

Support for pre-generated java.lang.invoke classes in CDS archive

- [Goal](#)
- [Background](#)
- [Estimated Benefits](#)
- [Why do it in CDS](#)
- [CDS Design](#)
 - [Static Dump](#)
 - [Dynamic Dump](#)

Goal

Improve start-up time by storing pre-generated LambdaForm handlers in an application-specific CDS archive.

Background

In [JDK-8086045 \(Improve the java.lang.invoke first initialization costs\)](#), the JDK can reduce the number of dynamically generated LambdaForm classes. Here's an example:

```
# Do this in your JDK build directory.
# The exploded JDK build:
$ jdk/bin/javac -J-verbose ~/tmp/HelloWorld.java | grep LambdaForm | grep __JVM_LookupDefineClass__
[0.142s][info][class,load] java.lang.invoke.LambdaForm$MH/0x0000000800066c40 source: __JVM_LookupDefineClass__
[0.143s][info][class,load] java.lang.invoke.LambdaForm$DMH/0x0000000800066040 source: __JVM_LookupDefineClass__
[0.143s][info][class,load] java.lang.invoke.LambdaForm$MH/0x0000000800066440 source: __JVM_LookupDefineClass__
[snip]

$ jdk/bin/javac -J-verbose ~/tmp/HelloWorld.java | grep LambdaForm | grep __JVM_LookupDefineClass__ | wc
   102    408   11400

# The final JDK image:
$ images/jdk/bin/javac -J-verbose ~/tmp/HelloWorld.java | grep LambdaForm | grep __JVM_LookupDefineClass__ | wc
    34    136    3800
```

Here's the difference between the exploded build and the final image:

```
$ jdk/bin/javap 'java.lang.invoke.DirectMethodHandle$Holder'
Compiled from "DirectMethodHandle.java"
final class java.lang.invoke.DirectMethodHandle$Holder {
    final java.lang.invoke.DirectMethodHandle this$0;
    java.lang.invoke.DirectMethodHandle$Holder(java.lang.invoke.DirectMethodHandle);
}

$ images/jdk/bin/javap 'java.lang.invoke.DirectMethodHandle$Holder'
Compiled from "DirectMethodHandle$Holder"
final class java.lang.invoke.DirectMethodHandle$Holder {
    static int invokeInterface(java.lang.Object, java.lang.Object, java.lang.Object);
    static java.lang.Object invokeSpecial(java.lang.Object, java.lang.Object, java.lang.Object, int);
    static java.lang.Object invokeSpecial(java.lang.Object, java.lang.Object, java.lang.Object, long);
    [~150 more lines ...]
}
```

The JDK image's version of the `java.lang.invoke.DirectMethodHandle$Holder` class is generated [here in GenerateJLIClassesPlugin.java](#). This plugin is executed when we generate the file `images/jdk/lib/modules`.

```
private static final String DIRECT HOLDER = "java/lang/invoke/DirectMethodHandle$Holder";
...
byte[] bytes = JLIA.generateDirectMethodHandleHolderClassBytes(
    DIRECT HOLDER, directMethodTypes, dmhTypes);
```

The input of generateDirectMethodHandleHolderClassBytes comes from the [default_jli_trace.txt](#) file that is generated in [GenerateLinkOptData.gmk](#):

```
$(FIXPATH) $(INTERIM_IMAGE_DIR)/bin/java -XX:DumpLoadedClassList=$@.raw.2 \  
-XX:SharedClassListFile=$@.interim -XX:SharedArchiveFile=$@.jsa \  
-Djava.lang.invoke.MethodHandle.TRACE_RESOLVE=true \  
-Duser.language=en -Duser.country=US \  
--module-path $(SUPPORT_OUTPUTDIR)/classlist.jar \  
-cp $(SUPPORT_OUTPUTDIR)/classlist.jar \  
build.tools.classlist.HelloClasslist \  
2> $(LINK_OPT_DIR)/stderr > $(JLI_TRACE_FILE)
```

Here's how roughly the contents of [default_jli_trace.txt](#) correspond to the generated methods in `DirectMethodHandle$Holder`

```
$ grep invokeSpecialIFC ./support/link_opt/default_jli_trace.txt  
[LF_RESOLVE] java.lang.invoke.DirectMethodHandle$Holder invokeSpecialIFC L3I_I (fail)  
[LF_RESOLVE] java.lang.invoke.DirectMethodHandle$Holder invokeSpecialIFC LLI_I (fail)  
  
$ images/jdk/bin/javap 'java.lang.invoke.DirectMethodHandle$Holder' | grep invokeSpecialIFC  
static int invokeSpecialIFC(java.lang.Object, java.lang.Object, java.lang.Object, int);  
static int invokeSpecialIFC(java.lang.Object, java.lang.Object, int);
```

The methods in `DirectMethodHandle$Holder` are used in [InvokerBytecodeGenerator.java](#) when compiling for LambdaForms:

```
private static MemberName lookupPregenerated(LambdaForm form, MethodType invokerType) {  
    if (form.customized != null) {  
        // No pre-generated version for customized LF  
        return null;  
    }  
    String name = form.kind.methodName;  
    switch (form.kind) {  
        [.....]  
        case DIRECT_INVOKE_INTERFACE: // fall-through  
        case DIRECT_INVOKE_SPECIAL: // fall-through  
        case DIRECT_INVOKE_SPECIAL_IFC: // fall-through  
        case DIRECT_INVOKE_STATIC: // fall-through  
        case DIRECT_INVOKE_STATIC_INIT: // fall-through  
        case DIRECT_INVOKE_VIRTUAL: return resolveFrom(name, invokerType, DirectMethodHandle.Holder.  
class);  
    }  
    return null;  
}
```

Estimated Benefits

The expected benefits will be application specific. Here's an experiment with Javac (I modified the JDK makefile to invoke `com.sun.tools.javac.Main` while generating [default_jli_trace.txt](#)).

```
Results of " perf stat -r 50 bin/javac -J-Xshare:on -J-XX:SharedArchiveFile=javac2.jsa Bench_HelloWorld.java "  
1: 2239149872 2203293903 (-35855969) ---- 375.580 367.190 (-8.390) ----  
2: 2245304165 2198900711 (-46403454) ----- 375.350 366.910 (-8.440) ----  
3: 2242558895 2205167765 (-37391130) ---- 376.720 366.480 (-10.240) -----  
4: 2246851428 2198990987 (-47860441) ----- 375.119 367.010 (-8.109) ----  
5: 2238549008 2202480450 (-36068558) ---- 374.100 367.600 (-6.500) ---  
6: 2240816859 2200725384 (-40091475) ---- 375.743 366.880 (-8.863) ----  
7: 2243272464 2199639926 (-43632538) ----- 374.296 365.990 (-8.306) ----  
8: 2239650882 2203211291 (-36439591) ---- 375.315 365.520 (-9.795) -----  
9: 2242434935 2202380291 (-40054644) ---- 376.052 367.305 (-8.747) ----  
10: 2238725993 2201342398 (-37383595) ---- 375.478 366.743 (-8.735) ----  
=====   
2241729810 2201612439 (-40117371) ---- 375.375 366.762 (-8.612) ----  
instr delta = -40117371 -1.7896%  
time delta = -8.612 ms -2.2943%
```

Why do it in CDS

- We cannot store all possible LambdaForm handlers in the standard JDK image – we cannot enumerate all possible forms that may be used by all possible apps.
- We could also do this as part of jlink, when creating a custom JDK image. However, some users may not want to create a new JDK image.
 - Also, there's currently no work-flow to include profiling data when building a custom JDK image (although this can be changed, probably as part of Project Leyden)
- CDS dynamic dump can be used without generating profiling data in a separate step, so the usability is better.

CDS Design

We regenerate the XXX\$Holder classes to contain all necessary methods used by the application.

Static Dump

This has been implemented in [JDK-8247536](#) (Support for pre-generated java.lang.invoke classes in CDS static archive - [integrated into JDK 16](#)).

During the trial run (with `-XX:DumpLoadedClassList=classlist`), we can save the list of LambdaForm handlers using a special syntax like:

```
# regular class specifier
java/lang/Object
# LambdaForm specifier
@lambda-form-invoker [LF_RESOLVE] java.lang.invoke.DirectMethodHandle$Holder invokeSpecialIFC L3I_I
@lambda-form-invoker [SPECIES_RESOLVE] java.lang.invoke.SimpleMethodHandle
```

We collect all the `@lambda-form-invoker` lines when parsing the classlist. At the end of the static dump:

- We invoke `generateDirectMethodHandleHolderClassBytes` to generate a customized version of `DirectMethodHandle$Holder` (as well as other holder classes that are currently generated by [GenerateJLIClassesPlugin.java](#))
 - This `DirectMethodHandle$Holder` class has all the methods needed for resolving the MethodHandle LambdaForms during the trial run.
 - Store the InstanceKlass of the customized version of `DirectMethodHandle$Holder` into the CDS archive.

Preliminary webrev:

- <http://cr.openjdk.java.net/~minqi/2020/8247536/webrev-01/>
- The generation of the customized `DirectMethodHandle$Holder` is done in `MetaspaceShared::regenerate_holder_classes()`.

Dynamic Dump

- See [JDK-8255493](#) (Support for pre-generated java.lang.invoke classes in CDS dynamic archive)
- Collect a `@lambda-form-invoker` list (in memory) when `-XX:ArchiveClassesAtExit` is specified.
- When the dynamic archive is dumped, use this list to create customized Holder classes.
- TBD