Calum McCartan
cm4114

Project Second Progress Report

Overview
For my project, I am working on implementing an extension for Visual Studio Code that assists with introducing parallel Web Workers into JavaScript programs in order to improve performance of intensive tasks.

Research Questions

- How does the automatically refactored code compare with manually refactored code?
- Is this tool useful? What performance gain is obtained by the refactored code?
- Which of the parallel refactoring tool objectives (specified in my paper) does the tool accomplish? Would it be possible for this particular tool to achieve more of these objectives?
- What restrictions are there on where the tool can be used?
- What can the tool do that a library could not?
- What improvements/features does the tool have the potential for if I continued to develop it?

Challenges
This is the first time I have written a custom VSCode extension and so this took some time to learn. It also took me some time to get the Babel library working, however I now have the extension working so that I can select, parse, and automatically modify a highlighted region of code and am currently working on actually extracting features of the selection and correctly placing them into web workers.

Demo
I plan to demo my project by demonstrating how my VSCode extension is used - selecting a region of code, choosing the option to refactor the selected loop into parallel web workers, and showing the replaced code. I will run the code before and after to demonstrate the speedup.

Changes
The novelty, value to user community, and comparison subjects are currently the same as in the first progress report (as below).

Novelty
This project follows the same subject as my midterm paper - parallelism refactoring tools, but unlike the papers I covered, my project targets the JavaScript language, and the VSCode IDE. Not only is VSCode a novel choice for this type of tool, but it is also a very popular IDE for JavaScript / web applications, and it has significant support for writing custom extensions. JavaScript is a novel choice for this type of tool likely because it is not typically a language where parallelism is used for performance, however, there are definitely still use cases.

Value to User Community
While using web workers in JS is easier than using parallelism in other languages, it will still be beneficial for those who are new to using Web Workers, and even experienced users should find that the tool can save them some substantial time. This extension will be useful to developers when writing web applications that perform some intensive client side tasks (eg. file processing).

Datasets
I will first write some very small example programs that can be used to test and clearly demonstrate how the extension is used. Next I will look for one or more larger existing applications where my project can hopefully be used to improve performance. This is the approach taken by similar work when evaluating a parallelism refactoring tool.

Comparison Subjects
In my midterm paper I identified some goals of parallel refactoring tools. I will evaluate the extent to which my tool covers these, and compare it to the features of other tools which I covered in my paper. In particular I will be able to compare the workflow and features of my extension with that of the Paraformance tool which I evaluated in my midterm paper experiment.

Delivery
The project will be made available in a GitHub repository. The extension will either be made available for download on the VSCode marketplace, or will at least be available for manual install by copying from the GitHub repo into a user's VSCode extensions directory.