Java 9 Support in Apache Hadoop

May 18, 2017
NTT DATA Corporation
Akira Ajisaka
Self introduction

Akira Ajisaka

- Open-Source Software (OSS) Professional Services Team
  - Technical support related to Hadoop/ OSS for our customers
  - Design, integrate, deploy, and operate clusters in the range of 10 - 1200+ servers

- Apache Hadoop Committer & PMC member
  - Fixing test failures
  - Upgrading and managing dependencies
  - Help release process
Agenda

- What is Java 9?
- Why Apache Hadoop does not support Java 9 now?
- Jigsaw
- Classpath isolation in Apache Hadoop
Apache Hadoop and Java

- Hadoop is mainly written in Java (93.1%)
- Hadoop 3.x supports Java 8 only
- Hadoop 2.7+ supports Java 7 and 8
Java 9

- will be released in Jul. 27
- b169 is available as of May 13
- Many new features (e.g. Jigsaw)
- Many incompatible changes

- Meanwhile, Java 8 will be EoL soon
  - Oracle ends public update in Sep 2017
  - Redhat ends public update in Oct 2020
- Need to prepare!
Now Apache Hadoop doesn't work with Java 9

$ mvn install -DskipTests

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 19.719 s
[INFO] Finished at: 2017-02-20T15:25:07+09:00
[INFO] Final Memory: 82M/275M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-war-plugin:2.4:war
  (default-war) on project hadoop-auth-examples: Execution default-war of goal or
  g.apache.maven.plugins:maven-war-plugin:2.4:war failed: Unable to load the mojo
  'war' in the plugin 'org.apache.maven.plugins:maven-war-plugin:2.4' due to an AP
  I incompatibility: org.codehaus.plexus.component.repository.exception.ComponentL
  ookupException: null
[ERROR] ------------------------------------------------------------------------
```

Cannot compile!
Why compile fails

- 6 problems
  - Encapsulated internal APIs (JEP 260)
  - Banned _ one character identifier (JEP 213)
  - New Version-String Scheme (JEP 223)
  - HTML5 Javadoc (JEP 224)
  - Libraries does not support Java 9
  - Jigsaw
- Work in progress ([HADOOP-11123](https://issues.apache.org/jira/browse/HADOOP-11123), Umbrella JIRA)
/**
 * Unmaps the block from memory.  See munmap(2).
 *
 * There isn't any portable way to unmap a memory region in Java.
 * So we use the sun.nio methods.
 *
 * Note that unmap(MappedByteBuffer) doesn't continue
 * manually to do it, and...
 *
 * @param buffer ... The buffer to unmap.
 */

public static void munmap(MappedByteBuffer buffer) {  
  if (buffer instanceof sun.nio.ch.DirectBuffer) {  
    sun.misc.Cleaner cleaner =  
      ((sun.nio.ch.DirectBuffer)buffer).cleaner();  
    cleaner.clean();  
  }
}
JEP 260: Encapsulate most internal APIs

- sun.misc.Cleaner was moved to sun.misc.Unsafe::invokeCleaner

- Usage
  - Explicit caching mechanism for HDFS
  - Cleaner cleans up off-heap caches
  - equivalent to munmap(2)
Support *both* Java 8 and Java 9

- Use Reflection to call methods directly
  - First, call `sun.misc.Unsafe#invokeCleaner`
  - If exception is thrown, then call `sun.misc.Cleaner`

- Apache Lucene hits the same problem
  - Apache Lucene is a pioneer!
  - Fixed in [LUCENE-6989](https://issues.apache.org/jira/browse/LUCENE-6989)

- Patch available in [HADOOP-12760](https://issues.apache.org/jira/browse/HADOOP-12760)
Use of '_' as an identifier might not be supported in releases after Java 8. More...
Banned _ one character identifier
Frequently used by Hamlet, original framework for Hadoop Web UI
Inspired from Haml

```html
<html>
  <body>
    <table id="applications">
      <thead>
        <tr>
          <td>ApplicationId</td>
          <td>ApplicationState</td>
        </tr>
      </thead>
      <tbody>
        <!-- Table body goes here -->
      </tbody>
    </table>
  </body>
</html>
```
Be careful with compatibility

- Do not just replace _ with __
  - It affects YARN application
    - e.g. Apache Slider (Incubating)
- How to deal with this problem
  - Create new Hamlet2 package with __
  - Deprecate old Hamlet package
  - Replace the usage of _ with __
  - Ignore old Hamlet when compile with Java 9
    - Configure via Maven Compiler Plugin
- Patch available in HADOOP-11875
Fail to recognize Java version
JEP 223: New Version-String Scheme

- Java 8: “1.8.0_”
- Java 9: “9.X.X”

- Affected if regular expression is used to detect Java version
  - Old Maven Javadoc Plugin is affected
  - You must upgrade to 2.10.4+
  - Fixed by [HADOOP-14056](HADOOP-14056)
Fail with Java 9
JEP 224: HTML5 Javadoc

- Validation for existing html files become strict to support HTML5
- Example
  - `<table>` tag requires summary or caption
  - ‘<‘ in `<pre>` tag must be rewritten to `&lt;`
- Fixed by [HADOOP-14057](HADOOP-14057)
Update libraries to the version that supports Java 9

- JUnit 3, 4 -> 5
- Mockito 1 -> 2
- Log4J 1 -> 2
- and many more...
$ mvn install -DskipTests

[ERROR] Number of foreign imports: 1
[ERROR] import: Entry[import from realm ClassRealm[project>org.apache.hadoop:hadoop-main:3.0.0-alpha3-SNAPSHOT, parent: ClassRealm[maven.api, parent: null]]]
[ERROR]
[ERROR] ----------------------------------------------------------------------: ExceptionInInitializerError: Unable to make field private final java.util.Comparator java.util.TreeMap.comparator accessible: module java.base does not "opens java.util" to unmanaged module @87cb1d8

What happened?
private fields/methods cannot be accessed from outside
- Use Field/Method.setAccessible(true) to access
- However, in Java 9, the method can successfully be executed from only the configured 'modules'
- What is 'module'?
Quote from “JavaOne 2015 keynote”

What’s this?
Long classpath -> JAR hell

- Answer: Hadoop classpath
- The very long classpath often cause version conflicts of the libraries (JAR hell) between Hadoop and its applications
- Example:
  - Hadoop uses Guava 21.0
  - Hadoop uses HBase as the backend for YARN Timeline Service v2
  - HBase uses Guava 11.0.2

Now 11.0.2 and the version is now configurable. (HADOOP-14380, HADOOP-14386)
Jigsaw introduces ‘module’

- Create module-info.java for each module to define the dependency between modules

```java
$ cat src/com.greetings/module-info.java
module com.greetings {
    requires com.astro;
}
```

```
$ cat src/com.astro/module-info.java
module com.astro {
    exports com.astro;
}
```

- Require external com.astro package
- Expose com.astro package for other module
‘module’ can enforce the visibility

- Hadoop is using @interfaceAudience annotation to specify the visibility
  - @Private is internal use within the project, but it is public
  - Public is TOO public :( 
- ‘module’ can enforce the visibility :)

```
$ cat src/com.astro/module-info.java
module com.astro {
  exports com.astro to com.greetings;
}
```

Expose com.astro package to only com.greetings module
JAR hell will be fixed by Jigsaw?

- If Apache Hadoop supports Jigsaw...
  - Only the public API of Apache Hadoop is exposed
  - Public API of the dependencies is not exposed

- Therefore, JAR hell will be fixed!

- However, there are a lot of work to do
TODO list for Jigsaw support

- Fix incompatibility introduced by Jigsaw (MWAR-405, etc.)
  - There is ‘--permit-illegal-access’ option for workaround
- Create module-info.java for each module ([HADOOP-14269](HADOOP-14269))
  - jdeps command can help
- Confirm Hadoop can successfully compiled with both Java 8 and 9
  - Java 8 cannot compile module-info.java, so configure maven-compiler-plugin to ignore
Jigsaw has not approved by Java Community Process yet

Jigsaw feature will be updated, so my slide can become incorrect afterwards
We can’t wait for Java 9 Jigsaw!

- To fully support Jigsaw, update all the dependencies that supports Jigsaw
  - If there are no module-info.java, the module is ‘unnamed module’
  - Probably, it takes a very long time to remove ‘unnamed module’ from classpath

- Jigsaw is not only the solution for JAR hell

- Classpath isolation is in progress (HADOOP-11656)
  - Shading Hadoop client artifacts (HADOOP-11804)
  - Classloader improvement (HADOOP-13070)
Shading Hadoop client artifacts (HADOOP-11804)

- Introduce 2 new modules to avoid leaking Hadoop's dependencies onto the applications' classpath
- hadoop-client-api module
  - removed all the transitive dependencies from hadoop-client module
  - only org.apache.hadoop.* are included
- hadoop-client-runtime module
  - add 3rd party dependencies to hadoop-client-api
  - replace the dependency under org.apache.hadoop.shaded. by maven-shade-plugin
- Available in Apache Hadoop 3.0.0-alpha2
Users can use the different versions of Hadoop's dependency

- Set hadoop-client-runtime with runtime scope
- Set the dependency and its version as you like

```xml
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.8.7</version>
</dependency>
```

- You can use the two different version

```java
import com.fasterxml.jackson.databind.ObjectMapper;

public class Example {
  private static final ObjectMapper newMapper = new ObjectMapper();
  private static final ObjectMapper oldMapper = new org.apache.hadoop.shaded.com.fasterxml.jackson.databind.ObjectMapper();
```
Classloader improvement (HADOOP-13070)

- Now user class can load a class from Hadoop's dependencies with or without ApplicationClassLoader
  - That way dependency conflicts can occur

- In this issue, we modify ApplicationClassLoader to prevent a user class from loading a class from the parent classpath
  - Check the caller when loading a class
  - If the caller is an user class, prevent loading a class from the parent classpath

- Patch available in HADOOP-13398
Conclusion

- Now Apache Hadoop does not support Java 9
  - The big work is in progress
  - Your contribution is welcome

- 'JAR hell' problem will be gradually resolved
  - Let's try the new hadoop-client-api/hadoop-client-runtime modules and the new class loader!