**DistributedCache**

**Introduction**

When we write applications using Map Reduce, we may require to share some files across all nodes in Hadoop Cluster. It can be simple properties file or can be executable jar file.

Hadoop Map Reduce Framework provides us this facility with something called as DistributedCache.

DistributedCache is a facility provided by the Map-Reduce framework to cache files (text, archives, jars etc.) needed by applications.

Applications specify the files, via urls (hdfs:// or http://) to be cached via the **JobConf.**

The DistributedCache assumes that the files specified via urls are already present on **the**[**FileSystem**](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html) at the path specified by the url and are accessible by every machine in the cluster.

The framework will copy the necessary files on to the slave node before any tasks for the job are executed on that node.

DistributedCache can be used to distribute simple, read-only data/text files and/or more complex types such as archives, jars etc.

Clearly the cache files should not be modified by the application or externally while the job is executing.

**Setting up the cache for the application**

1. Copy the requisite files to the FileSystem:

$ bin/hadoop fs -copyFromLocal lookup.dat /myapp/lookup.dat

2. Setup the application's JobConf:

JobConf job = new JobConf();

DistributedCache.addCacheFile(new URI("/myapp/lookup.dat"),job);

3. Use the cached files in the [Mapper](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/Mapper.html) or [Reducer](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/Reducer.html):

public static class MyMapper

extends Mapper<LongWritable,Text, Text, Text> {

protected void setup(Context context)

throws java.io.IOException, InterruptedException{

**// Get the cached archives/files**

Path[] uri = DistributedCache.getLocalCacheFiles(context.getConfiguration());

BufferedReader fis = new BufferedReader(new FileReader(uri[0].toString()));

}

protected void map(LongWritable key, Text value, Context context)

throws java.io.IOException, InterruptedException {

**// Use data from the cached archives/files here**

**// ...**

**// ...**

output.collect(k, v);

}

}