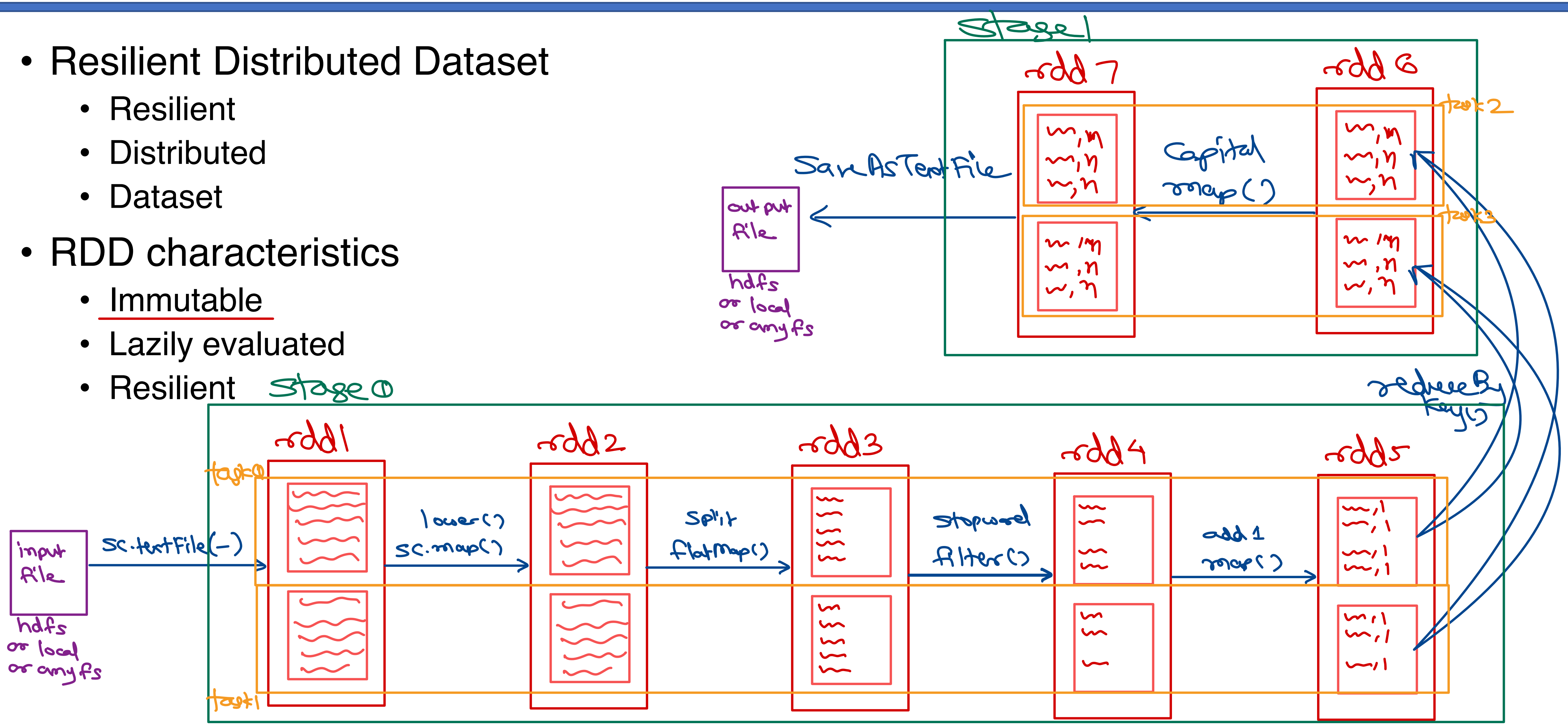
Apache Spark

Sunbeam Infotech

Spark RDD

- Resilient Distributed Dataset
  - Resilient
  - Distributed
  - Dataset
- RDD characteristics
  - Immutable
  - Lazily evaluated
  - Resilient

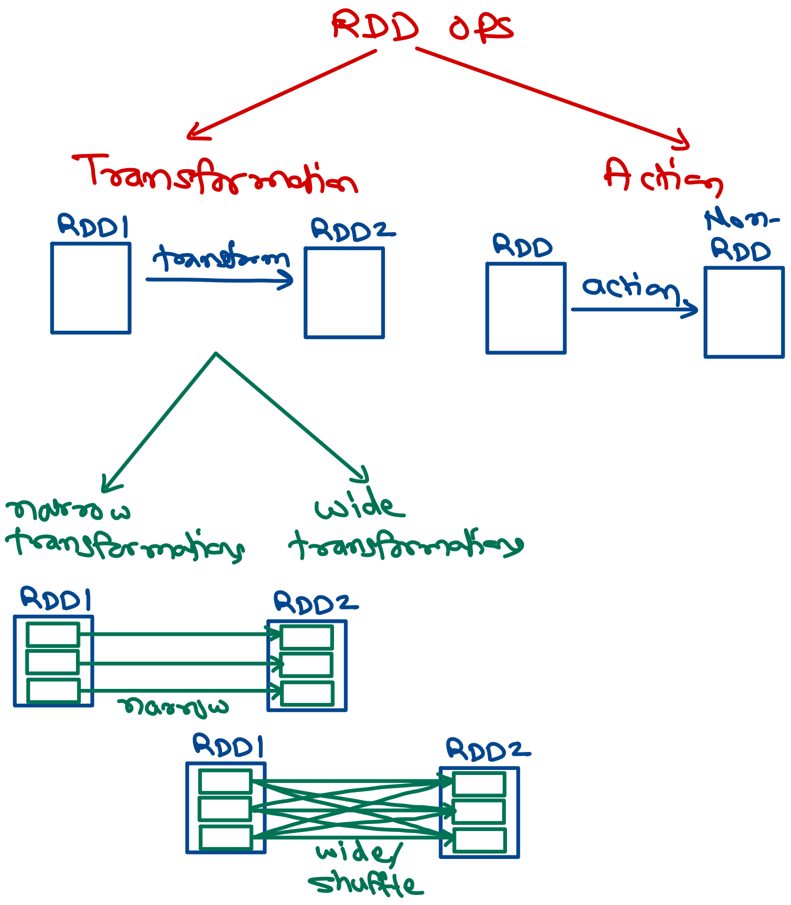


Spark RDD – creation

- sc.parallelize(collection, partitions)
  - convert collection into rdd with given partitions
- sc.textFile(path) → file:/// → hdfs:// → s3://
  - hdfs or local or s3 file or directory
- sc.wholeTextFiles(path)
  - hdfs or local or s3 directory, one file = one record
- sc.binaryRecords(path, recLen)
  - hdfs or local or s3 file or directory, recLen bytes = one record
- sc.wholeBinaryFiles(path)
  - hdfs or local or s3 directory, one file = one record
- sc.hadoopFile(path, inputFormat, …)
  - hdfs file or directory, number of partitions = number of input splits

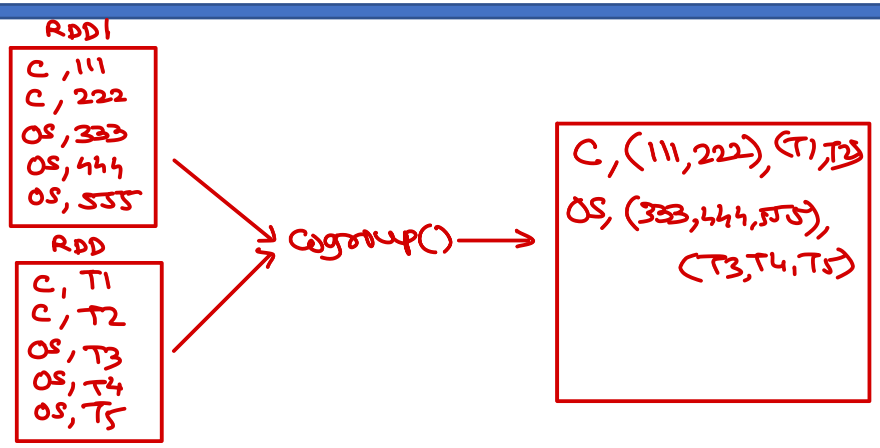
≈ number of hdfs blocks

RDD Operations



RDD Operations

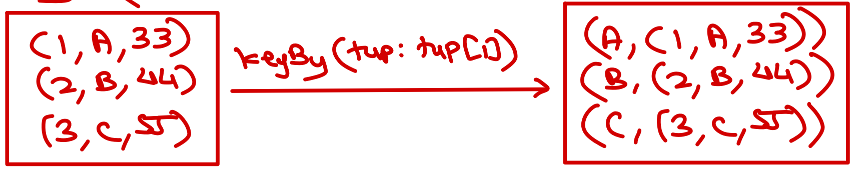
- Transformations: Returns RDD (narrow + wide)
  - distinct()
  - filter()
  - map()
  - flatMap()
  - sortBy(keyValue, ascending, numOfPartitions) – shuffle
  - sortByKey() – shuffle
  - pipe() – similar to hadoop streaming (execute program in other lang).
  - keyBy() → add to KV pair RDD
  - countByKey() → action
  - mapValues()
  - groupByKey() → shuffle
  - aggregateByKey() → shuffle
  - cogroup()
  - zip()
  - join() – joins two RDDs by key.
  - reduceByKey()



C, 111
C, 222
OS, 333
OS, 444
OS, 555

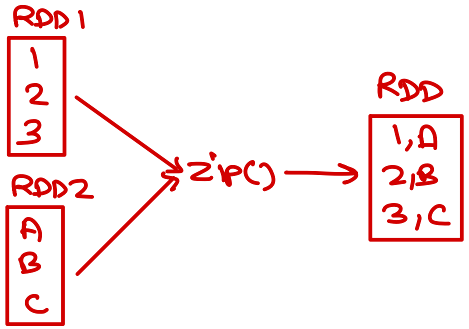
C, T1
C, T2
OS, T3
OS, T4
OS, T5

C, (111, 222), (T1, T2)
OS, (333, 444, 555), (T3, T4, T5)



(1, A, 33)
(2, B, 44)
(3, C, 55)

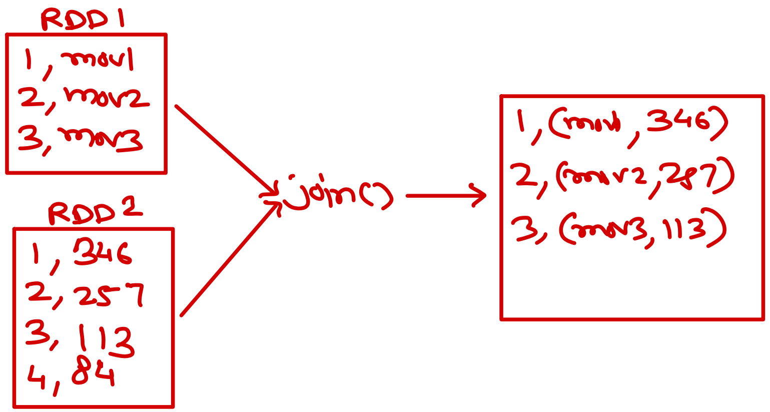
(A, (1, A, 33))
(B, (2, B, 44))
(C, (3, C, 55))



1
2
3

A
B
C

1, A
2, B
3, C



1, mov1
2, mov2
3, mov3

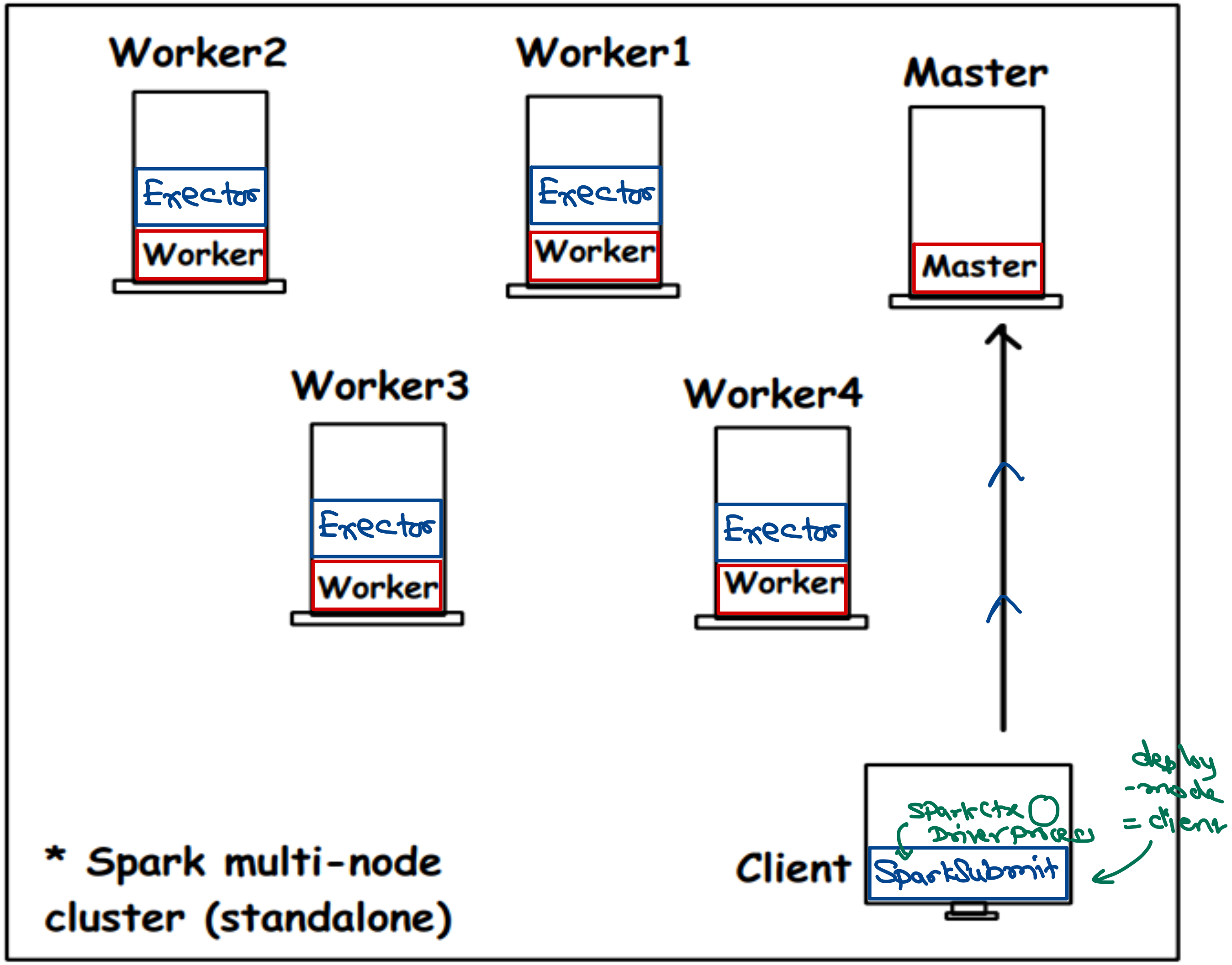
1, 346
2, 257
3, 113
4, 84

1, (mov1, 346)
2, (mov2, 287)
3, (mov3, 113)

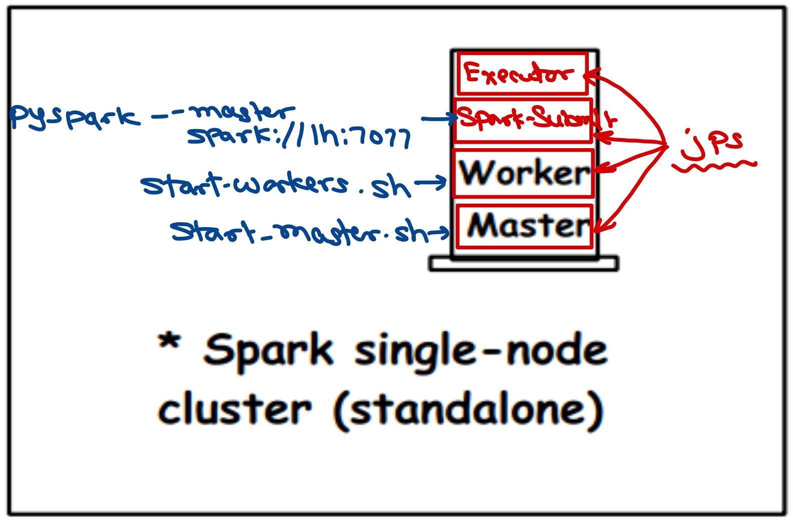
RDD Operations

- Transformations
  - Narrow transformations
    - A partition of new RDD is computed from single partition of source RDD
  - Wide transformations
    - Partition of new RDD is computed from multiple partitions of source RDD
    - These transformations cause shuffling data across partitions.
- Actions: Returns non-RDD
  - count()
  - countApprox()
  - reduce()
  - countByValue()
  - first()
  - max()
  - min()
  - collect()
  - take() – collect n elements
  - takeOrdered()
  - top()
  - saveAsTextFile()
  - saveAsObjectFile() – seq file
  - lookup() – lookup by key

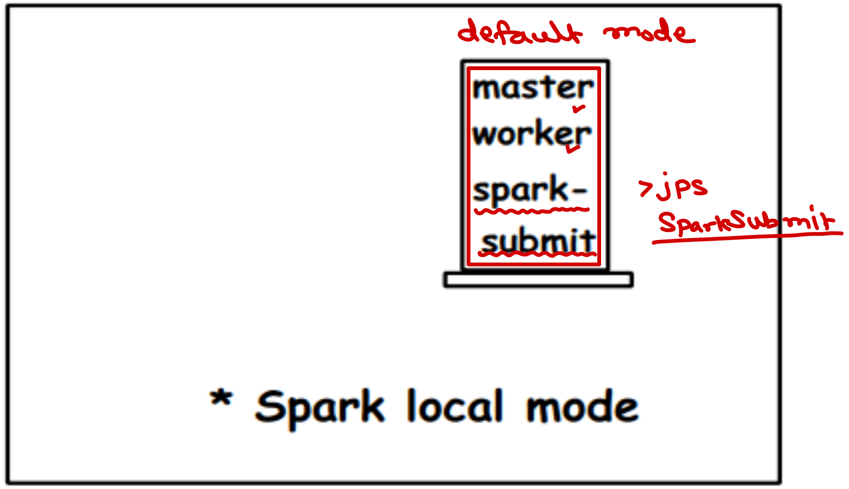
Spark Installation Modes



* Spark multi-node cluster (standalone)

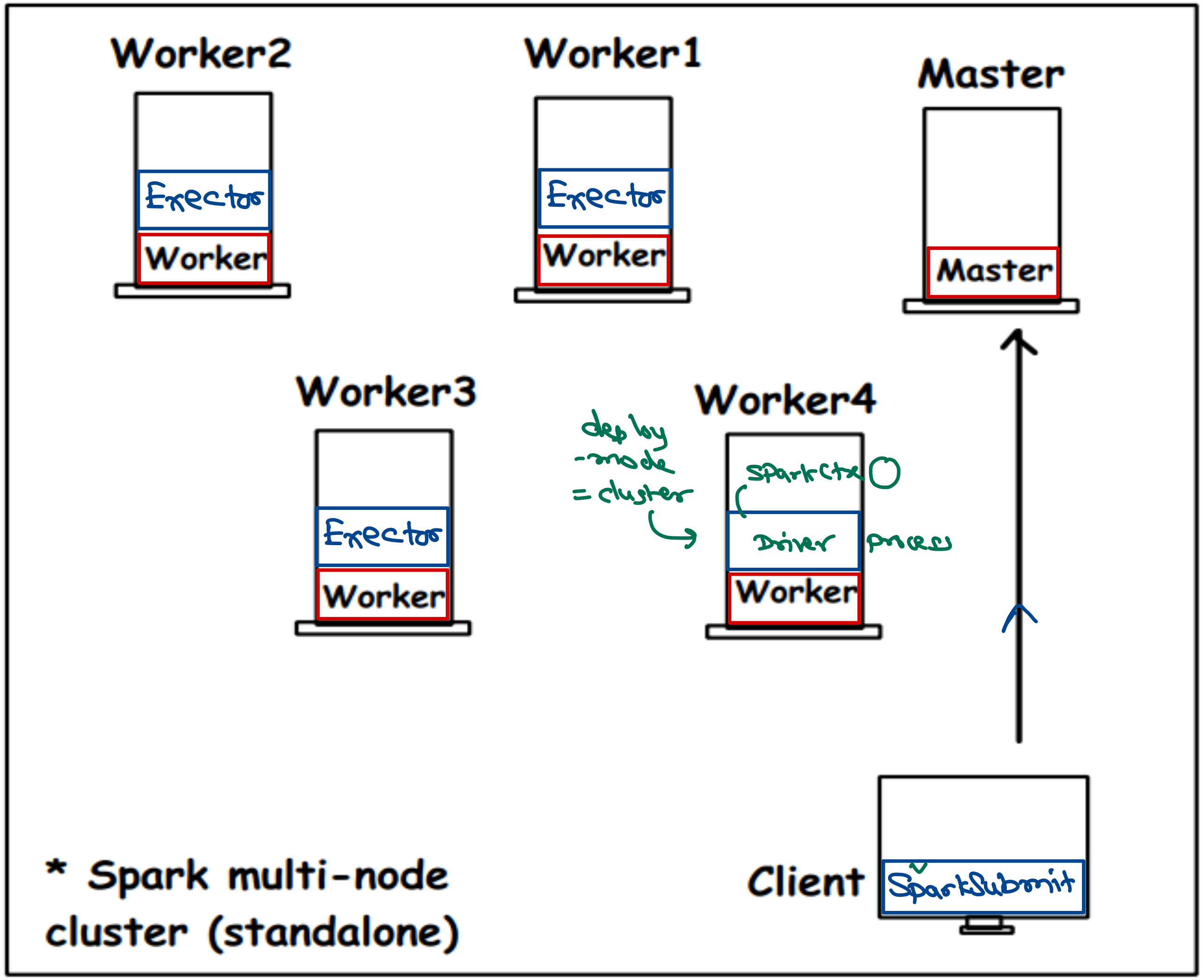


* Spark single-node cluster (standalone)

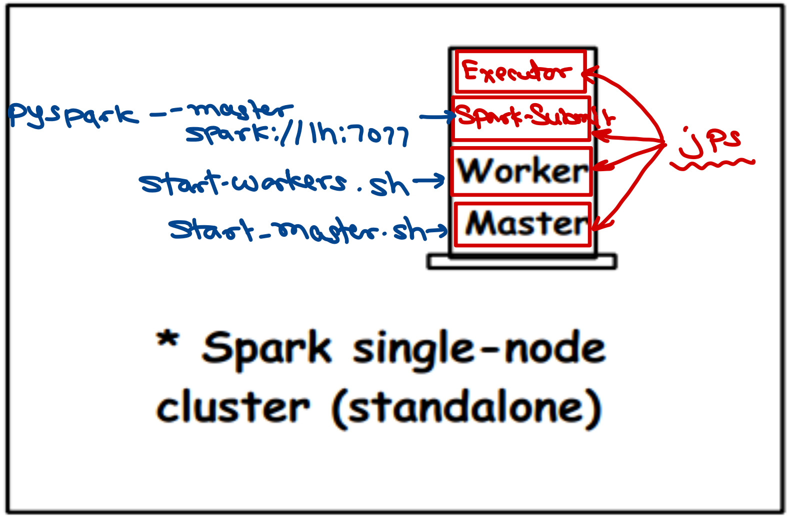


* Spark local mode

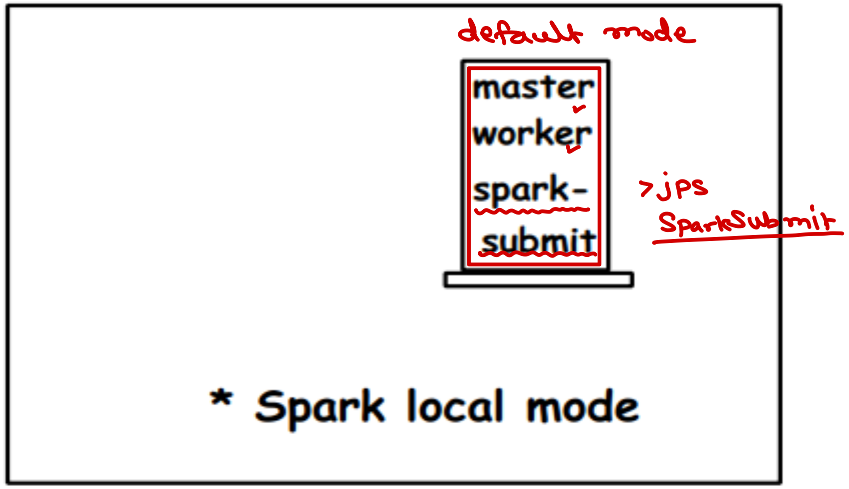
Spark Installation Modes



* Spark multi-node cluster (standalone)



* Spark single-node cluster (standalone)



* Spark local mode

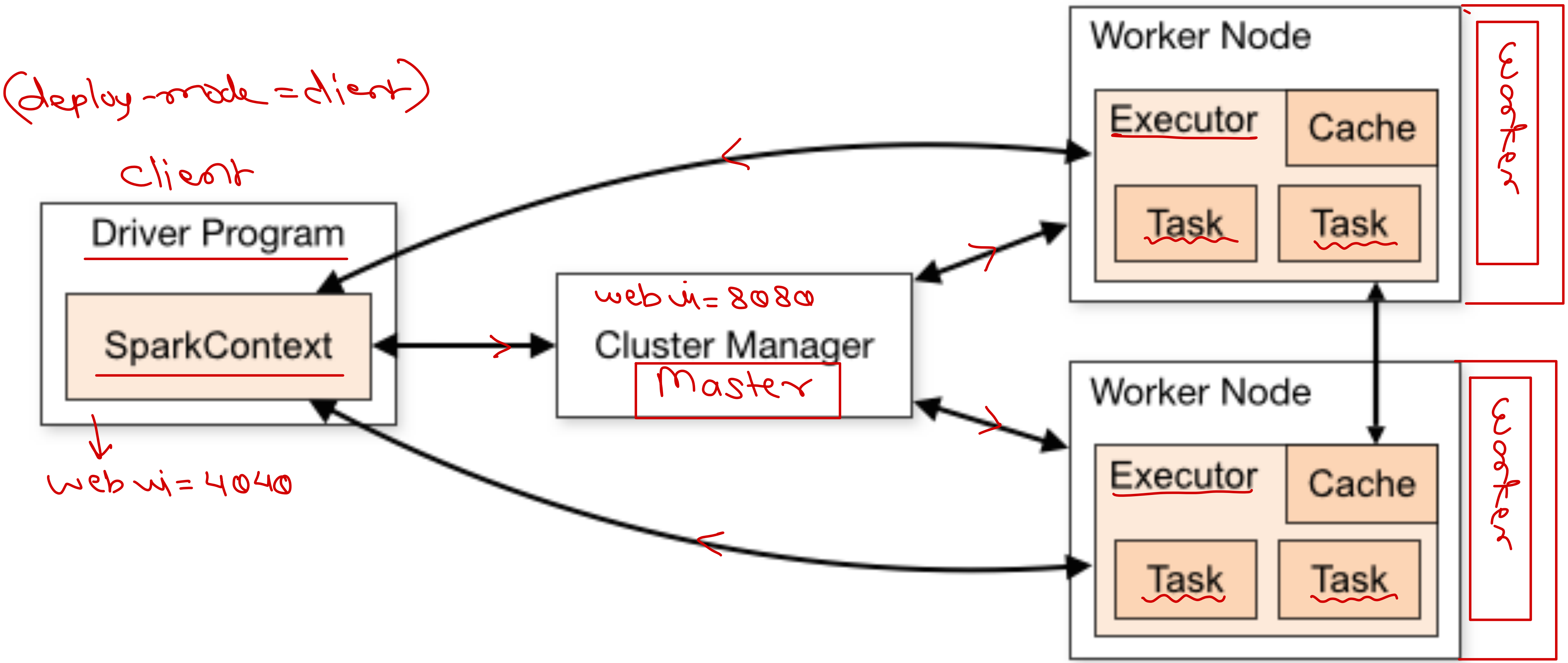
Spark Single Node Cluster

- Download & extract spark-x.y.z-bin-hadoop2.7.tgz.
- In ~/.bashrc
  - export SPARK_HOME=$HOME/spark-x.y.z-bin-hadoop2.7
  - export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
- In $SPARK_HOME/conf/spark-env.sh
  - export SPARK_MASTER_HOST=localhost
  - export SPARK_LOCAL_IP=localhost
- In $SPARK_HOME/conf/spark-defaults.conf
  - spark.master spark://localhost:7077
- Start master & slaves.
  - terminal> start-master.sh
  - terminal> ~~start-slaves.sh~~ start-workers.sh
  - terminal> jps
- Using cluster:
  - pyspark --master spark://localhost:7077 ✓
  - spark-submit

$SPARK_HOME/conf/workers

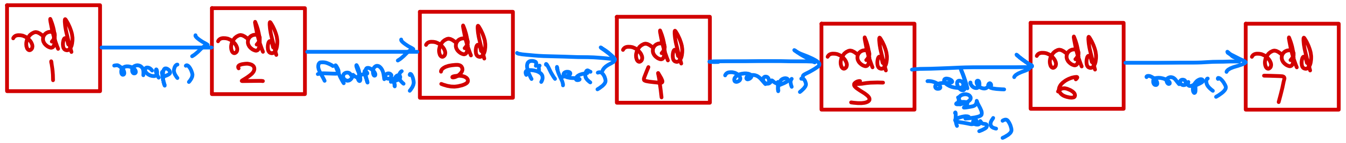
localhost

Spark Cluster Manager - spark standalone mode



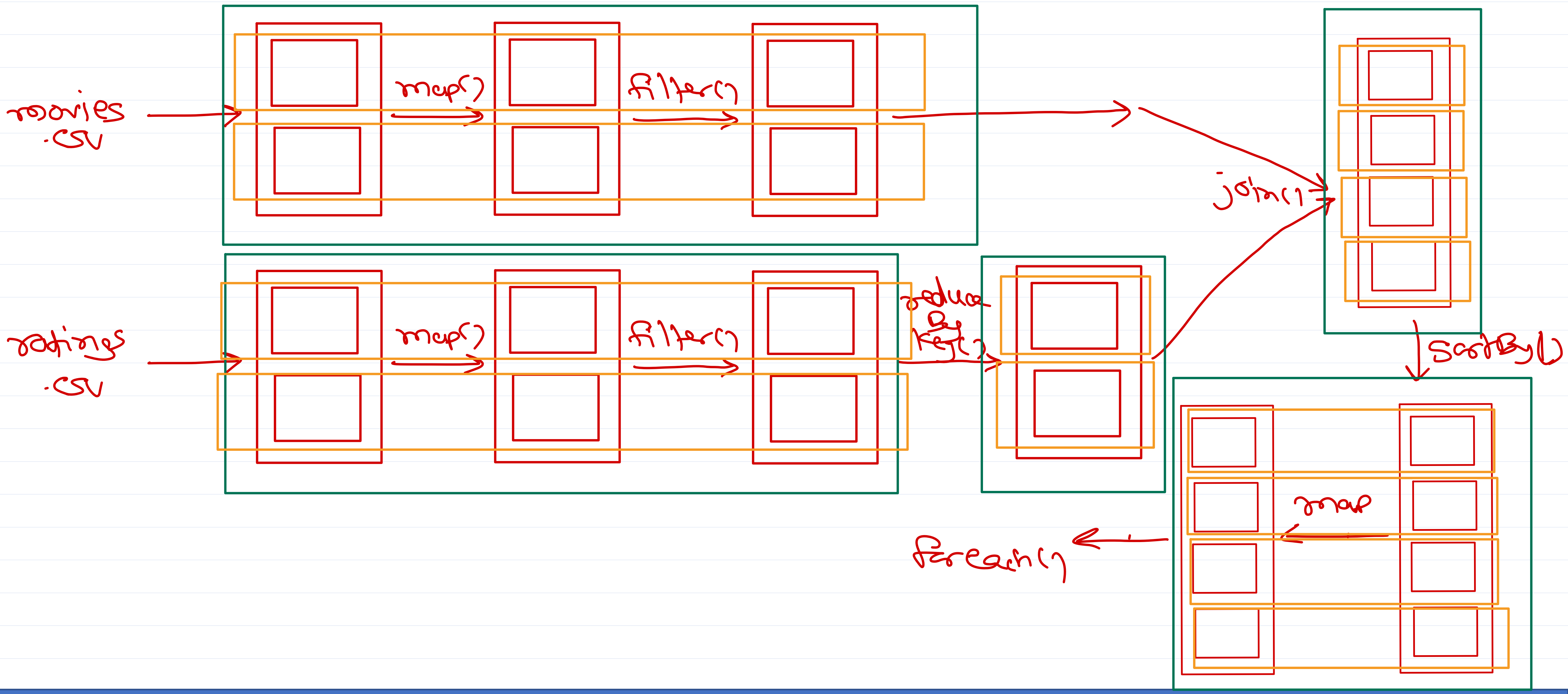
Spark Execution – Terminologies

- RDD: Resilient Distributed Dataset
  - Distributed across nodes -- Partition.
- DAG: Execution plan
  - Graph: Nodes=RDDs, Edges=Operations
- SparkContext: In-charge of execution
  - Maintain all info for execution/communication
  - Creates RDDs
- Application: Set of jobs
- Job: A DAG of execution
  - An action triggers job
- Stages: DAG is divided into stages
  - Stages are separated by shuffle operations (wide transformations)
- Task: Set of operation performed in a stage on a partition
  - A thread is created for each task.



Spark Execution – Terminologies

- Driver: Process that creates SparkContext → created on client or worker node depending --deploy-mode
  - Create DAG
  - Submit DAG to the Master for execution.
  - Keep track of execution progress.
- Master: Cluster manager
  - It is a Java process. web ui = 8080
  - It manage and allocate resources.
  - Spark standalone mode: Master URL = spark://master:7077/ ← IPC port
- Worker: Each node in cluster
  - It is a Java process. web ui = 8081
  - It executes application in separate process -- executor.
- Executor
  - It is a Java process.
  - Created by the worker for execution of an application.
  - All tasks (threads) are created in executor.


movies.csv
map()
filter()
ratings.csv
map()
filter()
reduce By Key()
join()
sortBy()
map
foreach()

Spark Multi Node Cluster (Standalone)

- On Master machine, conf/~~slaves~~ workers make entries of all slaves.
- In all machines, conf/spark-defaults.conf
  - spark.master spark://master:7077
- On master machine, set SPARK_LOCAL_IP & SPARK_MASTER_HOST to be set (in conf/spark-env.sh) to the IP address of network.
  - export SPARK_LOCAL_IP=master
  - export SPARK_MASTER_HOST=master
- On each slave machine, set SPARK_LOCAL_IP (in conf/spark-env.sh) to the IP address of network.
  - export SPARK_LOCAL_IP=~~vmX~~ worker ?
- from master
  - terminal> start-master.sh
  - terminal> start~~-slaves.sh~~ -workers.sh
- Using spark cluster
  - terminal> spark-submit --master spark://master:7077 --deploy-mode client app.py

✓ one master & three worker nodes

✓ in /etc/hosts → make entries of master & workers

✓ send ssh-id of master to all workers.

✓ download & extract spark in home dir of same user (hduser).

✓ in .bashrc of all make $SPARK_HOME & $PATH entries.

✓ change hostname of all machines.

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>