Universität Göttingen
Department of Computer Science

Julian Kunkel

Exercise Week 2
HPDA / 2022
240 Minutes Total
Discussion in Week 3: 2023-11-06

---

1. The tasks described in this worksheet are part of the formative assessment. They serve the purpose to prepare you for the examination. We will discuss the solutions during the next **interactive session** after they are handed out – while they fit to the lecture of the week they are handed out, they might be discussed in two weeks time due to the bi-weekly exercise schedule.
2. Make sure to plan your time for the whole sheet carefully. The complete exercise should represent approximately three hours of independent study. The time limit indicates how much time you should spend on each task, and not how much time you may actually need; it is important that you engage with the material and not that you complete all tasks perfectly. Feel free to collaborate and team up.
3. The exercises are designed to challenge you and train you further as guided self-study. The time limit might be too ambitious for you; you may team up with colleagues. It is not an issue as long as you manage to at least partially resolve each task within the time budget. If you (and your team) are struggling, reach out for help in Teams! You may also share your thoughts via the Studip Forum.
4. We recommend that you create a (private) Git repository (see `https://gitlab.gwdg.de`) where you store your findings and outcomes while processing the exercises. This portfolio of work can be useful in the future.

Contents

Task 1: Introduction: Understanding Architectures and Performance (60 min)

Performance is an important aspect of **High-Performance** Data Analytics (HPDA) and of parallel and distributed systems. Imagine you are 20% more efficient using server infrastructure than your competition - and you have 1000s of servers. That is a business advantage...

In this task, you will research relevant performance characteristics of a single computer system and a distributed system using literature.

Tasks

1. Research what the term "performance" could mean for an HPDA application. Note that there is not a single performance characteristic but multiple characteristics that are meaningful. Document why a use case is high-performance. Your notes should cover about 1/4-page.

2. Research alternatives for "supercomputer architecture" in the literature, e.g., using Google. Research two architectures and select diagrams that describe them; the architecture could be a specific supercomputer. Your notes should cover about 1/4-page.
3. Research what the term "performance" could mean for a single computer system. Document relevant characteristics for a single computer system, i.e., a server or desktop system. Find typical (performance) values for a decent (current) computer system. Your notes should cover about 1/4-page.
4. Think about performance of multiple computers that are interconnected to a distributed system. How would you expand upon your performance characteristics for a single computer system? Look at the TOP500 list[1] and document which performance characteristics are provided in the list.

Portfolio (directory: `2/performance`)

`2/performance/architectures/`	Store supercomputer architecture documents here.
`2/performance/performance-hpda.txt`	Performance characteristics of the HPDA application.
`2/performance/performance-1node.txt`	Performance characteristics of a single node.
`2/performance/performance-distributed.txt`	Performance characteristics of a distributed system and performance metrics used in the TOP500.

Task 2: Data Models for Wikipedia Articles (90 min)

In this task, we will create various data models for typical Wikipedia data.

Firstly, we need to consider the data we want to store. Each article should store the following properties:

- Title
- Text
- Category
- Links to related articles

The following operations should be possible:

- Access article details based on the article's "title"
- Finding related articles (those that link to one another) from a given article
- Retrieving all articles for one category

You can view some Wikipedia articles to understand this better.

Document how you would create the following data models:

- Relational
- Columnar
- Key-Value
- Document
- Graph

Think about what a fact-based model implementation could add to any of these data models.

[1] https://www.top500.org

Portfolio (directory: `2/data-model`)

`2/data-model/data-model.txt`	Store a description of the respective data model, describe how the operations can be executed, and give at least one example.
`2/data-model/fact-based-discussion.txt`	Discuss here what the fact-based data model would add.

Task 3: Word Frequencies in Moby-Dick (90 min)

In this task, you will write a Python program to compute word frequencies for a text. We will work with the book ”Moby Dick” by Herman Melville.

You can find it as a text file `moby.dick.txt` via Google.

3.1 Visualization of Individual Word Frequencies

Your program should take a word as input via command line argument, then split the book into its chapters and for each chapter calculate the relative frequency of the word given by the command line.

Your program will compute the occurrences of any given term across the books chapters. Visualize the frequency in a graph (see the hints).

3.2 Aggregation of Word Occurrences

Now the goal is to output the absolute occurrences of *each* word for every chapter. Your program output should be a CSV file with the following header:
`chapter-name, chapter-length, unique-words`
These should give the name of the chapter, the length of the chapter in number of words and the number of unique words in the chapter respectively.
Furthermore, your program should output a JSON file that includes the number of occurrences of words per chapter. This can be achieved by storing the words and number of occurrences into a Python dictionary and later dumping the dictionary as a JSON file.

3.3 Most Common Words

Write a program that reads in the JSON file from the previous task and prints the 10 most common words in the whole book.

3.4 Hints and Code-Skeleton

Visualization of the word frequency can be done modifying the code provided below:

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt

if __name__ == "__main__":
    numbers = [1,22,33,5,44] # Load your samples here
    plt.bar(range(len(numbers)), numbers, 0.35)
    plt.ylabel("selected word")
    plt.xlabel("occurrence over all chapters")
    plt.show()
```

Prepare the code such that it produces a graph that shows the occurrences of a specific word for each chapter.

For this task the module *matplotlib* is required, it is already installed on the cluster. In case you want to test your work locally, install the package via the command `$ pip` or your distribution’s package manager.

You may use dictionaries, the `csv` and the `json` module.

Portfolio (directory: `2/word-freq`)

`2/word-freq/graph.py`	Your source code to visualize a word‘s occurrence.
`2/word-freq/csv.py`	Your source code to extract the word occurrences.
`2/word-freq/most-common.py`	Your source code to find the most common words.