Subhranil Patra
Email: subhranilpatra69@gmail.com
Phone: 6294833235
Github: https://github.com/Subh09
Time zone: UTC+5.30
Linkedin : www.linkedin.com/in/subhranil-patra-797582163
Town: Bankura,
State: West Bengal
Country: India

GSoC 2025 Proposal: Webpack Benchmark Tooling

Project Details

Difficulty: Hard

Project Size: 90 hours

Problem Statement

Performance bottlenecks and regressions can adversely impact Webpack's usability. Currently, maintainers lack a reliable and automated system for evaluating the performance implications of pull requests. This gap makes it challenging to identify and address regressions before they merge into the codebase.

Objectives

I) Develop a CI-integrated benchmarking system for Webpack.

II) Automate the execution of performance benchmarks for every pull request.

III) Provide maintainers with clear insights into performance impacts, enabling swift action.

IV) Foster better feedback mechanisms for contributors, encouraging quality submissions

Proposed Solution -

We will develop a robust benchmarking system that seamlessly fits into Webpack's existing CI pipeline. The system will include automated running of benchmarks,end-to-end performance reporting, and regression trend v isualization tools. Contributors will receive actionable feedback, and maintainers will be able to block significant regressions early.

Deliverables -

1. **Benchmarking Framework**:
    - A CI-integrated tool to automate benchmark execution for pull requests.
    - Mechanisms to analyze and report key performance metrics.
    - Features to track and visualize trends in regression data.
2. **Documentation and Testing**:
    - Comprehensive user and developer documentation.
    - Unit tests to validate the reliability and accuracy of benchmarks.
    - End-to-end tests for the integrated CI pipeline.

Tech Stack

- **Languages**: JavaScript, NodeJS, CSS, and HTML.
- **CI Tools**: GitHub Actions, Jenkins, or similar frameworks.
- **Performance Tools**: Benchmark.js, Webpack Bundle Analyzer, etc.

Documentation and Tests

Documentation

- **Developer Documentation**: A comprehensive guide detailing the architecture, implementation, and functionality of the benchmarking system. It will include:
    - Overview of the benchmarking framework.
    - Setup instructions for maintainers.
    - How-to-use guides for contributors to interpret performance feedback.
- **User Documentation**: User-friendly instructions tailored for contributors, describing the submission process and how benchmarks are triggered and evaluated.

Testing

- **Unit Tests**: Focused on validating individual components of the benchmarking framework, such as the execution module and reporting functionalities.
- **Integration Tests**: Ensure seamless integration of the benchmarking tool with Webpack's CI pipeline.
- **End-to-End Tests**: Validate the entire process—from triggering benchmarks on a pull request to generating performance reports.
- **Regression Tests**: Focused on detecting regressions in performance metrics, ensuring historical baselines are preserved.

Expected Outcomes

1. A fully functional benchmarking system integrated into Webpack's CI pipeline.
2. A reliable mechanism to detect and block performance regressions.
3. Clear documentation to guide maintainers and contributors in using the system.
4. Enhanced collaboration and feedback mechanisms to improve contributor experience.

Timeline

Pre-GSOC Work:

To get familiar with the project and show my interest, I have started working on Webpack's benchmarking system. I created a script that runs benchmark cases using Webpack's Node API and records performance metrics like execution time.

I also added a helper file to handle benchmarking logic and a basic GitHub Actions workflow to automate it in CI. This setup helps detect performance regressions early.

Key Components

- **`scripts/runBenchmarkCI.js`**
  A script that detects and runs all benchmark test cases in the `test/benchmarkCases` directory and saves performance results in a structured JSON format.
- **`test/helpers/runBenchmark.js`**
  A helper utility that uses the Webpack Node API and `Benchmark.js` to run each case and collect statistics like mean execution time and sample size.
- **`benchmark.yml`**
  An initial GitHub Actions workflow file to enable CI-based benchmarking. This is currently a work in progress.
- **Pull Request:**
  feat/benchmark-ci-setup

May 8–May 14

- Begin the community bonding period.
- Read Webpack documentation thoroughly.
- Familiarize myself with the Webpack codebase, focusing on performance tracking and CI modules.
- Schedule an introductory meeting with mentors Alexander and Nitin to discuss project goals, expectations, and deliverables.

May 15–May 21

- Set up the development environment, including tools, libraries, and dependencies.
- Research existing benchmarking frameworks and tools like Benchmark.js and Webpack Bundle Analyzer.
- Share a preliminary plan and initial observations with mentors for feedback.

May 22–June 1

- Finalize the project plan based on mentor feedback.
- Review Webpack’s current CI setup to identify integration points.
- Complete initial preparations for the coding phase.

June 2–June 8

- Coding officially begins!
- Define the architecture and workflow for the benchmarking framework.
- Create a basic prototype of the benchmarking system.
- Start integrating the prototype with Webpack’s CI pipeline.

June 9–June 15

- Conduct initial testing of the benchmarking framework to identify and resolve issues.
- Implement basic reporting functionality to visualize performance metrics.
- Collaborate with mentors to refine the system based on feedback.

June 16–June 22

- Expand core functionalities of the benchmarking system, such as automated benchmark execution.
- Begin developing unit and integration tests for the framework.
- Document early progress and update mentors on key milestones.

June 23–June 29

- Conduct thorough testing of the prototype for accuracy and reliability.
- Address bugs or inconsistencies identified during testing.
- Continue iterating on the framework to enhance usability.

June 30–July 6

- Finalize the initial version of the benchmarking system.
- Ensure compatibility with Webpack’s existing development workflows.
- Prepare for the midterm evaluation by consolidating testing results and documentation.

July 7–July 13

- Conduct a final review of the system before the midterm evaluation.
- Submit the benchmarking framework for evaluation.
- Gather feedback from mentors and plan revisions for the next phase.

July 14–July 20

- Work on enhancements based on midterm evaluation feedback.
- Start implementing regression detection and performance trend analysis tools.
- Begin testing the new features for accuracy and functionality.

July 21–July 27

- Collaborate with mentors to refine regression detection mechanisms.
- Conduct quality assurance testing for all features in the benchmarking system.
- Address issues related to usability and performance metrics.

July 28–August 3

- Finalize regression detection tools and trend analysis features.
- Prepare the benchmarking system for deployment in Webpack's CI pipeline.
- Begin drafting user and developer documentation.

August 4–August 10

- Complete and review the documentation with input from mentors.
- Conduct testing of the system in real-world scenarios using actual Webpack PRs.
- Begin fixing any remaining bugs or inconsistencies.

August 11–August 17

- Finalize deployment of the benchmarking system in Webpack's CI pipeline.
- Conduct extensive testing to ensure seamless operation and integration.
- Continue collaborating with mentors to polish the system.

August 18–August 24

- Prepare the final project deliverables for submission.
- Conduct end-to-end testing of the deployed system.
- Address any last-minute issues and finalize documentation.

August 25–September 1

- Submit the final work product and mentor evaluation.
- Ensure all deliverables are complete and meet project expectations.
- Celebrate the successful completion of the project!

After GSoC:

Continue maintaining and improving the benchmarking system developed during GsoC.

- Add more benchmark test cases to cover a wider range of Webpack features and use cases.
- Optimize the benchmarking workflow for better accuracy, speed, and developer experience.
- Assist in detecting and resolving performance regressions using benchmark reports.
- Contribute to documentation to help onboard new contributors to the benchmarking system.
- Stay involved with the Webpack community and contribute to other areas as needed.

Mentor Information

Mentors: Alexander, Nitin Both mentors are highly experienced in performance tooling and CI integrations, and will provide the guidance necessary to ensure the project's success.