Project Proposal

Comparing and Contrasting Performance Metrics for RUST and C/C++, by Detecting and Analyzing Phase Change Metrics.

Kushagra Srivastava

Under Guidance of Prof. Meng-Chieh Chiu

Statement of Objectives

The following Research Proposal draws inspiration from and is used as an extension of the paper, "Real-Time Program-Specific Phase Change Detection for Java Programs", authored by Prof. Meng-Chieh Chiu, Prof. Eliot Moss, and Prof. Benjamin Marlin.

During the compilation of a program in a given programming language, the program goes through different phases, such as: initialization, parsing, semantic analysis, optimization, and code generation. By analyzing phase changes in different programming languages, we can notably infer how quick and robust a given programming language can be in these stages. This data may help us in comparing performance metrics between different programming languages, as well as provide tools for other developers or researchers for further implementations.

The main idea behind this project would be to detect Real-time Phase Changes in the compilation and execution of different programming languages in different languages. Notably, we plan to target newer programming languages (RUST, in our case), and compare their performance against established System Programming languages like C/C++.

We plan to detect the phase changes in a very similar way as outlined in the above-mentioned paper: by collecting certain benchmark points, and creating a real-time phase-change detector in order to correctly analyze the performance. The model to detect the phase changes would be based on the one implemented for Java programs in the above-mentioned paper: it would detect phase changes through recording various time intervals between different phases, and to cluster them based on the similarity of their feature vectors: the number of similar properties that specific phase consisted of (which would tell how similar or different a given phase is). These metrics would then be analyzed using the Gaussian Mixture Model (GMM): which is a “probabilistic model generalizing k-means clustering to incorporate information about the covariance structure of the clusters.” (Chiu *et al.*, 2016)

The benchmarks which would be used under this project are still under consideration, but we believe that since we are dealing with a relatively newer language, there may exist a greater flexibility in terms of observable properties, thus leading to a better understanding and wider range of data collection.

The following independent study would help us get a better picture on how newer languages may be better than older languages, especially when both languages are used for very specific and similar purposes (Systems Programming, in the case of this project). Both RUST and C offer a high degree of flexibility in terms of user control over processes, memory, and are very strongly typed. However, RUST benefits from the fact that it is relatively newer, has better memory management, and focuses explicitly on preventing memory leaks and minimizing user error. Thus, it may be interesting to see how a newly developed language performs, as compared to an older, established language.

Such an insight would help programmers working in Systems gain a better understanding of the robustness of newer languages as compared to older ones, and may help in choosing the best fit for their use case. Moreover, this project would also act as a means of contributing valuable RUST tools for programmers and researchers.

Criteria for Evaluation, and Planned Activities

The main criteria for evaluation for the project would be two-fold: a paper describing the different findings and analyses of the project, as well as the tools developed for the programming languages which would help us track the Phase Changes. However, since the project involves a huge development effort across various programming languages, a log and a code repository may be used for evaluation purposes in the initial stages.

The project takes place across the Spring and the Fall 2023 semester, with a final paper by Spring 2024 (if the additional time is required and necessitated). This is because I plan to make this Independent Study my Honors Thesis in the future semesters.

A detailed plan of the study is as follows:

1. Spring 2023: Gather relevant research work in order to choose the Benchmarks needed to be tested, and design and develop the Performance Measuring Tools (the Phase Change Detector) for RUST, and confirm if it works as expected. Continue working through Summer, if necessary. For this time period, we will be evaluating based on a weekly log, a project repository, and weekly meetings with the professor.
2. Fall 2023: Gather data for the Phase Changes for various different RUST programs. Start development for similar tools in C/C++, and conduct the same. Criteria for evaluation remains the same. Moreover, this research becomes part of my Honors Thesis.
3. Fall 2023/Spring 2024: Finalize final details of project, work on compiling findings in a proper research paper.

My expectation for the project for the upcoming Spring Semester would be to conduct it for about 3 credits, which roughly translates to about 8-10 hours per week. However, I would

also take this project over the next Fall semester as well, as part of my Honors Thesis under the Computer Science Departmental Honors. The timeline above assumes a conservative estimate, and the actual project may be completed in a shorter timespan.

For the Spring 2023 Semester, I believe a deeper timeline as follows:

1. Understanding the RUST documentation, and interaction with System Calls, in-built functions, and other open source tools that have been developed for the same. The process will help in understanding the different system calls and functions that the language supports, and help in creating a more precise roadmap for the entire project. (2 weeks)
2. Searching, shortlisting, and creating the benchmarks to be tested: Once we have a better picture of how program compilations, executions, garbage collection, and the like take place on RUST and on C, we can select what benchmarks we want to take to be able to compare between the two. (2 Weeks)
3. Design and Development of the Performance Measuring Tools (7 weeks): During this time, we will be focusing on the development of tools to measure performance across compilation, execution, etc. take place on the RUST programming language. We plan to use a similar method as described in Prof. Chiu's paper: "Real-time Program Specific Phase Change Detection for Java Programs", where we utilize Gaussian Mixture Models to cluster different time intervals detected across our benchmarks. We assume the breakdown to be as thus:
    a. Developing tools that utilize system calls and the like to detect the execution and completion of different benchmarks, and recording the time intervals for RUST (2 weeks).
    b. Perfecting the same and trying to run it across different programs to check for differences, and record observations. (2 weeks).

    c. Creating the tools that take into account the time intervals, and conduct analyses on given data (GMM algorithm as described above). (3 weeks).

4. Debugging and Verification of the tools created (2 weeks): At this stage, we would verify if the tools created are working correctly by checking programs of different types, sizes, and different processes. We hope to clear out errors and bugs in this stage, so that we can move forward to actually collecting data and working with the tools to make proper observations and compare the two programming languages.

At this stage, we would also move the project further during the Fall 2023 semester, when we would start taking the data and working with these tools to make proper observations, and start working on the actual research paper. While we expect a lot of the details for the Fall 2023 semester to become more concrete as we move forward with this project, we predict that we would develop the exact tools for the C programming language during the Fall semester, and also conduct testing of various different programs on RUST and take our observations.

By the Spring 2024 semester, we would like to test the same for C, as well as wrap up on the final project paper. I would like to use 3-4 credits each semester for the same (or roughly 10-12 hours per week), however I may opt in for a higher or lower number of credits in a future semester as deemed necessary.

Throughout the semester (and the following two semesters), I would like to set up a weekly meeting with Professor Chiu to discuss the progress of the project, as well as be open to setting up more meetings throughout the semester to discuss and approach the project as needed. Moreover, I would also be keeping a log of all the advancements and work put in on a weekly basis, as well as maintaining a git repository of all our code. The repository would help in releasing the project code to the general public, as well as help in keeping a log of all advancements made in the code process.