CPrE/SE 419: SOFTWARE TOOLS FOR LARGE-SCALE DATA ANALYSIS, SPRING 2022

LAB #7: FLINK

Purpose

In this lab, the goal is to learn how to use Apache Flink. The practice-aspect will be to solve some related problems in a streaming fashion – i.e., in this lab, you will write programs with pipelined jobs to some real world problems that can be solved using Flink.

Submission

Create a single zip archive with the following and hand it in through Canvas:

- Reports including screenshots of your results for each individual experiment.
- Commented Code for your program. Include all source files needed for compilation and make sure it compiles successfully. Make sure you output the results to a specified folder of each experiment.

Flink resources

Apache Flink documentation:

https://ci.apache.org/projects/flink/flink-docs-stable/

Batch Flink sample problems:

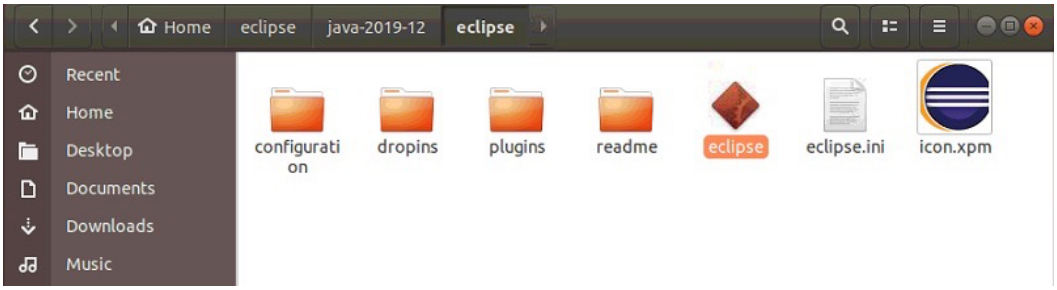
https://ci.apache.org/projects/flink/flink-docs-release-1.10/dev/batch/

Streaming Flink sample problems:

https://ci.apache.org/projects/flink/flink-docs-release-1.10/dev/datastream_api.html

Flink execution

Eclipse can be found in VM by “Files” -> “eclipse” -> “java-2019-12” -> “eclipse”



In order to be able to compile a Flink program in Eclipse, you need to have the following dependencies in your maven project:

```
<dependencies>
	<dependency>
		<groupId>org.apache.flink</groupId>
		<artifactId>flink-java</artifactId>
		<version>1.10.0</version>
	</dependency>

	<dependency>
		<groupId>org.apache.flink</groupId>
		<artifactId>flink-core</artifactId>
		<version>1.10.0</version>
	</dependency>

	<dependency>
		<groupId>org.apache.flink</groupId>
		<artifactId>flink-streaming-java_2.11</artifactId>
		<version>1.10.0</version>
	</dependency>

</dependencies>
```

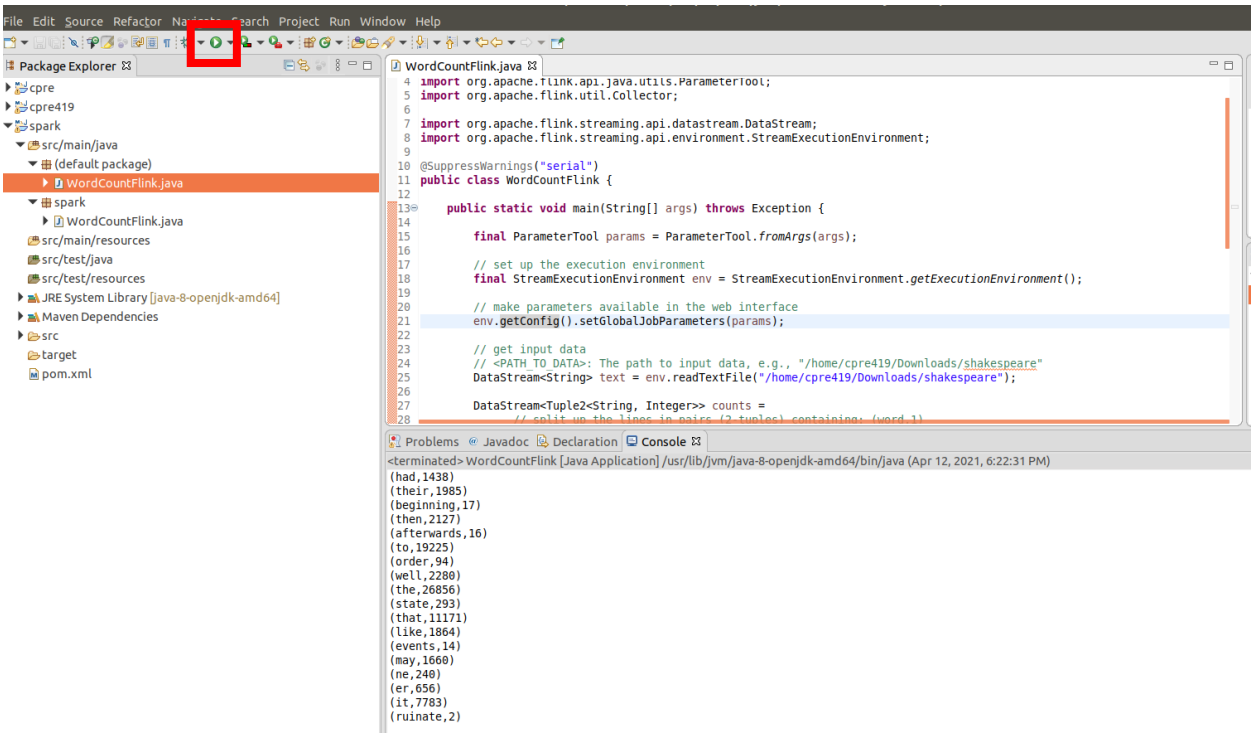
Experiment 1 (30 points) Word Frequency Distribution

The WordCount program counts the number of occurrences of every distinct word in a document.

Download “WordCountFlink.java” from Canvas and compile the program in Eclipse.
(The way to compile the program would the similar as Hadoop labs – create Maven project, switch JRE system library, add above dependencies in pom.xml)

Note that:
(1). You are NOT supposed to use command line to run Flink code. Instead, after completing your code in Eclipse, click on “Run” button (shown in red rectangle as below) and the job will be automatically submitted to Job Manager. In other words, no need to export .jar file;
(2). Make sure the input dataset is on your local file system, NOT HDFS (e.g., Shakespeare dataset is assumed to be under “Downloads” folder. Note that, in the code, the local path needs to be absolute path)

You should have the initial results for the wordcount program.



Write a Flink program to count the frequency of every distinct word from a given stream. Order word frequency distribution sorted by the frequency in descending order (may consider “addsink” function on Flink DataStream datatype, https://ci.apache.org/projects/flink/flink-docs-release-1.9/api/java/org/apache/flink/streaming/api/datastream/DataStream.html#addSink-org.apache.flink.streaming.api.functions.sink.SinkFunction-) and output the top 10 most frequent words along with their frequencies.

You may output/println your results to the Console or store/writeAsText to file.

Bonus 5 points: Find the top 10 most frequent words along with their frequencies for each 1000 words of input (in tumbling window).

You might want to consider "countWindowAll", https://ci.apache.org/projects/flink/flink-docs-master/api/java/org/apache/flink/streaming/api/datastream/DataStream.html, to window steam into tumbling count windows, and "apply", https://ci.apache.org/projects/flink/flink-docs-master/api/java/org/apache/flink/streaming/api/datastream/AllWindowedStream.html , to apply computation inside the windows.

Experiment 2 (40 points) Bigram Counting

A bigram is a contiguous sequence of two words within the same sentence. For instance, the text "This is a car. This car is fast." has the following bigrams: "this is", "is a", "a car", "this car", "car is", "is fast". Note that the two words within a bigram must be a part of the same sentence. For example, "car this" is not a bigram since these words are not a part of the same sentence. Also note that you need to convert all upper case letters to lower case first. The sentence ends with ".", "?" or "!".

Write a Flink program to count the frequencies of bigrams from a given stream. Read the input stream from Shakespeare dataset. You may output/println your results to the Console or store/writeAsText to file.

Bonus 5 points: Find the top 10 most frequent bigrams along with their frequencies for each 1000 words of input (in tumbling window style).