Research Proposal for CS4900: UGRC-I Designing a Static Compiler-Assisted Prefetching Scheme

Mohammed Rizan Farooqui
CS20B052
Guide: Prof. V. Krishna Nandivada

April 2023

1 Background

In languages like Java, programs are not converted into executable files, but rather into bytecode, which are then executed at runtime by the JVM. We say it follows a two-step compilation i.e., static and JIT (Just In Time) compilation. JIT compilers often forgo the accuracy of program analysis in favor of efficiency as the time required for JIT compilation is added to the application's execution time. Statically analyzing the entire program as an alternative ignores the examination of libraries, which are only accessible during runtime, and produces inaccurate conclusions. This problem is addressed by the PYE(Precise Yet Efficient) framework [4] by producing accurate analysis findings at runtime at a very low cost.

To summarize, the current JIT compilers and managed runtimes like JVMs frequently do not execute numerous expensive analyses as it affects the overall execution time, and which in turn results in a significant amount of optimization chances being lost. As a result, many of the features offered by modern architectures are not fully being used in Java applications, which represent one of the keys to be resolved [4].

2 Goals for the course

There are currently no works that make use of PYE's capabilities to make use of modern architectural characteristics to optimize low-level memory operations, despite the fact that PYE is effective at providing high-level compiler analysis and compiler optimizations.

The goal of this project is to design and implement a static-compiler-assisted prefetching scheme for Java programs using the PYE framework, with the aim of optimizing low-level memory operations during JIT compilation. Specifically using prefetching techniques available in modern processors [1]. This project also aims to contribute to the ongoing research efforts in compiler-assisted prefetching techniques [2] [3].

3 Tentative Plan

The project will follow the following plan:

1. Conduct a comprehensive analysis of the available prefetch instructions supported by modern processors, with a specific focus on the Intel Sapphire Lake processor.

   This will involve gathering relevant literature and conduct a detailed analysis of the available prefetch instructions. We will identify the most relevant and effective instructions that can enhance the performance of Java programs during JIT compilation.

2. Conduct a thorough review of existing research papers and develop an algorithm that can effectively leverage performance gains from prefetching in a wide range of scenarios.

   Based on the analysis conducted in step 1, we will review existing research papers on prefetching and develop a general-purpose algorithm that can effectively leverage performance gains from prefetching. The algorithm will be designed to be adaptable to different applications, with a specific focus on irregular programs such as graph analytics.

3. Evaluate the performance gains from the solution developed in a general use case with some runtime information present.

   2 To assess the effectiveness of the developed algorithm, we will evaluate its performance gains in a general use case scenario. The evaluation will be conducted using a set of standard benchmarks and some runtime information. This will enable a comparison of the performance of the Java program with and without the developed algorithm, to determine the extent to which the algorithm can improve program efficiency.

4. Utilize the algorithm developed above and leverage PYE's accurate analysis findings at runtime to prefetch data, which can significantly improve the efficiency of Java programs during their JIT compilation.

   Utilizing the developed algorithm, we will leverage PYE's accurate analysis findings at runtime to prefetch data. This will enable significant improvement in the efficiency of Java programs during their JIT compilation by reducing memory access latency and enhancing the efficiency of low-level memory operations.

5. Document and publish the technical details and evaluations performed in a manuscript.

   We will document and publish the technical details and evaluations performed in a manuscript to disseminate the research findings and contribute to the existing literature on prefetching techniques in Java programs.

The above plan will be carried out over the duration of the project, with each step building on the previous one. The goal is to develop a comprehensive solution that can significantly improve the efficiency of Java programs during their JIT compilation.

Note: It is most likely that the fourth and fifth components of the tentative plan will be completed as part of another UGRC/BTP.

References

[1] Intel 64 and ia-32 architectures software developer manuals.

[2] Brad Beckmann and Xidong Wang. Adaptive prefetching java objects.

[3] Sparsh Mittal. A survey of recent prefetching techniques for processor caches. *ACM Computing Surveys (CSUR)*, 49(2):1–35, 2016.

[4] Manas Thakur and V Krishna Nandivada. Pye: a framework for precise-yet-efficient just-in-time analyses for java programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 41(3):1–37, 2019.