Implementation of Automated Loop-bound Analysis for Static Timing Analysis Framework

Balaji V. Iyer and Won So
{bviyer, wso}@ncsu.edu
http://www4.ncsu.edu/~wso/csc714

Fall 2005 CSC714 Project Proposal

1. Introduction

Many existing timing analyzers require that user specify the number of iterations for each loop in the program [Healy98]. The bounds of every loop must be specifically provided by the user. This can give room too many possible errors. For example, the user might have modified the code and have forgotten to modify the annotations. Errors in these annotations can drastically affect the Worst Case Execution Time (WCET) or the Best Case Execution Time (BCET). Compilers today can implement many optimizations that are not implemented before. For example, compilers can software pipeline loops, which can affect the given annotations.

It would be very beneficial if the compiler analyze the optimized code during compile time and provide the appropriate annotations about loop bounds, induction variables and so forth. Some timing analysis tools have the capability to perform these automations [Byhlin05]. However, the available timing analysis framework does not have the ability to extract this information from the given code.

The main purpose of this work is to incorporate this feature in our timing analysis system. We propose to analyze loops in the code and let the compiler provide the appropriate annotations. These loop bounds are passed to the timing analyzer along with the user's annotations.

In the next section, we explain the goals. Section 3 describes different steps necessary to complete the specific project goals.

2. Project Goals

These are the major milestones we propose to do:

- We propose to find the loop bound for single-entry, single-exit loops (excluding the back-edge) with one induction variable, which only determines the loop exit condition. For the rest of this document, we call this type of loops, "simple loops".

- If time permits, we plan to extend this work for more complicated loops such as:
    a. Loops with multiple induction variables.
    b. Loops with multiple exits [Healy98].
    c. Conditionally executed inner loops.

3. Project Plan

The steps to complete this project are as follows:

1) Survey the existing static-timing analysis framework to find the implemented functionality that could aid this project such as control flow graph (CFG) construction. At this point, we are planning to exploit optimizing backend compiler framework 'opt' which construct useful set of information such as CFG from Sparc assembly code. Therefore, our first job is to modify this tool to work with PISA assembly code.
2) Set up an algorithm to find the loop-bound of simple loops. For this we maybe have to reference the existing algorithms and implementations such as [Ermedahl97] and VPO.
3) Implementation of these algorithms. This step may involve detection of loops, induction variables and loop exit conditions.
4) If time permits, we will try to implement support for complex loops described in step 2 in section 2.
5) Find appropriate benchmarks to evaluate our optimizations on the tools and compare the results with the base system.

4. Disclaimer

This paper and associated software changes are intended for **informational purposes only**. Therefore, any use of the information presented in this student work is at your own risk. Balaji V. Iyer, and Won So provide **no warranties of any kind** surrounding the use of this material.

5. References

[Byhlin05] D. Byhlin, A. Ermedahl, J. Gustafsson, B. Lisper. "Applying Static WCET Analysis to Automotive Communication Software," ECRTS'05.

[Healy98] C. Healy, M. Sjodin, V. Rustagi, D. Walley, "Bound loop iterations for Timing Analysis," RTAS'98

[Ermedahl97] A. Ermedahl and J. Gustafsson "Deriving Annotations for Tight calculation of Execution-Time", Proceedings of the European Conference on Parallel Processing, 1994