**CS435: Distributed Systems (242)**

# Assignment 1: A Parallel Word Frequency Calculator

**Due: Thursday, this week** **Weight (3 points)**

deepseek

Into the unknown

DeepSeek is an advanced AI model that excels in understanding and generating human-like text by leveraging deep learning techniques, context, and semantic relationships. It goes beyond basic word patterns to deliver more coherent and contextually accurate responses.

A word frequency calculator analyzes text to determine how often each word appears, helping models understand word patterns and distributions. This is key for generating coherent and contextually appropriate responses. By prioritizing high-frequency words, which are often more relevant, DeepSeek improves its ability to predict word combinations and produce natural-sounding text. Unlike simpler models, DeepSeek leverages advanced deep learning techniques, considering context, syntax, and semantics, going beyond basic word patterns.

For large datasets, calculating word frequency across multiple files can be time-consuming. This assignment/task involves creating a parallel word frequency calculator in Java using the Fork/Join framework. The program will read multiple text files from a directory and compute word frequencies, using parallel processing to speed up calculations across files.

The program runs from the console as follows:

```
java -jar JARFILE <Directory Path> <Parallelism> <Task size> <Keyword 1>
<Keyword 2> … <Keyword n>
```

Sample Run:

```
java -jar ./dataset 4 4 the end
```

For this run, the program would read all the text files in folder dataset, the parallelism is set to 4. The task size is set to 4. Only two keywords "**the**" and "**end**" are provided.

Note: The above is an example, your program should work with any number of parallelism (<100), task size (<1000000) and up to n number of keywords (0<n<10).

A starter java file is given to you (assignment1.java). Use it and follow the TODO tasks to complete your work.

Your program shoud print the frequency values of each word in each file to the console.

You must do the following tasks:

- Read in multiple text files from a directory. This must be done in the "compute" method to be executed by threads (parallelism).
- Calculate the frequency of the GIVEN keywords in each file. Store the result in an integer array (shared memory).
- Parallelize the frequency calculation using the Fork/Join framework, the threads write to the integer_array shared memory.
- Output the frequency values of each word in each file, to the console.
- Add other features from the fork/join framework as necessary.
- Display the run-times for each run, the number of threads used, and the thread task size.

The skeleton file assignment1.java given to you has:

a. A **main** method contains TO DO tasks. Follow the previous tutorial to complete the TO DO tasks. The files are read by the threads in parallel. The compute method in the **ComputeFrequency** class will compute the frequency values for each text file. The main method will output those values to the console (serial and not parallel).
b. A class **ComputeFrequency** is given to implement the Fork/Join Task: This class will represent a task that can be submitted to the Fork/Join pool. It will take in an array with text file names; a starting index and ending index to specify which files to process. It will then apply the recursive fork, where for each forked task compute method is called. The compute method implements the word frequency computation logic.

Download the datasets to test your work

- Dataset 1: 12 text files of varying size
- Dataset 2: 2000 text files of varying size

To verify the correctness of your program, the following are the frequencies of keywords "the" and "and" in the Datasets:

| | **frequencies** | |
|---|---|---|
| **Keyword** | **the** | **and** |
| **Dataset1** | 166 | 66 |
| **Dataset2** | 31519 | 12072 |

**Assignment Deliverables**

The deliverables for the project are the following. These need to be uploaded to LMS.

- A word document (Report) that includes:
    - Section 1: Experimental evaluation
        - Verification: Discuss test cases to ensure the program works correctly
        - Execution time with parallelism 2, 16, 256, 8192.
        - Execution time with task size per thread (4 files, 8 files, 16 files etc).
    - Section 2: Conclusion: write about the effect of task size and parallelism.
    - Section 3: Java code for the word frequency calculator using Fork/Join

**Submission and Grading**

All deliverables are appended in a WORD file. Submit this (ONE) file to LMS.

**Grading**:

- Correct usage of **ComputeFrequency** class and related methods: 40%
- Correct usage of ForkJoinPool and related calls: 20%
- Correct I/O and display in main: 10%
- Report quality (Graphs, tables etc): 15%
- Quality of conclusions in the report: 15%

**Additional Note:**

Any Student would be requested to present their work. The instructor reserves the right to "**interview**" any student on their submission to see the understanding of the submission. The instructor may also ask the student to modify the code to satisfy any test-case(s) there in.

**Code Inspection and plagiarism:**

The code would be inspected by the instructor. The instructor would use the MOSS tool (https://theory.stanford.edu/~aiken/moss/) to determine the originality of submission.

**If the code similarity is above 50%, the students would earn ZERO score on the assignment.**

**Submission Dead-Line**:

The submission deadline is final. Late Submissions will be awarded ZERO points.

**Important Notes:**

- It is the student's responsibility to check/test/verify/debug the code before submission.
- It is the student's responsibility to check/test/verify all submitted work (including jar files)
- It is the student's responsibility to verify that all files have been uploaded to the LMS.
- Incomplete or wrong file types that do not execute will NOT be graded.
- After an assignment/project has been graded, re-submission with an intention to improve an assignments score will not be allowed.
- The instructor has the right to share project execution reports that may have been auto-generated on the course website.