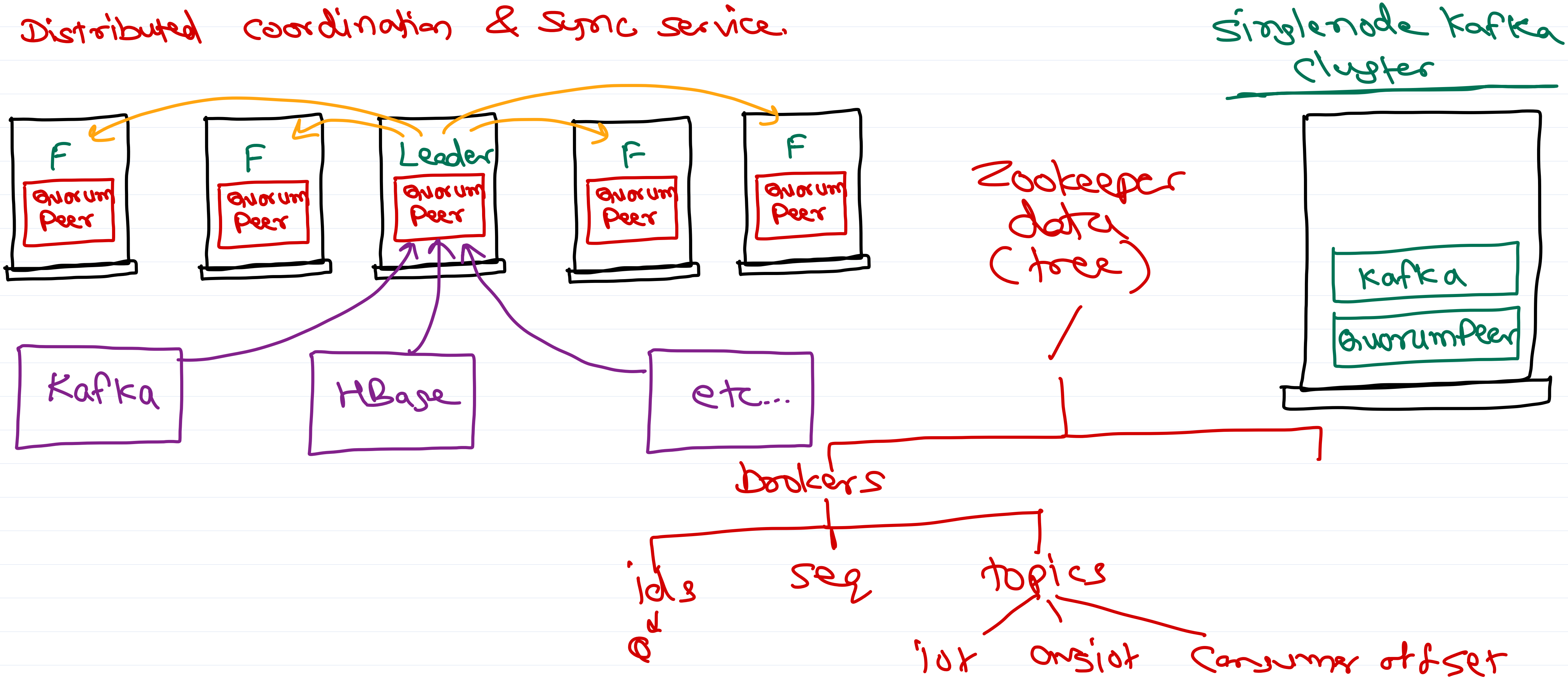
Big Data Technologies

Trainer: Mr. Nilesh Ghule.

Zookeeper

Distributed coordination & sync service



Spark Core — Low level apis

Spark Core

① RDD & DAG execution

② cluster (master/worker)

RDD: Resilient Distributed Dataset

Resilient ↓ can recover (Lineage)

Distributed ↓ split into partitions on multiple worker nodes

Dataset ↓ in-mem

Logical entity

Physical entity

RDD characteristics

① immutable

② resilient

③ lazy evaluated

RDD programming = Functional programming

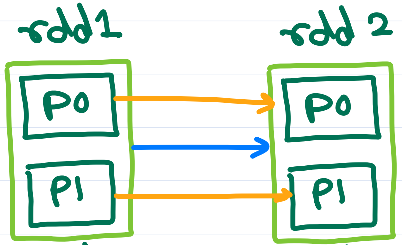
Basics of fn prog:

① immutability

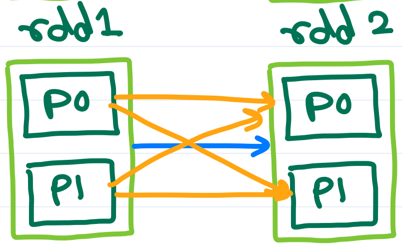
② location transparency

RDD operations

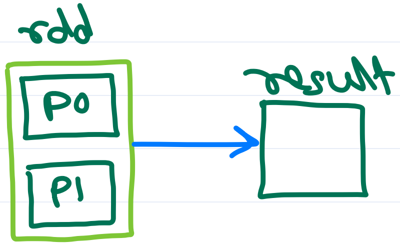
① narrow transformations:



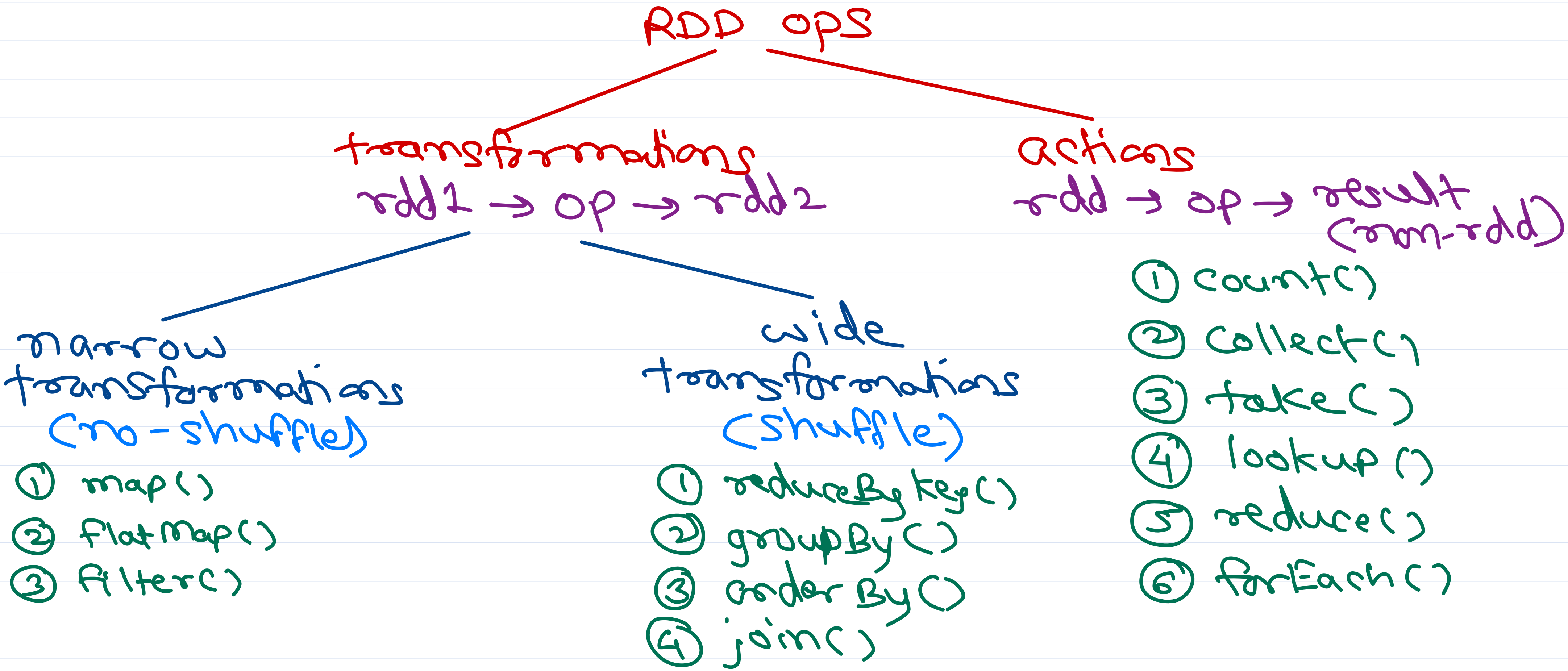
② wide transformations:



③ actions:



RDD operations



RDD ops

transformations

rdd1 → op → rdd2

actions

rdd → op → result (non-rdd)

narrow transformations (no-shuffle)

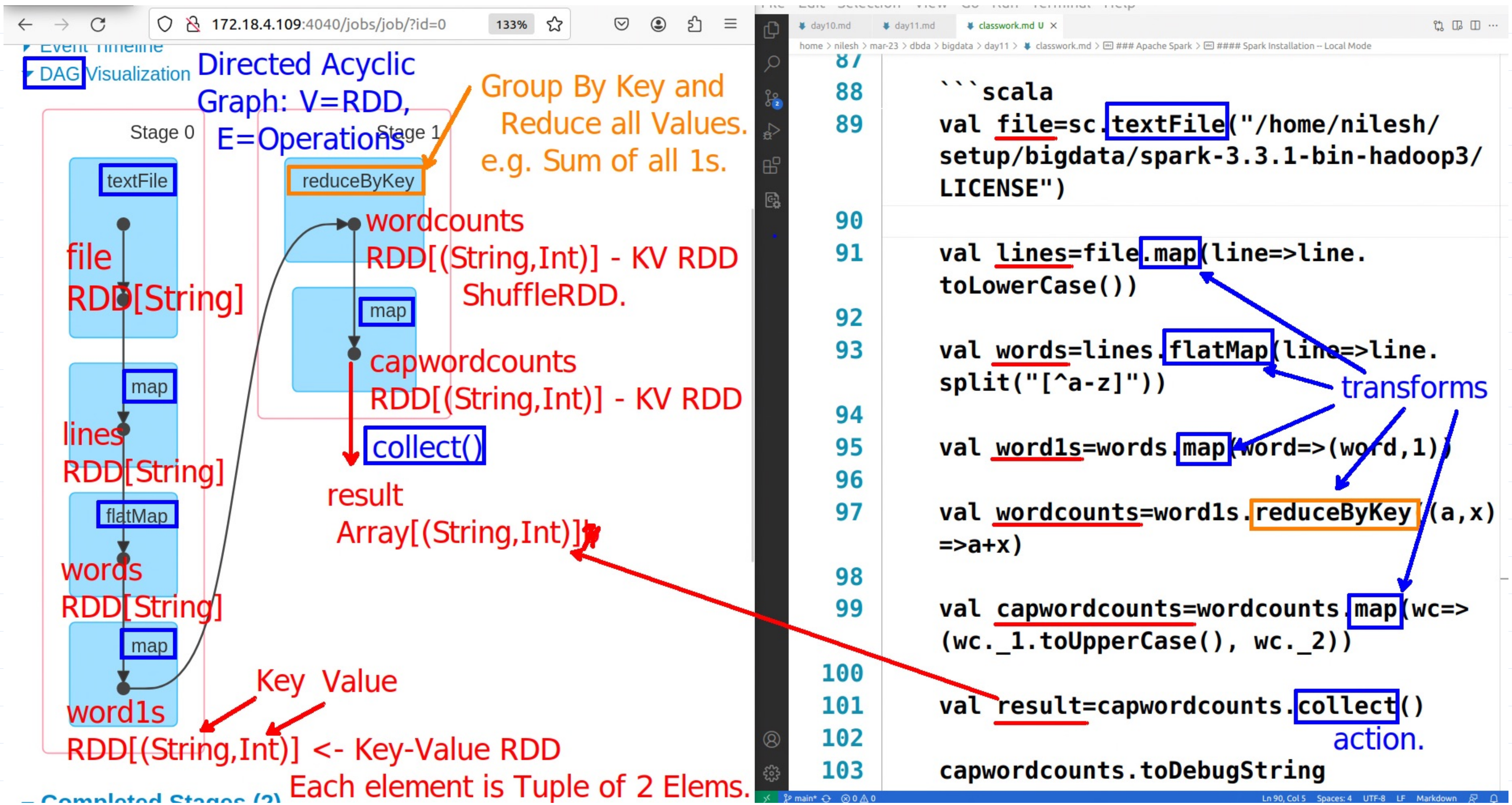
1. map()
2. flatMap()
3. filter()

wide transformations (shuffle)

1. reduceByKey()
2. groupBy()
3. orderBy()
4. join()

1. count()
2. collect()
3. take()
4. lookup()
5. reduce()
6. forEach()

Spark RDD - WordCount application in Scala



~~~
```scala
val file=sc.textFile("/home/nilesh/setup/bigdata/spark-3.3.1-bin-hadoop3/LICENSE")

val lines=file.map(line=>line.toLowerCase())

val words=lines.flatMap(line=>line.split("[^a-z]"))

val word1s=words.map(word=>(word,1))

val wordcounts=word1s.reduceByKey((a,x)=>a+x)

val capwordcounts=wordcounts.map(wc=>(wc._1.toUpperCase(), wc._2))

val result=capwordcounts.collect()

capwordcounts.toDebugString
~~~

Directed Acyclic Graph: V=RDD, E=Operations

Group By Key and Reduce all Values. e.g. Sum of all 1s.

file RDD[String]

lines RDD[String]

words RDD[String]

word1s RDD[(String,Int)] <- Key-Value RDD

Key Value

Each element is Tuple of 2 Elems.

wordcounts RDD[(String,Int)] - KV RDD ShuffleRDD.

capwordcounts RDD[(String,Int)] - KV RDD

collect()

result Array[(String,Int)]

transforms

action.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>