Item 374
git_comments:

git_commits:

1. **summary:** Cleanup transitive dependencies for zookeeper (#3154)
   message: Cleanup transitive dependencies for zookeeper (#3154) -- exclude log4j and slf4j -- exclude jline as thats only required for cli

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings
2. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings
3. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 |

+--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings

4. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings
5. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings
6. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 ->

1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings

7. **title:** Avoid packaging log4j dependencies in invoker
   body: Currently invoker module pulls in log4j jars via transitive dependency on zookeeper ``` $./gradlew :core:invoker:dependencies +--- org.apache.zookeeper:zookeeper:3.4.11 | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | +--- org.slf4j:slf4j-log4j12:1.6.1 | | +--- org.slf4j:slf4j-api:1.6.1 -> 1.7.25 | | \--- log4j:log4j:1.2.16 | +--- log4j:log4j:1.2.16 | +--- jline:jline:0.9.94 | | \--- junit:junit:3.8.1 | +--- org.apache.yetus:audience-annotations:0.5.0 | \--- io.netty:netty:3.10.5.Final ``` This causes following warning logs in invoker logs and test runs ``` SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/invoker/lib/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in [jar:file:/invoker/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder] ``` With this PR we exclude following transitive dependencies * log4j and slf4j - Prevents pulling of log4j related jars in invoker distribution * jline - This is required only for CLI which also pull in junit!. For 3.5.0 version these have been made optional ([ZOOKEEPER-1655][1]) With these exclusion invoker.tar size reduces from 46.4M to 45.7M and also the slf4j warnings are not seen. This would also ensure that logging backend is deterministic and not [dependent on runtime][1] > The way SLF4J picks a binding is determined by the JVM and for all practical purposes should be considered random. As of version 1.6.6, SLF4J will name the framework/implementation class it is actually bound to. [1]: https://issues.apache.org/jira/browse/ZOOKEEPER-1655 [2]: https://www.slf4j.org/codes.html#multiple_bindings

github_pulls_comments:

1. @mhenke1 Can you please review?
2. @chetanmeh, can you fix the conflicting build.gradle file?
3. Fixed the conflict
4. Thanks @chetanmeh! Any idea how to avoid the `[INFO] Slf4jLogger started` messages that appear in the logs?
5. > Any idea how to avoid the [INFO] Slf4jLogger started messages that appear in the logs? @dubeejw This log comes from [akka SLF4JLogging][1] which is probably to confirm that slf4j is setup properly. We can disable this by setting log level for `akka.event.slf4j.Slf4jLogger` to WARN [1]: https://github.com/akka/akka/blob/009214ae07708e8144a279e71d06c4a504907e31/akka-slf4j/src/main/scala/akka/event/slf4j/Slf4jLogger.scala#L89

github_pulls_reviews:

jira_issues:

1. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
2. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.

3. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
4. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
5. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
6. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
7. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
   label: code-design
8. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
9. **summary:** Make jline dependency optional in maven pom
   description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
10. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
11. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
12. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
13. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
14. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.
15. **summary:** Make jline dependency optional in maven pom
    description: Old JLine version used in ZK CLI should not be pulled into downstream projects.

jira_issues_comments:

1. Dependency in POM now generated as: {code} <dependency> <groupId>jline</groupId> <artifactId>jline</artifactId> <version>0.9.94</version> <optional>true</optional> </dependency> {code}
2. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12571355/ZOOKEEPER-1655.patch against trunk revision 1448007. +1 @author. The patch does not contain any @author tags. -1 tests included. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. +1 javadoc. The javadoc tool did not generate any warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. +1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed core unit tests. +1 contrib tests. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-ZOOKEEPER-Build/1414//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-ZOOKEEPER-Build/1414//artifact/trunk/build/test/findbugs/newPatchFindbugsWarnings.html Console output: https://builds.apache.org/job/PreCommit-ZOOKEEPER-Build/1414//console This message is automatically generated.
3. I'm trying to understand the implications of this patch. One of the concerns in HADOOP-9342 was the version of jline. Should we bump up the version here? Otherwise it sounds good to make it optional.
4. Thanks for the feedback, Thomas. Could you update the patch to upgrade the jline version number, please? If you haven't done it, please make sure it applies to both trunk and 3.4 branch.
5. Cancelling patch to upgrade the jline version.
6. [~isnogood], Do you think you can update the patch to upgrade the jline version?
7. **body:** I'm not sure it is a drop in replacement, need to try it. Would prefer to separate this from the leaking dependency though. Even if we manage to bump the version, we should still not export the dependency via the Maven pom.
   label: code-design
8. I'm concerned about this change for 3.4. It's less of a bug fix and more of an improvement imo - a change which could destabilize existing downstream users. It's easy enough to specify an exclude if you don't

want. I'm good for changing this in trunk, but I don't see this for a fix release on a stable branch. (just my opinion though, lmk if you feel strongly otw). If this is ok lmk and I'll commit to trunk.
9. If JLine gets upgraded in the 3.x line (ZOOKEEPER-1718), then I would suggest to also make the change to the pom to not export the dependency since downstream would be affected in any case. Commit to trunk would be great in any case.
10. Great. Let's shoot for trunk then (3.5.0).
11. Updated the patch for ZOOKEEPER-1718 (upgrade to jline2)
12. Committed to trunk. Thanks Thomas!
13. SUCCESS: Integrated in ZooKeeper-trunk #2076 (See [https://builds.apache.org/job/ZooKeeper-trunk/2076/]) ZOOKEEPER-1655. Make jline dependency optional in maven pom (Thomas Weise via phunt) (phunt: http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1528224) * /zookeeper/trunk/CHANGES.txt * /zookeeper/trunk/ivy.xml
14. Adding deleted comment: Thomas Weise (duplicate account) added a comment - 02/Sep/13 09:19 I think leaking the jline dependency is not desirable because it is only needed to run the ZK command line tools while most projects would only use the client API. For example, jline made its way into the Hadoopdistribution until it was excluded explicitly. As for upgrading jlline, that's a good idea as it would make features such as reverse history search available in the ZK CLI.