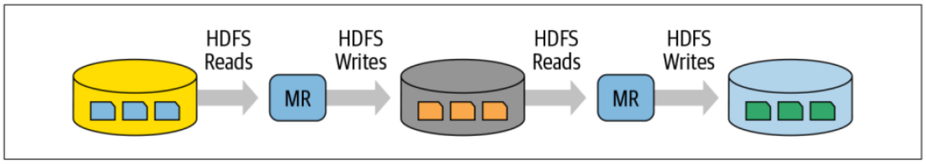
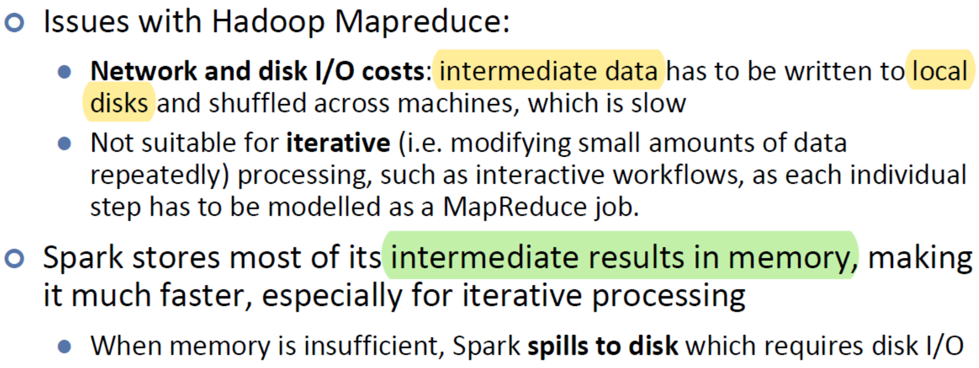
CS4225/CS5425 BIG DATA SYSTEMS FOR DATA SCIENCE

Tutorial 3: Spark

a) Why Spark is more suitable for iterative processing compared to Hadoop?

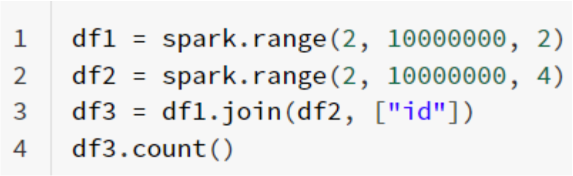
Answer:
Spark stores most of its intermediate results in memory, making it much faster, especially for iterative processing.





- Issues with Hadoop Mapreduce:
  - **Network and disk I/O costs**: intermediate data has to be written to local disks and shuffled across machines, which is slow
  - Not suitable for **iterative** (i.e. modifying small amounts of data repeatedly) processing, such as interactive workflows, as each individual step has to be modelled as a MapReduce job.
- Spark stores most of its intermediate results in memory, making it much faster, especially for iterative processing
  - When memory is insufficient, Spark **spills to disk** which requires disk I/O

b) In the below spark code block, please indicate which lines are transformation and which lines are action. For transformation, please also indicate whether it is a narrow transformation or wide transformation.

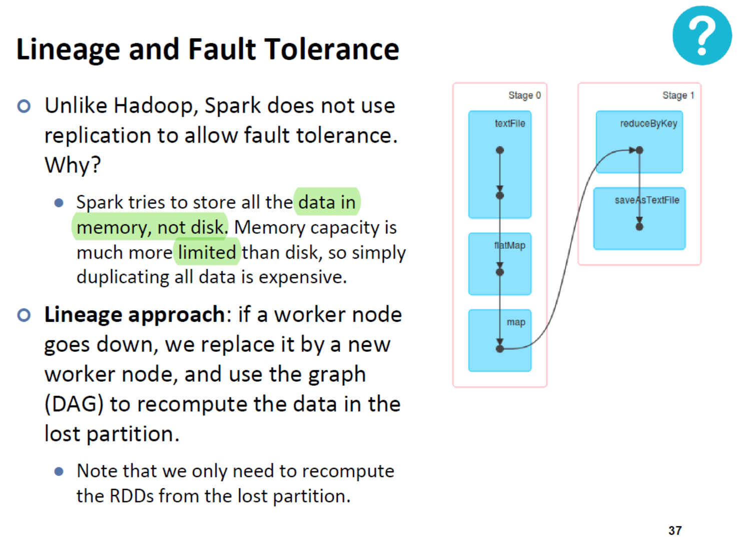


```
1  df1 = spark.range(2, 10000000, 2)
2  df2 = spark.range(2, 10000000, 4)
3  df3 = df1.join(df2, ["id"])
4  df3.count()
```

Answer:
Line 1: Narrow Transformation
Line 2: Narrow Transformation
Line 3: Wide Transformation
Line 4: Action

c) In HDFS, each chunk is replicated for three times by default. In contrast, in Spark, RDD uses lineage for reliability. What is a major problem if Spark also uses replications for reliability?

Answer:
Consumes a lot of memory; memory is much more scarce than disk space



Lineage and Fault Tolerance

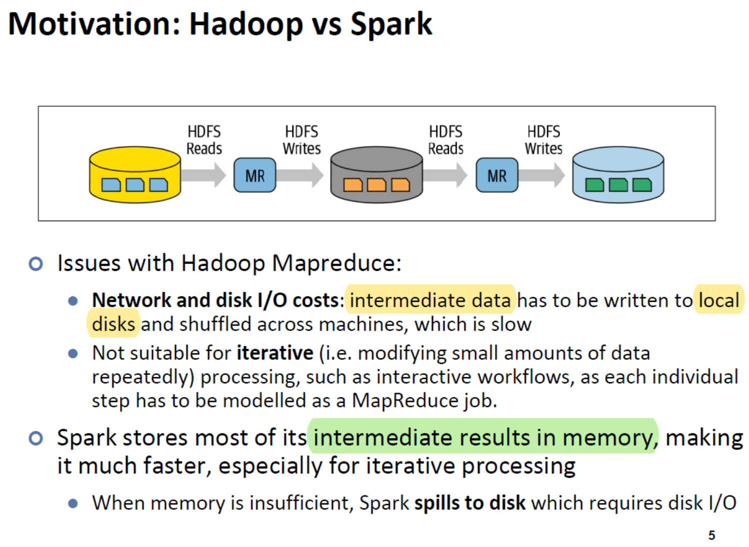
- Unlike Hadoop, Spark does not use replication to allow fault tolerance. Why?
  - Spark tries to store all the data in memory, not disk. Memory capacity is much more limited than disk, so simply duplicating all data is expensive.
- **Lineage approach**: if a worker node goes down, we replace it by a new worker node, and use the graph (DAG) to recompute the data in the lost partition.
  - Note that we only need to recompute the RDDs from the lost partition.

37

d) Is it true that in the Spark runtime, RDD cannot reside in the hard disk?

Answer:

False. RDD can also be in the disk if out of memory



Motivation: Hadoop vs Spark

- Issues with Hadoop Mapreduce:
  - **Network and disk I/O costs**: intermediate data has to be written to local disks and shuffled across machines, which is slow
  - Not suitable for **iterative** (i.e. modifying small amounts of data repeatedly) processing, such as interactive workflows, as each individual step has to be modelled as a MapReduce job.
- Spark stores most of its intermediate results in memory, making it much faster, especially for iterative processing
  - When memory is insufficient, Spark **spills to disk** which requires disk I/O

5

e) Explain how the following program can be sped up.

```
1   lines = spark.textFile("hdfs://log")
2   errors = lines.filter(lambda s: s.startswith("INFO"))
3   info = errors.map(lambda s: s.split("\t")[2])
4   info.filter(lambda s: "hadoop" in s).count()
5   info.filter(lambda s: "spark" in s).count()
```

Line 1: Reads a file from HDFS
Line 2: Filter to extract lines starting with "INFO"
Line 3: Split string by tab and extract 2nd component
Line 4: Count the number of lines with "hadoop"
Line 5: Count the number of lines with "spark"

Answer:

How to speed-up: we should add info.cache() (or info.persist()) before line 4, to cache the RDD in memory (or hard disk) so it doesn't have to be re-computed in line 5.