# Arvados - Story #11580

# [Workbench] Improve format of /container_requests index page

04/27/2017 02:25 PM - Radhika Chippada

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Start date:** | 05/16/2017 |
| **Priority:** | Normal | **Due date:** | |
| **Assigned To:** | Radhika Chippada | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | 2017-05-24 sprint | | |

**Description**

**Subtasks:**

| | |
|---|---|
| Task # 11667: Review 11580-container-requests | **Resolved** |
| Task # 11697: Review 11580-cr-display-perf | **Closed** |

**Related issues:**

| | | |
|---|---|---|
| Related to Arvados - Story #11710: [Workbench][Performance] container / conta... | **Resolved** | **05/18/2017** |

## Associated revisions

**Revision 6e5079ff - 05/17/2017 06:28 PM - Radhika Chippada**

closes #11580
Merge branch '11580-container-requests'

## History

**#1 - 05/02/2017 02:32 PM - Radhika Chippada**

*- Story points set to 1.0*

- Display the columns show icon, name, description, status, created_at (sort by this column).
- Optionally, include clickable workflow template link (if present), runtime_constraints, inputs from mount (if present), workflow.json from mounts (if it is a collection), outputs, log, and command by refactoring / reusing code from work_units display
- Include search box to limit search to container_requests only.
- Performance issue. This is page is getting the links, collections, and containers one at a time and hence takes a long to load. Get these in batches.

**#2 - 05/02/2017 06:50 PM - Radhika Chippada**

*- Target version set to Arvados Future Sprints*

**#3 - 05/02/2017 06:51 PM - Radhika Chippada**

*- Target version changed from Arvados Future Sprints to 2017-05-24 sprint*

**#4 - 05/10/2017 07:29 PM - Radhika Chippada**

*- Assigned To set to Radhika Chippada*

**#5 - 05/11/2017 03:13 PM - Radhika Chippada**

*- Status changed from New to In Progress*

**#6 - 05/12/2017 01:23 PM - Radhika Chippada**

Tom, in the container_work_unit children method, we are creating CR work_unit objects and (if present) the container for each child's container_uuid is being fetched one at a time. This is the first / low hanging fix I am aiming at.

I am trying to preload all the containers for the container_uuid's of all children and use them. However, this is not working because the object_for_dataclass from ApplicationController is not accessible from within the work unit.

Below the error log and my thoughts about a potential fix. You can see this in branch 10645-cr-display-perf @ 3fbba8072b4bf37a8b3e90ae60e2a79a77fee41e

The ApplicationController.helpers is an instance of ActionView::Base and hence not finding this method @ /home/radhika/arvados/apps/workbench/vendor/bundle/ruby/2.3.0/gems/actionpack-4.1.12/lib/abstract_controller/helpers.rb:70

```
App 3108 stdout: *** ApplicationController.helpers #<ActionView::Base:0x007ffb7e7d7ea0 @_config={}, @view_rend
erer=#<ActionView::Renderer:0x007ffb7e7d7c98 @lookup_context=#<ActionView::LookupContext:0x007ffb7e7d7e28 @det
ails_key=nil, @details={:locale=>[:en], :formats=>[:html, :text, :js, :css, :ics, :csv, :vcf, :png, :jpeg, :gi
f, :bmp, :tiff, :mpeg, :xml, :rss, :atom, :yaml, :multipart_form, :url_encoded_form, :json, :pdf, :zip], :vari
ants=>[], :handlers=>[:erb, :builder, :raw, :ruby, :coffee]}, @skip_default_locale=false, @cache=true, @prefix
es=[], @rendered_format=nil, @view_paths=#<ActionView::PathSet:0x007ffb7e7d7d60 @paths=[]>>>, @_assigns={}, @_
controller=nil, @view_flow=#<ActionView::OutputFlow:0x007ffb7e7d7c20 @content={}>, @output_buffer=nil, @virtua
l_path=nil>
```

Error log:

```
#<NoMethodError: undefined method `object_for_dataclass' for nil:NilClass>
/home/radhika/arvados/apps/workbench/vendor/bundle/ruby/2.3.0/gems/actionpack-4.1.12/lib/abstract_controller/h
elpers.rb:70:in `object_for_dataclass'
/home/radhika/arvados/apps/workbench/app/models/container_work_unit.rb:11:in `initialize'
/home/radhika/arvados/apps/workbench/app/models/container_request.rb:15:in `new'
/home/radhika/arvados/apps/workbench/app/models/container_request.rb:15:in `work_unit'
/home/radhika/arvados/apps/workbench/app/views/container_requests/_name_and_description.html.erb:2:in `_app_vi
ews_container_requests__name_and_description_html_erb__509120087467518462_43165140'
/home/radhika/arvados/apps/workbench/vendor/bundle/ruby/2.3.0/gems/actionview-4.1.12/lib/action_view/template.
rb:145:in `block in render'
/home/radhika/arvados/apps/workbench/vendor/bundle/ruby/2.3.0/gems/activesupport-4.1.12/lib/active_support/not
ifications.rb:161:in `instrument' ...
```

Code at /home/radhika/arvados/apps/workbench/vendor/bundle/ruby/2.3.0/gems/actionpack-4.1.12/lib/abstract_controller/helpers.rb:70 (controller.send(...))

```
      def helper_method(*meths)
        meths.flatten!
        self._helper_methods += meths

        meths.each do |meth|
          _helpers.class_eval <<-ruby_eval, __FILE__, __LINE__ + 1
            def #{meth}(*args, &blk)                               # def current_user(*args, &blk)
              controller.send(%(#{meth}), *args, &blk)             #   controller.send(:current_user, *args, &
blk)
            end                                                    # end
          ruby_eval
        end
      end
```

**Possible solution:** It appears that we cannot access ApplicationController helper methods from a work_unit (or a model class), but the methods from ApplicationHelper are accessible (please see proxy_work_unit -> show_runtime etc). Can we move all preloading related helper methods code out of ApplicationController into ApplicationHelper (I did not yet look deeply into any issues in moving this code).

#### #7 - 05/12/2017 06:15 PM - Tom Clegg

If we move the preload stuff out of ApplicationController then instance variables won't be scoped to the current request, so we will need to find somewhere else to store the data.

Perhaps it would make sense to connect the "preload" mechanism with the Rails.cache-based mechanism in source:apps/workbench/app/models/arvados_base.rb, so once the controller calls "preload_objects_for_dataclass", the cached objects are also accessible via WhateverClass.find(uuid). Something like this?

```
--- a/apps/workbench/app/controllers/application_controller.rb
+++ b/apps/workbench/app/controllers/application_controller.rb
@@ -1234,8 +1234,16 @@ class ApplicationController < ActionController::Base
          @objects_for[obj.name] = obj
        end
      else
+       key_prefix = "request_#{Thread.current.object_id}_#{dataclass.to_s}_"
        dataclass.where(uuid: uuids).each do |obj|
          @objects_for[obj.uuid] = obj
+         if dataclass == Collection
+           # The collecions#index API defaults to "select everything
+           # except manifest_text" so obj isn't suitable for preloading
+           # the find() cache.
+         else
+           Rails.cache.write(key_prefix + obj.uuid, obj)
+         end
        end
      end
```

```
     @objects_for
```

This logic could potentially move to ArvadosResourceList itself, so "preload" merely has to call where(...).to_a and everything ends up in the find() cache. This would mean checking "select" (and other stuff?) which can make the "index" results look different than the "find" results. The above change is less likely to create subtle problems elsewhere, and might solve the issue at hand.

Of course, to make use of the find() cache, we'd also need something like

```
--- a/apps/workbench/app/models/container_work_unit.rb
+++ b/apps/workbench/app/models/container_work_unit.rb
@@ -6,7 +6,7 @@ class ContainerWorkUnit < ProxyWorkUnit
     if @proxied.is_a?(ContainerRequest)
       container_uuid = get(:container_uuid)
       if container_uuid
-        @container = Container.where(uuid: container_uuid).first
+        @container = Container.find?(container_uuid)
       end
     end
   end
```

**#8 - 05/16/2017 12:05 AM - Radhika Chippada**

Branch 11580-cr-display-perf @commit:1d45ac3d3

- Preloads containers of the main object being displayed. This now results in not fetching each child container object individually. For qr1hi-dz642-uypqs2exiy7so7l, this resulted in about 40% improvement and it now takes about 35s (compared to earlier 65s)

- Major portion of the remaining time is being spent in computing the cputime (which loads children of the children etc). For container_work_unit => children method uses "ContainerRequest.where(requesting_container_uuid: container_uuid).results" to fetch the children of any given C or CR (at a given depth). There doesn't seem to be any easier way to "preload" all grandchildren at a level. We may want to consider an alternative such as:
  - While the object is not yet in Final state (still running), do not compute the cputime and display the message of the sort "It ran for 2d15h9m and used 29d17h55m of node allocation time (11.3⨯ scaling)"
  - When the object is in Final state, compute it once and store it in properties and use it the next time around. Alternatively, we may want to add this on the server side.

**#9 - 05/16/2017 02:57 PM - Tom Clegg**

The addition to _show_status.html doesn't seem right to me. It looks like it's trying to preload the children of the current container in a batch. But the children() method of ContainerWorkUnit already seems to do this query as a batch by itself -- without requiring the view to predict whether loading children will be necessary.

It seems to me we should do something like

- Get a page worth of container request records with one API call
- Preload all of their corresponding container records (just the top level) with one API call
- Display the page
- If there's anything we can't display without fetching the containers' descendants (like total node allocation time), don't display that on the index page.

65 to 35 seconds is an improvement, but users should expect <1 second.

**#10 - 05/16/2017 07:40 PM - Radhika Chippada**

> It seems to me we should do something like
>
> > Get a page worth of container request records with one API call
> > Preload all of their corresponding container records (just the top level) with one API call
> > Display the page
> > If there's anything we can't display without fetching the containers' descendants (like total node allocation time), don't display that on the index page.

I created #11710 to address #show page performance separate from index page display.

**#11 - 05/16/2017 08:31 PM - Lucas Di Pentima**

*- File Captura de pantalla 2017-05-16 17.25.18.png added*

Reviewed 701b7fa87

- File apps/workbench/app/views/container_requests/_show_recent.html.erb:L18: Adding a space “ContainerRequest" => "Container Request”

> would improve readability.
- When scrolling the index page on qr1hi, lots of "BUG: access non-loaded attribute updated_at" lines are being logged. Is that a real warning?
- When scrolling, it seems that the new pages don't keep the created_at ordering (attached screen capture)

**#12 - 05/17/2017 01:15 PM - Radhika Chippada**

*- File pipeline_index_sort_order.png added*

> When scrolling, it seems that the new pages don't keep the created_at ordering (attached screen capture)

I see this with /pipeline_instances page as well. Seems to happen only once with the first "page". The first page is not sending any filters to API server and hence it does not actually seem to get data sorted by created_at for the first page. (Instead we are sorting during display).  I think this is the case with all our index pages.

```
API client: 0.000152568 Prepare request POST https://qr1hi.arvadosapi.com/arvados/v1/pipeline_instances  {"api
_token"=>"43hmc16qq1ejm9cp7tvdnxcwff19i0ich92thspdwdcmmhlnye", "reader_tokens"=>"[false]", "current_request_id
"=>"1495026533-776268497", :_method=>"GET", :where=>"{\"created_by\":\"qr1hi-tpzed-ktpvhqu89qoib9f
\"}", :limit=>"1", :offset=>"0", "_profile"
```

I think we need something as follows:

```
   def find_objects_for_index
+    @filters << ['created_at', '>', Time.now - 100*365*24*60*60] if @filters.empty?
     @objects ||= model_class
     @objects = @objects.filter(@filters).limit(@limit).offset(@offset)
     @objects.fetch_multiple_pages(false)
```

**#13 - 05/17/2017 01:59 PM - Radhika Chippada**

> File apps/workbench/app/views/container_requests/_show_recent.html.erb:L18: Adding a space "ContainerRequest" => "Container Request" would improve readability.

Renamed it "Container request", similar to "Created at"

> When scrolling the index page on qr1hi, lots of "BUG: access non-loaded attribute updated_at" lines are being logged. Is that a real warning?

I had been seeing these for sometime now and couldn't yet find out what if anything we should do. I will create a new ticket for this.
http://stackoverflow.com/questions/12139174/some-data-displayed-as-not-loaded-in-rails-console

I also noticed and remembered that we should use a smaller @limit size for this page. We use @limit of 20 for pipeline instances and all_processes pages. Hence, I reduced this to 20 for this page as well and it loads much faster now

> When scrolling, it seems that the new pages don't keep the created_at ordering (attached screen capture)

Seems like we have a bug on API server? I created #11716 for this.

Thanks.

**#14 - 05/17/2017 02:11 PM - Lucas Di Pentima**

LGTM, thanks!

**#15 - 05/17/2017 06:35 PM - Radhika Chippada**

*- Status changed from In Progress to Resolved*

*- % Done changed from 0 to 100*

Applied in changeset arvados|commit:6e5079ff9a0349c57ab2cf06398413018a921cc0.

## Files

| | | | |
|---|---|---|---|
| Captura de pantalla 2017-05-16 17.25.18.png | 93.6 KB | 05/16/2017 | Lucas Di Pentima |
| pipeline_index_sort_order.png | 132 KB | 05/17/2017 | Radhika Chippada |