Exercise 02a: Map Reduce applications for Word Counting

Previous exercise described how to save input file in to HDFS. This exercise train students to do MapReduce process using word counting application.

Prerequisites

Ensure that Hadoop is installed, configured and is running. More details:

Single Node Setup for first-time users.

Cluster Setup for large, distributed clusters.

MapReduce Overview

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuratio n allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master `ResourceManager`, one worker `NodeManager` per cluster-node, and `MRAppMaster` per application.

Minimally, applications specify the input/output locations and supply *map* and *reduce* functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the *job configuration*.

The Hadoop *job client* then submits the job (jar/executable etc.) and configuration to the `ResourceManager` which then assumes the responsibility of distributing the software/configuration to the workers, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

Inputs and Outputs

The MapReduce framework operates exclusively on `<key, value>` pairs, that is, the framework views the input to the job as a set of `<key, value>` pairs and produces a set of `<key, value>` pairs as the output of the job, conceivably of different types.

The `key` and `value` classes have to be serializable by the framework and hence need to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.

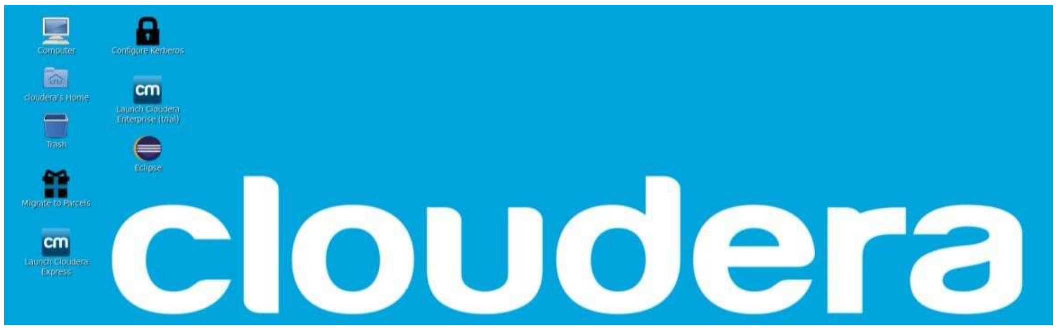
Input and Output types of a MapReduce job:

(input) `<k1, v1>` -> **map** -> `<k2, v2>` -> **combine** -> `<k2, v2>` -> **reduce** -> `<k3, v3>` (output)

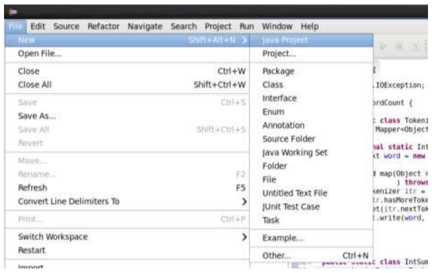
Step 1

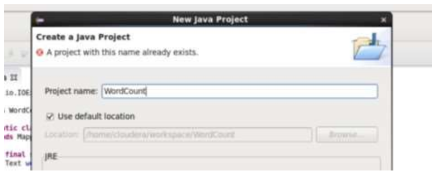
Compile `WordCount.java` and create a jar:

(i) Open Eclipse in Clouderea



(ii) Create ‘WordCount’ java project



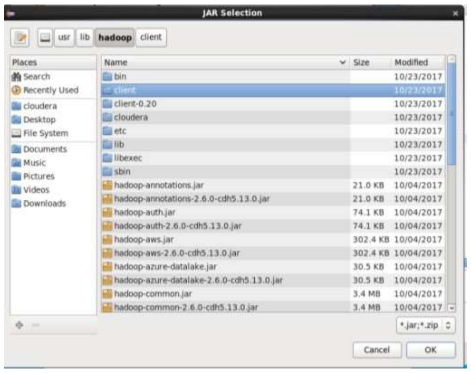


A project with this name already exists.

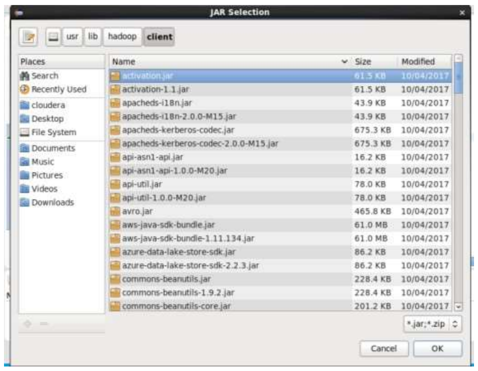
Project name: WordCount

Location: /home/cloudera/workspace/WordCount

Import following Jar files

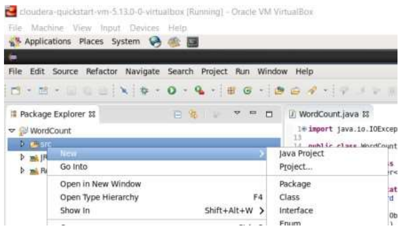


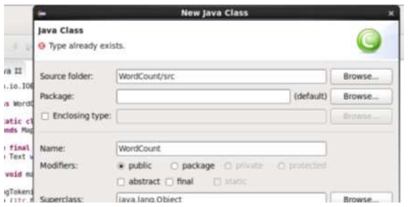
Name	Size	Modified
bin		10/23/2017
client		10/23/2017
client-0.20		10/23/2017
cloudera		10/23/2017
etc		10/23/2017
lib		10/23/2017
libexec		10/23/2017
sbin		10/23/2017
hadoop-annotations.jar	21.0 KB	10/04/2017
hadoop-annotations-2.6.0-cdh5.13.0.jar	21.0 KB	10/04/2017
hadoop-auth.jar	74.1 KB	10/04/2017
hadoop-auth-2.6.0-cdh5.13.0.jar	74.1 KB	10/04/2017
hadoop-aws.jar	302.4 KB	10/04/2017
hadoop-aws-2.6.0-cdh5.13.0.jar	302.4 KB	10/04/2017
hadoop-azure-datalake.jar	30.5 KB	10/04/2017
hadoop-azure-datalake-2.6.0-cdh5.13.0.jar	30.5 KB	10/04/2017
hadoop-common.jar	3.4 MB	10/04/2017
hadoop-common-2.6.0-cdh5.13.0.jar	3.4 MB	10/04/2017



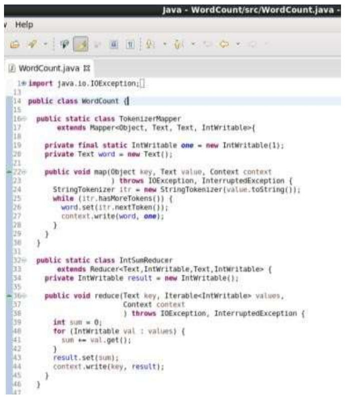
Name	Size	Modified
activation.jar	61.5 KB	10/04/2017
activation-1.1.jar	61.5 KB	10/04/2017
apacheds-i18n.jar	43.9 KB	10/04/2017
apacheds-i18n-2.0.0-M15.jar	43.9 KB	10/04/2017
apacheds-kerberos-codec.jar	675.3 KB	10/04/2017
apacheds-kerberos-codec-2.0.0-M15.jar	675.3 KB	10/04/2017
api-asn1-api.jar	16.2 KB	10/04/2017
api-asn1-api-1.0.0-M20.jar	16.2 KB	10/04/2017
api-util.jar	78.0 KB	10/04/2017
api-util-1.0.0-M20.jar	78.0 KB	10/04/2017
avro.jar	465.8 KB	10/04/2017
aws-java-sdk-bundle.jar	61.0 MB	10/04/2017
aws-java-sdk-bundle-1.11.134.jar	61.0 MB	10/04/2017
azure-data-lake-store-sdk.jar	86.2 KB	10/04/2017
azure-data-lake-store-sdk-2.2.3.jar	86.2 KB	10/04/2017
commons-beanutils.jar	228.4 KB	10/04/2017
commons-beanutils-1.9.2.jar	228.4 KB	10/04/2017
commons-beanutils-core.jar	201.2 KB	10/04/2017

(iii) Create ‘WordCount.java’ in src folder


cloudera-quickstart-vm-5.13.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
WordCount
src
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Shift+Alt+W
Java Project
Project...
Package
Class
Interface
WordCount.java

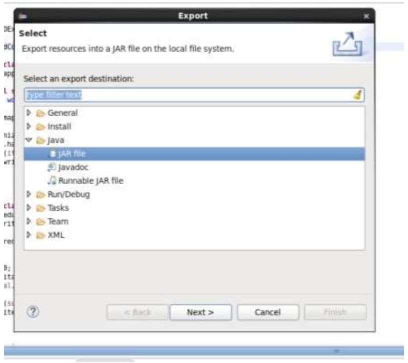


Source folder: WordCount/src
Package: (default)
Enclosing type:
Name: WordCount
Modifiers: [✓ public] package private protected
abstract final static
Superclass: java.lang.Object



```
import java.io.IOException;

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);
    }
  }
```

(iv) Create WordCount.jar file



Export resources into a JAR file on the local file system.

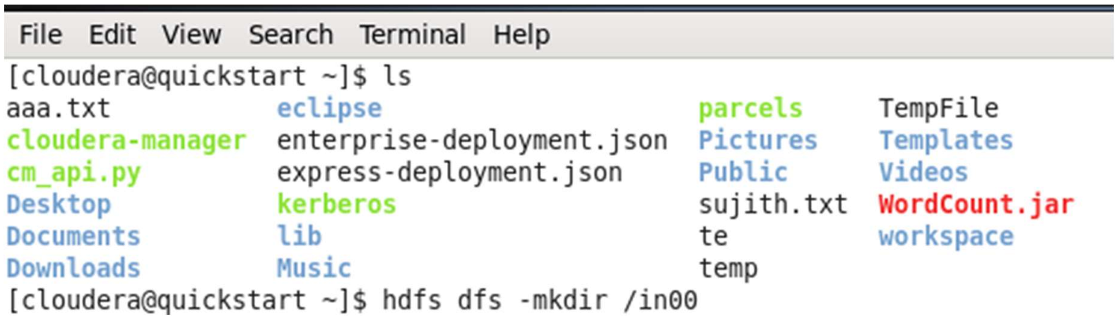
Select an export destination:

type filter text

Step 2

Create following folders in HDFS:

- `/input` - input directory in HDFS
- `/output` - output directory in HDFS



```
[cloudera@quickstart ~]$ ls
aaa.txt            eclipse                      parcels     TempFile
cloudera-manager   enterprise-deployment.json   Pictures    Templates
cm_api.py          express-deployment.json      Public      Videos
Desktop            kerberos                     sujith.txt  WordCount.jar
Documents          lib                          te          workspace
Downloads          Music                        temp
[cloudera@quickstart ~]$ hdfs dfs -mkdir /in00
```

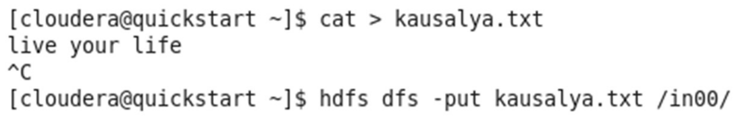
Step 3

Create and copy sample text-files into input folder:

```
[cloudera@quickstart ~]$ hdfs dfs -ls /in00/
```

Found 1 items

```
-rw-r--r--  1 cloudera supergroup        158 2021-08-15 04:32 /in00/WCFile.txt
```



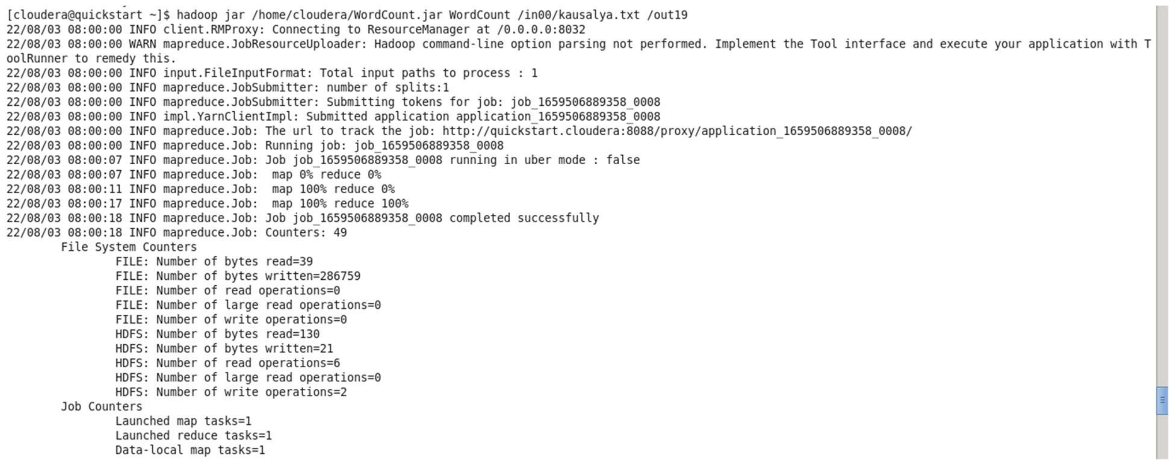
```
[cloudera@quickstart ~]$ cat > kausalya.txt
live your life
^C
[cloudera@quickstart ~]$ hdfs dfs -put kausalya.txt /in00/
```

Step 4

Run the MapReduce application:

```
hadoop jar /home/cloudera/WordCount.jar WordCount /in00/WCFile.txt /out00
```

Show MapReduce Framework



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /in00/kausalya.txt /out19
22/08/03 08:00:00 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/08/03 08:00:00 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with T
oolRunner to remedy this.
22/08/03 08:00:00 INFO input.FileInputFormat: Total input paths to process : 1
22/08/03 08:00:00 INFO mapreduce.JobSubmitter: number of splits:1
22/08/03 08:00:00 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1659506889358_0008
22/08/03 08:00:00 INFO impl.YarnClientImpl: Submitted application application_1659506889358_0008
22/08/03 08:00:00 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1659506889358_0008/
22/08/03 08:00:00 INFO mapreduce.Job: Running job: job_1659506889358_0008
22/08/03 08:00:07 INFO mapreduce.Job: Job job_1659506889358_0008 running in uber mode : false
22/08/03 08:00:07 INFO mapreduce.Job:  map 0% reduce 0%
22/08/03 08:00:11 INFO mapreduce.Job:  map 100% reduce 0%
22/08/03 08:00:17 INFO mapreduce.Job:  map 100% reduce 100%
22/08/03 08:00:18 INFO mapreduce.Job: Job job_1659506889358_0008 completed successfully
22/08/03 08:00:18 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=39
                FILE: Number of bytes written=286759
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=130
                HDFS: Number of bytes written=21
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
```

Step 5

Output:

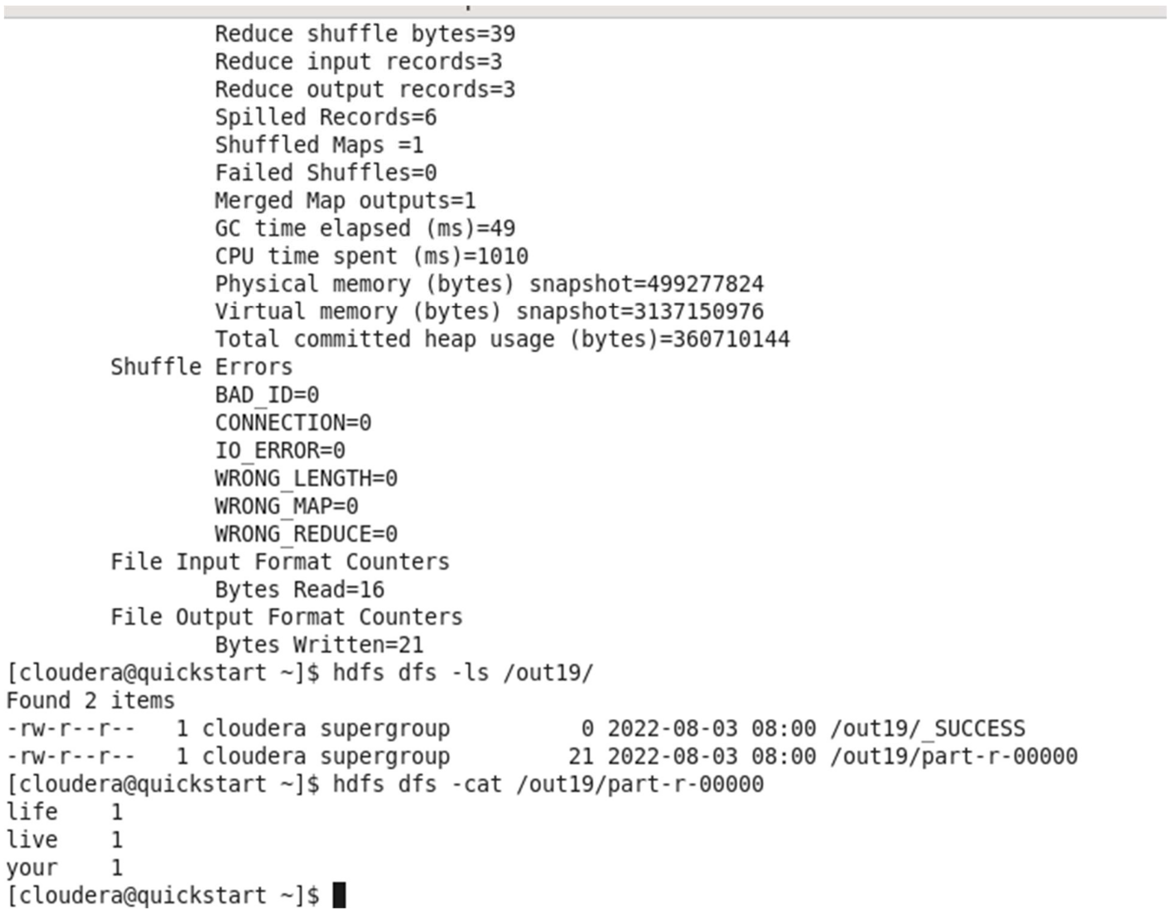
[cloudera@quickstart ~]$ hdfs dfs -ls /out00/

Found 2 items

-rw-r--r-- 1 cloudera supergroup   0 2021-08-15 04:41 /out00/_SUCCESS

-rw-r--r-- 1 cloudera supergroup   113 2021-08-15 04:41 /out00/part-r-00000

[cloudera@quickstart ~]$ hdfs dfs -cat /out00/part-r-00000



```
                Reduce shuffle bytes=39
                Reduce input records=3
                Reduce output records=3
                Spilled Records=6
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=49
                CPU time spent (ms)=1010
                Physical memory (bytes) snapshot=499277824
                Virtual memory (bytes) snapshot=3137150976
                Total committed heap usage (bytes)=360710144
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=16
        File Output Format Counters
                Bytes Written=21
[cloudera@quickstart ~]$ hdfs dfs -ls /out19/
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2022-08-03 08:00 /out19/_SUCCESS
-rw-r--r--   1 cloudera supergroup         21 2022-08-03 08:00 /out19/part-r-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /out19/part-r-00000
life    1
live    1
your    1
[cloudera@quickstart ~]$ 
```