CIE 6020 Information Theory (Spring, 2018)

Course Project: Profile-Guided Source Coding in Dual-Compilation

Name: Chenhao, WU ID Number: 117010285

- This cover sheet must be signed and submitted along with the report on additional sheets
- By submitting this report with my name affixed above,
  - I acknowledge that I am aware of the University policy concerning academic misconduct (appended below),
  - I attest that the work I am submitting for this assignment is solely my own, and
  - I understand that suspiciously similar homework submitted by multiple individuals will be reported to School and Registry for investigation.
- Academic Misconduct in any form is in violation of CUHK(SZ)'s Student Disciplinary Regulations and will not be tolerated. This includes, but is not limited to: copying or sharing answers on tests or assignments, plagiarism, having someone else do your academic work or working with someone on homework when not permitted to do so by the instructor. Depending on the act, a student could receive an F grade on the test/assignment, F grade for the course, and could be suspended or expelled from the University.

Profile-Guided Source Coding in Dual-Compilation

Chenhao, WU

Abstract

1 Introduction

Within this decade, the computing performance of mobiles and embedded systems have been significantly raised such that more computing workloads can be undertaken on mobiles and embedded systems, such as digital image processing, augmented-reality applications, and so on. In practice, one of the critical bottlenecks in mobiles and embedded systems design and optimization is the run-time memory space and power consumption. Since the complexity of program and data set grew much faster than the performance of memory capacity in the mobile and embedded system, rather than merely use expensive computing components to feed the needs of applications with high computing overhead, it is more promising to find ways to best utilize the system use of memory space. To realize this objective, we proposed a profile-guided source coding (PGSC), which will perform an adaptive compiler-guided data compression on memory space during run-time according to different applications and run-time environment.

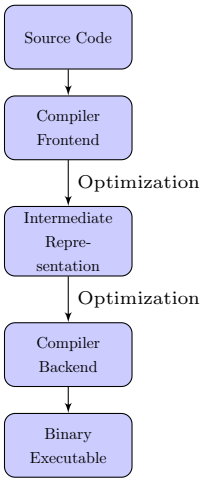

Figure 1: Traditional Compilation Procedure

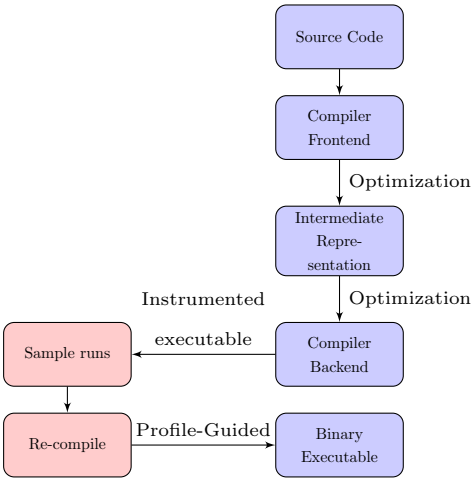

Figure 2: Compilation Procedure with PGO

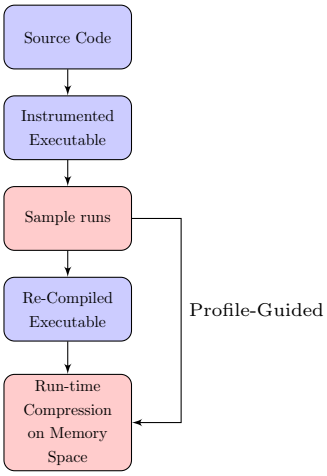

Figure 3: Profile-Guided Source Coding

The idea of this approach comes from a modern compilation technique, Profiled-Guided Optimization (PGO), that as shown in Figure 1 and Figure 2, differed from traditional approach, the compiler will

perform a recompilation based on initial executable and sample runs. A profile, for instance, breaks the source code into blocks and devises a table of the frequency each block is executed via sample runs, and during recompilation, code blocks will be reordered with respect to the frequency table. For example, blocks with higher frequency will be allocated in memory area with lower memory address and blocks with lower frequency will be allocated in higher address so that the additional I/O overhead consumed in function callbacks will be reduced. Similar to previous compiler optimization methods, like the deletion of unreachable code, the constant folding optimization and propagation optimization, the objective of PGO is to further reduce the redundancy of unnecessary and/or unimportant logic branches in the source program.

The usage of this approach, however, can be extended into attempts to reduce the memory space, that after sample runs, not only the frequency of code blocks can be collected, but also the preliminary investigation of frequency that each block of memory is accessed can be gathered. As shown in Figure 3, the procedure of PGSC consists of a dual-stage compilation, that the compiler will initially compile the source code into an instrumented executable, and based on a profile generated after sample runs, the compiler will re-compile the source code such that during the run-time, the program will compress specific areas of memory space in order to reduce the redundancy of memory use, and will instantaneously decompress the compressed contents whenever the program needs.

In the trend that more and more mobile applications buffer long arrays of audio, images and video clips into the memory space during run-time, compared with the static data compression and the original version of PGO, .

2 Related Works