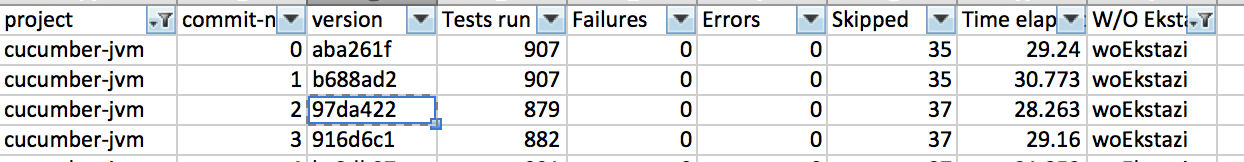
**SE Analysis**

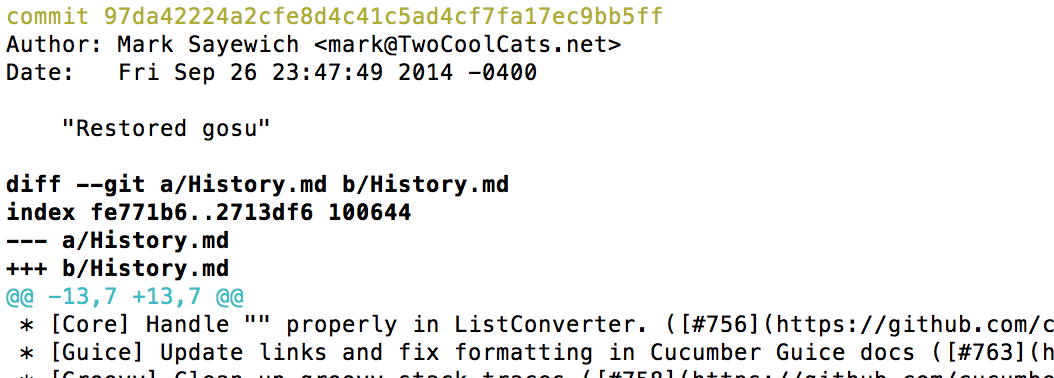
Execution Analysis for test without ekstazi

1. Cucumber-jvm



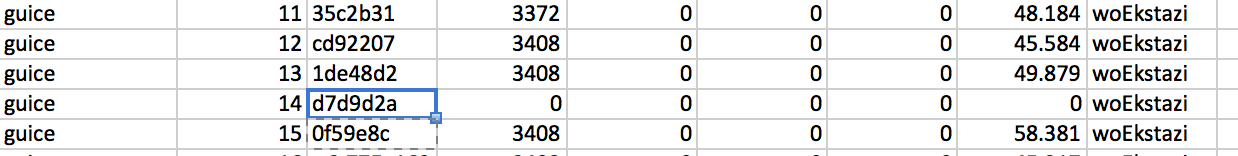
For verison **97da422** the number of tests run is 879, whereas for previous revision i.e. **b688ad2**, number of tests run are 907.

However, if we checkout **97da422** and execute **git log –p -2** to see the difference introduced, there are lot of tests and code added to 97d version. This is strange, as its running less test cases without ekstazi, whereas expected behavior is increase or same number of test case executions.

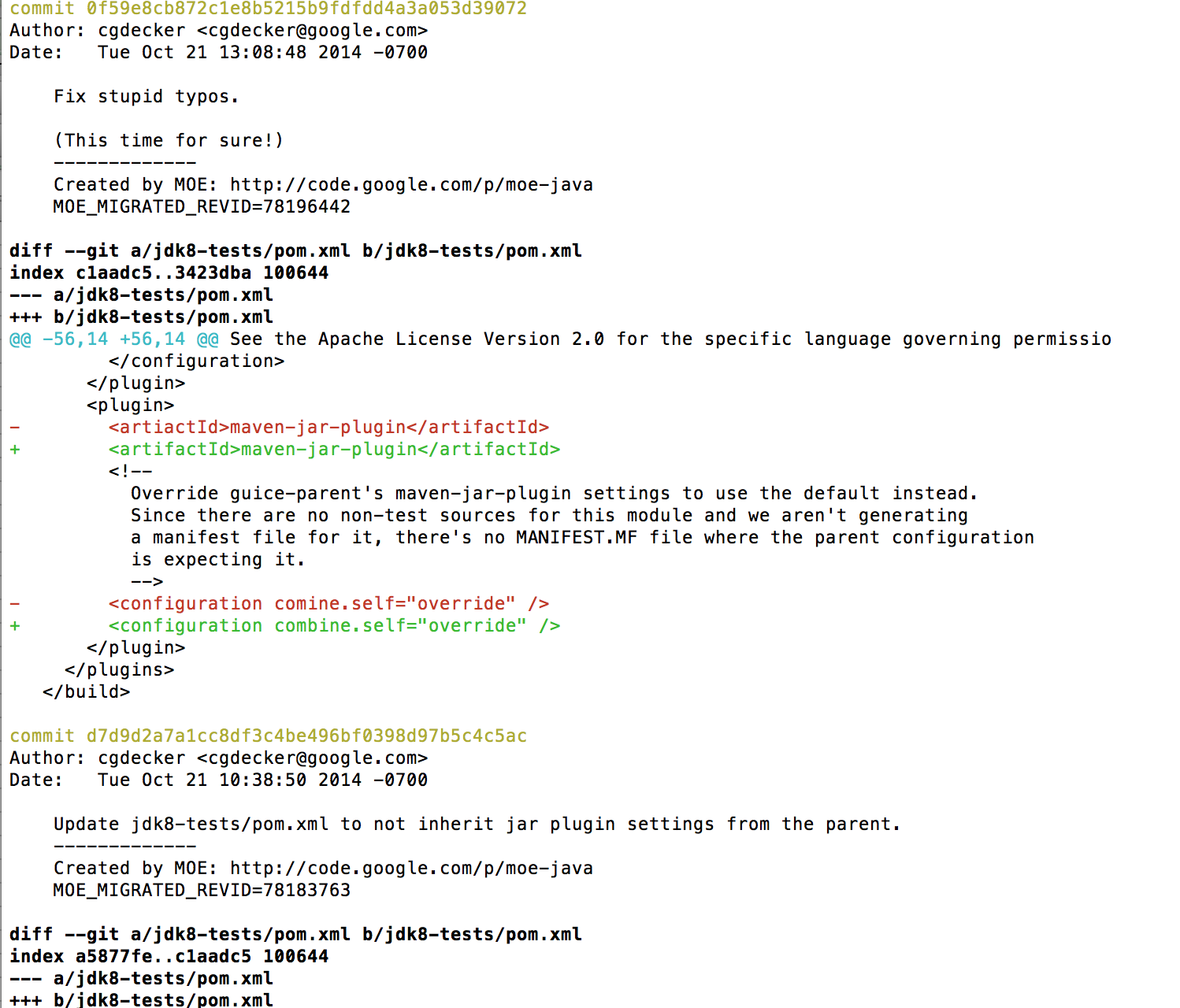


1. Guice

For revision **d7d9d2a**, number of tests run is 0, whereas for previous and next revision number of tests run are 3408.



When we checked the difference between 2 revisions, we found out that for this revision, they’ve updated jdk-8 tests to not inherit jar plugin settings



1. HttpClient

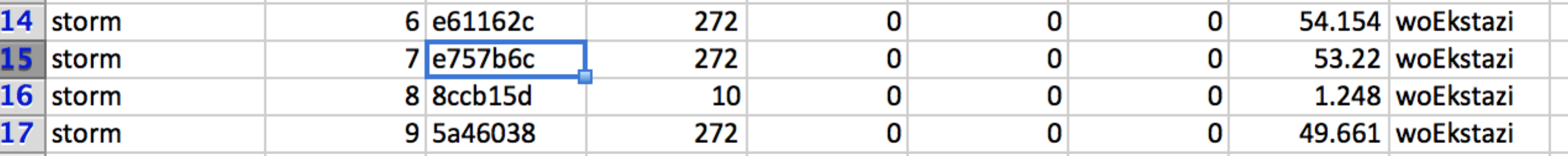
For revision 2c0bf9a the number of tests run are 821, but for revision 8a75863, number of tests run without Ekstazi are 1882. The reason was clear when we analyzed using git log –p -2, where for revision 8a75, more test cases were added compared to earlier version.

Also number of test cases for earlier version was again 1882, which is very high when compared to 821, this was the result of removing lot of import statements and huge number of test cases from rev 2c0bf.



4 . Apache Storm

In case of Storm for revision 8ccb15d, number of tests run are only 10, whereas for earlier revision e757b6c, the tests run are 272.



When we analyzed git log –p -2, we identified that this revision is result of merging of 2 branch and this might’ve caused the number of test cases run to reduce considerably.



**With Ekstazi:**

1. closure compiler

For revision 21a2f2a, the number of tests run are 7296, whereas for earlier version 0fc234b, the number of tests run is 0. We expected decrease in number of test cases run for rev 21a2f2, but it is increased.

When we checked out 21a2f2a and did git log –p -2, we got to know that more test code was added to this version and that might be the result for increase in number of tests run.

host-1-4:closure-compiler suhas$ git log -p -2

commit 21a2f2aaccd1a437edab9ff4297c913449cb76e8

Author: johnlenz <johnlenz@google.com>

Date: Sat Nov 1 16:16:05 2014 -0700

Add a suppression for bad goog.provide/goog.requires and make a better message.

-------------

Created by MOE: http://code.google.com/p/moe-java

MOE\_MIGRATED\_REVID=79004090

**diff --git a/src/com/google/javascript/jscomp/DiagnosticGroups.java b/src/com/google/javascript/jscomp/DiagnosticGroups.java**

**index ea94e35..b16575e 100644**

**--- a/src/com/google/javascript/jscomp/DiagnosticGroups.java**

**+++ b/src/com/google/javascript/jscomp/DiagnosticGroups.java**

@@ -343,6 +343,10 @@ public class DiagnosticGroups {

DiagnosticGroups.registerGroup("useOfGoogBase",

ProcessClosurePrimitives.USE\_OF\_GOOG\_BASE);

+ public static final DiagnosticGroup CLOSURE\_DEP\_METHOD\_USAGE\_CHECKS =

+ DiagnosticGroups.registerGroup("closureDepMethodUsageChecks",

+ ProcessClosurePrimitives.INVALID\_CLOSURE\_CALL\_ERROR);

+

// This group exists so that generated code can suppress these

// warnings. Not for general use. These diagnostics will most likely

// be moved to the suspiciousCode group.

**diff --git a/src/com/google/javascript/jscomp/ProcessClosurePrimitives.java b/src/com/google/javascript/jscomp/ProcessClosurePrimitives.java**

**index ac95d38..2afd594 100644**

**--- a/src/com/google/javascript/jscomp/ProcessClosurePrimitives.java**

**+++ b/src/com/google/javascript/jscomp/ProcessClosurePrimitives.java**

@@ -105,7 +105,7 @@ class ProcessClosurePrimitives extends AbstractPostOrderCallback

static final DiagnosticType INVALID\_CLOSURE\_CALL\_ERROR = DiagnosticType.error(

"JSC\_INVALID\_CLOSURE\_CALL\_ERROR",

- "Closure primitive methods must be called at file scope.");

+ "Closure dependency methods(goog.provide, goog.require, etc) must be called at file scope.");

static final DiagnosticType NON\_STRING\_PASSED\_TO\_SET\_CSS\_NAME\_MAPPING\_ERROR =

DiagnosticType.error(

@@ -231,7 +231,6 @@ class ProcessClosurePrimitives extends AbstractPostOrderCallback

public void visit(NodeTraversal t, Node n, Node parent) {

switch (n.getType()) {

case Token.CALL:

- boolean isExpr = parent.isExprResult();

Node left = n.getFirstChild();

if (left.isGetProp()) {

Node name = left.getFirstChild();

**diff --git a/src/com/google/javascript/jscomp/parsing/ParserConfig.properties b/src/com/google/javascript/jscomp/parsing/ParserConfig.properties**

**index 5a5a167..62ee6e9 100644**

**--- a/src/com/google/javascript/jscomp/parsing/ParserConfig.properties**

**+++ b/src/com/google/javascript/jscomp/parsing/ParserConfig.properties**

@@ -137,6 +137,7 @@ jsdoc.suppressions =\

checkStructDictInheritance,\

checkTypes,\

checkVars,\

+ closureDepMethodUsageChecks,\

const,\

constantProperty,\

deprecated,\

**diff --git a/test/com/google/javascript/jscomp/IntegrationTest.java b/test/com/google/javascript/jscomp/IntegrationTest.java**

**index 9bc84a3..b7b7f5f 100644**

**--- a/test/com/google/javascript/jscomp/IntegrationTest.java**

**+++ b/test/com/google/javascript/jscomp/IntegrationTest.java**

@@ -2372,6 +2372,19 @@ public class IntegrationTest extends IntegrationTestCase {

"a.b.c.myFunc = function(x) {};");

}

+ public void testSuppressBadGoogRequire() throws Exception {

+ CompilerOptions options = createCompilerOptions();

+ options.closurePass = true;

+ options.checkTypes = true;

+ test(

+ options,

+ "/\*\* @suppress {closureDepMethodUsageChecks} \*/\n" +

+ "function f() { goog.require('foo'); }\n" +

+ "f();",

+ "function f() { goog.require('foo'); }\n" +

+ "f();");

+ }

+

public void testIssue1204() {

CompilerOptions options = createCompilerOptions();

CompilationLevel.ADVANCED\_OPTIMIZATIONS

commit 0fc234b5fc10a7bb6545471f412ef99efe834149

Author: alusco <alusco@google.com>

Date: Fri Oct 31 17:11:27 2014 -0700

transition is a nullable promise on the $state object.

-------------

1. joda-time

For revision a25bf14, the number of tests run are 1232, whereas the expected behavior is test reduction. The following tree was observed when we did git log –graph –oneline

\* a25bf14 ZoneInfoProvider.getZone(String) throws a ClassCastException in some Android implementations of the String class.

\* 35a9f97 Merge pull request #99 from rowanhill/tiny-tidy-up

|\

| \* 39fff6f Tiny typo fixes

|/

\* 4058d72 Avoid another NPE in Partial.with

\* 9a62b06 Fix NPE in Partial.with()

\* 8612f9e Fix NPE in Partial.with()

\* 8d109fe Adding zero no longer changes the offset during DST overlap

\* 3ba9ba7 Prevent creation of invalid partials via Partial.with(DateTimeFieldType,int)

\* bcb0446 Fix NPE in DateTimeUtils.isContiguous with weird partials [#89]

\* 81a8184 DurationField is incorrect in complex DateTimeField instances

\* a61b3ff Simplify tests

when we did git log –p -2, we identified some code are added to this version. We feel this is the reason for increased tests execution

host-1-4:joda-time suhas$ git log -p -2

commit a25bf149a001e4b2d69e01f03376e698c760fc02

Author: Jorge Perez Alvarez <jperezalv@gmail.com>

Date: Tue Dec 31 13:34:49 2013 +0100

ZoneInfoProvider.getZone(String) throws a ClassCastException in some Android implementations of the String class.

String.equals(SoftReference<?>) throws a ClassCastException. Reordering the code checks first the instanceof case and only then calls the .equals() me

**diff --git a/src/main/java/org/joda/time/tz/ZoneInfoProvider.java b/src/main/java/org/joda/time/tz/ZoneInfoProvider.java**

**index 4558808..3e0e980 100644**

**--- a/src/main/java/org/joda/time/tz/ZoneInfoProvider.java**

**+++ b/src/main/java/org/joda/time/tz/ZoneInfoProvider.java**

@@ -141,11 +141,6 @@ public class ZoneInfoProvider implements Provider {

return null;

}

- if (id.equals(obj)) {

- // Load zone data for the first time.

- return loadZoneData(id);

- }

-

if (obj instanceof SoftReference<?>) {

@SuppressWarnings("unchecked")

SoftReference<DateTimeZone> ref = (SoftReference<DateTimeZone>) obj;

@@ -155,6 +150,9 @@ public class ZoneInfoProvider implements Provider {

}

// Reference cleared; load data again.

return loadZoneData(id);

+ } else if (id.equals(obj)) {

+ // Load zone data for the first time.

+ return loadZoneData(id);

}

// If this point is reached, mapping must link to another.

commit 35a9f97a047c9f7fe8499f1d45247c158d8c99b7

Merge: 4058d72 39fff6f

Author: Stephen Colebourne <scolebourne@joda.org>

Date: Thu Dec 19 16:43:15 2013 -0800

Merge pull request #99 from rowanhill/tiny-tidy-up

Tiny typo fixes

**ISSUES WITH SVN**

# CERTAIN ISSUES IN SUPPORTING SVN WITH MAVEN TEST { since we already have tested 4 SVN projects working good with ekstazi changes }

1) The SVN clone repo URL differs with GIT repo in such that all the projects when cloned create directory named trunk while the

http://svn.apache.org/repos/asf/commons/proper/collections/trunk,r1635351,,svn,commons-coll4

https://github.com/google/closure-compiler.git,65401150

2) We had to spend some time in analysing the equivalent code for git cleanup in SVN .

The SVN version of "svn cleanup" was found to differ in function compared to git version.

\* svn cleanup : only reverts the file changes which are in repo. Ignores the new files/folders created untouched.

Workaround is to look at svn status and clean up files through some script.

3) Major issue was that when we take the last N revisions given a base , we found some of the SHA's had the below problem.

svn: E160005: Target path '/commons/proper/collections/trunk' does not exist

It means that there was a change in a file in branch A which Subversion tried to merge into branch B, but the file does not exist in branch B, so it has nowhere to merge the change. So we could not come up with a sequeuce.