Arvados - Bug #12446

[crunch2] crunch-dispatch-slurm monitoring too many containers gets 414 error

10/12/2017 02:08 PM - Peter Amstutz

Status:	Resolved	**Start date:**	10/17/2017
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2017-10-25 Sprint		

Description

It seems that the "tracked" list has gotten big enough that passing the list of UUIDs of being tracked is exceeding the request URI size. This brings crunch-dispatch-slurm to a screeching halt.

```
Oct 10 09:47:50 crunch-dispatch-slurm: 2017/10/10 09:47:50 Error getting list of containers: "arva
dos API server error: 414: 414 Request-URI Too Large returned by xxxxxxxxxxxx.com"
```

Possible solutions:

1) Use POST with method="get" so there is no limit on the query size

2) Adjust queries to avoid to avoid passing the "tracked" UUID list in the first place.

Related to: https://support.curoverse.com/rt/Ticket/Display.html?id=512

Subtasks:

Task # 12458: Review 12446-dispatcher-query	**Resolved**

Associated revisions

Revision b51d376e - 10/19/2017 03:41 PM - Peter Amstutz

Merge branch '12446-dispatcher-query' closes #12446

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

History

#1 - 10/12/2017 02:20 PM - Peter Amstutz

- Description updated

#2 - 10/12/2017 03:31 PM - Tom Morris

- Description updated

- Target version set to Arvados Future Sprints

#3 - 10/12/2017 03:32 PM - Tom Morris

Can you expand on this? What does "tracked" mean in this context? What is being tracked and why?

#4 - 10/12/2017 03:42 PM - Peter Amstutz

"tracked" here refers to the container records being managed by crunch-dispatch-slurm and their corresponding on slurm jobs. It queries the API server to get the latest status of "tracked" containers here:

https://dev.arvados.org/projects/arvados/repository/revisions/master/entry/sdk/go/dispatch/dispatch.go#L60

However, I'm not entirely sure why it is querying for specific containers rather than querying by some more general filter.

#5 - 10/12/2017 04:07 PM - Tom Clegg

Peter Amstutz wrote:

> However, I'm not entirely sure why it is querying for specific containers rather than querying by some more general filter.

Because it needs to know whether any of the containers it's currently tracking have changed so they no longer meet the criteria for tracking, i.e.,

wouldn't match such a filter.

Using POST should work. Another approach would be to check the "tracked" list in batches, and only the subset that doesn't already get returned by one of the filter-based queries.

#6 - 10/12/2017 04:15 PM - Tom Clegg

```
        d.checkForUpdates([][]interface{}{
            {"uuid", "in", tracked}})
        d.checkForUpdates([][]interface{}{
            {"locked_by_uuid", "=", d.auth.UUID},
            {"uuid", "not in", tracked}})
        d.checkForUpdates([][]interface{}{
            {"state", "=", Queued},
            {"priority", ">", "0"},
            {"uuid", "not in", tracked}})
```

maybe it would be better to do

1. filter by locked_by_uuid == d.auth.UUID
2. filter by uuid in {tracked minus the UUIDs returned in the previous query}, either via POST or in batches of 10?
3. filter by state = Queued, ignoring tracked UUIDs instead of asking api to filter them out

#7 - 10/12/2017 05:39 PM - Peter Amstutz

- Target version deleted (Arvados Future Sprints)

We start tracking containers when:

- They are Queued and we are able to take the lock
- The are Locked or Running and locked_by_uuid is this dispatcher.

We stop tracking containers when:

- The container goes into Queued, Cancelled or Complete state.

Identifying containers:

- locked_by_uuid=self to get Locked and Running containers.
- state=Queued && priority > 0 to get Queued containers
- When a container goes into Cancelled or Complete state, locked_by_uuid is set to nil, but even if it was left alone, we don't want to get the entire history of all containers ever.
- We do want to know if a currently Locked or Running container goes to Queued, Cancelled or Complete state.

Currently, on container completion, either

- We notice the job is gone from squeue. This involves making an extra API call to check the container state.
- We notice the container status channel is closed, which unconditionally attempts to cancel the job. (See https://dev.arvados.org/issues/11669)
- If a container priority drops to 0, we only need to take action if it is Locked or Running, which means it should be returned by a previous query.

Can we simply assume that we stop tracking any container that isn't returned by locked_by_uuid=self or state=Queued && priority > 0 ?

#8 - 10/12/2017 05:40 PM - Peter Amstutz

- Target version set to Arvados Future Sprints

#9 - 10/12/2017 05:49 PM - Tom Clegg

If we do something like note-6 then we only expect each container to appear in that query string one time, right? So "batches of 10" would only come up in cases like "lots of containers finish at once."

I think it is better to get the current state and behave accordingly. If the container is finished, we'll do the right thing and the relevant UUID will never be in another of these queries. If not, there's some kind of error which we should not conveniently avoid seeing.

#10 - 10/17/2017 02:04 PM - Peter Amstutz

- Status changed from New to In Progress

- Assigned To set to Peter Amstutz

- Target version changed from Arvados Future Sprints to 2017-10-25 Sprint

#11 - 10/17/2017 02:53 PM - Peter Amstutz

For review:

12446-dispatcher-query @ 32713b4b3c2e1685b79acb24059a5b817cf6cbfc

#12 - 10/18/2017 08:18 PM - Tom Clegg

Batching bugfix LGTM.

The "seen" stuff looks like it's fixing a different condition, where a dispatcher is tracking a container but even fetching that container by UUID doesn't find it. This seems like it avoids a memory/goroutine leak in this unexpected situation, which is good, but

- I don't think we should do this when checkForUpdates() hits "Error getting list of containers". As it stands, a transient network error would easily cause dispatch to close all trackers, which seems wrong: even if we resume tracking anyway, this would produce a bunch of logging noise.
- Apart from the transient network/API error case, this is an unexpected situation, so we should log a warning.

Putting the "seen" flag in the tracker itself seems weird -- it causes a lot of mutexing, and it's not usable by anyone other than the loop in Run() anyway because timing. How about returning a list of seen uuids from checkForUpdates, and keeping all the "missing" logic in one place, something like

```
Run() {
  todo := map[string]*runTracker
  for uuid, rt := range d.trackers {
    todo[uuid] = rt
  }
  for _, uuid := d.checkForUpdates(...) { delete(todo, uuid) }
  ...
  for uuid, t := range todo {
    // log something
    t.close()
  }
}
```

#13 - 10/19/2017 03:19 PM - Peter Amstutz

Tom Clegg wrote:

> Batching bugfix LGTM.
>
> The "seen" stuff looks like it's fixing a different condition, where a dispatcher is tracking a container but even fetching that container by UUID doesn't find it. This seems like it avoids a memory/goroutine leak in this unexpected situation, which is good, but
>
> - I don't think we should do this when checkForUpdates() hits "Error getting list of containers". As it stands, a transient network error would easily cause dispatch to close all trackers, which seems wrong: even if we resume tracking anyway, this would produce a bunch of logging noise.

Good point. Fixed.

> - Apart from the transient network/API error case, this is an unexpected situation, so we should log a warning.

Agree. Done.

> Putting the "seen" flag in the tracker itself seems weird -- it causes a lot of mutexing, and it's not usable by anyone other than the loop in Run() anyway because timing. How about returning a list of seen uuids from checkForUpdates, and keeping all the "missing" logic in one place, something like

I went with a slightly different approach of passing in the "todo" map and removing each UUID as it is seen in checkListForUpdates(). This is mainly because checkForUpdates would otherwise have to build a list of uuids, return it, and then Run() would have to iterate over it in several places, which overall seems like a lot of boilerplate to get to a single delete() operation.

12446-dispatcher-query @ 1b411c1882a37ae8f88c0b770994dd257cf3dee4

https://ci.curoverse.com/job/developer-run-tests/484/

#14 - 10/19/2017 03:26 PM - Tom Clegg

LGTM, thanks

#15 - 10/19/2017 03:45 PM - Anonymous

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:b51d376ed64efc68f7ee27fd061323da43faabd5.