Contributing to a free software project

Experience with Hadoop

Grigori Rybkine

Laboratoire de l'Accélérateur Linéaire
Orsay

11èmes Journées Informatique IN2P3/IRFU
2 October 2018

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Apache Hadoop

- In the EventIndex project, we use the Apache Hadoop project software — a framework that allows for the distributed processing of large data sets.
- Hadoop includes the modules:
  - Hadoop Common
  - Hadoop Distributed File System (HDFS)
  - Hadoop YARN (Yet Another Resource Negotiator)
  - Hadoop MapReduce (A YARN-based system for parallel processing of large data sets).
- Hadoop Common supports basic file formats like `SequenceFile`, `MapFile` (two `SequenceFile`s — one with data, the other being an index for random data access). We use the latter to store and query the data.

EventIndex and Hadoop

- The space taken by the data is growing and we run out of space from time to time.
- I explored the different compression sub-formats of `SequenceFile` and found that the use of the most efficient of them (block compressed) could, in our case, reduce the size by a factor of 10.
- However, I also discovered that random access queries did not work on our files when block compressed.
- Thanks to the availability of the source code, I was able to fully investigate the problem and track it down to a bug in a `MapFile` method.
- Having corrected the issue and now using the patched version for EventIndex, I decided to submit the patch to the Hadoop project.

Build and test environment

- Hadoop uses Git as source code version control system.
- The easiest way to get a build and test environment with all the appropriate tools is by means of the provided Docker configuration.
  - This requires a version of docker installed and run as non-root.
- A script is provided that builds and runs the docker image.
- The build and run went off without a hitch except that the mounted source code tree was not accessible from the container — I sorted it out quite quickly, more on slide 11.
- In January 2018, the environment was
  - OS: Ubuntu 16.04.3 LTS
  - java: openjdk version "1.8.0_151"
  - mvn: Apache Maven 3.3.9 (build tool)

Contributing a fix

- File a bug report in JIRA, the project's bug-tracking system.
- Modify the source code by adding the fix.
  - Java code must be formatted according to Sun's conventions (with one exception)
  - Contributions must pass existing unit tests.
- New unit tests should be provided to demonstrate bugs and fixes. Hadoop uses JUnit v4 as test framework.
- Make sure that no new javac compiler warnings are introduced by your patch.
- Provide a patch by one of the following ways:
  - Create and attach a diff in ASF JIRA.
  - Create a pull request in GitHub.

Automated patch testing

- When you believe that your patch is ready to be committed, select the Submit Patch link on the issue's JIRA.
- Submitted patches will be automatically tested by Jenkins, the project's continuous integration engine.
- The Jenkins configuration uses Apache Yetus that
  - automatically checks new contributions against a variety of community accepted requirements
  - helps release managers generate release documentation based on the information provided by community issue trackers and source repositories
- Upon test completion, Jenkins/Yetus will add a success ("+1") or failure ("-1") message to your issue report in JIRA.

Further steps

- Once a "+1" comment is received from the automated patch testing system and a code reviewer has set the Reviewed flag on the issue's JIRA
  - a committer should then evaluate it within a few days and either commit it, or reject it with an explanation.
- Should your patch receive a "-1" from the Jenkins/Yetus testing
  - select the Cancel Patch on the issue's JIRA
  - upload a new patch with necessary fixes
  - select the Submit Patch link again.

Java code patch

genericqa added a comment - 10/Jan/18 15:51

-1 overall

Vote	Subsystem	Runtime	Comment
0	reexec	9m 51s	Docker mode activated.
			Prechecks
+1	@author	0m 0s	The patch does not contain any @author tags.
+1	test4tests	0m 0s	The patch appears to include 1 new or modified test files.
			trunk Compile Tests
+1	mvninstall	17m 14s	trunk passed
+1	compile	12m 49s	trunk passed
+1	checkstyle	0m 38s	trunk passed
+1	mvnsite	1m 5s	trunk passed
+1	shadedclient	11m 35s	branch has no errors when building and testing our client artifacts.
+1	findbugs	1m 27s	trunk passed
+1	javadoc	0m 52s	trunk passed
			Patch Compile Tests
+1	mvninstall	0m 43s	the patch passed
+1	compile	11m 50s	the patch passed
-1	javac	11m 50s	root generated 1 new + 1240 unchanged - 0 fixed = 1241 total (was 1240)
-0	checkstyle	0m 39s	hadoop-common-project/hadoop-common: The patch generated 17 new + 125 unchanged - 1 fixed = 142 total (was 126)
+1	mvnsite	1m 2s	the patch passed
-1	whitespace	0m 0s	The patch 24 line(s) with tabs.
+1	shadedclient	9m 40s	patch has no errors when building and testing our client artifacts.
+1	findbugs	1m 35s	the patch passed
+1	javadoc	0m 52s	the patch passed
			Other Tests
+1	unit	8m 11s	hadoop-common in the patch passed.
+1	asflicense	0m 33s	The patch does not generate ASF License warnings.
		90m 26s	

Test results for the first `MapFile.java` patch I provided in HADOOP-15151:

```
javac:
[WARNING] ... [deprecation] cleanup(Log,Closeable..
in IOUtils has been deprecated

checkstyle:
Line is longer than 80 characters (found 83).
[LineLength]
'if' construct must use '{}'s. [NeedBraces]
Name 'SIZE' must match pattern
'^[a-z][a-zA-Z0-9]*$'. [LocalFinalVariableName]
File contains tab characters (this is the first
instance). [FileTabCharacter]
'for' construct must use '{}'s. [NeedBraces]

whitespace:
File contains tab characters (instead of spaces).
```

Java code checkstyle test

The `checkstyle` test is done with the Checkstyle tool, also on Github, that

- helps programmers write Java code that adheres to a coding standard
- by default supports the Google Java Style Guide and Sun Code Conventions, but is highly configurable
- is integrated in the build process via maven-checkstyle-plugin

license: GNU LGPL v2.1

Shell script patch

genericqa added a comment - 01/Feb/18 14:21

-1 overall

Vote	Subsystem	Runtime	Comment
0	reexec	14m 52s	Docker mode activated.
			Prechecks
+1	@author	0m 0s	The patch does not contain any @author tags.
-1	test4tests	0m 0s	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.
			trunk Compile Tests
+1	mvninstall	16m 43s	trunk passed
-1	mvnsite	2m 0s	root in trunk failed.
+1	shadedclient	10m 6s	branch has no errors when building and testing our client artifacts.
			Patch Compile Tests
-1	mvnsite	5m 5s	root in the patch failed.
-1	shellcheck	0m 0s	The patch generated 4 new + 0 unchanged - 0 fixed = 4 total (was 0)
+1	shelldocs	0m 12s	There were no new shelldocs issues.
-1	whitespace	0m 0s	The patch 5 line(s) with tabs.
+1	shadedclient	10m 34s	patch has no errors when building and testing our client artifacts.
			Other Tests
-1	unit	20m 46s	root in the patch failed.
+1	asflicense	0m 25s	The patch does not generate ASF License warnings.
		81m 7s	

The other patch I submitted was for the `start-build-env.sh` script — to resolve the problem with the Docker mounted directories access on systems with SELinux[a] enabled mentioned on slide 5. As we can see on the left, the Jenkins/Yetus tests for HADOOP-15195 resulted in the `test4tests` failure which was due to new tests missing and also `shellcheck` failure with the error codes:

- SC2086: Double quote to prevent globbing and word splitting.
- SC2012: Use find instead of ls to better handle non-alphanumeric filenames.

[a] Security-Enhanced Linux

ShellCheck for shell code

The `shellcheck` test is done with the ShellCheck shell script analysis tool, also on Github, that points out and clarifies

- typical beginner's syntax issues that cause a shell to give cryptic error messages
- typical intermediate level semantic problems that cause a shell to behave strangely and counter-intuitively
- subtle caveats, corner cases and pitfalls that may cause an advanced user's otherwise working script to fail under future circumstances

license: GNU General Public License, version 3

Shell code testing

As for `test4tests`, the reviewer eventually asked to consider writing some bats unit tests for the new shell code.

- BATS stands for Bash Automated Testing System
  - Bats is a TAP-compliant testing framework for Bash.
  - It provides a simple way to verify that the UNIX programs you write behave as expected.
- TAP stands for Test Anything Protocol
- the new shell code bats tests results look like

```
1..2
ok 1 start-build-env.sh (Docker without z mount option)
ok 2 start-build-env.sh (Docker with z mount option)
```

Free and Open Source Software

- The Hadoop project software is released under Apache License 2.0
- This is a free software license according to the Free Software Definition:
  A program is free software if the program's users have the four essential freedoms:
  0. to run the program as you wish, for any purpose.
  1. to study how the program works, and change it so it does your computing as you wish.
  2. to redistribute copies so you can help your neighbor.
  3. to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes.

  Access to the source code is a precondition for freedom 1 and freedom 3.

Free and Open Source Software

In practice

- In our use of Hadoop, all the four freedoms turned out to be absolutely necessary.
- Free and Open source software describe almost the same category of software but see Richard Stallman's article on the differences in views.

Conclusions

- Hadoop is available under a free/open source software license, with source code accessible via a Git repository
- Hadoop has created an efficient infrastructure and stimulating atmosphere for project contributors with
  - easy access to full build and test environment based on Docker and a modern OS
  - use of modern compilers and build tools — Java 8, Maven 3.3
  - extensive use of unit testing and of advanced code quality assurance tools — JUnit and Checkstyle for Java code, BATS and Shellcheck for shell code
  - systematic and consistent use of an issue tracker — JIRA
  - automated contribution testing system — Yetus
  - expert and friendly contribution reviewers
- Contributors can learn from and adopt (some of) the Hadoop project practices and tools