Alibaba Cloud

Worldwide Cloud Services Partner

One More Gap Bridged Towards Practice

Support Serialization Feature in Native Image

Ziyi Lin, Kuai Wei, Sanhong Li

{cengfeng.lzy, kuaiwei.kw, sanhong.lsh} @alibaba-inc.com

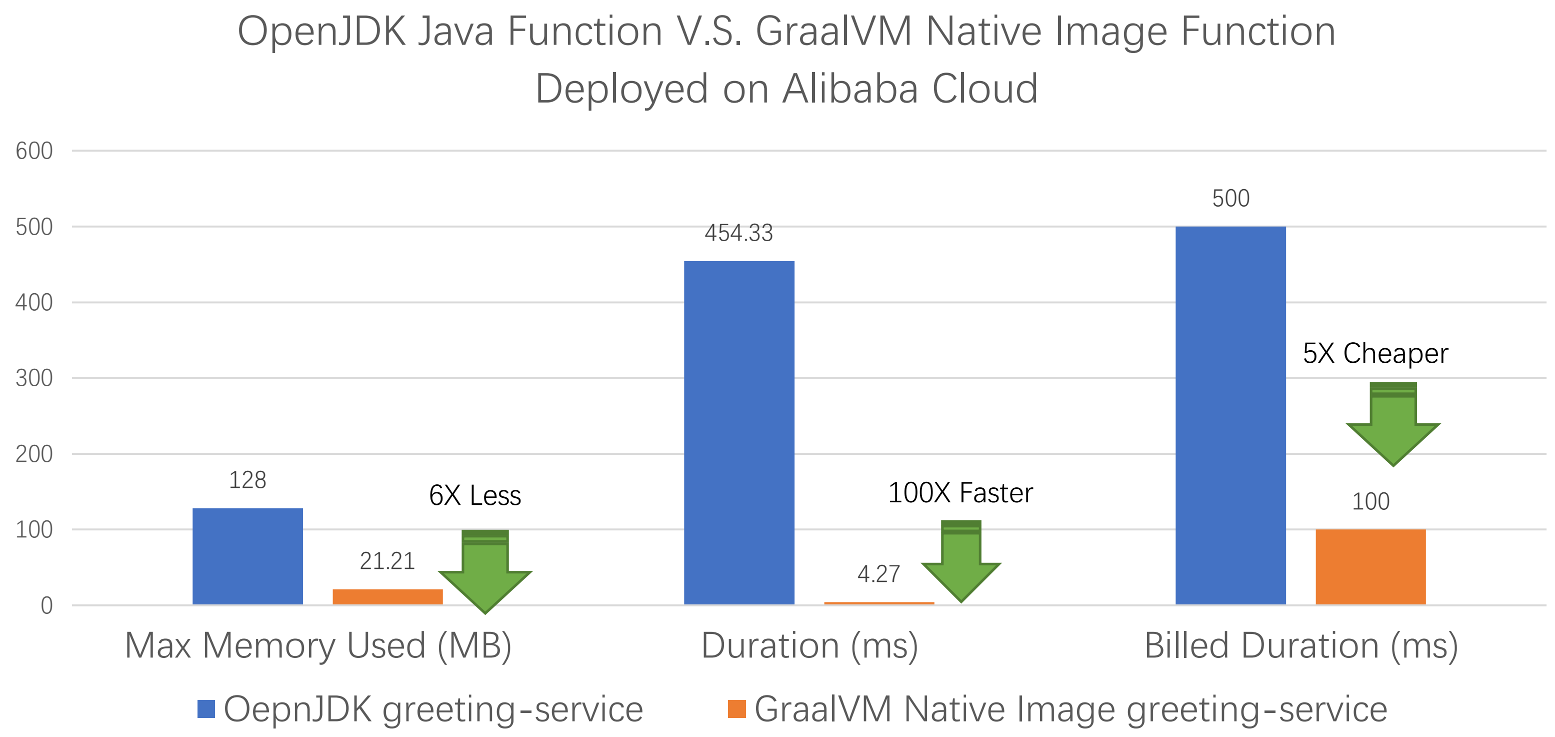
Alibaba Group Inc.

Shanghai, China

Alibaba Cloud | Worldwide Cloud Services Partner

Background

- Why are we interested in native image?
  - Fast startup
  - Less footprint
  - Ideal for FaaS
- Limitations: Not 100% compatible with OpenJDK
  - Not support some key features, e.g. serialization, dynamic class loading and multiple classloaders
  - Stability concerns



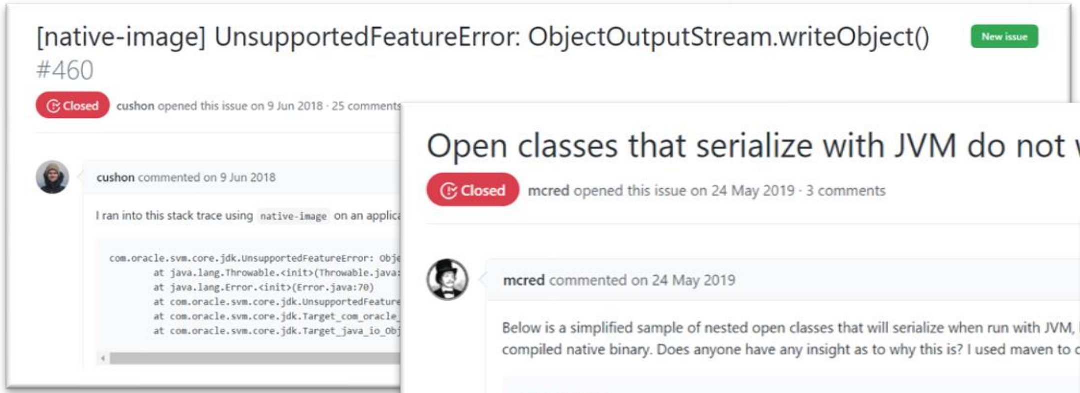
Demo source: https://github.com/micronaut-projects/micronaut-spring/tree/master/examples/greeting-service

Alibaba Cloud

Worldwide Cloud Services Partner

Motivation

- Java serialization is used in Alibaba middleware. Can’t work around when building native image for Alibaba applications.
- Serialization feature is demanded in the GraalVM community for a long time



2018

[native-image] UnsupportedFeatureError: ObjectOutputStream.writeObject() #460

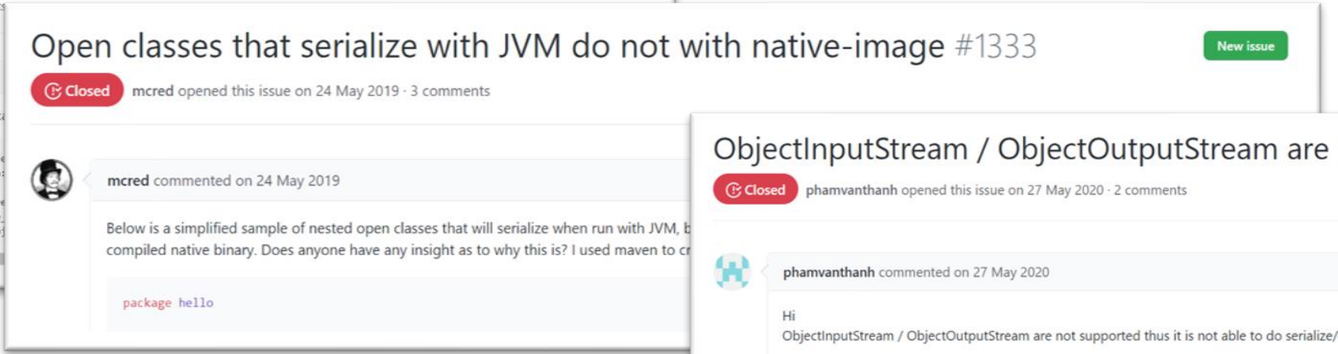
New issue

Closed cushon opened this issue on 9 Jun 2018 · 25 comments

cushon commented on 9 Jun 2018

I ran into this stack trace using `native-image` on an applica

```
com.oracle.svm.core.jdk.UnsupportedFeatureError: Obje
        at java.lang.Throwable.<init>(Throwable.java:
        at java.lang.Error.<init>(Error.java:70)
        at com.oracle.svm.core.jdk.UnsupportedFeature
        at com.oracle.svm.core.jdk.Target_com_oracle_
        at com.oracle.svm.core.jdk.Target_java_io_Obj
```



2019

Open classes that serialize with JVM do not with native-image #1333

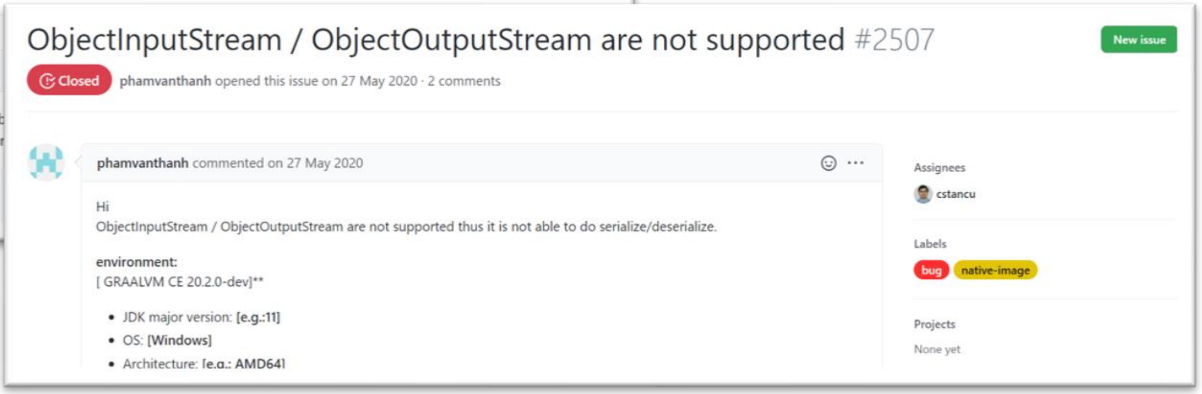
New issue

Closed mcred opened this issue on 24 May 2019 · 3 comments

mcred commented on 24 May 2019

Below is a simplified sample of nested open classes that will serialize when run with JVM, b
compiled native binary. Does anyone have any insight as to why this is? I used maven to cr

```
package hello
```



2020

ObjectInputStream / ObjectOutputStream are not supported #2507

New issue

Closed phamvanthanh opened this issue on 27 May 2020 · 2 comments

phamvanthanh commented on 27 May 2020

Hi
ObjectInputStream / ObjectOutputStream are not supported thus it is not able to do serialize/deserialize.

environment:
[GRAALVM CE 20.2.0-dev]**

- JDK major version: **[e.g.:11]**
- OS: **[Windows]**
- Architecture: **[e.g.: AMD64]**

Assignees

cstancu

Labels

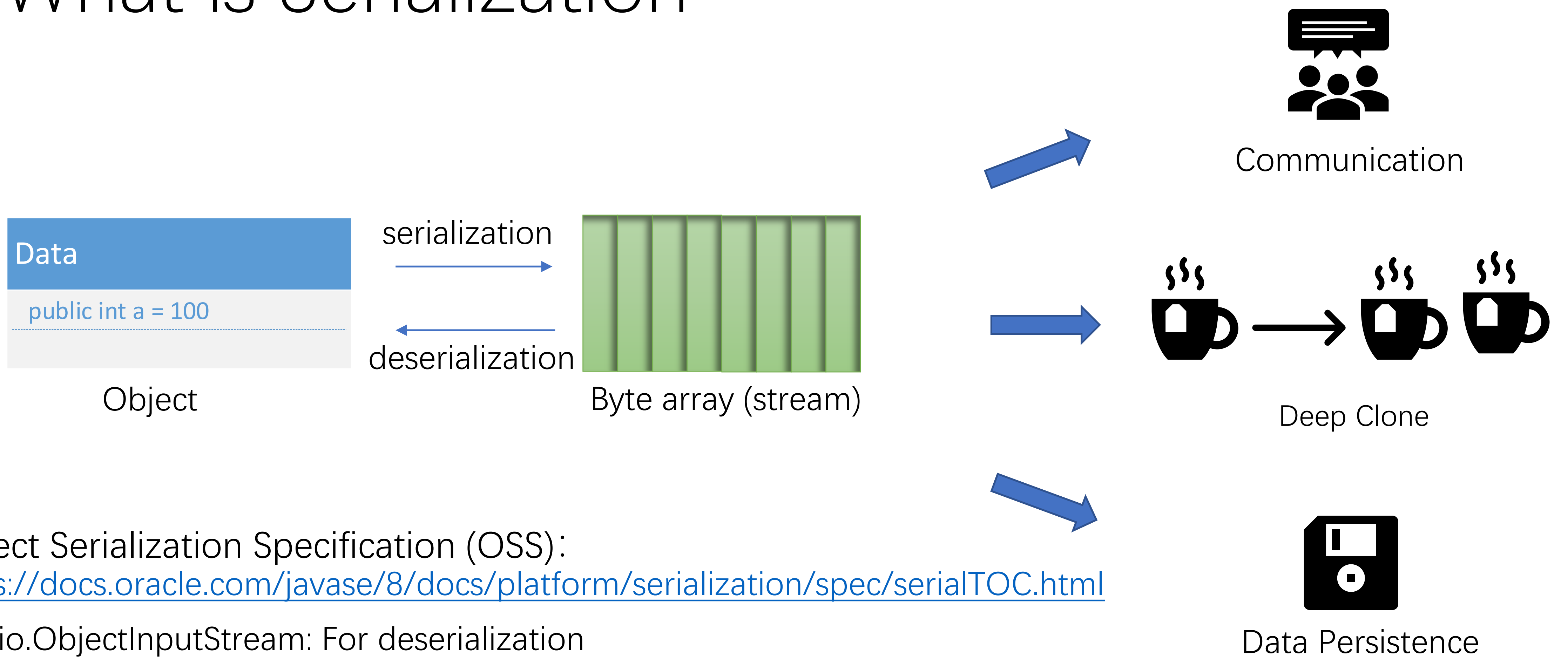
bug native-image

Projects

None yet

Worldwide Cloud Services Partner

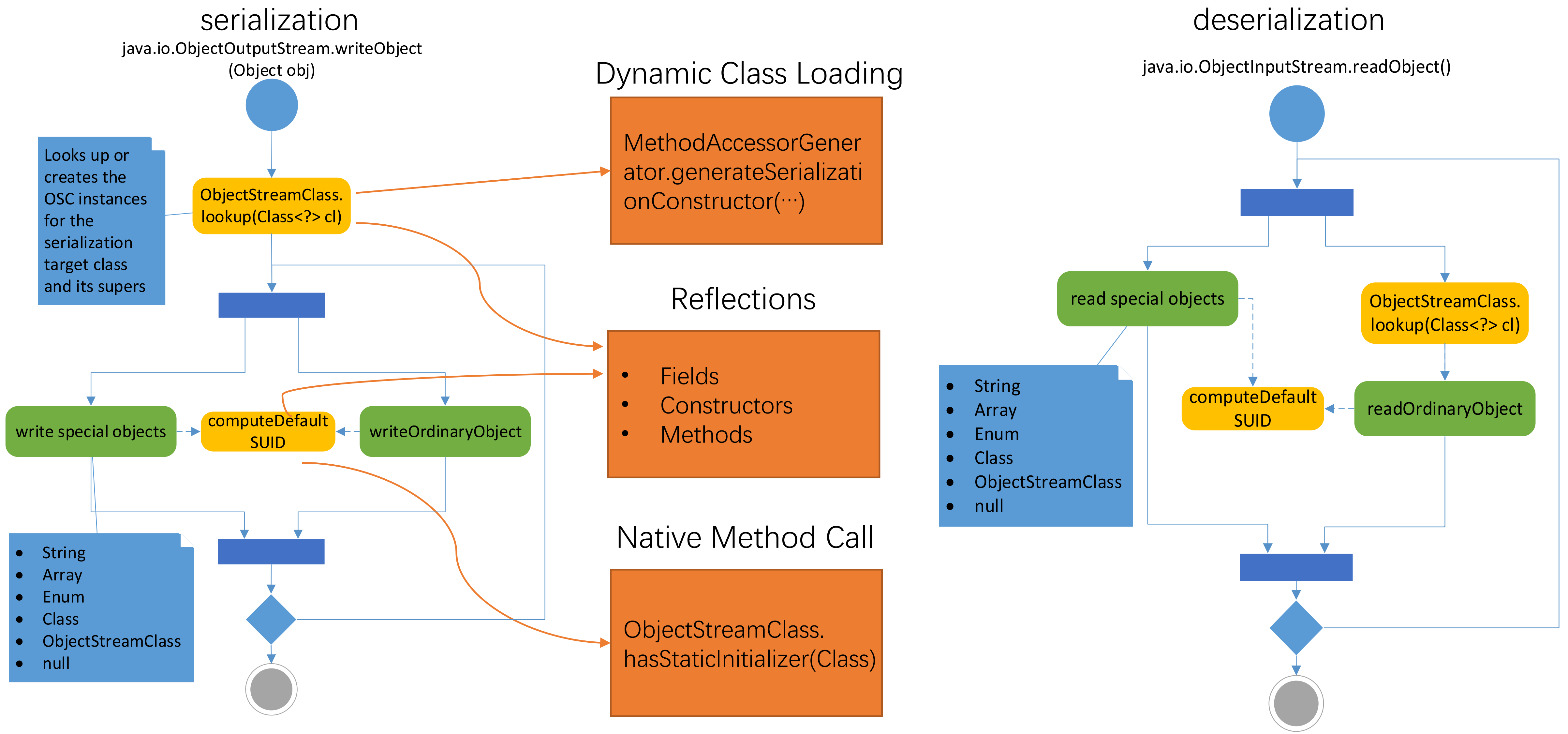
What is Serialization



- Object Serialization Specification (OSS)：
  https://docs.oracle.com/javase/8/docs/platform/serialization/spec/serialTOC.html
- java.io.ObjectInputStream: For deserialization
- java.io.ObjectOutputStream: For serialization
- java.io.ObjectStreamClass: Target class descriptor, short as OSC

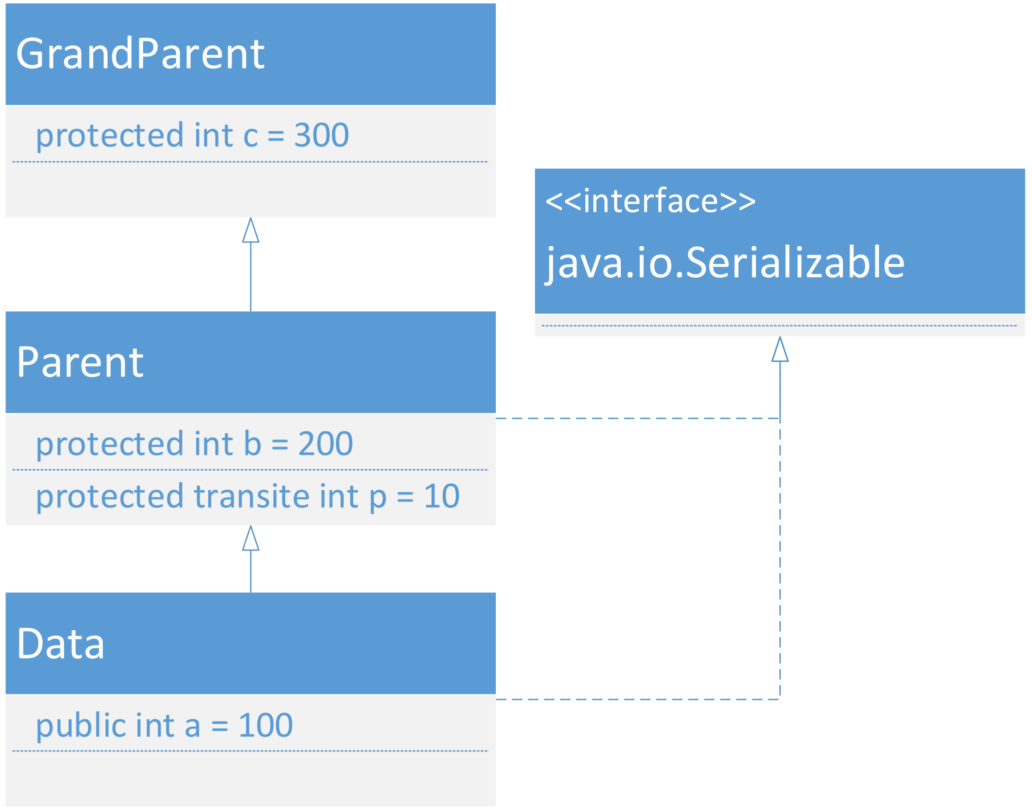
OpenJDK Implementation

Alibaba Cloud | Worldwide Cloud Services Partner


serialization
java.io.ObjectOutputStream.writeObject (Object obj)
Looks up or creates the OSC instances for the serialization target class and its supers
ObjectStreamClass. lookup(Class<?> cl)
write special objects
computeDefault SUID
writeOrdinaryObject
String
Array
Enum
Class
ObjectStreamClass
null
Dynamic Class Loading
MethodAccessorGenerator.generateSerializationConstructor(…)
Reflections
Fields
Constructors
Methods
Native Method Call
ObjectStreamClass. hasStaticInitializer(Class)
deserialization
java.io.ObjectInputStream.readObject()
read special objects
ObjectStreamClass. lookup(Class<?> cl)
computeDefault SUID
readOrdinaryObject
String
Array
Enum
Class
ObjectStreamClass
null

Alibaba Cloud | Worldwide Cloud Services Partner

Object Instantiation at Deserialization



```
private Object readOrdinaryObject(boolean unshared)
        throws IOException
    {
        ...
        Object obj;
        try {
            obj = desc.isInstantiable() ? desc.newInstance() : null;
        } catch (Exception ex) {
            ...
        }
        ...
        readSerialData(obj, desc);

        ...
        return obj;
}
```

a=0
b=0
p=0
c=300

"*the no-arg constructor for the first non-serializable supertype is run*" OSS 3.1.11.a.

Data data = new GrandParent()

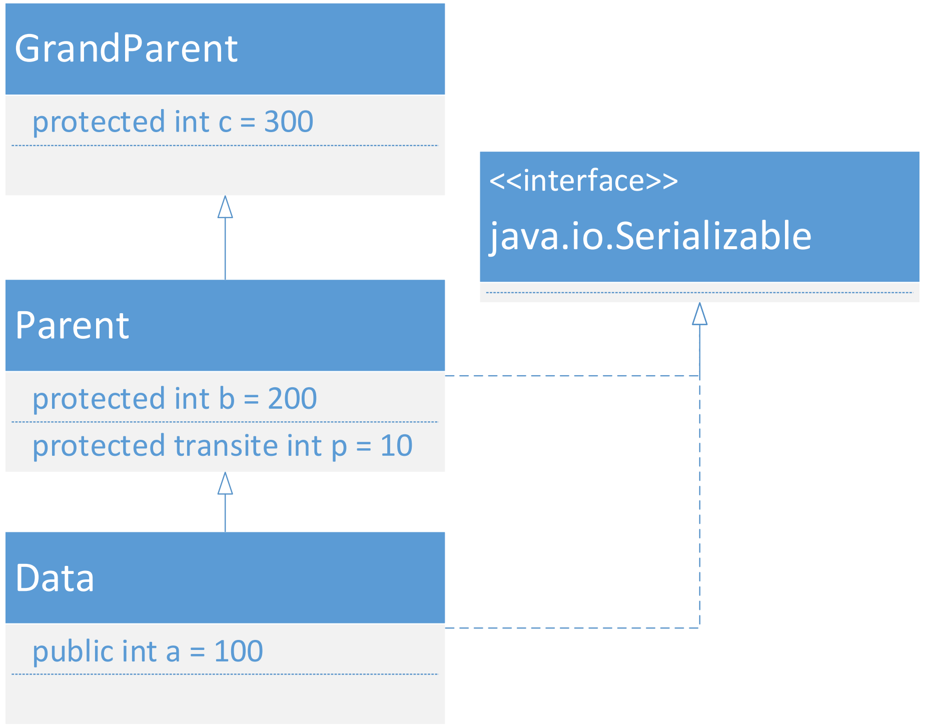
a=1
b=2
p=0
c=300

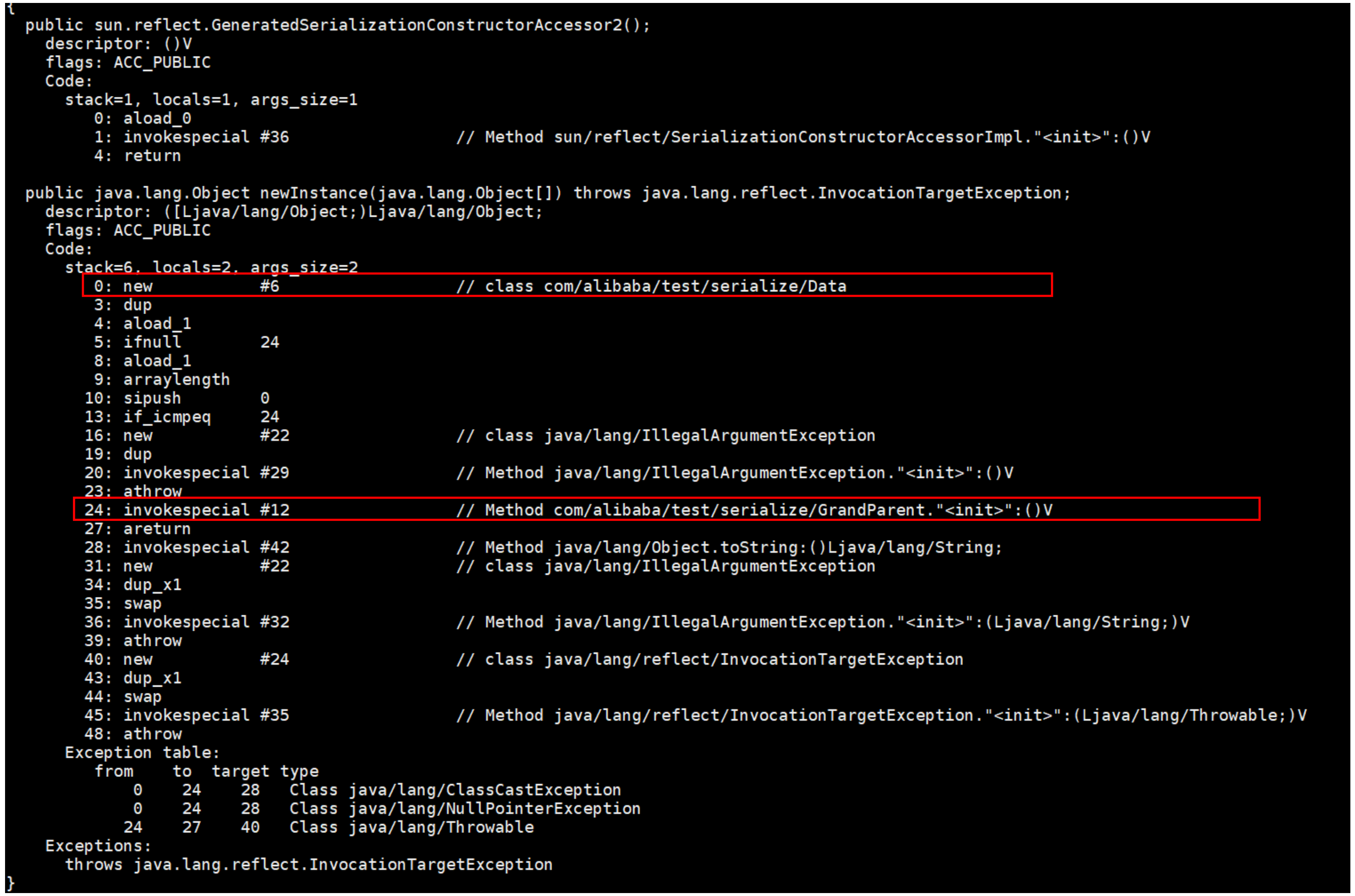
Data data: a=1, b=2, c=3, p=4

Worldwide Cloud Services Partner

Dynamic Class Loading Can Help

- Dynamically generated class “GeneratedSerializationConstructorAccessor” (GSCA)
- It’s almost constant, so is possible to turn to static





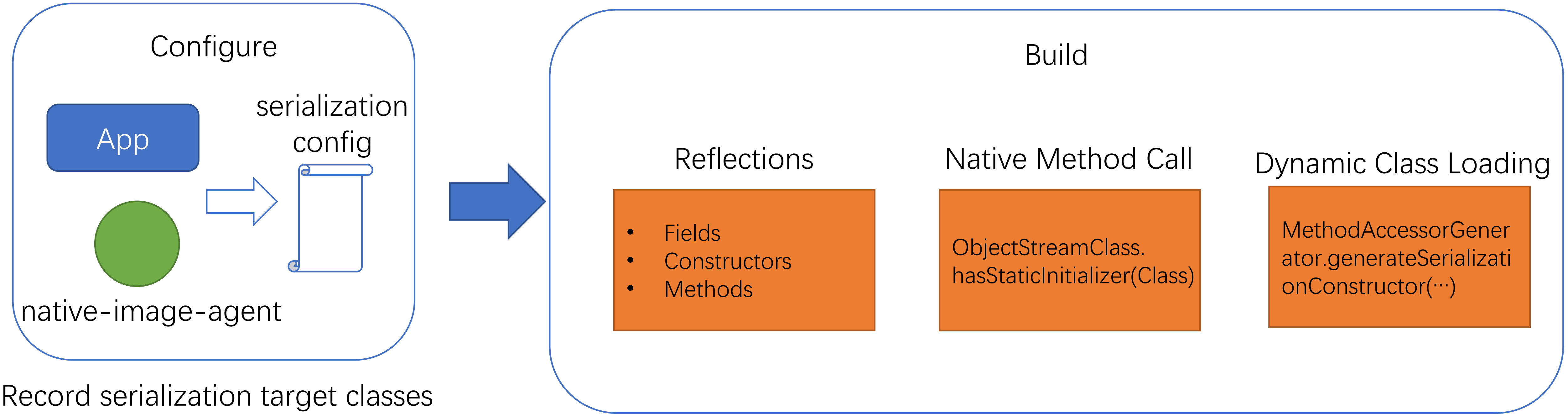
```
{
  public sun.reflect.GeneratedSerializationConstructorAccessor2();
    descriptor: ()V
    flags: ACC_PUBLIC
    Code:
      stack=1, locals=1, args_size=1
         0: aload_0
         1: invokespecial #36                 // Method sun/reflect/SerializationConstructorAccessorImpl."<init>":()V
         4: return

  public java.lang.Object newInstance(java.lang.Object[]) throws java.lang.reflect.InvocationTargetException;
    descriptor: ([Ljava/lang/Object;)Ljava/lang/Object;
    flags: ACC_PUBLIC
    Code:
      stack=6, locals=2, args_size=2
         0: new           #6                  // class com/alibaba/test/serialize/Data
         3: dup
         4: aload_1
         5: ifnull        24
         8: aload_1
         9: arraylength
        10: sipush        0
        13: if_icmpeq     24
        16: new           #22                 // class java/lang/IllegalArgumentException
        19: dup
        20: invokespecial #29                 // Method java/lang/IllegalArgumentException."<init>":()V
        23: athrow
        24: invokespecial #12                 // Method com/alibaba/test/serialize/GrandParent."<init>":()V
        27: areturn
        28: invokespecial #42                 // Method java/lang/Object.toString:()Ljava/lang/String;
        31: new           #22                 // class java/lang/IllegalArgumentException
        34: dup_x1
        35: swap
        36: invokespecial #32                 // Method java/lang/IllegalArgumentException."<init>":(Ljava/lang/String;)V
        39: athrow
        40: new           #24                 // class java/lang/reflect/InvocationTargetException
        43: dup_x1
        44: swap
        45: invokespecial #35                 // Method java/lang/reflect/InvocationTargetException."<init>":(Ljava/lang/Throwable;)V
        48: athrow
      Exception table:
         from    to  target type
             0    24    28   Class java/lang/ClassCastException
             0    24    28   Class java/lang/NullPointerException
            24    27    40   Class java/lang/Throwable
    Exceptions:
      throws java.lang.reflect.InvocationTargetException
}
```

Implementation Overview

Alibaba Cloud | Worldwide Cloud Services Partner

- Overall strategy:
  - Configure the target class
  - Fix the unsupported features, so that the serialization implementation in OpenJDK can be compiled into native image.

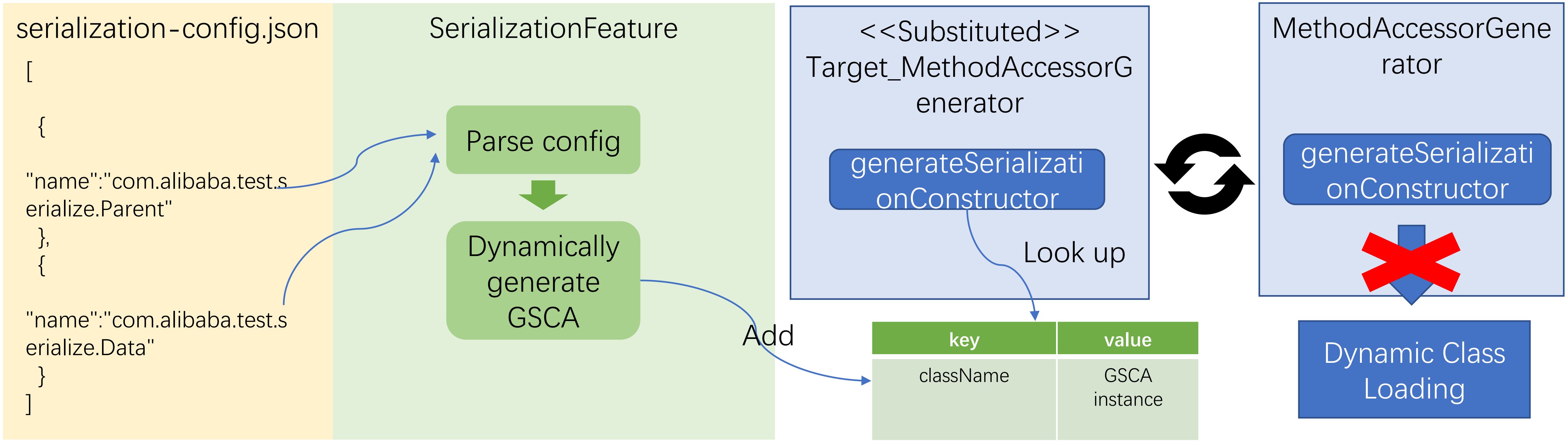


Record serialization target classes

Turn Dynamic to Static

Alibaba Cloud | Worldwide Cloud Services Partner

- The GSCA is almost constant expect target class
- *"Classes are identified by name."*-- OSS 5.3 Assumptions
- Cache GSCA at build time, fetch at runtime


serialization-config.json
SerializationFeature
Parse config
Dynamically generate GSCA
Add
<<Substituted>> Target_MethodAccessorGenerator
generateSerializationConstructor
Look up
key
value
className
GSCA instance
MethodAccessorGenerator
generateSerializationConstructor
Dynamic Class Loading

```
[
  {
"name":"com.alibaba.test.serialize.Parent"
  },
  {
"name":"com.alibaba.test.serialize.Data"
  }
]
```

Results

- Support Apache MINA's RPC now.
- Support JUnit now. We can write JUnit tests for native image programs.
- Support SPECjvm2008 serial benchmark now.

Alibaba Cloud

Worldwide Cloud Services Partner

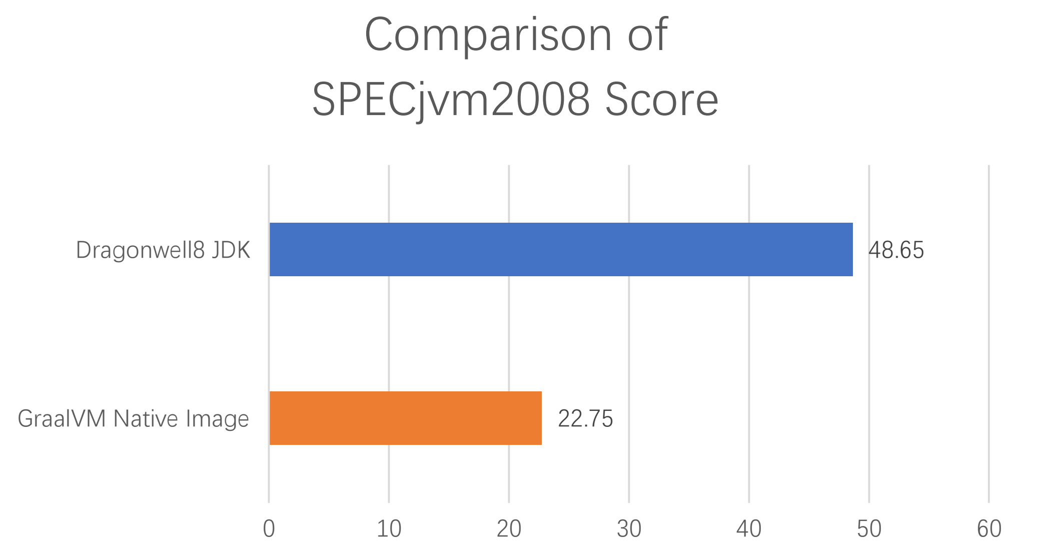
Performance-Setup

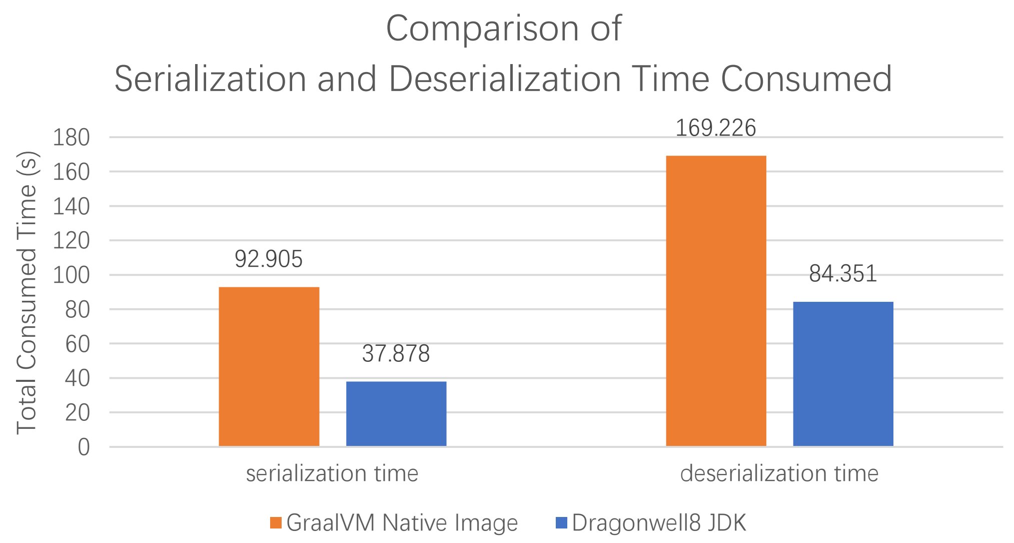
- Testing with SPECjvm2008's ***serial*** benchmark
  - java –Xmx2g –Xms1g -cp SPECjvm2008.jar spec.harness.Launch serial –opts 100 –bt 1
  - ./spec.harness.launch serial –opts 100 –bt 1 –Xmx2g –Xms1g
- GraalVM version:
  - Compiled on 20 Feb 2021 with master branch till commit: https://github.com/oracle/graal/commit/f38cc1648c28b1112f1ceac24d9bf17cc5ba4bca
- JDK version:
  - Alibaba Dragonwell8 JDK 8.6.5* (OpenJDK 8u_282)
- Hardware:
  - Alibaba Cloud Elastic Compute Service Instance
  - Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz, 4 cores
  - Memory 8G

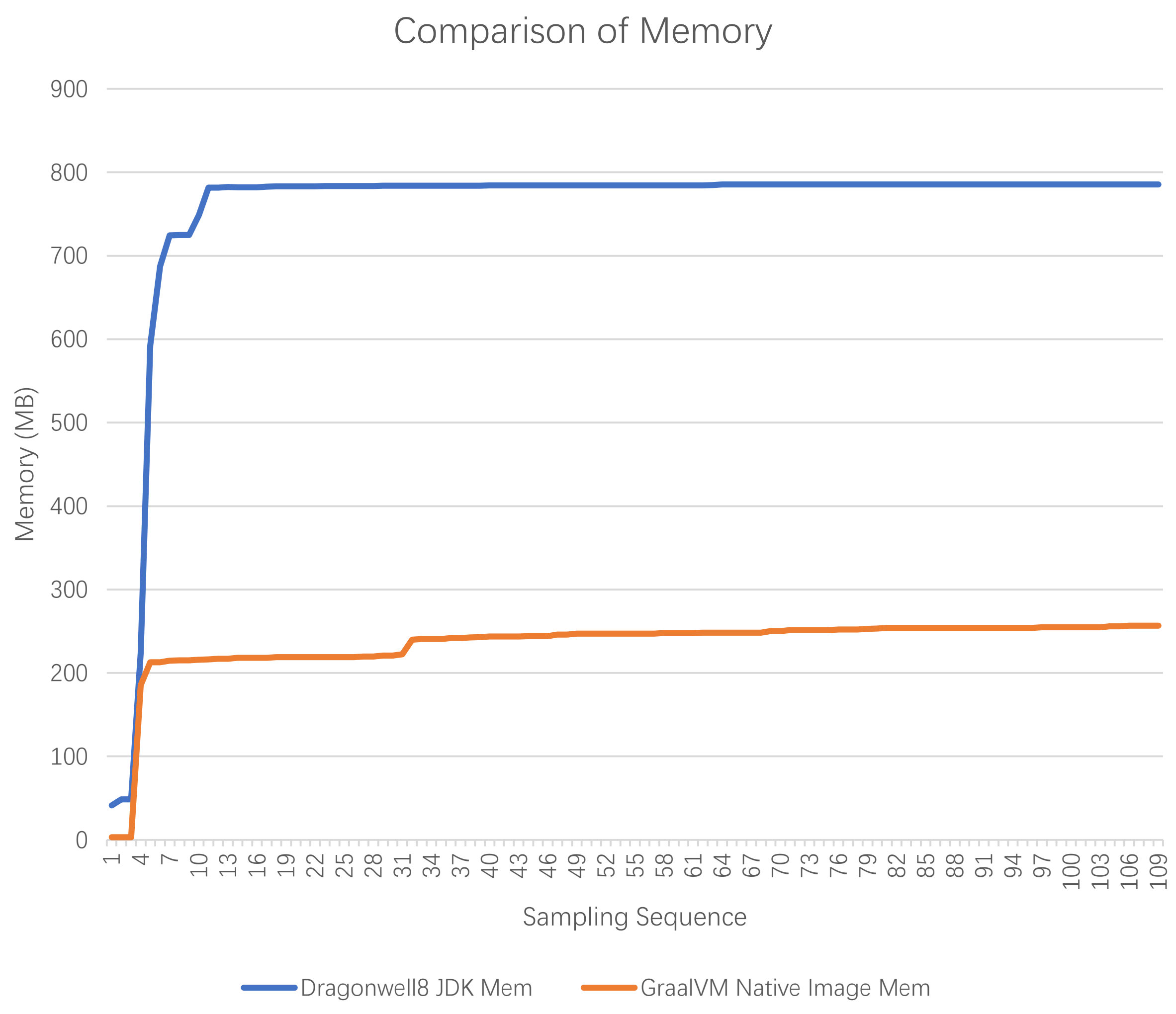
* a downstream of OpenJDK developed by Alibaba

Performance - Result

Alibaba Cloud | Worldwide Cloud Services Partner





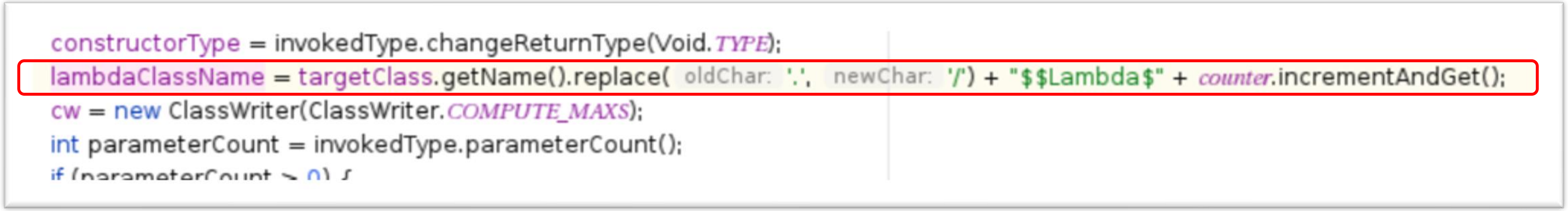


Limitation and Future Work

Alibaba Cloud | Worldwide Cloud Services Partner

- Implicit assumption: The serialization target class name is constant
- Lambda class breaks the assumption
  - Lambda class is dynamically generated
  - The class name is changing
- Future work
  - Solve the lambda limitation
  - Improve performance

java.lang.invoke.InnerClassLambdaMetafactory.<init>



```
constructorType = invokedType.changeReturnType(Void.TYPE);
lambdaClassName = targetClass.getName().replace( oldChar: '.',  newChar: '/') + "$$Lambda$" + counter.incrementAndGet();
cw = new ClassWriter(ClassWriter.COMPUTE_MAXS);
int parameterCount = invokedType.parameterCount();
if (parameterCount > 0) {
```

Worldwide Cloud Services Partner

Summary

- JDK serialization is widely used in Java world, supporting it can help more programs to adapt to native image.
- The keys of the implementation are:
  - Auto-config: Add JVMTI method breakpoint to auto-record all serialization target classes
  - Dynamic-to-static: Generate configured classes' GSCAs and cache them in the native image for runtime usage
- https://github.com/oracle/graal/pull/2730
- Officially released in GraalVM 21.0

Alibaba Cloud

Worldwide Cloud Services Partner

Thank you!