Masaryk University
Faculty of Informatics

Optimization of JVM settings for application performance

Master's Thesis

Josef Pavelec

Brno, Spring 2017

Masaryk University
Faculty of Informatics

Optimization of JVM settings for application performance

Master's Thesis

Josef Pavelec

Brno, Spring 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Josef Pavelec

Advisor: RNDr. Andriy Stetsko, PhD.

Acknowledgement

Will be written in the end.

Abstract

Will be written in the end.

Keywords

JVM settings, Optimization

Contents

1 Introduction

Will be written in the end.

2 Java Virtual Machine

Java Virtual Machine (JVM) is an abstract computing machine. JVM can not process any program written in Java language (stored in `java` source file) but it executes only program called *bytecode* which is stored in `class` file. A Java `class` file is produced from `java` source file by Java compiler. JVM, like a real computing machine, has own instruction set for processing *bytecode*. For running compiled Java program it's necessary to have only an implementation of JVM for a given platform. Described approach offers the ability for applications to be developed in a platform-independent manner and it can be shortened by Sun Microsystems slogan: "*Write once, run anywhere*".[1]

In the context of the JVM it should distinguish three terms:

- "**specification** is a document that formally describes what is required of a JVM implementation.
- **implementation** is a computer program that meets the requirements of the JVM specification.
- **instance** is an implementation running in a process that executes a computer program compiled into Java bytecode."[2]

"To implement the Java Virtual Machine (JVM) correctly, you need only be able to read the `class` file format and correctly perform the specified operations. Implementation details that are not part of the Java Virtual Machine's specification would unnecessarily constrain the creativity of implementors. For example, the memory layout of run-time data areas, the garbage-collection algorithm used, and any internal optimization of the JVM instructions (for example, translating them into machine code) are left to the discretion of the implementor." [1]

There are many implementations of JVM[1]. This chapter of thesis deals with the Java HotSpot Virtual Machine respective to Java Development Kit 8 (JDK 8) which is primary reference JVM implementation. This version is latest because JDK 9 release is scheduled for July 2017[2].

---

1. Extensive list of JVM implementation is accessible on `https://en.wikipedia.org/wiki/List_of_Java_virtual_machines`
2. `https://blogs.oracle.com/java/proposed-schedule-change-for-java-9`

2.1 HotSpot

The Java 8 HotSpot Virtual Machine implementation is maintained by Oracle corporation. It implements JVM 8 specification and trying to achieved best results for executing *bytecode* in areas such as automatic memory management or compilation to native code.

First version of JVM was interpreted. Since statement "*Java is slow*" endures notwithstanding it's not true in these days. There will be described why it's not true in next sections.

As mentioned earlier, main purpose of JVM is executing *bytecode* on specific platform which means translating *bytecode* to native code of CPU. Because interpreting of *bytecode* had appeared like inefficient there was introduced approach called *Just-In-Time* compilation (JIT) in Java 1.2 [3]. A JVM implementation with JIT compiler translates program to native code during running time.

2.1.1 GC

2.1.2 C1/C2

2.2 Setting options

[4]

- Standard options
- Nonstandard options

JVM je zasobnikovy pocitac Ergonomics - Automatic Selections and behavior tuning

Java Flight Recorder Moznosti analyzy vykonu - vmstat jconsole, prepinace -XX:PrintGCDetails/-XX:PrintGCTimeStamps

Bibliography

[1] T. Lindholm, F. Yellin, G. Bracha, and A. Buckley, *The Java® Virtual Machine Specification Java SE 8 Edition*, Oracle, 2 2015.

[2] A. Mehta, A. Saxena, and T. Bhawsar, "A brief study on jvm," *Asian Journal of Computer Science Engineering*, 2016.

[3] "Performance of Java versus C++," http://scribblethink.org/Computer/javaCbenchmark.html, [Online; visited 08-February-2017].

[4] "Java command documentation," http://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html, [Online; visited 7-December-2016].