Arvados - Bug #13108

arvados-cwl-runner dispatches container requests very slowly

02/21/2018 12:07 PM - Joshua Randall

Status:	Resolved	**Start date:**	04/06/2018
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-04-25 Sprint		

Description

I have observed that arvados-cwl-runner (a-c-r) seems to only be able to dispatch a container request once every 0.9s or so. We have a workflow in which at one point the workflow is scattered across both samples and genomic regions (this seems like a fairly standard way to split up work). For a variety of reasons, we are using 200 genomic intervals. At some point in our workflow we have n_samples*200 steps ready to run - for example, we are currently running a small test dataset of 147 samples against our GATK 4 pipeline, which results in 29400 invocations of HaplotypeCaller that are typically ready to run at around the same time.

My expectation would be that all 29400 container requests would be submitted within no more than a few minutes, and the subsequent containers and slurm jobs would then be scheduled, also within a few minutes.

What actually happens given the current performance characteristics of a-c-r is that it takes over 7 hours to submit all of the container requests. The result is that we cannot keep our compute nodes full of work, despite there being plenty of work to do.

Our current workaround for this is not attempt to run this workflow at all and instead to invoke a-c-r once per sample such that each has it's own RunnerContainer.

Subtasks:

Task # 13287: Parallel job submission	**Resolved**
Task # 13288: Review 13108-cwl-parallel-submit	**Resolved**
Task # 13350: Fix Docker image upload	**Resolved**
Task # 13357: Review 13108-acr-threading-fixes	**Resolved**
Task # 13375: Successful run on jenkins	**Resolved**

Related issues:

Related to Arvados - Bug #13351: Benchmark container_request creation and see...	**In Progress**

Associated revisions

Revision 20846db1 - 04/10/2018 07:38 PM - Peter Amstutz

Merge branch '13108-cwl-parallel-submit' closes #13108

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

Revision 605b7ae8 - 04/11/2018 08:21 PM - Tom Clegg

Merge branch '13348-recursive-submit'

refs #13348
refs #13108

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision b1160af5 - 04/13/2018 05:23 PM - Peter Amstutz

Merge branch '13108-acr-threading-fixes' refs #13108

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

Revision 8e8376ad - 04/17/2018 01:26 AM - Peter Amstutz

fix cwltest junit-xml dependency refs #13108

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

Revision c387fcfe - 04/18/2018 08:23 PM - Peter Amstutz

arv-put: restore_signal_handlers conditional on install_sig_handlers

arv-keepdocker: accept api object to use

refs #13108

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

Revision 6b7d586c - 04/18/2018 08:26 PM - Peter Amstutz

Bump arvados-python-client version dependency.

Set threadsafe api object to use calling command.keepdocker and
command.put.

refs #13108

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

History

#1 - 03/28/2018 03:52 PM - Tom Morris

- Target version set to 2018-04-11 Sprint

Perhaps investigate introducing concurrency.

#2 - 03/28/2018 04:21 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#3 - 04/02/2018 08:40 PM - Peter Amstutz

This create request has "duration":350.31,"view":0.34,"db":116.21.

This seems a little excessive. Increasing the number of submissions in flight might jam up the database without a significant impact on throughput.

Based on logs alone I can't account for all that latency.

```
{
  "method": "POST",
  "path": "/arvados/v1/container_requests",
  "format": "json",
  "controller": "Arvados::V1::ContainerRequestsController",
  "action": "create",
  "status": 200,
  "duration": 350.31,
  "view": 0.34,
  "db": 116.21,
  "request_id": "req-hc9pcexvfftojkmv4j70",
  "client_ipaddr": "172.17.0.1",
  "client_auth": "2tlax-gj3su-sq60vfd0xzi13wh",
  "params": {
    "scheduling_parameters": {},
    "runtime_constraints": {
      "vcpus": 1,
      "ram": 1073741824
    },
    "use_existing": true,
    "secret_mounts": {},
    "mounts": {
      "/tmp": {
        "kind": "tmp",
        "capacity": 1073741824
      },
      "/var/spool/cwl": {
        "kind": "tmp",
        "capacity": 1073741824
      }
    },
    "owner_uuid": null,
    "properties": {},
    "output_ttl": 0,
    "environment": {
      "HOME": "/var/spool/cwl",
      "TMPDIR": "/tmp"
    },
    "name": "step2_4",
    "priority": 500,
    "state": "Committed",
    "command": [
      "/bin/sh",
      "-c",
      "echo starting && sleep 194 && echo the number of the day is 4"
    ],
    "container_image": "arvados/jobs",
    "output_path": "/var/spool/cwl",
    "cwd": "/var/spool/cwl",
    "alt": "json",
    "container_request": {
      "owner_uuid": null,
      "name": "step2_4",
      "properties": {},
      "state": "Committed",
      "mounts": {
        "/tmp": {
          "kind": "tmp",
          "capacity": 1073741824
        },
        "/var/spool/cwl": {
          "kind": "tmp",
          "capacity": 1073741824
        }
      },
      "runtime_constraints": {
        "vcpus": 1,
        "ram": 1073741824
      },
      "container_image": "arvados/jobs",
      "environment": {
        "HOME": "/var/spool/cwl",
        "TMPDIR": "/tmp"
      },
      "cwd": "/var/spool/cwl",
      "command": [
        "/bin/sh",
        "-c",
        "echo starting && sleep 194 && echo the number of the day is 4"
      ],
      "output_path": "/var/spool/cwl",
      "priority": 500,
      "use_existing": true,
      "scheduling_parameters": {},
      "output_ttl": 0
    }
  },
  "@timestamp": "2018-04-02T20:26:45.095963550Z",
  "@version": "1",
  "message": "[200] POST /arvados/v1/container_requests (Arvados::V1::ContainerRequestsController#create)"
}
```

#4 - 04/05/2018 04:16 PM - Tom Clegg

While I also like the idea of making the "create" call faster, I think we should start by adding concurrency to arvados-cwl-runner.

- Adding concurrency to arvados-cwl-runner is necessary to support large workflows efficiently, no matter how fast we make the "create container request" operation. (When API server is working well, request turnaround time is limited by network latency and single-threaded Ruby code.)
- Limiting concurrency to 1 in each arvados-cwl-runner process is not an effective way to avoid overloading an API server with more requests than it can handle. This can only be addressed on the API side.
- a-c-r must behave well when API server is processing requests slowly -- whether that's because this a-c-r process is overloading API, or something else is overloading API.

Ideally API server would tell clients where to cap concurrent connections, so it could be site-configurable and/or dynamic in response to load, but a command-line option (defaulting to 1? 4? ncpus?) would be a good start.

#5 - 04/05/2018 04:29 PM - Tom Clegg

Even now, the timing figures in note-3 suggest we spend 2x as much time in Ruby as we do waiting for the DB, which seems to mean the database won't even be busy with one query at a time until we are doing at least 3 concurrent API requests. PostgreSQL is good at concurrent queries, so I expect even that example (dev?) site can do way more than 3 efficiently.

#6 - 04/05/2018 05:42 PM - Peter Amstutz

So the main reason I started looking into it was that on my development install, with a-c-r submitting a 100-way scatter of container requests with no parallelization, with "top" I was seeing four maxed out ruby processes.

But that's probably not representative of production, and clearly we want parallel queries to overcome other sources of latency. But I thought it was notable that transactions involving container requests take 3x-6x as long as transactions involving collections (for example).

I'll get started on parallel requests.

#7 - 04/05/2018 07:31 PM - Peter Amstutz

Notes to self:

- Need to use ThreadSafeApiCache
- Spin off ArvadosContainer.run() into its own thread (or queue or threadpool)
- Need to keep a "pending" counter of requests in flight, decrement when container uuid is entered into process table (add methods to ArvCwlRunner instead of changing processes directly)
- SecretStore is not threadsafe but it doesn't matter, secrets are only entered during initialization, after that it is read only
- CollectionFsAccess should be threadsafe via CollectionCache
- Need to take workflow evaluation lock (ArvCwlRunner.cond) around output_callback
- arv_docker_get_image needs to be support concurrent invocation without stepping on itself

#8 - 04/06/2018 05:39 PM - Peter Amstutz

13108-cwl-parallel-submit @ 091c2bc58ff1a8d1c43abf0e334837d8872b914d

https://ci.curoverse.com/job/developer-run-tests/681/

#9 - 04/09/2018 05:02 PM - Peter Amstutz

- Status changed from New to In Progress

#10 - 04/09/2018 07:40 PM - Lucas Di Pentima

Here're some comments/questions:

- File sdk/python/arvados/commands/keepdocker.py:
  - Lines 425-429: For the sake of having less lines of code, could we say something like parent_project_uuid = args.project_uuid or api.users().current().execute(…) ?
- File sdk/cwl/arvados_cwl/__init__.py:
  - Lines 168, 169: I think we could save 1 indentation level by putting all those conditionals on a single if statement
  - Lines 181, 182: Should those be inside the with: block? If not, why is it necessary to use a lock to just get a value from a dict?
  - Line 389: Why passing thread_count through cwltool/arv_executor instead of just using it at ArvCwlRunner instantiation?
  - Line 554: Small typo 'pendingjobs' on log message
- Would it be convenient to write some tests for TaskQueue? It's written as it would be useful to other modules as needed, so I think it would be nice to guarantee that's working as expected.
- File sdk/cwl/arvados_cwl/task_queue.py:
  - Line 45: Should task() be inside a try: except: block?
  - Line 51: I think that task_queue.empty() check is unnecessary, get() will still raise a Queue.Empty exception when it should.

#11 - 04/10/2018 06:28 PM - Peter Amstutz

Lucas Di Pentima wrote:

> Here're some comments/questions:
>
> - File sdk/python/arvados/commands/keepdocker.py:
>   - Lines 425-429: For the sake of having less lines of code, could we say something like parent_project_uuid = args.project_uuid or api.users().current().execute(…) ?

Done.

> - File sdk/cwl/arvados_cwl/__init__.py:
>   - Lines 168, 169: I think we could save 1 indentation level by putting all those conditionals on a single if statement

Done.

> - Lines 181, 182: Should those be inside the with: block? If not, why is it necessary to use a lock to just get a value from a dict?

The other lines don't need to be in the with: block. You're right it isn't technically necessary to get a lock to get a value from a dict, however updating the processes dict can occur together in the same critical section as other operations so it is better to be on the safe side.

- Line 389: Why passing thread_count through cwltool/arv_executor instead of just using it at ArvCwlRunner instantiation?

You're right, that's better. Done.

- Line 554: Small typo ‘pendingjobs’ on log message

Fixed.

- Would it be convenient to write some tests for TaskQueue? It’s written as it would be useful to other modules as needed, so I think it would be nice to guarantee that’s working as expected.

I added a couple of simple tests.

- File sdk/cwl/arvados_cwl/task_queue.py:
  - Line 45: Should task() be inside a try: except: block?

No. When task() is called synchronously, if there is an unhandled exception, it should go to the caller.

- Line 51: I think that task_queue.empty() check is unnecessary, get() will still raise a Queue.Empty exception when it should.

No, when Queue.get() is called without block=False or a timeout, it won't raise an Empty() exception. However, it also seems that calling get() will raise the Empty exception in any case that it can't get the queue lock, not just when the queue is actually empty, so getting the Empty exception doesn't guarantee that the queue is really empty. So we still want to explicitly check if it is empty.

However it is also possible that something could sweep in a grab the last item between checking empty() calling get() so I added a timeout.

13108-cwl-parallel-submit @ 138fef8ee97f3cbd335434ad6acd26771fd0b762

#12 - 04/10/2018 07:35 PM - Lucas Di Pentima

This LGTM. Thanks!

#13 - 04/10/2018 07:46 PM - Peter Amstutz

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

Applied in changeset arvados|20846db140dbba8a688718716e1e0f99ccfb3b51.

#14 - 04/11/2018 03:24 PM - Peter Amstutz

- Status changed from Resolved to Feedback

- Target version changed from 2018-04-11 Sprint to 2018-04-25 Sprint

#15 - 04/13/2018 02:31 AM - Peter Amstutz

13108-acr-threading-fixes @ 19da21ab8e56154d7db15c2643524cb8348a7a8a

- crunch_script.py initializes ArvCwlRunner with ThreadSafeApiCache
- Fix "Runner.done" method to use arvrunner.process_done (which has correct locking)
- ArvadosJob.update_pipeline_component takes lock to avoid update races
- Don't try to install signal handler on background when invoking arv-put via arv-keepdocker
- Do install signal handler in main thread to convert SIGTERM into KeyboardInterrupt

https://ci.curoverse.com/job/developer-run-tests/685/

#16 - 04/13/2018 01:32 PM - Tom Morris

- Status changed from Feedback to In Progress

#17 - 04/13/2018 02:05 PM - Lucas Di Pentima

This LGTM, the only detail that may or may not be an issue is that arv-put logs a stack trace when receiving signals and it assumes that KeyboardInterrupt is "translated" to SystemExit (put.py lines 576-588)

#18 - 04/13/2018 04:05 PM - Peter Amstutz

Lucas Di Pentima wrote:

> This LGTM, the only detail that may or may not be an issue is that arv-put logs a stack trace when receiving signals and it assumes that KeyboardInterrupt is "translated" to SystemExit (put.py lines 576-588)

I refactored things a bit, so arv-put and a-c-r use the same code to install the signal handler.

13108-acr-threading-fixes @ 8e3adbcf390deaffed7f2449056959252e1a49f4

https://ci.curoverse.com/job/developer-run-tests/686/

#19 - 04/13/2018 05:04 PM - Lucas Di Pentima

This LGTM, thanks!

#20 - 04/13/2018 08:23 PM - Peter Amstutz

20-way scatter

disable reuse	synchronous submit	8s
disable reuse	asynchronous submit	3s
enable reuse	synchronous submit	4s
enable reuse	asynchronous submit	3s

100-way scatter

enable reuse	synchronous submit	16s
enable reuse	asynchronous submit	13s

note: about 6s of that is the workflow engine constructing each scatter job. This isn't parallelized, only the submission to Arvados.

#21 - 04/13/2018 11:09 PM - Peter Amstutz

- Related to Bug #13351: Benchmark container_request creation and see if there are opportunities for optimization added

#22 - 04/25/2018 02:53 PM - Peter Amstutz

- Status changed from In Progress to Resolved

#23 - 07/23/2018 06:52 PM - Tom Morris

- Release set to 13