In this Assignment, we will be importing two data sets. One data set is small enough to be transported to each node through distributed cache. Using distributed cache , we will be editing out larger dataset.
This is an example of map-side join.

Mapper class

```
public class map_class extends MapReduceBase implements
Mapper<LongWritable,Text,Text,patent>{

    int i=0;
    long pat;
    String last,first,middle,s;
    String country=" ",state=" ";
    HashMap<String, String> stateMap = new HashMap<String,String>();
    HashMap<String, String> countryMap = new HashMap<String, String>();

    public void configure (JobConf job){
        String line = null;
        StringTokenizer words;
        Path[] path = new Path[1];
        try {
            path = DistributedCache.getLocalCacheFiles(job);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Getting paths from Distributed cache.

```
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(path[0].toString()));
        } catch (FileNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        try {
            line = br.readLine();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
```

Reading file contents using bufferedReader() by the path provided by distributed cache

```
        while(!(line.equals("Code Country")||line==null)){
            if(line.length()!=0){
                words = new StringTokenizer(line,"    ");
                s = words.nextToken();
                if(s.length()==2){
                    while(words.hasMoreTokens())
                        state = state +" "+ words.nextToken();
                    stateMap.put(s, state);
                    state = " ";
                }

            }
            try {
                line=br.readLine();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }
        try {
            line=br.readLine();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        while(line!=null){
```

State definitions being inserted in stateMap hashmap.

country definitions being inserted in countryMap hashmap.

```
            if(line.length()!=0){
                words = new StringTokenizer(line,"    ");
                if(words.hasMoreTokens()){
                    s = words.nextToken();
                    if(s.length()==2 && words.hasMoreTokens()){
                        while(words.hasMoreTokens())
                            country = country +" "+ words.nextToken();
                        countryMap.put(s, country);
                        country = " ";
                }
                }
            }
            try {
                line=br.readLine();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        try {
            br.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

HashMaps have been created in configure stage . Configure() is run once for each maptask while map() is run for every record.

```
    public void map(LongWritable key, Text value,OutputCollector<Text,patent>
output,Reporter reporter) throws IOException{
        StringTokenizer tokens = new StringTokenizer(value.toString(),",");
        s=tokens.nextToken();
        if(s.length()==7){
            pat=Long.parseLong(s.substring(0, s.length()));
            s=tokens.nextToken();
            if(s.length()>2)
                last = s.substring(1,s.length()-1);
            else
                last = " ";
            s=tokens.nextToken();
            if(s.length()>2)
                first = s.substring(1,s.length()-1);
            else
                first = " ";
            s=tokens.nextToken();
            if(s.length()>2)
                middle = s.substring(1,s.length()-1);
            else
                middle = " ";

            for(int i=1;i<=4;i++)
                s=tokens.nextToken();
            if(s.length()>2){
                state = s.substring(1,s.length()-1);
            }
            else
                state="unknown";
            s=tokens.nextToken();
            if(s.length()>2)
            country = s.substring(1,s.length()-1);
            if (country.equals("US")){
                last = last + '\t' + state;
            }
            if(countryMap.containsKey(country))
                country = countryMap.get(country);

            patent p = new patent(pat,first,middle,last);
            output.collect(new Text(country), p);          }}}
```

Data Extraction for various fields :

1) Country
2) State(for US)
3) Last name
4) First name
5) Middle name

Reducer Class

```
public class reduce_class extends MapReduceBase implements
Reducer<Text,patent,NullWritable,Text>{
      MultipleOutputs mo;
      NullWritable out = NullWritable.get();
      long n;
      int j;
      String name;
      String s;
      ArrayList<String> names;
      Iterator i;
      public void reduce(Text key,Iterator<patent>
values,OutputCollector<NullWritable,Text>output,Reporter reporter) throws IOException{
            HashMap<Long,ArrayList<String>> m = new HashMap<Long,ArrayList<String>>();
            while(values.hasNext()){
                  patent p=values.next();
                  n = p.getPatentNumber().get();
                  s = String.valueOf(n);

                  if(m.containsKey(n)){
                        names = m.get(n);
                        names.add(p.getName().toString());
                        m.put(n, names);
                  }
                  else{
                        names = new ArrayList<String>();
                        names.add(0, p.getName().toString());
                        m.put(n, names);
                  }
            }
            i = m.entrySet().iterator();
            while(i.hasNext()){
                  output.collect(out,new Text("------------------------------------"));
                  Map.Entry pairs = (Map.Entry)i.next();
                  name =  pairs.getKey().toString();
                  names = (ArrayList)pairs.getValue();

                  output.collect(out, new Text(name + " " +key.toString()));
                  Iterator<String> t = names.iterator();
                  while(t.hasNext()){
                        output.collect(out,new Text(t.next()));
                  }}}}
```

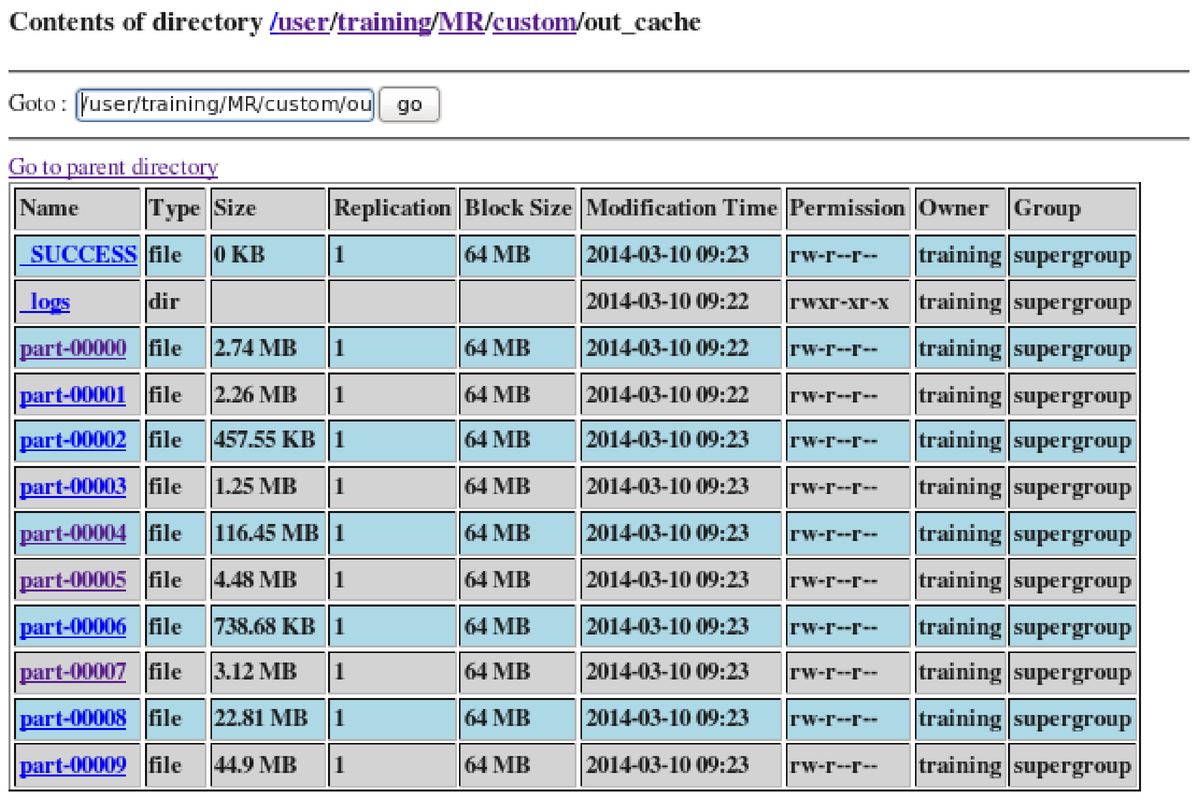
For a particular country(reduce key), all names corresponding to a particular patent are stored in hashmap m.

Runner Class

```
public class runner {
      public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
            JobConf conf = new JobConf(runner.class);
            conf.setJobName("cahce join");

            conf.setMapperClass(map_class.class);
            conf.setReducerClass(reduce_class.class);

            conf.setOutputKeyClass(NullWritable.class);
            conf.setOutputValueClass(Text.class);
            conf.setMapOutputKeyClass(Text.class);
            conf.setMapOutputValueClass(patent.class);

            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);

            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            DistributedCache.addCacheFile(new Path(args[2]).toUri(), conf);
            conf.setNumReduceTasks(10);;

            JobClient.runJob(conf);
            }}
```

Patent Class is similar as of previous assignments

Outputs



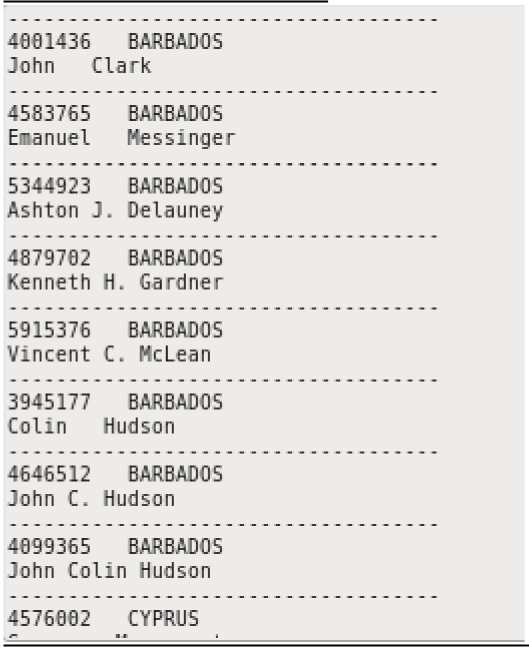
Contents of directory /user/training/MR/custom/out_cache

Goto : /user/training/MR/custom/ou go

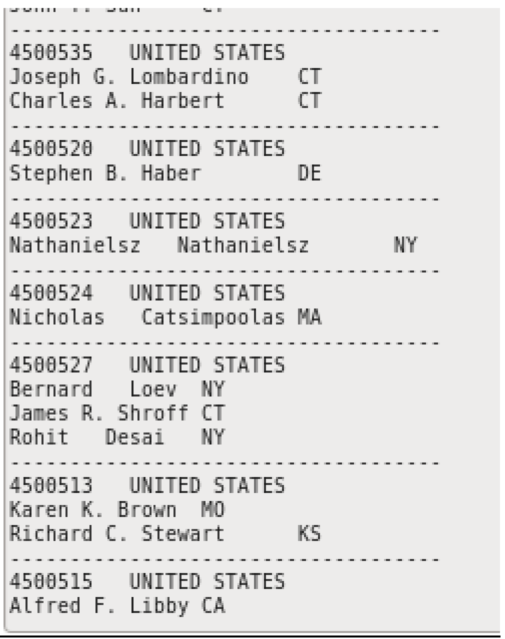
Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
_logs	dir				2014-03-10 09:22	rwxr-xr-x	training	supergroup
part-00000	file	2.74 MB	1	64 MB	2014-03-10 09:22	rw-r--r--	training	supergroup
part-00001	file	2.26 MB	1	64 MB	2014-03-10 09:22	rw-r--r--	training	supergroup
part-00002	file	457.55 KB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00003	file	1.25 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00004	file	116.45 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00005	file	4.48 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00006	file	738.68 KB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00007	file	3.12 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00008	file	22.81 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup
part-00009	file	44.9 MB	1	64 MB	2014-03-10 09:23	rw-r--r--	training	supergroup

Contents of individual files:



```
-------------------------------------
4001436   BARBADOS
John   Clark
-------------------------------------
4583765   BARBADOS
Emanuel   Messinger
-------------------------------------
5344923   BARBADOS
Ashton J. Delauney
-------------------------------------
4879702   BARBADOS
Kenneth H. Gardner
-------------------------------------
5915376   BARBADOS
Vincent C. McLean
-------------------------------------
3945177   BARBADOS
Colin   Hudson
-------------------------------------
4646512   BARBADOS
John C. Hudson
-------------------------------------
4099365   BARBADOS
John Colin Hudson
-------------------------------------
4576002   CYPRUS
```



```
-------------------------------------
4500535   UNITED STATES
Joseph G. Lombardino    CT
Charles A. Harbert      CT
-------------------------------------
4500520   UNITED STATES
Stephen B. Haber        DE
-------------------------------------
4500523   UNITED STATES
Nathanielsz   Nathanielsz      NY
-------------------------------------
4500524   UNITED STATES
Nicholas   Catsimpoolas MA
-------------------------------------
4500527   UNITED STATES
Bernard   Loev  NY
James R. Shroff CT
Rohit   Desai   NY
-------------------------------------
4500513   UNITED STATES
Karen K. Brown  MO
Richard C. Stewart      KS
-------------------------------------
4500515   UNITED STATES
Alfred F. Libby CA
```