Hadoop Assignment - NWEN 406

Zoltan Debre - 300360191

Original repository: https://github.com/zoltan-nz/hadoop

My goal with this assignment is not only to be able to run Hadoop tasks on a previously created environment, but also I would like to learn and setup Hadoop environment from scratch. Additionally, I prefer fully reproducible, portable solution, which can be installed by anyone, without using a predefined setup, so it can work outside of our campus. For this reason I will use `maven` for managing Java dependencies.

At the end of this report I describe how I created an up to date portable and standard Docker container for Hadoop and how I use that main container image inside this project. It is recommended to run these scripts using Docker container, especially if the environment (Java, Maven, Hadoop) is not available on the host computer.

PART 0 - Setup Hadoop on MacOS

Firstly, I setup a development environment on MacOS.

Prerequisites, documentation

- Brew package manager: Homebrew
- Setting up Hadoop on Mac OSX
- Java 8 and Maven

Install Hadoop on Mac

Please install Java 8

```
$ brew install java
$ brew install maven
$ brew install hadoop
```

Run `brew info hadoop` to get the install directory, which need to setup `HADOOP_HOME` system environment variable.

Setup ENV variables in bash or zsh profile:

```
export JAVA_HOME="$(/usr/libexec/java_home)"
export HADOOP_HOME=path from 'brew info hadoop'
export HADOOP_PREFIX=$HADOOP_HOME/libexec
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

Install Hadoop on Ubuntu

If you prefer Ubuntu instead of Mac, the following description is a great starting point:

- Install Hadoop on Ubuntu

Start Hadoop

```
$ cd $HADOOP_HOME
$ bin/hdfs namenode -format
$ sbin/start-dfs.sh
$ sbin/start-yarn.sh
```

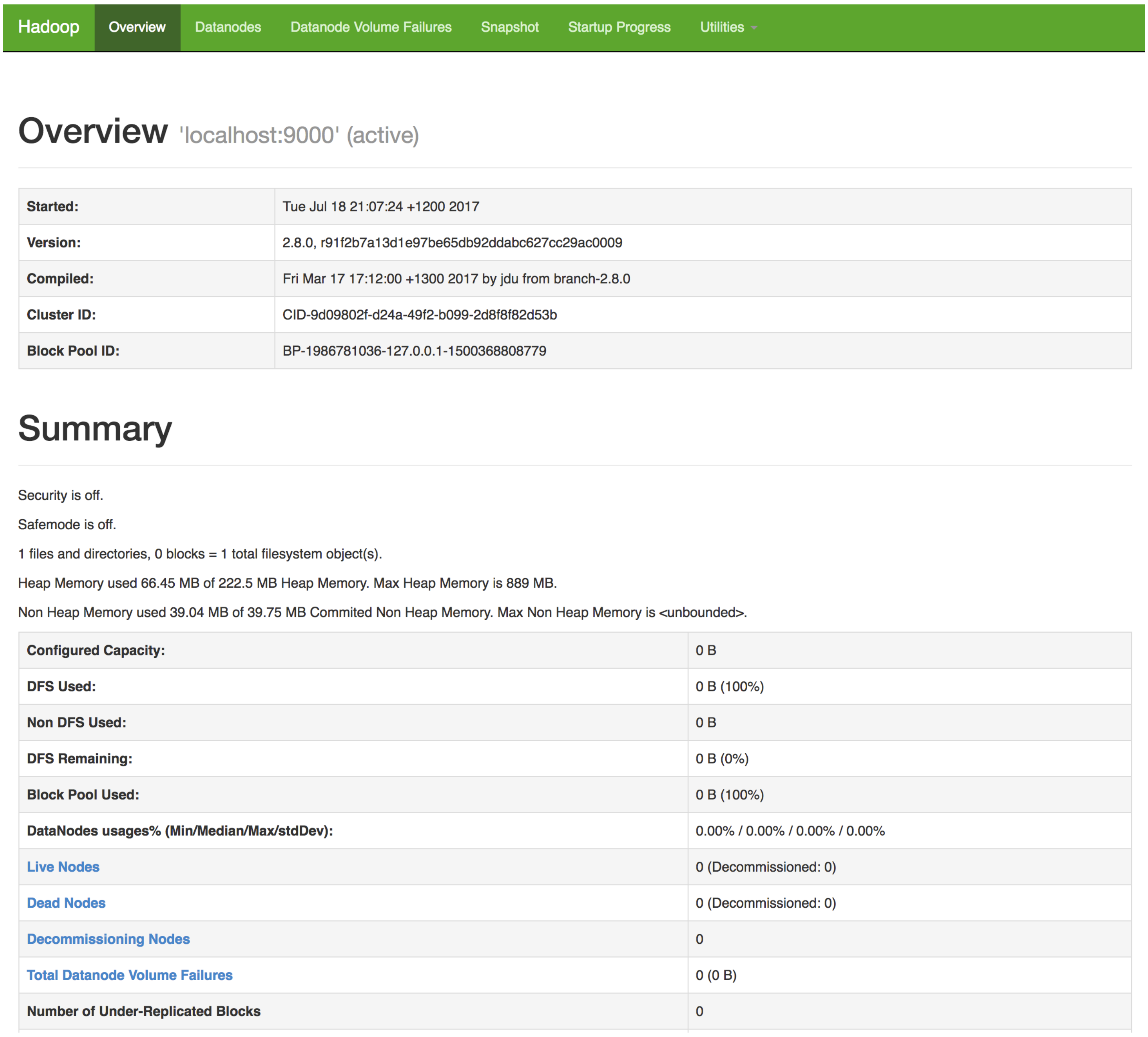
Create user folder

```
$ bin/hdfs dfs -mkdir /user
$ bin/hdfs dfs -mkdir /user/zoltan
```

Open Hadoop admin pages

Browse the web interface for the NameNode:

```
$ open http://localhost:50070/
```



Overview 'localhost:9000' (active)

Started:	Tue Jul 18 21:07:24 +1200 2017
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 17:12:00 +1300 2017 by jdu from branch-2.8.0
Cluster ID:	CID-9d09802f-d24a-49f2-b099-2d8f8f82d53b
Block Pool ID:	BP-1986781036-127.0.0.1-1500368808779

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

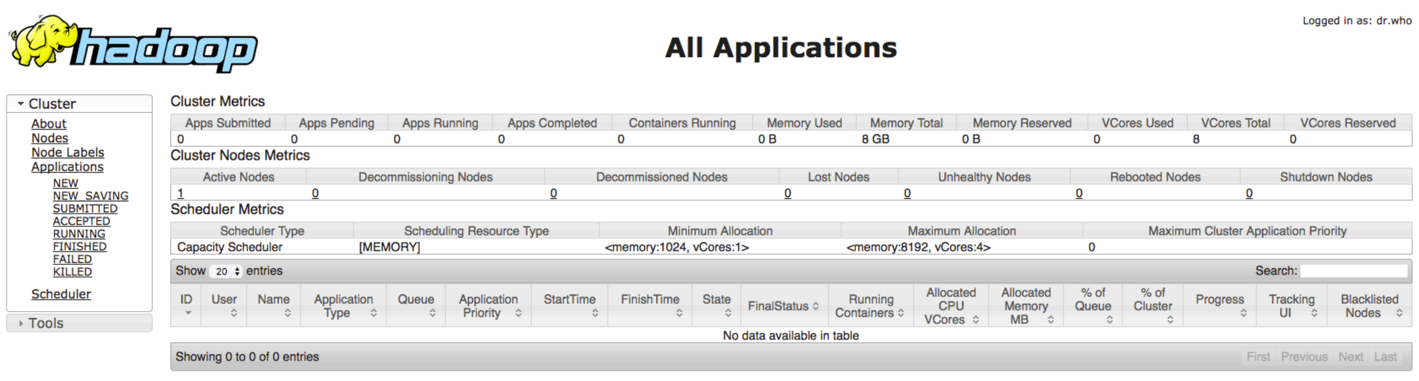
Heap Memory used 66.45 MB of 222.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 39.04 MB of 39.75 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B
DFS Remaining:	0 B (0%)
Block Pool Used:	0 B (100%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	0 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0

Open The ResourceManager:

```
$ open http://localhost:8088/
```



Logged in as: dr.who

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

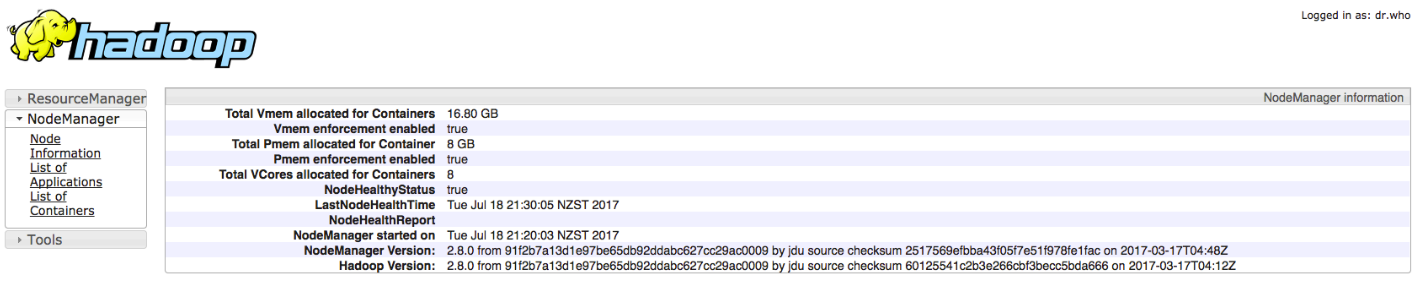
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																	

Showing 0 to 0 of 0 entries First Previous Next Last

```
$ open http://localhost:8042
```



Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Tue Jul 18 21:30:05 NZST 2017
NodeHealthReport	
NodeManager started on	Tue Jul 18 21:20:03 NZST 2017
NodeManager Version:	2.8.0 from 91f2b7a13d1e97be65db92ddabc627cc29ac0009 by jdu source checksum 2517569efbba43f05f7e51f978fe1fac on 2017-03-17T04:48Z
Hadoop Version:	2.8.0 from 91f2b7a13d1e97be65db92ddabc627cc29ac0009 by jdu source checksum 60125541c2b3e266cbf3becc5bda666 on 2017-03-17T04:12Z

PART 1 - Map Reduce Tutorial with Maven and IntelliJ

I prefer to use a modern IntelliJ IDEA editor for Java projects with Maven package manager.

We can use Maven archetype to create a simple Java project quickly and Maven can download package dependencies also. In this case we need a few Hadoop packages.

A useful article in this topic:

- http://www.soulmachine.me/blog/2015/01/30/debug-hadoop-applications-with-intellij/

Create the skeleton project using Maven from command line.

```
$ mvn archetype:generate -DgroupId=nz.zoltan -DartifactId=wordcount -DarchetypeArtif
```

Alternative option, using IntelliJ IDEA create new project with `org.apache.maven.archetypes:maven-archetype-quickstart`

Our new project will be generated inside the `wordcount` folder of this repository.

Maven creates a default `App.java` file and connected test file. We can delete these files. Create a new `WordCount.java`. The file structure should look like the following.

```
$ tree wordcount
wordcount
├── pom.xml
├── src
|   ├── main
|   |   └── java
|   |       └── nz
|   |           └── zoltan
|   |               ├── HelloWorld.java
|   |               └── WordCount.java
|   └── test
|       └── java
|           └── nz
|               └── zoltan
|                   └── HelloWorldTest.java
```

We have to update the `pom.xml` to setup the Hadoop dependencies:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/mave
  <modelVersion>4.0.0</modelVersion>
  <groupId>nz.zoltan</groupId>
  <artifactId>wordcount</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>wordcount</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <hadoop.version>2.8.1</hadoop.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client</artifactId>
      <version>${hadoop.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-core</artifactId>
      <version>LATEST</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>2.4</version>
        <configuration>
          <archive>
            <manifest>
              <mainClass></mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Important note for running different main class with hadoop: `mainClass` has to be empty in `build/plugins/plugin` section. Otherwise running the jar file with `hadoop` will always run the mainClass file and will ignore the specified class in the command line.

`WordCount.java` :

```
// Original source: https://ecs.victoria.ac.nz/foswiki/pub/Courses/NWEN406_2017T2/Lo
package nz.zoltan;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

import java.io.IOException;
import java.util.Iterator;
import java.util.StringTokenizer;

public class WordCount {
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Te
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, OutputCollector<Text, IntWrit
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWri
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<T
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("wordcount");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);

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

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

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

Create a `wordcount/jobs` folder for Hadoop jobs. We can save in this folder all the text files what we would like to process with our WordCount app.

Using IntelliJ IDEA help us to debug our source code interactively. We can jump into the Hadoop source code also, because IntelliJ can download automatically the connected packages source files.

There are three options to launch our map reduce app. One of them is as simple Java process, using the IntelliJ `Run` option. Alternative option is using `mvn exec:java`. Third option is using `hadoop jar`.

Change directory:

```
$ cd wordcount
```

Using `mvn exec`:

```
$ mvn exec:java -Dexec.mainClass="nz.zoltan.WordCount" -Dexec.args="jobs/example1/ir
```

Building the jar file:

```
$ mvn clean package
```

Using `hadoop`:

```
$ hadoop jar target/wordcount-1.0-SNAPSHOT.jar nz.zoltan.WordCount jobs/example1/inp
```

However, the last solution will work only if we removed the `mainClass` from `pom.xml` as described above at `pom.xml` setup.

WordCount2.java

Original source:
https://ecs.victoria.ac.nz/foswiki/pub/Courses/NWEN406_2017T2/LabTutorial1/WordCount2.java

```
// Source: https://ecs.victoria.ac.nz/foswiki/pub/Courses/NWEN406_2017T2/LabTutorial
package nz.zoltan;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.net.URI;
import java.util.ArrayList;
import java.util.HashSet;
```

```java
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.StringUtils;

public class WordCount2 {

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable> {

        enum CountersEnum {INPUT_WORDS}

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        private boolean caseSensitive;
        private Set<String> patternsToSkip = new HashSet<String>();

        private Configuration conf;
        private BufferedReader fis;

        @Override
        public void setup(Context context) throws IOException, InterruptedException
            conf = context.getConfiguration();
            caseSensitive = conf.getBoolean("wordcount.case.sensitive", true);
            if (conf.getBoolean("wordcount.skip.patterns", false)) {
                URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
                for (URI patternsURI : patternsURIs) {
                    Path patternsPath = new Path(patternsURI.getPath());
                    String patternsFileName = patternsPath.getName().toString();
                    parseSkipFile(patternsFileName);
                }
            }
        }

        private void parseSkipFile(String fileName) {
            try {
                fis = new BufferedReader(new FileReader(fileName));
                String pattern = null;
                while ((pattern = fis.readLine()) != null) {
                    patternsToSkip.add(pattern);
                }
            } catch (IOException ioe) {
                System.err.println("Caught exception while parsing the cached file '
                        + StringUtils.stringifyException(ioe));
            }
        }

        @Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String line = (caseSensitive) ?
                    value.toString() : value.toString().toLowerCase();
            for (String pattern : patternsToSkip) {
                line = line.replaceAll(pattern, "");
            }
            StringTokenizer itr = new StringTokenizer(line);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
                Counter counter = context.getCounter(CountersEnum.class.getName(),
                        CountersEnum.INPUT_WORDS.toString());
                counter.increment(1);
            }
        }
    }

    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);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        GenericOptionsParser optionParser = new GenericOptionsParser(conf, args);
        String[] remainingArgs = optionParser.getRemainingArgs();
        if ((remainingArgs.length != 2) && (remainingArgs.length != 4)) {
            System.err.println("Usage: wordcount <in> <out> [-skip skipPatternFile]'
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount2.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        List<String> otherArgs = new ArrayList<String>();
        for (int i = 0; i < remainingArgs.length; ++i) {
            if ("-skip".equals(remainingArgs[i])) {
                job.addCacheFile(new Path(remainingArgs[++i]).toUri());
                job.getConfiguration().setBoolean("wordcount.skip.patterns", true);
            } else {
                otherArgs.add(remainingArgs[i]);
            }
        }
        FileInputFormat.addInputPath(job, new Path(otherArgs.get(0)));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs.get(1)));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

One of the option to run different class from the same `.jar` file is the following:

```
cd ./wordcount
mvn exec:java -Dexec.mainClass="nz.zoltan.WordCount2" -Dexec.args="jobs/example2/inp
mvn exec:java -Dexec.mainClass="nz.zoltan.WordCount2" -Dexec.args="jobs/example3/inp
```

Or using `hadoop`:

```
cd ./wordcount
hadoop jar target/wordcount-1.0-SNAPSHOT.jar nz.zoltan.WordCount2 jobs/example2/inpu
hadoop jar target/wordcount-1.0-SNAPSHOT.jar nz.zoltan.WordCount2 jobs/example3/inpu
```

Please note, there is a small bug in Hadoop 2.8 and earlier version which caused by `StatisticsDateReferenceCleaner`, because it swallows interrupt exceptions. You will see a `WARNING` note when we run the above `maven` commands, but we can ignore it now.

I realized, the best for running our task using the IntelliJ IDEA run option, because in this case we can setup different params.

Program arguments with case sensitive enabled and with skip patterns:

```
-Dwordcount.case.sensitive=true jobs/example3/input jobs/example3/case-true-with-ski
```

Patent Citation Example

You can find `PatentCitation.java` file in `./wordcount/src/main/java/nz/co.blogspot.anlisaldhana/` folder.

I moved the `cite75_99.txt` to `./wordcount/jobs/patent-citation/input` folder. (GitHub has a file size limit, so you can find this file zipped, please unzip it before running the hadoop process.)

I run the the `PatentCitation` with IntelliJ runner with the following params:

Program arguments: `jobs/patent-citation/input jobs/patent-citation/output` And the working directory is point to `wordcount` folder.

The result saved in `jobs/patent-citation/output` folder. (This file is also zipped because of GitHub limitation.)

Waitangi Treaty

`treaty.txt` file is saved in `./wordcount/jobs/part-1-waitangi/input`.

Run word counter with maven:

```
mvn exec:java -Dexec.mainClass="nz.zoltan.WordCount" -Dexec.args="jobs/part-1-waitar
```

Result saved in `./wordcount/jobs/part-1-waitangi/output`

PART 2 - TASK 1

Find the search history for a specified user. The output format is: {AnonID, Query, ItemRank, ClickURL}.

- Project folder: `./aol`

- Text files (excluded from github): `./aol/jobs/aol`
- Output file: `.aol/jobs/part-2-task-1/output`

For solving AOL related tasks, I downloaded the aol search log from our campus server. Because of the size of these files, they cannot be part of the repository. If you have these files, please copy them in `./aol/jobs/aol` folder.

Please remove the `README.txt` file from the `aol` folder.

The following `TaskOne.java` program create a comma separated list from all query.

Location: `./aol/src/java/nz/zoltan/TaksOne.java` :

```
package nz.zoltan;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.FileNotFoundException;
import java.io.IOException;

public class TaskOne extends Configured implements Tool {
    private static final String JOB_NAME = "task1";
    private static final String ANON_ID = "anonId";

    public static class TaskOneMapper extends Mapper<LongWritable, Text, Text, Text>

        protected void map(LongWritable key, Text searchLogLine, Context context) th
            int anonIdFilter = context.getConfiguration().getInt(ANON_ID, 0);
            int anonId = 0;

            String[] columns = searchLogLine.toString().split("\t");

            try {
                anonId = Integer.parseInt(columns[0]);
            } catch (NumberFormatException e) {
                return; // The data in the first column is not an integer, read the
            }

            if (anonIdFilter != anonId) return;

            String id = "";
            String query = "";
            String itemRank = "";
            String clickUrl = "";

            try {
                id = columns[0];
                query = columns[1];
                itemRank = columns[3];
                clickUrl = columns[4];
            } catch (ArrayIndexOutOfBoundsException e) {
            }

            String result = id + ", " + query + ", " + itemRank + ", " + clickUrl;

            context.write(new Text(id), new Text(result));
        }
    }

    public static class TaskOneReducer extends Reducer<Text, Text, Text, Text> {

        protected void reduce(Text anonId, Iterable<Text> searchResults, Context cor

            StringBuilder csvExport = new StringBuilder();

            for (Text result : searchResults)
                csvExport
                        .append(result.toString())
                        .append("\n");

            String header = "ANON_ID, QUERY, ITEM_RANK, CLICK_URL\n";

            context.write(new Text(header), new Text(csvExport.toString()));
        }

    }

    public int run(String[] args) throws Exception {

        validateParams(args);
        Path inputDir = new Path(args[0]);
        Path outputDir = new Path(args[1]);
        int anonId = parseAnonId(args[2]);

        deleteFilesInDirectory(outputDir);

        Job job = Job.getInstance(getConf(), JOB_NAME);
        job.setJarByClass(TaskOne.class);

        FileInputFormat.setInputPaths(job, inputDir);
        FileOutputFormat.setOutputPath(job, outputDir);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(TaskOneMapper.class);
        job.setCombinerClass(TaskOneReducer.class);
        job.setReducerClass(TaskOneReducer.class);

        Configuration conf = job.getConfiguration();
        conf.setInt(ANON_ID, anonId);

        return job.waitForCompletion(false) ? 0 : -1;
    }

    public static void main(String[] args) throws Exception {
        System.exit(ToolRunner.run(new Configuration(), new TaskOne(), args));
    }

    private static void validateParams(String[] args) {

        int numberOfParams = args.length;

        if (numberOfParams != 3) {
            throw new IllegalArgumentException("TaskOne expects exactly 3 params: ir
        }
    }

    private static int parseAnonId(String anonIdParam) {

        int anonId;

        try {

            anonId = Integer.parseInt(anonIdParam);
            if (anonId <= 0) throw new NumberFormatException();

        } catch (NumberFormatException e) {
            throw new IllegalArgumentException("Invalid ANON_ID. It must be a positi
        }

        return anonId;
    }

    private static void deleteFilesInDirectory(Path f) throws IOException {

        Configuration config = new Configuration();
        FileSystem hdfs = FileSystem.get(config);

        if (!hdfs.delete(f, true))
            throw new FileNotFoundException("Failed to delete file: " + f);
    }
}
```

Build the project:

```
$ cd ./aol
$ mvn clean package
```

Run:

```
mvn exec:java -Dexec.mainClass="nz.zoltan.TaskOne" -Dexec.args="jobs/aol jobs/task-c
```

Notes for the source code:

- Using the latest mapreduce API.
- In the `main` method, we call the `ToolRunner` for running our map-reduce task.
- There are three utility functions: `validateParams` for command param validation, `parseAnonId` for validation of the given `anonId` param and `deleteFilesInDirectory` which will delete the output directory (it is from our tutorial's `PatentCitation.java`)
- `TaskOneMapper` class implements our `map` method, which iterates through our input file, split the search log to columns. Ignore lines without valid `AnonId` and ignore lines if they do not have the requested id (`anonIdFilter`).
- The code using the `context` configuration object to pass command line param to the map task. `context.getConfiguration().getInt(ANON_ID, 0)`
- The output from the `map` method is a list of key value pair, where key is the requested `id` and the value is our comma concatenated string in the requested format.

- `TaskOneReducer` class's `reduce` function will get the above generated list. Because the `anonId` is the same, it will get all the connected result as the value iterator. With a simple `for` we concatenate it to a single csv file.

PART 2 - TASK 2

Collect summary statistics. In particular:

```
(1) number of searches;
(2) number of unique users;
(3) overall number of clicks.
```

Source code for this solution placed in `./aol/src/java/nz/zoltan/TaskTwo.java`:

```
package nz.zoltan;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.FileNotFoundException;
import java.io.IOException;

public class TaskTwo extends Configured implements Tool {
    private static final String NUMBER_OF_SEARCH_JOB_NAME = "task2-searches";
    private static final String NUMBER_OF_USERS_JOB_NAME = "task2-users";
    private static final String NUMBER_OF_CLICKS_JOB_NAME = "task2-clicks";

    // Using for counting simple rows
    private static final LongWritable ONE = new LongWritable(1);

    /**
     * Generates a simple KEY -> VALUE list, where key is a unique ID generated by H
```

```java
     * so finally the reducer is a simple counter.
     */
    public static class NumberOfSearchMapper extends Mapper<LongWritable, Text, Text

        protected void map(LongWritable key, Text searchLogLine, Context context) th

            String[] columns = searchLogLine.toString().split("\t");

            try {
                Integer.parseInt(columns[0]);
            } catch (NumberFormatException e) {
                return; // The data in the first column is not an integer, read the
            }

            // This is a basic line counter implementation, if a search logged, we s
            // to the reducer, so the reducer can simply just sum how many rows we h
            context.write(new Text(key.toString()), ONE);
        }
    }

    public static class NumberOfSearchReducer extends Reducer<Text, LongWritable, Te

        Long numberOfSearchCounter = new Long(0);

        protected void reduce(Text key, Iterable<LongWritable> ones, Context context

            for (LongWritable one : ones) {
                numberOfSearchCounter++;
            }

            String message = "Number of Searches: ";

            context.write(new Text(message), new LongWritable(numberOfSearchCounter)
        }

    }

    /**
     * For counting the number of unique users we need a Mapper, a Combiner and a Re
     * Mapper creates a KEY -> VALUE list, where KEY is the userId (ANON_ID), and th
     * <p>
     * The Combiner can work as an aggregator, because the reducer function will be
     * Basically, we have to count, how many times the combiner was called.
     * <p>
     * From the combiner, we just generate a list of "users" -> 1 list, so a simple
     * requested value. This step can be done in the last reducer call.
     */
    public static class NumberOfUsersMapper extends Mapper<LongWritable, Text, Text,
        protected void map(LongWritable key, Text searchLogLine, Context context) th

            String[] columns = searchLogLine.toString().split("\t");

            Integer userId = 0;

            try {
                userId = Integer.parseInt(columns[0]);
            } catch (NumberFormatException | ArrayIndexOutOfBoundsException e) {
                return; // The data in the first column is not an integer or there i
            }

            context.write(new Text(userId.toString()), ONE);
        }
    }

    public static class NumberOfUsersCombiner extends Reducer<Text, LongWritable, Te

        protected void reduce(Text userId, Iterable<LongWritable> ones, Context cont
            String key = "users";
            context.write(new Text(key), ONE);
        }
    }

    public static class NumberOfUsersReducer extends Reducer<Text, LongWritable, Tex

        protected void reduce(Text userId, Iterable<LongWritable> ones, Context cont

            Long numberOfUsersCounter = 0L;

            for (LongWritable ignored : ones) {
                numberOfUsersCounter++;
            }

            String message = "Number of Users: ";

            context.write(new Text(message), new LongWritable(numberOfUsersCounter))
        }
    }

    /**
     * Counting clicks logic is similar to counting searches, the only differences i
     * <p>
     * If a line does not have itemRank data, it means, it is not a click, so we igr
     */
    public static class NumberOfClicksMapper extends Mapper<LongWritable, Text, Text
```

```
        protected void map(LongWritable key, Text searchLogLine, Context context) th

            String[] columns = searchLogLine.toString().split("\t");

            try {
                Integer.parseInt(columns[0]);
            } catch (NumberFormatException e) {
                return; // The data in the first column is not an integer, read the
            }

            String itemRank = "";

            try {
                itemRank = columns[3];
            } catch (ArrayIndexOutOfBoundsException e) {
                return; // No itemRank, no click, try the next.
            }

            // If the search log line contains an itemRank than it was clicked, so w
            context.write(new Text(key.toString()), ONE);
        }
    }

    public static class NumberOfClicksReducer extends Reducer<Text, LongWritable, Te

        Long numberOfClicksCounter = 0L;

        protected void reduce(Text key, Iterable<LongWritable> ones, Context context

            for (LongWritable ignored : ones) {
                numberOfClicksCounter++;
            }

            String message = "Number of Clicks: ";

            context.write(new Text(message), new LongWritable(numberOfClicksCounter)
        }
    }

    public int run(String[] args) throws Exception {

        validateParams(args);

        Path inputDir = new Path(args[0]);
        Path outputDir = new Path(args[1]);

        deleteFilesInDirectory(outputDir);
        Path searchOutputDir = new Path(outputDir + "/search");
        Path usersOutputDir = new Path(outputDir + "/users");
        Path clicksOutputDir = new Path(outputDir + "/clicks");


        Job numberOfSearchJob = Job.getInstance(getConf(), NUMBER_OF_SEARCH_JOB_NAME
        Job numberOfUsersJob = Job.getInstance(getConf(), NUMBER_OF_USERS_JOB_NAME);
        Job numberOfClicksJob = Job.getInstance(getConf(), NUMBER_OF_CLICKS_JOB_NAME
        numberOfSearchJob.setJarByClass(TaskTwo.class);
        numberOfUsersJob.setJarByClass(TaskTwo.class);
        numberOfClicksJob.setJarByClass(TaskTwo.class);

        FileInputFormat.setInputPaths(numberOfSearchJob, inputDir);
        FileInputFormat.setInputPaths(numberOfUsersJob, inputDir);
        FileInputFormat.setInputPaths(numberOfClicksJob, inputDir);

        FileOutputFormat.setOutputPath(numberOfSearchJob, searchOutputDir);
        FileOutputFormat.setOutputPath(numberOfUsersJob, usersOutputDir);
        FileOutputFormat.setOutputPath(numberOfClicksJob, clicksOutputDir);

        numberOfSearchJob.setInputFormatClass(TextInputFormat.class);
        numberOfUsersJob.setInputFormatClass(TextInputFormat.class);
        numberOfClicksJob.setInputFormatClass(TextInputFormat.class);

        numberOfSearchJob.setOutputFormatClass(TextOutputFormat.class);
        numberOfUsersJob.setOutputFormatClass(TextOutputFormat.class);
        numberOfClicksJob.setOutputFormatClass(TextOutputFormat.class);

        numberOfSearchJob.setOutputKeyClass(Text.class);
        numberOfUsersJob.setOutputKeyClass(Text.class);
        numberOfClicksJob.setOutputKeyClass(Text.class);

        numberOfSearchJob.setOutputValueClass(LongWritable.class);
        numberOfUsersJob.setOutputValueClass(LongWritable.class);
        numberOfClicksJob.setOutputValueClass(LongWritable.class);

        numberOfSearchJob.setMapperClass(NumberOfSearchMapper.class);
        numberOfSearchJob.setCombinerClass(NumberOfSearchReducer.class);
        numberOfSearchJob.setReducerClass(NumberOfSearchReducer.class);

        numberOfUsersJob.setMapperClass(NumberOfUsersMapper.class);
        numberOfUsersJob.setCombinerClass(NumberOfUsersCombiner.class);
        numberOfUsersJob.setReducerClass(NumberOfUsersReducer.class);

        numberOfClicksJob.setMapperClass(NumberOfClicksMapper.class);
        numberOfClicksJob.setCombinerClass(NumberOfClicksReducer.class);
        numberOfClicksJob.setReducerClass(NumberOfClicksReducer.class);
        return (numberOfSearchJob.waitForCompletion(false)
                && numberOfUsersJob.waitForCompletion(false)
                && numberOfClicksJob.waitForCompletion(false))
                ? 0 : -1;
    }

    public static void main(String[] args) throws Exception {
        System.exit(ToolRunner.run(new Configuration(), new TaskTwo(), args));
    }

    private static void validateParams(String[] args) {

        int numberOfParams = args.length;

        if (numberOfParams != 2) {
            throw new IllegalArgumentException("TaskTwo expects exactly 2 params: ir
        }
    }

    private static void deleteFilesInDirectory(Path f) throws IOException {

        Configuration config = new Configuration();
        FileSystem hdfs = FileSystem.get(config);

        if (!hdfs.delete(f, true))
            throw new FileNotFoundException("Failed to delete file: " + f);
    }
}
```

Implementation notes:

- One of the challenges was to solve, how can we keep counters up to date between reducer tasks. Playing with it, I realized, we cannot share state, because reducer tasks run parallel, however Hadoop solve this problem for us.
- In this implementation our code fires 3 separate jobs for each expected calculation: `numberOfSearchJob`, `numberOfUsersJob`, `numberOfClicksJob`.
- There are mapper, combiner and reducer tasks.
- Each task has own Mapper, Combiner and Reducer class.
- There are more comments in the code, please check the details there.

PART 2 - TASK 3

For parallel computing, the optimal speedup gained through parallelization is linear with respect to the number of jobs running in parallel. For example, with 5 reducers, ideally we would expect parallel computing to take 1/5 wall clock time of single machine run. However, this optimal speedup is usually not achievable. In this question, set the number of reducers (job.setNumReduceTasks()) in your Hadoop run to 2, 4, 6 and record the wall clock time. Plot a curve, where the horizontal axis is the number of reducers, and the vertical axis is the wall time. Is the wall time linear with respect to the number of reducers? Explain what you observed. You will get the necessary output when you run your job.

In this task I used the same code what was written in Task 2. There is only one change. This program can accept an extra param, which modify the `numReduceTask` option.

For this experiment I used the `time` command to measure the run time of a complex task:

```
$ cd ./aol

$ time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/aol jo
mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree"   314.54s user 17.63s system 11
```

Running the experiment with smaller sample package. Only 2 files from the original aol folder copied to `jobs/short-aol` folder. "` time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/short-aol jobs/task-three/output 1" mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" 69.19s user 3.88s system 124% cpu 58.754 total

time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/short-aol jobs/task-three/output 2" mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" 68.53s user 4.03s system 124% cpu 58.299 total

time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/short-aol jobs/task-three/output 3" mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" 68.03s user 4.04s system 123% cpu 58.251 total

time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/short-aol jobs/task-three/output 4" mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" 69.04s user 3.96s system 122% cpu 59.549 total "`

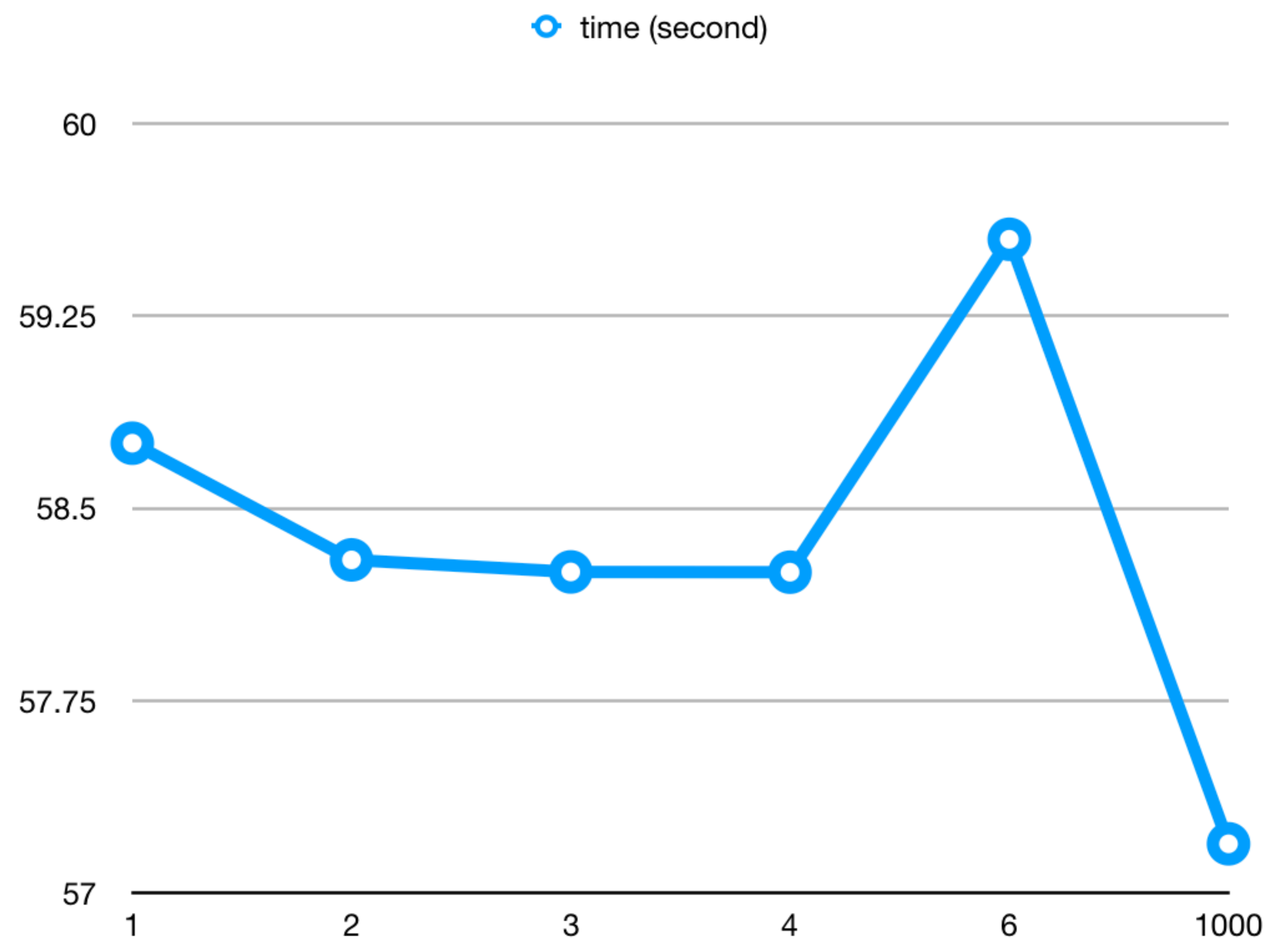
The above commands used mvn, the following used hadoop with jar…

```
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/short-aol  6  62.86s
```

And an extreme number, using `1000` for `numReduceTasks`.

```
$ time mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree" -Dexec.args="jobs/short-
mvn exec:java -Dexec.mainClass="nz.zoltan.TaskThree"   84.19s user 9.74s system 100%
```

```
1 -> 58.754s
2 -> 58.299s
3 -> 58.252s
4 -> 58.251s
6 -> 59.549s
1000 -> 57.192s
```

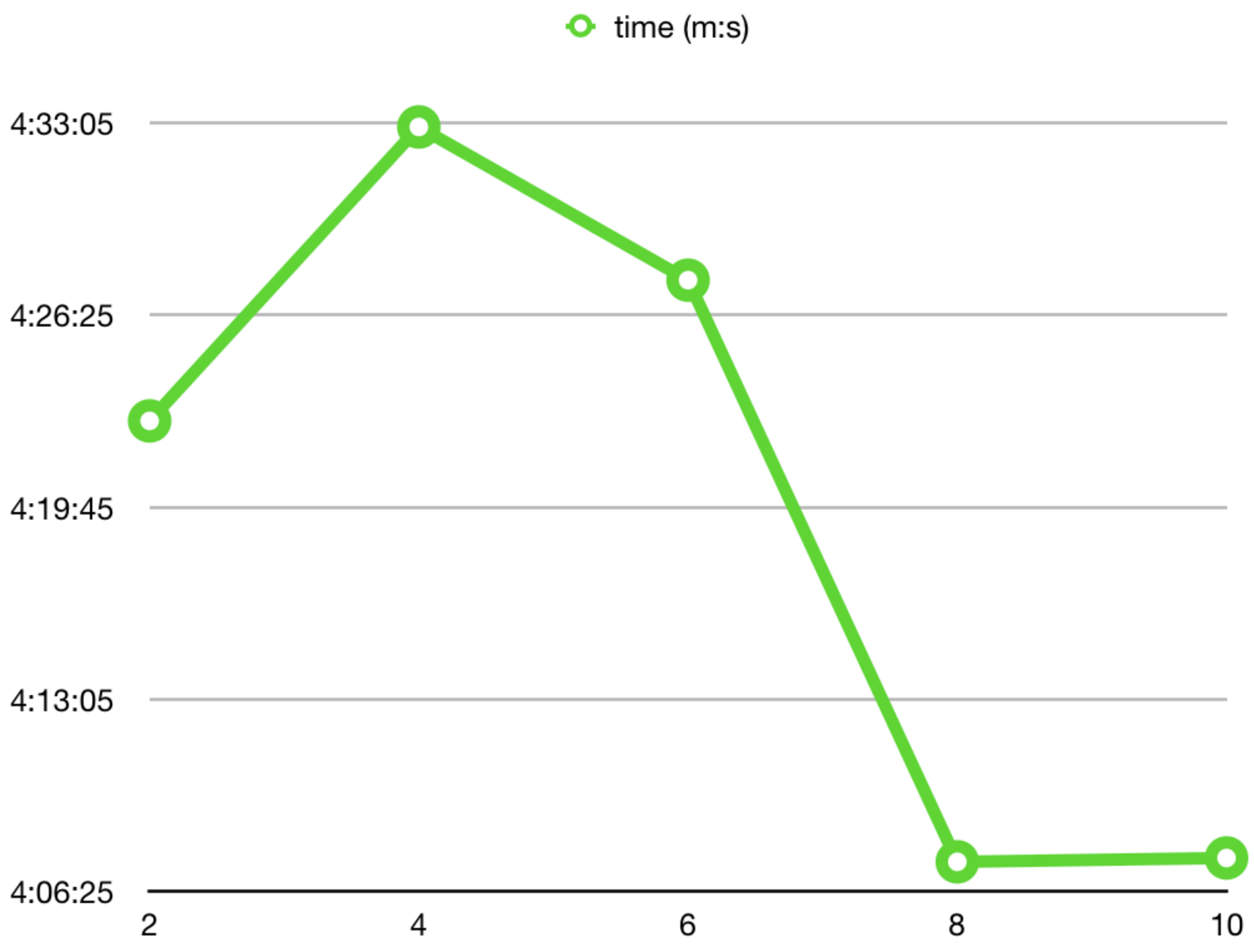


For the full experiment you can run the following command. It will build the latest version of the jar file and run the whole calculation

```
mvn clean package && time hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree
time hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol jobs/task-t
time hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol jobs/task-t
time hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol jobs/task-t
time hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol jobs/task-t
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol  2 &>   295.81s
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol  4 &>   298.93s
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol  6 &>   301.78s
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol  8 &>   280.92s
hadoop jar target/aol-1.0-SNAPSHOT.jar nz.zoltan.TaskThree jobs/aol  10 &>   285.72s
```

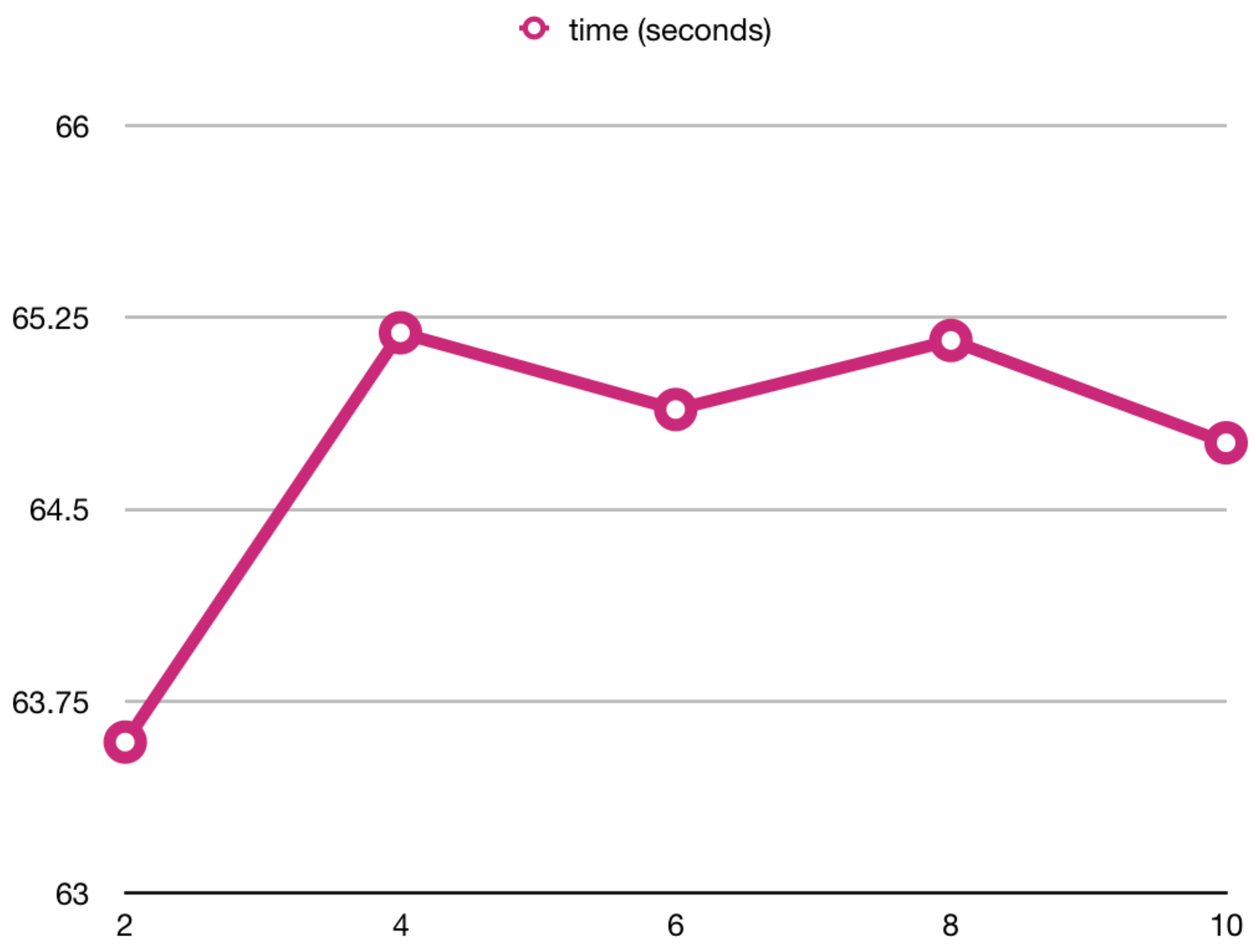
From the above experiment:

```
2 -> 4:22.45
4 -> 4:32.57
6 -> 4:27.38
8 -> 4:07.27
10 -> 4:07.35
```



Running map reduce only on a smaller sample again:

```
2 -> 1:03.59 -> 63.59s
4 -> 1:05.19 -> 65.19s
6 -> 1:04.89 -> 64.89s
8 -> 1:05.16 -> 65.16s
10 -> 1:04.76 -> 64.76s
```

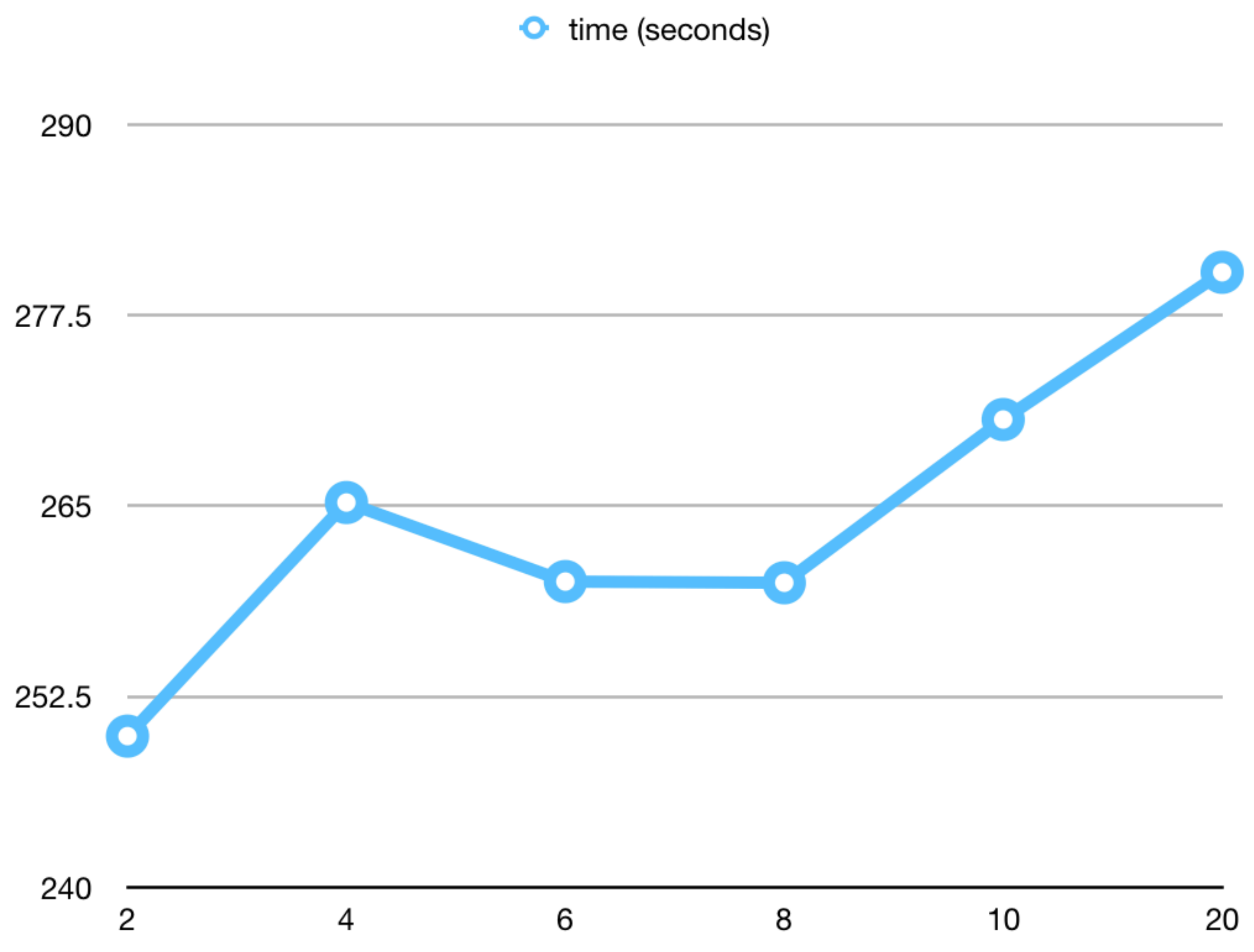


For the above numbers, I run the following command using Docker.

```
time docker exec -it nwen406 sh -c 'cd /aol && mvn exec:java -Dexec.mainClass="nz.zc
```

Running map reduce on the full AOL search sample again:

```
2 -> 4:09.91 -> 249.91s
4 -> 4:25.22 -> 265.22s
6 -> 4:20.04 -> 260.04s
8 -> 4:19.96 -> 259.96s
10 -> 4:30.66 -> 270.66s
20 -> 4:40.31 -> 280.31s
```



Using the following command:

```
time docker exec -it nwen406 sh -c 'cd /aol && mvn exec:java -Dexec.mainClass="nz.zc
```

Observations and opinion:

- We can see in our `output` folder, that the result is fragmented, instead of having a clear total number, we still should sum up the partial results.
- The above experiments shows, that we cannot increase the performance and run our MapReduce task faster if we increase the paralellization. Sometimes we see the performance decreasing.
- I think, our limitation is in the hardware. For running more processes on the same computer cannot increase performance, because we still use the same amount of processor power and memory.
- Parallel computing can be a solution if we can share processes between additional computers or virtual machines with additional processor power and memory.
- Hadoop is heavily optimized for performance. It tries to use all the available resource. This is why we see quite similar results in each cases.

Bonus Tasks

Unit testing Hadoop jobs

- https://dzone.com/articles/unit-testing-java-hadoop-job

Wrapping the whole project in Docker

Setup environment using Docker.

Useful links:

```
* [Install Docker on Ubuntu](https://docs.docker.com/engine/installation/linux/docke
* [Install Docker on Windows](https://docs.docker.com/docker-for-windows/install/)
* [Install Docker on Mac](https://docs.docker.com/docker-for-mac/install/)
```

Some useful commands:

```
$ docker build -t zoltannz/hadoop:0.1 .
$ docker images
```

We can run the container in daemon mode, so it stays in the background. In this case, we can connect to it with `docker container exec -it` command.

```
$ docker run -dit zoltannz/hadoop:0.1
$ docker container ls
$ docker container exec -it {name-of-the-instance} bash
```

Other way to run the image using the `-it` params. In this case we have to pass a runnable script at the end of the command line. In the following example we run the bash, however, when we exit from the bash, the container will stop.

```
$ docker run -it zoltannz/hadoop:0.1 /bin/bash
```

Creating a standard Hadoop docker image

Unfortunately, the official Docker images for Hadoop are not maintained regularly. My goal was to use the latest official Ubuntu base with latest Java 8 and with Hadoop 2.8.1.

Official docker image repositories:

- https://github.com/sequenceiq/docker-hadoop-ubuntu
- https://github.com/sequenceiq/hadoop-docker

Based on the above `Dockerfile` , I created a new one:

- https://github.com/zoltan-nz/docker-hadoop-ubuntu

A built image on my DockerHub:

- https://hub.docker.com/r/zoltannz/hadoop-ubuntu/

You can download the above image with the following command:

```
$ docker pull zoltannz/hadoop-ubuntu:2.8.1
```

It uses the following `Dockerfile` :

```
# Original repository https://github.com/sequenceiq/docker-hadoop-ubuntu
#
# Update to Java 8 and Hadoop 2.8.1 using the latest stable Ubuntu
#
# docker build -t zoltannz/hadoop-ubuntu:2.8.1 .
# docker pull zoltannz/hadoop-ubuntu:2.8.1

FROM ubuntu

USER root

# install dev tools
RUN apt-get update
RUN apt-get install -y curl tar sudo openssh-server openssh-client rsync

# passwordless ssh
RUN rm -f /etc/ssh/ssh_host_dsa_key /etc/ssh/ssh_host_rsa_key /root/.ssh/id_rsa
RUN ssh-keygen -q -N "" -t dsa -f /etc/ssh/ssh_host_dsa_key
RUN ssh-keygen -q -N "" -t rsa -f /etc/ssh/ssh_host_rsa_key
RUN ssh-keygen -q -N "" -t rsa -f /root/.ssh/id_rsa
RUN cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys

# java
RUN apt-get install -y default-jdk

ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64/jre
ENV PATH $PATH:$JAVA_HOME/bin
```

```
# hadoop
RUN curl -s http://www-us.apache.org/dist/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.ta
RUN cd /usr/local && ln -s ./hadoop-2.8.1 hadoop

ENV HADOOP_PREFIX /usr/local/hadoop
ENV HADOOP_COMMON_HOME /usr/local/hadoop
ENV HADOOP_HDFS_HOME /usr/local/hadoop
ENV HADOOP_MAPRED_HOME /usr/local/hadoop
ENV HADOOP_YARN_HOME /usr/local/hadoop
ENV HADOOP_CONF_DIR /usr/local/hadoop/etc/hadoop
ENV YARN_CONF_DIR $HADOOP_PREFIX/etc/hadoop

RUN sed -i '/^export JAVA_HOME/ s:.*:export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-am
RUN sed -i '/^export HADOOP_CONF_DIR/ s:.*:export HADOOP_CONF_DIR=/usr/local/hadoop/
RUN . $HADOOP_PREFIX/etc/hadoop/hadoop-env.sh

RUN mkdir $HADOOP_PREFIX/input
RUN cp $HADOOP_PREFIX/etc/hadoop/*.xml $HADOOP_PREFIX/input

# pseudo distributed
ADD core-site.xml.template $HADOOP_PREFIX/etc/hadoop/core-site.xml.template
RUN sed s/HOSTNAME/localhost/ /usr/local/hadoop/etc/hadoop/core-site.xml.template >
ADD hdfs-site.xml $HADOOP_PREFIX/etc/hadoop/hdfs-site.xml

ADD mapred-site.xml $HADOOP_PREFIX/etc/hadoop/mapred-site.xml
ADD yarn-site.xml $HADOOP_PREFIX/etc/hadoop/yarn-site.xml

RUN $HADOOP_PREFIX/bin/hdfs namenode -format

ADD ssh_config /root/.ssh/config
RUN chmod 600 /root/.ssh/config
RUN chown root:root /root/.ssh/config

ADD bootstrap.sh /etc/bootstrap.sh
RUN chown root:root /etc/bootstrap.sh
RUN chmod 700 /etc/bootstrap.sh

ENV BOOTSTRAP /etc/bootstrap.sh

# workingaround docker.io build error
RUN ls -la /usr/local/hadoop/etc/hadoop/*-env.sh
RUN chmod +x /usr/local/hadoop/etc/hadoop/*-env.sh
RUN ls -la /usr/local/hadoop/etc/hadoop/*-env.sh

# fix the 254 error code
RUN sed  -i "/^[^#]*UsePAM/ s/.*/#&/"  /etc/ssh/sshd_config
RUN echo "UsePAM no" >> /etc/ssh/sshd_config
RUN echo "Port 2122" >> /etc/ssh/sshd_config

RUN service ssh start && $HADOOP_PREFIX/etc/hadoop/hadoop-env.sh && $HADOOP_PREFIX/s
RUN service ssh start && $HADOOP_PREFIX/etc/hadoop/hadoop-env.sh && $HADOOP_PREFIX/s

CMD ["/etc/bootstrap.sh", "-bash"]

# Hdfs ports
EXPOSE 50010 50020 50070 50075 50090 8020 9000
# Mapred ports
EXPOSE 10020 19888
#Yarn ports
EXPOSE 8030 8031 8032 8033 8040 8042 8088
#Other ports
EXPOSE 49707 2122 22
```

Using our standard Hadoop docker image in our assignment project

Create a `Dockefile` in the main folder:

```
FROM zoltannz/hadoop-ubuntu:2.8.1

RUN apt-get install maven -y

# Define mountable directories.
VOLUME ["/aol"]
VOLUME ["/wordcount"]

# Define working directory.
WORKDIR /
```

We can use our previously distributed image. Because our project uses `maven` we install it in an extra step. Furthermore we create a two now volumes, where we can attache our external source code, so we can run our code inside the docker instance however it will use the external code.

When we run the created image, we have to map together our external folder with the created map.

Build the image first. Let's call the image `zoltannz/hadoop-nwen406` and use a version number.

```
$ docker build -t zoltannz/hadoop-nwen406:0.1 .
```

Run the image using the `-v` param to attache folder as volume. Please note, the path on your local computer has to be absolute path not relative.

```
$ docker run -ti --name nwen406 -v "$PWD/wordcount":/wordcount -v "$PWD/aol":/aol zc
```

(On Windows you cannot use `$PWD` .)

The above command launches the docker instance and automatically enters in bash, however we can use `-d` to launch it as a deamon.

Please note, we use `--name nwen406` , so we can access with this name to our running instance.

This instance can be stopped and restarted

```
$ docker stop nwen406
$ docker start nwen406
```

Let's run a few command in our container.

Getting a bash prompt:

```
$docker exec -it nwen406 bash
```

Running `mvn install` or `mvn exec` :

```
$ docker exec -it nwen406 sh -c 'cd /wordcount && mvn install'
$ docker exec -it nwen406 sh -c 'cd /wordcount && mvn exec:java -Dexec.mainClass="nz
```

Run Part 2 - Task 1:

```
$ docker exec -it nwen406 sh -c 'cd /aol && mvn clean install'
$ docker exec -it nwen406 sh -c 'cd /aol && mvn exec:java -Dexec.mainClass="nz.zolta
```

Run Part 2 - Task 2:

In this case we wanna see how long does it take, using `time` :

```
$ time docker exec -it nwen406 sh -c 'cd /aol && mvn exec:java -Dexec.mainClass="nz.

docker exec -it nwen406 sh -c   0.06s user 0.08s system 0% cpu 4:33.92 total
```