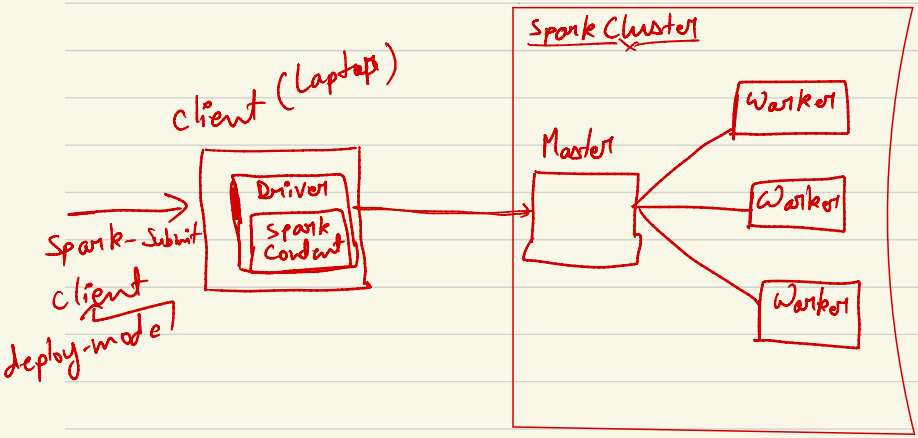
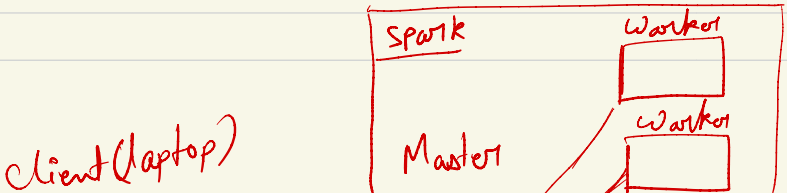
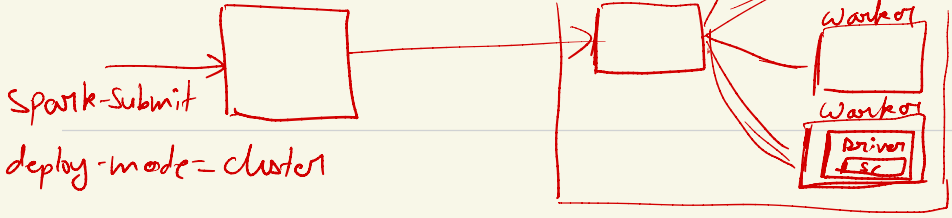
Client mode vs Cluster mode in Spark

Syntax for Spark Submit

```
> spark-submit \
  --master <master-url> \
  --deploy-mode <cluster or client> \
  --conf <Key:Value> \
  --driver-memory ng \
  --executor-memory ng \
  --executor-cores n \
  --num-executors n \
  --jars <comma separated jar file names> \
  wordCount.py [space separated command line argument]
```







What is RDD in Spark?

RDD - Resilient Distributed Dataset

↳ RDD are the main logical data units in Spark. They are distributed collection of objects, which are stored in memory or on disks of different machines on cluster.

Features of RDD

* Resilience: RDDs track data lineage information (lineage Graph) to recover from failed state automatically. It is also known as fault-tolerance.

* Distributed: Data present in an RDD resides on multiple nodes.

* Lazy Evaluation: Data does not get loaded in an RDD even if you define it.

* Immutability: Data stored in an RDD is in the read-only mode - you can't

edit the data which is present in RDD.

*) In-memory Computation:

An RDD stores any intermediate data that is generated in the RAM so that faster access can be provided.

Actions & Transformations in Spark

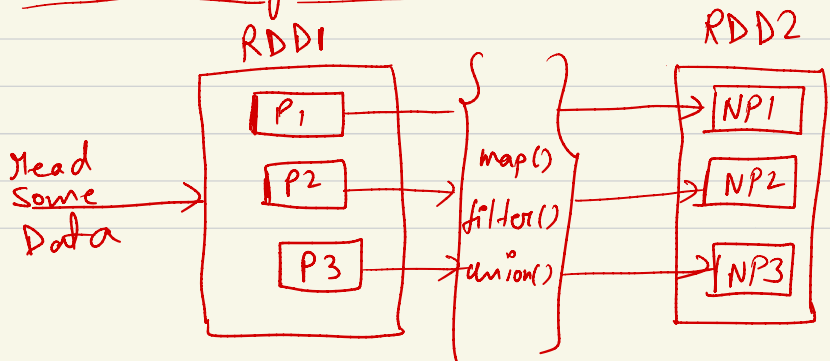
Transformations

It is a function that produces a new RDD from existing rdd. It takes RDD as input & generate new RDD as a output.

There are two types of transformations.

- Narrow Transformation (No Data Shuffling)
- Wide Transformation (Data Shuffling)

* Narrow Transformation

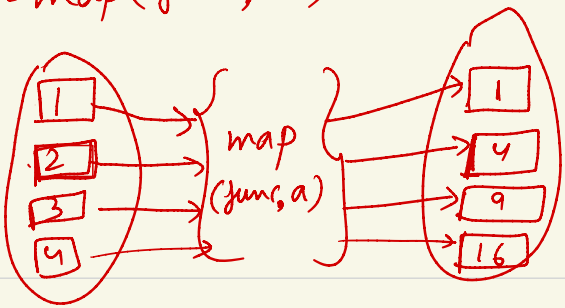


→ map()
→ FlatMap()
→ Filter()
→ Union()
→ Sample()

a = [1, 2, 3, 4]

func = lambda x : x**2

result = map(func, a)

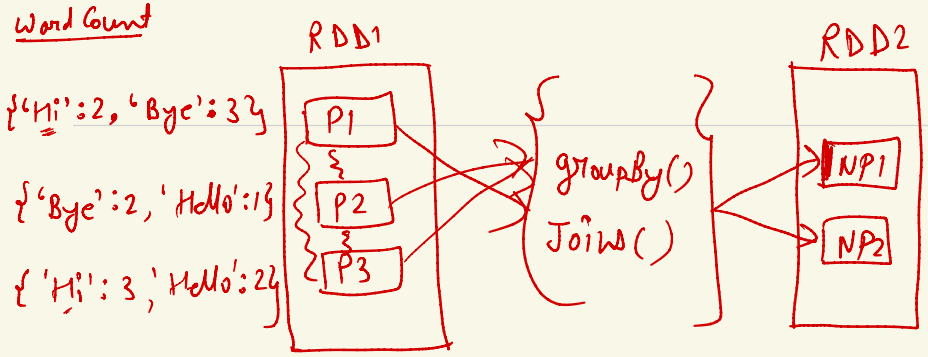


*) Wide Transformation

↳ All the elements that are required to compute the records in a single partition, may live in many partitions of the Parent RDD.

→ ReduceByKey
→ GroupByKey
→ Join
→ CartesianJoin
→ Repartition
→ Coalesce

Word Count



Actions in Spark

↳ Action is one way of sending data from executor to the driver.

- → count()
- → collect()
- → take(n)
- → top()
- → reduce()
- → aggregate()
- → foreach()

Demo PySpark application

```
spark = SparkSession.getOrCreate()

empRdd = spark.read_csv("employee.csv")
deptRdd = spark.read_csv("department.csv")
```

```
filterRdd = empRdd.filter("country='INDIA'")

indiaDeptSalaryRdd = filterRdd.groupBy("dept_id")
                      .agg(sum(), salary)

departLevelSalary = indiaDeptSalaryRdd.join(
                      dept_rdd,
                      on=[dept_id],
                      "inner_join"
                    )
```

→ Transformation

employee.csv
- emp_id
- emp_name
- Salary
- dept_id
- country

department.csv
- dept_id
- dept_name

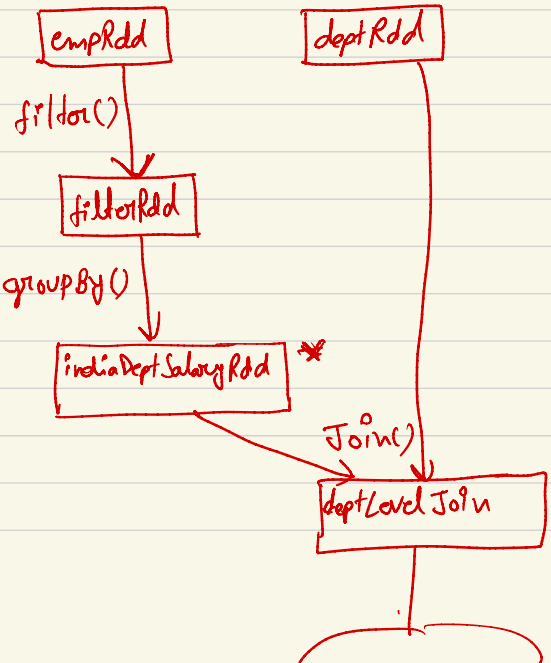
9.) write a application to get total salary in each department. for INDIA

departLevelSalary.top() → action

→ Lazy Execution

DAG (directed) (acyclic) (graph)

Lineage Graph →



Action

Job Execution in Spark