Item 340
git_comments:

git_commits:

1. **summary:** Check Dataflow Job Status Before Terminate
   message: Check Dataflow Job Status Before Terminate This closes #951

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Check Dataflow Job Status Before Terminate
   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). --- Add job status check before canceling job in order to avoid cancel terminated job and throw exception.

github_pulls_comments:

1. +R: @peihe
2. PTAL: @peihe
3. LGTM R: @lukecwik to merge
4. PTAL @lukecwik
5. Its easier to review the diff if you make new commits instead of squashing, just mark them with !fixup at the beginning as per the Beam dev guide and they will be squashed when merging
6. LGTM, merging

github_pulls_reviews:

1. jobs.update() is not supposed to be called, right? could you add Mockito.verify()?
2. Yes, job status should return immediately after status check. But still need to verify Jobs.update() will not be called. done.
3. sorry, I meant: 1. remove the mock setup for update. 2. verify all other mocks are executed (also in testCancelUnterminatedJob()). 3. verify(mockJobs, never()).update(any(...), any(...), any(...));
4. thanks for clarify :) done.
5. Are you able to look at the failure reason in addition to checking job state? This way it closes a race condition where the job reaches a terminal condition between the fetching state and cancel calls
6. Do you mean that look at the IOException from execute() calls? Got confused how it will help to stop the race condition since the race condition is already happened. Or the "failure reason" means other messages?
7. It's possible that job status can be changed between fetching state and execute() calls. So one suggestion to avoid this race condition is to make cancel calls whatever job state is and fetching state when Exception is thrown. Return current state if it's terminated, otherwise throw exception.
8. Yes, look at the IOException, if it is a failure to cancel because the job is already in a terminal state then you know you don't have to rethrow and can just do the warn message.
9. done.
10. you can drop times(1), it is the default
11. I don't think this is needed for your test anymore since the state check is only done on failure now nor the mocking of job state higher above
12. I don't think you need this line for your test anymore
13. add verifyNoMoreInteractions(mockJobs)
14. add verifyNoMoreInteractions(mockJobs)
15. You shouldn't need this line anymore

16. Move this line to be after the execute(), this will prevent coding bugs where default fallthrough returns cancelled status
17. The concern of keeping it is to make sure the initial state of job is desired. But yes, state check will not be reached in this case. Remove it and the mocking of job state may be better.
18. done
19. thanks! done
20. see, will pay attention to it in future. done.
21. done
22. done
23. done
24. see. Thanks for pointed out. done
25. When using "when" clauses they are not strict and use functions like any(Job.class)/anyString only using something specific if a different result needs to be returned based upon an input parameter. The "verify" clauses should be the strict ones. This helps make tests not fail due to NPEs since a null would be returned if the mock call wasn't satisfied and will instead fail with a much cleaner message saying parameters X, Y, and Z were expected but got A, B, and C This applies to this test and the ones you added below
26. It's a very good explanation for "when" use case! I spend more time reading about "verify" and "when", and become more clear when/how to use them.
27. But "verify" can't be used to mock function return. How about using any(Job.class)/anyString in "when" clauses and verify the function parameters using "verify" with eq. I think this is what you suggest right?
28. Yes, that is what I was getting at that you need to use both when and verify. You should use when+any+anyString+... to setup what should be returned and verify+eq+... to ensure that the correct parameters were given.

jira_issues:

jira_issues_comments: