Arvados - Bug #14458
[controller] collection federation panic send on closed channel
11/07/2018 05:35 PM - Peter Amstutz

Status: Resolved  Priority: Normal  Assigned To: Peter Amstutz
% Done: 100%  Target version: 2018-11-14 Sprint

Description
2018-11-07_17:33:16.37829 ("RequestID":"req-tivf25vy461cnxx5be7","level":"info","msg":"response", "remoteAddr":"127.0.0.1:49982","reqBytes":0,"reqForwardedFor":"172.17.0.6","reqHost":"172.17.0.6:8000","reqMethod":"GET","reqPath":"/arvados/v1/collections/cba47aefe5eb3a014a26ec00316b30c1+57","req Query":"alt=json","respBytes":239,"respStatus":"OK","respStatusCode":200,"time":"2018-11-07T17:33:16.377898690Z","timeToStatus":0.258516,"timeTotal":0.258538,"timeWriteBody":0.000022}

2018-11-07_17:33:16.38499 panic: send on closed channel
2018-11-07_17:33:16.38501
2018-11-07_17:33:16.38502 goroutine 251 [running]:
2018-11-07_17:33:16.38502 git.curoverse.com/arvados.git/lib/controller.(*collectionFederatedRequestHandler).ServeHTTP.func1(0xc4200231a0, 0xc4201a6230, 0x8a4bc0, 0xc42022a4c0, 0xc420216040, 0xc420196070, 0xc420360240, 0xc4200283d8, 0x23, 0xc4203601e0, ...)

Subtasks:
Task # 14460: Review 14458-controller-panic  Resolved

Related issues:
Related to Arvados - Feature #14198: [CWL] run steps on remote clusters  Resolved  11/01/2018

Associated revisions
Revision 2bade90b - 11/08/2018 06:25 PM - Peter Amstutz
Merge branch '14458-controller-panic' refs #14458

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

History
#1 - 11/07/2018 05:35 PM - Peter Amstutz
- Status changed from New to In Progress

#2 - 11/07/2018 05:35 PM - Peter Amstutz
- Description updated

#3 - 11/07/2018 05:36 PM - Peter Amstutz
- Related to Feature #14198: [CWL] run steps on remote clusters added

#4 - 11/07/2018 08:30 PM - Peter Amstutz
- Assigned To set to Peter Amstutz

#5 - 11/07/2018 08:57 PM - Peter Amstutz
14458-controller-panic @ 9257e4c35fb335b3f57000371592db9f2206766

#6 - 11/07/2018 09:42 PM - Tom Clegg
With a buffered errorChan, it seems like wg.Done() and cancelFunc() can be called after the error is sent but before <-errorChan wins the select race, which means <-sharedContext.Done() can win the select race while the errors slice is empty, even though there is in fact an error to report.
With a buffered errorChan, it seems like wg.Done() and cancelFunc() can be called after the error is sent but before <-errorChan wins the select race, which means <-sharedContext.Done() can win the select race while the errors slice is empty, even though there is in fact an error to report.

Here's my reasoning:

1. wg.Wait() unblocks to call cancelFunc() only after all goroutines have completed.
2. At that point, every goroutine that is going to produce an error will have already put its error in the buffered channel.
3. In the loop, the receive on the error channel has priority over the cancellation channel.
4. Therefore, the error channel will be drained before matching the <-sharedContext.Done() case.

... so on further reading, I see that the select statement in Go don't actually have priority between channels, so it won't work like that. Argh!

(So the real reason I originally wrote this code using mutexes and shared state is that it was frankly easier for me to reason about than Go channels and all their edge cases)

You're right, it is simpler to keep them in the buffered channel and drain it when it is done.

Now 14458-controller-panic @ 6938e8cf1f1632d597cfbd333e44d5176805b628c6

With for range errChan, the behavior changes a bit: if the parent context cancels, the handler blocks until all of the remote-req goroutines notice that and return. (Previously, and still in the success case, the handler returns immediately, letting the remote-req goroutines run until they notice the cancelled context.) Is this deliberate? I suggested "for len(errChan) > 0" because I thought the "return immediately, let goroutines run" strategy seemed sensible to use across the board -- e.g., the "request finished" log message would happen right away.

LGTM

No, that wasn't deliberate, I wasn't thinking about the case of the parent context being cancelled. I see what you mean, draining the buffer but not waiting for it to close would allow it to return slightly earlier if there are outstanding requests.

LGTM

Changed range errorChan to len(errorChan) > 0 and merged.