Technical University of Munich

School of Computation, Information and Technology
-- Informatics --

Bachelor's Thesis in Informatics

Load Testing and Performance Evaluation of the Theia Online IDE

Belastungstests und Leistungsbewertung der Theia Online IDE

Author:	Tobias Wen Klingenberg
Supervisor:	Prof. Dr. Stephan Krusche
Advisors:	Matthias Linhuber, M.Sc.
Start Date:	03.07.2025
Submission Date:	03.11.2025

Abstract

Theia Cloud is a fully browser-based IDE that enables users to write, execute, and manage code without local installations. In large-scale educational platforms, ensuring the functional correctness and performance stability of browser-based IDEs like Theia poses a scientific challenge due to their interactive complexity and highly concurrent usage patterns.

This thesis aims to investigate these challenges by developing a testing framework for Theia Cloud, focusing on end-to-end UI testing to verify the correctness and stability of interactive features and scalability and load testing by simulating realistic user interactions to evaluate Theia Cloud's performance under high concurrent usage. The proposed approach involves designing a maintainable Playwright test suite, developing a scalable test infrastructure, and analyzing system performance.

1 Introduction

Modern software development increasingly relies on cloud-based Integrated Development Environments (IDEs) to provide scalable, accessible, and collaborative coding platforms [Wu+11]. One such environment is Theia Cloud, a fully browser-based IDE designed for extensibility and device-independent access. Its flexibility makes it particularly attractive for educational settings, where students and instructors require consistent, ready-to-use development environments.

At the Technical University of Munich (TUM), Theia Cloud is integrated into Artemis, a learning management system that supports programming, modeling, and quiz-based exercises with (semi-)automatic assessment [Sch24]. This integration enables students to complete coding assignments directly in the browser and allows instructors to evaluate submissions efficiently [KS18]. In the Winter Semester 2024/25, the School of Computation, Information and Technology enrolled

over 14,000[1] students, many of whom actively used Artemis.

Theia Cloud connects to Artemis via a custom extension that launches a dedicated Theia instance for each student [Jan24]. While this enables a seamless development and submission workflow, the integration introduces technical challenges—particularly in terms of performance and reliability under high load. These aspects are critical during exam scenarios, where hundreds of students may interact with the system simultaneously.

2 Problem

Theia Cloud enables students to write and execute code directly in the browser as part of their coursework in Artemis. However, its reliability and responsiveness under growing user demand remain uncertain. Without systematic testing and performance evaluation, the platform may exhibit issues that affect usability and learning outcomes.

A key issue is the lack of automated end-to-end (E2E) testing to verify Theia Cloud's interactive features [Pau01]. Students frequently use functionalities such as code editing, file management, terminal access, and version control integration, all of which require a seamless user experience. However, regressions and UI inconsistencies may go unnoticed without proper automated testing, leading to unexpected failures during assignments or exams. Since developers actively maintain Theia Cloud, manual testing is impractical, leaving gaps in ensuring feature stability.

Additionally, Theia Cloud's scalability has not been systematically evaluated. During programming exams, hundreds of students interact with the system concurrently. Without structured load testing, bottlenecks and system failures may only surface under real-world conditions, causing slow responses or crashes. These failures disrupt student workflows and burden instructors with technical issues.

There is limited visibility into performance bottlenecks, which makes it difficult for system administrators to address issues proactively. Although monitoring

[1] TUM in Zahlen, 10.03.2025

tools like Grafana[2] offer insights into system metrics, without structured load testing it remains unclear when the system fails to scale or which components are most vulnerable to performance degradation.

3 Motivation

Cloud-based IDEs like Theia Cloud are increasingly important in modern education, where interactive coding environments support practical learning [Zan24]. As these platforms become more widely adopted, their reliability and scalability directly affect the learning experience. In Artemis, a well-tested Theia Cloud improves accessibility and efficiency, particularly during periods of high usage such as programming exams.

A robust and responsive IDE means seamless learning experiences without technical disruptions for students. When programming exams and assignments run without problems, participating students can focus on solving problems without any worries about potential system issues [Tur20]. Automated testing and performance evaluation ensure that code editing, execution, and version control work as expected across different scenarios, reducing frustration and improving overall engagement. Research on online learning environments highlights that system stability directly influences student satisfaction and learning outcomes [MM21].

Instructors benefit from a consistent and reliable development environment that supports the design of complex exercises without technical constraints. End-to-end testing contributes to maintaining this consistency by validating core functionality across updates.

Scientifically, testing cloud-based, interactive systems remains a relevant challenge in the fields of software quality assurance and performance engineering [Pau01]. This thesis contributes a systematic approach to evaluating the correctness and scalability of Theia Cloud, with the broader goal of supporting the reliable operation of browser-based IDEs in educational contexts.

[2]https://grafana.gchq.ase.in.tum.de/

4 Objective

This thesis aims to ensure the reliability, usability, and scalability of Theia Cloud within Artemis by developing a structured, automated testing framework. This involves both E2E UI testing and scalability testing, enabling the identification of performance bottlenecks under real-world conditions [Pau01]. The key objectives of this thesis are:

1. Develop an automated Theia E2E testing
2. Implement a scalable load testing framework
3. Analyze system performance and identify bottlenecks
4. Create randomized and personalized tests using LLMs

4.1 Develop an Automated Theia E2E Testing Suite

To ensure the correctness and stability of Theia Cloud's interactive features, this thesis develops an automated E2E test suite using Playwright[3]. The tests cover essential functionalities, including code editing, file management, terminal interactions, and version control integration, as shown in Figure 1. Since developers actively enhance Theia Cloud, automated tests help detect regressions and UI inconsistencies early in development.

This thesis designs the test suite to be maintainable and scalable, enabling future extensions as developers introduce new features. It pays special attention to handling asynchronous behavior, such as delayed UI updates and dynamic content loading, challenges commonly encountered in web-based IDEs. The implementation includes test automation best practices, such as using selectors resilient to UI changes and structuring tests modularly for reusability.

The suite integrates tests for interactions with Artemis directly into Artemis' existing CI/CD pipeline, automatically validating any changes to Theia Cloud against the test suite. This integration helps maintain a high level of confidence

[3]https://playwright.dev/

in the system's correctness and stability, reducing the risk of regressions and improving overall user experience.

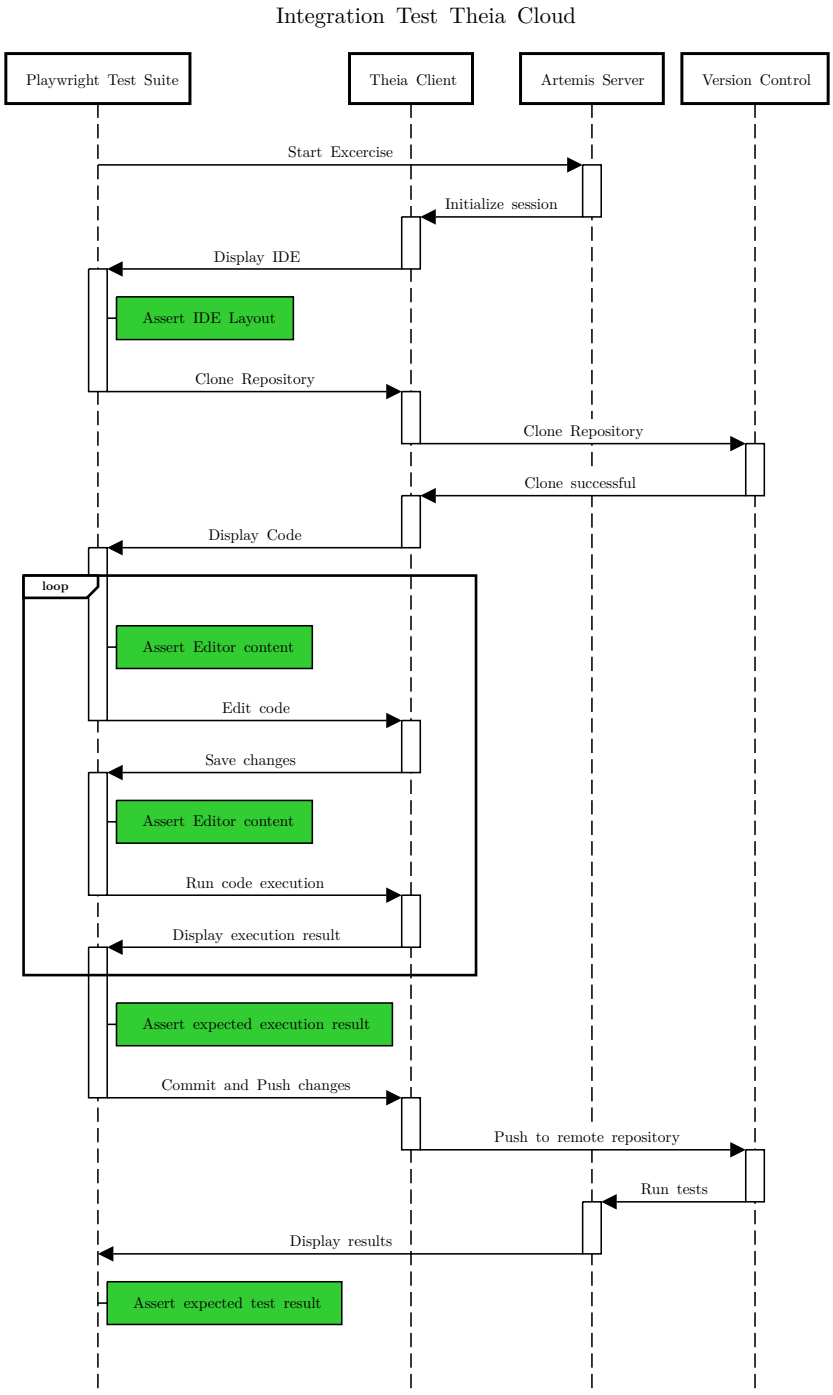

Figure 1: Sequence diagram of a complete End-to-End test scenario for Theia Cloud integrated with Artemis

4.2 Implement a Scalable Load Testing Framework

In educational settings, Theia Cloud must support hundreds of students working simultaneously, especially during programming exams [Tur20]. However, developers have not yet systematically tested its ability to handle high user loads. This thesis develops a load testing framework, as seen in Figure 2, that can simulate realistic student interactions at scale. For illustration, the diagram presents a simplified architecture of the load testing framework using an arbitrary number of instances of Theia Cloud. In reality, the framework scales up to a large number of instances of Theia Cloud, depending on the available resources. The test suite performs functional and scalability testing by interfacing with individual Theia instances using Playwright. It emulates user interactions by targeting UI elements through CSS selectors in a controlled, headless browser context.

The framework models typical user workflows such as editing code, executing programs, and using the terminal to replicate real-world usage patterns. The tests measure latency, response times, and resource consumption under increasing load. They also identify the system's breaking point, where performance degrades or failures occur, to support targeted optimizations. The goal is to develop a scalable and repeatable methodology for evaluating the performance of Theia Cloud over time.

This thesis sets up a separate testing environment to ensure that scaled tests do not interfere with the CI/CD development process. This environment is isolated from the production system, allowing for extensive load testing without impacting real users or pipeline building.

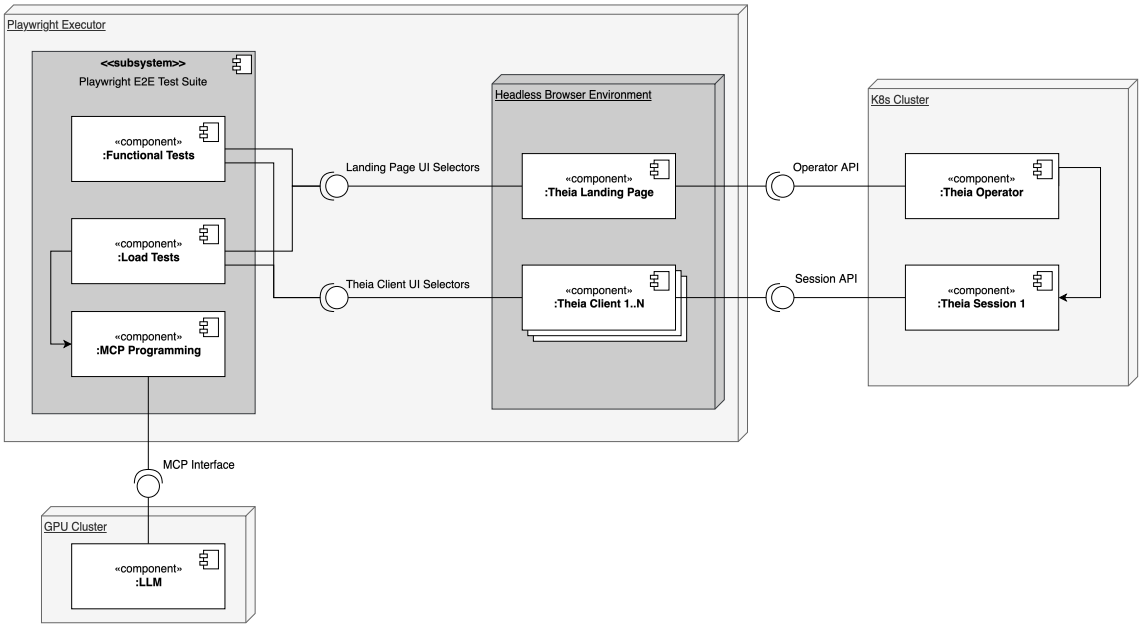

Figure 2: Deployment Diagram of the scalable load testing framework using N instances of Theia Cloud

4.3 Analyze System Performance and Identify Bottlenecks

An important aspect of performance evaluation is understanding how system components behave under load. This thesis uses Grafana and other available monitoring tools to analyze key performance metrics, including CPU and memory usage, response times, and request throughput.

By correlating test results with system metrics, this analysis provides actionable insights into performance bottlenecks, such as slow UI interactions, server-side processing delays, or database constraints. The goal is to comprehensively evaluate Theia Cloud's scalability limits, helping system administrators and developers make informed decisions about optimizations and infrastructure scaling.

4.4 Create Randomized and Personalized Tests Using LLMs

The Model Context Protocol[4] (MCP) is a tool for interacting with LLMs with context. By leveraging the MCP, this thesis also explores the possibility of generating tests that adapt to individual user profiles, preferences, and learning styles. This approach aims to enhance the realism and relevance of the testing scenarios, making them more representative of actual user interactions.

The thesis uses the MCP to create diverse test cases covering various programming languages, IDE features, and user behaviors. This allows for a more comprehensive evaluation of Theia Cloud's capabilities and performance under different conditions. The MCP component integrates the generated tests into the existing E2E test suite and thoroughly tests the system across various scenarios.

5 Schedule

The following key milestones structure the implementation timeline of this thesis. The Bachelor's thesis starts on 03.07.2025 and ends on 03.11.2025. Each milestone represents a concrete capability or achievement necessary to reach the final goal of a scalable, reliable testing solution for Theia Cloud within Artemis.

- **Milestone 1 (by 30.07.2025): Functional Integration Tests in Theia**

  A developer can run functional integration tests in Theia IDE locally using the Theia IDE Docker image. The tests can be executed directly on the production instance of Theia IDE and cover the following aspects:

  ▸ Basic functionality of the IDE (e.g., editor, terminal, file navigation)

- **Milestone 2 (by 30.08.2025): Integration into Theia Cloud**

  A developer can run functional integration tests on the production and testing server in Theia Cloud. The tests functionally test the landing page and login functionality. Integration with Artemis is tested by executing E2E tests in the Artemis repository. The tests cover the following aspects:

[4] https://modelcontextprotocol.io/

- Basic functionality of the Landing Page (e.g., login, navigation)
- Basic functionality of the Artemis integration (e.g., submitting exercises, checking results)
- Testing of multiple Programming Languages (e.g., Java, Python, C) and their respective compilers
- Testing of building and running of projects

- **Milestone 3 (by 30.09.2025): Load and Scalability Testing**

  A developer can run automated (using GitHub Workflows) load and scalability tests in Theia Cloud using Playwright Artillery on the production and testing server. The tests cover the following aspects:
  - Load testing of the IDE with multiple users
  - Scalability testing of the IDE with multiple users
  - Performance testing of the IDE with multiple users

- **Milestone 4 (by 03.11.2025): LLM/MCP Integration**

  A developer can run randomized, realistic test cases reflecting student behaviors and programming environments. By modeling context dynamically, tests adapt to a wide range of scenarios, including different programming languages and usage patterns. This enables scalable and flexible test generation that better simulates real-world conditions in systems like Theia Cloud.

Bibliography

[Wu+11] L. Wu, G. Liang, S. Kui, and Q. Wang, "CEclipse: An Online IDE for Programing in the Cloud," in *2011 IEEE World Congress on Services*, Washington, DC, USA: IEEE, July 2011, pp. 45–52. doi: 10.1109/SERVICES.2011.74.

[Sch24] Y. Schmidt, "Inclusive Learning Environments in the Cloud: Scalable Online IDEs for Higher Education," *Master Thesis at TUM*, Dec. 2024.

[KS18] S. Krusche and A. Seitz, "Artemis: An Automatic Assessment Management System for Interactive Learning," in *Proceedings of the 49th Technical Symposium on Computer Science Education (SIGCSE)*, ACM, 2018, pp. 284–289. doi: 10.1145/3159450.3159602.

[Jan24] D. Jandow, "Scorpio: A Visual Studio Code Extension for Interactive Learning Platforms," *Bachelor Thesis at TUM*, Nov. 2024.

[Pau01] R. Paul, "End-to-End Integration Testing," in *Proceedings Second Asia-Pacific Conference on Quality Software*, Hong Kong, China: IEEE Comput. Soc, 2001, pp. 211–220. doi: 10.1109/APAQS.2001.990022.

[Zan24] P. Zander, "Developing an Improved Code Editor for Learning Platforms," *Master Thesis at TUM*, Sept. 2024.

[Tur20] A. Turdiu, "Online Exams in Artemis," *Master Thesis at TUM*, Nov. 2020.

[MM21] C. Müller and T. Mildenberger, "Facilitating Flexible Learning by Replacing Classroom Time with an Online Learning Environment: A Systematic Review of Blended Learning in Higher Education," *Educational Research Review*, vol. 34, p. 100394, Nov. 2021, doi: 10.1016/j.edurev.2021.100394.

Transparency in the use of AI tools

In preparing this proposal, I used Grammarly to improve grammar and style, ensuring clarity and consistency throughout the text.

I also used ChatGPT to assist in generating initial drafts and expanding ideas. I carefully reviewed and revised all AI-generated content to ensure it is accurate, context-appropriate, and aligned with the proposal's objectives.