

# Big data: architectures and data analytics

## Distributed cache

## Distributed cache

- Some applications need to share and cache (small) read-only files to perform efficiently their task
- These files should be accessible by all nodes of the cluster in an efficient way
  - Hence a copy of the shared/cached files should be available in all nodes used to run the application
- **DistributedCache** is a facility provided by the Hadoop-based MapReduce framework to cache files
  - E.g., text, archives, jars needed by applications

3

## Distributed cache

- In the Driver of the application, the set of shared/cached files are specified
  - By using the job.**addCacheFile**(path) method
- During the initialization of the job, Hadoop creates a “local copy” of the shared/cached files in all nodes which are used to execute some tasks (mappers or reducers) of the job (i.e., of the running application)
- The shared/cache file is read by the mapper (or the reducer), usually in its setup method
  - Since the shared/cached file is available **locally** in the node, its content can be read efficiently

4

## Distributed cache

- The efficiency of the distributed cache depends on the number of multiple mappers (or reducers) running on the same node
  - For each node a local copy of the file is copied during the initialization of the job
  - The local copy of the file is used by all mappers (reducers) running on the same node (server)
- Without a distributed approach, each mapper (reducer) should read, in the setup method, the shared HDFS file
  - Hence, more time is needed because reading data from HDFS is more inefficient than reading data from the local file system of the node running the mappers (reducers)

5

## Distributed cache

Structure

6

## Distributed cache: driver

```
public int run(String[] args) throws Exception {  
    .....  
  
    // Add the shared/cached HDFS file in the  
    // distributed cache  
    job.addCacheFile(new Path("hdfs  
    path").toUri());  
  
    .....  
}
```

7

## Distributed cache: mapper/reducer

```
protected void setup(Context context) throws IOException,  
    InterruptedException {  
    .....  
    String line;  
  
    // Retrieve the paths of the local copies of  
    // the distributed files  
    URI[] urisCachedFiles = context.getCacheFiles();
```

8

## Distributed cache: mapper/reducer

```
// Read the content of the cached file and process it
// in this example the content of the first shared file is opened
BufferedReader file = new BufferedReader(new FileReader(
    new File(urisCachedFiles [0].getPath())));

// Iterate over the lines of the file
while ((line = file.readLine()) != null) {
    // process the current line
    .....
}

file.close();
}
```