

## BDA Experiment no.: 8

**Aim:** Experiment on Hadoop Map Reduce: Implement simple Algorithm Matrix Multiplication.

### **Theory:**

MapReduce is a technique in which a huge program is subdivided into small tasks and run parallelly to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts:

- **Mapper:** It takes raw data input and organizes it into key, value pairs. For example, In a dictionary, you search for the word “Data” and its associated meaning is “facts and statistics collected together for reference or analysis”. Here the Key is Data and the Value associated with is facts and statistics collected together for reference or analysis.
- **Reducer:** It is responsible for processing data in parallel and producing final output.

Let us consider the matrix multiplication example to visualize MapReduce. Consider the following matrix:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Here matrix A is a 2×2 matrix which means the number of rows(i)=2 and the number of columns(j)=2. Matrix B is also a 2×2 matrix where number of rows(j)=2 and number of columns(k)=2. Each cell of the matrix is labeled as A<sub>ij</sub> and B<sub>jk</sub>. Ex. element 3 in matrix A is called A<sub>21</sub> i.e. 2nd-row 1st column. Now One step matrix multiplication has 1 mapper and 1 reducer. The Formula is:

Mapper for Matrix A (k, v)=((i, k), (A, j, A<sub>ij</sub>)) for all k Mapper

for Matrix B (k, v)=((i, k), (B, j, B<sub>jk</sub>)) for all i Therefore

computing the mapper for Matrix A:

# k, i, j computes the number of times it occurs.

# Here all are 2, therefore when k=1, i can have

# 2 values 1 & 2, each case can have 2 further

# values of j=1 and j=2. Substituting all values

# in formula

k=1 i=1 j=1 ((1, 1), (A, 1, 1))

j=2 ((1, 1), (A, 2, 2)) i=2

j=1 ((2, 1), (A, 1, 3))

j=2 ((2, 1), (A, 2, 4))

k=2 i=1 j=1 ((1, 2), (A, 1, 1))

j=2 ((1, 2), (A, 2, 2)) i=2

j=1 ((2, 2), (A, 1, 3))

j=2 ((2, 2), (A, 2, 4))

Computing the mapper for Matrix B

i=1 j=1 k=1 ((1, 1), (B, 1, 5))

k=2 ((1, 2), (B, 1, 6)) j=2

k=1 ((1, 1), (B, 2, 7)) j=2

((1, 2), (B, 2, 8))

i=2 j=1 k=1 ((2, 1), (B, 1, 5))

k=2 ((2, 2), (B, 1, 6)) j=2

k=1 ((2, 1), (B, 2, 7)) k=2

((2, 2), (B, 2, 8)) The formula for

Reducer is:

Reducer(k, v)=(i, k)=>Make sorted Alist and Blist

(i, k) => Summation (Aij \* Bjk) for j Output

=>((i, k), sum)

Therefore computing the reducer:

# We can observe from Mapper computation

# that 4 pairs are common (1, 1), (1, 2),

# (2, 1) and (2, 2)

# Make a list separate for Matrix A & #

B with adjoining values taken from

# Mapper step above:

(1, 1) => Alist = {(A, 1, 1), (A, 2, 2)}

Blist = {(B, 1, 5), (B, 2, 7)}

Now Aij x Bjk: [(1\*5) + (2\*7)] = 19 -----(i)

(1, 2) => Alist = {(A, 1, 1), (A, 2, 2)}

Blist = {(B, 1, 6), (B, 2, 8)}

Now Aij x Bjk: [(1\*6) + (2\*8)] = 22 -----(ii)

[(2, 1) => Alist = {(A, 1, 3), (A, 2, 4)}

Blist = {(B, 1, 5), (B, 2, 7)}

Now Aij x Bjk: [(3\*5) + (4\*7)] = 43 -----(iii)

(2, 2) => Alist = {(A, 1, 3), (A, 2, 4)}

Blist = {(B, 1, 6), (B, 2, 8)}

Now Aij x Bjk: [(3\*6) + (4\*8)] = 50 -----(iv) From (i), (ii), (iii) and (iv) we conclude that

((1, 1), 19)

((1, 2), 22)

((2, 1), 43)

((2, 2), 50)

Therefore the Final Matrix is:

19	22
43	50

**Program:** Mapper.py:

```
import sys for line
```

```
in sys.stdin:
```

```
    line = line.strip()
```

```
entry = line.split(",") key =
```

```
entry[0]          value= line
```

```
if key=='a':
```

```
    print('{0}\t{1}'.format(key,value))
```

```
elif key=='b':
```

```
print('{0}\t{1}'.format(key,value))
```

**Reducer.py** import sys

```
a={} b={} for input_line
```

```
in sys.stdin:
```

```
    input_line = input_line.strip()
```

```
    this_key,value = input_line.split("\t",1)
```

```
v = value.split(",")    if this_key=='a':
```

```
    a[(int(v[1]),int(v[2]))]=int(v[3])
```

```
elif this_key=='b':
```

```
    b[(int(v[1]),int(v[2]))]=int(v[3])
```

```
#and fill the blanks
```

```
#for i in range(0,5):
```

```
#    for j in range(0,5):
```

```
#        if (i,j) not in a.keys():
```

```
#            a[(i,j)]=0
```

```
#        if (j,i) not in b.keys():
```

```
#            b[(j,i)]=0
```

```
result=0
```

```
#compute the multiplication A*Bij = SUM(Aik * Bkj) for k in 0..4
```

```
for i in range(0,4):    for j in range(0,5):        for k in
```

```
range(0,3):            result = result + a[(i,k)]*b[(k,j)]
```

```
    print("({0},{1})\t{2}".format(i,j,result))
```

```
    result =0
```

**Output:**

```
cloudera@quickstart:~/MatrixMultiplication
File Edit View Search Terminal Help
(4,1) 0
(4,2) 0
(4,3) 0
(4,4) 0
[cloudera@quickstart MatrixMultiplication]$ cat /home/cloudera/MatrixMultiplication/input.txt | /home/cloudera/MatrixMultiplication/Matrix_Mapper.py
a a,0,0,5
a a,0,1,3
a a,0,2,1
a a,1,0,-2
a a,1,1,3
a a,1,2,-1
a a,2,0,7
a a,2,1,1
a a,2,2,-1
a a,3,0,0
a a,3,1,1
a a,3,2,1
b b,0,0,5
b b,0,1,9
b b,0,2,2
b b,0,3,3
b b,0,4,-4
b b,1,0,5
b b,1,1,2
b b,1,2,-6
b b,1,3,3
b b,1,4,9
b b,2,0,-10
b b,2,1,0
b b,2,2,3
b b,2,3,-9
b b,2,4,7
[cloudera@quickstart MatrixMultiplication]$

17/10/22 08:13:27 INFO mapreduce.Job: Running job: job_1508558815903_0007
17/10/22 08:19:50 INFO mapreduce.Job: Job job_1508558815903_0007 running in uber mode : false
17/10/22 08:19:50 INFO mapreduce.Job: map 0% reduce 0%
^[[1;5017/10/22 08:28:56 INFO mapreduce.Job: map 100% reduce 0%
17/10/22 08:29:41 INFO mapreduce.Job: map 100% reduce 100%
17/10/22 08:29:43 INFO mapreduce.Job: Job job_1508558815903_0007 completed successfully
17/10/22 08:29:44 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=338
  FILE: Number of bytes written=382206
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=542
  HDFS: Number of bytes written=181
  HDFS: Number of read operations=9
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Launched map tasks=2
  Launched reduce tasks=1
  Data-local map tasks=2
  Total time spent by all maps in occupied slots (ms)=987039
  Total time spent by all reduces in occupied slots (ms)=42985
  Total time spent by all map tasks (ms)=987039
  Total time spent by all reduce tasks (ms)=42985
  Total vcore-milliseconds taken by all map tasks=987039
  Total vcore-milliseconds taken by all reduce tasks=42985
  Total megabyte-milliseconds taken by all map tasks=1010727936
  Total megabyte-milliseconds taken by all reduce tasks=44016640
Map-Reduce Framework
  Map input records=27
  Map output records=27
  Map output bytes=278
  Map output materialized bytes=344
  Input split bytes=206
  Combine input records=0
  Combine output records=0
```

**Conclusion:** Thus, we implemented an Experiment on Hadoop MapReduce of simple Algorithm Matrix Multiplication.