

**Laboratory Manual**

**Subject: Machine Learning–IV (PCCS7010T) Semester: VII Class: B. Tech Experiment No. : 2**

**Title :** Perform Sorting using MapReduce.

**Pre-requisite:** Basics of Machine Learning

Hardware Requirements (if any): Windows 64 bit processor

Theory/Concept Explanation (Compulsory):

Introduction:

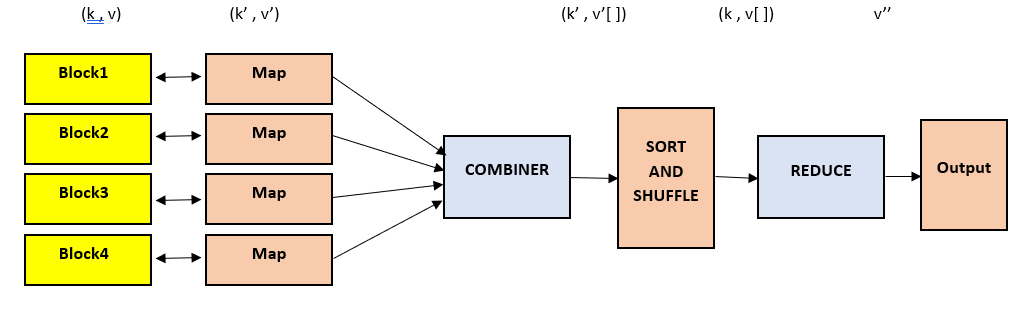
**Map Reduce:-**

It is a framework in which we can write applications to run huge amount of data in parallel and in large cluster of commodity hardware in a reliable manner.

**Different Phases of MapReduce:-**

MapReduce model has three major and one optional phase.​

* Mapping
* Shuffling and Sorting
* Reducing
* Combining
* **Mapping:-** It is the first phase of MapReduce programming. Mapping Phase accepts key-value pairs as input as (k, v), where the key represents the Key address of each record and the value represents the entire record content.​The output of the Mapping phase will also be in the key-value format (k’, v’).
* **Shuffling and Sorting: -** The output of various mapping parts (k’, v’), then goes into Shuffling and sorting phase.​ All the same values are deleted, and different values are grouped together based on same keys.​ The output of the Shuffling and Sorting phase will be key-value pairs again as key and array of values (k, v[ ]).
* **Reducer:-**  The output of the Shuffling and Sorting phase (k, v[]) will be the input of the Reducer phase.​ In this phase reducer function’s logic is executed and all the values are collected against their corresponding keys. ​Reducer stabilize outputs of various mappers and computes the final output.​
* **Combining: -** It is an optional phase in the MapReduce phases.​ The combiner phase is used to optimize the performance of MapReduce phases. This phase makes the Shuffling and Sorting phase work even quicker by enabling additional performance features in MapReduce phases.



## **Numerical:-**

#### Movie Lens Data

**USER\_ID            MOVIE\_ID             RATING             TIMESTAMP**

**196                              242                                  3                                   881250949**

**186                              302                                  3                                   891717742**

**196                              377                                  1                                   878887116**

**244                              51                                    2                                   880606923**

**166                              346                                  1                                   886397596**

**186                              474                                  4                                   884182806**

### *Solution: –*

***Step 1***– First we have to map the values, it is happen in 1st phase of Map Reduce model.

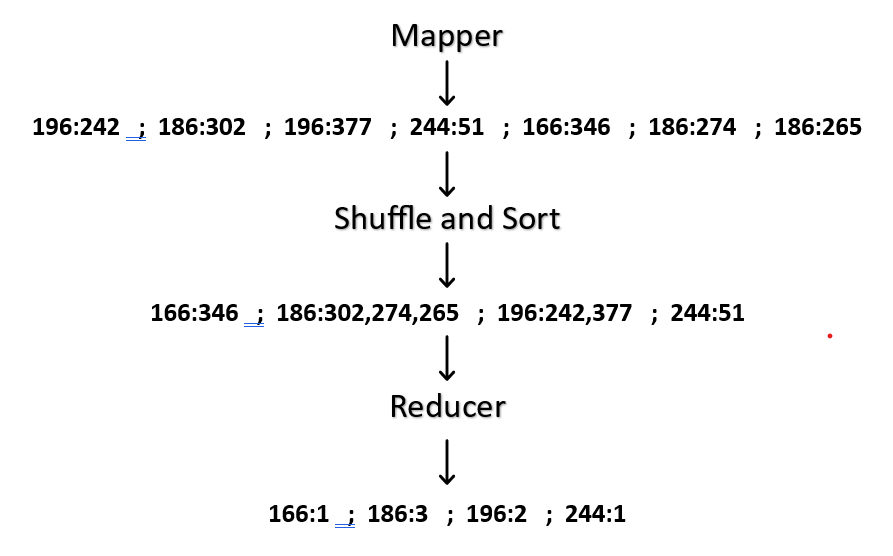
**196:242   ; 186:302   ; 196:377   ; 244:51   ; 166:346   ; 186:274   ; 186:265**

***Step 2*** – After Mapping we have to shuffle and sort the values.

**166:346   ; 186:302,274,265   ; 196:242,377   ; 244:51**

***Step 3*** – After completion of step1 and step2 we have to reduce each key’s values.

Now, put all values together



**Implementation:**

**Input:**

Data = [4, 2, 8, 6, 5, 1, 3, 7]

**Code:**

from collections import defaultdict

from functools import reduce

# Sample input data

data = [4, 2, 8, 6, 5, 1, 3, 7]

# Step 1: Map - Convert each element into key-value pairs

def map\_function(element):

return [(element, None)]

# Step 2: Shuffle and Sort - Group by key (in this case, just sort)

def shuffle\_and\_sort(mapped\_data):

mapped\_data.sort()

return mapped\_data

# Step 3: Reduce - Combine values for each key (identity function in this case)

def reduce\_function(key, values):

return key

# Implement MapReduce

def map\_reduce(data):

# Step 1: Map

mapped\_data = reduce(lambda x, y: x + y, map(map\_function, data), [])

# Step 2: Shuffle and Sort

shuffled\_and\_sorted = shuffle\_and\_sort(mapped\_data)

# Step 3: Reduce

result = defaultdict(list)

for key, value in shuffled\_and\_sorted:

result[key].append(value)

final\_output = reduce(lambda x, y: x + y, [reduce\_function(key, values) for key, values in result.items()])

return final\_output

# Run MapReduce on the sample data

output = map\_reduce(data)

print("Input Data:", data)

print("Sorted Output:", output)

**Output:**

Sorted Output: [1, 2, 3, 4, 5, 6, 7, 8]