Alibaba - Scale Computing with Java

Sanhong Li
Denghui Dong
Alibaba Cloud, Java Team
Agenda

• **Introduction:** Java at Alibaba

• **Challenges&Solution:** Approaches to Improve the Engineering Productivity

• **Practice:** Tooling Support for Efficient Upgrade and Diagnostics
  • Eclipse Migration Toolkit for Java (EMT4J)
  • Eclipse Jifa
01 Java at Alibaba
for extreme scaling on Alibaba cloud
Alibaba Lives on the JVM

Java Apps

JVM – where Java performance meets hardware

Platform

(downstream of OpenJDK)
The Evolution of Java at Alibaba

- **2004**: Switch from PHP to Java
- **2012**: Join Apache Foundation
- **2015**: Sign OCA and Participant in OpenJDK
- **2018**: Become a Member of JCP EC
- **2019**: Make Alibaba Dragonwell Open Source
- **2021**: Join Eclipse Foundation
  - Join in Adoptium WG
  - Open-source Jifa&EMT4J projects

Java is becoming the driving force behind Alibaba e-commerce platforms: Alibaba.com, Taobao, TMall, etc.
Alibaba Java and Open Source

Framework/ Middleware

- Usage in Alibaba (order of magnitude)
  - 10,000 developers
  - 100,000 applications
  - 1,000,000 JVM instances

OLAP/OLTP/Big Data

- Building most of Java software based on the rich open-source ecosystem
- Building 1st class support for Java on Alibaba Cloud
02 Engineering Challenges in the development lifecycle
Problems and Challenges faced in each stage of software development

- How we can improve the code quality in development phase? (Shift left practice)
- What we can do to speed up the performance of automation build tools?
- How we can support the JDK upgrade for large-scale applications?

- What we can do to improve the efficiency of problem diagnostic?
- What is my Java application doing? How we can improve the production performance by the guide of characterization of workloads?
Approaches to Improve the Productivity

- **Coding Quality**
  - Alibaba Java Coding Guidelines
  - P3C
- **Automation Build Efficiency**
  - The Standalone Pointsto Analysis Tool
- **JDK Upgrade**
  - EMT4J
- **Problem Diagnostic**
  - Eclipse Jifa
- **Production Performance**
  - Alibaba Dragonwell

Requirements:

- Best programming practices for Java developers
- Static code analyzer/IntelliJ IDEA/Eclipse plugin
- Advanced static code analyzer tool, based on GraalVM
- ✓ Build-in profiling capacities
- ✓ Optimized OpenJDK for cloud workloads

Solution:

JCP EC 2023.4
03 Tooling Support

for boosting the engineering productivity
Challenges in Traditional Upgrade Method

• **Document Guide:** Requiring developers to resolve the problems manually by experience

• **Bad Scalability:** Upgrade efforts(almost same) repeated from team to team, experiences are not accumulated as sharable tooling infra

• **Uncontrollable Upgrade Cost:**
  - Much more incompatibility issues introduced by upgrading from 8 to 11/17(compared with upgrading Java 7 -> 8), hurting the stability of online application when they are not handling properly(especially many corner cases)
  - Uncontrollable increased cost when the application is relying on many libraries(dependencies)
Eclipse Migration Toolkit for Java (EMT4J)

Help your projects succeed in the long term

- Open sourced to the Eclipse community by Alibaba in 2022
- Incubated as an Eclipse Adoptium sub-project
- A toolkit to make JDK migration easy

support upgrading to LTS versions

https://projects.eclipse.org/projects/adoptium.emt4j
https://github.com/adoptium/emt4j
Design Principles of EMT4J

1. Upgrading issues are mapped as configurable rules

2. Rule-based analysis
   - static via Maven plugin
   - dynamic via Java agent
EMT4J based Workflow (8 -> 11)

1. As a Maven plugin, EMT4J has been integrated as part of CICD infra, and finds the upgrade problems via static code analysis
   - Common libraries (dependencies) have been classified/tagged properly for Java11

2. Fix the problems
   - modify app code
   - modify JVM options
   - upgrade dependencies
   - ...

3. Run with JDK11 at staging environment
   - Dynamic analysis
   - Functional/Perf verification

4. Run with JDK11 in production
   - Monitoring, Health check
Report Example (Java 8 -> 11)

Organized by modules and dependencies

Group by the compatibility problem
Eclipse Jifa Project

Java Issue Finder Assistant (Jifa)

A web application for online troubleshooting
Challenge: Heap Dump Analysis in Cloud

- Requiring heap dump file transfer from cloud to local
- Requiring large memory for large heap analysis
Eclipse Jifa

- Open-sourced by Alibaba under Eclipse Foundation in March 2020
- Eclipse Public License 2.0
- Github: https://github.com/eclipse/jifa

https://projects.eclipse.org/proposals/eclipse-jifa
Architecture Overview of Jifa

• Web application, designed for troubleshooting Java application in the cloud

• Analytic Engine
  ▪ Heap Dump Analyzer
    • implemented based on Eclipse MAT
  ▪ GC Log Analyzer
  ▪ Thread Dump Analyzer
Report Example (Heap Analysis)

Report Details per View

**Different GC Roots**

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Objects</th>
<th>Shallow Heap</th>
<th>Retained Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69926</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2819</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>152</td>
<td>344</td>
<td></td>
</tr>
</tbody>
</table>

**The chain of objects which keep live and traced from GC Root (Thread)**

Follow the same guide from Eclipse MAT

**Different Views**

- Overview
- Leak Suspects
- GC Roots
- Dominator Tree
- Histogram
- Unreachable Objects
- Duplicated Classes
- Class Loaders

**Report Example**


Follow the same guide from Eclipse MAT
GC Log Analyzer Introduction

- GC algorithm support
  - Serial GC / Parallel GC / CMS / G1 / ZGC
- Java version support: 8/11/17
- Feature list
  - Diagnostic recommendation
  - Interactive graphs
  - Key performance indicators
  - GC pause statistics
Thread Dump Analyzer Introduction

- Visualizing the output of jstack
- Java version support: 8 / 11 / 17
- Feature list
  - Thread group summary
  - Lock info
    - deadlock analysis
  - Call Site Tree