Universität Göttingen
Department of Computer Science

Julian Kunkel

Exercise Week 9
HPDA / 2022
240 Minutes Total
Discussion in Week 9: 2023-01-16

---

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: Consistent Hashing (30 min)

This task practices consistent hashing. You will add and remove some servers and data on the virtual ring for consistent hashing.

Consider the following scenario:

- $M = 101$
- Three servers with the names **s[1-3]** and the IP addresses 10.0.0.[1,40,80]
- Replicate each server twice using the hash functions $f(ip) = ip \mod M$ and $g(ip) = ((ip + 50) \cdot 150) \mod M$ (use the last block of the IP, i.e., 1-3).
- Data items with the key (d) must each be replicated twice using the hash functions $f(d) = d \mod M$ and $g(d) = (d + 50) \mod M$.

Visualize the ring (could be an array, if necessary) during the processing of the following steps:

1. Server **s1** is started
2. Server **s2** is started

3. data item ($d = 10$) is added
4. data item ($d = 30$) is added
5. data item ($d = 70$) is added
6. Server **s3** is started; rebalance the data if necessary!
7. data item ($d = 20$) is added
8. data item ($d = 80$) is added
9. Server **s2** fails; rebalance the data if necessary!

Which server is responsible for what data item?

Portfolio (directory: `9/hashing`)

`9/hashing/ring-5.pdf`	A visualisation of the ring with the data after the completion of Step 5
`9/hashing/ring-6.pdf`	A visualisation of the ring with the data after the completion of Step 6
`9/hashing/ring-end.pdf`	A visualisation of the ring with the data after all steps are executed

Task 2: API via HTTP (105 min)

As part of this task, we explore the prototyping of a HTTP-API server using the Flask microframework.

It should be noted that the term *REST API* has been widely misused and in most cases does not reflect the definition of it that Fielding had created. In praxis most modern APIs that claim to be RESTful are actually JSON-APIs driven by HTTP verbs that fulfill some characteristics of REST but not all. Most commonly the "self-describing" and "hypertext-driven" characteristics are ignored and data is transferred in JSON format and the API is described through external documentation. Also see the definition of REST and this comment from Fielding himself on the topic. For the sake of practicality, we will still link to external resources that use the term REST incorrectly and expect you to work with them while keeping in mind what REST actually is. Nevertheless, an API that does not fully conform to all criteria of REST can still be very useful and is much better than having no API at all.

Follow this tutorial to prototype a HTTP-API service with a POST and GET methods (leave authentication alone)[1]. Test this service using CURL. Write down CURL commands that test your endpoints.

Think about the syntax and semantics of the HTTP-API service specified in the tutorial. How are these supported by the HTTP methods?

Think about how a HTTP-API service could be used to expose a hierarchical file system, i.e., being able to upload/download files/directories. How could the syntax and semantics of the API look like?

Portfolio (directory: `9/API`)

`9/API/app.py`	The flask microservice
`9/API/discussion-file-system-API.txt`	The syntax and semantics of the HTTP-API for a hierarchical file system

[1] If you have no or little Python experience, recheck your knowledge about REST and spend the time to learn Python more. Try to understand the tutorial and concepts. You may not be able to complete the tutorial in this time frame, but that doesn't matter, focus on the next questions.

Hints

- There is no need to set up a virtualenv (as described in the tutorial). Instead use

  ```
  $ pip3 install --user <PACKAGE NAME>
  ```

  to install the packages described.
- If necessary, you may use http://httpbin.org/ to debug your HTTP requests issued with pip.
- For the hierarchical file system, you may want to check the Hadoop File System RESTful API again.

Further Reading

1. Publicly available APIs: https://github.com/public-apis/public-apis

Task 3: Interactive API Documentation (30 min)

In the previous task you have built an API that is only defined through its implementation. The next logical step is to use a specification that defines the API and which serves as the source of truth for the implementation and the users. A standard for such specifications is OpenAPI[2], which is supported by many platforms that can generate server code, documentation and client requests from it.

Have a look at the following two interactive API documentations:

- https://petstore.swagger.io/
- https://rapidocweb.com/examples/example100.html

Try out both interfaces and run the API calls via the web UI.
Try creating a pet, searching for it, ordering and deleting it in the end.

Think about the advantages of this approach compared to REST and its limitations.
Write down your thoughts.

Portfolio (directory: `9/interactive-api-docs`)

`9/interactive-api-docs/discussion.txt`	Your discussion of the comparison between REST and specification based APIs.

Task 4: I/O Performance Analysis (75 min)

Understanding if observed performance is acceptable is important to identify optimization potential. A compute job contains time for I/O and processing, we want to consider just I/O time for now. In this task, imagine you have obtained a set of performance measurements on different systems, and you shall assess the performance.

Consider the following alternative system characteristics:

1. Network: 10 GBit Ethernet, or 100 GBit Infiniband.
2. 10 or 100 storage servers. A server may have either 1 or 4 network interfaces.
3. Storage nodes are equipped with either SSDs or HDDs, and either 1, 5, 8, or 16 storage media.

[2] https://www.openapis.org/

4. There are a various number of client nodes available, but a node has only one network interface.

We measured the following client configurations with given total amount of data and measured runtime of the overall applications:

- 1 Client node, 100 MByte of data in 0.1s.
- 10 Client nodes, 10 GByte of data in 1s.
- 100 Client nodes, 1 TByte of data in 100s.
- 1000 Client nodes, 100 TByte of data in 100s.

Discuss which combination of system configurations are probably impossible, efficient or rather inefficient.

Portfolio (directory: `9/performance`)

`9/performance/discussion.txt` Your discussion of the performance analysis.