



**FACULTY
OF MATHEMATICS
AND PHYSICS**
Charles University

MASTER THESIS

Jakub Háva

**Monitoring Tool for Distributed Java
Applications**

Department of Distributed and Dependable Systems

Supervisor of the master thesis: Mgr. Pavel Parízek, Ph.D.

Study programme: Computer Science

Study branch: Software Systems

Prague 2017

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In date

signature of the author

Title: Monitoring Tool for Distributed Java Applications

Author: Jakub Háva

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Parízek, Ph.D., Department of Distributed and Dependable Systems

Abstract: TODO: Abstract

Keywords: monitoring cluster instrumentation

I would like to thank my thesis supervisor Mgr. Pavel Parizek, Phd. for leading me throughout the whole thesis and willing to help with any questions I've ever had. I would also like to thank H2O.ai for being able to write this thesis under their coordination as well, particularly Michal Malohlava for providing very useful technical and theoretical advice

Contents

1	Introduction	4
1.1	Project Goals	4
2	Analysis	5
2.1	Background and Tools	5
2.1.1	Monitoring Tools	5
2.1.2	Profiling Tools	5
2.2	Related Work	5
2.2.1	Google Dapper	5
2.2.2	Zipkin	5
2.3	Instrumentation libraries	5
2.3.1	Javassist	5
2.3.2	ByteBuddy	5
2.3.3	CGlib	5
2.3.4	ASM	5
2.4	Communication Middleware	5
2.4.1	ZeroMQ	5
2.4.2	NanoMSG	5
2.5	Comparison of Agent Approaches	5
2.5.1	Java Agent Solution	5
2.5.2	Native Agent Solution	5
3	Related Technologies	6
3.1	Java	6
3.1.1	Class Initialization Process	6
3.1.2	JVMTI	6
3.1.3	JNI	6
3.1.4	ClassLoaders	6
3.2	ByteBuddy	6
3.2.1	Main Concept	6
3.2.2	Transformers	6
3.2.3	Intereptors	6
3.2.4	Class File Locator	6
3.2.5	Advice API	6
3.3	NanoMgs	6
3.3.1	C++11 Mapping	6
3.3.2	Java Mapping	6
3.4	spdlog	6
4	Overview	7
4.1	Architecture Description	7
4.2	Communication	7

5	Design/Architecture	8
5.1	Native Agent	8
5.1.1	Structure Overview	8
5.1.2	Instrumentation API	8
5.1.3	Byte Class Parsing	8
5.1.4	Instrumentation	8
5.1.5	Native Agent Arguments	8
5.2	Instrumentation Server	8
5.2.1	Instrumentation Handling	8
5.2.2	Communication modes	8
5.2.3	Class Caching	8
5.3	Instrumentation Library	8
5.3.1	Custom Service Loader	8
5.3.2	Public interfaces	8
5.3.3	Extending the Library	8
5.3.4	ClassLoaders	8
5.3.5	JSON Generation	8
5.4	User Interface	8
5.4.1	Zipkin Overview	8
5.4.2	Zipkin Data Model	8
5.4.3	Zipkin JSON Format	8
5.5	Collectors	8
5.6	Docker Support	9
6	Implementation Details	10
6.1	Native Agent	10
6.2	Instrumentation Library	10
7	Evaluation	11
7.1	Deployment Strategies	11
7.1.1	Instrumentor on the same node with the Application . . .	11
7.1.2	Instrumentor available over the Network	11
7.1.3	Bundling the application classes with the Instrumentor . .	11
7.2	Platform demonstration	11
7.2.1	Bulding Monitoring tool on top of Distrace	11
7.2.2	Basic Demonstration	11
7.2.3	Optimizing the instrumentation	11
8	Conclusion	12
8.1	Comparison to Related Work	12
8.2	Future plans	12
8.2.1	Integration with well-known data collectors	12
8.2.2	Add support for Flame charts	12
	Conclusion	13
	List of Figures	14
	List of Tables	15

List of Abbreviations	16
Attachments	17

1. Introduction

1.1 Project Goals

2. Analysis

2.1 Background and Tools

2.1.1 Monitoring Tools

2.1.2 Profiling Tools

System Profilers

Application Specific Profilers

2.2 Related Work

2.2.1 Google Dapper

2.2.2 Zipkin

2.3 Instrumentation libraries

2.3.1 Javassist

2.3.2 ByteBuddy

2.3.3 CGlib

2.3.4 ASM

.. just give brief overview what were the instrumentation libraries choices. The selected one will be described in the next section

2.4 Communication Middleware

2.4.1 ZeroMQ

2.4.2 NanoMSG

2.5 Comparison of Agent Approaches

2.5.1 Java Agent Solution

2.5.2 Native Agent Solution

3. Related Technologies

3.1 Java

3.1.1 Class Initialization Process

3.1.2 JVMTI

3.1.3 JNI

3.1.4 ClassLoaders

3.2 ByteBuddy

3.2.1 Main Concept

3.2.2 Transformers

3.2.3 Intereptors

3.2.4 Class File Locator

3.2.5 Advice API

3.3 NanoMgs

3.3.1 C++11 Mapping

3.3.2 Java Mapping

3.4 spdlog

logging library used

4. Overview

4.1 Architecture Description

4.2 Communication

5. Design/Architecture

5.1 Native Agent

5.1.1 Structure Overview

5.1.2 Instrumentation API

5.1.3 Byte Class Parsing

5.1.4 Instrumentation

5.1.5 Native Agent Arguments

5.2 Instrumentation Server

5.2.1 Instrumentation Handling

5.2.2 Communication modes

5.2.3 Class Caching

5.3 Instrumentation Library

5.3.1 Custom Service Loader

5.3.2 Public interfaces

5.3.3 Extending the Library

.. instrumentation server can run on the same node or over the network. Instrumentation server can have client code attached or not.

5.3.4 ClassLoaders

5.3.5 JSON Generation

5.4 User Interface

5.4.1 Zipkin Overview

5.4.2 Zipkin Data Model

5.4.3 Zipkin JSON Format

5.5 Collectors

Should I mention the collectors ? It may be sufficient to have send data right to zipkin for demonstration purposes

5.6 Docker Support

6. Implementation Details

Mention interesting parts of the implementation

6.1 Native Agent

6.2 Instrumentation Library

7. Evaluation

7.1 Deployment Strategies

- 7.1.1 Instrumentor on the same node with the Application
- 7.1.2 Instrumentor available over the Network
- 7.1.3 Bundling the application classes with the Instrumentor

7.2 Platform demonstration

- 7.2.1 Building Monitoring tool on top of Distrace
- 7.2.2 Basic Demonstration
- 7.2.3 Optimizing the instrumentation

8. Conclusion

8.1 Comparison to Related Work

8.2 Future plans

8.2.1 Integration with well-known data collectors

8.2.2 Add support for Flame charts

An example citation: ?

Conclusion

List of Figures

List of Tables

List of Abbreviations

Attachments