FAF.PAD16.2 Autumn 2023

Lab 1: Web Proxy

Handed out: September 4, 2023
Due: October 26, 2023

House Rules - mandatory for reading

In this course, you will undertake two laboratory assignments, with a designated time frame of 15-16 weeks (depending on the university schedule). Which means that after passing the deadline (of 15-16 weeks), the labs won't be accepted anymore. These laboratories together form an integral project, meaning that completing the second lab without doing first one will be impossible, kindly take note of this aspect. Each lab is divided into three checkpoints, and each checkpoint will be subject to stringent deadlines (the deadlines can be found attached on Else and in Teams within this group). Upon the expiration of a checkpoint's deadline, **ALL** students will receive a grade. If a student has defended/presented the lab, they will receive a grade as negotiated with the professor. Failing to do so, will result in an automatic grade of 1, which will be input into the digital record without the possibility of alteration. The lab grade is composed of the average of the scores from the checkpoints within that particular lab.

Every time you intend to present a lab, you will need to reserve a presentation slot. A few days before the lab day, the professor will post an Excel sheet on Teams, indicating the maximum number of students allowed to present the lab on that day (this number will be estimated as follows: total number of students in the group / number of days designated for presenting one checkpoint + 1). Students will input the required information as instructed by the professor (e.g., Docker image, repository link). Roughly 24 hours before the lab day, access to the Excel sheet will be restricted, preventing further sign-ups. On the lab day, only students on the list will be allowed to present.

Lab Presentation Procedure: On the presentation day, each student will draw two cards, each containing topics from different domains. The student must then respond to the selected topics, having the entire lab duration to prepare beforehand. They are allowed to utilize any physical or digital information sources. Each card carries a value of 0.5 points towards the grade, meaning that students can receive -0.5, 0, or 0.5 points for each response. Also students are required to present the laboratory assignment in accordance with the criteria outlined in this document. The sequence doesn't play a role, students may present the lab before or after addressing the questions.

Students have the option to work individually or in pairs, with the criteria for each scenario detailed below. **IMPORTANT**: For students working in teams, evaluations will be conducted on an individual basis even if the project is collaborative. **HOW?** Responses to the card questions will be taken into consideration, Git history will be analyzed, and if necessary, questions related to the code or theoretical knowledge that the student should be familiar with. If both students have adequately presented their work, the grade will be negotiated according to the

criteria outlined in this document. In case that professor identifies a discrepancy in the contribution between team members, the student that contributed the less will automatically receive a grade of 1, and the grade will be negotiated with their teammate.

All the above mentioned are the laboratory conduct House Rules. In case of any uncertainties or inquiries, the professor is available to provide clarification. Should students find any of these rules inequitable, incorrect, or wish to suggest additional elements, they have the option to propose their own ideas or modifications to the professor (only well-constructed and substantiated ideas will be entertained). Such proposals will be considered exclusively during the FIRST WEEK OF THE SEMESTER (up until 11:59 PM on September 8th). After this period, the new document will be updated if there have been modifications, and the conditions and terms stated herein will no longer be subject to discussion by either the professor or the students.

Checkpoints Terms

Checkpoint 1

Select a pertinent and suitable topic within the scope of the PAD course, as the professor will reject any idea that does not make sense to be realized through the implementation of distributed systems. The list provided below represents the tasks you must accomplish for Checkpoint 1. At the beginning of each item on the list, you will find the number of points awarded upon fulfilling the condition (the total points sum up to 10). Please take into consideration that there is no need to create extensive microservices, services containing 4-5 endpoints are sufficient (maybe with extra futures, if necessary). The most crucial aspect is to consider the requirements outlined in the **Requirement** section, including both the number of languages used and the necessary features to be implemented.

- **2p** - Assess Application Suitability: Before implementing microservices, it's crucial to assess whether your application is a good fit. Microservices work well for complex applications with multiple components that can be developed and maintained independently. Compose a list of reasons why your idea is relevant and why its implementation through distributed systems is necessary. Provide real-world examples of well-known projects (such as Facebook) that are similar and employ microservices.
- **2p** - Define Service Boundaries: Microservices require clear service boundaries. Each microservice should encapsulate specific functionality, such as user authentication, product catalog, or order processing, to ensure modularity and independence. Create simple system architecture diagram [1].
- **2p** - Choose Technology Stack and Communication Patterns: When implementing microservices, you must carefully choose the technology stack for each service. This includes decisions on programming languages, frameworks, and communication patterns if any. For example, you might use RESTful APIs (synchronous communication) or message queues (asynchronous communication) to enable inter-service communication.

• **3p** - Design Data Management: Consider how data will be managed across microservices. You can opt for separate databases for each service, use APIs for data access and so on. Additionally, here you will enumerate all the endpoints across all your services and define the data to be transferred, including its format and type (you may present it in JSON format, Protobuf format, or any preferred format, with clarity being utmost importance). Furthermore, you will define the response returned by each endpoint.

• **1p** - Set Up Deployment and Scaling: Determine how you'll deploy and scale your microservices. Consider containerization (Docker) and orchestration (Kubernetes) for managing deployment and scaling.

Checkpoint 2

For Checkpoint 2, student is required to implement a big part of the laboratory work, see Requirements section for tasks and marking system. Interpret it as being a Minimum Viable Product (MVP), and defend to the professor a functional or partially functional implementation at 70-80 percentage completion of the lab. Therefore, for a 10, the student should cover the conditions for a grade of 7 or 8. If the student aims for a laboratory evaluated for 6, he/she may cover the conditions for a grade of 4 or 5.

The grading for this checkpoint will be specific. Students who miss the deadline will receive a grade of 1, which will be automatically recorded in the grade book. On the other hand, receiving a '+' means that the student is guaranteed a temporary mark. When student presents Checkpoint 3, this '+' will automatically convert into the grade received at the third checkpoint.

Checkpoint 3

For Checkpoint 3, the student must Dockerize the project, regardless of the grade they are aiming for. The Docker image will be sent in advance to the professor for local execution and testing. Further details on the process of advance submission of the checkpoint and testing conditions will be provided by the professor a few days before the presentation day.

Important to mention

Laboratory 2 will build upon the first lab, hence, some features in lab 2 will rely on points implemented in the first lab. For Lab 2, you **may** require: cache, gateway, at least 2 microservices and 2 databases.

Requirements

Mark	Team size: 1	Team size: 2
1	just be	(just be) x2
2	• Mark 1 • 2 services/APIs that communicate with each over in any way [2]	• Mark 1 • 3 services/APIs that communicate with each over in any way written in 2 different languages [2]
3	• Mark 2 • **Status Endpoint** - simple endpoint implemented for all services [3] • **Task Timeouts** for all requests [4]	• Mark 2 • **Status Endpoint** - endpoint implemented for all services that ping periodically to check health (once in 10-15 seconds) [3] • **Task Timeouts** for all requests [4]
4	• Mark 3 • each service must have its own database [5] • **Concurrent tasks limit** [6]	• Mark 3 • one service should have SQL, another NoSQL(document) and last one NoSQL(graph) database [5] • **Concurrent tasks limit** [6]
5	• Mark 4 • implement **Gateway** in a different language than other 2 micro-services [7] • implement **Cache** [8]	• Mark 4 • implement **Gateway** in a different language than other services [7] • implement **Cache**, separate, personal service as cache (no Redis or another cache db) [8]
6	• Mark 5 • **Service Discovery** in the same language as Gateway [9] • **Status Endpoints** for Gateway and Service Discovery [3]	• Mark 5 • **Service Discovery** in the same language as Gateway [9] • **Status Endpoints** for Gateway and Service Discovery [3]
7	• Mark 6 • **Load Balancing: Round Robin** (3-4 replicas per service) [10] • **Circuit Breaker:** Trip (log) if a call to a service fails (3 errors in [task timeout limit * 3.5]) for all endpoints - implement in each service that make calls [11]	• Mark 6 • **Load Balancing: Round Robin** (3-4 replicas per service) [**?**] • **Circuit Breaker:** Trip (log) if a call to a service fails (3 errors in [task timeout limit * 3.5]) for all endpoints - implement in each service that make calls [11]
8	• Mark 7 • **Unit Testing** for a single service [12]	• Mark 7 • **Unit Testing** for all services [12] • **Integration Testing** [13]

9	• Mark 8 • **RPC** [14] • **Health Monitoring and Alerts** - raise an alert message when the load number is critical (define critical load for your system, e.g: 60 pings per second for a specific service)	• Mark 8 [15] • **RPC** [14] • **Health Monitoring and Alerts** - raise an alert message when the load number is critical (define critical load for your system, e.g: 60 pings per second for a specific service) [15]
10	• Mark 9 • instead of Load Balancing: Round Robin implement **Load Balancing: Service Load** [10] • instead of previous Circuit Breaker implement **Circuit Breaker:** Remove service if threshold is reached(3 errors in [task timeout limit * 3.5]) for all endpoints. [11] 1	• Mark 9 • instead of Load Balancing: Round Robin implement **Load Balancing: Service Load** [10] • instead of previous Circuit Breaker implement **Circuit Breaker:** Remove service if threshold is reached(3 errors in [task timeout limit * 3.5]) for all endpoints. [11] 1

Readings

[1] Example. "Microservice Architecture". https://microservices.io/i/Microservice_Architecture.png

[2] AWS. "What is an API".

https://aws.amazon.com/what-is/api/

[3] "API health check". https://testfully.io/blog/api-health-check-monitoring/#what-is-an-api-health-check

[4] "408 Request Timeout". https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/408

[5] Mark Smallcombe. "SQL vs NoSQL: 5 Critical Differences". https://www.xplenty.com/blog/the-sql-vs-nosql-difference

[6] "About maximum concurrent tasks". https://docs.infor.com/ism/5.x/en-us/ism_onlinehelp/lsm1454147676513.html

[7] "What is a Gateway and What Does it Do?". https://whatismyipaddress.com/gateway

[8] Julien Le Coupanec. "Redis Cheatsheet - Basic Commands You Must Know". https://gist.github.com/LeCoupa/1596b8f359ad8812c7271b5322c30946

[9] Simone Cusimano. "Service Discovery in Microservices". https://www.baeldung.com/cs/service-discovery-microservices

[10] "Load Balancing As A Service (LBaaS)". https://avinetworks.com/glossary/load-balancing-as-a-service/

[11] Martin Fowler. "CircuitBreaker". https://martinfowler.com/bliki/CircuitBreaker.html

[12] Thomas Hamilton. "Unit Testing Tutorial – What is, Types and Test Example".

https://www.guru99.com/unit-testing-guide.htmlhttps://www.guru99.com/unit-testing-guide.html

[13] Thomas Hamilton. "Integration Testing: What is, Types with Example". https://www.guru99.com/integration-testing.html

[14] Phil Sturgeon. "Understanding RPC Vs REST For HTTP APIs". https://www.smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/

[15] "Pattern: Health Check API". https://microservices.io/patterns/observability/health-check-api.html

Good Luck!