Pulp - Task #1600

Store content using consistent and deterministic paths

01/29/2016 07:13 PM - jortel@redhat.com

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:			
Platform Release:	2.8.0	**Tags:**	Pulp 2
Groomed:	Yes	**Sprint:**	
Sprint Candidate:	No	**Quarter:**	

Description

Store content using consistent and deterministic paths. In 2.8, the platform determines the _storage_path (with some help from plugins). It determines the path using the UUID as:

```
/var/lib/pulp/content/units/<type_id>/<unit_id>[0:4]/<unit_id>/<file>
```

To support recovery scenarios, using the hash of unit_key instead of unit_id will be required. This makes the storage path deterministic.

Suggested:

```
/var/lib/pulp/content/units/<type_id>/<hash>[0:4]/<hash>/<file>
```

Related issues:

Blocks RPM Support - Story #236: Don't re-download rpms if they exist on disk	**CLOSED - CURRENTRELEASE**

Associated revisions

Revision ec3fafac - 02/04/2016 02:23 PM - jortel@redhat.com

Unit storage path based on sha256 of unit key instead of using UUID. closes #1600

Revision ec3fafac - 02/04/2016 02:23 PM - jortel@redhat.com

Unit storage path based on sha256 of unit key instead of using UUID. closes #1600

History

#1 - 01/29/2016 08:40 PM - bmbouter

In the suggestion, when you say hash do you mean <hash>?

Also can you remind me again what motivates the hash[0:4] part of the storage path? I figured the layout format would be:

```
/var/lib/pulp/content/units/<type_id>/<hash>/<file>
```

#2 - 01/29/2016 08:56 PM - jortel@redhat.com

The term *hash* in the path refers to the SHA256 hex digest of the unit key. Perhaps *digest* would be more accurate. The motivation for *hash[0:4]/hash* was to reduce the possibility of exceeding the maximum number of sub-directories in the units/.

#3 - 01/29/2016 09:37 PM - bmbouter

- Description updated

jortel@redhat.com wrote:

> The term *hash* in the path refers to the SHA256 hex digest of the unit key. Perhaps *digest* would be more accurate. The motivation for *hash[0:4]/hash* was to reduce the possibility of exceeding the maximum number of sub-directories in the units/.

hash is a fine term to use. I was more clarifying if you mean hash as a value which is different for each unit or the string 'hash'. Usually the changing values have <> around them. I just added brackets to the issue description.

I don't think /<hash>[0:4]/<hash>/ is helping us avoid running out of too many files in a directory any more than just using /<hash>/ instead because there is a 1:1 correspondence between /<hash>[0:4]/ and /<hash>/. Am I thinking about this right?

#4 - 01/29/2016 10:07 PM - jortel@redhat.com

bmbouter wrote:

> jortel@redhat.com wrote:
>
> > The term *hash* in the path refers to the SHA256 hex digest of the unit key. Perhaps *digest* would be more accurate. The motivation for *hash[0:4]/hash* was to reduce the possibility of exceeding the maximum number of sub-directories in the units/.
>
> hash is a fine term to use. I was more clarifying if you mean hash as a value which is different for each unit or the string 'hash'. Usually the changing values have <> around them. I just added brackets to the issue description.
>
> I don't think /<hash>[0:4]/<hash>/ is helping us avoid running out of too many files in a directory any more than just using /<hash>/ instead because there is a 1:1 correspondence between /<hash>[0:4]/ and /<hash>/. Am I thinking about this right?

The idea comes from how ostree stores files in objects/. I think the expectation is that given enough values, that the 1st 4 digits of the hash would duplicate sufficiently to build additional directories. Although, now that you ask, I think the probability of this is worth investigating. This really should be: units/type_id/<hash>[0:4]/<hash>[4:]/<file>

For example: hash values (shortened for illustration):

```
123401231
1234A1232
1234B1233
123501234
1235A1235
1235B1236
475649487
994049858
```

Would produce this tree:

```
1234/
   01231/
   A1232/
```

```
   B1232/
1235/
   01234/
   A1235/
   B1236/
4756/
   49487/
9940/
   49858/
```

#5 - 01/29/2016 10:27 PM - bmbouter

Oh that is interesting. I had not considered that.

I expect the hashing algorithm to provide good randomness in terms of the first 4 characters produced. Assuming the hash algorithm fills the possible combinations evenly and only allowing characters [A-Z] and [0-9], we would expect duplicates after 36^4 units are in the database. That's 1,679,616 units.

Given that, I propose:

```
/var/lib/pulp/content/units/<type_id>/<hash>/<file>
```

What do you think given all of this?

#6 - 01/29/2016 10:34 PM - bmbouter

- Blocks Story #236: Don't re-download rpms if they exist on disk added

#7 - 01/29/2016 11:15 PM - jortel@redhat.com

I did a quick test which generated 100,000 hashes using sha256 on unique strings. This produced enough duplicates of hash[0:4] to create 29,000 directories each containing 2-5 items. This seems to support that there is value if this approach.

#8 - 01/29/2016 11:22 PM - bmbouter

jortel@redhat.com wrote:

> I did a quick test which generated 100,000 hashes using sha256 on unique strings. This produced enough duplicates of hash[0:4] to create 29,000 directories each containing 2-5 items. This seems to support that there is value if this approach.

Nice test! +1 to keeping it as it is written in the issue:

```
/var/lib/pulp/content/units/<type_id>/<hash>[0:4]/<hash>/<file>
```

#9 - 02/01/2016 03:39 PM - jcline@redhat.com

I don't know how many units people keep in Pulp of a given type, and I don't know what filesystems people use. I do know that ext3 has sub-directory limits of 31998, which is less than 36**3 (I'm assuming a 36 character alphabet for our hash language). I also suspect that performance starts to degrade pretty seriously with large numbers of sub-directories, but without knowing the nitty-gritty details of each filesystem, that's a bit hand-wavy.

My personal opinion is that a prefix of 2 quite sufficient. That's 1296 possible sub-directories (assuming a 36 character alphabet) and if they are evenly distributed we should, on average, maintain sub-directory counts less than or equal to 1296 until well over 1 million units. I don't think an even distribution is necessary guarantee of a hashing algorithm, but it's not hugely damaging if we assume it is will be near to even.

#10 - 02/01/2016 04:57 PM - jortel@redhat.com

Using 2 digits seems to produce more favorable results. Running the same test for 100,000 units using hash[0:2] and SHA256 results in creating 256 directories each containing 300-400 subsirectories.

#12 - 02/01/2016 08:42 PM - jortel@redhat.com

If only using 2 digits, I'm fine with either.

#13 - 02/02/2016 10:21 AM - mhrivnak

+1 to all of this. 2 characters should be enough and is a common approach I've seen in other situations.

One additional motivation: on some filesystems, there can be a real performance impact on file access if a directory listing gets very large. Especially on some older filesystems, the directory data structure is not optimized for random access.

#14 - 02/02/2016 04:28 PM - jortel@redhat.com

- Status changed from NEW to ASSIGNED

#15 - 02/02/2016 11:05 PM - jortel@redhat.com

- Status changed from ASSIGNED to POST

https://github.com/pulp/pulp/pull/2385

#16 - 02/08/2016 05:37 PM - jortel@redhat.com

- Status changed from POST to MODIFIED

- % Done changed from 0 to 100

Applied in changeset pulp|ec3fafac36510e6d9216afbe22a9645f5f2c6e8d.

#17 - 02/23/2016 09:56 PM - dkliban@redhat.com

- Status changed from MODIFIED to 5

#18 - 03/23/2016 07:13 PM - dkliban@redhat.com

- Status changed from 5 to CLOSED - CURRENTRELEASE

#19 - 04/15/2019 10:36 PM - bmbouter

- Tags Pulp 2 added