Execution Time Disambiguation Profiling

by

Aniket Kumar Lata

B.E., University of Mumbai, 2012

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

2016

This thesis entitled:
Execution Time Disambiguation Profiling
written by Aniket Kumar Lata
has been approved for the Department of Electrical and Computer Engineering

_____

Prof. Pavol Černý

_____

Prof. Ashutosh Trivedi

_____

Prof. Bor-Yuh Evan Chang

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Lata, Aniket Kumar (Electrical and Computer Engineering)

Execution Time Disambiguation Profiling

Thesis directed by Prof. Pavol Černý

We consider the problem of developing an efficient cost model for Java Bytecode, where cost is determined in terms of execution time and memory usage. The key insight is that the execution time of Java applications can be predicted by modeling these applications based on the method invocation counts and their linear fitting to get an accurate estimate. Cost estimates for an application can be useful in determining parameters such as worst case execution time which can form a basis for static analyses and can be useful in distributed applications.

Dedication

To all my family, friends for the support and guidance.

Acknowledgements

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

The cost model provides estimates pertaining to the execution time for expensive methods in Java bytecode. Java applications are profiled to generate:

(1) Counts of method invocations and

(2) Total execution time of the application for a large number of runs.

This profiling is performed with the help of bytecode instrumentation. The data generated from these runs is fed to a Linear Regression toolbox to efficiently predict the "time per method" metric for that specific application.

Chapter 2

Motivating Examples

Chapter 3

Model

Our cost model needs to predict the value of the dependent variable – execution time, using a linear function of the independent variables – method counts.

A naive approach to bucketing would be accounting for each method used in the application as a separate bucket. The standard error for coefficients in multiple regression can be high if we don't account for the correct subsets of independent variables in the model. The variance of coefficient for a variable could increase if there is high correlation among with another variable. The mean squared error for all the variables gives a good indication of the estimate provided by the cost model. Even though a model shows an ideal coefficient of determination, it may not be sufficient to give an accurate estimate for an independent test application using the same independent variables i.e. methods in our case. There is a need to refine the cost model to address the standard error among variables within a model and the mean squared error for accurate estimation of test applications.

Model refinement addresses two specific problems: **Subset selection** and **Estimation Accuracy**.

3.1 Subset Selection

An important decision to be made is the bucketing (categorization) of methods in a Java application to perform multiple linear regression. This is termed as "subset selection" for the model. The naive approach considers all independent variables in the data for prediction. There are several reasons why this could be undesirable:

- Estimates of regression coefficients are likely to be unstable due to multi-collinearity in models with many variables.
- We get better insights into the influence of regressors from models with fewer variables as the coefficients are more stable for parsimonious models.
- It can be shown that using independent variables that are uncorrelated with the dependent variable will increase the variance of predictions.
- It can be shown that dropping independent variables that have small (non-zero) coefficients can reduce the average error of predictions.

Chapter 4

Algorithm

Chapter 5

Implementation

Chapter 6

Experiments

Chapter 7

Conclusion