Energy consumption of Garbage Collectors

Auteur :
Anis TELLO

18th January 2016

Table of contents

Introduction **4**

1 Technical work **5**

1.1 Goal 5

1.2 Implementation 5

2 Evaluation **6**

2.1 Performance 6

2.2 Validation 6

2.3 Limitations 6

Conclusion **7**

Références **8**

Introduction

"Every program is guilty until proven innocent".

During the development phase, developers spend lots of time writing tests for code they have written. But these tests can be very so tedious and repetitive that developers sometimes botch or simply skip this very important part of software development.

But what if developers didn't have to spend hours writing tests to find bugs and have a good code coverage of their program? What if there was a magic stick that can generate a bunch of unit tests?

Since Java is one of the most used programming languages nowadays, we implemented a solution that would generate automatically Java unit tests. This solution would save time for developers, and give the time and the energy to focus on working on the business layer.

This project is an extended version of an application developed by Valentin Lefils and Quentin Marrecau [**?**] [**?**] . The first version of the application treated the same problems we are facing, but only on small examples of Java programs.

In this project, we present our tool: JUnitMe2.0. Our tool can generate automatically Java unit tests for any Java open source application. From a description of specification, our tool generate all instances that covere the data specifications, then generate the unit tests corresponding to these instances.

Chapter 1

Technical work

1.1 Goal

The goal of this study is to estimate the energetic consumption of different Garbage collection(GC)[1] mechanism in Java Virtual machine (JVM)[2].

1.2 Implementation

Since we know that objects that are not referenced will be removed from memory by the garbage collector. And in order to realise this study I have modified an open source application[3] which test garbage collection behavior so the application would generate a larg number of objects and then remove the reference pointing to these objects.
Each minute a separate thread calls System.gc() to encourage JVM to run the garbage collector. Using PowerAPI[4]: "wattmeter software based on a model of actors, en uses this model to estimate software energetic consumption"[5]

Here talk runing on different GC

Paragraph about GC explaning before Impl sub paragraph per GC (4-5)

Chapter 2

Evaluation

2.1 Performance

2.2 Validation

2.3 Limitations

Conclusion

Using Spoon Java library to analyze and trasform source code, Alloy a language and tool for relational models, Alloy Analyzer a solver that takes the constraints of a model and finds structures that satisfy them and CodeModel a Java library for code generators we have succeeded in creating a tool capable of generating Java unit tests a given Java program. Our tool can verify that there no actual error exists for an application. This tool can be extended in the future to be able to treat a bigger variety of Java programs. Today, our tool has been tested on couple of Java open source project and it is able to generate Java unit tests that can achieve up to 88% of code coverage.

Bibliography

[1] Wikipedia. Garbage collection. https://en.wikipedia.org/wiki/Garbage_collection_(computer_science).

[2] Wikipedia. Jvm: Java virtual machine. https://en.wikipedia.org/wiki/Java_virtual_machine.

[3] herongyang.com. Testing garbage collection behavior. http://www.herongyang.com/JVM/GC-Garbage-Collection-Test-Program.html.

[4] Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loïc Huertas, Romain Rouvoy, and Anita Sobe. Prspoonocess-level power estimation in vm-based systems. https://hal.inria.fr/hal-01130030/file/paper.pdf.

[5] Maxime Colmant, Romain Rouvoy, and Lionel Seinturier. Prspoonocess-level power estimation in vm-based systems. https://hal.inria.fr/hal-01171696/document.