Pulp - Story #6858

As a user I can track progress of the task group with a task group progress report

05/29/2020 02:52 PM - ipanova@redhat.com

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	ipanova@redhat.com	**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:	3.5.0		
Platform Release:		**Tags:**	
Groomed:	Yes	**Sprint:**	Sprint 76
Sprint Candidate:	No	**Quarter:**	

Description

GroupProgressReport will be a similar model as ProgressReport.

Each group progress report will have a message, code, done, total, and relation to the TaskGroup. Plugin writers will create these objects to show progress of work that is expected to be completed by the tasks in the group.

Tasks that belong to the TaskGroup will update the progress report. All group progress reports needs to be created in advance so a Task can find the appropriate one, by code or message and update it. Tasks will need to handle logic and figure out what exactly they need to update in the group progress report. For example a task in the migration plugin called complex repo migration will create a repo version, publication and distribution. That means, the task will update 3 group progress reports.

To avoid race conditions / cache invalidation issues, this pattern needs to be used so that operations are performed directly inside the database

```
.update(done=F('done') + 1)
```

See: https://docs.djangoproject.com/en/3.0/ref/models/expressions/#f-expressions

Question: How do we figure out 'total' per each group report?

==========

~~Alternative Solution (a modification of the earlier "progressreport aggregation" strategy): on the TaskGroup serializer add another field called progress_report. It will query the db and look for tasks that belong to the group and aggregate the results by task 'code'. For example group will have 4 syncing tasks, each task has code 'sync', in the aggregated report there will be total of 4 syncing repos and based on the each tasks status the done will be calculated. This implementation is limited to the name of the task, and it does not have that much flexibility in case tasks creates more resources. For example: a task in the migration plugin called complex repo migration will create a repo version, publication and distribution. However in the progress report field it will only record that '1 complex repo migration done'.~~

Related issues:

Blocks Migration Plugin - Story #6769: As a user, I can track the progress of...	**CLOSED - CURRENTRELEASE**

Associated revisions

Revision 104aed7b - 06/23/2020 08:33 PM - ipanova@redhat.com

Add GroupProgressReport model and serializer.

closes #6858

https://pulp.plan.io/issues/6858

History

#1 - 05/29/2020 02:52 PM - ipanova@redhat.com

- Description updated

#2 - 05/29/2020 03:25 PM - ipanova@redhat.com

- Description updated

#3 - 05/29/2020 04:06 PM - ipanova@redhat.com

- Description updated

#4 - 05/29/2020 04:10 PM - ipanova@redhat.com

- Description updated

#5 - 05/29/2020 05:01 PM - dalley

- Description updated

#6 - 05/29/2020 10:20 PM - bmbouter

I believe users have needs to understand the overall progress of a group of tasks and distinctly see the progress of a single task. To the extent that is true, the design to have TaskGroupProgress be totally new objects that don't aggregate will force plugin writers to intentionally think about what progress is reported at each level. This is an application of the explicit is better than implicit design principal. The aggregation design would work in most cases, and be easier for plugin writers, but I don't think as strong for users.

For the question on figuring out 'total', generally the plugin writer sets it based on the workload they understand. I don't expect GroupProgressReport to make this race condition free in all cases. Here are three scenarios I think about regarding how various workloads could handle this depending on various needs.

- There is no race condition on 'total'. The 'total' is known at some point and is only ever set once. No one but the user ever reads it.
- There is only a write-read race condition on 'total'. In this case the writer sets it, calls save() and other tasks that read it, they can't be guaranteed 'total' is up to date unless it's set once. In the case it's set multiple times, they would need to have some sort of synchronization but TaskGroupProgress would not handle this for them in any way. I think this is also unlikely to be needed.
- There is a write-write race condition. In this case multiple processes are writing to 'total' based on portions of the work they are discovering. In this case the F() values are the way. Or someone can use a database transaction and handle the transaction-failed errors when one saves and the other doesn't.

Overall I don't think TaskGroupProgress needs to provide much except a F() based implementation for implementing the 'done' count because that's the one that is likeliest to be incremented across multiple processes.

#7 - 06/04/2020 03:41 PM - daviddavis

The TaskGroupProgress seems reasonable to me. I'm guessing we'll probably have to use F() to update totals.

> The 'total' is known at some point and is only ever set once.

Could you give more information about how this would work?

#8 - 06/05/2020 06:02 PM - bmbouter

daviddavis wrote:

> The TaskGroupProgress seems reasonable to me. I'm guessing we'll probably have to use F() to update totals.
>
> > The 'total' is known at some point and is only ever set once.
>
> Could you give more information about how this would work?

Sure. The import/export example I think is this case actually. IIRC, for imports, the first task to run reads the archive to import and determines how many/which repos need updating and dispatches one "sub-task" for each of them. The first task after reading the number of repos that need updating would set total when it is known. The sub-tasks work through the work but do not modify total again.

#9 - 06/08/2020 07:25 PM - ttereshc

I agree that we might find a way to set the total only once for the current use cases. I'm not entirely sure about the migration plugin because some of its work can be identified only during the migration itself but there is a chance that all those items are happening before subtasks for more well defined scope are dispatched.

+1 to start with F() for done. And if we have a use case, we can add it for total later.

#10 - 06/09/2020 12:51 PM - ipanova@redhat.com

- Description updated

#11 - 06/09/2020 12:53 PM - ipanova@redhat.com

- Blocks Story #6769: As a user, I can track the progress of pulp2->pulp3 migrations added

#12 - 06/09/2020 01:06 PM - ipanova@redhat.com

- Description updated

#13 - 06/09/2020 01:07 PM - ipanova@redhat.com

- Groomed changed from No to Yes

- Sprint set to Sprint 74

#14 - 06/11/2020 10:27 PM - rchan

- Sprint changed from Sprint 74 to Sprint 75

#15 - 06/16/2020 08:59 PM - ipanova@redhat.com

- Status changed from NEW to ASSIGNED

- Assignee set to ipanova@redhat.com

#16 - 06/26/2020 06:04 PM - rchan

- Sprint changed from Sprint 75 to Sprint 76

#17 - 06/29/2020 04:30 PM - ipanova@redhat.com

https://github.com/pulp/pulpcore/pull/755

#18 - 06/30/2020 10:15 AM - ipanova@redhat.com

- Status changed from ASSIGNED to MODIFIED

- % Done changed from 0 to 100

Applied in changeset pulpcore|104aed7b099305457d55d47accf72c0c3693a84a.

#19 - 07/08/2020 09:00 PM - fao89

- Sprint/Milestone set to 3.5.0

#20 - 07/09/2020 05:06 PM - pulpbot

- Status changed from MODIFIED to CLOSED - CURRENTRELEASE