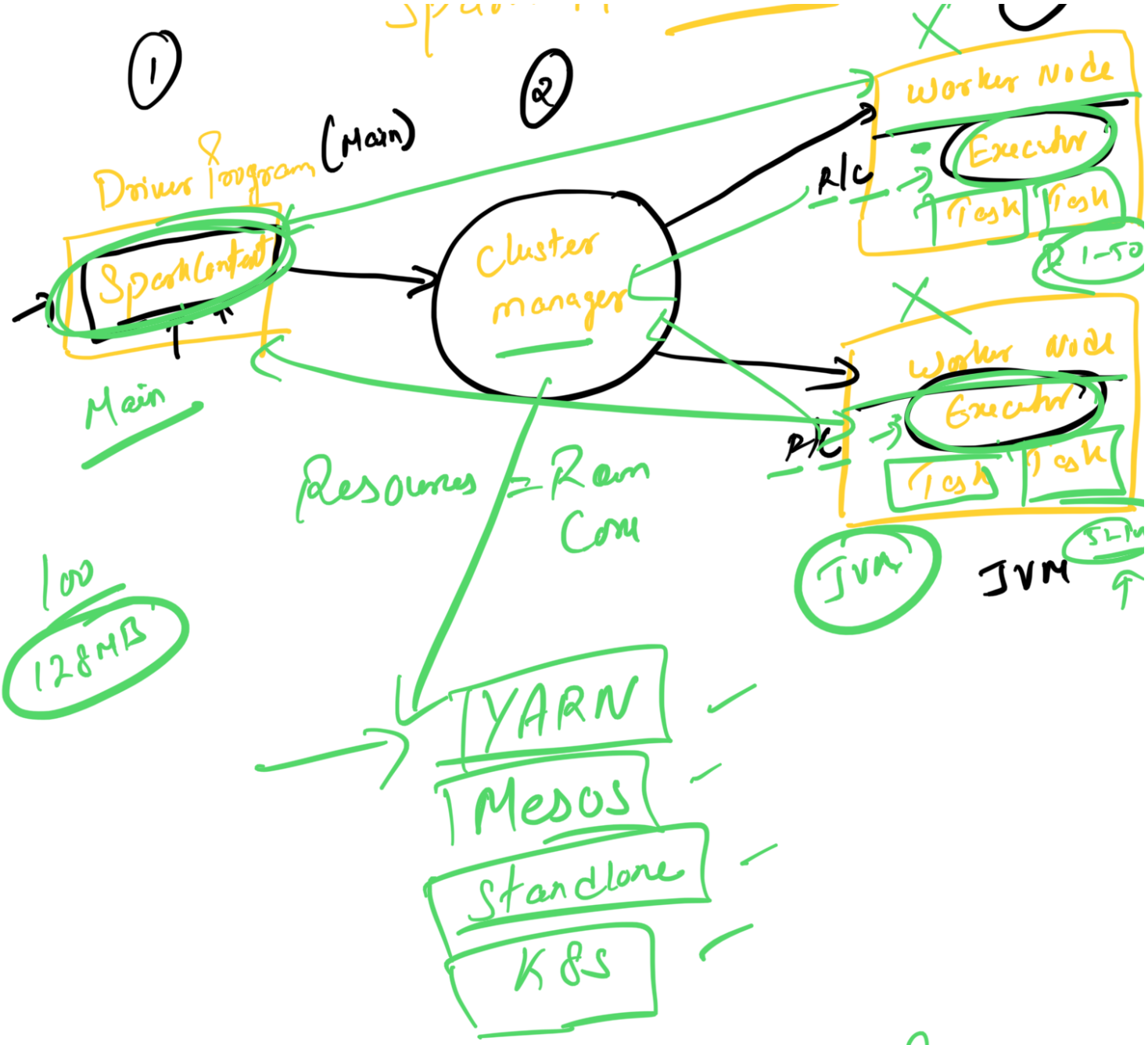
1. IR IW
2. Memory Based Computations

map Reduce → Java
Spark → Java, Python, R, Scala



Take motion

Spark Architecture



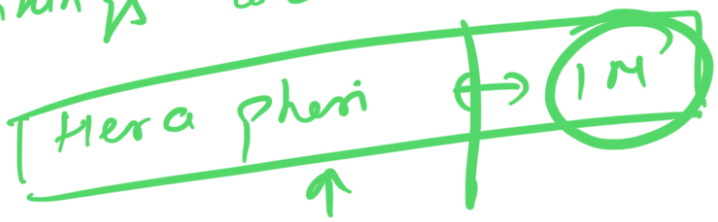
How to interact with Spark?

① Interactive :- Spark-shell → Scala
Pyspark → Python
SparkR → R

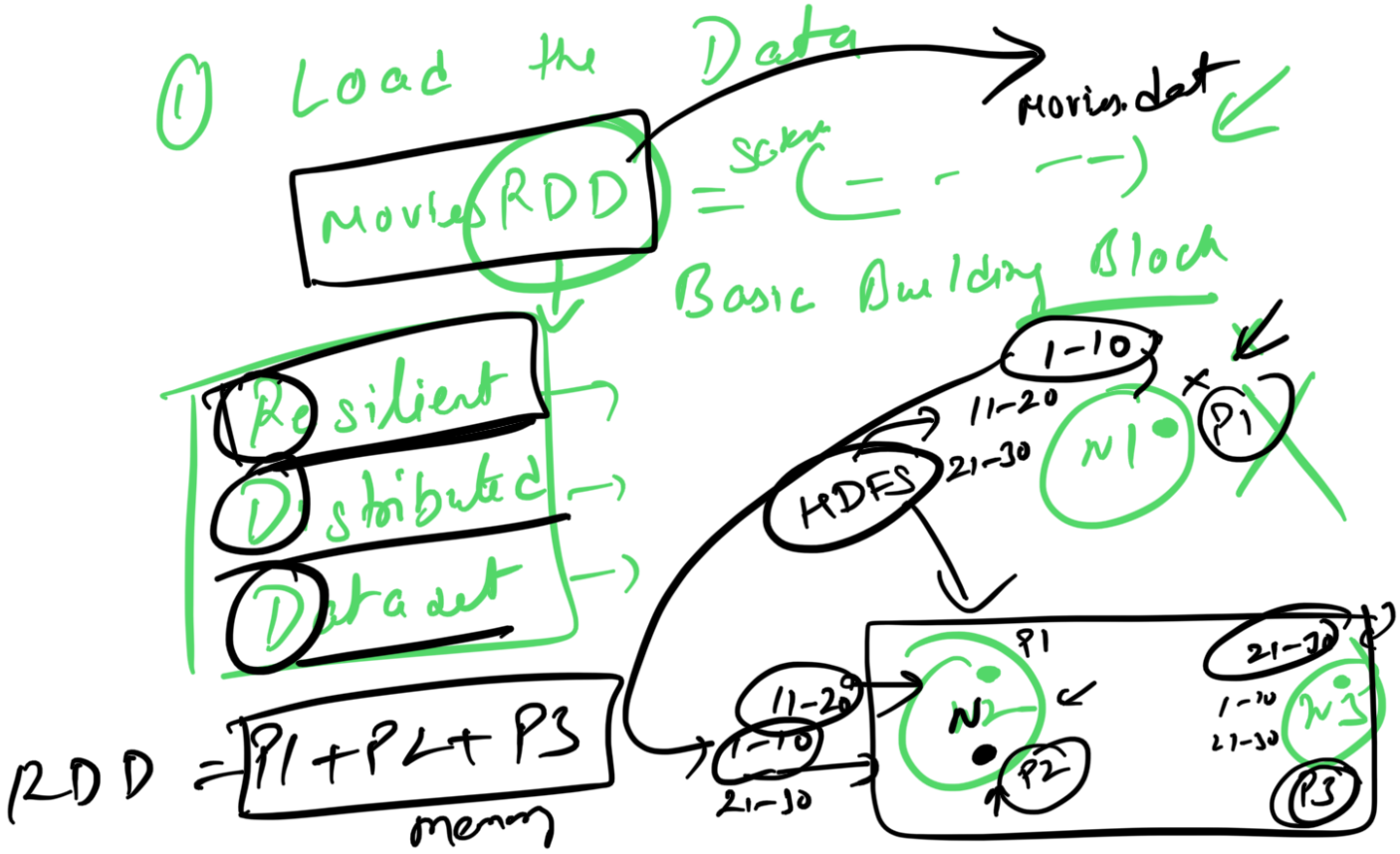
Development & Testing

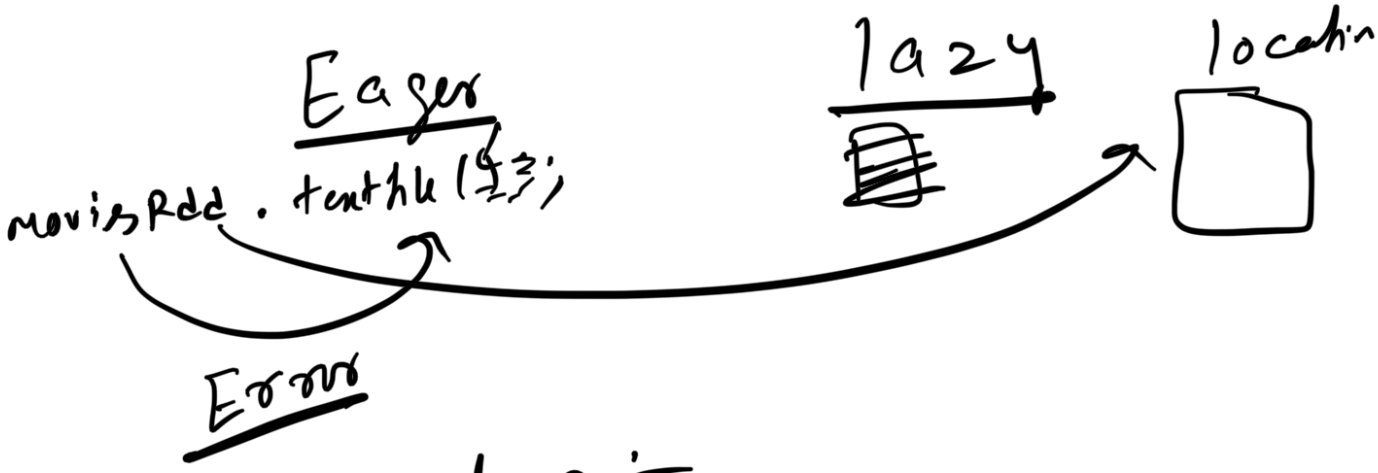
② Application :- Spark-Submit → .py

Qn.1 which Comedy movies hast most rankings and return the no of ranking.

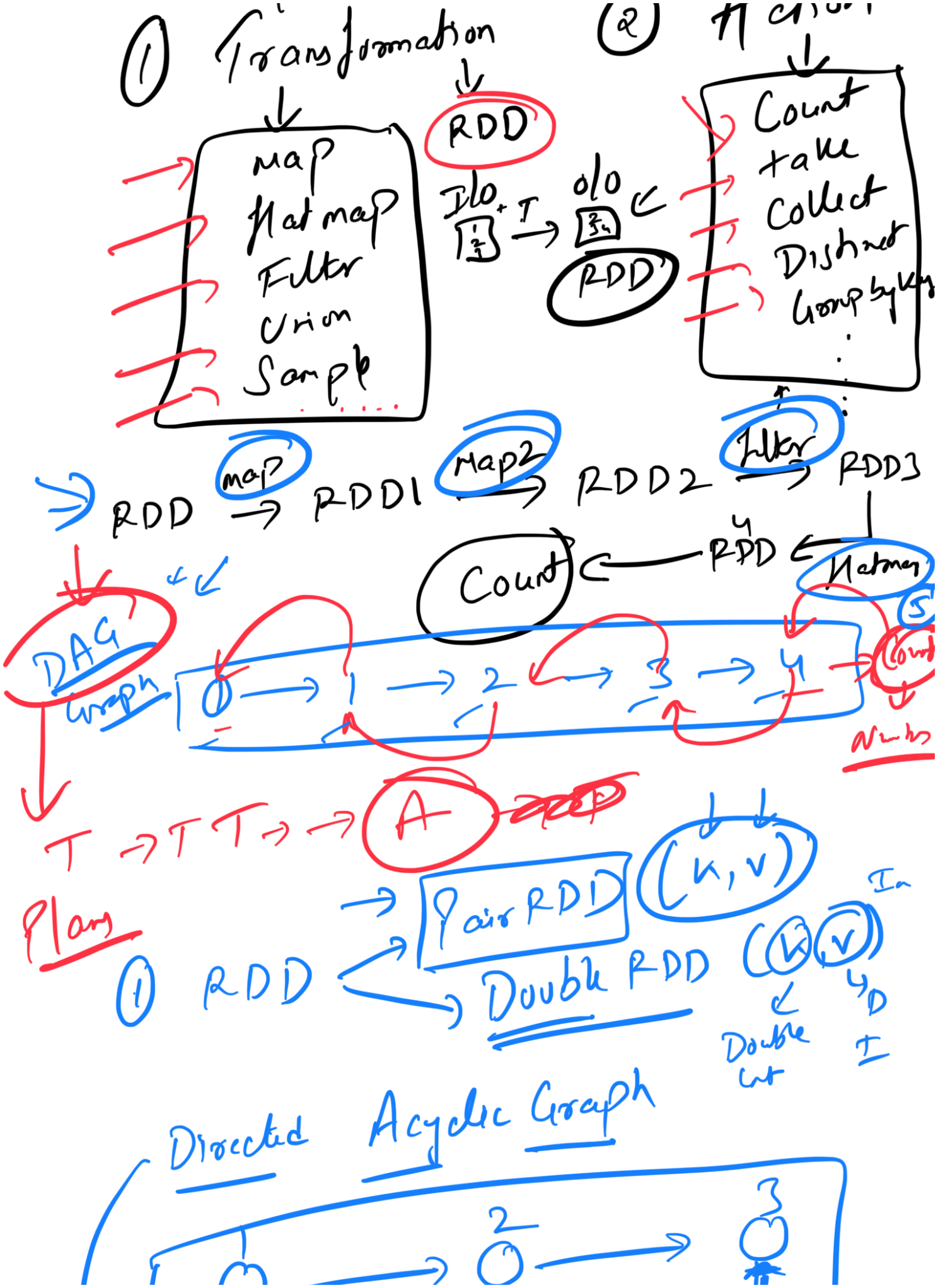


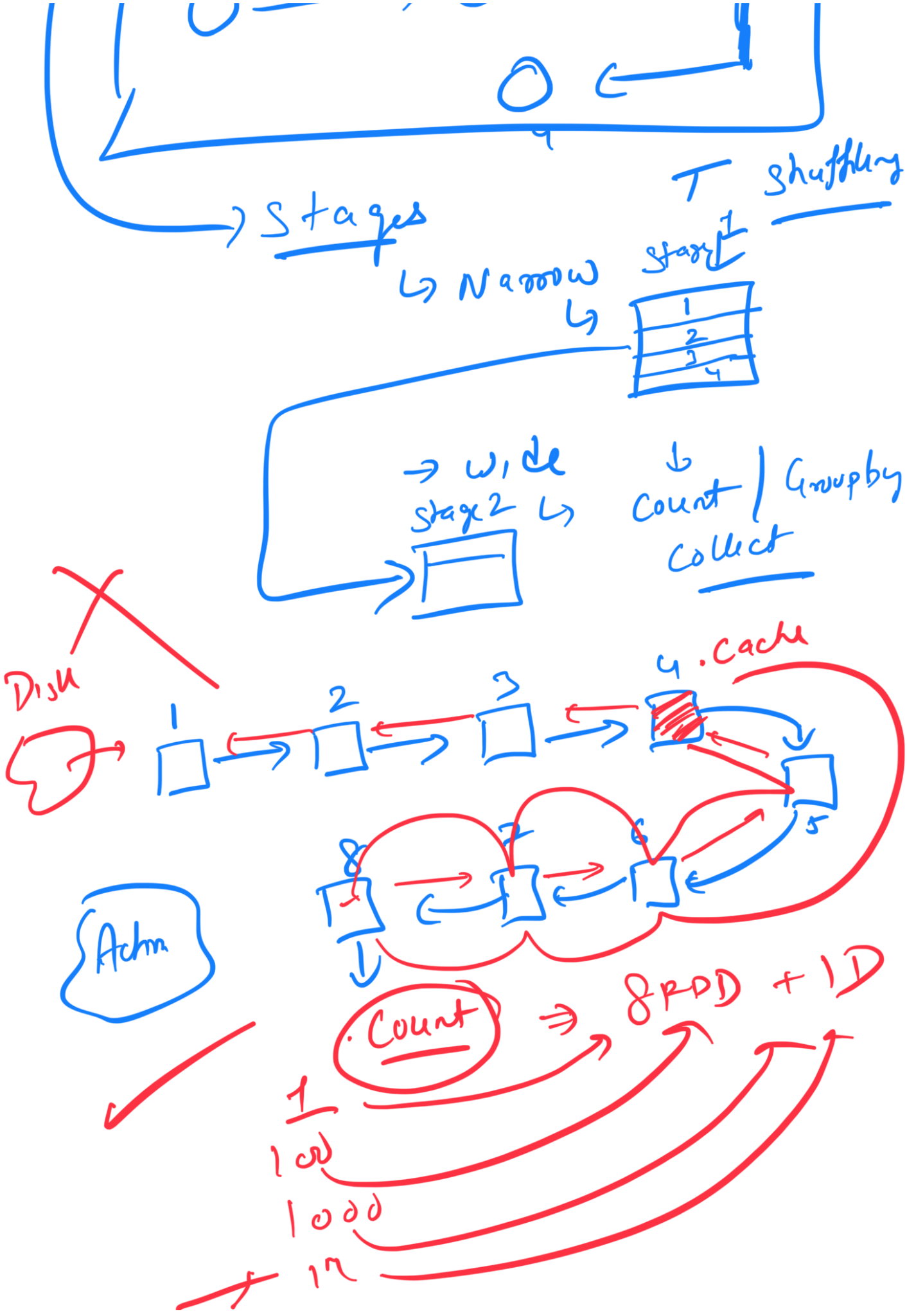
An.1 Spark Context





operation :-


1 Transformation
RDD
map
flatmap
Filter
Union
Sample
I/O
I
O/O
RDD
2
Count
take
collect
Distinct
GroupByKey
map
RDD
RDD1
Map2
RDD2
Filter
RDD3
Count
RDD
flatmap
DAG
Graph
0
1
2
3
4
Actions
T → T T → A
Plans
1 RDD
Pair RDD
(K, V)
Double RDD
K
V
Double cut
I
Directed Acyclic Graph
2
3


Stages
↳ Narrow
stage1
1
2
3
4
T shuffling
→ wide
stage2 ↳
↳
Count | Groupby
Collect
Disk
1
2
3
4 . cache
5
8
2
6
Actn
Count
⇒ 8 RDD + 1D
1
100
1000
+ 17

.cache()