Alibaba -Scale Computing with Java

Sanhong Li
Denghui Dong
Alibaba Cloud, Java Team

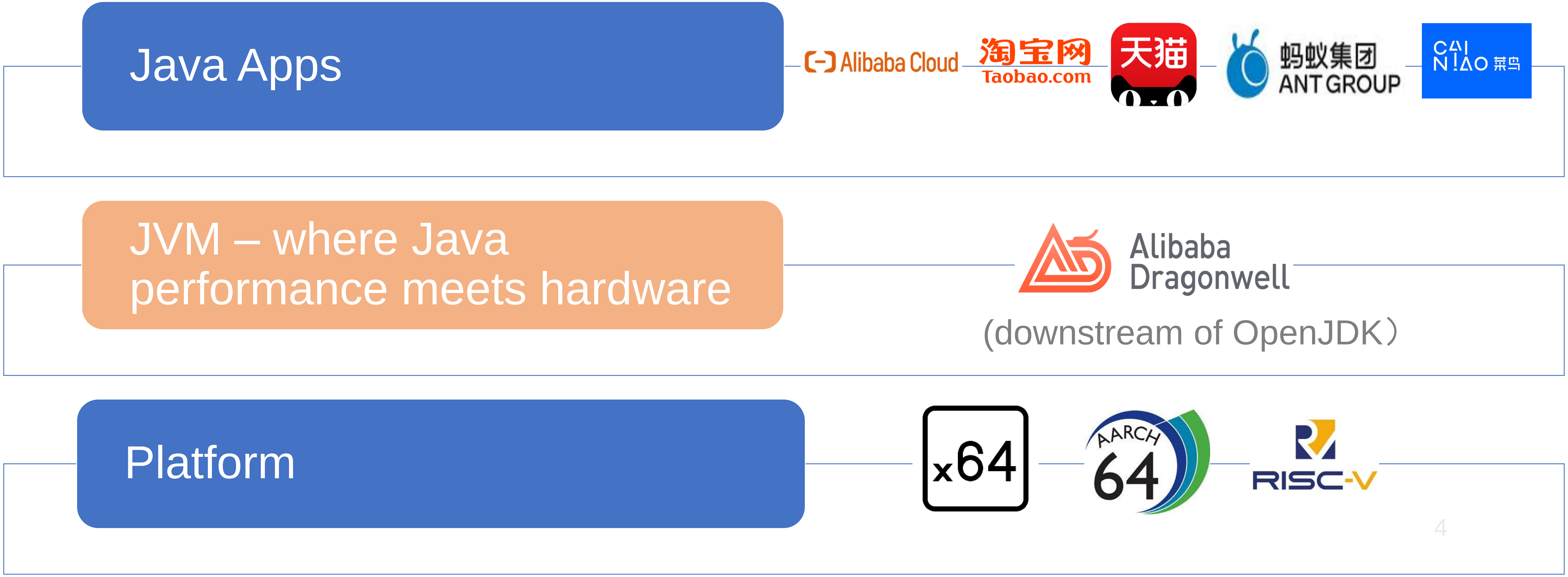
Agenda

- **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

Java is becoming the driving force behind Alibaba e-commernce platforms:Alibaba.com Taobao, TMall , etc.

2012

Sign OCA and Participant in OpenJDK

2015

Join Apache Foundation

Biggest contributor to Apache Flink, Dubbo, RocketMQ

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

Alibaba Java and Open Source

Framework/ Middleware

OLAP/OLTP/Big Data

Java

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

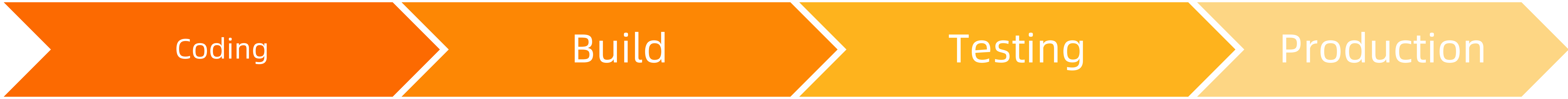
- Building most of Java software based on the rich open-source ecosystem
- Building 1st class support for Java on Alibaba Cloud

02 Engineering Challenges

in the development lifecycle

Problems and Challenges

faced in each stage of software development



Coding

Build

Testing

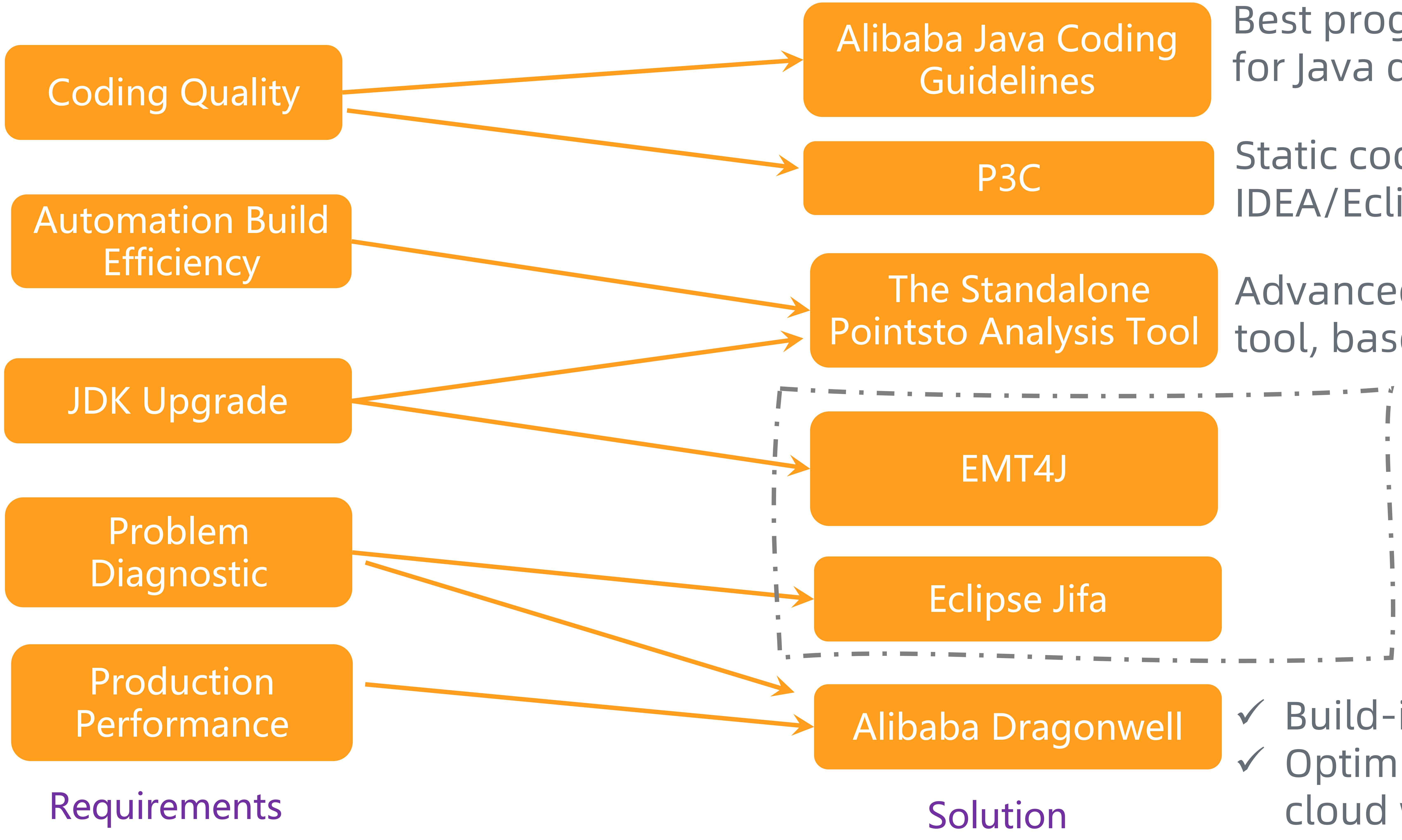
Production

- How we can improve the code quality in development phase?(Shift left practice)
- What we can do to speedup 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?

Continuous Integration/Delivery/Deployment

Approaches to Improve the Productivity



Coding Quality

Automation Build Efficiency

JDK Upgrade

Problem Diagnostic

Production Performance

Requirements

Alibaba Java Coding Guidelines

P3C

The Standalone Pointsto Analysis Tool

EMT4J

Eclipse Jifa

Alibaba Dragonwell

Solution

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

03 Tooling Support

for boosting the engineering productivity

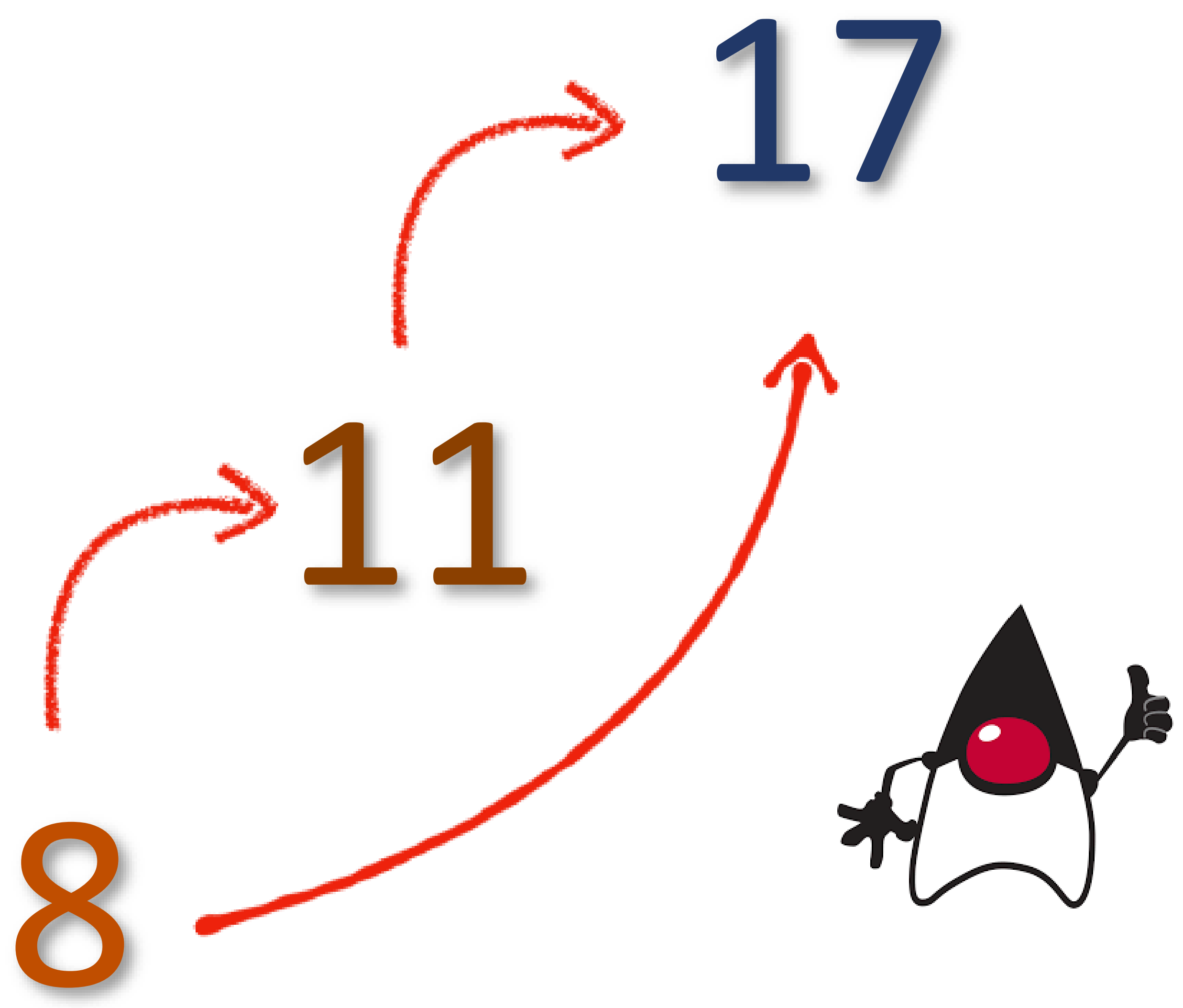
Challenges in Traditional Upgrade Method

- **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



8

11

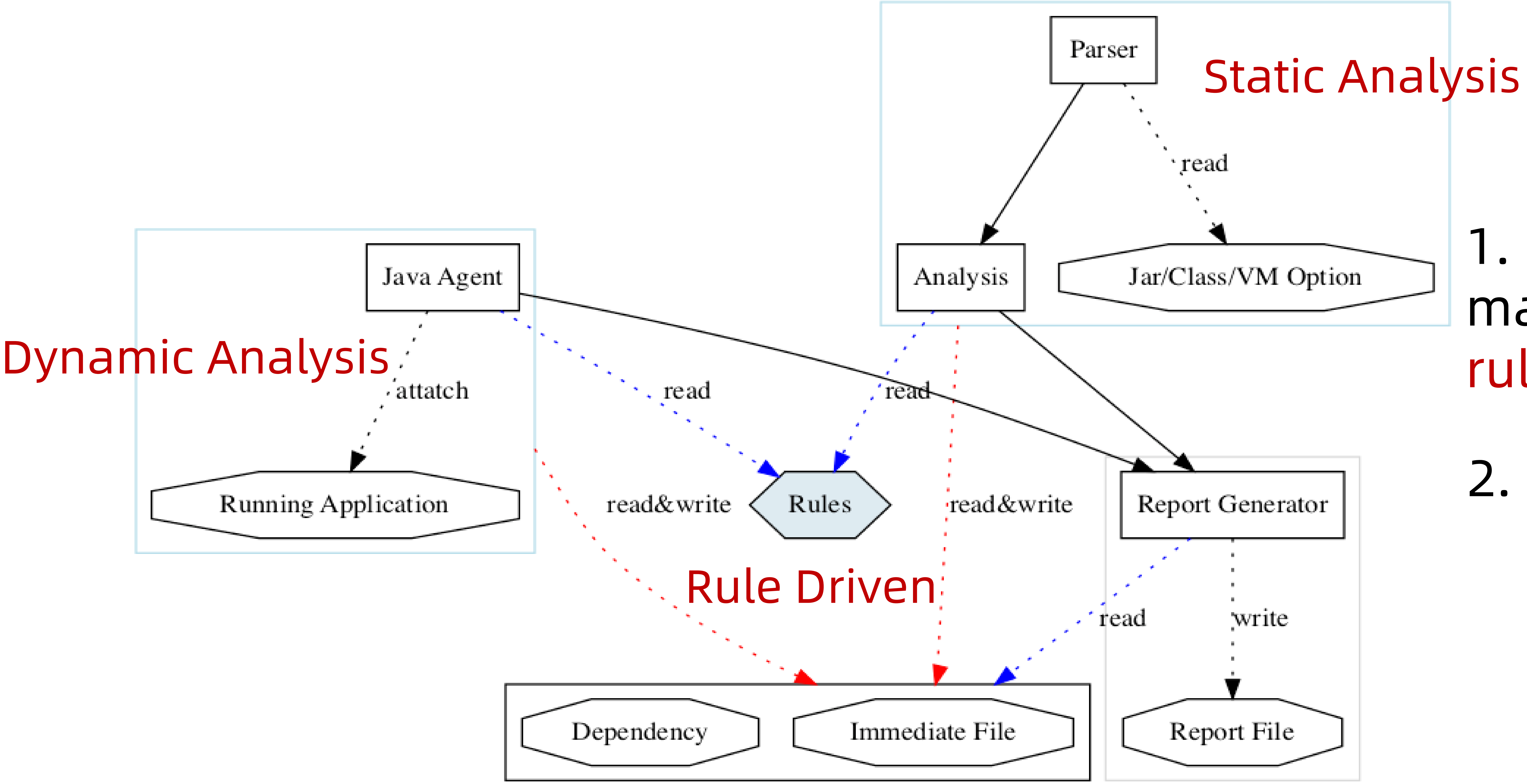
17

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

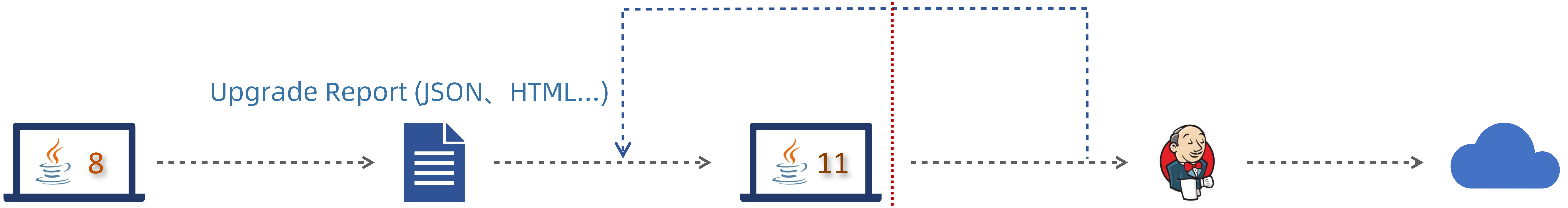
2. Rule-based analysis

- static via Maven plugin
- dynamic via Java agent

EMT4J based Workflow(8 -> 11)

Development

Deployment



Go back to modify if more issues found

Upgrade Report (JSON、HTML...)

1. As a Maven plugin, EMT4J has been integrated as part of CICD infra, and finds the upgrade problems via static code analysis

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

- Common libraries(dependencies) have been classified/tagged properly for Java11

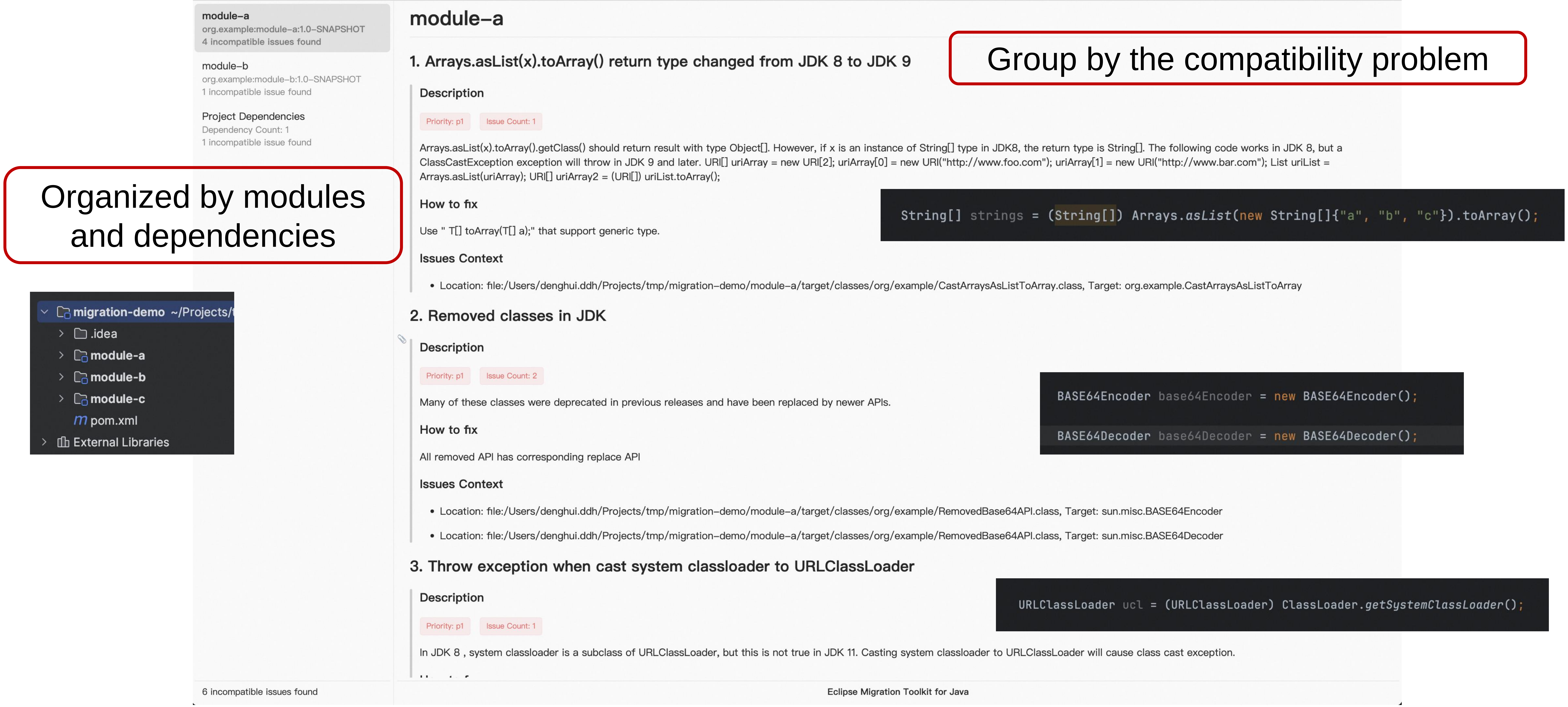
3. Run with JDK11 at staging environment
   - Dynamic analysis
   - Functional/Perf verification

4. Run with JDK11 in production
   - Monitoring, Health check

Report Example(Java 8 -> 11)

Group by the compatibility problem

Organized by modules and dependencies



module–a
org.example:module–a:1.0–SNAPSHOT
4 incompatible issues found

module–b
org.example:module–b:1.0–SNAPSHOT
1 incompatible issue found

Project Dependencies
Dependency Count: 1
1 incompatible issue found

6 incompatible issues found

module–a

1. Arrays.asList(x).toArray() return type changed from JDK 8 to JDK 9

Description

Priority: p1 Issue Count: 1

Arrays.asList(x).toArray().getClass() should return result with type Object[]. However, if x is an instance of String[] type in JDK8, the return type is String[]. The following code works in JDK 8, but a ClassCastException exception will throw in JDK 9 and later. URI[] uriArray = new URI[2]; uriArray[0] = new URI("http://www.foo.com"); uriArray[1] = new URI("http://www.bar.com"); List uriList = Arrays.asList(uriArray); URI[] uriArray2 = (URI[]) uriList.toArray();

How to fix

Use " T[] toArray(T[] a);" that support generic type.

```
String[] strings = (String[]) Arrays.asList(new String[]{"a", "b", "c"}).toArray();
```

Issues Context

- Location: file:/Users/denghui.ddh/Projects/tmp/migration–demo/module–a/target/classes/org/example/CastArraysAsListToArray.class, Target: org.example.CastArraysAsListToArray

2. Removed classes in JDK

Description

Priority: p1 Issue Count: 2

Many of these classes were deprecated in previous releases and have been replaced by newer APIs.

```
BASE64Encoder base64Encoder = new BASE64Encoder();
BASE64Decoder base64Decoder = new BASE64Decoder();
```

How to fix

All removed API has corresponding replace API

Issues Context

- Location: file:/Users/denghui.ddh/Projects/tmp/migration–demo/module–a/target/classes/org/example/RemovedBase64API.class, Target: sun.misc.BASE64Encoder
- Location: file:/Users/denghui.ddh/Projects/tmp/migration–demo/module–a/target/classes/org/example/RemovedBase64API.class, Target: sun.misc.BASE64Decoder

3. Throw exception when cast system classloader to URLClassLoader

Description

Priority: p1 Issue Count: 1

```
URLClassLoader ucl = (URLClassLoader) ClassLoader.getSystemClassLoader();
```

In JDK 8 , system classloader is a subclass of URLClassLoader, but this is not true in JDK 11. Casting system classloader to URLClassLoader will cause class cast exception.

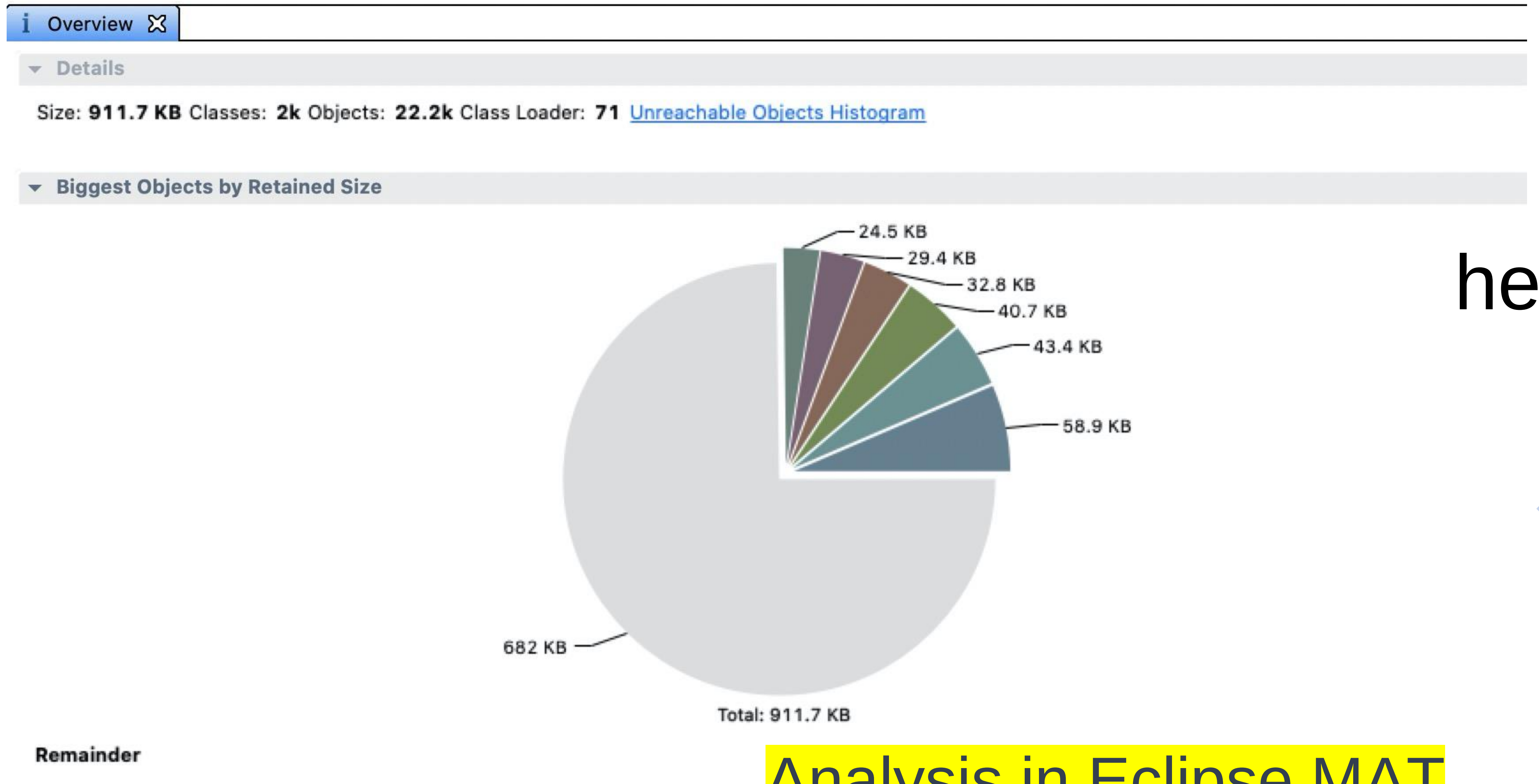
Eclipse Migration Toolkit for Java

Eclipse Jifa Project

Java Issue Finder Assistant(Jifa)

A web application for online troubleshooting

Challenge: Heap Dump Analysis in Cloud



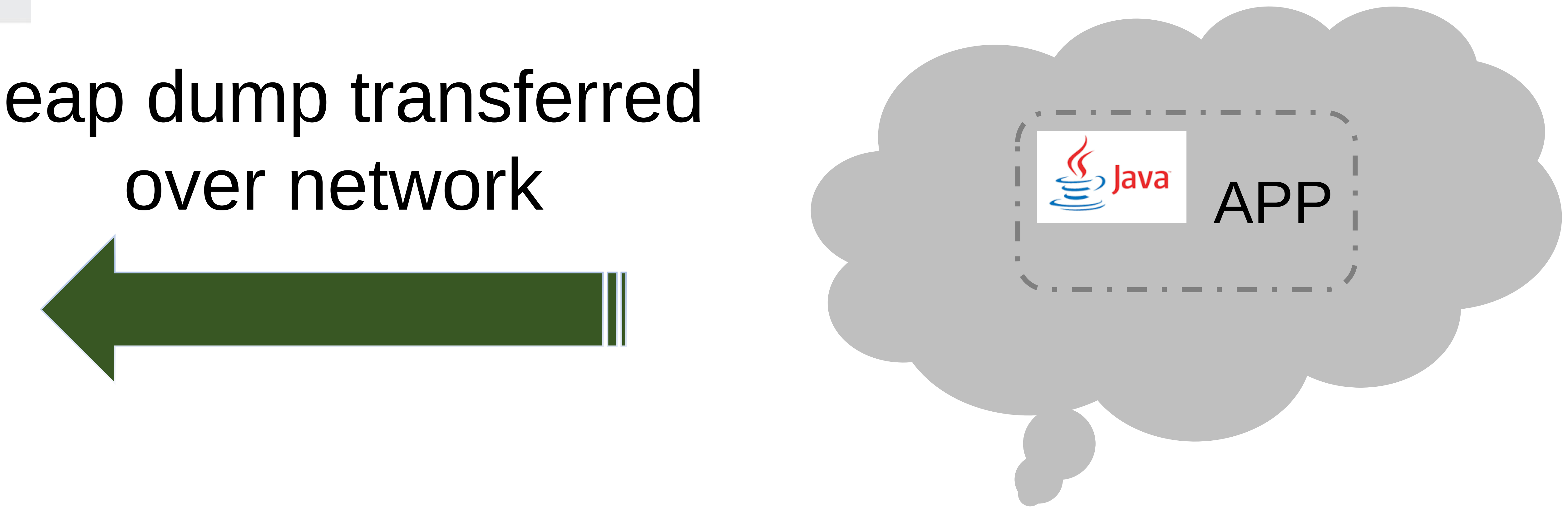
Overview

Details

Size: **911.7 KB** Classes: **2k** Objects: **22.2k** Class Loader: **71** Unreachable Objects Histogram

Biggest Objects by Retained Size

Analysis in Eclipse MAT



heap dump transferred over network

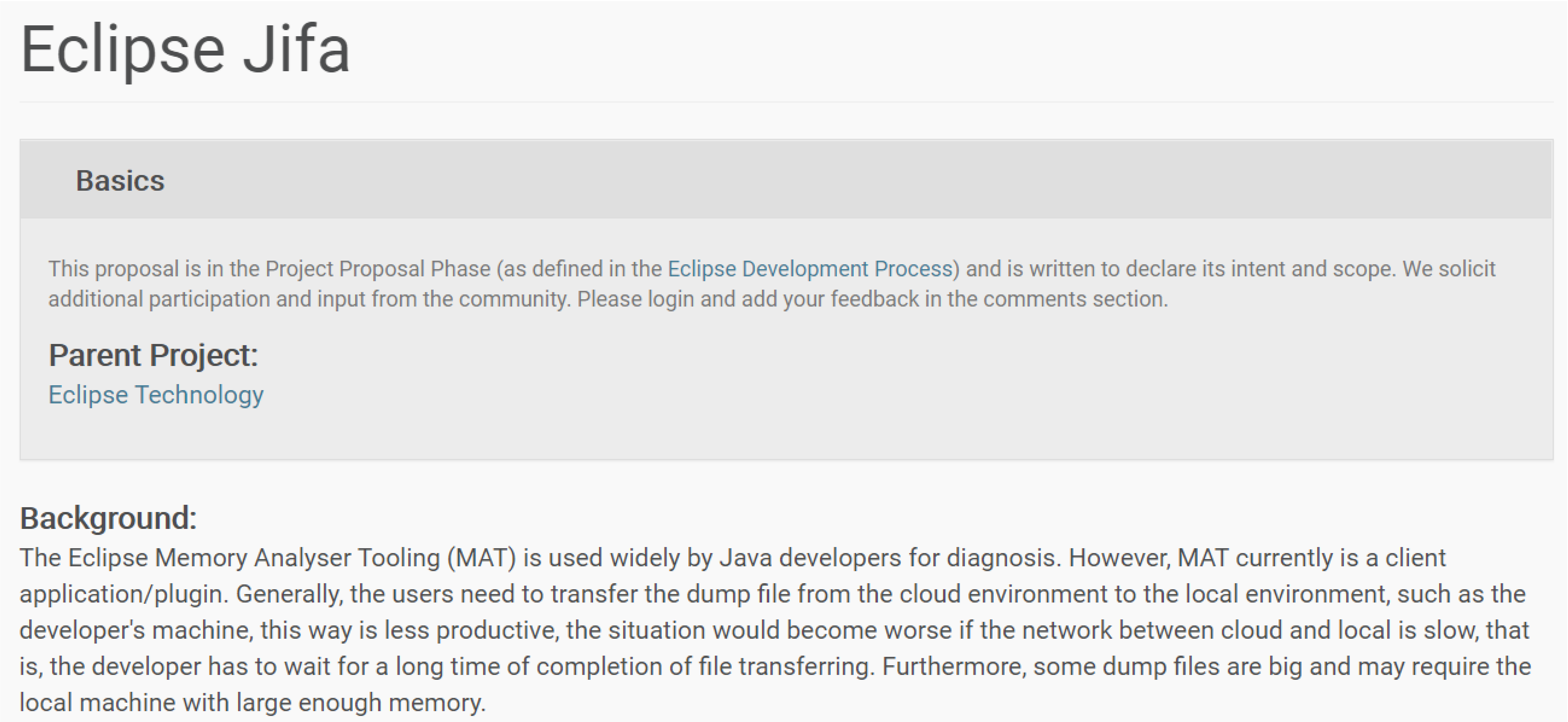
‘Cloud’

‘Local’

- 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



Eclipse Jifa

Basics

This proposal is in the Project Proposal Phase (as defined in the Eclipse Development Process) and is written to declare its intent and scope. We solicit additional participation and input from the community. Please login and add your feedback in the comments section.

Parent Project:
Eclipse Technology

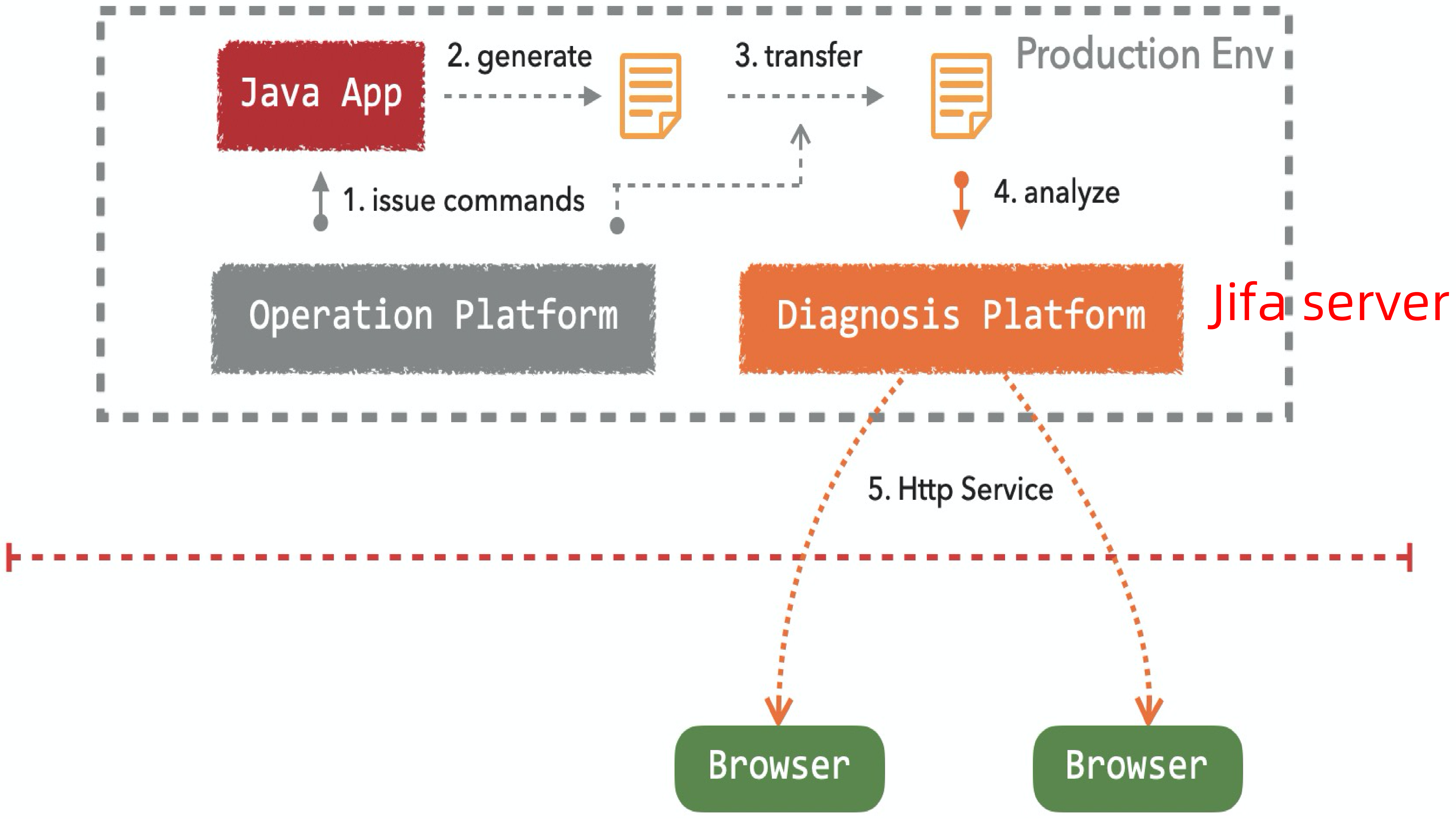
Background:
The Eclipse Memory Analyser Tooling (MAT) is used widely by Java developers for diagnosis. However, MAT currently is a client application/plugin. Generally, the users need to transfer the dump file from the cloud environment to the local environment, such as the developer's machine, this way is less productive, the situation would become worse if the network between cloud and local is slow, that is, the developer has to wait for a long time of completion of file transferring. Furthermore, some dump files are big and may require the local machine with large enough memory.

https://projects.eclipse.org/proposals/eclipse-jifa

Community Involvement NETFLIX Alibaba Cloud

Architecture Overview of Jifa

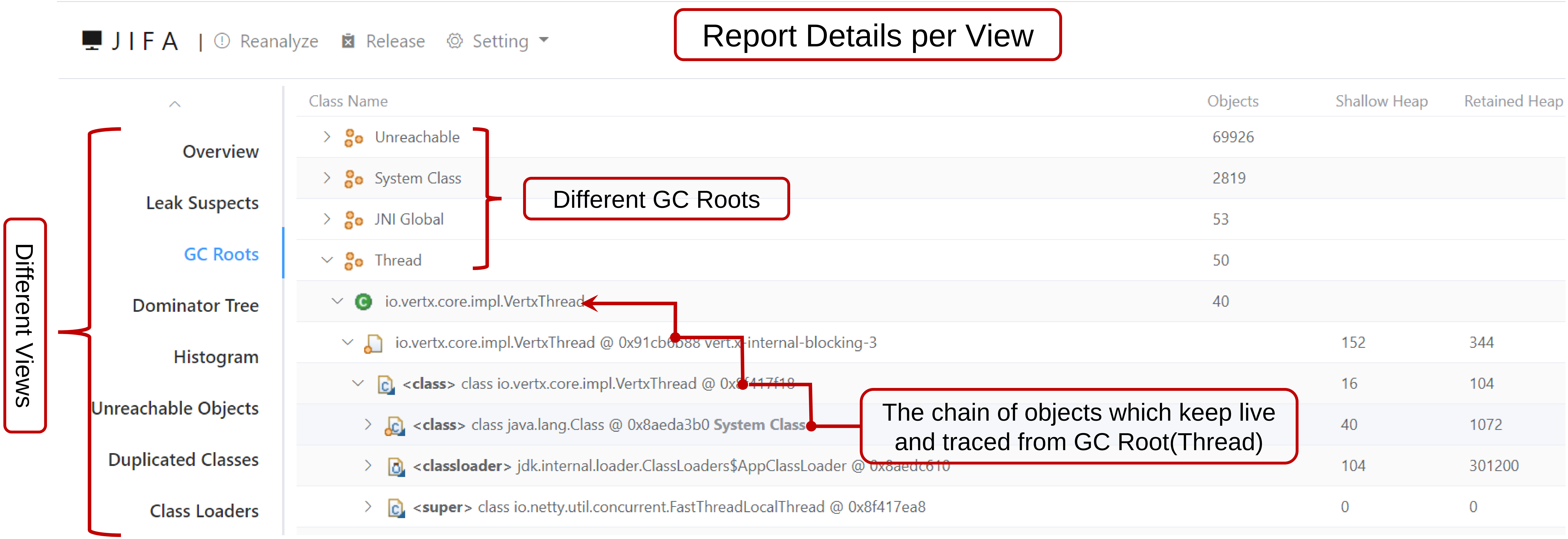
- Web application, designed for troubleshooting Java application in the cloud
- Analytic Engine
  - Heap Dump Analyzer
    - implemented based on Eclipse MAT
  - GC Log Analyzer
  - Thread Dump Analyzer



Jifa server

Jifa client

Report Example(Heap Analysis)



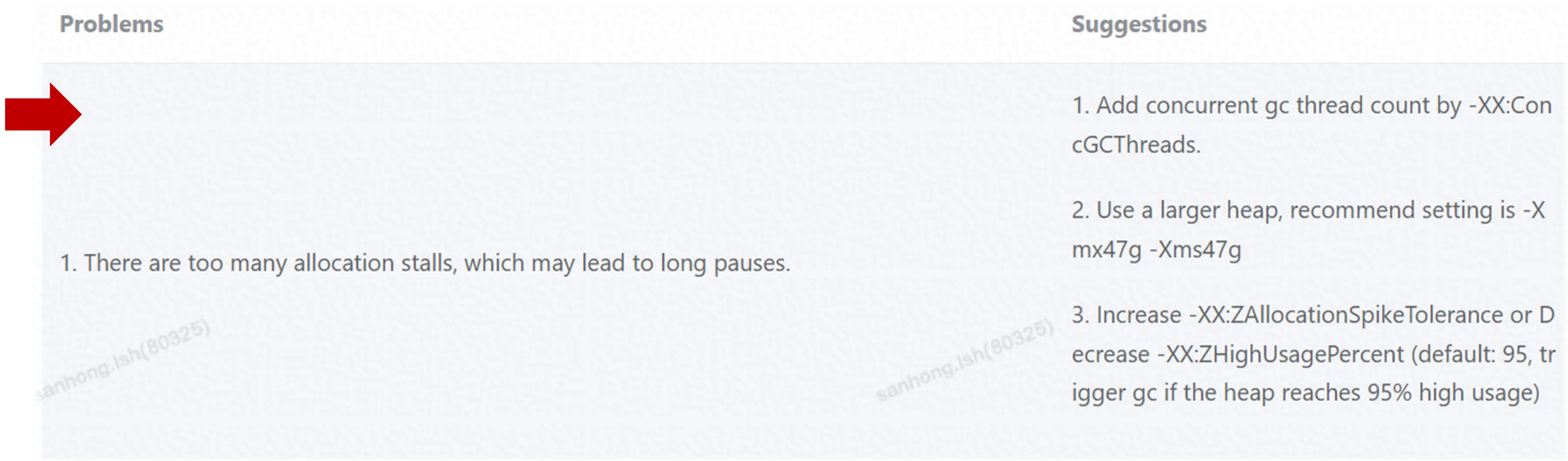
Class Name	Objects	Shallow Heap	Retained Heap
Unreachable	69926		
System Class	2819		
JNI Global	53		
Thread	50		
io.vertx.core.impl.VertxThread	40		
io.vertx.core.impl.VertxThread @ 0x91cb6b88 vert.x-internal-blocking-3		152	344
<class> class io.vertx.core.impl.VertxThread @ 0x8f417f18		16	104
<class> class java.lang.Class @ 0x8aeda3b0 **System Class**		40	1072
<classloader> jdk.internal.loader.ClassLoaders$AppClassLoader @ 0x8aedc610		104	301200
<super> class io.netty.util.concurrent.FastThreadLocalThread @ 0x8f417ea8		0	0

http://jifa.dragonwell-jdk.io/heapDump?file=1617170189191-demo.hprof

Follow the same guide from Eclipse MAT

GC Log Analyzer Introduction

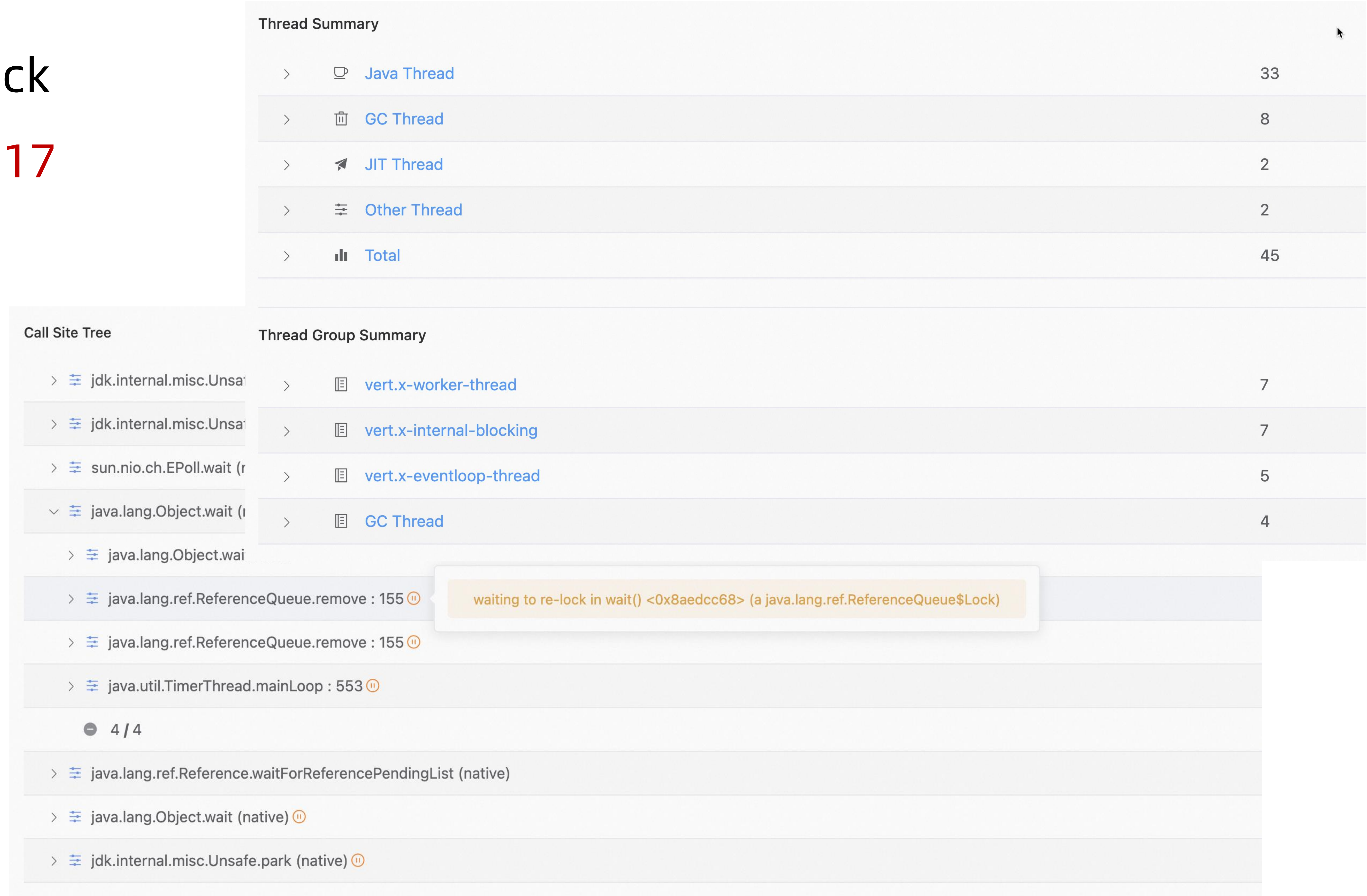
- GC algorithm support
  - Serial GC / Parallel GC / CMS / G1 / ZGC
- Java version support : 8/11/17
- Feature list
  - Diagnostic recommendation
  - Interactive graphs
  - Key performance indicators
  - GC pause statistics



Problems	Suggestions
1. There are too many allocation stalls, which may lead to long pauses.	1. Add concurrent gc thread count by -XX:ConcGCThreads. 2. Use a larger heap, recommend setting is -Xmx47g -Xms47g 3. Increase -XX:ZAllocationSpikeTolerance or Decrease -XX:ZHighUsagePercent (default: 95, trigger gc if the heap reaches 95% high usage)

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



Thread Summary

Java Thread	33
GC Thread	8
JIT Thread	2
Other Thread	2
Total	45

Thread Group Summary

vert.x-worker-thread	7
vert.x-internal-blocking	7
vert.x-eventloop-thread	5
GC Thread	4

Call Site Tree

```
> jdk.internal.misc.Unsa
> jdk.internal.misc.Unsa
> sun.nio.ch.EPoll.wait (r
v java.lang.Object.wait (r
  > java.lang.Object.wai
  > java.lang.ref.ReferenceQueue.remove : 155   waiting to re-lock in wait() <0x8aedcc68> (a java.lang.ref.ReferenceQueue$Lock)
  > java.lang.ref.ReferenceQueue.remove : 155
  > java.util.TimerThread.mainLoop : 553
    4 / 4
> java.lang.ref.Reference.waitForReferencePendingList (native)
> java.lang.Object.wait (native)
> jdk.internal.misc.Unsafe.park (native)
```

Alibaba Cloud

Worldwide Cloud Services Partner

WWW.ALIBABA CLOUD.COM