Item 50
git_comments:

1. suppress MavenModelInspection
2. Keep aligned with preqrequisite section below.

git_commits:

1. **summary:** This closes #2096
   message: This closes #2096

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [BEAM-1092] Shade commonly used libraries (e.g. Guava) to avoid class conflicts
   body: Be sure to do all of the following to help us incorporate your contribution quickly and easily: - [x] Make sure the PR title is formatted like: `[BEAM-<Jira issue #>] Description of pull request` - [x] Make sure tests pass via `mvn clean verify`. (Even better, enable Travis-CI on your fork and ensure the whole test matrix passes). - [x] Replace `<Jira issue #>` in the title with the actual Jira issue number, if there is one. - [ ] If this contribution is large, please file an Apache [Individual Contributor License Agreement](https://www.apache.org/licenses/icla.txt). ---

github_pulls_comments:

1. R: @davorbonaci @dhalperi CC: @francesperry @jbonofre
2. [![Coverage Status](https://coveralls.io/builds/10314239/badge)](https://coveralls.io/builds/10314239) Coverage decreased (-0.01%) to 69.303% when pulling **ff322937ffddcf3f3b2333534327ff2d11a164ab on aviemzur:shade-guava-generically** into **3d0fe853941ec4a6190ea7891d063674b5eef067 on apache:master**.
3. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/beam_PreCommit_Java_MavenInstall/7813/ --none--
4. We looked into this exact approach in the past, and rolled it back because `project.artifactId` is not a legal java package name (e.g., if it included `-`). Did you find a way to solve this?
5. (#744)
6. I can replace the `-` with `.` programatically using build helper maven plugin.
7. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/beam_PreCommit_Java_MavenInstall/7831/<h2>Build result: ABORTED</h2>[...truncated 833.54 KB...] ... 31 moreCaused by: java.lang.RuntimeException: The forked VM terminated without properly saying goodbye. VM crash or System.exit called?Command was /bin/sh -c cd /home/jenkins/jenkins-slave/workspace/beam_PreCommit_Java_MavenInstall@2/sdks/java/io/hbase && /usr/local/asfpackages/java/jdk1.8.0_121/jre/bin/java org.apache.maven.surefire.booter.ForkedBooter /home/jenkins/jenkins-slave/workspace/beam_PreCommit_Java_MavenInstall@2/sdks/java/io/hbase/target/surefire/surefire7450086805414423461tmp /home/jenkins/jenkins-slave/workspace/beam_PreCommit_Java_MavenInstall@2/sdks/java/io/hbase/target/surefire/surefire_83942865920716290877tmp at org.apache.maven.plugin.surefire.booterclient.ForkStarter.fork(ForkStarter.java:590) at org.apache.maven.plugin.surefire.booterclient.ForkStarter.fork(ForkStarter.java:460) at org.apache.maven.plugin.surefire.booterclient.ForkStarter.run(ForkStarter.java:229) at org.apache.maven.plugin.surefire.booterclient.ForkStarter.run(ForkStarter.java:201) at org.apache.maven.plugin.surefire.AbstractSurefireMojo.executeProvider(AbstractSurefireMojo.java:1026) at org.apache.maven.plugin.surefire.AbstractSurefireMojo.executeAfterPreconditionsChecked(AbstractSurefireMojo.java:862) at org.apache.maven.plugin.surefire.AbstractSurefireMojo.execute(AbstractSurefireMojo.java:755) at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo(DefaultBuildPluginManager.java:134) ... 32 more2017-02-24T22:25:19.043 [ERROR] 2017-02-24T22:25:19.043 [ERROR] Re-run Maven using the -X switch to enable full debug logging.2017-02-24T22:25:19.043 [ERROR] 2017-02-24T22:25:19.043 [ERROR] For more information about the errors and possible solutions, please read the following articles:2017-02-24T22:25:19.043 [ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/PluginExecutionException2017-02-24T22:25:19.043 [ERROR] 2017-02-24T22:25:19.043 [ERROR] After correcting the problems, you can resume the build with the command2017-02-24T22:25:19.043 [ERROR] mvn <goals> -rf :beam-sdks-java-io-hbaseBuild was abortedchannel stoppedSetting status of af0c88f43508c87052d0b9405102b6ff5b7b65c2 to FAILURE with url https://builds.apache.org/job/beam_PreCommit_Java_MavenInstall/7831/ and message: 'Build finished. 'Using context: Jenkins: Maven clean install --none--
8. Retest this please.
9. [![Coverage Status](https://coveralls.io/builds/10326524/badge)](https://coveralls.io/builds/10326524) Coverage increased (+0.01%) to 69.327% when pulling **af0c88f43508c87052d0b9405102b6ff5b7b65c2 on aviemzur:shade-guava-generically** into **3d0fe853941ec4a6190ea7891d063674b5eef067 on apache:master**.
10. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/beam_PreCommit_Java_MavenInstall/7857/ --none--
11. @dhalperi Addressed your comment in my latest commit. PTAL?
12. Re: testlib -- I did some GitHub blame/history diving and found JIRA https://issues.apache.org/jira/browse/BEAM-557 and https://github.com/apache/beam/pull/832 . Maybe @swegner has more context about which specific ways the Guava testlib is a

public dependency.

13. **body:** Re: general approach. I see two improvements: 1. Providing a simple, uniform way to name shaded artifacts by establishing a central rewritten `artifactId`. This is succeeding where #744 failed! 2. Providing a centralized way to ensure that Guava is not on the public API surface of any module. I see two downsides: 1. Extra executions for modules that don't depend on Guava. This will slow the build. 2. Modules that need to shade more than just Guava will have to override the shading config. They may actually get it wrong after that, so Guava might still be exposed. -- Thus, while this reduces the amount of config necessary in common cases, does guarantee solution. I would probably: * move the shading config back into individual modules. build perf + accuracy. But use the new `artifactId` trick to make the config more uniform. * find a better way to actually verify that the final output artifacts don't expose Guava. (out of scope for this PR.) This is in-line with prior PR comments. In otherwords, I would probably take improvement 1, but not 2. 1 is a clear win, but 2 has some drawbacks and doesn't quite guarantee success to justify those drawbacks IMO.
    label: code-design
14. Another good opinion to get would be @davorbonaci .
15. @dhalperi I understand the concerns you raise. I'll try to address them from my point of view. > 1. Extra executions for modules that don't depend on Guava. This will slow the build. Many of the modules already shade, so regarding slowing down of the build I’m not sure this concern is that pertinent if new way of shading is an improvement. > 2. Modules that need to shade more than just Guava will have to override the shading config. They may > actually get it wrong after that, so Guava might still be exposed. -- Thus, while this reduces the amount of config necessary in common cases, does guarantee solution. I agree that human error could cause Guava to be forgotten from shading configuration added to child modules, but is this different than today? Can this human error not happen now?
16. **body:** I'm not totally sold on the premise that we should shade Guava in every single module. A couple of reasons: * We now have ~30 modules. We'll probably have 50+ sometime this year. Guava is not that small in size -- bringing in so many copies is just wasteful for a simple call to, say, `Preconditions`. Minimization could mitigate this concern, but I'm yet-to-be-convinced it works well for libraries. * We care that Guava is not leaked to our users -- this is totally legit. On the other hand, we don't have a strong reason to avoid Guava on an internal interface between, say, `runners-core` and `core-sdk`. It just doesn't matter to users. Of course, taking a stronger stance can be fine, but I'm a little worried it will bite us (for no real reason). * Build time. Separately, the amount of "magic" worries me too. This will work perfectly in Maven, but what happens when someone imports it into Eclipse, IntelliJ or elsewhere? At least in some IDEs, it will fail. On the other hand, I'm really strongly in favor of one common configuration. There's literally zero chance that this will work if this configuration is offloaded to every single module. This one outweighs them all for me. (And, I was bitten by Guava 19 vs. 20 just today -- so, I know the pain.) In summary, I'm leaning towards +1 on this. However, I'd be more comfortable if we were explicit in saying: * how much the build time has gone up * how much the total bundled jar, across the project, has gone up * to investigate minimization (separately from this PR) * that we'll make enforcement that Guava in not brought in transitively (which is currently broken at least in one module).
    label: code-design
17. Build time (MacBook Pro, skipping tests) - **Before:** 02:13 min **After:** 02:35 min Jar size (Sum of all jars generated) - **Before:** 93.93 MB **After:** 151.34 MB
18. Build time -- less than I would have forecasted. SGTM. Jar size -- more than I would have forecasted -- it is 50% more. I think minimization is in order.
19. [![Coverage Status](https://coveralls.io/builds/10580212/badge)](https://coveralls.io/builds/10580212) Coverage decreased (-0.003%) to 70.14% when pulling **7eb64346005e31a31a52f6e616f8faff72cfe5d2 on aviemzur:shade-guava-generically** into **9ad01fb1501a5b40b80791d051c040a13224acf7 on apache:master**.
20. Refer to this link for build results (access rights to CI server needed): https://builds.apache.org/job/beam_PreCommit_Java_MavenInstall/8378/ --none--
21. * Rebased on top of master. * Added ServiceResourceTransformer to the common shading configuration to match what was done in #2182 If there are modules which do not need to shade and relocate Guava - for example if they actively need to not do this and/or we feel that their file size has inflated without any reason - we can add the following <plugin> node to their `<build> <plugins>` to "opt-out" of the default shading configuration: ```xml <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-shade-plugin</artifactId> <executions> <execution> <id>bundle-and-repackage</id> <phase>none</phase> </execution> </executions> </plugin> ``` Another way to tackle large file sizes would be to configure minimization by default. I think we should start a minimization track separately from this change, as there are some issues with it: 1. There is a bug in shade-plugin where if minimization is declared on a project with pom packaging it fails. I have created a [ticket](https://issues.apache.org/jira/browse/MSHADE-253) and [PR](https://github.com/apache/maven-plugins/pull/107) with a fix for this but could take a while before it is available to us. 2. Minimization on all of our modules adds a significant amount of build time, which, as discussed, we wish to avoid. (Up from 02:30 mins for a build skipping tests on my laptop to over 5 mins)
22. @aviemzur, btw, I noticed the version of the shade plugin went down. Was this intentional? If yes, great. If not, perhaps push a separate PR to get it back up.

github_pulls_reviews:

1. this looks like it's being lost. Is this right?
2. how do we ensure this only executes in the modules that use guava? What happens in the modules that need to shade more than guava?
3. same q?
4. All the tests pass without this, so I'm not sure why this is needed?
5. This will execute in every module, to ensure forward compatability, when new modules are introduced, they will not need to know they should shade guava and add this shading to the pom, this will happen automatically. (If they don't depend on guava, no extra classes will be shaded into the resulting jar). For modules that need to shade more than guava we should ask the following: Should this artifact be shaded in all modules that will use it? If so: add it to the root pom If not: override shading for that specific module.
6. This was added in PR #832 because the rules only shade `com.google.guava:guava` and not `com.google.guava:guava-testlib`. If we relocate symbols from `guava-testlib` without shading them, we can get runtime `ClassNotFoundException`s on projects

which depend on our artifacts. IIRC, this issue did not manifest within the the beam project structure because each project had equivalent repackaging rules and could source the repackaged `guava-testlib` symbols from its own local dependencies. I believe I ran into the issue when creating my own project that didn't have shading and consumed the shaded beam artifacts. I've been away from Beam development for some months so unfortunately I don't have recent context on whether this is still needed.
7. Thanks @swegner for your comment! All dependencies on `guava-testlib` in the project currently are in scope `test`. My change removes shading of test jars in modules which matched the shading rules that are in the root pom in my branch (They now do not shade their test jars) The only test jar that remains with shading is the one of Dataflow runner, which still has this exclusion in its shade-plugin configuration: [google-cloud-dataflow-java/pom.xml](https://github.com/aviemzur/beam/blob/shade-guava-generically/runners/google-cloud-dataflow-java/pom.xml#L138) Is this sufficient?
8. Sounds good to me; thanks for the thorough investigation! (Please continue working with @dhalperi for the rest of the PR)

jira_issues:

1. **summary:** Shade commonly used libraries (e.g. Guava) to avoid class conflicts
description: Beam shades away some of its dependencies like Guava to avoid user classes from clashing with these dependencies. Some of the artifacts, e.g. KafkaIO, do not shade any classes and directly depend on potentially conflicting libraries (e.g. Guava). Also, users might manually add such libraries as dependencies. Runners who add classes to the classpath (e.g. Hadoop) can run into conflict with multiple versions of the same class. To prevent that, we should adjust the Maven archetypes pom files used for the Quickstart to perform shading of commonly used libraries (again, Guava is often the culprit). To prevent the problem in the first place, we should expand the shading of Guava and other libraries to all modules which make use of these. To solve both dimensions of the issue, we need to address: 1. Adding shading of commonly used libraries to the archetypes poms 2. Properly shade all commonly used libraries in the SDK modules 2) seems to be of highest priority since it affects users who simply use the provided IO modules.

jira_issues_comments:

1. **body:** Is 1) necessary if we do 2)? I think shading is very necessary (unfortunately) so I like this issue but I would also like to keep the example/archetype POMs as simple as possible.
label: code-design
2. **body:** 2) is for avoiding user classes to clash with all SDK-related parts. 1) is for avoiding user classes to clash with other libraries added to the classpath which do not properly shade common libraries (e.g. Hadoop) The archetype does not become more complicated because of 1). It should be more or less be transparent to the user who uses the archetype.
label: code-design
3. Guava is one library that we should shade, but we have the same issue with Protobuf (with Hive or HBase for instance).
4. Maybe, more than maven-archetype we can have a "shade" module providing the shaded artifacts. Then, we use those artifacts even for our internal dependency.
5. Yes, Guava was just one example of a commonly used library which is backwards-incompatible.
6. Doesn't sound like a release blocker, so removing the Fix Versions field. This is a larger item, across all project modules -- we should definitely do this, but it requires some thought, I think.
7. [~davor] I agree that a general solution requires some thought. Regarding KafkaIO this should probably be shaded in its pom for now as this is commonly used in clusters which may have a different version of guava. I myself had my application crash in a Spark cluster due to KafkaIO not having its guava dependency shaded. While it is true that users can shade their applications themselves it might be too much to expect them to go through this process, figure out that they have a transient dependency on a different version of guava then shade it.
8. [~aviemzur] I strongly agree that we should move to aggressive shading where we can. Send a PR for KafkaIO module?
9. [~dhalperi@google.com] Sure. See: https://issues.apache.org/jira/browse/BEAM-1379 and https://github.com/apache/beam/pull/1906
10. GitHub user aviemzur opened a pull request: https://github.com/apache/beam/pull/2096 [BEAM-1092] Shade commonly used libraries (e.g. Guava) to avoid class conflicts Be sure to do all of the following to help us incorporate your contribution quickly and easily: - [ ] Make sure the PR title is formatted like: `[BEAM-<Jira issue #>] Description of pull request` - [ ] Make sure tests pass via `mvn clean verify`. (Even better, enable Travis-CI on your fork and ensure the whole test matrix passes). - [ ] Replace `<Jira issue #>` in the title with the actual Jira issue number, if there is one. - [ ] If this contribution is large, please file an Apache [Individual Contributor License Agreement](https://www.apache.org/licenses/icla.txt). --- You can merge this pull request into a Git repository by running: $ git pull https://github.com/aviemzur/beam shade-guava-generically Alternatively you can review and apply these changes as the patch at: https://github.com/apache/beam/pull/2096.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #2096 ---- commit ff322937ffddcf3f3b2333534327ff2d11a164ab Author: Aviem Zur <aviemzur@gmail.com> Date: 2017-02-24T12:42:27Z [BEAM-1092] Shade commonly used libraries (e.g. Guava) to avoid class conflicts ----
11. Github user asfgit closed the pull request at: https://github.com/apache/beam/pull/2096
12. GitHub user tgroh opened a pull request: https://github.com/apache/beam/pull/2256 [BEAM-1092] Rollback global shading of Guava Be sure to do all of the following to help us incorporate your contribution quickly and easily: - [ ] Make sure the PR title is formatted like: `[BEAM-<Jira issue #>] Description of pull request` - [ ] Make sure tests pass via `mvn clean verify`. (Even better, enable Travis-CI on your fork and ensure the whole test matrix passes). - [ ] Replace `<Jira issue #>` in the title with the actual Jira issue number, if there is one. - [ ] If this contribution is large, please file an Apache [Individual Contributor License Agreement](https://www.apache.org/licenses/icla.txt). --- Revert PRs #2096 and #2249 PR #2096 has caused BigtableITs to fail, as of https://builds.apache.org/job/beam_PostCommit_Java_MavenInstall/2906/ The `BigtableDataClient` has methods that return Guava types, e.g. `sampleRowKeys` returns an `ImmutableList`, and we rewrite those calls to reference the repackaged type, but we don't repackage the actual client libraries; as a result when the JVM tries to find the `sampleRowKeys(SampleRowKeysRequest) -> ImmutableList<SampleRowKeysResponse>` it fails. We can certainly not shade guava in the google-cloud-platform IOs module; I believe it is also possible to repackage the appropriate Bigtable libraries to ensure the return types of those methods are also repackaged. However, `BigtableIO` uses classes that are defined in the

`Bigtable` libraries, so we would have to force our version on users. @davorbonaci @aviemzur You can merge this pull request into a Git repository by running: $ git pull https://github.com/tgroh/beam no_repackage_gcp_io Alternatively you can review and apply these changes as the patch at: https://github.com/apache/beam/pull/2256.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #2256 ---- commit 506a2cfa0b730d9b1ec8ec07223797c697b061f1 Author: Thomas Groh <tgroh@google.com> Date: 2017-03-15T22:28:45Z Revert "Increment shade-plugin version back to 3.0.0" This reverts commit 8f7c320846b598dc650a7b786182f7cecdd9c601. commit dcfe32a14318d0daebad7319ba4143a25fc02be4 Author: Thomas Groh <tgroh@google.com> Date: 2017-03-15T22:28:47Z Revert "[BEAM-1092] Replace hyphens in artifactId with dots in shading relocations" This reverts commit 2b55e0798f44e3fa644e7df1eaab43898ecf9c46. commit 12873b9500c31b78d84ec6d52c0d9cf1a68edc39 Author: Thomas Groh <tgroh@google.com> Date: 2017-03-15T22:28:49Z Revert "[BEAM-1092] Shade commonly used libraries (e.g. Guava) to avoid class conflicts" This reverts commit 03d78097ba0a2d55a9e8dcbaf11e0f8182f223c1. ----

13. Github user asfgit closed the pull request at: https://github.com/apache/beam/pull/2256
14. IMO we've made Guava shaded everywhere by default, which is the biggest offender. This bug is a little too abstract to be left unresolved (when would we close it?), so I think I'll call it resolved. Thanks [~aviemzur]!