Project Proposal: Automatic Differentiation

Natthapee Sriarunluck 6480266

Objective

The primary objective of this project is to implement and parallelize an Automatic Differentiation (AD) system using Scala. The objective is to compare approaches to parallelism/concurrency such as Future and Promises and threading. If time permits, the implementation will be extended to the Rust programming language for comparative analysis.

To present the result, a simple GUI may be written to show results for each approach. The user may enter the equation they want to be differentiated, the mode of the AD, and which method of parallelization they want to use.

Approach

1. **Base Implementation:** Develop a base AD system in Scala using a parser to handle mathematical expressions. This will serve as the foundation for parallelization.
2. **Parallelization Implementation:** Identify independent computational tasks within the AD process and parallelize them using Scala's different approaches.
3. **Performance Comparison:** Measure and compare the performance of the sequential, Futures and Promises, and threading implementations. Use benchmarking tools to assess runtime and resource utilization.
4. **Optional Rust Implementation:** If time permits, implement the AD system in Rust and parallelize it using similar techniques. Compare its performance with the Scala implementations.

Success Measurement

To measure the success of the project, the following criteria will be used:

- **Correctness:** Ensure that the Automatic Differentiation system produces accurate and reliable derivatives for a variety of input functions. The correctness will be verified through comprehensive testing against known derivatives.
- **Performance:** Achieve significant performance improvements through parallelization. This will be quantified by measuring the runtime and resource utilization of the parallel implementations (Futures and Promises, threading) compared to the sequential implementation.
- **Usability:** Develop a user-friendly interface that allows users to input functions, select the mode of AD, and choose the parallelization method. The interface should provide clear and understandable output.

Through this project, the aims are to practice and implement different approaches to concurrency/parallelism, practice writing concurrency/parallelism in both Rust and Scala, and explore potential extensions to Machine Learning concepts.