
# Bonus Points Lab Assignment Distributed Systems

Authors
**Jan S. Rellermeyer**
**Stefan Petrescu**

Distributed Systems

Leibniz University Hannover

October 2023

# Contents

# Introduction

Distributed systems (DS) are critical for deploying applications at large scale, e.g., Google Search, YouTube, Facebook are just a few examples of such systems, used daily by millions. In actuality, many, if not all, large-scale applications are distributed systems, as the need to transition to distributed architectures naturally arises for any business that has to scale out. In such scenarios, distributed systems experts are called to arms, and asked to apply their expertise. Without their knowledge, companies would not be able to accommodate the ever-increasing market demands for their products.

Designing and implementing distributed systems, however, is far from easy. Actually, it is so difficult that it is recommended to avoid them, if the problem can be solved in more traditional ways, and to consider distributed systems if there are no other alternatives. So, DS are more of a necessity rather than a nice-to-have — where all other systems fail, DS prevail.

With all of this in mind, the main purpose of this assignment is to get you acquainted to the world of distributed systems. You will gain invaluable real-world experience about how such systems work by tackling some of the challenges associated with creating one.

## Learning Objectives

We designed this assignment to give you enough context about the task at hand, while giving you enough space to be creative, hoping to motivate you to aim as high as you can. At the end of this assignment, we would like you to tackle the following points:

1. Implement complex operations of cloud computing in realistic scenarios.
2. Analyze the tradeoffs inherent in the design of cloud-computing-based applications.
3. Evaluate the functional and non-functional requirements of a distributed system through experimentation.

You can either opt for Track 1 or Track 2. However, if you have another problem in mind that you would like to solve, this is also possible. If that is the case, first write a proposal, and subsequently discuss it with the TA during the consultation hours.

# 1 Track 1: Discover trends in data

Trends GmbH provides insights about potential market trends by processing large amounts of textual data scraped from the internet. Specifically, they count the frequency of unique words in these data, information which is then subsequently used by other machine learning processes. Counting the frequency of words in the scraped data is essential for the company to provide value and attract new customers with their predictions.

However, as Trends GmbH has been growing very rapidly, a new issue has emerged: more and more data needs to be scraped to accommodate an ever-increasing number of clients. The current data volume scraped by the company has become too large for them to process it in reasonable time in their current setup. Even worse, they are currently unable to serve insights to all of their customers: even though they had invested in a new machine 10x more powerful than the previous one, its compute capabilities quickly fell short, rendering it insufficient.

Nevertheless, the company's CTO has identified the issue: their current sequential implementation is designed to work on a single machine, which in turn is bound to fail as data volumes keep on increasing. The CTO understands that a distributed system is necessary in this situation.

This is where you come in: they need someone to design a new way of processing their data, that potentially exploits the parallelism intrinsic to the problem. After meeting with the company, you both agree that you are to design a MapReduce system, a system that counts the frequency of unique words in a list of files provided by the company and return the results.

## 1.1 Assignment

You are to implement a framework (Figure 1) that essentially speeds up processing large volumes of data. Intuitively, such a speedup is possible as the problem benefits from parallelism, by distributing the computation by processing files on different machines, and subsequently aggregating the results.

**Input & Output.** You are given a list of **n** files, and you must produce **m** output files, which should collectively contain the unique words discovered with their associated frequencies. These files should not contain duplicates, neither in a single file instance nor across multiple files. In other words, a single file should not contain the same word twice, and a particular word should be present only in one of the output files.

The output files should contain the words with their respective frequency separated by a whitespace character; each word should start on a new line (see output sample in Figure 2). Furthermore, the output should be sorted in ascending order alphabetically, while also accounting for the ASCII value of the first character. The ASCII consideration for the first character of the word is for scenarios where there are words that start with capital letters. For instance, if both "student" and "Student" are present in the input files, "Student" should be appear before "student" in the output, as "S" is 83 in ASCII compared to "s" which is 115. We provide a sample of input-output files in Figure 2.

For inspiration on how to design the MapReduce framework, please refer to the original MapReduce paper: *MapReduce: Simplified Data Processing on Large Clusters* [1].

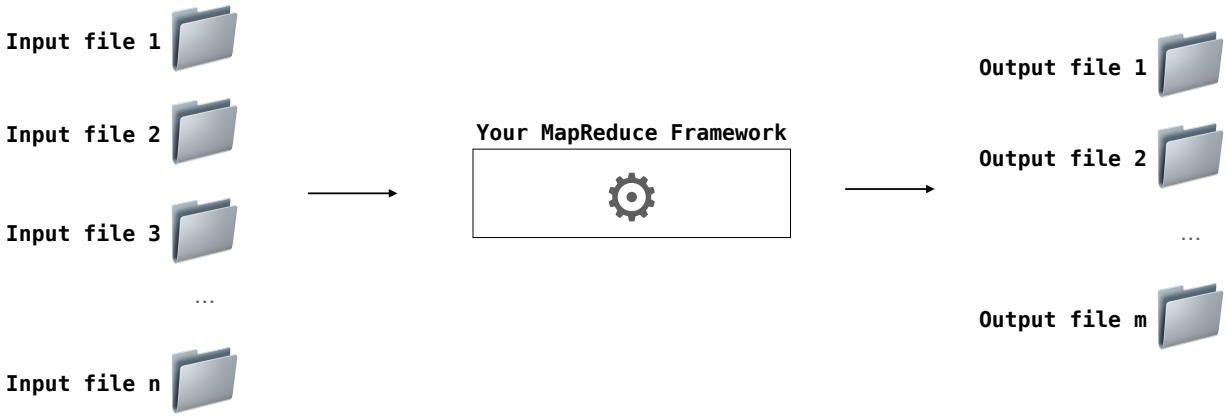


Figure 1: Given **n** files your are supposed to implement a MapReduce framework that counts all words in the input files and produces **m** output files with the results.

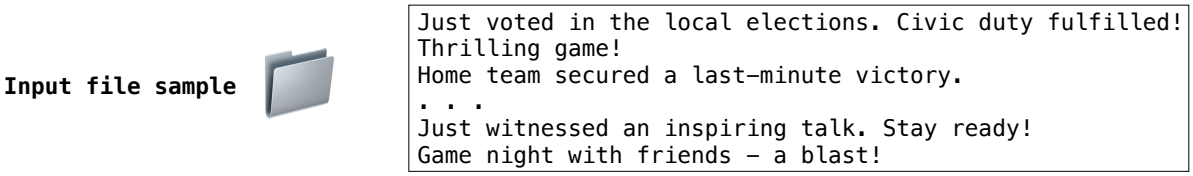



Figure 2: Input and output sample.

## 1.2 Mandatory requirements

You must design and implement a distributed (i.e., with multiple server nodes), replicated, and fault-tolerant system for processing the files, and assess its properties. Your system should meet the following requirements:

1. *System operation requirements*: You must implement the framework according to the design specified by Trends GmbH. The correctness of the results after processing the files has to be ensured by tests. You must demonstrate that your system can distribute the computation to multiple machines, i.e., GCP cloud instances, and return the correct results.

2. *Fault tolerance*: Consider a simple failure model, in which master and worker nodes may crash and restart. Your system must be resilient against master and client node crashes. In addition, all system events (e.g., Master starts Map phase, Worker $n$ starts Map processing, etc.) must be logged in the order in which they occur, on at least two server nodes.

3. *Scalability requirements*: The properties of your system must be demonstrated when it contains at least 8 Workers, and at least 2 Master nodes. Also, you must ensure that the processing of data is correctly completed.

## 1.3 Instructions to get you started

**Files for running experiments**
To access the files for the experiments and other instructions related to that, make sure to clone the following GitHub repository.

```
$ git clone https://github.com/LUH-VSS/distributed-systems-experiments.git
$ cd distributed-systems-experiments
$ cd track_1
$ ls
1_customer_trends.txt
2_customer_trends.txt
3_customer_trends.txt
...
```

**Google Cloud Credits**
For this course you benefit of 50 EUR in credits on Google Cloud. To claim these credits you have to sign up for the project, by showing up during the first lab meeting. (October 23, 2023)

## 1.4 Deliverables

1. **A presentation** for the project at the end of the course.
2. **A public GitHub repository** that showcases your work for the course, published for the world to see. Note that this repository is not the same as the one in GitHub Classroom. This repository (the one intended to be published for the world to see) is especially relevant to encourage you to work on showcasing your skills to the world, to impress future recruiters, and contribute to the community.
3. **A private GitHub repository in GitHub classroom** that contains: (1) the presentation and (2) the code for the project, and make sure to provide a README.md file that contains setup instructions and explains the structure of the project.

## 1.5 Milestones

1. **October 23, 2023: Sign-up for the project.**
   (a) Enroll using GitHub Classrom link.
   (b) Make an initial commit with a README.md file that at least contains the result PINs (ErgebnisPINs) of all the group members.
   (c) Retrieve Google Cloud Coupon.
2. **October 30, 2023: Finalise project proposal: discuss with TA during office hours.**

3. **December 18, 2023: Finalise evaluation strategy (discuss with TA during office hours or, in exceptional cases, by appointment).**
4. **January 25, 2024: Presentation of your system.**
5. **by January 26, 2024: Submit deliverables to GitHub.**

## 1.6 Use of Existing Technology

- You CAN use the Google Cloud Platform for which we have secured cloud credits. Instructions for obtaining the cloud credits will be shared with you during the first milestone meeting.
- We recommend that you implement the project in a high-level language like Go, Java, or Python.
- You should NOT use libraries that fully automate the functionalities of your distributed system.
- When using an external library, give due credit in your code. When in doubt about technology you intend to use, consult with the TA.

## 1.7 Sign-up instructions

To sign-up for the project, make an initial commit with a README.md file that at least contains the result PINs (ErgebnisPINs) of all the group members. Sign-up and project creation are performed through this link: https://classroom.github.com/a/IuYHBysz. Finally, to complete the sign-up for the project you have to show up to the first milestone meeting and claim your Google Cloud Platform credits, using a code which will be provided by the TA.

# 2 Track 2: Sort Spendings of Bank Customers

HannoverBank GmbH has collected a large amount of data containing customer information and is now faced with the challenge of processing this data efficiently. The bank's objective is to arrange the data in a structured manner. Unfortunately, their initial attempts to achieve this on their local system proved unsuccessful and inefficient.

In response, your task is to implement a data sorting solution that can handle their data and sort it. Your solution needs to be optimized for performance and scalability, ensuring that the processing task can be completed effectively even when the data grows.

As an expert in the field, you are entrusted with designing and implementing a sorting mechanism that can seamlessly handle the substantial data volume while providing HannoverBank with the ordered results they require.

## 2.1 Assignment

You are to implement a framework (Figure 3) that essentially speeds up processing large volumes of data. Intuitively, such a speedup is possible as the problem benefits from parallelism, by distributing the computation by processing files on different machines, and subsequently aggregating the results.

**Input & Output.** You have been given a list of **n** files that contain information about customers, and you are asked to produce **one** output file that contains the sorted data. Each file contains entries of customers that include a unique identifier, customer name, and transaction amount. You are required to sort the input files by shuffling the entries contained within them and return the sorted data in one output file. In Figure 4 we provide an example of input and output samples.

For inspiration on how to design the MapReduce framework, please refer to the original MapReduce paper: *MapReduce: Simplified Data Processing on Large Clusters* [1].

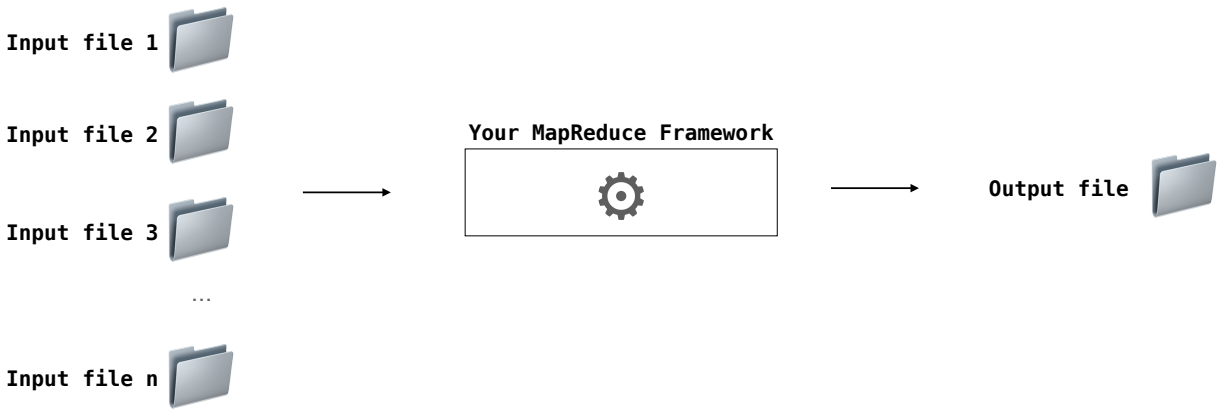


Figure 3: Given **n** files your are supposed to implement a MapReduce framework that sorts all customer data based on the amount and produce one output file with the results.

**Input file sample**

```
sd82349sd84fjss, Sophie Schmidt,  500.25
ah5123shfn493ks, Felix Wagner,    1000.75
bf6f82jkdk82wqp, Hannah Fischer,  250.50
sd82hsk12h8s7af, Tim Klein,       750.00
gk92h3lf823sd1c, Lea Schmitt,     1200.20
. . .
lq0fh2sl2393d9w, Anna Maier,      350.90
p3ksdf2j83wkefw, Max Bauer,       600.30
hs3df89s2dsfw1m, Emma Zimmermann, 950.80
```

**Output file sample**

```
gk92h3lf823sd1c, Lea Schmitt,     1200.20
ah5123shfn493ks, Felix Wagner,    1000.75
. . .
lq0fh2sl2393d9w, Anna Maier,      350.90
bf6f82jkdk82wqp, Hannah Fischer,  250.50
```

Figure 4: Input and output sample.

## 2.2 Mandatory requirements

You must design and implement a distributed (i.e., with multiple server nodes), replicated, and fault-tolerant system for processing the files, and assess its properties. Your system should meet the following requirements:

1. *System operation requirements*: You must implement the framework according to the design specified by HannoverBank GmbH. The correctness of the results after processing the files has to be ensured by tests. You must demonstrate that your system can distribute the computation to multiple machines, i.e., GCP cloud instances, and return the correct results.

2. *Fault tolerance*: Consider a simple failure model, in which master and worker nodes may crash and restart. Your system must has to be resilient against master and client node crashes. In addition, all system events (e.g., Master starts Map phase, Worker $n$ starts Map processing, etc.) must be logged in the order in which they occur, on at least two server nodes.

3. *Scalability requirements*: The properties of your system must be demonstrated when it contains at least 8 Workers, and at least 2 Master nodes; also, you must ensure that the processing of data is correctly completed.

## 2.3 Instructions to get you started

### Files for running experiments

```
$ git clone https://github.com/LUH-VSS/distributed-systems-experiments.git
$ cd distributed-systems-experiments
$ cd track_2
```

```
$ ls
1_bank_customers.txt
2_bank_customers.txt
3_bank_customers.txt
...
```

To access the files for the experiments and other instructions related to that, make sure to clone the following GitHub repository.

**Google Cloud Credits**
For this course you benefit of 50 EUR in credits on Google Cloud. To claim these credits you have to sign up for the project, by showing up during the first lab meeting. (October 23, 2023)

## 2.4 Deliverables

1. **A presentation** for the project at the end of the course.
2. **A public GitHub repository** that showcases your work for the course, published for the world to see. Note that this repository is not the same as the one in GitHub Classroom. This repository (the one intended to be published for the world to see) is especially relevant to encourage you to work on showcasing your skills to the world, to impress future recruiters, and contribute to the community.
3. **A private GitHub repository in GitHub classroom** that contains: (1) the presentation and (2) the code for the project, and make sure to provide a README.md file that contains setup instructions and explains the structure of the project.

## 2.5 Milestones

1. **October 23, 2023: Sign-up for the project.**
   (a) Enroll using GitHub Classrom link.
   (b) Make an initial commit with a README.md file that at least contains the result PINs (ErgebnisPINs) of all the group members.
   (c) Retrieve Google Cloud Coupon.
2. **October 30, 2023: Finalise project proposal: discuss with TA during office hours.**
3. **December 18, 2023: Finalise evaluation strategy (discuss with TA during office hours or, in exceptional cases, by appointment).**
4. **January 25, 2024: Presentation of your system.**
5. **by January 26, 2024: Submit deliverables to GitHub.**

## 2.6 Use of Existing Technology

- You CAN use the Google Cloud Platform for which we have secured cloud credits. Instructions for obtaining the cloud credits will be shared with you during the first milestone meeting.
- We recommend that you implement the project in a high-level language like Go, Java, or Python.
- You should NOT use libraries that fully automate the functionalities of your distributed system.
- When using an external library, give due credit in your code. When in doubt about technology you intend to use, consult with the TA.

## 2.7 Sign-up instructions

To sign-up for the project, make an initial commit with a README.md file that at least contains the result PINs (ErgebnisPINs) of all the group members. Sign-up and project creation are performed through this link: https://classroom.github.com/a/IuYHBysz. Finally, to complete the sign-up for the project you have to show up to the first milestone meeting and claim your Google Cloud Platform credits, using a code which will be provided by the TA.

# 3 Rubric

| Category \ Score | Excellent (1) | Good (0.75) | Satisfactory (0.5) | Insufficient (0.25) | Missing (0) |
|---|---|---|---|---|---|
| **Design (1)** | Very well thought through design, clear and deliberate decisions, creative solution | Good design, mostly clear and deliberate decisions | Decent design, some decisions not justified, mostly straightforward solution | Incoherent or technically incorrect design, decisions are not justified, solutions is trivial | Design not present |
| **Implementation (2)** | Excellent implementation, fully functional, true to the design or well justified changes where necessary, good coding practices, well documented | Good implementation, mostly functional, mostly true to the design, good coding practices, some documentation | Decent implementation, partly functional, partly deviates from the design without justification or due to time management issues, not using good coding practices, almost no documentation | Implementation insufficient, not functional, original design not honored, not understandable | No code |
| **Evaluation (1)** | Excellent evaluation, very good design of experiments, all important system properties are evaluated | Good evaluation, adequate design of experiments, most important system properties evaluated | Decent evaluation, experiments partially evaluate important system properties | Insufficient evaluation, experiments fail to evaluate important system properties | No evaluation |
| **Presentation (2)** | Clearly presented, well structured, clear message and results | Well presented, good structure, good results | Decent presentation, some structure is present, results are recognizable | Work is not always recognizable, structure is mostly missing, results absent or insufficient | No presentation |

Figure 5: Rubric for the assignment.

Each score has an associated weight, Excellent – 1, Good – 0.75, etc. The final grade is computed by multiplying each category by with its respective score (weight). For example, if Design is Excellent (1), Implementation is Excellent (1), Evaluation is Good (0.75), and Presentation is Excellent (1), the final grade for the assignment will be computed as:

```
Example final grade assignment = 1 * 1 + 2 * 1 + 1 * 0.75 + 2 * 1 = 5.75
```

# 4 Anti-Fraud Policy (Zero-Tolerance)

Our anti-fraud policy is simple: zero-tolerance, within the limits set by LUH. We will pursue each case of potential fraud we discover. We will use to the maximum extent the means provided by LUH to punish (attempts to) fraud.

# References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, (San Francisco, CA), pp. 137–150, 2004.

# 5 Miscellaneous: VSS Thesis Projects

At the VSS department, we are keen on collaborating with students. Your generation will not only manage the ever-growing software infrastructure that has become engrained in the fabric of our society, but it will also solve the challenges of tomorrow, challenges that will arise with the evolution of software. So, we are very interested in collaborating with you, and helping you grow as much as we can, as the future of software will be in your hands. We have a broad spectrum of topics available for collaboration, ranging all the way from from compilers to machine learning. So, if you are interested in working with us, don't hesitate to reach out.

To show that we practice what we preach, we have set up a small playground exercise which helps you train your first ever ML model. The goal of this exercise is to help you get some real-world experience with ML — a topic that has become increasingly relevant over the past decade. As ML can be intimidating, especially for those that do not have any prior experience with Python or AI courses, we have tried to provide an exercise that is as easy to play with as possible. The playground exercise takes approximately 5 minutes to complete, and by the end of it, you will be able to say: "I trained my first ML model!". Specifically, at the end of it, you will have trained a ML model that is able to recognize clothing items, such as dresses, coats, t-shirts, etc. See some example images below that your model will be able to recognize.


Figure 6: Images your model will be able to recognize.

To find out more about the playground exercise, simply click the link below:
Playground Excercise: Train your first ever ML model.