THE UNIVERSITY
of ADELAIDE

Assignment 1: Basics & Hadoop

Mining Big Data
(COMP SCI 7306)

Yogeshvar Senthilkumar
a1817369

Manivannan Meenakshi Sundaram
a1842066

March 26, 2022

Exercise 1 *Suspected Pairs*

From the given data, total population P size is 5 million

$$5 \cdot 10^9$$

From [3] we already know the expected number of events is obtained by multiplying number of pairs of people, number of four days quadruple. Also we know from [3], for large n

$$\binom{n}{2} \approx \frac{n^2}{2}$$

For the given data, we can simplify the number of pairs of two people in the given population is

$$\frac{5 \cdot 10^9}{2} = \frac{25 \cdot 10^{19}}{2} \implies 1.25 \cdot 10^{19} \tag{1}$$

Number of four days quadruple equalss

$$\binom{5000}{4} \approx \binom{n}{r} = \frac{n!}{(r!(n-r)!)} = \frac{5000!}{4! \cdot 4996!} = 26010428123750 = 2.6{\cdot}10^{13} \tag{2}$$

Probability of pair visit a hotel in same day [3]

$$0.0001$$

Chances two people visit same hotel on same day

$$\frac{0.0001}{500000} = \frac{1 \cdot 10^{-4}}{5 \cdot 10^5} = \frac{10^-9}{5} = 2 \cdot 10^{-10} \tag{3}$$

Probability of person visit hotel

$$\frac{1}{n}$$

Probability of both persons going to same hotel

$$\left(\frac{1}{n}\right)^2 \cdot n = \frac{1}{n}$$

Probability of both persons going to same hotel on four days

$$From\ 3,\ (2 \cdot 10^{-10})^4 = 1.6 \cdot 10^{-39} \tag{4}$$

To find out the how many "events" will indicate evil-doing, we will have to find

$$From\ 1, 2, 4, 1.25 \cdot 10^{19} \times 2.6 \cdot 10^{13} \times 1.6 \cdot 10^{-39} = 5.2 \cdot 10^{-7}$$

Exercise 2 *TF-IDF*

Q1. Term Frequency times In-verse Document is defined as the formal measure of how concentrated into relatively few documents are the occurrences of a given word. Assume we have a collection of N documents. We could say the *frequency* of a term (word) i in document j is f_{ij}. The *Term Frequency* TF_{ij}, the term frequency of term i in document f is f_{ij} is normalized by dividing it by the maximum number of occurrences of any term in the same document j. TF will be 1 for the most frequent term in document j, whereas all other terms gets fraction for this document.

$$TF_{ij} = \frac{f_{ij}}{max_k \cdot f_{kj}} \tag{1}$$

Suppose term i appears in n_i of the N documents in the collection. Then, define the IDF_i to be:

$$IDF_i = \log_2 \left(\frac{N}{n_i} \right) \tag{2}$$

The TF.IDF score for term i in document j is defined as

$$From\ 1\ \&\ 2\ , TF.IDF = TF_{ij} \cdot IDF_i$$

Q2. From given data, a repository `R` has `D` documents `i.e`

$$D = 10 \cdot 10^6$$

a. IDF for a word `w` appears in 40 documents

$$IDF_i = \log_2 \left(\frac{N}{n_i} \right) \implies IDF_w = \log_2 \left(\frac{10 \cdot 10^6}{4 \cdot 10^1} \right)$$
$$= \log_2(25 \cdot 10^4) = 17.931 \approx 18$$

b. IDF for a word `w` appears in 10000 documents

$$IDF_i = \log_2 \left(\frac{N}{n_i} \right) \implies IDF_w = \log_2 \left(\frac{10 \cdot 10^6}{10 \cdot 10^3} \right)$$
$$= \log_2(1000) = 9.965 \approx 10$$

Q3. From given data, a repository `R` has `D` documents `i.e`

$$D = 10 \cdot 10^6$$

and word `w` appears in 320 in them.

$$IDF_w = \log_2 \left(\frac{10^7}{32 \cdot 10^1} \right) = 14.194 \tag{3}$$

a. TF.IDF for the `w` to appears once will be calculated as,

$$From\ 3,\ 14.194 \cdot \frac{1}{15} = 0.9 \approx 1$$

b. TF.IDF for the `w` to appears 5 times will be calculated as,

$$From\ 3,\ 14.194 \cdot \frac{5}{15} = 4.97 \approx 5$$

Exercise 3 *Hadoop Basics*

a. Job on the attached file "100-0.txt" in two modes.

For spinning up the Hadoop environment, We have used both Docker as Mac book with M1 Chip does not support Virtual Machine Software's and Cloudera Virtual Machine in Windows. The screenshots, steps for running the application is written commonly which will be working for both the OS.

1. Standalone Mode
   (a) Necessary input files, source code files for the application is downloaded from the Myuni website and jar file is built.
   (b) Copy the Files to the container in which Hadoop is running.
   (c) For standalone mode, the input and output will be used from the local file system. With the every necessary files in the machine is copied.
   Use `hadoop jar jars/WordCount.jar input/100-0.txt output`
   (d) Job will be configured and once the job is done. The output files will be obtained in the `output/` directory.
   (e) The output file will have the word count given in the input file "100-0.txt"

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart WordCount]$ ls
100.txt  bin  conf  hadoop.log  hadoop.log.2022-03-18  outstand  src  Wordcountjar.jar
[cloudera@quickstart WordCount]$ cd outstand/
[cloudera@quickstart outstand]$ ls
part-r-00000  _SUCCESS
[cloudera@quickstart outstand]$ head -n 10 part-r-00000
        82964
"       49
"Air,"  1
"Defects,"      1
"Information    1
"Neither        1
"Plain  2
"Project        5
"Right  1
"so     1
[cloudera@quickstart outstand]$
```

2. Pseudo-distributed Mode

   (a) The jar file is obtained from the earlier step when built for the standalone mode.

   (b) Copy the Files to the `namenode` container where will be running the Hadoop.

   (c) For Pseudo-distributed Mode or Single Cluster Mode, Hadoop Daemons will be running on the same machine to enable the distributed real-world operation of Hadoop.

   (d) Create & Move the input files to the HDFS (Hadoop Distributed File System). Verify if the correct files are moved to HDFS.

   (e) Check if every Master/Slave nodes heartbeat before running the application

   (f) Use `hadoop jar jars/WordCount.jar input/100-0.txt output` Job will be configured and once the job is done. The output files will be obtained in the `output/` directory in HDFS.

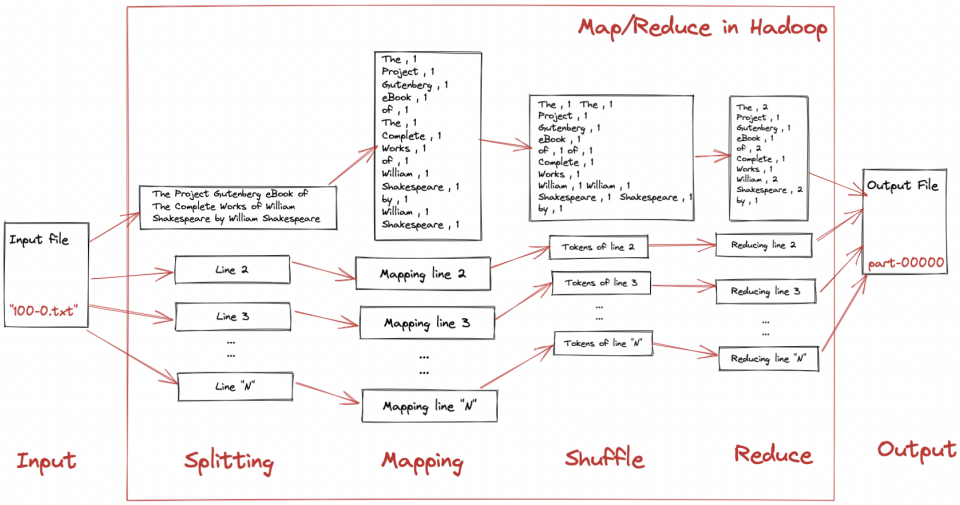
   (g) Get the output directory from HDFS to the local file system using the command `hdfs dfs -get outputDirectory/`

   (h) output file will have the word count given in the input file "100-0.txt"

```
[cloudera@quickstart WordCount]$ hdfs dfs -ls .
Found 1 items
drwxr-xr-x   - cloudera cloudera          0 2022-03-21 17:24 input
[cloudera@quickstart WordCount]$ hadoop jar Wordcountjar.jar input/ outcluster &>/dev/null
[cloudera@quickstart WordCount]$ hdfs dfs -get outcluster
[cloudera@quickstart WordCount]$ head -n 10 out
outcluster/ outstand/
[cloudera@quickstart WordCount]$ head -n 10 out
outcluster/ outstand/
[cloudera@quickstart WordCount]$ head -n 10 outcluster/part-r-00000
        82964
"       49
"Air,"  1
"Defects,"      1
"Information    1
"Neither        1
"Plain  2
"Project        5
"Right  1
"so     1
[cloudera@quickstart WordCount]$ SS
```

b. Operational differences of Modes

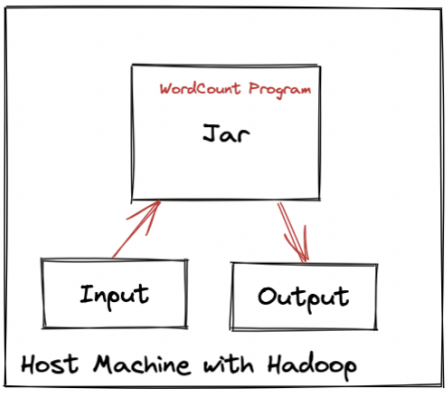
The given code is a simple application that returns the total number of occurrences of a word in the given input file. The following dataflow diagram shows what `WordCount.java` program is trying to achieve.



1. Standalone Mode

   (a) Default & Fastest Mode of Operations provided by Hadoop is the Standalone mode where the local file system can be used for both input and output.

   (b) The file are not moved to HDFS which could easily lead to the single point of failure design flaw if the data loss happens.



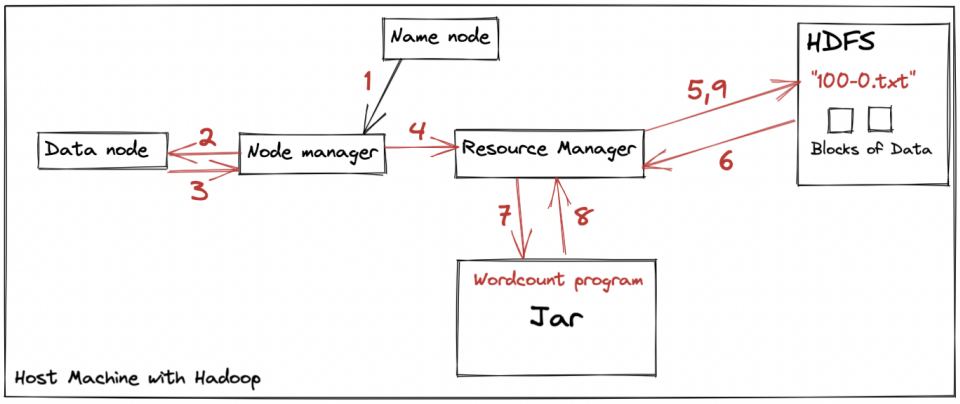
   (c) Standalone mode is a single process in the system and it is not distributed. The hadoop operation runs on the JVM instance without any daemons which also means it could be used for the quality control, debugging of the logical implementation of the application, yet no real clustering simulations.

2. Pseudo-Distributed Mode

(a) Also known as Single Cluster Mode, where the Master & Slave Nodes work on the same machine and uses HDFS to prevent the single point of failure as HDFS replicate the data in Slave nodes which makes it fault-tolerant.

(b) Every Hadoop Component is spawned as a separate JVM instance so the communication happens over the network sockets, giving the simulations of fully-functioning disturbed system on a single machine.

(c) One of the component responsible for processing the MapReduce Jobs and manages cluster resources is `YARN` ; Yet Another Resource Negotiator composed of NodeManager, ResourceManager, DataNode, NameNode.



(d) Namenode, the master node will notify the node manager for running a job. The Node manager will check for the slave nodes for processing the jobs. Once the resources are available, the resource manager runs the application with HDFS. The jobs and the corresponding logs can be found on the slave nodes.

(e) The infrastructure runs on a single machine, thus it is called as Single Cluster mode.

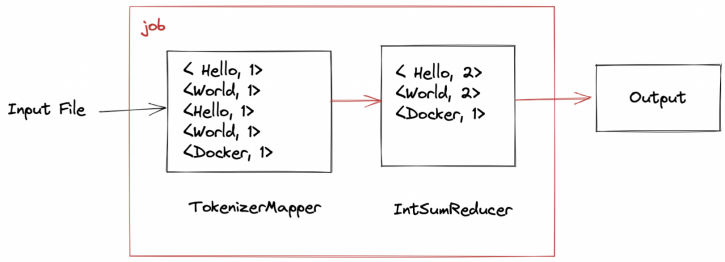
Difference in the outputs Only the working operation of the modes differ, the output files are identical. There's no difference in the output data.

```
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart WordCount]$ diff -s outstand/part-r-00000 outcluster/part-r
-00000
Files outstand/part-r-00000 and outcluster/part-r-00000 are identical
[cloudera@quickstart WordCount]$ S
```

Exercise 4 *Map Reduce in Hadoop*

Part 1:

Program to count the number of words with the specific number of letters uses a Mapper to convert the input to key-value pairs of `<length, 1>` and Reducer will reduce the key-value pairs to `<SameLength, n>`. The input string is sanitized against delimiter to get the proper words. The full documentation can be found at https://beingmani.github.io/MBD-Assignment The data flow of the written program is given below



Part 2:

Q1. *How many words are there with length 10 in FirstInputFile?*
10649

Q2. *How many words are there with length 4 in FirstInputFile?*
210602

Q3. *What is the longest length between words and what is its frequency in FirstInputFile?*
Longest length is **27** with frequency **1**

Q4. *How many words are there with length 2 in SecondInputFile?*
63728

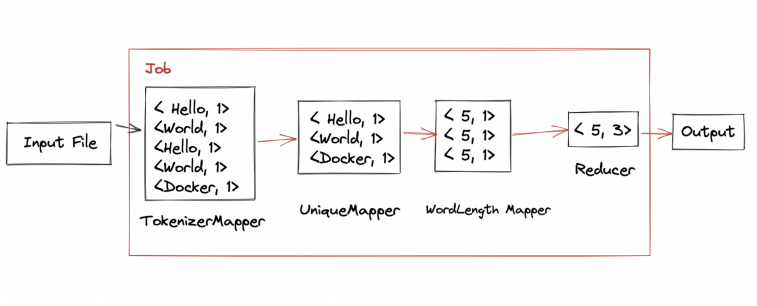
Q5. *How many words are there with length 5 in SecondInputFile?*
35757

Q6. *What is the most frequent length and what is its frequency in SecondInputFile?*
Most frequent length is **3** with frequency **73168**

Part 3:

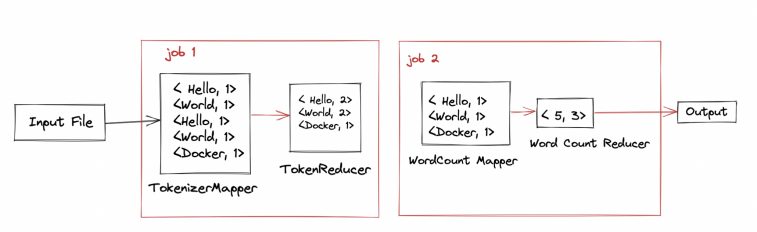
Program to count the number of words (unique) with the specific number of letters, we have decided to submit two solutions as both have their significant impact on the computation.

a. Using `ChainMapper`, the data flow of the program with the ChainMapper will look like



Although, the runtime is slower than the other solution. The utilisation of the resources is only limited to a Single Job.

b. Using `Two MapReduce Jobs`, the program runtime is faster than the `ChainMapper` yet the resources are utilised more when compared to the other program workflow.



Part 4:

Q1. *How many words are there with length 10 in FirstInputFile?*
1700

Q2. *How many words are there with length 4 in FirstInputFile?*
2056

Q3. *What is the most frequent length and what is its frequency in FirstIn-putFile?*
Most frequent length is **7** with frequency **4284**

Q4. *How many words are there with length 5 in SecondInputFile?*
1824

Q5. *How many words are there with length 2 in SecondInputFile?*
99

Q6. *What is the second-most frequent length and what is its frequency in SecondInputFile?*
Second Most frequent length is **8** with frequency **2628**

Exercise 5 *Summary of 2.4 and 2.5*

2.4 Extensions to MapReduce

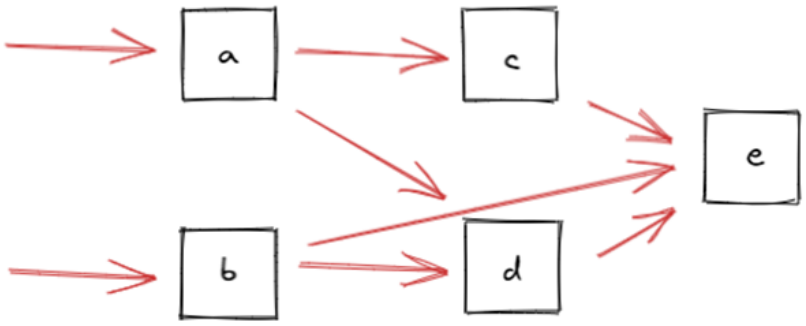
Map-Reduce is considered to be one of the most influential part as it paved way for many extensional modifications as they incorporate failures during the execution of jobs with huge chunks of data .Due to this the jobs have to be restarted from the beginning.This gave rise to a section called workflow which is actually an extension of Map-Reduce which supports acyclic network of functions.

Some of the extended systems are,

- UC Berkeley's Spark
- TensorFLow
- Google's Pregel

WorkFlow Systems
The Extension of the map reduce is just a simple two step process where Map functions feeds in to a reduce function. For example, considering any acyclic work flow a →b, The function a's output is fed in to function b as an input. Workflow system is one of the early working version coping

up with failures. It is widely classified in two ways, (1)having a single independent function where it consists of Map and reduce functions which are applied to each individual elements .(2) Having multiple functions where it consists of multiple inputs to a function, where each files are processed in various ways and fed as an input to other functions.



Spark

Spark is the advanced version of work flow system where it has proved its ability over the early workflow system. Some of them are

- Proved its efficiency by coping up with failures than early relelase.
- Proved its efficiency in Grouping and Scheduling different computer nodes
- Introduction of acyclic workflow class with a concept of adding loops from programming languages.

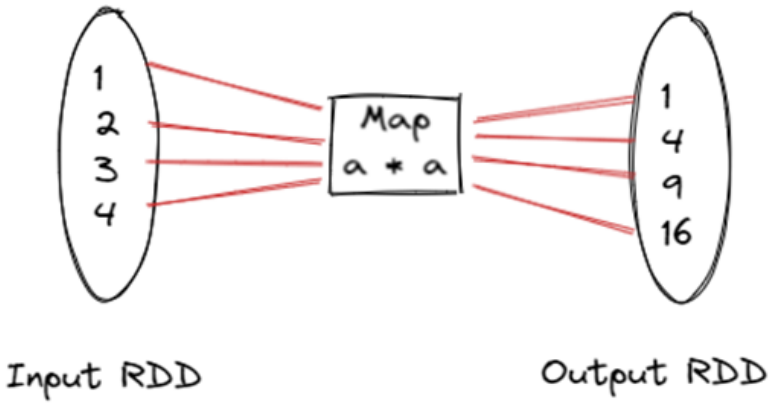
Resilient Distributed data set or RDD is an advanced version on key-value pairs that are being used in map-reduce, But in Distributed system they are broken in chunks of data that are held in different computer nodes. With this approach they are able to recover the data from any loss or failure form any chunks of data, Hence the word resilient. RDD has two operations namely Transformations and Actions.

Transformations are functions where you use an input RDD to get an resultant RDD, Example : Filter(), Union().

Actions are functions which return a result of a program which is delivered, example : Count(), First()

There are sequence of steps happening in Spark namely Transformations, actions and Spark program. In this section we will see only few Basic RDD operations which are commonly used.

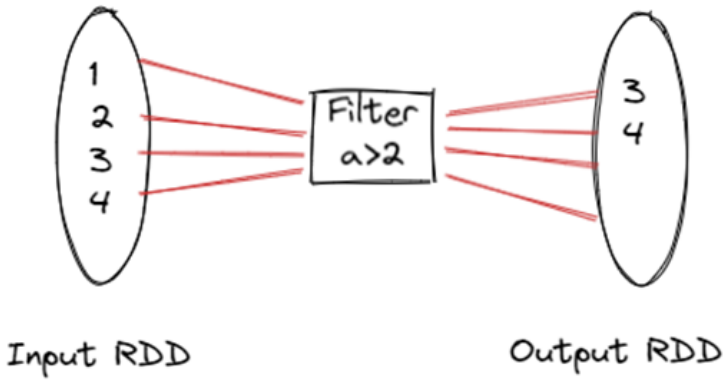
- **Map**
  Map is Simple transformation that takes applies a function to each individual input RDD. On the other hand, the resultant RDD will have a individual value for each element respectively. Input and the output RDD has the same length in map transformation. In



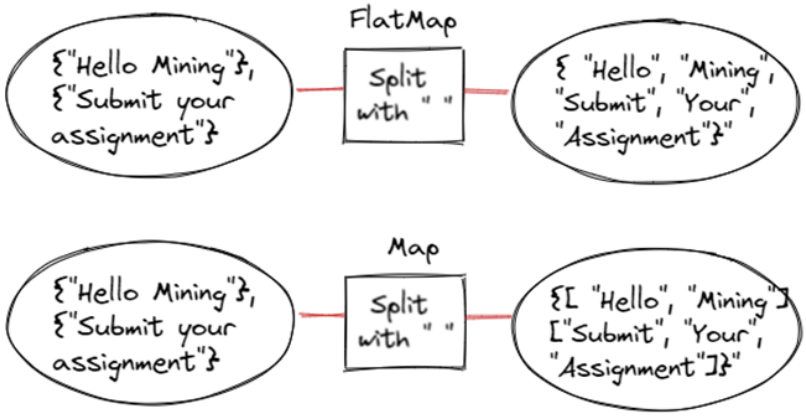
  the above stated example, Input RDD passes through a Mapper function where it mulitiples with its equivalent number and delivers an resultant RDD. Its basically an square function for an input a and delivers an squared output for an input a.

- **Filter**
  Filter is another simple transformations that returns new RDD only when the elements passes the function with a filter Condition. In this example, We have a filter function with an condition where all the inputs with value greater than 2 has to be filtered and moved to resultant RDD.

- **FlatMap**
  It's a Transformation which is similar to Map function , But the only difference is Flat map returns multiple values for each element in the input RDD where as in Map it doesn't return multiple values. For example : If the input RDD has "Hello Mining" and "Submit your assignment" and undergoes a Flatmap transformation with a split function of " ", the resultant RDD has multiple



values as “Hello”,”Mining”,”Submit”,”Your”,”Assignment”. If the input goes along with a transformation of Map, it still returns only two resultant values as the input RDD has 2 elements. This is because the input and the output of the map transformations are always the same.

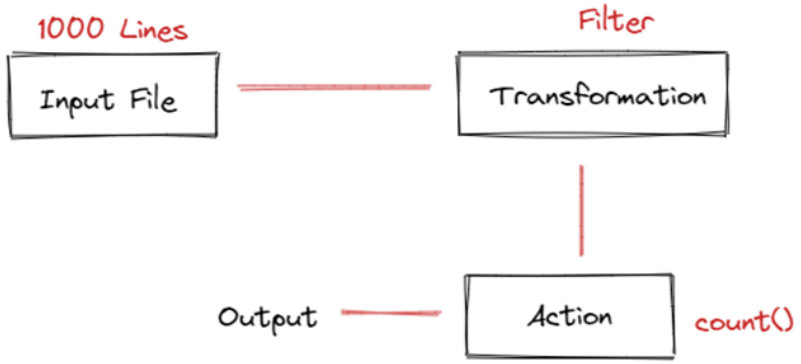


- **Reduce**
  Reduce is considered as an action and not a transformation. Reduce function actually returns an value and not an resultant RDD as we have seen in the above scenarios. Consider the below example where you have a sample numbers as a input RDD and it undergoes an reduction function and it returns an value based on the simple math equation.

WorkFlow of Spark

The implementation of spark differs from other Hadoop or Map-Reduce functions . As said earlier Spark Implementation is considered to be most efficient way of working with Big data. This is because of the two improvements that has been made in spark.

- **Lazy evaluation**
  This is one of the finest technique which is used in used in spart in order to increase the efficiency. Consider, there is a file with 1000 lines which has to undergo a filter transformation and undergoes another action to count the total values to give the resultant output. In general, the workflow happens in such a way that it reads all the 1000 lines first and then undergoes the transformation and action. Reading all the lines from the input takes a lot of time and storage to process the transformations and actions. In spark, Work flow reads all the transformations and actions in the system and works accordingly on the required output. Working on this procedure saves a lot of time and reduces storage spaces.

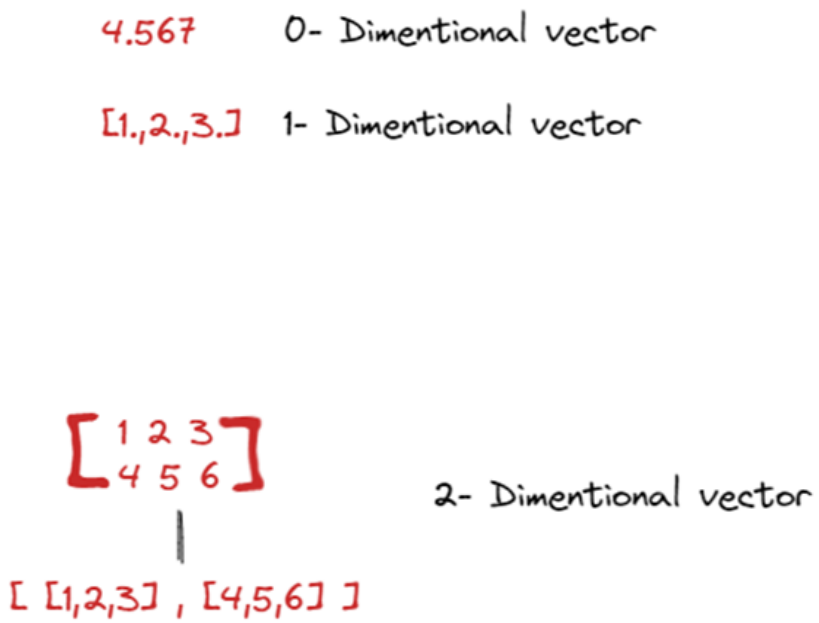


- **Lineage :**
  Lineage in spark is considered as a savior source as it has a redundant source of values which can be accessed in case of any failures.It is a collection of datasets which are places across the nodes and can be recovered in case of any failures or any issues with the chunks of data.

Tensor Flow: As Spark, Tensor flow is a similar project developed by Google which is open source supported for many machine learning

applications. Tensor Flow has the same advantage as spark and they support programming languages with loops. One of the major difference between spark and tensor flow is instead of RDD in spark, Tensor flow uses tensors which are nothing but Matrix multiplication.

Tensor flow tensors are represented as



Recursive Extensions to map reduce: Page rank is one of the Main example for recursive extension of map reduce. The iteration of Map- reduce goes on continuously for a sequence of results until they believe the convergence is happening. In this recursion, there is a serious problem of failure recovery. It iterates for a number of times with a lack of visibility for a certain operation**********

Some of the approaches that are diagnosed for dealing failures in recursion are:

- **Iterated map reduce :** Approaching them with a series of recursion repepated execution of various Map reduce jobs . With this approach we can handle failures at any step.
- **The Spark approach :** This approach basically works along with programming block concept such as for-loops which implements the recursions. With lineage and lazy evaluation the spark approach works efficiently than other implementations

- **Bulk Synchronous system:** This is a graph based system which process massive amount of bulk synchronous data . All the inputs and data are represented via graph. Graph consist of nodes and each nodes are represented as tasks. Output messages are generated based on the nodes to their destined nodes on the graph. Similar to map reduce each nodes receive inputs and outputs based on the graph they allocated.

Disadvantage in Pregel based Bulk Synchronous system :

At any point is any of the nodes get failure, there is no way back to restart the failed node. Pregel has checkpoints at different stages of nodes, if there is any failure discovered, all the nodes from the checkpoint has to be restarted or to be computed to get the resultant output.

2.5 Communication Cost Model

From [1], Assume we have a hash join of two 64MB chunks from two relations. If we wanted to take the chunks for the Mapping task in a data node. The time taken by the node to read the chunks from the disk will be more than reading it from the main memory for processing. Therefore, the cost of a task can be measured by the amount of data in its input. This size of the input is usually measured in bytes. For an algorithm, the cost will be measured by the sum of the costs of all the tasks involved during the algorithm. We refer this cost as "Communication cost" since the cost represents the time it takes to prepare the input for computation. Usually the mapping tasks will be running on the same location where the input data resides, so the communication cost for the transporting the input data to computation can be nullified because the cost will be small fractions. Often, communication costs dominates the cost of the algorithm [2].

Points to justify communication cost measures the efficiency of an algorithm, not the running time of the algorithm.

- The algorithm execution seems to be very straightforward and simple, it is linear in the size of the input.
- The processing of an input element will be minimal when compared to the time it takes for that element to be delivered. Due to the interconnect speed and high traffic of node communications at the same time.
- The task will be taken care by different compute nodes, so the chunks of data will reside on disk. Moving the data from disk to main memory will be more time consuming than the actual execution of the task on the data when it is available.

- We could argue we count only the input size of the task for the communication cost not the output size. Two reasons for this argument,

    a. In general, the nature of algorithm is to aggregate or reduce the massive data to much simpler data for understanding. It is rare for an algorithm to produce an output that is larger than the input itself.

    b. If a task τ output is used as an input for an other task, then there no use to measure the τ's size unless it has an impact over the entire algorithm.

Communication cost of a model influence over the algorithmic choice to use, importance of the ***Wall Clock time*** must be considered as well. The time taken by the parallel algorithm to finish, there will be trade-off between communication cost and computation time. Here are some possible scenarios which could lead to different wall-clock time,

- For example, one can minimize the communication cost by assigning all the task to a reducer. Thus, the total communication cost is minimized. But the trade-off here is that, if the reducer has so much data to operate it has to swap memory with main and secondary memory which leads to longer computational time. Also, the wall-clock time such algorithm would be high.

- If the task is divided equally among the nodes available, the wall-clock time would be minimal. Trade-off with this scenario is that, when we use more reducers we will increase the communication cost as more reducers are used.

Multiway Joins

For simple version of Multiway joins is that, the task of a mapper needs to be replicated multiple times to write the final output in the distributed file. Instead of repetitive operation, the intermediate results will be joined with the partition for writing the output in the distributed file. We will look at the general theory and an example to understand the working of a multiway joins.

- Partition certain attributes of the relations involved in the natural joins of three or more relations of their hashed values, each to some number of buckets. The number of buckets for each attributes is subjected to constraint that product of number of buckets for each attributes is `k`

- The `k` reducers will have the relation job of the intermediate results as a vector of bucket numbers
- We have to join all the relations by passing the tuples of each relation to all `k` reducers.

For example, we will take a look at the exercise 2.5.2 [3]. Consider the cyclic join

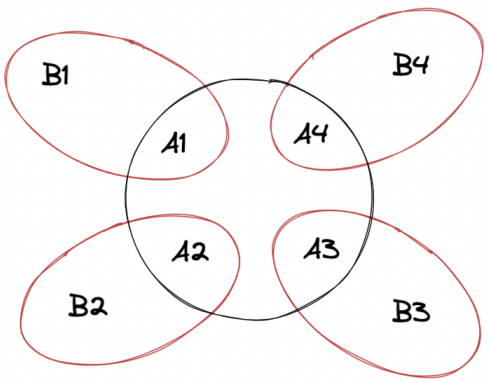
$$R(A, B) \bowtie S(B, C) \bowtie T(C, D)$$

The relations R, S, and T have sizes r, s and t respectively. Also given that, hash values of the attributes A, B, and C to a, b and c buckets. Suppose that we plan to use k reducers for this MapReduce task, where $abc = k$. Assuming we have hash functions h, g, f that sends the particular value of attributes to the respective buckets. Each reducer corresponds to A, B, C. As a cyclic join, an R-tuple and S-tuple natural join on B, whereas the an S-tuple and T-tuple natural join on C. As the cyclic dependency of Map tasks send to the reducers, we can conclude

$$Communication\ Cost = rc + sa + tb$$

Star Joins

One of the structure for data mining of Commercial data, where the join combines fact table $F(A_1, A_2..A_n)$ with the dimension tables $D_1(A_1, B_{11}, B_{12}, ...B_{1i}), ...D_n(A_n, B_{n1}, B_{n2}, ..B_{ni})$. Facts are the measured or recorded metric. Dimensions are the information about entity which helped to build the facts. Analysts answer the analytic queries using the star joins between the fact table and the dimension tables.



The communication cost of a star join can be minimised with multiway joining of relation table with the fact table will be more efficient.

References

[1] Foto N. Afrati et al. “Cluster Computing, Recursion and Datalog”. In: *Proceedings of the First International Conference on Datalog Reloaded*. Datalog'10. Oxford, UK: Springer-Verlag, 2010, pp. 120–144. ISBN: 9783642242052. DOI: 10.1007/978-3-642-24206-9_8. URL: https://doi.org/10.1007/978-3-642-24206-9_8.

[2] Anish Das Sarma et al. “Upper and Lower Bounds on the Cost of a Map-Reduce Computation”. In: *Proc. VLDB Endow.* 6.4 (2013), pp. 277–288. ISSN: 2150-8097. DOI: 10.14778/2535570.2488334. URL: https://doi.org/10.14778/2535570.2488334.

[3] Jure Leskovec, Anand Rajaraman, and Jeffrey D Ullman. *Mining of massive datasets*. Cambridge Cambridge University Press, 2020, pp. 6–7.