Item 92
git_comments:

git_commits:

1. **summary:** GEODE-8875: Removing packages element from log4j2 configuration file
   message: GEODE-8875: Removing packages element from log4j2 configuration file This element causes log4j to scan all geode jars on startup. This slows down the startup time of all geode processes, including gfsh. It is not necessary because log4j has an annotation processor to generate a config file that helps it find plugs that it needs.
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Geode is spending a lot of time in log4j classpath scanning on startup
   description: When running gfsh, it takes a long time even to run the `gfsh help` command. For example, with our docker image it takes 10 seconds just to display the help from {code} docker run apachegeode/geode:latest time gfsh help {code} Doing some CPU sampling, it looks like the majority of the time is in log4j2 scanning for plugins. {noformat} TRACE 300879: (thread=200001)
   java.lang.ClassLoader.defineClass1(ClassLoader.java:Unknown line)
   java.lang.ClassLoader.defineClass(ClassLoader.java:763)
   java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
   java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
   java.net.URLClassLoader.access$100(URLClassLoader.java:74)
   java.net.URLClassLoader$1.run(URLClassLoader.java:369)
   java.net.URLClassLoader$1.run(URLClassLoader.java:363)
   java.security.AccessController.doPrivileged(AccessController.java:Unknown line)
   java.net.URLClassLoader.findClass(URLClassLoader.java:362)
   java.lang.ClassLoader.loadClass(ClassLoader.java:424)
   sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:349)
   java.lang.ClassLoader.loadClass(ClassLoader.java:357)
   org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.addIfMatching(ResolverUtil.java:449)
   org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.loadImplementationsInJar(ResolverUtil.java:351)
   org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.findInPackage(ResolverUtil.java:233)
   org.apache.logging.log4j.core.config.plugins.util.PluginRegistry.loadFromPackage(PluginRegistry.java:221)
   org.apache.logging.log4j.core.config.plugins.util.PluginManager.collectPlugins(PluginManager.java:152)
   org.apache.logging.log4j.core.config.AbstractConfiguration.initialize(AbstractConfiguration.java:224)
   org.apache.logging.log4j.core.config.AbstractConfiguration.start(AbstractConfiguration.java:288)
   org.apache.logging.log4j.core.LoggerContext.setConfiguration(LoggerContext.java:579) {noformat} After further investigation, this scanning appears to be caused by this line in our log4j2-cli.xml file, which sets the packages to be "org.apache.geode" This causes log4j2 to scan all of our jar files. {code} <Configuration status="WARN" shutdownHook="disable" packages="org.apache.geode"> {code} This scanning is completely unnecessary because log4j already has an annotation processor which generates a file that log4j2 can find without any scanning. This probably affects geode server startup, not just gfsh, because log4j2.xml has this setting as well.
2. **summary:** Geode is spending a lot of time in log4j classpath scanning on startup
   description: When running gfsh, it takes a long time even to run the `gfsh help` command. For example, with our docker image it takes 10 seconds just to display the help from {code} docker run apachegeode/geode:latest time gfsh help {code} Doing some CPU sampling, it looks like the majority of the time is in log4j2 scanning for plugins. {noformat} TRACE 300879: (thread=200001)
   java.lang.ClassLoader.defineClass1(ClassLoader.java:Unknown line)
   java.lang.ClassLoader.defineClass(ClassLoader.java:763)

java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
java.net.URLClassLoader.access$100(URLClassLoader.java:74)
java.net.URLClassLoader$1.run(URLClassLoader.java:369)
java.net.URLClassLoader$1.run(URLClassLoader.java:363)
java.security.AccessController.doPrivileged(AccessController.java:Unknown line)
java.net.URLClassLoader.findClass(URLClassLoader.java:362)
java.lang.ClassLoader.loadClass(ClassLoader.java:424)
sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:349)
java.lang.ClassLoader.loadClass(ClassLoader.java:357)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.addIfMatching(ResolverUtil.java:449)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.loadImplementationsInJar(ResolverUtil.java:351)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.findInPackage(ResolverUtil.java:233)
org.apache.logging.log4j.core.config.plugins.util.PluginRegistry.loadFromPackage(PluginRegistry.java:221)
org.apache.logging.log4j.core.config.plugins.util.PluginManager.collectPlugins(PluginManager.java:152)
org.apache.logging.log4j.core.config.AbstractConfiguration.initialize(AbstractConfiguration.java:224)
org.apache.logging.log4j.core.config.AbstractConfiguration.start(AbstractConfiguration.java:288)
org.apache.logging.log4j.core.LoggerContext.setConfiguration(LoggerContext.java:579) {noformat} After further investigation, this scanning appears to be caused by this line in our log4j2-cli.xml file, which sets the packages to be "org.apache.geode" This causes log4j2 to scan all of our jar files. {code} <Configuration status="WARN" shutdownHook="disable" packages="org.apache.geode"> {code} This scanning is completely unnecessary because log4j already has an annotation processor which generates a file that log4j2 can find without any scanning. This probably affects geode server startup, not just gfsh, because log4j2.xml has this setting as well.
label: code-design

3. **summary:** Geode is spending a lot of time in log4j classpath scanning on startup
description: When running gfsh, it takes a long time even to run the `gfsh help` command. For example, with our docker image it takes 10 seconds just to display the help from {code} docker run apachegeode/geode:latest time gfsh help {code} Doing some CPU sampling, it looks like the majority of the time is in log4j2 scanning for plugins. {noformat} TRACE 300879: (thread=200001)
java.lang.ClassLoader.defineClass1(ClassLoader.java:Unknown line)
java.lang.ClassLoader.defineClass(ClassLoader.java:763)
java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
java.net.URLClassLoader.access$100(URLClassLoader.java:74)
java.net.URLClassLoader$1.run(URLClassLoader.java:369)
java.net.URLClassLoader$1.run(URLClassLoader.java:363)
java.security.AccessController.doPrivileged(AccessController.java:Unknown line)
java.net.URLClassLoader.findClass(URLClassLoader.java:362)
java.lang.ClassLoader.loadClass(ClassLoader.java:424)
sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:349)
java.lang.ClassLoader.loadClass(ClassLoader.java:357)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.addIfMatching(ResolverUtil.java:449)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.loadImplementationsInJar(ResolverUtil.java:351)
org.apache.logging.log4j.core.config.plugins.util.ResolverUtil.findInPackage(ResolverUtil.java:233)
org.apache.logging.log4j.core.config.plugins.util.PluginRegistry.loadFromPackage(PluginRegistry.java:221)
org.apache.logging.log4j.core.config.plugins.util.PluginManager.collectPlugins(PluginManager.java:152)
org.apache.logging.log4j.core.config.AbstractConfiguration.initialize(AbstractConfiguration.java:224)
org.apache.logging.log4j.core.config.AbstractConfiguration.start(AbstractConfiguration.java:288)
org.apache.logging.log4j.core.LoggerContext.setConfiguration(LoggerContext.java:579) {noformat} After further investigation, this scanning appears to be caused by this line in our log4j2-cli.xml file, which sets the packages to be "org.apache.geode" This causes log4j2 to scan all of our jar files. {code} <Configuration status="WARN" shutdownHook="disable" packages="org.apache.geode"> {code} This scanning is completely unnecessary because log4j already has an annotation processor which generates a file that log4j2 can find without any scanning. This probably affects geode server startup, not just gfsh, because log4j2.xml has this setting as well.
label: code-design

jira_issues_comments:

1. upthewaterspout opened a new pull request #5957: URL: https://github.com/apache/geode/pull/5957 This element causes log4j to scan all geode jars on startup. This slows down the startup time of all geode processes, including gfsh. It is not necessary because log4j has an annotation processor to generate a config file that helps it find plugs that it needs. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
2. upthewaterspout merged pull request #5957: URL: https://github.com/apache/geode/pull/5957 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. Commit d058f0dd1f23a6339cd0860933e58e428cb9eea7 in geode's branch refs/heads/develop from Dan Smith [https://gitbox.apache.org/repos/asf?p=geode.git;h=d058f0d] GEODE-8875: Removing packages element from log4j2 configuration file This element causes log4j to scan all geode jars on startup. This slows down the startup time of all geode processes, including gfsh. It is not necessary because log4j has an annotation processor to generate a config file that helps it find plugs that it needs.
4. upthewaterspout merged pull request #5957: URL: https://github.com/apache/geode/pull/5957 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org