w.c.s. - Development #23028

perf : chargement des formdefs

09 avril 2018 10:08 - Frédéric Péters

Statut:	Fermé	**Début:**	09 avril 2018
Priorité:	Normal	**Echéance:**	
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Sur des pages comme la vue globale, ou dans des API comme la récupération de la liste des démarches, un temps important est passé sur le chargement de tous les formdefs. (en local par exemple, sur un api/categories/foobar/formdefs/ 0,32 secondes sur le chargement des formdefs, sur un total de 0,37 secondes).

À mesurer les choses ce ne sont pas les accès disques (0,002 secondes) mais bien l'unpickling (0,32 secondes, du coup); on utilise pickle et non cPickle pour gérer un déplacement de module qui a eu lieu il y a très longtemps (cf UnpicklerClass) mais passer à cPickle n'est pas une modif extraordinaire, genre ça descend à 0,25 secondes.

Pour mieux faire, mon plan est de séparer les champs des autres données. Ça permet de descendre ainsi vers 0,05 secondes.

(patch en cours)

Révisions associées

Révision 46ef21e5 - 02 mai 2018 12:32 - Frédéric Péters

formdef: store fields in a different pickle chunk (#23028)

Historique

#1 - 09 avril 2018 12:33 - Frédéric Péters

- Fichier 0001-formdef-store-fields-in-a-different-pickle-chunk-230.patch ajouté

- Statut changé de Nouveau à En cours

- Patch proposed changé de Non à Oui

#2 - 01 mai 2018 13:12 - Thomas Noël

C'est funky. Ack.

#3 - 01 mai 2018 13:17 - Thomas Noël

Juste, quand même, pour ma gouverne:

```
    def storage_dumps(cls, object):
        if getattr(object, 'fields', None) is Ellipsis:
            raise RuntimeError('storing a lightweight object is not allowed')
        return pickle.dumps(object) + pickle.dumps(object.fields)
```

J'aurai presque ajouté un commentaire avant les pickle.dumps, qui explique que l'object va être stocké dans self.fields grâce au *getstate* codé plus haut. Parce que sinon cette dernière ligne semble presque mystérieuse ; en tout cas pour moi.

#4 - 01 mai 2018 13:44 - Frédéric Péters

> l'object va être stocké dans self.fields

sans self.fields, est bien ce que tu voulais écrire ?

Je propose :

```
@@ -1362,6 +1362,9 @@ class FormDef(StorableObject):
```

```
     def storage_dumps(cls, object):
         if getattr(object, 'fields', None) is Ellipsis:
             raise RuntimeError('storing a lightweight object is not allowed')
+        # use two separate pickle chunks to store the formdef, the first field
+        # is everything but fields (excluded via __getstate__) while the second
+        # chunk contains the fields.
         return pickle.dumps(object) + pickle.dumps(object.fields)
```

#5 - 01 mai 2018 14:01 - Benjamin Dauvergne

Je me demande ici si ce ne serait pas plus simple de les mettre en cache et de ne les recharger que si (stat.st_mtime, stat.st_ino) est identique depuis la dernière utilisation (et aussi un stat sur le répertoire, ça devrait suffire pour les créations) avec un timeout à 30s, au cas où ça manquerait un truc (peut-être pas toujours mais avoir un get_cached_formdefs()/get_cached_formdef(id) pour cela, ou alors d'ajouter la fonctionnalité dans StorableObject).

La façon dont on met à jour les fichiers avec atomic_rewrite() garantit un changement d'inode je pense (dans le cas hautement improbable d'une collision de st_mtime).

#6 - 01 mai 2018 14:31 - Frédéric Péters

L'infrastructure de cache de django ne fonctionnerait pas, même le backend local memory utilise pickle pour mémoriser l'objet (j'ai juste regardé à 1.11); ça veut dire mettre en place un autre module de cache qui garderait l'objet Python "natif" en mémoire locale; bien sûr même en local memory s'assurer qu'on profite réellement du cache (après combien de temps un processus uwsgi est-il abandonné ?), puis s'assurer d'un tas de trucs que je n'arrive même pas à énumérer (il faudrait éviter la modification d'objet sans .store() derrière, faire gaffe au _start_page qui devrait être réinitialiser à chaque requête, etc.).

D'un regard vraiment honnête à la question, non je ne pense pas que ça soit plus simple.

#7 - 01 mai 2018 14:58 - Benjamin Dauvergne

Je pensais juste à mettre les objets en cache localement (dans le worker, dans un bête dico, éventuellement thread local pour pas se bloquer l'usage du multithreading); je ne vois pas la différence entre un worker qui charge un formdef du disque et s'en sert pendant 30s alors que pendant ce temps un autre worker vient d'en écrire un autre. Le fonctionnement actuel n'a pas grande différence avec celui-ci en fait, une fois chargé on peut dire que le FormDef est en cache pour la durée de la requête (enfin de la variable qui pointe dessus).

Concernant la modification aux objets du cache, on peut faire un copy.deepcopy() avant de retourner les objets (ça résout l'histoire de _start_page aussi je pense).

#8 - 01 mai 2018 15:09 - Benjamin Dauvergne

Je viens d'avoir un doute pour les montages NFS, mais visiblement il rapporte de manière synchrone le vrai mtime et le véritable inode.

#9 - 01 mai 2018 15:39 - Frédéric Péters

Pour cette discussion sur le cache, si elle te semble intéressante, un nouveau ticket serait plus approprié, ici j'essayais juste de répondre pour dire que non, ça ne me semblait pas plus simple. (ça ne me le semble toujours pas)

#10 - 01 mai 2018 19:03 - Thomas Noël

Frédéric Péters a écrit :

> l'object va être stocké dans self.fields

sans self.fields, est bien ce que tu voulais écrire ?

Bien sûr... désolé.

> Je propose :

Impec. J'ai tout compris ! :)

(pour la discussion sur l'utilisation d'un cache, je suis dans l'ancienne école qui fait trop confiance au buffer/cache du noyau pour avoir envie de le ré-implémenter)

#11 - 02 mai 2018 10:47 - Benjamin Dauvergne

Frédéric Péters a écrit :

> Pour cette discussion sur le cache, si elle te semble intéressante, un nouveau ticket serait plus approprié, ici j'essayais juste de répondre pour dire que non, ça ne me semblait pas plus simple. (ça ne me le semble toujours pas)

D'ac.

#12 - 02 mai 2018 12:34 - Frédéric Péters

- Statut changé de En cours à Résolu (à déployer)

```
commit 46ef21e541151a190cf17f228d50aa457c8db84a
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Apr 9 12:21:44 2018 +0200

    formdef: store fields in a different pickle chunk (#23028)
```

#13 - 02 mai 2018 13:22 - Frédéric Péters

Installé sur auquo.dev, exécuté en runscript sur tous vhosts :

```
from wcs.formdef import FormDef

for formdef in FormDef.select():
    formdef.store()
```

et tout m'a eu l'air ok.

#14 - 23 décembre 2018 14:41 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

0001-formdef-store-fields-in-a-different-pickle-chunk-230.patch	13,4 ko	09 avril 2018	Frédéric Péters