**PARALLEL DISTRIBUTING COMPUTING**

**Assignment Title:** HADOOP

**Assignment No:** 03

**Name:** Muhammad Sharjeel Akhtar

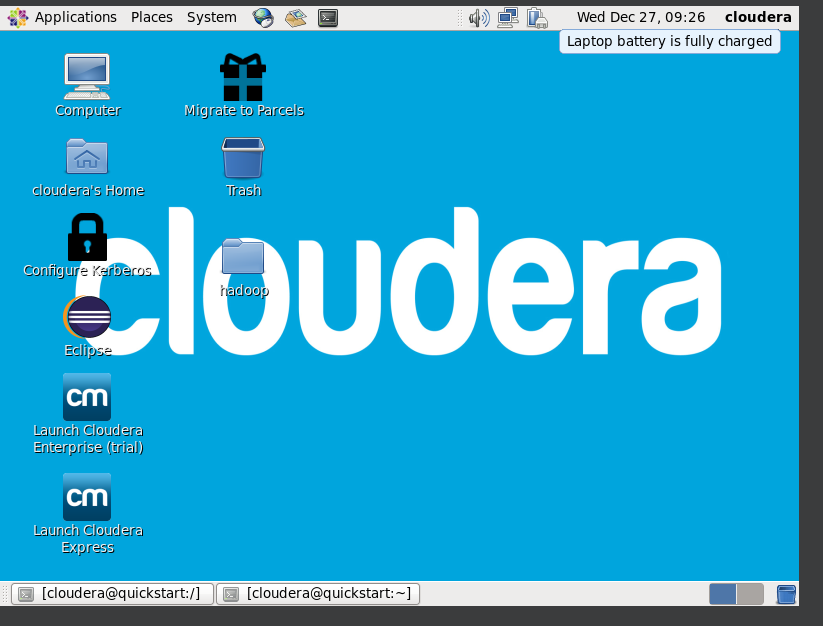
**Roll No:** 20P-0101

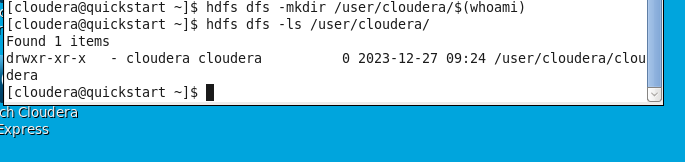
**Submitted To Respected Sir:** **Dr OMAR USMAN Khan**

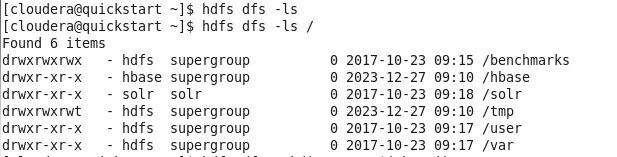
**Section:** BCS-7F

**Task 1: Creating Your Directory Space**

Performed this tasks using cloudera on Virtual Machine







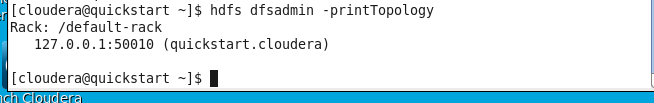


**Task 2: Understanding the System**

Using the following commands, address the questions:

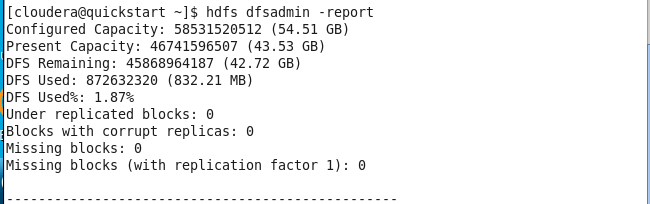
Command : hdfs dfsadmin -printTopology

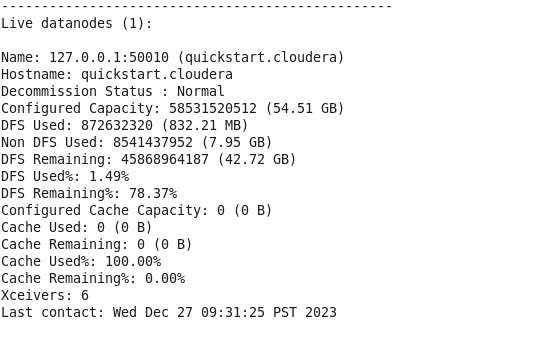
Output:



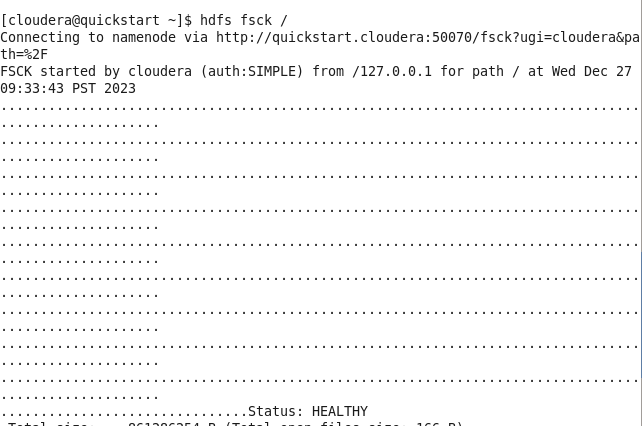
Command: hdfs dfsadmin -printTopology

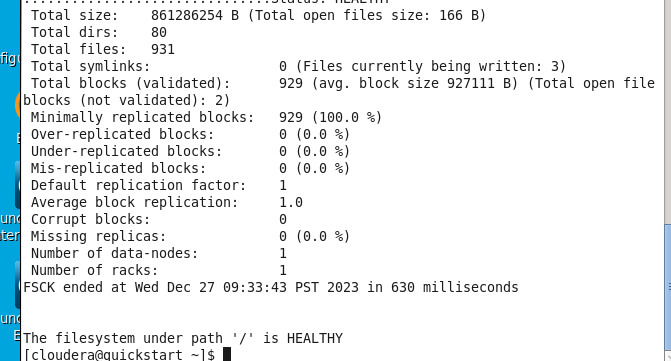
Output:





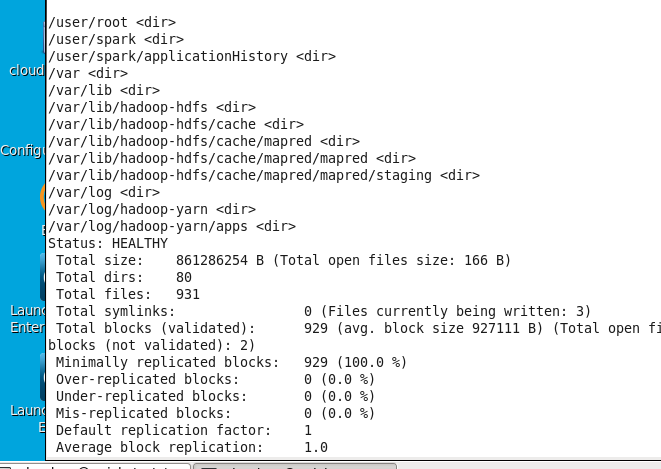
Command: hdfs fsck /

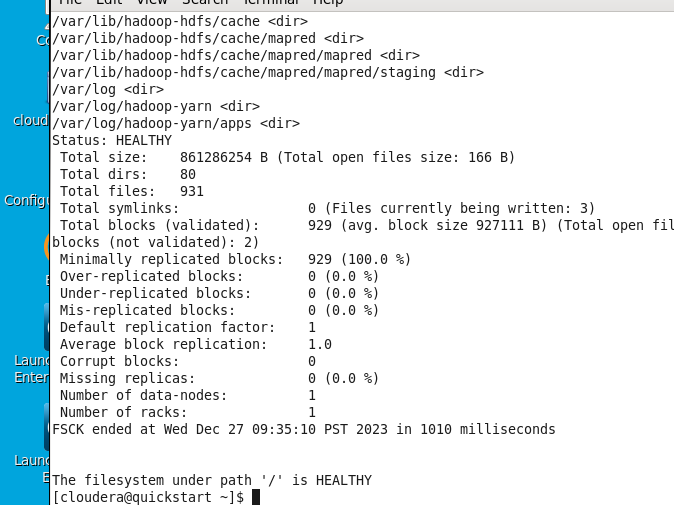




Command: hadoop fsck / -files -blocks -locations

Output:





**Questions :**

1. **How many datanodes are part of the hadoop topology?**

Ans :1

**2 What are the IP addresses of these datanodes?**

Ans: 127.0.0.1:50010

**3 What is the configured and present capacity of the HDFS?**

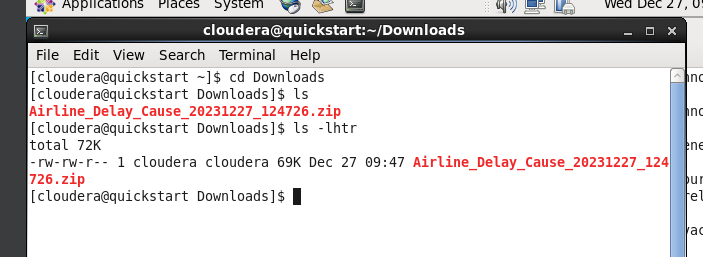
Ans: 861286254 B

**4 What is the default file replication count?**

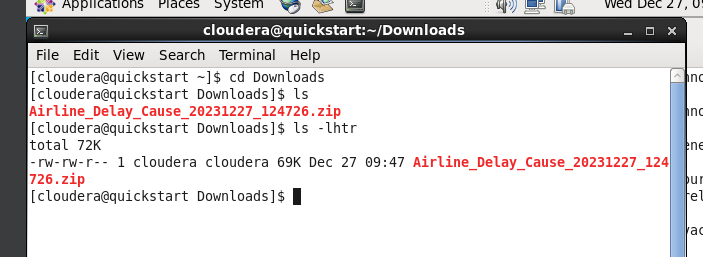
Ans:1

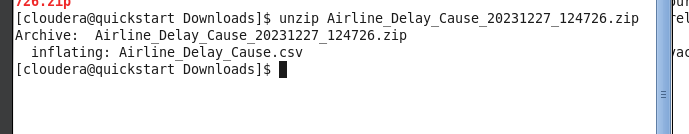
**Task 3: Getting Sample Data**

**Data Download**

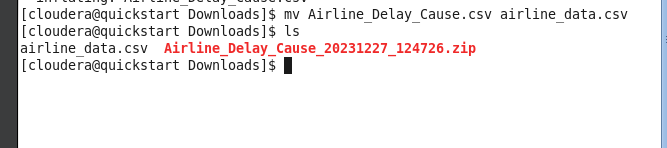


**The zip file should be the last line you see. Next step, extract the zip file using the command:**



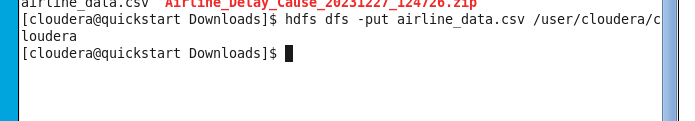


**Rename the CSV file to something simpler like airline\_data.csv:**

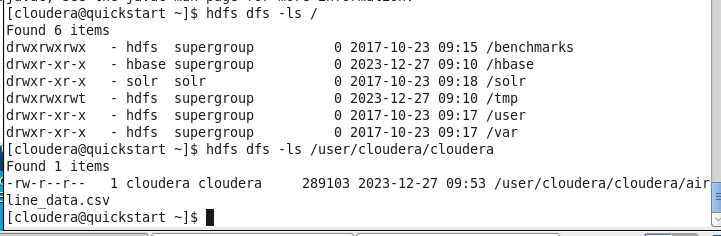


**Move Data to HDFS**

**Copy over your data using:**



**Verify that it exists by:**



**Question Answer**

**1 What is the default block size (in Mb) of the airline\_data.csv file?**

Ans: block size 289103 B

**2 Is there any missing replicas for the file airline\_data.csv?**

Ans: Missing Replicas : 0

**3 What command will you use to change this block size to 6 Mb (remember to convert into bytes)**

Ans : bytes 6MB×1024KB/MB×1024bytes/KB=6291456bytes

So, the command to set the Hadoop block size to 6 MB in bytes would be:

**bash Copy code**

**hadoop fs -D dfs.blocksize=6291456 -put <your-input-file> <your-output-directory>**



**4 How many blocks are used by airline\_data.csv after changing block size in Question 2?**

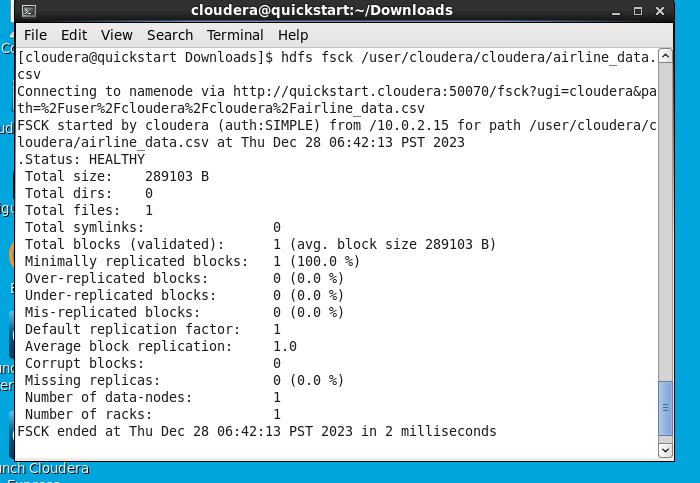
Ans : 2

**5 How many missing replicas are there for file airline\_data.csv after block change?**

**Ans : 0**

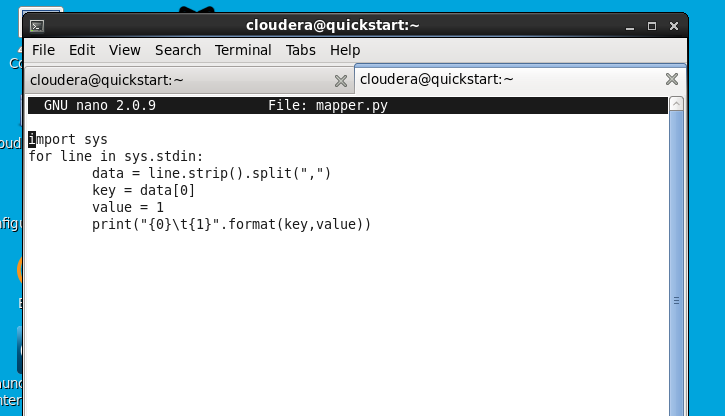
**6 Why are there missing replicas?**

Ans: Missing replicas in Hadoop may occur due to factors such as Data Node failures, under-replicated blocks, network issues, maintenance activities, configuration errors, or manual intervention. Hadoop's automatic recovery mechanisms, like replication and balancing, are designed to address these issues, but monitoring and proper configuration are essential for a well-functioning cluster.

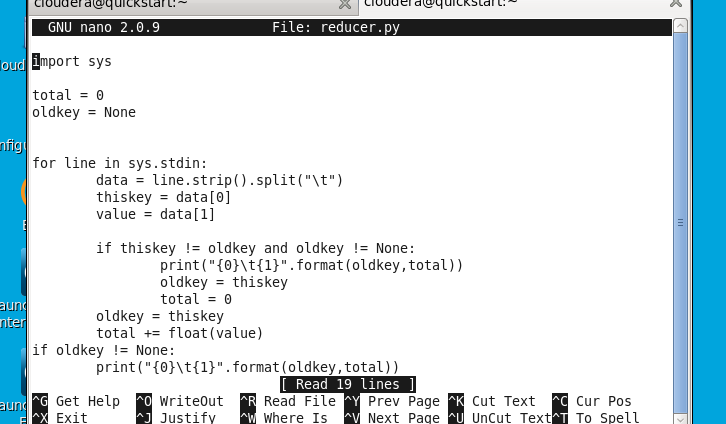


**Task 4: Setting up First Map Reduce Job**

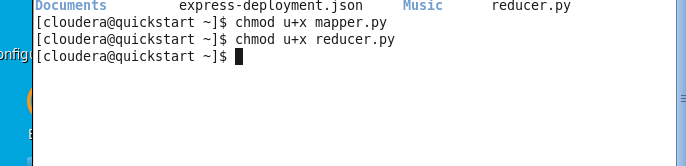
**Mapper.py code**



**Reducer.py code**

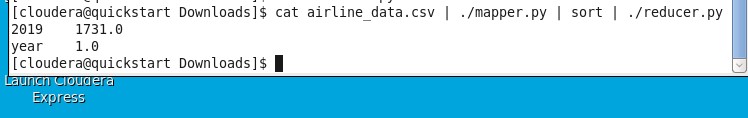


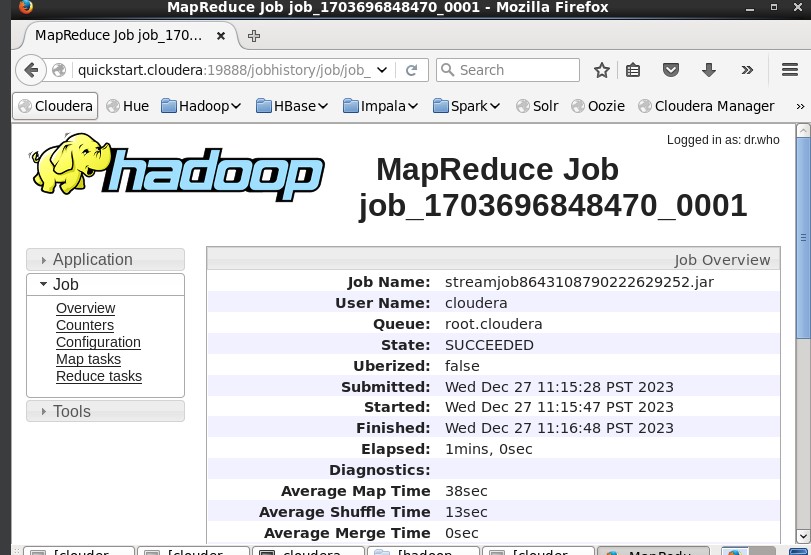
**Give both the mapper.py and Reducer.py executable permissions:**



**Testing Locally**

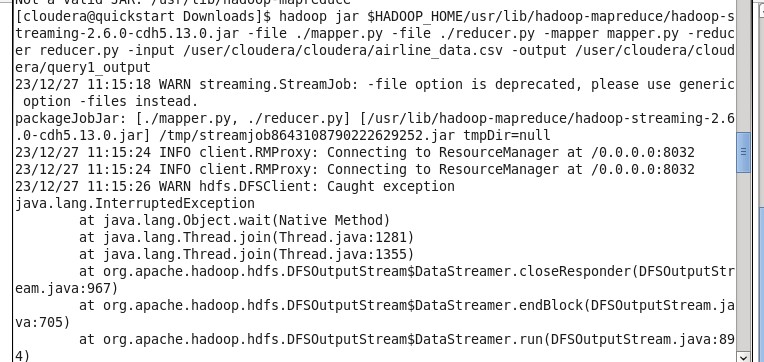
**Test the mapper and reducer on your local directory first, without map reduce:**

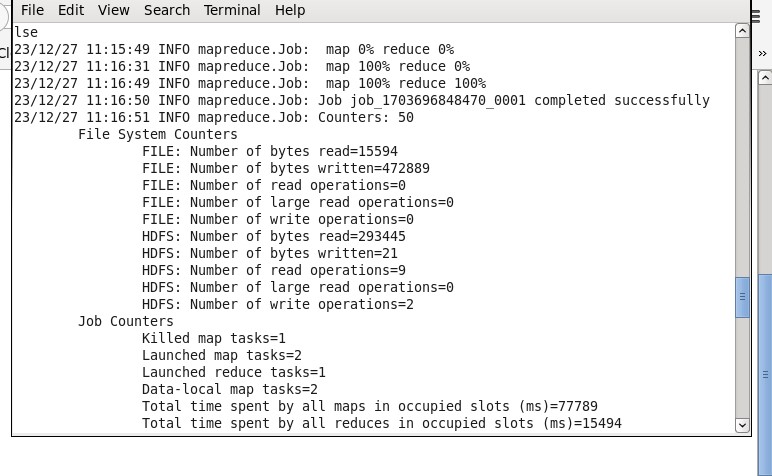


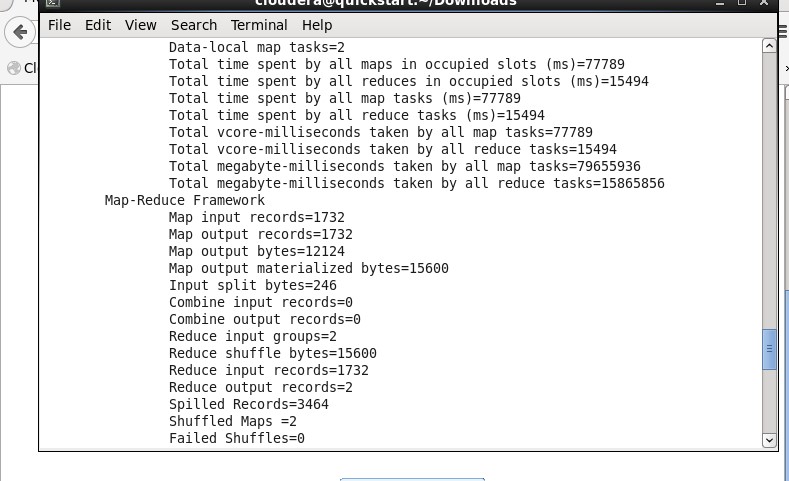


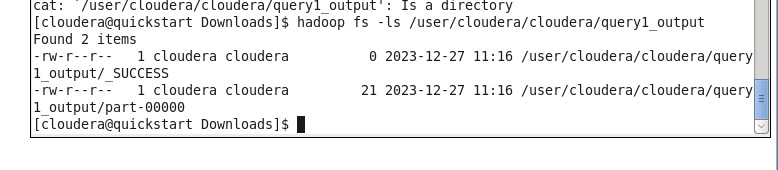
**Testing on Hadoop**

**Test the mapper and reducer using hadoop:**









**Question**

**1 What was the <key,value> pair used in this query?**

**Key, Value: 2019, 1731.0 year, 1.0**

**How many mapper threads used?**

**ANS: 2 mapper threads are used**

**How many reducers threads used?**

**ANS: 1 reducer threads are used**

**What was the time spent by all mapper threads?**

**ANS: total time spent by all map tasks: 77789 ms**

**5 What was the time spent by all reducer threads?**

**ANS: total time spent by all map tasks: 15494 ms**

**6 What is the file name in which your output is located?**

**ANS: /user/cloudera/cloudera/query1\_output**

Variation 1

For this task, you need to calculate execution time (mapper + reducer) by two variations:

1) play with block size of airline\_data.csv using the “-D dfs.blocksize=<>” argument.

2) Play with thread variation using the “-D mapred.reduce.tasks=<>”, or the “-jobconf

mapred.reduce.tasks=<>” argument.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of Reducer  Tasks |  | # of block size |  |  |
|  | **2** | **4** | **8** | **16** |
| 2 | **0m55.481s** | **0m56.993s** | **0m56.989** | **0m57.171** |
| 4 | **1m31.392s** | **1m31.106s** | **1m15.288s** | **1m17.178s** |
| 8 | **2m20.816s** | **2m8.526s** | **2m13.253s** | **2m5.728s** |
| 16 | **3m21.319s** | **3m45.676** | **2m53.510s** | **2m55.018s** |

Question Answer

1 How many output files are produced for 16 reducer

threads.

Ans : 16 output files are produced for 16 reducer threads

2 Why are some output files having 0 byte size?

Ans : Output files with 0-byte size in a Hadoop MapReduce job can be caused by: Reducer Did Not Receive Data: Uneven data distribution or partitioning issues may lead to some reducers not receiving data. Skewed Data Distribution: Skewed data, where certain keys have significantly more data, can result in some reducers having little or no data. Reducer Logic Issues: Issues in reducer logic might cause it to produce empty output, resulting in 0-byte files. Empty Output for Some Keys: If reducer logic generates empty output for certain keys, it can lead to 0-byte files.

Variation 2

For this task, you need to calculate execution time (mapper + reducer) by two variations:

1) play with block size of airline\_data.csv using the “-D dfs.blocksize=<>” argument.

2) Play with thread variation using the “-D mapred.map.tasks=<>”, or the “-jobconf

mapred.map.tasks=<>” argument.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of Map  Tasks |  | # of block size |  |  |
|  | **2** | **4** | **8** | **16** |
| 2 | **0m48.339s** | **0m47.612s** | **0m48.121s** | **0m48.816s** |
| 4 | **1m3.852s** | **1m2.836s** | **1m1.575s** | **1m2.467s** |
| 8 | **2m7.126s** | **2m29.984s** | **1m45.561s** | **1m35.970s** |
| 16 | **2m39.526s** | **3m14.034s** | **2m40.7823s** | **2m39.739s** |

Variation 3

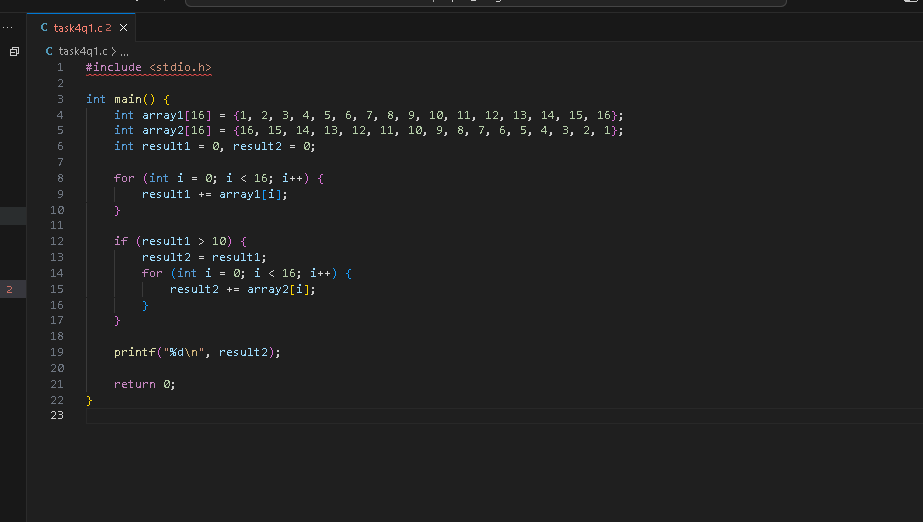
From the Variation 1 or Variation 2, choose the airline\_data.csv block size which is giving best performance. and then, for this task, you need to calculate execution time (mapper + reducer) by two variations:

1) Play with thread variation using the “-D mapred.reduce.tasks=<>”, or the “-jobconf mapred.reduce.tasks=<>” argument.

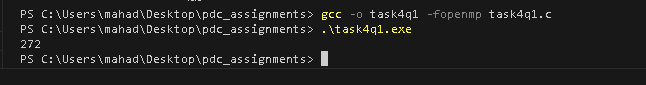
2) Play with thread variation using the “-D mapred.map.tasks=<>”, or the “-jobconf mapred.map.tasks=<>” argument.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of Map  Tasks |  | # of Reducer Tasks |  |  |
|  | **2** | **4** | **8** | **16** |
| 2 | **1m8.676s** | **1m32.515s** | **2m6.185s** | **3m54.471s** |
| 4 | **1m23.501s** | **1m58.632s** | **2m46.928s** | **4m15.635s** |
| 8 | **2m14.683s** | **2m28.847s** | **2m54.861s** | **4m37.770s** |
| 16 | **3m10.423s** | **3m52.590s** | **4m32.098s** | **5m23.564s** |

1. **and array2.Code:**

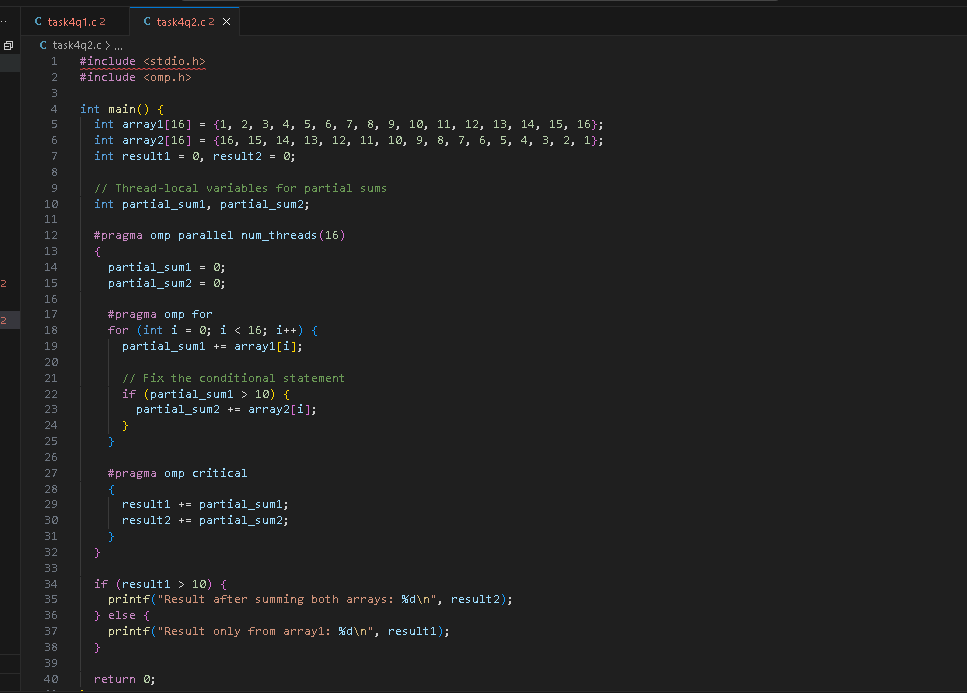


**Output:**

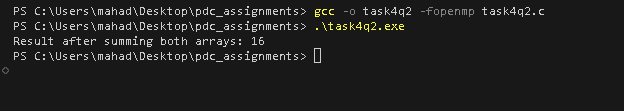


Q2 **Convert it into Parallel using 16 threads.**

**Code:**

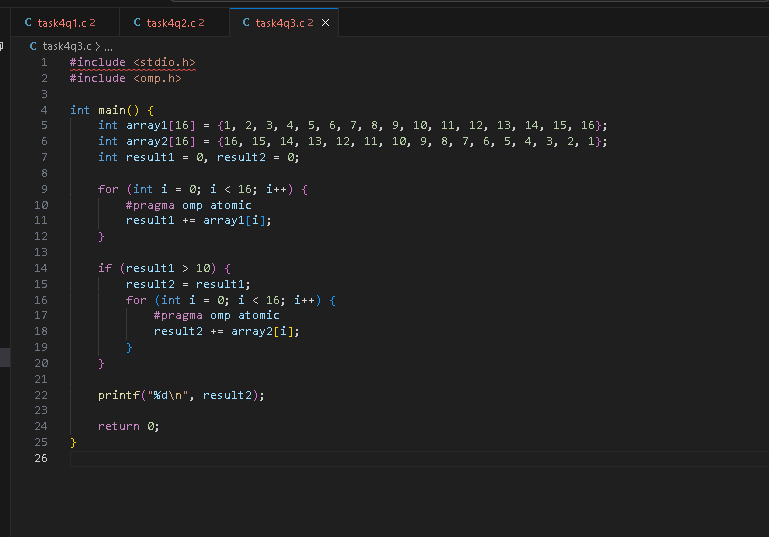


**Output:**



Q3 **Try removing the reduction() clause and add #pragma omp atomic just beore the +=. What is the effect on result? Explain.**

**Code :**



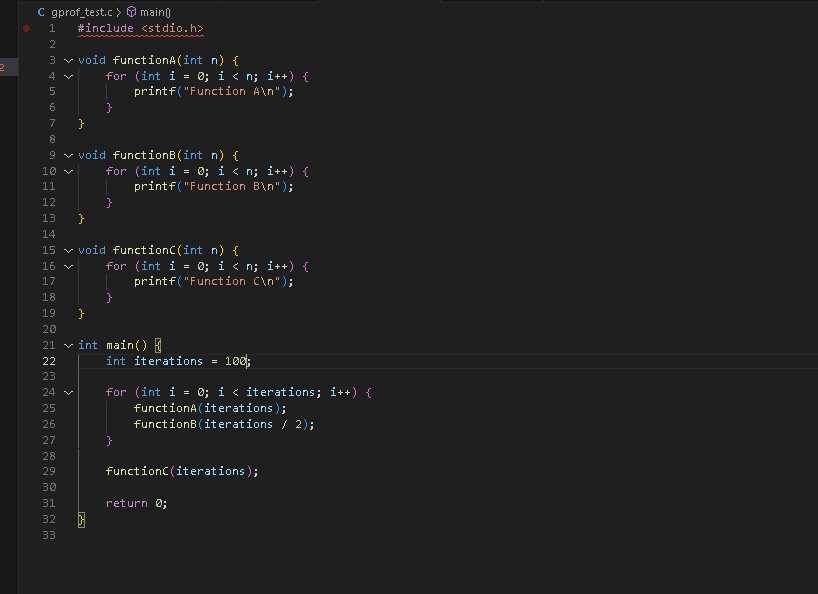
**Output:**



**Effect of result:** The #pragma omp atomic directive ensures that the specified operation is executed atomically, avoiding race conditions that may occur in parallel regions. However, using atomic operations can introduce contention, and in some cases, it might lead to decreased performance compared to using a reduction clause. In this specific code, since the updates to result1 and result2 are performed atomically, the final result should still be correct. However, the performance characteristics may vary depending on the specifics of the system and workload

Task 5

**Code:**



**Output:**

