Pulp - Issue #3812

Issue # 3770 (CLOSED - NOTABUG): Pulp 3 is about 2x slower than pulp 2 in syncing a large file repo

Pulp3 Content models are not compatible with bulk_save

07/02/2018 09:43 PM - bmbouter

Status:	CLOSED - WONTFIX	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	dalley	**Estimated time:**	0:00 hour
Category:			
Sprint/Milestone:	3.0.0		
Severity:	2. Medium	**Groomed:**	Yes
Version:		**Sprint Candidate:**	Yes
Platform Release:		**Tags:**	Performance
OS:		**Sprint:**	
Triaged:	Yes	**Quarter:**	

Description

Motivation

Content Units are not compatible with bulk_save() because they use Master/Detail and therefore multi-table inheritance which fails by raising a: ValueError: Can't bulk create a multi-table inherited model Exception.

Solution

TBD, write ideas in comments

Related issues:

Related to Pulp - Issue #3767: Unable to save models with relation to Content...	**CLOSED - NOTABUG**

History

#1 - 07/03/2018 04:37 PM - amacdona@redhat.com

- Related to Issue #3767: Unable to save models with relation to Content with changeset added

#2 - 07/03/2018 04:37 PM - amacdona@redhat.com

- Triaged changed from No to Yes

Whatever we come up with here, it might be relevant to take a look at this one also: https://pulp.plan.io/issues/3767

#3 - 07/05/2018 10:10 PM - dalley

- Status changed from NEW to ASSIGNED

- Assignee set to dalley

As discussed with Brian and David, first step is to make a flat version of a ContentUnit table and benchmark save() vs bulk_create() to see how much faster it actually is.

#4 - 07/10/2018 09:20 PM - dalley

Results:

```
(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 1000
1000 units: indidvidual save in seconds: 5.763491153717041
1000 units: bulk save in seconds: 0.1316215991973877

= 43.7x speedup

(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 5000
5000 units: indidvidual save in seconds: 28.695824146270752
5000 units: bulk save in seconds: 0.41101503372192383
```

```
= 69.8x speedup

(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 5000
5000 units: indidvidual save in seconds: 25.684953689575195
5000 units: bulk save in seconds: 0.4922950267791748

= 52.2x speedup

(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 5000
5000 units: indidvidual save in seconds: 25.778226375579834
5000 units: bulk save in seconds: 0.5030674934387207

=51.2x speedup

(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 5000
5000 units: indidvidual save in seconds: 28.895294189453125
5000 units: bulk save in seconds: 0.4287230968475342

=67.4x speedup

(bench) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py --num 10000
10000 units: indidvidual save in seconds: 51.1158390045166
10000 units: bulk save in seconds: 0.9658312797546387

= 52.9x speedup
```

Code:

https://github.com/dralley/pulp_content_benchmarks/

#5 - 07/11/2018 08:32 PM - bmbouter

- Status changed from ASSIGNED to NEW

- Assignee deleted (dalley)

Setting back to new because I'm not actively working on it.

#6 - 07/11/2018 08:35 PM - bmbouter

- Sprint Candidate changed from No to Yes

I updated the wrong issue, whoops. I think if its in new state though it will show up at sprint planning so maybe this is good.

#7 - 07/11/2018 08:45 PM - daviddavis

- Groomed changed from No to Yes

#8 - 07/11/2018 09:05 PM - jortel@redhat.com

Looking at the code, the *individual saves* test is being done with a commit per insert which is the slowest way possible. The bulk create by nature will be doing a single commit. Each commit is very expensive. For the metric to be useful, both tests need to do the same number of commits. Suggest modifying the *individual saves* test to run in a single transaction.

#9 - 07/12/2018 05:26 PM - dalley

New content benchmarks:

```
(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=1000
1000 multi-table content: individual save in seconds: 6.193058490753174
1000 multi-table content: individual save w/ transaction in seconds: 1.801978588104248
1000 single-table content: individual save w/ transaction in seconds: 1.0102925300598145
1000 single-table content: bulk save in seconds: 0.06991720199584961

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=1000
1000 multi-table content: individual save in seconds: 5.5913708209991455
1000 multi-table content: individual save w/ transaction in seconds: 1.853820562362671
1000 single-table content: individual save w/ transaction in seconds: 0.9765522480010986
1000 single-table content: bulk save in seconds: 0.07139015197753906

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=5000
5000 multi-table content: individual save in seconds: 25.336352109909058
5000 multi-table content: individual save w/ transaction in seconds: 8.87453031539917
5000 single-table content: individual save w/ transaction in seconds: 4.718692064285278
```

```
5000 single-table content: bulk save in seconds: 0.39882731437683105

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=5000
5000 multi-table content: individual save in seconds: 28.781572580337524
5000 multi-table content: individual save w/ transaction in seconds: 8.198915958404541
5000 single-table content: individual save w/ transaction in seconds: 4.499735116958618
5000 single-table content: bulk save in seconds: 0.4010334014892578

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=10000
10000 multi-table content: individual save in seconds: 57.38842058181763
10000 multi-table content: individual save w/ transaction in seconds: 16.86572265625
10000 single-table content: individual save w/ transaction in seconds: 9.358201503753662
10000 single-table content: bulk save in seconds: 0.8168251514434814

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=10000
10000 multi-table content: individual save in seconds: 58.31086325645447
10000 multi-table content: individual save w/ transaction in seconds: 16.490723609924316
10000 single-table content: individual save w/ transaction in seconds: 9.385928392410278
10000 single-table content: bulk save in seconds: 0.827233076095581

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=20000
20000 multi-table content: individual save in seconds: 114.19883108139038
20000 multi-table content: individual save w/ transaction in seconds: 34.82098698616028
20000 single-table content: individual save w/ transaction in seconds: 17.453803062438965
20000 single-table content: bulk save in seconds: 1.6993021965026855

(env) [vagrant@pulp3 models_benchmark]$ python3 benchmark.py  --num=20000
20000 multi-table content: individual save in seconds: 115.88576149940491
20000 multi-table content: individual save w/ transaction in seconds: 51.81009030342102
20000 single-table content: individual save w/ transaction in seconds: 24.00934410095215
20000 single-table content: bulk save in seconds: 2.081272840499878
```

benchmark.py code also updated.

#10 - 07/16/2018 02:20 AM - dkliban@redhat.com

- Sprint set to Sprint 40

#11 - 07/25/2018 11:19 PM - bmbouter

- Status changed from NEW to ASSIGNED

- Assignee set to bmbouter

#12 - 08/06/2018 03:16 PM - rchan

- Sprint changed from Sprint 40 to Sprint 41

#13 - 08/07/2018 06:08 PM - jortel@redhat.com

- Subject changed from Pulp3 Content Units are not compatible with bulk_save to Pulp3 Content models are not compatible with bulk_save

#14 - 08/07/2018 06:18 PM - jortel@redhat.com

Please include a design (change) proposal that specifies:

- Proposed Content model hierarchy.
- How content within a repository-version will be queried.
- How content will be associated to a repository-version.

so that it can be vetted before implementation.

#15 - 08/13/2018 11:35 PM - bmbouter

This is a good article on MultiTable inheritance https://godjango.com/blog/django-abstract-base-class-multi-table-inheritance/. It identifies that anytime you have a Concrete model inheriting from another concrete Django model you'll get multi-tale inheritance. That is specifically the thing that won't work with bulk_create according to their docs: https://docs.djangoproject.com/en/dev/ref/models/querysets/#bulk-create

So we have to make Content an abstract django model and have any subclasses, e.g. FileContent be the concrete model. This way you will be compatible with bulk_create(). With this change you can no longer make calls like Content.objects. becuase an abstract class does not have a Manager method, e.g. objects.

There are a few places where users or plugin writers treat mixed-type content as one type, e.g. the add_content(), remove_content() calls. These will need to move toward something that has distinct type anytime we deal with Queryset objects which cannot span single-tables.

This diff show the model changes that were necessary. The next step is to port the Querset areas to use a dictionary of Querysets, one per type. For

example the content argument to https://github.com/pulp/pulp/blob/master/pulpcore/pulpcore/app/models/repository.py#L337 would become:

```
{
    'pulp_ansible.AnsibleContent': 'QuerySetobjects',
    'pulp_rpm.UpdateContent': 'QuerySetobjects'
}
```

#16 - 08/27/2018 03:03 PM - rchan

- Sprint changed from Sprint 41 to Sprint 42

#17 - 09/17/2018 07:18 PM - rchan

- Sprint changed from Sprint 42 to Sprint 43

#18 - 10/03/2018 06:38 PM - dalley

- Assignee changed from bmbouter to dalley

#19 - 10/08/2018 03:07 PM - rchan

- Sprint changed from Sprint 43 to Sprint 44

#20 - 11/06/2018 07:51 PM - rchan

- Sprint changed from Sprint 44 to Sprint 45

#21 - 11/28/2018 05:19 PM - dalley

- Status changed from ASSIGNED to CLOSED - WONTFIX

- Sprint deleted (Sprint 45)

Closing as it looks like the model changes are unlikely to move forwards. The API changes will be added in a new story.

#22 - 04/25/2019 06:45 PM - daviddavis

- Sprint/Milestone set to 3.0.0

#23 - 04/26/2019 10:34 PM - bmbouter

- Tags deleted (Pulp 3)

#24 - 06/16/2020 10:59 PM - bmbouter

- Tags Performance added

- Tags deleted (Sync Performance)