LIST OF EXPERIMENTS

WEEK 1, 2

Implement the following Data structures in Java

a)Linked Lists b) Stacks c) Queues d) Set e) Map

WEEK 3,4

(i)Perform setting up and Installing Hadoop in its three operating modes:
Standalone,
Pseudo distributed,
Fully distributed
(ii)Use web based tools to monitor your Hadoop setup

Week 5:

Implement the following file management tasks in Hadoop:

- Adding files and directories
- Retrieving files
- Deleting files

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

Week 6:

Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

Week 7:

Write a Map Reduce program that mines weather data.
Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with MapReduce, since it is semi structured and record-oriented

Week 8:

Implement Matrix Multiplication with Hadoop Map Reduce

Week 9,10:

Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data

Week 11,12:

Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes

WEEK 1, 2

Aim:

Implement the following Data structures in Java

1) **Linked Lists b) Stacks c) Queues d) Set e) Map**

1a) Implement Linked List in java

It uses doubly linked list to store the elements.

Features

- It contains duplicate elements
- It maintains insertion order
- It can be used as list, stack or queue.

Constructors

- LinkedList()-constructs an empty list
- LinkedList(Collection c)-constructs list containing elements of Collection.

Methods

- add(int index,Object element)-adds element at the specified position in list
- addFirst(Object o)-adds the element at the beginning of list
- addLast(Object o)-appends the specified element at the end of list
- size()-returns the number of elements in list
- remove(Object o)-removes first occurrence of specified element in list

Program:

```
import java.util.LinkedList;//LinkedList class uses doubly linked list for storing elements
import java.util.Scanner;//Scanner class reads data from input stream
class Linkedlistnew
{
 public static void main(String[] args)
 {
  LinkedList<Integer> obj=new LinkedList<Integer>();
  int e;
  Object x;
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Add Element at beginning");
   System.out.println("2.Add Element at End");
```

```
  System.out.println("3.Add Element at specified position");
  System.out.println("4.Delete Element at beginning");
  System.out.println("5.Delete Element at End");
  System.out.println("6.Delete specified element");
  System.out.println("7.Display");
  System.out.println("8.Quit");
  System.out.println("Enter choice");
  int choice=s.nextInt();
  switch(choice)
  {
   case 1:System.out.println("Enter the element to be added at beginning");
        e=s.nextInt();
        obj.addFirst(e);
        break;
   case 2:System.out.println("Enter the element to be added at end");
        e=s.nextInt();
        obj.addLast(e);
        break;
   case 3:System.out.println("Enter position");
        int pos=s.nextInt();
        System.out.println("Enter the element to be added at"+pos+" position");
        e=s.nextInt();
        obj.add(pos,e);
        break;
   case 4:if(obj.isEmpty())
        System.out.println("List is empty");
        else
        {
        x=obj.getFirst();//getFirst() returns the first element in linkedlist and returns an Object
reference
        if(obj.remove(x))
        System.out.println(x+ "is deleted");
        }
        break;
   case 5:if(obj.isEmpty())
        System.out.println("List is empty");
        else
        {
        x=obj.getLast();//getLast() returns the last element in linkedlist and returns Object
reference
```

```
        if(obj.remove(x))
        System.out.println(x+ "is deleted");
        }
        break;
    case 6:if(obj.isEmpty())
        System.out.println("List is empty");
        else
        {
        System.out.println("Enter position to delete the element");
        pos=s.nextInt();
        if(pos<obj.size())
        {
        x=obj.get(pos);//get() returns the position of element in linkedlist
        obj.remove(x);
        System.out.println(x+"is deleted");
        }
        else
        System.out.println("element not found in list at that position");
        }
        break;
    case 7:if(obj.isEmpty())
        System.out.println("List is empty");
        else
        {
        System.out.println("Linked List elements");
        for(int i=0;i<obj.size();++i)//obj.size() gives no of elements in LinkedList
        System.out.println(obj.get(i));//get()-returns the element from LinkedList based on
position
        }
       break;
    case 8:System.exit(0);
    default:System.out.println("Wrong choice");
   }
  }
 }
}
```

Expected Output

1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position

4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
1
Enter the element to be added at beginning
10
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
2
Enter the element to be added at end
20
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
1
Enter the element to be added at beginning
30
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display

8.Quit
Enter choice
3
Enter position
2
Enter the element to be added at2 position
40
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
7
Linked List elements
30
10
40
20
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
4
30is deleted
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display

8.Quit
Enter choice
7
Linked List elements
10
40
20
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
6
Enter position to delete the element
2
20is deleted
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
7
Linked List elements
10
40
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display

8.Quit
Enter choice
5
40is deleted
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
7
Linked List elements
10
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
4
10is deleted
1.Add Element at beginning
2.Add Element at End
3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
7
List is empty
1.Add Element at beginning
2.Add Element at End

3.Add Element at specified position
4.Delete Element at beginning
5.Delete Element at End
6.Delete specified element
7.Display
8.Quit
Enter choice
8

1b) Implement Stack in java

Program

```
import java.util.Stack;
import java.util.Scanner;//Scanner class reads data from input stream
class Stackdemo
{
 public static void main(String[] args)
 {
  int size=5;
  Stack<Integer> obj=new Stack<Integer>();//Stack class is accepting wrapper class Integer type
of data
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Push");
   System.out.println("2.Pop");
   System.out.println("3.Display");
   System.out.println("4.Quit");
   System.out.println("Enter choice");
   int choice=s.nextInt();
   switch(choice)
   {
    case 1:System.out.println("Enter the element to be pushed into stack");
          int e=s.nextInt();
          if(obj.size()==size)
          System.out.println("Stack overflow");
          else
          obj.push(e);//pushes element into stack
          break;
```

```
    case 2:if(obj.isEmpty())//checks whether stack is empty or not
           System.out.println("Stack underflow");
           else
           System.out.println("Element popped from stack"+obj.pop());
           break;
    case 3:if(obj.isEmpty())
           System.out.println("Stack is empty");
           else
           {
           System.out.println("Stack elements");
           for(int i=obj.size()-1;i>=0;--i)//obj.size() gives no of elements in stack
           System.out.println(obj.get(i));//get()-returns the element from stack based on position
           }
           break;
    case 4:System.exit(0);
    default:System.out.println("Wrong choice");
   }
  }
 }
}
```

Expected Output

```
1.Push
2.Pop
3.Display
4.Quit
Enter choice
1
Enter the element to be pushed into stack
10
1.Push
2.Pop
3.Display
4.Quit
Enter choice
1
Enter the element to be pushed into stack
20
1.Push
2.Pop
3.Display
```

```
4.Quit
Enter choice
1
Enter the element to be pushed into stack
30
1.Push
2.Pop
3.Display
4.Quit
Enter choice
1
Enter the element to be pushed into stack
40
1.Push
2.Pop
3.Display
4.Quit
Enter choice
1
Enter the element to be pushed into stack
50
1.Push
2.Pop
3.Display
4.Quit
Enter choice
1
Enter the element to be pushed into stack
60
Stack overflow
1.Push
2.Pop
3.Display
4.Quit
Enter choice
3
Stack elements
50
40
30
```

```
20
10
1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Element popped from stack50
1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Element popped from stack40
1.Push
2.Pop
3.Display
4.Quit
Enter choice
3
Stack elements
30
20
10
1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Element popped from stack30
1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Element popped from stack20
```

1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Element popped from stack10
1.Push
2.Pop
3.Display
4.Quit
Enter choice
3
Stack is empty
1.Push
2.Pop
3.Display
4.Quit
Enter choice
2
Stack underflow
1.Push
2.Pop
3.Display
4.Quit
Enter choice
4

1 c)Implement Queue in java

Program

```
import java.util.Queue;//interface where objects cant be created
import java.util.PriorityQueue;//PriorityQueue class provides the facility of implementing queue operations but it order the elements in queue
import java.util.Scanner;//Scanner class reads data from input stream
class Queuedemo
{
 public static void main(String[] args)
 {
  int size=5;
```

```
Queue<Integer> obj=new PriorityQueue<Integer>();
Scanner s=new Scanner(System.in);
while(true)
{
 System.out.println("1.Insert");
 System.out.println("2.Delete");
 System.out.println("3.Display");
 System.out.println("4.Quit");
 System.out.println("Enter choice");
 int choice=s.nextInt();
 switch(choice)
 {
  case 1:System.out.println("Enter the element to be inserted into queue");
        int e=s.nextInt();
        if(obj.size()==size)
        System.out.println("Queue overflow");
        else
        obj.add(e);//inserts element into queue
        break;
  case 2:if(obj.isEmpty())//checks whether queue is empty or not
        System.out.println("Queue underflow");
        else
        System.out.println("Element deleted from queue"+obj.remove());
        break;
  case 3:if(obj.isEmpty())
        System.out.println("Queue is empty");
        else
        {
        System.out.println("Queue elements");
        for(Integer x:obj)//using enhanced for loop or forEach loop
        System.out.println(x);
        }
        break;
  case 4:System.exit(0);
  default:System.out.println("Wrong choice");
 }
}
```

```
  }
}
```

Expected Output

```
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
10
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
20
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
30
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
40
1.Insert
2.Delete
```

3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
50
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
60
Queue overflow
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
3
Queue elements
10
20
30
40
50
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
2
Element deleted from queue10
1.Insert
2.Delete
3.Display

4.Quit
Enter choice
2
Element deleted from queue20
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
2
Element deleted from queue30
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
2
Element deleted from queue40
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
1
Enter the element to be inserted into queue
60
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
3
Queue elements
50
60
1.Insert
2.Delete

3.Display
4.Quit
Enter choice
2
Element deleted from queue50
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
2
Element deleted from queue60
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
2
Queue underflow
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
3
Queue is empty
1.Insert
2.Delete
3.Display
4.Quit
Enter choice
4

1 d) Implement Set using java

HashSet

It is used to create collection which uses hash table for storage of data and implements Set Interface.

Features

- It stores the elements by using mechanism called Hashing
- It contains unique elements only.
- It does not maintain insertion order

Constructors

- HashSet()-initializes empty HashSet
- HashSet(Collection c)-constructs HashSet by using the elements of Collection

Methods

- add(Object o)-adds the specified element to this set if it is not already present
- remove(Object o)-removes specified element from set
- clear()-removes all elements from the set
- size()-counts the number of elements in set
- iterator()- iterates over the elements in set

Program on HashSet

```
import java.util.Set;
import java.util.HashSet;
import java.util.Scanner;//Scanner class reads data from input stream
class Hashsetdemo
{
 public static void main(String[] args)
 {
  Set<Integer> obj=new HashSet<Integer>();//Set is interface and HashSet is class
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Add");
   System.out.println("2.Remove");
   System.out.println("3.Display");
   System.out.println("4.Quit");
   System.out.println("Enter choice");
   int choice=s.nextInt();
   switch(choice)
   {
    case 1:System.out.println("Enter the element to be added to hashset");
           int e=s.nextInt();
           obj.add(e);//add element to hash set
           break;
    case 2:if(obj.isEmpty())//checks whether hashset is empty or not
           System.out.println("Hashset is empty");
```

```
        else
        {
        System.out.println("Enter the element to be removed from hashset");
        e=s.nextInt();


        if(obj.contains(e))//contains() checks whether hashset contains element or not
        {
        obj.remove(e);
        System.out.println(e+" is removed from HashSet successfully");
        }
        else
        System.out.println("Element not found in hashset to be deleted");
        }
        break;
    case 3:if(obj.isEmpty())
         System.out.println("Hashset is empty");
        else
        {
        System.out.println("HashSet elements");
        for(Integer i:obj)//enhanced for loop
        System.out.println(i);
       //prints the elements using size()
      // convert hashset  to array
       // Integer[] i=obj.toArray(new Integer[obj.size()]);
        //for(int x=0;x<i.length;++x)
        //System.out.println(i[x]);
        // using Iterator interface to print all the elements in hashset
        //java.util.Iterator<Integer> y=obj.iterator();
        //while(y.hasNext())// checks whether hashset contains any more elements or not
        //System.out.println(y.next());//prints element
        }
        break;
    case 4:System.exit(0);
    default:System.out.println("Wrong choice");
   }
  }
 }
}
```

Expected Output

```
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to hashset
10
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to hashset
20
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to hashset
30
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to hashset
40
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
HashSet elements
20
```

```
40
10
30
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from hashset
12
Element not found in hashset to be deleted
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from hashset
40
40 is removed from HashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
HashSet elements
20
10
30
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from hashset
30
30 is removed from HashSet successfully
```

```
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
HashSet elements
20
10
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from hashset
10
10 is removed from HashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
HashSet elements
20
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from hashset
20
20 is removed from HashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
```

2
Hashset is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
Hashset is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
4

LinkedHashSet

This class is used to create collection which extends HashSet class and implements Set interface.

Features

- It stores the elements by using mechanism called Hashing
- It contains unique elements only.
- It maintains insertion order

LinkedHashSet program

```
import java.util.Set;
import java.util.LinkedHashSet;
import java.util.Scanner;//Scanner class reads data from input stream
class Linkedhashsetdemo
{
 public static void main(String[] args)
 {
  Set<Integer> obj=new LinkedHashSet<Integer>();//Set is interface and LinkedHashSet is class
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Add");
   System.out.println("2.Remove");
   System.out.println("3.Display");
   System.out.println("4.Quit");
   System.out.println("Enter choice");
   int choice=s.nextInt();
   switch(choice)
```

```
{
 case 1:System.out.println("Enter the element to be added to linked hashset");
        int e=s.nextInt();
        obj.add(e);//add element to linked hash set
        break;
 case 2:if(obj.isEmpty())//checks whether linked hashset is empty or not
        System.out.println("Linked Hashset is empty");
        else
        {
        System.out.println("Enter the element to be removed from Linked hashset");
        e=s.nextInt();
        if(obj.contains(e))//contains() checks whether linkedhashset contains element or not
        {
        obj.remove(e);
        System.out.println(e+" is removed from LinkedHashSet successfully");
        }
        else
        System.out.println("Element not found in linkedehashset to be deleted");
        }
        break;
 case 3:if(obj.isEmpty())
         System.out.println("LinkedHashset is empty");
        else
        {
        System.out.println("LinkedHashSet elements");
        for(Integer i:obj)//enhanced for loop
        System.out.println(i);
       //prints the elements using size()
      // convert hashset  to array
       // Integer[] i=obj.toArray(new Integer[obj.size()]);
       //for(int x=0;x<i.length;++x)
       //System.out.println(i[x]);
       // using Iterator interface to print all the elements in hashset
       //java.util.Iterator<Integer> y=obj.iterator();
       //while(y.hasNext())// checks whether hashset contains any more elements or not
       //System.out.println(y.next());//prints element
        }
        break;
 case 4:System.exit(0);
 default:System.out.println("Wrong choice");
```

```
    }
   }
  }
}
```

Expected Output

```
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to linked hashset
10
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to linked hashset
20
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to linked hashset
30
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to linked hashset
40
1.Add
2.Remove
```

```
3.Display
4.Quit
Enter choice
1
Enter the element to be added to linked hashset
50
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
LinkedHashSet elements
10
20
30
40
50
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from Linked hashset
30
30 is removed from LinkedHashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from Linked hashset
25
Element not found in linkedehashset to be deleted
1.Add
2.Remove
3.Display
4.Quit
```

```
Enter choice
3
LinkedHashSet elements
10
20
40
50
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from Linked hashset
40
40 is removed from LinkedHashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from Linked hashset
10
10 is removed from LinkedHashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
LinkedHashSet elements
20
50
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
```

Enter the element to be removed from Linked hashset
20
20 is removed from LinkedHashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from Linked hashset
50
50 is removed from LinkedHashSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Linked Hashset is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
LinkedHashset is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
4

TreeSet

It uses the tree for storage of elements which are maintained in increasing order.

Features

- Contains unique elements
- Fast retrieval and access of elements
- Maintains ascending order

Constructors

- TreeSet()- constructs an empty TreeSet that will be sorted in an ascending order
- TreeSet(Collection c)-constructs new TreeSet that contains the elements of collection

Methods

- add(Object o)-adds the specified element to this set if it is not already present
- remove(Object o)-removes specified element from set
- clear()-removes all elements from the set
- size()-counts the number of elements in set
- first()-returns the lowest element currently in this sorted set
- last()-returns highest element currently in this sorted set
- iterator()- iterates over the elements in set

TreeSet program

```
import java.util.Set;
import java.util.TreeSet;
import java.util.Scanner;//Scanner class reads data from input stream
class Treesetdemo
{
 public static void main(String[] args)
 {
  Set<Integer> obj=new TreeSet<Integer>();//Set is interface and TreeSet is class
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Add");
   System.out.println("2.Remove");
   System.out.println("3.Display");
   System.out.println("4.Quit");
   System.out.println("Enter choice");
   int choice=s.nextInt();
   switch(choice)
   {
    case 1:System.out.println("Enter the element to be added to treeset");
           int e=s.nextInt();
           obj.add(e);//add element to tree set
           break;
    case 2:if(obj.isEmpty())//checks whether treeset is empty or not
           System.out.println("Hashset is empty");
           else
           {
```

```
        System.out.println("Enter the element to be removed from treeset");
        e=s.nextInt();
        if(obj.contains(e))//contains() checks whether treeset contains element or not
        {
        obj.remove(e);
        System.out.println(e+" is removed from treeSet successfully");
        }
        else
        System.out.println("Element not found in treeset to be deleted");
        }
        break;
    case 3:if(obj.isEmpty())
         System.out.println("treeset is empty");
        else
        {
        System.out.println("TreeSet elements");
        for(Integer i:obj)//enhanced for loop
        System.out.println(i);
       //prints the elements using size()
      // convert hashset  to array
       // Integer[] i=obj.toArray(new Integer[obj.size()]);
        //for(int x=0;x<i.length;++x)
        //System.out.println(i[x]);
        // using Iterator interface to print all the elements in hashset
        //java.util.Iterator<Integer> y=obj.iterator();
        //while(y.hasNext())// checks whether hashset contains any more elements or not
        //System.out.println(y.next());//prints element
        }
        break;
    case 4:System.exit(0);
    default:System.out.println("Wrong choice");
   }
  }
 }
}
```

Expected Output

1.Add
2.Remove
3.Display
4.Quit

```
Enter choice
1
Enter the element to be added to treeset
20
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to treeset
10
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to treeset
25
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to treeset
45
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the element to be added to treeset
42
1.Add
2.Remove
3.Display
4.Quit
```

```
Enter choice
3
TreeSet elements
10
20
25
42
45
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from treeset
25
25 is removed from treeSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
TreeSet elements
10
20
42
45
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the element to be removed from treeset
44
Element not found in treeset to be deleted
1.Add
2.Remove
3.Display
```

4.Quit
Enter choice
2
Enter the element to be removed from treeset
10
10 is removed from treeSet successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
TreeSet elements
20
42
45
1.Add
2.Remove
3.Display
4.Quit
Enter choice
4

1 e) Implement Maps in java

It contains values on the basis of key i.e. key and value pair. Each key and value pair is known as an entry. It contains only unique keys. It is useful if you have to search, update or delete elements on the basis of key.

Methods of Map interface

- put(Object key,Object value)-inserts an entry into map
- get(Object key)-returns value for specified key
- remove(Object key)-removes an entry for the specified key
- containsKey(Object key)-search specified key from the map
- keySet()-returns Set view containing all keys
- entrySet()-returns Set view containing all keys and values
- putAll(Map m)-inserts specified map in this map**.**

Implementation of Map interface

- HashMap
- LinkedHashMap
- TreeMap

TreeMap

It implements Map interface using a tree which provides efficient way of storing key/value pairs in sorted order

Features

- It contains values based on key
- It contains only unique elements
- It is similar to HashMap instead it store the key/value pairs in sorted order.

Constructors

- TreeMap()-constructs an empty treemap that will be sorted based on key
- TreeMap(Map m)-initializes treemap with m which will be sorted based on key

TreeMap program

```
import java.util.Map;//Map is interface
import java.util.Map.Entry;//Map.Entry is sub interface of Map
import java.util.TreeMap;
import java.util.Scanner;//Scanner class reads data from input stream
class Treemapnew
{
 public static void main(String[] args)
 {
   Map<Integer,String> obj=new TreeMap<Integer,String>();//Map is interface and TreeMap is class
  Scanner s=new Scanner(System.in);
  while(true)
  {
   System.out.println("1.Add");
   System.out.println("2.Remove");
   System.out.println("3.Display");
   System.out.println("4.Quit");
   System.out.println("Enter choice");
   int choice=s.nextInt();
   switch(choice)
   {
    case 1:System.out.println("Enter the key(id) to be added to treemap");
          int key=s.nextInt();
          System.out.println("Enter the value(name) to be added to treemap");
          String value=s.next();
         obj.put(key,value);//place <key,value> into treemap
          break;
    case 2:if(obj.isEmpty())//checks whether treemap is empty or not
          System.out.println("Treemap is empty");
          else
```

```
        {
        System.out.println("Enter the key to be removed from treemap");
        key=s.nextInt();
        if(obj.containsKey(key))//containsKey() checks whether treemap contains key or not
        {
        value=obj.get(key);
        obj.remove(key);
        System.out.println("<"+key+","+value+"> pair is removed from treemap successfully");
        }
        else
        System.out.println("Key not found in treemap to be deleted");
        }
        break;
    case 3:if(obj.isEmpty())
         System.out.println("treemap is empty");
        else
        {
        System.out.println("Treemap elements");
        for( Map.Entry<Integer,String> m:obj.entrySet())
       //Map.entry interface and entrySet() returns <key,value> pair
        System.out.println(m.getKey()+","+m.getValue());
      //Object[] a=obj.keySet().toArray();//obj.keySet() returns set of keys and converts the given
set into array
      //now using get(key) of TreeMap retrieve value
  /*for(int i=0;i<obj.size();++i)
  {
   System.out.println(obj.get(a[i]));
  }  */
       }
        break;
    case 4:System.exit(0);
    default:System.out.println("Wrong choice");
   }
  }
 }
}
```

Expected Output

1.Add
2.Remove
3.Display

```
4.Quit
Enter choice
1
Enter the key(id) to be added to treemap
100
Enter the value(name) to be added to treemap
raju
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the key(id) to be added to treemap
95
Enter the value(name) to be added to treemap
ravi
1.Add
2.Remove
3.Display
4.Quit
Enter choice
1
Enter the key(id) to be added to treemap
98
Enter the value(name) to be added to treemap
venu
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
Treemap elements
95,ravi
98,venu
100,raju
1.Add
2.Remove
3.Display
```

```
4.Quit
Enter choice
2
Enter the key to be removed from treemap
96
Key not found in treemap to be deleted
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the key to be removed from treemap
95
<95,ravi> pair is removed from treemap successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
Treemap elements
98,venu
100,raju
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the key to be removed from treemap
100
<100,raju> pair is removed from treemap successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
Treemap elements
```

98,venu
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Enter the key to be removed from treemap
98
<98,venu> pair is removed from treemap successfully
1.Add
2.Remove
3.Display
4.Quit
Enter choice
3
treemap is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
2
Treemap is empty
1.Add
2.Remove
3.Display
4.Quit
Enter choice
4

Viva Questions

1. **Define Data Structure**

Ans: It refers to the systematic way to organize the data so that meaningful information is produced in an efficient manner

2. **List the advantages of Data structures.**

Ans: The advantages of data structures are summarized below

Efficiency: Proper choice of data structure makes program efficient**.**

Re-usability: Data structures are reusable i.e, a particular data structure can be used in any other place depending on the requirement.

Abstraction: A data structure is specified by ADT which provides level of abstraction. The client program uses the data structure through the interface only without getting into implementation details**.**

3. Define Stack data structure.

It is a linear data structure in which insertion of elements into stack and deletion of elements from stack are performed at only one end called top of the stack. It is LIFO(Last in First out) structure

4. **Define Queue data structure**

Ans: It is linear data structure in which elements can be inserted at one end called rear end of queue and elements can be deleted at other end called front end of queue. It is FIFO(First in First Out) structure

5. **List the applications of queue**

Ans:

1. Resource sharing among multiple users
2. Printer spooling

6. **Write the code snippet for push and pop operations on stacks using arrays**

Ans.**Push operation**

```
if(top==size-1)
System.out.println("Stack overflow");
else
{
  top=top+1;
  arr[top]=element;
 }
```

Pop operation

```
if(top==-1)
System.out.println("Stack underflow");
else
{
   System.out.println(arr[top]);
   top=top-1;
 }
```

7. **Write the code snippet for insert and delete operations on queues using arrays**

Ans.**Insert operation**

```
if(rear==size-1)
System.out.println("Queue overflow");
else
{
  if(front==-1)
  front=0;
```

```
  arr[++rear]=element;
 }
```

Delete operation

```
if(front==-1 || front==rear+1)
System.out.println("queue underflow");
else
{
   System.out.println(arr[front]);
   front=front+1;
}
```

8. **Define Collections**

Ans: It is a framework that provides an architecture to store and manipulate group of objects. Operations such as searching, sorting, insertion, deletion etc are performed by collections. It provides many interfaces(Set, List, Queue, Dequeue etc) and classes(ArrayList, LinkedList, HashSet,LinkedHashSet etc)

9.**List the features of ArrayList class.**

Ans:

Features

- It contains duplicate elements
- It maintains insertion order
- It allows random access

10. **List the features of LinkedList class.**

Ans:

- It contains duplicate elements
- It maintains insertion order
- It can be used as list, stack or queue.

11. **List the differences between ArrayList and LinkedList**

Ans:

ArrayList	LinkedList
It uses dynamic array for storing the elements	It uses doubly linked list for storing the elements
Manipulation is slow since it internally uses array	Manipulation is faster than ArrayList since it uses doubly linked list

It can act as list only because it implements List interface only	It can act as list and queue both since it implements List and Dequeue interfaces
It is better for storing and accessing the data	It is better for manipulating the data

12. **List the differences between List and Set interfaces**

Ans:

List	Set
It is ordered collection of elements	It is unordered collection of elements
It contains duplicate values	It does not contain duplicate values
Implementations of list: ArrayList, LinkedList etc	Implementations of Set: HashSet, TreeSet,LinkedHashSet etc
It permits any number of null values to be stored	It permits only one null value to be stored
It can be traversed in forward and backward directions using ListIterator	It can be traversed in only forward direction with the help of iterator

13. **Define Map interface**

Ans: It contains values on the basis of key i.e. key and value pair. Each key and value pair is known as an entry. It contains only unique keys. It is useful if you have to search, update or delete elements on the basis of key.

14. **Define Generics and its advantages**

Ans:It is creation of programming constructs that allows the method to operate on objects of various types while providing compile time safety

Advantages

- type casting is not required
- provides type checking of objects which can be done at compile time only.

15. **Define Generic class and its syntax**

Ans: It is class which can refer to any type. T type parameter is used to create generic class of specific type.

Syntax for creating generic class

```
class Genericclassname<typevar1,typevar2.....typevarn>
{
   instance variables
   constructors
   methods
}
```

16. **List the type parameters in Generics**

Ans:
T-General type
E-Element type in collection
K-key type in map
V-value type in map

17. Define**Generic method and its syntax**

Ans: Generic method is a method with type parameter.

Syntax for declaring generic method

```
accessmodifier<typevar1,typevar2,.....typevarn> returntype methodname(parameters)
{
  //body
}
```

18. **Define Wrapper classes and its usage**

Ans: They are the classes whose object contains or wraps primitive data types

Need for Wrapper Classes

- They convert primitive data types into objects(Autoboxing) and objects to primitive data types(Unboxing)
- Data Structures in Collection Framework such as ArrayList and Vector store only objects

19. **Define Serialization**

Ans: It is mechanism of writing the state of object into byte stream. It is mainly used in Hibernate, EJB, JMS technologies etc. The reverse operation of serialization is called deserialization. Serializable interface is used to explain serialization process.

20. **Define Serializable interface**

Ans: It is marker interface(consists of no data members and methods) which must be implemented by class whose object needs to persist. Cloneable and Remote are also marker interfaces. The String class and all the wrapper classes implement this interface by default

WEEK 3, 4

Aim:

(i) Perform setting up and Installing Hadoop in its three operating modes:
Standalone,
Pseudo distributed,
Fully distributed

(ii)Use web based tools to monitor your Hadoop setup

Ans) It is open source framework written in java which is used to store, analyze and process huge amounts of data in distributed environment across clusters of computers in an efficient manner. It provides capability to process distributed data using simplified programming model. It is used by Google, facebook, yahoo, youtube, twitter etc. It is developed by Doug Cutting at Yahoo in 2006 which is inspired from Google File System and Google Map Reduce algorithm. It is file system provided by linux to store the data.

Operational modes of configuring Hadoop cluster

Hadoop can be run in one of the three supported modes

1) Local(Standalone) mode-By default, Hadoop is configured to run in a single-node, non-distributed mode, as a single Java process. This is useful for debugging. The usage of this mode is very limited and it can be only used for experimentation.

2) Pseudo-Distributed mode-Hadoop is run on a single-node in a pseudo distributed mode where each Hadoop daemon(Namenode, Datanode, Secondary Namenode, Jobtracker, Tassktracker) runs in a separate Java process. In Local mode, Hadoop runs as a single Java process

3) Fully distributed mode-In this mode, all daemons are executed in a separate nodes forming a multi node cluster. This setup offers true distributed computing capability and offers built-in reliability, scalability and fault tolerance

Standalone mode

1) Add Java software information to repository

$ sudo add-apt-repository ppa:webupd8team/java

2) Update repository

$ sudo apt-get update

3) Install Java 8

$ sudo apt-get install oracle-java8-installer

4) Verify which java version is installed

$ java -version

5) Install Hadoop-2.8.1

$ wget

http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.tar.gz

(or) $ wget

http://www-us.apache.org/dist/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.tar.gz

6) Extract tar.gz file and hadoop-2.8.1 folder is created

$ tar -zxvf hadoop-2.8.1.tar.gz

7) Ensure HADOOP_HOME is correctly set in .bashrc file

export HADOOP_HOME=hadoop-2.8.1

export PATH=$PATH:$HADOOP_HOME/bin

8) Evaluate .bashrc file

$ source ~/.bashrc

9) Verify hadoop is working or not by issuing the following command

$ hadoop version

Pseudo-distributed mode

1) Configure Hadoop

Hadoop configuration files can be found in $HADOOP_HOME/etc/hadoop. In order to develop hadoop programs in java, location of java must be set in hadoop-env.sh file

export $JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131

2) Several files needed to configure Hadoop located in $HADOOP_HOME/etc/hadoop are described below

a) core-site.xml (contains configuration settings that hadoop uses when started. It contains information such as the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, and size of Read/Write buffers. It also specifies where Namenode runs in the cluster.

```
<configuration>
    <property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:54310</value>
```

<description>name of default file system. URI is used to specify hostname, port number for file system.</description>

</property>

</configuration>

b) hdfs-site.xml (contains information such as the value of replication data, namenode path, and datanode paths of local file systems)

```
<configuration>
   <property>
     <name>dfs.replication</name>
     <value>1</value>
     <description> actual  number of block replications(copies) can be specified when file is created.</description>
    </property>
    <property>
       <name>dfs.namenode.name.dir</name>
       <value>hadoop-2.8.1/namenodenew</value>
       <description> directory for namenode</description>
   </property>
   <property>
        <name>dfs.datanode.data.dir</name>
         <value>hadoop-2.8.1/datanodenew</value>
         <description>directory for data node</description>
    </property>
</configuration>
```

1.3.**Format namenode**

$ hdfs namenode -format

1.4. **start single node cluster**

$ start-dfs.sh

1.5. **check hadoop daemons are running or not by using jps(java virtual machine process status tool)**

$ jps

13136 DataNode

13427 SecondaryNameNode

12916 NameNode

13578 Jps

6. Access hadoop on browser by http://localhost:50070 (50070 is default port number to access hadoop on browser)

Fully-distributed mode

Steps to be followed on configuring master and slave nodes

1.Create same account on all nodes to use hadoop installation

$ sudo useradd cse (cse is username)

$ sudo passwd cse (Enter password for cse)

2.Edit /etc/hosts file on all nodes which specifies IP address of each node followed by hostnames(assumption) and add the following lines

192.168.100.22 master
192.168.100.23 slave1
192.168.100.24 slave2

3) Install SSH on all nodes

$ sudo apt-get install openssh-server

4) Configure key based login on all nodes which communicate with each other without prompting for password

$ su cse (super user switching to cse account)

$ ssh-keygen -t rsa -P "" (public key is generated)

$ ssh-copy-id -i /home/cse/.ssh/id_rsa.pub cse@slave1 (copy public key from master to slave nodes)

$ ssh-copy-id -i /home/cse/.ssh/id_rsa.pub cse@slave2

$ exit

Installing Hadoop on Master nodes

5.$ sudo mkdir /usr/local/hadoop (create hadoop folder)

6. **$wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.tar.gz**
7. Extract tar.gz file and hadoop-2.8.1 is created

   $ tar -zxvf hadoop-2.8.1.tar.gz

8. **sudo mv hadoop-2.8.1 /usr/local/hadoop (move hadoop installation folder to newly created directory)**
9. **sudo chown -R cse /usr/local/hadoop/hadoop-2.8.1 (making cse owner of hadoop folder)**

Configuring Hadoop on master node

10.

a) **core-site.xml (contains configuration settings that hadoop uses when started. It contains information such as the port number used for Hadoop instance,**

memory allocated for the file system, memory limit for storing the data, and size of Read/Write buffers. It also specifies where Namenode runs in the cluster.

```
<configuration>
    <property>
      <name>fs.default.name</name>
      <value>hdfs://master:54310</value>
            <description>name of default file system. URI is used to specify hostname, port number for file system.</description>
    </property>
</configuration>
```

b) hdfs-site.xml (contains information such as the value of replication data, namenode path, and datanode paths of local file systems)

```
<configuration>
   <property>
     <name>dfs.replication</name>
     <value>3</value>
     <description> actual  number of block replications(copies) can be specified when file is created.</description>
    </property>
    <property>
      <name>dfs.namenode.name.dir</name>
      <value>/usr/local/hadoop/hadoop-2.8.1/namenodenew</value>
      <description> directory for namenode</description>
   </property>
   <property>
       <name>dfs.datanode.data.dir</name>
        <value>/usr/local/hadoop/hadoop-2.8.1/datanodenew</value>
        <description>directory for data node</description>
    </property>
</configuration>
```

11) Ensure HADOOP_HOME is correctly set in .bashrc file

```
export HADOOP_HOME=/usr/local/hadoop/hadoop-2.8.1
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

12) Evaluate ~/.bashrc file

```
$ source ~/.bashrc
```

13) Hadoop configuration files can be found in $HADOOP_HOME/etc/hadoop. In order to develop hadoop programs in java, location of java must be set in hadoop-env.sh file

```
export  $JAVA_HOME=/usr/lib/jvm/jdk1.8.0_131
```

14) update remaining configuration files in master node

$ sudo gedit slaves ($HADOOP_HOME/etc/hadoop folder)
slave1
slave2
$ sudo gedit masters
master

15) transfer hadoop folder from master node to slave nodes
$ scp -r /usr/local/hadoop cse@slave1:/home/cse
$ scp -r /usr/local/hadoop cse@slave2:/home/cse

16) format namenode on master node
$ hdfs namenode -format

17) start hadoop cluster on master node
$ start-dfs.sh

18) Verify hadoop daemon on slave nodes or master nodes using jps
$ jps

19) access hadoop on browser on slave nodes using http://master:50070

Viva Questions

1. Describe the categories of Big data.

Ans: Big data can be categorized into three types. They are
a) Structured Data- It is defined as data organized in predefined manner which is simple to store and analyze.
Eg: RDBMS tools
b) Unstructured Data-It is defined as data which is not organized in predefined manner so it is difficult for data analysts to understand and process.
Eg: Text files, images, videos, email, PDF files, PPT, social media data, web pages etc
c) Semi-structured data-It is form of structured data that does not conform with the formal structure of data models associated with relational databases.
Eg: CSV, XML, JSON etc

2. Describe characterization of Big data.

Ans: Big data is characterized as 3V's. They are
a) Volume-amount of data which is increased to range of petabytes
b) Variety-data comes in different types of formats such as text, image, audio, video etc
c) Velocity-it is measure of how data is generating at faster rate. Eg: share market, video streaming

3. **List the advantages of Big data**.
Ans:

1) it allows businesses to develop effective strategies towards competitors in less time
2) it improves decision making capabilities
3) it allows businesses to improve customer service

4. **Define Apache Hadoop**

Ans: It is open source framework written in java which is used to store, analyze and process huge amounts of data in distributed environment across clusters of computers in an efficient manner. It provides capability to process distributed data using simplified programming model. It is used by Google, facebook, yahoo, youtube, twitter etc. It is developed by Doug Cutting at Yahoo in 2006 which is inspired from Google File System and Google Map Reduce algorithm. It is file system provided by linux to store the data.

5. **List the characteristics of Hadoop**

Ans:

1) Robust
2) Simple
3) Scalable
4) Cost effective
5) Fault tolerance(as blocks are replicated in serveral nodes. If any one of the node fails then data can be retrieved from the remaining nodes)

6.**List the differences between Hadoop and RDBMS**

Ans:

Hadoop	RDBMS
Handles massive amount of data	Handles limited data
It deals with structured and unstructured data	It deals with structured data
It is cost effective compared to RDBMS	It is not cost effective compared to Hadoop
Schema is required on read	Schema is required on write
Reads and writes are fast	Reads are fast

7. **Describe core components of Hadoop ecosystem**

Ans: Hadoop ecosystem consists of four major components. They are

1. Hadoop common-contains set of predefined utilities and libraries that can be used by other modules within hadoop ecosystem
2. Hadoop Distributed File system-stores very large files running on cluster of commodity hardware. It creates several replicas of the data block to be distributed across different clusters for reliable and quick data access

3. Hadoop YARN(Yet Another Resource Negotiator)-framework responsible for providing the computational resources (e.g., CPUs, memory, etc.) needed for application execution.
4. Hadoop Mapreduce- It is java based system created by Google where the actual data from the HDFS store gets processed efficiently. It is based on YARN architecture. It consists of two major tasks
a) Map
b) Reduce

8. **Define HDFS**

Ans It is the world's most reliable storage system and stores very large files running on a cluster of commodity hardware. It stores data reliably even in the case of hardware failure. It provides high throughput by providing the data access in parallel.

9. **Define Block**

Ans: HDFS splits huge files into smaller chunks called blocks which are smallest unit of data in file system. The default size of block is 64MB(hadoop 1.x version) and 128MB (Hadoop 2.x version)

10. **Define Namenode**

Ans: It is also known as *master node*. It stores *meta-data* i.e. file name,location of file, data blocks, replicas and other details. This meta-data is available in memory in the master for faster retrieval of data. It maintains and manages the slave nodes, and assigns tasks to them.

11. **Define datanode**

Ans: It is also known as *slave node* where it stores actual data in HDFS. It performs **read and write operations** as per the request of the client.

12. **Define secondary namenode**

Ans: As the NameNode is the single point of failure in HDFS, if NameNode fails entire HDFS file system is lost. So in order to overcome this, Hadoop implemented Secondary NameNode whose main function is to store a copy of **FsImage** file and **edits** log file. It **periodically applies edits log records to FsImage file and refreshes the edits log** and sends this updated FsImage file to NameNode so that NameNode doesn't need to re-apply the EditLog records during its start up process. Thus Secondary NameNode makes NameNode start up process fast.

13. **Define Replication factor?**

Ans:

It is defined as the number of copies of each block of file. The default replication factor is 3 which are again configurable.

14. **What is default port number for Namenode?**

Ans: The default port number for namenode is 50070

15. **List different operational modes of configuring Hadoop cluster**

Ans:

Hadoop can be run in one of the three supported modes

1) Local(Standalone) mode-By default, Hadoop is configured to run in a single-node, non-distributed mode, as a single Java process. This is useful for debugging. The usage of this mode is very limited and it can be only used for experimentation.

2) Pseudo-Distributed mode-Hadoop is run on a single-node in a pseudo distributed mode where each Hadoop daemon(Namenode, Datanode, Secondary Namenode, Jobtracker, Tassktracker) runs in a separate Java process. In Local mode, Hadoop runs as a single Java process

3) Fully distributed mode-In this mode, all daemons are executed in a separate nodes forming a multi node cluster. This setup offers true distributed computing capability and offers built-in reliability, scalability and fault tolerance

16. **Since the data is replicated thrice in HDFS, does it mean that any calculation done on one node will also be replicated on the other two nodes?**

Ans: No calculation will be done only on original data. Namenode knows which node has exactly particular data.If one of datanodes is not responding it is assumed to be failed. Only then required calculation will be done on the second replica node

17. W**hat is heartbeat in HDFS?**

Ans: It is signal indicating that it is alive. A datanode sends heartbeat to namenode and if namenode does not receive heartbeat then namenode assumes that there is problem in datanode.

18. **What is communication channel between client and namenode?**

Ans: SSH(Secure shell) is communication channel between client and namenode

19. **Can hadoop handle streaming data?**

Ans:It is possible to handle large scale streaming of data through Apache kafka, Apache flume & Apache spark.

20. **List the building blocks of Hadoop or daemons of Hadoop**

Ans:

Namenode, Datanode,secondary namenode,jobtracker and task tracker

WEEK 5

Aim:

Implement the following file management tasks in Hadoop:

a) Adding files and directories

b) Retrieving files

c) Deleting files

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

Program:

The most common file management tasks in Hadoop includes:

- Adding files and directories to HDFS
- Retrieving files from HDFS to local filesystem
- Deleting files from HDFS

Hadoop file commands take the following form:

hadoop fs -cmd

Where cmd is the specific file command and <args> is a variable number of arguments. The command cmd is usually named after the corresponding Unix equivalent. For example, the command for listing files is ls as in Unix.

a) Adding Files and Directories to HDFS

Creating Directory in HDFS

$ **hadoop fs -mkdir foldername (syntax)**

$ hadoop fs -mkdir cse

Hadoop's mkdir command automatically creates parent directories if they don't already exist. Now that we have a working directory, we can put a file into it.

Adding File in HDFS

Create some text file on your local filesystem called example.txt. The **Hadoop command put** is used to copy files from the local system into HDFS.

Syntax

$hadoop fs -put source destination

The command above is equivalent to:

$ hadoop fs -put example.txt cse

b) Retrieving files from HDFS

Hadoop command get gets the data from HDFS to local filesystem.

Syntax

$hadoop fs -get source destination

$hadoop fs -get cse/example.txt Desktop (copies data from HDFS(cse/example.txt) to local file system(Desktop)

c) Deleting Files

Hadoop command rm removes the files from HDFS.

$hadoop fs -rm cse/example.txt (removes file from HDFS)

Viva Questions

1. **What is command to list all the files in directory of HDFS?**

Ans: $hadoop fs -ls cse (let cse be directory)

2. **What is command to copy data from local file system to HDFS using copyFromLocal?**

Ans: $hadoop fs -copyFromLocal <source> <destination> (it is similar to put command except that source is restricted to local file reference)

3. **What is command to copy data from HDFS to local file system using copyToLocal?**

Ans: $hadoop fs -copyToLocal <source> <destination> (it is similar to get command except that destination is restricted to local file reference)

4. **What is command to display contents of file in HDFS?**

Ans: $hadoop fs -cat cse/A.java

5. **What is command to display last 1KB of particular file in HDFS?**

Ans: $hadoop fs -tail <filename>

6. **What is command used to change replication factor for files or directories in HDFS?**

Ans : $hadoop fs -setrep -w <value> <filename or directory> (-w flag requests that the command waits for the replication process to get completed.)

7. **What is command to show disk usage in bytes for all files/directories of path in HDFS?**

Ans: $hadoop fs -du <path>

8. **What is command used to display free space in HDFS?**

Ans: $hadoop fs -df -h

9. **What is command used to create new file at the path containing the current time as a timestamp in HDFS?**

Ans: $hadoop fs -touchz <path>

10. **What is command used to take source file from HDFS and outputs the given file in text format?**

Ans: $hadoop fs -text <source>

11. **What is command used to display information about the path?**

Ans: $hadoop fs -stat <path>

12. **What is command used to apply permissions to file in HDFS?**

Ans: $hadoop fs -chmod <value> <file or directory>(for eg : value=777 where owner,group and others can read,write & execute read=4,write=2,execute=1)

13. **What is command used to counts the number of directories, number of files present and bytes under the path?**

Ans: $hadoop fs -count <path>

Sample Output

1 3 1050120 cse (1 is directory, 3 is no of files , 1050120 are no of bytes & cse is directory)

14. **What is command used to get usage of particular command?**

Ans: $hadoop fs -usage <commandname>

15. **What is command used to empty trash?**

Ans: $hadoop fs -expunge

WEEK 6

Aim: Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm

Program:

Source Code

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;// provides access to configuration parameters
import org.apache.hadoop.fs.Path;// Path class names a file or directory in a HDFS
import org.apache.hadoop.io.IntWritable;// primtive Writable Wrapper class for integers.
import org.apache.hadoop.io.Text;// This class stores text and provides methods to serialize, deserialize, and compare texts at byte level
import org.apache.hadoop.mapreduce.Job;//Job class allows the user to configure the job, submit it, control its execution, and query the state
//The Hadoop Map-Reduce framework spawns one map task for each InputSplit generated by the InputFormat for the job
```

import org.apache.hadoop.mapreduce.Mapper;//Maps input key/value pairs to a set of intermediate key/value pairs.
import org.apache.hadoop.mapreduce.Reducer;//Reduces a set of intermediate values which share a key to a smaller set of values.
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;//A base class for file-based InputFormats.
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; //A base class for file-based OutputFormats.

```
public class WordCount {

 public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
 }

 public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
```

```
    }
  }
 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Usage

$export
CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.1.jar:$HADOOP_HOME/share/hadoop/coomon/hadoop-common-2.8.1.jar"

Compile WordCount.java and create a jar:

$ javac WordCount*.java
$ jar -cvf wc.jar WordCount*.class

Assuming that:

- **input - input directory in HDFS**
- **output - output directory in HDFS**

Sample text-files as input:

$ hadoop fs -put file1.txt input

$ bin/hadoop fs -cat input/file1.txt
Hai welcome Hai cse

Run the application:

$ hadoop jar wc.jar WordCount input/file1.txt output

Output:

$ bin/hadoop fs -cat output/part-r-00000`
cse 1
Hai 2
welcome 1

Viva Questions

1. **Define Map Reduce?**

Ans: It is programming framework to process large datasets in paralle across thousands of servers in hadoop cluster. It consists of two major tasks

a) map (converts given set of data into <key,value> pairs.)

b) reduce(the given set of <key,value> pairs from map job are combined into several smaller set of tuples)

2. **What is purpose of Driver class in Map reduce?**

Ans: It provides configuration parameters for processing job

3. **List the features of Context object?**

Ans:

1) It is used to report progress and provides application level status updates.

2) It has configuration details for the job and also interfaces that helps it to generate output

3) It is used to help mapper/reducer to interact with other hadoop systems.

4. **What is the significance of setup() of Mapper/Reducer class?**

Ans: It is called once at the beginning of the task which is used for configuring parameters like data size, distributed cache, heap size etc.

5. **What is the significance of map() of Mapper class?**

Ans: It is heart of Mapper class which is called once for each key/value pair in the input split.

6. **What is the significance of reduce() of Reducer class?**

Ans: It is heart of Reducer class which is called once for each key in the associated reduce task.

7. **Define Shuffling in Mapreduce?**

Ans: It is process of transferring sorted output from each Mapper to the Reducer.

8. **What main configuration parameters are specified in Map-Reduce?**

Ans:

1) input location of job in HDFS

2) output location of job in HDFS

3) input and output format

4) classes containing map and reduce functions respectively

5) .jar file for mapper, reducer and driver class

9. **Define Job Tracker**

Ans: It is daemon that runs on namenode for submitting and tracking map reduce jobs in Hadoop.

10. **Define Task Tracker**

Ans: It is a daemon that runs on data nodes which manages execution of individual tasks

11. **Whenever client submits hadoop job, who receives it?**

Ans:

Namenode receives hadoop job and provides block information.Job Tracker takes care of resource allocation of hadoop job to ensure timely completion.

12. **What is process to change files at arbitrary locations in HDFS?**

Ans: HDFS does not support modifications at arbitrary locations in file and writes to file in HDFS are always made at the end of file.

13. **What happens when user submits hadoop job when namenode is down-does job get in to hold or does it fail?**

Ans: Hadoop job fails when namenode is down.

14. **What happens when user submits hadoop job when namenode is down-does job get in to hold or does it fail?**

Ans: Hadoop job fails when jobtracker is down.

15. **What is command to list all running jobs in hadoop?**

Ans: $hadoop job -list.

16. **What is command to stop a running job?**

Ans: $hadoop job -kill <jobid>

17. **What is default port number for Job Tracker and TaskTracker?**

Ans: The default port number for JobTracker is 50030 and TaskTracker is 50060

18. List the different modes of execution of map-reduce programs?

Ans:

- Local(default)
- classic
- yarn(Yet another resource negotiator

19. **List the important classes present in org.apache.hadoop.mapreduce package?**

Ans:

The important classes present in org.apache.hadoop.mapreduce package are as follows

1) InputFormat
2) InputSplit
3) Job
4) RecordReader
5) Mapper
6) Partitioner
7) Reducer
8) RecordWriter
9) OutputFormat

20. List **the important classes and interfaces present in org.apache.hadoop.mapred package?**

Ans:

The important interfaces present in org.apache.hadoop.mapred package are as follows

1) InputFormat- It describes the input specification for map-reduce job.
2) InputSplit- It represents the data to be processed by an individual Mapper.
3) RecordReader- It reads <key,value> pairs from InputSplit.
4) Mapper- It maps input key/value pairs to a set of intermediate key/value pairs.
5) Partitioner-It controls the partitioning of the keys of the intermediate map-outputs
6) Reducer-It reduces a set of intermediate values which share a key to a smaller set of values.
7) OutputCollector- collects the <key,value> pairs output by Mappers and Reducers.
8) RecordWriter- It writes the output <key, value> pairs to an output file.
9) OutputFormat-It describes the output-specification for a Map-Reduce job.

The important classes present in org.apache.hadoop.mapred package are as follows

1) JobClient- primary interface for the user-job to interact with the cluster and provides facilities to submit jobs, track their progress, access component-tasks' reports/logs, get the Map-Reduce cluster status information etc.

2) JobConf- A map/reduce job configuration. It is primary interface for a user to describe a map-reduce job to the Hadoop framework for execution
3) FileInputFormat- base class for file based InputFormat
4) FileOutputFormat- base class for OutputFormat.

WEEK 7

Aim:

Write a Map Reduce program that mines weather data.

Weather sensors collecting data every hour at many locations across the globe gather a

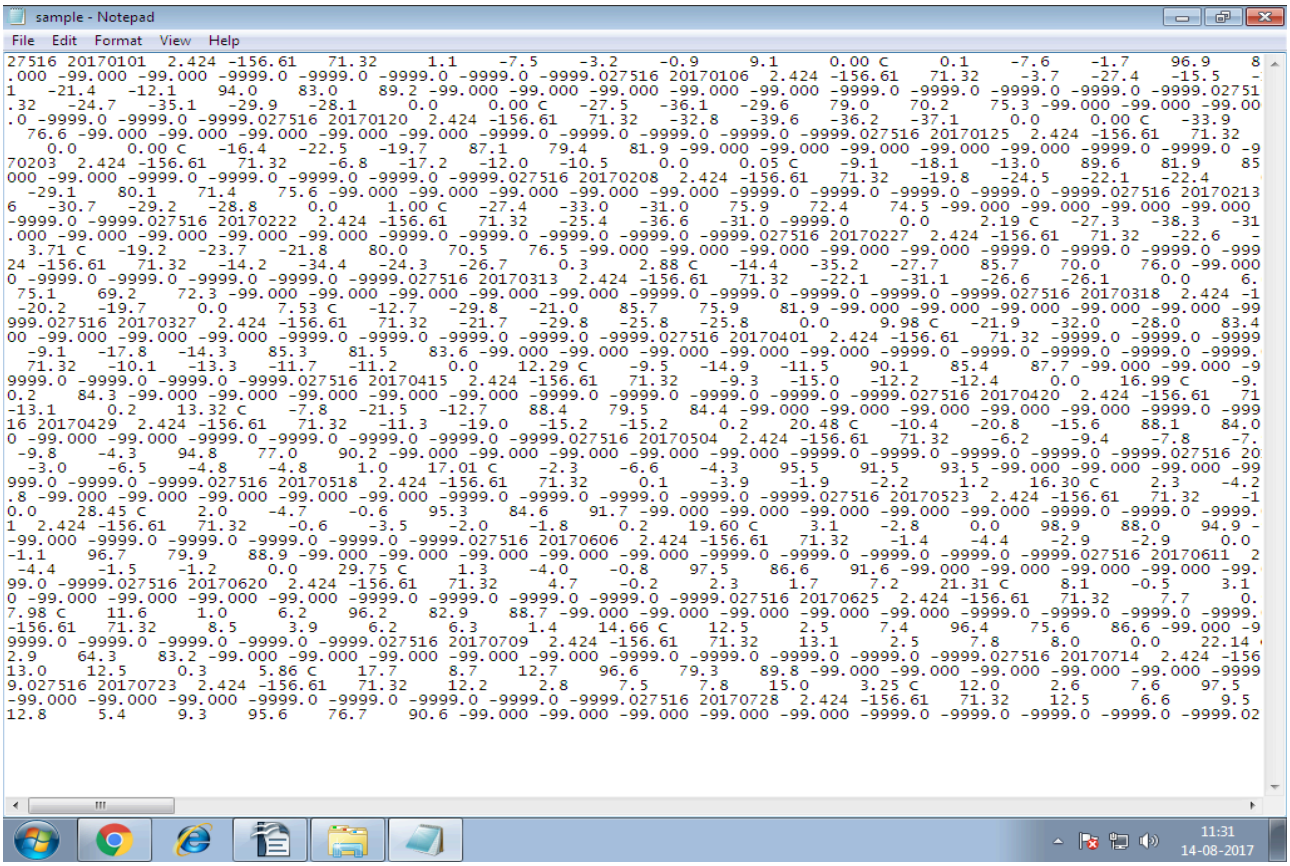
Large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented

Program:

National Climatic Data Center (**NCDC**) is responsible for preserving, monitoring, assessing, and providing public access to weather data.
NCDC provides access to daily data from the U.S. Climate Reference Network / U.S. Regional Climate Reference Network (USCRN/USRCRN) via anonymous ftp at

ftp://ftp.ncdc.noaa.gov/pub/data/uscrn/products/daily01

The weather dataset(2017) collected shown below



Readme.txt file is shown below

ll daily data are calculated over the station's 24-hour LST day.

Field#	Name	Units
1	WBANNO	XXXXX
2	LST_DATE	YYYYMMDD
3	CRX_VN	XXXXXX
4	LONGITUDE	Decimal_degrees
5	LATITUDE	Decimal_degrees
6	T_DAILY_MAX	Celsius
7	T_DAILY_MIN	Celsius
8	T_DAILY_MEAN	Celsius
9	T_DAILY_AVG	Celsius
10	P_DAILY_CALC	mm
11	SOLARAD_DAILY	MJ/m^2
12	SUR_TEMP_DAILY_TYPE	X
13	SUR_TEMP_DAILY_MAX	Celsius
14	SUR_TEMP_DAILY_MIN	Celsius
15	SUR_TEMP_DAILY_AVG	Celsius
16	RH_DAILY_MAX	%
17	RH_DAILY_MIN	%
18	RH_DAILY_AVG	%
19	SOIL_MOISTURE_5_DAILY	m^3/m^3
20	SOIL_MOISTURE_10_DAILY	m^3/m^3
21	SOIL_MOISTURE_20_DAILY	m^3/m^3
22	SOIL_MOISTURE_50_DAILY	m^3/m^3
23	SOIL_MOISTURE_100_DAILY	m^3/m^3
24	SOIL_TEMP_5_DAILY	Celsius
25	SOIL_TEMP_10_DAILY	Celsius
26	SOIL_TEMP_20_DAILY	Celsius
27	SOIL_TEMP_50_DAILY	Celsius
28	SOIL_TEMP_100_DAILY	Celsius

1 WBANNO [5 chars] cols 1 -- 5
The station WBAN number.

2 LST_DATE [8 chars] cols 7 -- 14
The Local Standard Time (LST) date of the observation.

3 CRX_VN [6 chars] cols 16 -- 21
The version number of the station datalogger program that was in effect at the time of the observation. Note: This field should be treated as text (i.e. string).

4 LONGITUDE [7 chars] cols 23 -- 29
Station longitude, using WGS-84.

5 LATITUDE [7 chars] cols 31 -- 37
Station latitude, using WGS-84.

6 T_DAILY_MAX [7 chars] cols 39 -- 45
Maximum air temperature, in degrees C. See Note F.

7 T_DAILY_MIN [7 chars] cols 47 -- 53
Minimum air temperature, in degrees C. See Note F.

8 T_DAILY_MEAN [7 chars] cols 55 -- 61
Mean air temperature, in degrees C, calculated using the typical historical approach: (T_DAILY_MAX + T_DAILY_MIN) / 2. See Note F.

9 T_DAILY_AVG [7 chars] cols 63 -- 69
Average air temperature, in degrees C. See Note F.

10 P_DAILY_CALC [7 chars] cols 71 -- 77
Total amount of precipitation, in mm. See Note F.

11 SOLARAD_DAILY [8 chars] cols 79 -- 86
Total solar energy, in MJ/meter^2, calculated from the hourly average global solar radiation rates and converted to energy by integrating over time.

12 SUR_TEMP_DAILY_TYPE [1 chars] cols 88 -- 88
Type of infrared surface temperature measurement. 'R' denotes raw measurements, 'C' denotes corrected measurements, and 'U' indicates unknown/missing. See Note G.

13 SUR_TEMP_DAILY_MAX [7 chars] cols 90 -- 96
Maximum infrared surface temperature, in degrees C.

14 SUR_TEMP_DAILY_MIN [7 chars] cols 98 -- 104
Minimum infrared surface temperature, in degrees C.

15 SUR_TEMP_DAILY_AVG [7 chars] cols 106 -- 112
Average infrared surface temperature, in degrees C.

16 RH_DAILY_MAX [7 chars] cols 114 -- 120
Maximum relative humidity, in %. See Notes H and I.

17 RH_DAILY_MIN [7 chars] cols 122 -- 128
Minimum relative humidity, in %. See Notes H and I.

18 RH_DAILY_AVG [7 chars] cols 130 -- 136
Average relative humidity, in %. See Notes H and I.

19 SOIL_MOISTURE_5_DAILY [7 chars] cols 138 -- 144
Average soil moisture, in fractional volumetric water content (m^3/m^3), at 5 cm below the surface. See Notes I and J.

20 SOIL_MOISTURE_10_DAILY [7 chars] cols 146 -- 152
Average soil moisture, in fractional volumetric water content (m^3/m^3), at 10 cm below the surface. See Notes I and J.

21 SOIL_MOISTURE_20_DAILY [7 chars] cols 154 -- 160
Average soil moisture, in fractional volumetric water content (m^3/m^3), at 20 cm below the surface. See Notes I and J.

22 SOIL_MOISTURE_50_DAILY [7 chars] cols 162 -- 168
Average soil moisture, in fractional volumetric water content (m^3/m^3), at 50 cm below the surface. See Notes I and J.

23 SOIL_MOISTURE_100_DAILY [7 chars] cols 170 -- 176
Average soil moisture, in fractional volumetric water content (m^3/m^3), at 100 cm below the surface. See Notes I and J.

24 SOIL_TEMP_5_DAILY [7 chars] cols 178 -- 184
Average soil temperature, in degrees C, at 5 cm below the surface. See Notes I and J.

25 SOIL_TEMP_10_DAILY [7 chars] cols 186 -- 192
Average soil temperature, in degrees C, at 10 cm below the surface. See Notes I and J.

26 SOIL_TEMP_20_DAILY [7 chars] cols 194 -- 200
Average soil temperature, in degrees C, at 20 cm below the surface. See Notes I and J.

27 SOIL_TEMP_50_DAILY [7 chars] cols 202 -- 208
Average soil temperature, in degrees C, at 50 cm below the surface. See Notes I and J.

28 SOIL_TEMP_100_DAILY [7 chars] cols 210 -- 216
Average soil temperature, in degrees C, at 100 cm below the surface. See Notes I and J.

Program

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {


	//Mapper

	/**
	*MaxTemperatureMapper class is static and extends Mapper abstract class
	having four hadoop generics type LongWritable, Text, Text, Text.
	*/


	public static class MaxTemperatureMapper extends
			Mapper<LongWritable, Text, Text, Text> {

		/**
		* @method map
		* This method takes the input as text data type.
		* Now leaving the first five tokens,it takes 6th token is taken as temp_max and
		* 7th token is taken as temp_min. Now temp_max > 35 and temp_min < 10 are passed
to the reducer.
		*/

		@Override
		public void map(LongWritable arg0, Text Value, Context context)
				throws IOException, InterruptedException {

		//Converting the record (single line) to String and storing it in a String variable line

			String line = Value.toString();
```

```
            //Checking if the line is not empty

                if (!(line.length() == 0)) {

                    //date

                    String date = line.substring(6, 14);

                    //maximum temperature

                    float temp_Max = Float
                            .parseFloat(line.substring(39, 45).trim());

                    //minimum temperature

                    float temp_Min = Float
                            .parseFloat(line.substring(47, 53).trim());

                    //if maximum temperature is greater than 35 , its a hot day

                    if (temp_Max > 35.0) {
                        // Hot day
                        context.write(new Text("Hot Day " + date),
                                new Text(String.valueOf(temp_Max)));
                    }

                    //if minimum temperature is less than 10 , its a cold day

                    if (temp_Min < 10) {
                        // Cold day
                        context.write(new Text("Cold Day " + date),
                                new Text(String.valueOf(temp_Min)));
                    }
                }
            }

    }

//Reducer

    /**
    *MaxTemperatureReducer class is static and extends Reducer abstract class
```

```
having four hadoop generics type Text, Text, Text, Text.
*/

public static class MaxTemperatureReducer extends
                Reducer<Text, Text, Text, Text> {

        /**
        * @method reduce
        * This method takes the input as key and list of values pair from mapper, it does
aggregation
        * based on keys and produces the final context.
        */

        public void reduce(Text Key, Iterator<Text> Values, Context context)
                        throws IOException, InterruptedException {


                //putting all the values in temperature variable of type String

                String temperature = Values.next().toString();
                context.write(Key, new Text(temperature));
        }

}

/**
* @method main
* This method is used for setting all the configuration properties.
* It acts as a driver for map reduce code.
*/

public static void main(String[] args) throws Exception {

        //reads the default configuration of cluster from the configuration xml files
        Configuration conf = new Configuration();

        //Initializing the job with the default configuration of the cluster
        Job job = new Job(conf, "weather example");

        //Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);
```

```
                //Key type coming out of mapper
                job.setMapOutputKeyClass(Text.class);

                //value type coming out of mapper
                job.setMapOutputValueClass(Text.class);

                //Defining the mapper class name
                job.setMapperClass(MaxTemperatureMapper.class);

                //Defining the reducer class name
                job.setReducerClass(MaxTemperatureReducer.class);

                //Defining input Format class which is responsible to parse the dataset into a key value
pair
                job.setInputFormatClass(TextInputFormat.class);

                //Defining output Format class which is responsible to parse the dataset into a key value
pair
                job.setOutputFormatClass(TextOutputFormat.class);

                //setting the second argument as a path in a path variable
                Path OutputPath = new Path(args[1]);

                //Configuring the input path from the filesystem into the job
                FileInputFormat.addInputPath(job, new Path(args[0]));

                //Configuring the output path from the filesystem into the job
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                //deleting the context path automatically from hdfs so that we don't have delete it
explicitly
                OutputPath.getFileSystem(conf).delete(OutputPath);

                //exiting the job only if the flag value becomes false
                System.exit(job.waitForCompletion(true) ? 0 : 1);

        }
}
```

Usage

$export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.1.jar:$HADOOP_HOME/share/hadoop/coomon/hadoop-common-2.8.1.jar"

Compile MyMaxMin.java and create a jar:

$ javac MyMaxMin*.java
$ jar -cvf maxmin.jar MyMaxMin*.class

Assuming that:

- **input - input directory in HDFS**
- **output - output directory in HDFS**

Sample text-files as input:

$ hadoop fs -put sample.txt input

Run the application:

$ hadoop jar maxmin.jar MyMaxMin input/sample.txt output

Output:

$ hadoop fs -cat output/part-r-00000

Cold Day 20170101	-7.5
Cold Day 20170102	-9.7
Cold Day 20170103	-24.1
Cold Day 20170104	-21.4
Cold Day 20170105	-22.9
Cold Day 20170106	-27.4
Cold Day 20170107	-3.8
Cold Day 20170108	-9.0
Cold Day 20170109	-13.8
Cold Day 20170110	-20.3
Cold Day 20170111	-22.8
Cold Day 20170112	-23.4
Cold Day 20170113	-27.9
Cold Day 20170114	-28.0
Cold Day 20170115	-35.1
Cold Day 20170116	-37.2
Cold Day 20170117	-30.3
Cold Day 20170118	-32.2
Cold Day 20170119	-39.4
Cold Day 20170120	-39.6
Cold Day 20170121	-33.9
Cold Day 20170122	-36.0
Cold Day 20170123	-37.2

Cold Day 20170124 -28.2
Cold Day 20170125 -27.8
Cold Day 20170126 -24.5
Cold Day 20170127 -21.0
Cold Day 20170128 -20.0
Cold Day 20170129 -22.4
Cold Day 20170130 -18.3
Cold Day 20170131 -11.1
Cold Day 20170201 -23.6
Cold Day 20170202 -14.1
Cold Day 20170203 -17.2
Cold Day 20170204 -21.1
Cold Day 20170205 -17.4
Cold Day 20170206 -15.5
Cold Day 20170207 -22.8
Cold Day 20170208 -24.5
Cold Day 20170209 -30.7
Cold Day 20170210 -25.2
Cold Day 20170211 -34.1
Cold Day 20170212 -34.1
Cold Day 20170213 -26.0
Cold Day 20170214 -22.5
Cold Day 20170215 -28.6
Cold Day 20170216 -29.9
Cold Day 20170217 -30.7
Cold Day 20170218 -32.4
Cold Day 20170219 -28.6
Cold Day 20170220 -25.7
Cold Day 20170221 -34.8
Cold Day 20170222 -36.6
Cold Day 20170223 -33.7
Cold Day 20170224 -26.1
Cold Day 20170225 -28.6
Cold Day 20170226 -22.9
Cold Day 20170227 -29.7
Cold Day 20170228 -31.5
Cold Day 20170301 -31.9
Cold Day 20170302 -25.0
Cold Day 20170303 -22.1
Cold Day 20170304 -30.9

Cold Day 20170305 -34.0
Cold Day 20170306 -29.1
Cold Day 20170307 -31.2
Cold Day 20170308 -34.4
Cold Day 20170309 -14.3
Cold Day 20170310 -8.8
Cold Day 20170311 -15.7
Cold Day 20170312 -28.2
Cold Day 20170313 -31.1
Cold Day 20170314 -26.5
Cold Day 20170315 -33.6
Cold Day 20170316 -36.3
Cold Day 20170317 -37.3
Cold Day 20170318 -35.5
Cold Day 20170319 -37.6
Cold Day 20170320 -30.9
Cold Day 20170321 -27.1
Cold Day 20170322 -27.2
Cold Day 20170323 -25.0
Cold Day 20170324 -22.7
Cold Day 20170325 -24.2
Cold Day 20170326 -29.7
Cold Day 20170327 -29.8
Cold Day 20170328 -30.5
Cold Day 20170329 -27.8
Cold Day 20170330 -28.1
Cold Day 20170403 -30.6
Cold Day 20170404 -23.5
Cold Day 20170405 -16.0
Cold Day 20170406 -16.7
Cold Day 20170407 -16.3
Cold Day 20170408 -19.8
Cold Day 20170409 -15.4
Cold Day 20170410 -13.3
Cold Day 20170411 -12.4
Cold Day 20170412 -13.2
Cold Day 20170413 -14.7
Cold Day 20170414 -16.4
Cold Day 20170415 -15.0
Cold Day 20170416 -15.0

Cold Day 20170417	-17.1
Cold Day 20170418	-19.6
Cold Day 20170419	-21.4
Cold Day 20170420	-22.9
Cold Day 20170421	-22.2
Cold Day 20170422	-19.6
Cold Day 20170424	-19.2
Cold Day 20170425	-14.5
Cold Day 20170426	-15.5
Cold Day 20170427	-20.6
Cold Day 20170428	-21.6
Cold Day 20170429	-19.0
Cold Day 20170430	-20.3
Cold Day 20170501	-16.7
Cold Day 20170502	-12.4
Cold Day 20170503	-11.6
Cold Day 20170504	-9.4
Cold Day 20170505	-9.4
Cold Day 20170506	-8.0
Cold Day 20170507	-9.8
Cold Day 20170508	-8.5
Cold Day 20170509	-7.0
Cold Day 20170510	-7.5
Cold Day 20170511	-8.8
Cold Day 20170512	-11.2
Cold Day 20170513	-6.5
Cold Day 20170514	-5.7
Cold Day 20170515	-4.1
Cold Day 20170516	-3.3
Cold Day 20170517	-4.2
Cold Day 20170518	-3.9
Cold Day 20170519	-4.4
Cold Day 20170520	-6.3
Cold Day 20170521	-7.1
Cold Day 20170522	-6.6
Cold Day 20170523	-4.8
Cold Day 20170524	-2.1
Cold Day 20170525	-1.6
Cold Day 20170526	-2.0
Cold Day 20170527	-4.1

Cold Day 20170528	-3.7
Cold Day 20170529	-1.0
Cold Day 20170530	-1.6
Cold Day 20170531	-1.6
Cold Day 20170601	-3.5
Cold Day 20170602	-5.5
Cold Day 20170603	-4.7
Cold Day 20170604	-4.7
Cold Day 20170605	-5.4
Cold Day 20170606	-4.4
Cold Day 20170607	-4.9
Cold Day 20170608	-3.9
Cold Day 20170609	-5.3
Cold Day 20170610	-3.5
Cold Day 20170611	-3.0
Cold Day 20170612	-2.9
Cold Day 20170613	-4.5
Cold Day 20170614	-5.7
Cold Day 20170615	-4.4
Cold Day 20170616	-2.3
Cold Day 20170617	-0.5
Cold Day 20170618	-1.3
Cold Day 20170619	-1.4
Cold Day 20170620	-0.2
Cold Day 20170621	0.7
Cold Day 20170622	2.3
Cold Day 20170623	4.1
Cold Day 20170624	5.5
Cold Day 20170625	0.9
Cold Day 20170626	0.9
Cold Day 20170627	1.3
Cold Day 20170628	0.8
Cold Day 20170629	1.1
Cold Day 20170630	-0.3
Cold Day 20170701	-0.6
Cold Day 20170702	0.3
Cold Day 20170703	0.4
Cold Day 20170704	3.9
Cold Day 20170705	7.3
Cold Day 20170706	8.7

Cold Day 20170707	7.4
Cold Day 20170708	5.2
Cold Day 20170709	2.5
Cold Day 20170710	2.5
Cold Day 20170711	3.1
Cold Day 20170712	3.1
Cold Day 20170713	4.1
Cold Day 20170714	5.8
Cold Day 20170715	3.9
Cold Day 20170716	3.9
Cold Day 20170717	5.8
Cold Day 20170718	8.7
Cold Day 20170719	3.5
Cold Day 20170720	1.3
Cold Day 20170721	2.0
Cold Day 20170722	2.1
Cold Day 20170723	2.8
Cold Day 20170724	2.2
Cold Day 20170725	0.4
Cold Day 20170726	0.2
Cold Day 20170727	3.2
Cold Day 20170728	6.6
Cold Day 20170729	5.1
Cold Day 20170730	3.3
Cold Day 20170731	6.3
Cold Day 20170801	5.8
Cold Day 20170802	3.5
Cold Day 20170803	2.4
Hot Day 20170216	9999.0
Hot Day 20170331	9999.0
Hot Day 20170401	9999.0
Hot Day 20170402	9999.0
Hot Day 20170423	9999.0
Hot Day 20170515	9999.0
Hot Day 20170516	9999.0
Hot Day 20170520	9999.0

Viva Questions

1. **What is default framework for running map reduce programs?**

Ans:local is default framework which can be set using mapreduce.framework.name property for running map reduce programs

2. **What are the parameters for Mapper and Reducer class?**

Ans:**The parameters for Mapper class are**

KEYIN,VALUEIN is input parameter for Mapper class and KEYOUT,VALUEOUT are output parameter which is input for Reducer.

The parameters for Reducer class are

KEYIN,VALUEIN is input parameter for Reducer which are coming from Mapper class. KEYOUT,VALUEOUT are output parameter which will go in files of final Output directory in HDFS

3. **Explain RecordReader in Map Reduce**

Ans: It breaks the data into <key,value> pairs for input to the Mapper

4. **What is significance of Combiner in Map Reduce?**

Ans:It is used to summarize the map output records with the same key and used between Mapper class & Reducer class to reduce the volume of data transfer between them.

5. **What is significance of Partitioner in Map Reduce**?

Ans:It partitions <key,value> pairs using user defined condition and number of partitioners is equal to number of reducer tasks for the job i.e.,partitioner will divide the data according to the number of reducers.

6. **What is the purpose of getSplits() of InputFormat interface in mapred package?**

Ans: It logically split the set of input files for the job.

7. **What is the purpose of getMemoryForMapTask() of JobConf class in mapred package?**

Ans: It is used to get the memory required to run the map task of the job in MB

8. **What is the purpose of getInstance(Configuration) of Job class in mapreduce package?**

Ans: It is used to create new job with the given configuration.

9. **What is the purpose of setPriority() of Job class in mapreduce package?**

Ans: It is used to set the priority of running job.

WEEK 8

Aim: Implement Matrix Multiplication with Hadoop Map Reduce

Program:

Input Dataset

The input data set(mat.txt) containing these matrices are represented as follows.

a, 0, 0, 63 (matrix1, row,col,value)
a, 0, 1, 45
a, 0, 2, 93
a, 0, 3, 32
a, 0, 4, 49
a, 1, 0, 33
a, 1, 3, 26
a, 1, 4, 95
a, 2, 0, 25
a, 2, 1, 11
a, 2, 3, 60
a, 2, 4, 89
a, 3, 0, 24
a, 3, 1, 79
a, 3, 2, 24
a, 3, 3, 47
a, 3, 4, 18
a, 4, 0, 7
a, 4, 1, 98
a, 4, 2, 96
a, 4, 3, 27
b, 0, 0, 63
b, 0, 1, 18
b, 0, 2, 89
b, 0, 3, 28
b, 0, 4, 39
b, 1, 0, 59
b, 1, 1, 76
b, 1, 2, 34
b, 1, 3, 12
b, 1, 4, 6

b, 2, 0, 30
b, 2, 1, 52
b, 2, 2, 49
b, 2, 3, 3
b, 2, 4, 95
b, 3, 0, 77
b, 3, 1, 75
b, 3, 2, 85
b, 4, 1, 46
b, 4, 2, 33
b, 4, 3, 69
b, 4, 4, 88

Implementation

In this implementation for ease of understanding, the dimension of matrix is given as (5 * 5).

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Matrix
{
public static class Matrix_Mapper extends Mapper<LongWritable,Text,Text,Text>
{
/**
* Map function will collect/ group cell values required for
* calculating the output.
* @param key is ignored. Its just the byte offset
```

```
* @param value is a single line. (a, 0, 0, 63) (matrix name, row, column, value)
*
*/
@Override
protected void map(LongWritable key, Text value,Context context)
throws IOException, InterruptedException
{
String line = value.toString();
String[] entry = line.split(",");
String sKey = "";
String mat = entry[0].trim();
String row, col;
Configuration conf = context.getConfiguration();
String dimension = conf.get("dimension");
int dim = Integer.parseInt(dimension);
if(mat.matches("a"))
{
for (int i =0; i < dim ; i++) // hard coding matrix size 5
{
row = entry[1].trim(); // rowid
sKey = row+i;
context.write(new Text(sKey),value);
}
}
if(mat.matches("b"))
{
for (int i =0; i < dim ; i++)
{
col = entry[2].trim(); // colid
sKey = i+col;
context.write(new Text(sKey),value);
}
}
}
}
public static class Matrix_Reducer extends Reducer<Text, Text, Text, IntWritable>
{
```

```
/**
* Reducer do the actual matrix multiplication.
* @param key is the cell unique cell dimension (00) represents cell 0,0
* @value values required to calculate matrix multiplication result of that cell.
*/
@Override
protected void reduce(Text key, Iterable<Text> values,Context context)
throws IOException, InterruptedException
{
Configuration conf = context.getConfiguration();
String dimension = conf.get("dimension");
int dim = Integer.parseInt(dimension);
//System.out.println("Dimension from Reducer = " + dimension);
int[] row = new int[dim]; // hard coding as 5 X 5 matrix
int[] col = new int[dim];
for(Text val : values)
{
String[] entries = val.toString().split(",");
if(entries[0].matches("a"))
{
int index = Integer.parseInt(entries[2].trim());
row[index] = Integer.parseInt(entries[3].trim());
}

if(entries[0].matches("b"))
{
int index = Integer.parseInt(entries[1].trim());
col[index] = Integer.parseInt(entries[3].trim());
}
}
// Let us do matrix multiplication now..
int total = 0;
for(int i = 0 ; i < 5; i++)
{
total += row[i]*col[i];
}
context.write(key, new IntWritable(total));
```

```
}
}
public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException
{
Configuration conf = new Configuration();
conf.set("dimension", "5"); // set the matrix dimension here.
Job job = Job.getInstance(conf);
//conf.set("fs.defaultFS", "hdfs://localhost:54310"); // take this value from core-site.xml
FileSystem fs = FileSystem.get(conf);
job.setJarByClass(Matrix.class);
// Need to set this since, map out is different from reduce out
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setMapperClass(Matrix_Mapper.class);
job.setReducerClass(Matrix_Reducer.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path input = new Path(args[0]);
Path output = new Path(args[1]);
fs.close();
FileInputFormat.addInputPath(job, input);
FileOutputFormat.setOutputPath(job, output);
job.waitForCompletion(true);
}
}
```

Usage

$export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.1.jar:$HADOOP_HOME/share/hadoop/coomon/hadoop-common-2.8.1.jar"

Compile Matrix.java and create a jar:

$ javac Matrix*.java
$ jar -cvf mat.jar Matrix*.class

Assuming that:

- **input1- input directory in HDFS**
- **output1 - output directory in HDFS**

Sample text-files as input:

$ hadoop fs -put mat.txt input1

Run the application:

$ hadoop jar mat.jar Matrix input1/mat.txt output1

Output:

$ hadoop fs -cat output1/part-r-00000

00 11878
01 14044
02 16031
03 5964
04 15874
10 4081
11 6914
12 8282
13 7479
14 9647
20 6844
21 9880
22 10636
23 6973
24 8873
30 10512
31 12037
32 10587
33 2934
34 5274
40 11182
41 14591
42 10954
43 1660
44 9981

Viva Questions

1. **How to set the number of mappers and reducers for Hadoop jobs?**

Ans: Users can configure JobConf variable say job to set the number of mappers and reducers i.e., job.setNumMapTasks(), job.setNumReduceTasks()

2. **List Writable data types in Map Reduce?**

Ans: Text, IntWritable,LongWritable,FloatWritable,BooleanWritable etc

3. **What is default Inputfomat and Outputformat for Map-reduce implementation?**

Ans: TextInputFormat and TextOutputFormat

4. **What is the difference between HDFS block and InputSplit?**

Ans: HDFS blocks splits the given data physically whereas InputSplit in Map-Reduce splits the given input files logically. The size of splits are user defined since InputSplit controls the number of mappers. If the size of split is not defined by the user, it takes HDFS default block size.

WEEK 9, 10

Aim: Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data.

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Pig.To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators

using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Download and Install Apache Pig

First Java and Hadoop must be installed before proceeding to install apache pig.

Download Apache Pig

Step 1:

First of all, download the latest version of Apache Pig from the following website −**https://pig.apache.org/**

Install Apache Pig:

After downloading the Apache Pig software, install it in your Linux environment by following the steps given below.

Step 2:

Extract the downloaded tar files using commands as shown below.

$ tar -zxvf pig-0.17.0.tar.gz

Configure Apache Pig

After installing Apache Pig, we have to configure it. To configure, we need to edit two files − **bashrc and pig.properties**.

.bashrc file

In the **.bashrc** file, set the following variables −

- **PIG_HOME** folder to the Apache Pig's installation folder,
- **PATH** environment variable to the bin folder, and
- **PIG_CLASSPATH** environment variable to the etc (configuration) folder of your Hadoop installations (the directory that contains the core-site.xml, hdfs-site.xml and mapred-site.xml files).

```
export PIG_HOME = pig-0.17.0
export PATH  = $PATH:$PIG_HOME/bin
export PIG_CLASSPATH = $HADOOP_HOME/etc/hadoop
```

pig.properties file

In the **conf** folder of Pig, we have a file named **pig.properties**. In the pig.properties file,various parameters can be set as given below.

```
pig -h properties
```

The following properties are supported −

Logging: verbose = true|false; default is false. This property is the same as -v
switch brief=true|false; default is false. This property is the same
as -b switch debug=OFF|ERROR|WARN|INFO|DEBUG; default is INFO.
This property is the same as -d switch aggregate.warning = true|false; default is true.
If true, prints count of warnings of each type rather than logging each warning.

Performance tuning: pig.cachedbag.memusage=<mem fraction>; default is 0.2 (20% of all memory).
Note that this memory is shared across all large bags used by the application.
pig.skewedjoin.reduce.memusagea=<mem fraction>; default is 0.3 (30% of all memory).
Specifies the fraction of heap available for the reducer to perform the join.
pig.exec.nocombiner = true|false; default is false.
Only disable combiner as a temporary workaround for problems.
opt.multiquery = true|false; multiquery is on by default.
Only disable multiquery as a temporary workaround for problems.
opt.fetch=true|false; fetch is on by default.
Scripts containing Filter, Foreach, Limit, Stream, and Union can be dumped without MR jobs.
pig.tmpfilecompression = true|false; compression is off by default.
Determines whether output of intermediate jobs is compressed.
pig.tmpfilecompression.codec = lzo|gzip; default is gzip.
Used in conjunction with pig.tmpfilecompression. Defines compression type.
pig.noSplitCombination = true|false. Split combination is on by default.
Determines if multiple small files are combined into a single map.

pig.exec.mapPartAgg = true|false. Default is false.
Determines if partial aggregation is done within map phase, before records are sent to combiner.
pig.exec.mapPartAgg.minReduction=<min aggregation factor>. Default is 10.
If the in-map partial aggregation does not reduce the output num records by this factor, it gets disabled.

Miscellaneous: exectype = mapreduce|tez|local; default is mapreduce. This property is the same as -x switch
pig.additional.jars.uris=<comma seperated list of jars>. Used in place of register command.
udf.import.list=<comma seperated list of imports>. Used to avoid package names in UDF.
stop.on.failure = true|false; default is false. Set to true to terminate on the first error.
pig.datetime.default.tz=<UTC time offset>. e.g. +08:00. Default is the default timezone of the host.
Determines the timezone used to handle datetime datatype and UDFs.

```
Additionally, any Hadoop property can be specified.
```

Verifying the Installation

Verify the installation of Apache Pig by typing the version command. If the installation is successful, you will get the version of Apache Pig as shown below.

```
Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58
```

Pig Latin

It is language used to analyze the data in hadoop using Apache pig.

Apache Pig Execution Mechanisms:

Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

- **Interactive Mode** (Grunt shell) − You can run Apache Pig in interactive mode using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).
- **Batch Mode** (Script) − You can run Apache Pig in Batch mode by writing the Pig Latin script in a single file with**.pig** extension.
- **Embedded Mode** (UDF) − Apache Pig provides the provision of defining our own functions (**U**ser **D**efined**F**unctions) in programming languages such as Java, and using them in our script.

Invoking the Grunt Shell using local mode

$pig -x local (local mode which invokes grunt shell)

grunt>

Pig load operator

This operator is used to load data from local file system.

Syntax

relation_name=load 'path' [using function] as (var1:datatype1, var2:datatype2 .....varn:datatypen)

Example

Consider text file(details.txt)

100,Raju,vizag

101,Ravi,hyderabad,

102,Venu,vizag

103,srinu,bangalore

104,vijay,hyderabad

105,teja,bangalore

grunt> student=load 'details.txt' using PigStorage(',') as (id:int,name:chararray, city:chararray);

to view output using dump operator

grunt> dump student;

(100,Raju,vizag)

(101,Ravi,hyderabad)

(102,Venu,vizag)

(103,srinu,bangalore)

(104,vijay,hyderabad)

(105,teja,bangalore)

grunt>

Invoking the Grunt Shell using Map-Reduce mode

$hadoop fs -put details.txt cse.(store details.txt in HDFS)

$pig -x mapreduce (mapreduce mode which invokes grunt shell)

grunt>

Pig load operator

This operator is used to load data from HDFS

Syntax

relation_name=load 'path' [using function] as (var1:datatype1, var2:datatype2 .....varn:datatypen)

Example

Consider text file(details.txt)

100,Raju,vizag

101,Ravi,hyderabad,

102,Venu,vizag

103,srinu,bangalore

104,vijay,hyderabad

105,teja,bangalore

grunt> student = load 'hdfs://localhost:54310/user/koushikhp/cse/details.txt' using PigStorage(',') as (id:int,name:chararray,city:chararray);

to view output using dump operator

grunt> dump student;

(100,Raju,vizag)

(101,Ravi,hyderabad)

(102,Venu,vizag)

(103,srinu,bangalore)

(104,vijay,hyderabad)

(105,teja,bangalore)

grunt>

Executing Apache Pig in Batch Mode:

You can write an entire Pig Latin script in a file and execute it using the **–x command**. Let us suppose we have a Pig script in a file named **script.pig** as shown below.

script.pig

```
Student = load  'details.txt' using PigStorage(',') as (id:int,name:chararray,
city:chararray);
Dump Student;
```

Executing pig script under local mode

```
$ pig -x local script.pig
(100,Raju,vizag)
(101,Ravi,hyderabad)
(102,Venu,vizag)
(103,srinu,bangalore)
(104,vijay,hyderabad)
(105,teja,bangalore)
```

1) Pig Latin script to sort data under batch mode

sort.pig (sorts the data in ascending order)

```
studentdet = load  'hdfs://localhost:54310/user/koushikhp/cse/det.txt' using PigStorage(',')
as (id:int,name:chararray, city:chararray);
orderdet = order studentdet by city;
dump orderdet;
```

Executing pig script

```
$pig -x mapreduce sort.pig
(105,teja,bangalore)
(103,srinu,bangalore)
(104,vijay,hyderabad)
(101,Ravi,hyderabad)
(102,Venu,vizag)
(100,Raju,vizag)
```

2) Pig Latin script to filter data under batch mode

filter.pig (filter operator is used to select the required tuples from a relation based on a condition.)

```
studentdet = load  'hdfs://localhost:54310/user/koushikhp/cse/det.txt' using PigStorage(',')
as (id:int,name:chararray, city:chararray);
filterdet = filter studentdet by city == 'vizag';
```

dump filterdet;

$pig -x mapreduce filter.pig

(100,Raju,vizag)

(102,Venu,vizag)

3) Pig Latin script to join data under batch mode using join operator

Syntax for join operator

Relation3 = join relation1 by key,relation2 by key.....relationn by key;

orders.txt (store in hdfs)

102,2016,3000

104,2016,3200

103,2016,3400

105,2016,3700

100,2016,3900

101,2016,3300

join.pig

```
studentdet = load  'hdfs://localhost:54310/user/koushikhp/cse/det.txt' using PigStorage(',')
as (id:int,name:chararray, city:chararray);
orderdet = load  'hdfs://localhost:54310/user/koushikhp/cse/orders.txt' using PigStorage(',')
as (id:int,year:int,amount:double);
joindet = join studentdet by id,orderdet by id;
dump joindet;
```

$pig -x mapreduce join.pig

(100,Raju,vizag,100,2016,3900.0)

(101,Ravi,hyderabad,101,2016,3300.0)

(102,Venu,vizag,102,2016,3000.0)

(103,srinu,bangalore,103,2016,3400.0)

(104,vijay,hyderabad,104,2016,3200.0)

(105,teja,bangalore,105,2016,3700.0)

4) **Pig Latin script to group data under batch mode using join operator**

group.pig

```
studentdet = load  'hdfs://localhost:54310/user/koushikhp/cse/det.txt' using PigStorage(',')
as (id:int,name:chararray, city:chararray);
groupdet = group studentdet by city;
dump groupdet;
```

the output consists of 2 parts

a) column which we have grouped the relation

b) bag which contains the group of tuples, student records with the respective column

$ pig -x mapreduce group.pig

(vizag,{(102,Venu,vizag),(100,Raju,vizag)})

(bangalore,{(105,teja,bangalore),(103,srinu,bangalore)})

(hyderabad,{(104,vijay,hyderabad),(101,Ravi,hyderabad)})

5. **Pig Latin script to generate specified data transformations based on the column data using foreach operator**

   foreach.pig

```
studentdet = load  'hdfs://localhost:54310/user/koushikhp/cse/det.txt' using PigStorage(',')
as (id:int,name:chararray, city:chararray);
foreachdet = foreach studentdet generate id,name;
dump foreachdet;
```

$pig -x mapreduce foreach.pig

(100,Raju)

(101,Ravi)

(102,Venu)

(103,srinu)

(104,vijay)

(105,teja)

Viva Questions

1. **List the features of Apache Pig?**

Ans:

a) Rich set of operators

b) Ease of programming

c) Optimization opportunities

d) extensibility

e) handles all kind of data

2. **List the differences between Apache Pig and Map-Reduce?**

Ans:

Apache Pig	**MapReduce**

Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

3. **List the different relational operators in pig?**

Ans: The different relational operators in pig latin are given below

a) load

b) store

c) filter

d) distinct

e) group

f) cogroup

h) join

i) order

j) limit

k) union

l) dump etc

4. **What is operator used to view schema of a relation?**

Ans:

describe operator used to view schema of a relation

5. **What is operator used to display the logical, physical, and MapReduce execution plans of a relation.**

Ans:

explain operator used to display the logical, physical, and MapReduce execution plans of a relation.

6. **What is operator used to give you the step-by-step execution of sequence of statements.**

Ans:

illustrate operator used to give you the step-by-step execution of sequence of statements.

7. **What is operator used to compute the cross product of two or more relations**

Ans:

cross operator used to compute the cross product of two or more relations.

8. **What is operator used to merge the content of two relations.**

Ans:

union operator used to merge the content of two relations.

9. **What is operator used to remove duplicate tuples from relation.**

Ans:

distinct operator used to remove duplicate tuples from relation.

10. **What is operator used to get a limited number of tuples from a relation.**

Ans:

limit operator is used to get a limited number of tuples from a relation.

11. **List the different modes of execution in Apache pig?**

Ans:

Apache pig can run in two modes

1) Local mode
2) Hadoop map-reduce command mode

12. What is use of foreach operation in pig scripts?

Ans: It is used to apply transformation to each element in bag so respective action is performed to generate new data items.

13. **List the complex data types in pig?**

Ans:

Pig supports three complex data types.

a) Maps- These are key, value pairs joined together using #.

b) Tuples- It is similar to the row in a table, where different items are separated by a comma. Tuples can have multiple attributes.

c) Bags- Unordered collection of tuples. It allows multiple duplicate tuples.

14. What are the different types of UDF's in Java supported by Apache Pig?

Ans: Algebraic, Eval and Filter functions are the various types of UDF's supported in Pig.

15. What is a UDF in Pig?

Ans:

If the in-built operators do not provide some functions then programmers can implement those functionalities by writing user defined functions using other programming languages like Java, Python, Ruby, etc. These User Defined Functions (UDF's) can then be embedded into a Pig Latin Script.

16. List the scalar data types in pig?

Ans: The scalar data types in pig are int, float, double, long, chararray, and bytearray.

17. Is Pig script case sensitive?

Ans: Pig script is both case sensitive and case insensitive. For example, in user defined functions, the field name, and relations are case sensitive ,i.e., INTELLIPAAT is not same as intellipaat or M=load 'test' is not same as m=load 'test'. And Pig script keywords are case insensitive i.e., LOAD is same as a load

18. How can we see only top 15 records from the student.txt out of100 records in the HDFS directory?

Ans: Let rename student.txt into student relation so the top 15 records can be viewed using limit operator

limit student 15

19. **Differentiate between Pig Latin and Pig Engine.?**

Ans: Pig Latin is scripting language like Perl for searching huge data sets and it is made up of a series of transformations and operations that are applied to the input data to produce data.
Pig engine is an environment to execute the Pig Latin programs. It converts Pig Latin operators into a series of MapReduce jobs.

20. List any two differences between Pig and Sql?

Ans:

Pig	Sql
It is procedural language	It is declarative language
It specifies how the task is to be performed	It specifies what is the task to be accomplished

WEEK 11, 12

Aim: Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Download and Install Apache Hive

First Java and Hadoop must be installed before proceeding to install apache hive

Download Apache Hive

Step 1:

First of all, download the latest version of Apache Hive from the following website **http://www-eu.apache.org/dist/hive/**

Install Apache Hive:

After downloading the Apache Hive software, install it in your Linux environment by following the steps given below.

Step 2:

Extract the downloaded tar files using commands as shown below.

$ tar -zxvf apache-hive-2.3.0-bin.tar.gz

Configure Apache Hive

Step 3:

After installing Apache Hive, we have to configure it. To configure, we need to edit .bashrc file.

export HIVE_HOME=apache-hive-2.3.0-bin

export PATH=$PATH:$HIVE_HOME/bin

$source ~/.bashrc

Step 4:

For Hive to interact with Hadoop HDFS, it must know the path to the hadoop installation directory. This can be achieved by configuring one of the hadoop hive configuration files **hive-config.sh file.**

export HADOOP_HOME=hadoop-2.8.1

Step 5:

Create a directory for the hive warehouse into hdfs. This directory will be used by Hive to store all the data into HDFS

$ hadoop fs -mkdir hive

Set permissions for hive warehouse

$ hadoop fs -chmod 777 hive

Step 6:

inform hive about the database that it should use for its schema definition. The below command tells hive to use derby database as its metastore database.

```
$ schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/koushikhp/apache-hive-2.3.0-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/koushikhp/hadoop-2.8.1/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL: jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User: APP
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.derby.sql
Initialization script completed
schemaTool completed
```

Step 7:

Verify hive installation

```
$hive
hive>
```

Hive usage to create database

Creating database

```
hive> create database college;
OK
Time taken: 0.074 seconds
```

Hive usage to create table

Syntax for creating table

create table [IF NOT EXISTS] [db_name.] table_name

[col1 datatype1, col2 datatype2.....coln datatypen]

[COMMENT table_comment]

[ROW FORMAT row_format]

[STORED AS file_format]

hive> create table if not exists college.student (id int,name String,city String)

> row format delimited

> fields terminated by ','

> lines terminated by '\n';

OK

Time taken: 0.02 seconds

LOADING DATA INTO TABLE

hive> LOAD DATA LOCAL INPATH 'det.txt' overwrite into table college.student;

Loading data to table college.student

OK

Time taken: 0.577 seconds

hive> select * from college.student;

OK

100 Raju vizag

101 Ravi hyderabad

102 Venu vizag

103 srinu bangalore

104 vijay hyderabad

105 teja bangalore

Time taken: 0.164 seconds, Fetched: 6 row(s)

Syntax for renaming table

alter table tablename rename to newtablename;

Syntax for adding columns

alter table add columns (col1 datatype1, col2 datatype2.....coln datatypen);

Syntax for inserting data into table

insert into table tablename values (col1 datatype1, col2 datatype2.....coln datatypen);

Syntax for dropping table

drop table [if exists] tablename;

Syntax for dropping database

drop database [if exists] databasename;

CREATING VIEW FROM EXISTING TABLE

hive> create view college.studentview as select * from college.student where city='vizag';

hive> select * from college.studentview;

OK

100 Raju vizag

102 Venu vizag

Time taken: 0.186 seconds, Fetched: 2 row(s)

Syntax for dropping view

drop view viewname;

Index

An Index is nothing but a pointer on a particular column of a table. Creating an index means creating a pointer on a particular column of a table. Its syntax is as follows:

```
create INDEX < INDEX_NAME> ON TABLE < TABLE_NAME(column names)> AS
INDEXHANDLER WITH DEFERRED REBUILD;
```

Example

hive> create index id_index on table college.student (id) AS 'COMPACT' with deferred rebuild;

hive> show indexes on college.studentid;

OK

id_index student id college__student_id_index__
compact Time taken: 0.115 seconds, Fetched: 1 row(s)

drop an index

hive> drop index id_index on college.student;

OK

Time taken: 2.089 seconds

hive> show indexes on student;

OK

Time taken: 0.044 seconds

SHOW DATABASES

hive> show databases;

OK

college

default

Time taken: 8.028 seconds, Fetched: 3 row(s)

USE DATABASE

hive> use college;

OK

Time taken: 0.017 seconds

SHOW TABLES IN DATABASE

hive> show tables;

OK

student

studentview

Time taken: 0.081 seconds, Fetched: 2 row(s)

Usage of functions

round() function

hive> select round(3.6);

OK

4

Time taken: 0.081 seconds, Fetched: 1 row(s)

count()

hive>select count(*) from college.student;

OK

6

Time taken: 5.274 seconds, Fetched: 1 row(s)

sum()

hive>select sum(id) from college.student;

MapReduce Jobs Launched:

Stage-Stage-1: HDFS Read: 216 HDFS Write: 0 SUCCESS

Total MapReduce CPU Time Spent: 0 msec

OK

615

Time taken: 5.422 seconds, Fetched: 1 row(s)

avg()

hive>select avg(id) from college.student;

OK

102.5

Time taken: 2.054 seconds, Fetched: 1 row(s)

min()

hive>select min(id) from college.student;

OK

100

Time taken: 1.93 seconds, Fetched: 1 row(s)

max()

hive>select max(id) from college.student;

OK

105

Time taken: 1.687 seconds, Fetched: 1 row(s)

Viva Questions

1. **What are the different types of tables available in hive?**

Ans:There are two types. Managed table and external table. In managed table both the data and schema in under control of hive but in external table only the schema is under control of Hive.

2. **Is hive suitable to be used for OLTP systems? Why?**

Ans:No Hive does not provide update & delete at row level. So it is not suitable for OLTP system.

3. **Define metastore in hive?**

Ans:It is a relational database storing the metadata of hive tables, partitions, Hive databases etc

4. **Why hive is needed?**

Ans:Hive is a tool in Hadoop ecosystem which provides an interface to organize and query data in a database like fashion and write SQL like queries. It is suitable for accessing and analyzing data in Hadoop using SQL syntax.

5. **Is there a date data type in hive?**

Ans:Yes. The TIMESTAMP data type stores date in java.sql.timestamp format

6. **What is default record delimitter for hive text files?**

Ans: The default record delimitter is '\n'

7. **How to list databases whose name starts with c?**

Ans:show databases like 'c.*';

8. **Does archiving of hive tables gives saving of space in HDFS?**

Ans:No. It only reduces the number of files which becomes easier for namenode to manage.

9. **While loading data into hive table using load data clause, how do you specify it in hdfs file and not local file?**

Ans:By Omitting the LOCAL CLAUSE in the LOAD DATA statement.

10. **How can hive avoid map-reduce?**

Ans:If setting the property hive.exec.mode.local.auto to true then hive will avoid mapreduce to fetch query results.

11. **What is difference between like and rlike operators in hive?**

Ans:The LIKE operator behaves the same way as the regular SQL operators used in select queries where as RLIKE operator uses more advance regular expressions which are available in java

12. Can the name of view will be same as name of table?

Ans:No. The name of a view must be unique when compared to all other tables and views present in the same database.

13. **List the differences between Pig and Hive?**

Ans:

Pig	Hive
Procedural language	Declarative language
Operates on client side of the cluster	Operates on the server side of cluster
It is mainly used by researchers and programmers	It is mainly used by data analysts
It loads the data effectively and quickly	It execute quickly but not load the data quickly
It is highly recommendable when having huge number of joins and filters	It is suitable when having limited number of joins and filters
Hadoop is not required to start for executing pig scripts	Hadoop is required to start for running hive
It supports both structured and unstructured data	It supports structured data

14. **List the commonly used Hive services?**

Ans:

Command line interface, Hive web interface, hive server, metastore etc

15. What is the use of Hcatalog?

Ans: Hcatalog can be used to share data structures with external systems. Hcatalog provides access to hive metastore to users of other tools on Hadoop so that they can read and write data to hive's data warehouse.

16. Where is table data stored in Apache Hive by default?

Ans: hdfs: //namenode_server/user/hive/warehouse

17. Are multiline comments supported in Hive?

Ans: No

18. Is it possible to overwrite Hadoop MapReduce configuration in Hive?

Ans: Yes, hadoop MapReduce configuration can be overwritten by changing the hive conf settings file.

19. What is the default database provided by Hive for Metastore ?

Ans: Derby is the default database.

20. How can you configure remote metastore mode in Hive?

Ans: To configure metastore in Hive, hive-site.xml file has to be configured with the below property –

hive.metastore.uris

thrift: //node1 (or IP Address):9083 specifies IP address and port of the metastore host