Lab - 12.3.2018

1. Create the employee details(eid,ename,designation) file and the department details(eid,department) file in HDFS. Let the department details can be stored using disttributed cache. Write the employee id, name, designation and department in the HDFS. Use map side join.

Note:

**public** **static** **class** Map **extends** Mapper<LongWritable,Text,Text,Text> {

Path[] cfile=**new** Path[0];

ArrayList<Text> empl=**new** ArrayList<Text>();

**public** **void** setup(Context context)

{

Configuration conf=context.getConfiguration();

**try**

{

cfile = DistributedCache.*getLocalCacheFiles*(conf);

BufferedReader reader=**new** BufferedReader(**new** FileReader(cfile[0].toString()));

String line;

**while** ((line=reader.readLine())!=**null**)

{

Text tt=**new** Text(line);

empl.add(tt);

}

}

**catch**(IOException e)

{

e.printStackTrace();

}

}

**public** **void** map(LongWritable key, Text value, Context context) **throws** IOException, InterruptedException {

String line2 = value.toString();

String[] elements=line2.split(",");

**for**(Text e:empl)

{

String[] line1 = e.toString().split(",");

**if**(elements[0].equals(line1[0]))

{

context.write(**new** Text(elements[0]),**new** Text(elements[1]+","+elements[2]+","+line1[1]));

}

}

}

}

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

Configuration conf = **new** Configuration();

Job job = **new** Job(conf, "distcache");

job.setJarByClass(distcache.**class**);

DistributedCache.*addCacheFile*(**new** Path(args[0]).toUri(),job.getConfiguration());

FileInputFormat.*addInputPath*(job, **new** Path(args[1]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[2]));

----

}

1. Create the customer details(cid,name) and the transaction details(cid, transaction amount) file in hdfs. Write the cname, number of transactions and transaction amount in hdfs. Use reduce side join.

Note:

**public** **static** **class** empmapper **extends**

context.write(**new** Text(line[0]), **new** Text("cust"+","+line[1]));

}

}

**public** **static** **class** depmapper **extends** Mapper<LongWritable,Text,Text,Text>

{

**public** **void** map(LongWritable key, Text value, Context context)

**throws** IOException, InterruptedException

context.write(**new** Text(line[0]), **new** Text("trans"+","+line[1]));

}

}

**public** **static** **class** jreducer **extends** Reducer<Text,Text,Text,Text>

{

String st1;

**public** **void** reduce(Text key, Iterable<Text> values, Context context )

**throws** IOException, InterruptedException

{

**int** c=0,amt=0;

**for**(Text val:values)

{

String[] line = val.toString().split(",");

**if** (line[0].equals("trans"))

{

c=c+1;

amt+= Integer.*parseInt*(line[1]);

}

**else** **if** (line[0].equals("cust"))

{

st1 = line[1];

}

}

context.write(**new** Text(st1), **new** Text(c+","+amt));

}

}