UNIVERSITY OF TARTU
Institute of Computer Science

(/)

Basics of Cloud Computing 2018/19 spring

Cus

- Main (https://courses.cs.ut.ee/2019/cloud/spring/Main/HomePage)
- Lectures (https://courses.cs.ut.ee/2019/cloud/spring/Main/Lectures)
- Practicals (https://courses.cs.ut.ee/2019/cloud/spring/Main/Practicals)
- Results (https://courses.cs.ut.ee/2019/cloud/spring/Main/Results)
- Submit Homework (https://courses.cs.ut.ee/2019/cloud/spring/Main/Homework)

Practice 4 - Introduction to MapReduce

In this Practice session we will start working with the Apache Hadoop MapReduce framework. You will learn how to set up and configure a MapReduce environment in your computer using either IntelliJ IDEA without having to install Hadoop on your computer. Then you will study example MapReduce programs source code, introduce a number of improvements and execute it in a pseudo distributed manner in your computer.

References

Useful links and documents that contain supportive information for the practice.

Manuals

- MapReduce tutorial: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html)
- Hadoop API: http://hadoop.apache.org/docs/stable/api/ (http://hadoop.apache.org/docs/stable/api/)
- IntelliJ IDEA: https://www.jetbrains.com/idea/ (https://www.jetbrains.com/idea/)
- Hadoop wiki https://hadoop.apache.org/ (https://hadoop.apache.org/)

Additional information

- List of companies using Hadoop and their use cases https://wiki.apache.org/hadoop/PoweredBy (https://wiki.apache.org/hadoop/PoweredBy)
- List of Hadoop tools, frameworks and engines https://hadoopecosystemtable.github.io/ (https://hadoopecosystemtable.github.io/)
- Commercial Hadoop distributions and support https://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support (https://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support)

Exercise 4.1. Installing prerequisites for Hadoop

A good IDE can help a lot when programming in any new language or a framework. In this exercise you will set up IDE and configure your computer so that you can run Hadoop MapReduce programs in local machine mode without actually having to install Hadoop.

- Install IntelliJ IDEA if don't have them already installed on your laptop.
  - You can download "IntelliJ IDEA Community eddition" from https://www.jetbrains.com/idea/ (https://www.jetbrains.com/idea/)
- You also need to install **Java SDK, Version 8** (if you don't yet have it installed)
- Download `hadoop-2.9.2-src.tar.gz` and `hadoop-2.9.2.tar.gz` files from: Hadoop downloads (https://archive.apache.org/dist/hadoop/common/hadoop-2.9.2/)
- Unpack the downloaded files on your hard disk.
  - In Windows you can use 7zip or WinRAR to unpack them
  - In Linux, you can use the `tar xf hadoop-2.9.2-src.tar.gz` command to unpack them

Additional tasks if you are using Windows (Otherwise skip this section)

- NB! You will run into problems while running Hadoop MapReduce programs if your Windows username includes a space character. We would suggest that you create an alternative user in your computer without any white space characters and complete the exercise under that user.
  - You should also avoid spaces in folder paths.
- Download and install Cygwin (http://cygwin.com/install.html)
  - Cygwin allows Hadoop MapReduce libraries to run Linux likes commands to manage files and folders in Windows machines.
  - Installation location should be: `C:\cygwin64`
  - Continue with the following tasks while Cygwin is being installed, but make sure to check the progress every once in a while, because it will ask you specify additional options during installation process
- Setup Windows Environment Variables for Cygwin.
  - Open your System Control Panel: **Control Panel** -> **System** -> **Advanced System Settings**
  - Find "**Environmental Variables**" Under the tab "**Advanced**"
  - Modify **PATH** under System variables:
    - Windows 8: Add the string `";C:\cygwin64\bin"` at the end of the PATH variable
    - Windows 10: Add a new string `"C:\cygwin64\bin"` to the list of PATH values
    - **NB!** Make sure the installation path is really "C:\cygwin64\" (It might be C:\cygwin\ or some other folder instead)
    - If this does not work for you, try using a Linux Virtual Machine or completing the exercises on lab computers.
- Download windows utilities required for Hadoop.
  - Newer Hadoop libraries require additional native Windows libraries to be built which are not distributed with the Hadoop binaries by default. Its possible to build them from the Hadoop source code, but we will download pre-built versions instead to save time.
  - Download `hadoop-2.8.1.zip` from GitHub repository (https://github.com/steveloughran/winutils/releases)
  - Unpack the container, scan it with anti virus and copy its content (only files) into the created `hadoop-2.9.2/bin` folder. (**NB!** not the source folder!)
- Setup Windows Environment Variables for Hadoop.
  - Open your System Control Panel: **Control Panel** -> **System** -> **Advanced System Settings**
  - Find "**Environmental Variables**" Under the tab "**Advanced**"
    - Add a new **User variable** named `HADOOP_HOME`. Its value should be the full path to the unpacked `hadoop-2.9.2.tar.gz` Hadoop folder (NB! Not the source folder) inside your computer.
      - For example its value should look something like this: `C:\Users\jakovits\Downloads\hadoop-2.9.2.tar\hadoop-2.9.2\`
    - Modify the USER **PATH** variable by adding the path to the `bin` folder inside the

unpacked Hadoop folder.
    - Windows 8: Add the string `;%HADOOP_HOME%\bin` at the end of the USER PATH variable.
    - Windows 10: Add a new string `%HADOOP_HOME%\bin` to the list of USER PATH values.
- If you get an error about `MSVCR100.dll` then you may have to download and install **Microsoft Visual C++ 2010 Redistributable Package** (It should be 64 bit version if you're using 64bit OS)

Exercise 4.2 Configuring IntelliJ IDEA for Hadoop

We will import and configure the MapReduce example Java Project into IntelliJ IDEA.

- Start the IntelliJ IDEA application
- Choose **import Project from existing sources**
- Import project from the `hadoop-mapreduce-project\hadoop-mapreduce-examples\pom.xml` file inside the Hadoop source folder you have previously unpacked.
- NB! Make sure to select `Import Maven project automatically`
- Wait until Maven has finished configuring the project dependencies.

Exercise 4.3. Running the WordCount example in IntelliJ IDEA

In this exercise you will learn how to execute the MapReduce WordCount example through the IntelliJ IDEA. You will also configure the input and output directories for WordCount.

- Find the WordCount class inside your project (src/main/java/org.apache.hadoop.examples/)
- Create a new folder named **input** inside your project. We will put all the input files for the WordCount MapReduce application there.
- Download 5 random books from Gutenberg in text (UTF-8) format:
    - http://www.gutenberg.org/ebooks/search/?sort_order=random (http://www.gutenberg.org/ebooks/search/?sort_order=random)
- Move the downloaded text files into the input folder.
- Lets also fix the output of the program. Currently it does not output any information about the executed program. Add the following two lines at the start of the `main()` method:

```
org.apache.log4j.BasicConfigurator.configure();
org.apache.log4j.Logger.getRootLogger().setLevel(org.apache.log4j.Level.INFO);
```

- Try to run the `main()` method in this class as a Java application.
    - Right click on the WordCount class and choose `Run 'Wordcount.main()'`
    - You will initially see an error concerning the number of supplied arguments.
- Modify the run configuration of the WordCount class to change what arguments should be supplied to it.
    - Go to `Run - > Edit Configurations` and select the WordCount class run configuration-
    - WordCount class requires two **Program arguments**: `input` folder and `output` folder
    - Specify the previously created folder (where you moved Gutenberg books) as input folder and an arbitrarily named folder as output
    - When using relative folder paths, folders are created inside the project main folder.
- NB! Modify the run configuration of the WordCount class to also have the `Include the dependencies with the "Provided" scope` option activated. This will make sure that all the **compile-time** only Maven dependencies are also available at **run-time**.
- If the execution is successful, output will be written into part-r-00000 file inside the **output** folder
    - If you run the application again, output folder must first be deleted, moved or re-configured as a new folder.

Exercise 4.4. Modifying the WordCount example.

Your goal in this task is to learn how to modify existing MapReduce applications. You will improve how the WordCount application splits lines of text into words and will change how the data is grouped between Map and Reduce tasks.

One of the main problems with the simple MapReduce WordCount example is that it does not remove punctuation marks and other special characters and as a result it counts words like `cat` and `cat.` separately.

We will also change how the data is grouped in the **Reduce** task. Initially all unique words are grouped together and their `Sum` is computed. In the second task of this exercise we will change the WordCount to calculate the `count of words for each input file` separately by simply changing how the data is grouped in the **Reduce** task.

1. Modify the WordCount **Map** method to `remove punctuation marks` (.,:;# etc.) and try to get it to output only clear `lowercase words`.
2. Modify the program so that instead of calculating **global** count of words, it calculates the count of words for each file separately.
   - Instead of the current output (`word, count`) of the application we want the output to be `word, file, count`, where the `count` is the number of `word` occurrences in `file`.
   - We have to change how the data is grouped between Map and Reduce tasks. Currently is simply grouped by unique words.
   - Change the map output **key** to also include the file name (`word;filename`)
   - You can find the name of the file from which the input it taken by using the following methods:
     - `FileSplit split = (FileSplit) context.getInputSplit();`
     - `String filename = split.getPath().toString();`

Bonus exercise

The goal of the bonus task is to investigate the other available MapReduce example applications with the goal to introduce additional improvements.

- Look through the other available MapReduce examples (which you previously unpacked) and **choose one** of them (other than **WordCount**) which you think could be improved in some way, like we did with WordCount.
  - However, the improvements **must be different** from the changes we did in **Exercise 4.4**.
    - For example: removing punctuation marks in a different MapReduce example will not be sufficient!
- `Document` all the made changes and `explain` why do you think these changes are beneficial.
- Submit the modified Java class file and your comments to get the bonus credit points.

Deliverables:

- Modified WordCount.java file
- Output of the modified program (part-r-00000 file)
- Answer the following questions:
  - Why is the `IntSumReducer` class in the WordCount example used not only as reducer (`job.setReducerClass`) but also as a combiner (`job.setCombinerClass`)
  - What advantages (if any) does it provide?
  - Can all possible Reduce functions be used directly as combiners?

Task: lab 4

Current submission: ZIP (/course-2065/submissions/get/4/B91541_zip/B91541_4300_1)

(02:23 13.03.2019)

If your homework consists of multiple files (for example HTML + CSS + JS) it should be archived before submitting.

File: Choose File no file selected

Comment: Submission of lab 4! Thanks!

Submit

Institute of Computer Science (http://cs.ut.ee/) | Faculty of Science and Technology (http://reaalteadused.ut.ee/) | University of Tartu (http://www.ut.ee/)

In case of technical problems or questions write to: ati.error@ut.ee (mailto:ati.error@ut.ee)

The courses of the Institute of Computer Science are supported by following programs: