Java Mission Control Evaluation

August 2018

AUTHOR:
Scott Hurley

University of Strathclyde

SUPERVISOR(S):
Luis Rodríguez Fernández

CERN openlab

PROJECT SPECIFICATION

Test and evaluate the monitoring and diagnostic tools of the JDK, focusing on advanced features like Java Mission Control and Java Flight Recording. Design a solution for integrating these tools with CERN Java applications. This implementation has to be pluggable in Virtual Machines, Kubernetes and Docker containers. Produce documentation on how to install, setup, configure and use the tool-set in different environments and a sample application implementing the common performance issues in Java applications.

ABSTRACT

This reports summarises the project I worked on during my internship with the IT-DB-IMS team. This report will detail my efforts to configure various technologies to work with Java Mission Control, the Java application I wrote to demonstrate common Java performance issues, the documentation I produced and my evaluation of the tools platform itself.

TABLE OF CONTENTS

1. Acknowledgements

I would like to thank my supervisor, Luis Rodríguez Fernández, for his invaluable support and the IT-DB-IMS team for their help and making me feel welcome at CERN.

2. Introduction

The objective of this project is to evaluate the viability of Java Mission Control to monitor CERN's Java applications on its production servers and document how to configure various technologies to work with Java Mission Control.

3. Java Performance App

In order to test Java Mission Control's effectiveness in diagnosing the common issues Java apps can suffer from, I wrote a program to demonstrate them. For it to be demonstrative of a real Java app deployed at CERN, it utilises Spring, MySQL and Maven and is deployed using tomcat. The app demonstrates the following performance issues:

- deadlock
- gridlock
- excessive garbage collection
- misconfigured thread pool
- memory leaks

To demonstrate deadlock, the application creates a scenario: two friends, Alphonse and Gaston bow to each other and they will only stop bowing when the other stops. As they are both waiting for the other to stop bowing, they are stuck.

To demonstrate gridlock, which is analogous to a traffic jam, the app creates a bank account and a number of threads attempting to change the owner and deposit money. One would not expect a thread changing the name of the account holder to have to wait for a thread depositing money as that would not raise any concurrency issues but in this example it does, resulting in unnecessary waiting.

The app instigates a memory leak by creating a static arraylist of strings and continually adding items to it. The Garbage collector can't clean up the array as it is static. This example was sourced from stacify[1].

Excessive garbage collection is demonstrated by creating many soft references to objects which the GC must take time to delete. This example was sourced from plumbr[2].

Finally, a misconfigured thread pool that is too small is demonstrated by creating just one thread executor and giving it many queries to execute on a database. A thread pool that is too big can also be created by passing a large number as a parameter for the number of thread executors. Please refer to the application documentation for details of passing parameters.

4. Documentation

There is an article explaining how to install and use Java Mission Control and how to navigate its interface[3]. It also explains how to use Java Flight Recorder to diagnose common performance issues. Documentation for the Java performance application and Docker project can be found on their respective git repos[4][5]. Their documentation details how to run them and gives an explanation for why certain arguments are used.

5. About Java Mission Control

Java Mission Control (JMC)[6] is a JDK profiling and diagnostics tools platform for the JVM. It is a tool suite for basic monitoring, managing, and production time profiling and diagnostics and is included with JDK 7 and beyond. The two plug-ins included that are of interest to CERN are Java Management Extensions (JMX) and Java Flight Recorder (JFR). Other plug-ins are available but are experimental and not suitable for production.

Java Management Extensions allows for the real-time monitoring of Java applications and their JVMs. Triggers can be created and set using JMX. Java Flight Recorder records statistics of a java app and its JVM and writes it to disk for future examination. Triggers activate actions when certain conditions are met. For example, when deadlocked threads are detected an email could be sent to the system administrator to alert them to the problem.

6. Test Environments

a. Tomcat

Apache Tomcat is an open-source Java Servlet Container developed by the Apache Software Foundation[7]. One way to configure Tomcat to get the Java performance app working is to add a setenv.sh file to the bin directory of your apache-tomcat installation. This file defines arguments that Tomcat will pass to the JVM and is available to review in the gitlab repository of the app[4]. Along with this you can copy your web application, in the form of a .war file, to the webapps folder of your Tomcat installation. There are other ways get the app running but this way is simple and a more in-depth guide for it is available in the app's readme.

b. Docker

Docker creates containers or "lightweight virtual machines" to run programs over[8]. The dockerfile must copy the necessary files into the container and run the script that injects the user-specified environment variables into the relevant files. When running the image, the user specifies which ports from the container are exposed to which ports on the client and what the login details for the JVM are. Documentation for docker configuration can be found in the docker image repository[5].

7. Conclusion - Java Mission Control Evaluation

Java Mission Control is a very powerful monitoring tool. Its user interface is simple and intuitive, it provides detailed and useful statistics for easy diagnosis of problems and has very little impact on the performance of the monitored application. It is very good for live monitoring and the ability to remotely connect to JVMs and customise flight recordings are also highly useful features. Triggers were of particular interest to CERN due to their potential to pre-emptively alert system admins to potential problems and produce recordings from crashed JVMs for post-mortem diagnosis.

However, JMC is over-reliant on its UI and cannot be used via the command line. Without this functionality triggers cannot be automated by configuration tools such as Puppet[9], making them impractical in large scale use cases like CERN's. Oracle currently has no plans to change this. Java Flight Recorder can however be used through the command line. It is difficult to save flight recordings when the JVM crashes and when the monitored JVM becomes unresponsive, so does JMC. Further investigation is required to determine whether JFR alone is worth the license fee or if there is a better alternative.

8. Future Work

Further evaluation of JMC with regards to Oracle WebLogic and Kubernetes is required.In addition, JMC's ability to monitor Java applications running on Weblogic also needs to be tested[10].

9. References

1. https://stackify.com/memory-leaks-java/
2. https://plumbr.io/blog/garbage-collection/weak-soft-and-phantom-references-impact-on-gc
3. https://twiki.cern.ch/twiki/bin/viewauth/DB/Private/JavaMissionControl
4. https://gitlab.cern.ch/db/java_perf_issues
5. https://gitlab.cern.ch/shurley/java-performance-issues-app-docker.git
6. https://docs.oracle.com/javacomponents/jmc-5-5/jmc-user-guide/jmc.htm#JMCCI111
7. http://tomcat.apache.org/
8. https://www.docker.com/
9. https://puppet.com/
10. https://cern.service-now.com/service-portal/sls_grid.do