Publik - Bug #18889

"--harakiri 30" c'est un peu violent

21 septembre 2017 19:11 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	21 septembre 2017
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Club:**	Non
Planning:	Non		

Description

Suite au ticket #18888 à Nanterre:

Le fonctionnement d'harakiri ne nous donne pas d'information sur l'origine du problème, je suggère quelque chose de ce genre en plus:

```
diff --git a/wcs/compat.py b/wcs/compat.py
index 756818b..1e6f60f 100644
--- a/wcs/compat.py
+++ b/wcs/compat.py
@@ -16,6 +16,7 @@

 import ConfigParser
 import os
+import signal

 from threading import Lock
 from contextlib import contextmanager
@@ -211,9 +212,25 @@ class CompatWcsPublisher(WcsPublisher):
 # the publisher instance can't be shared for concurrent requests.
 quixote_lock = Lock()

+
+class Timeout(Exception):
+    pass
+
+
 def quixote(request):
     pub = get_publisher()
-    return pub.process_request(pub.get_request())
+
+    def timeout(signal, frame):
+        raise Exception('timeout after 29 seconds')
+
+    signal.signal(signal.SIGALRM, timeout)
+    try:
+        signal.alarm(29)
+        resp = pub.process_request(pub.get_request())
+        signal.alarm(0)
+    except Timeout:
+        pass
+    return resp

 @contextmanager
```

L'idée c'est que si Timeout est raisé dans publisher ce sera choppé et une exception générée par le publisher, si on se trouve après le try_publish() l'exception remontera jusqu'à Django qui enverra un mail, je serai d'avis d'en plus tuer le worker uwsgi si on trouve comment faire parce qu'un process qui timeout c'est un process potentiellement en vrac.

Historique

#1 - 21 septembre 2017 19:11 - Benjamin Dauvergne

- Description mis à jour

#2 - 21 septembre 2017 22:57 - Frédéric Péters

- Projet changé de w.c.s. à Publik

Je ne pense pas que ça soit spécifique à w.c.s. : de manière générale, on devrait avoir une configuration nous permettant de pointer les moments lents pour l'usager, et 30 secondes c'est déjà beaucoup trop, et le sujet serait alors de marquer une différence entre les requêtes "client/navigo" et les requêtes "inter serveurs".

#3 - 22 septembre 2017 01:27 - Thomas Noël

Frédéric Péters a écrit :

> Je ne pense pas que ça soit spécifique à w.c.s. : de manière générale, on devrait avoir une configuration nous permettant de pointer les moments lents pour l'usager, et 30 secondes c'est déjà beaucoup trop, et le sujet serait alors de marquer une différence entre les requêtes "client/navigo" et les requêtes "inter serveurs".

Il y a malgré tout en client/navigo des opérations qui semblent pouvoir prendre plus de 30 secondes (changement d'un workflow => remise à plat de toutes les demandes liées). Mais bon, je dis ça juste comme ça, ce n'est pas lié à harakiri, pas seulement.

#4 - 22 septembre 2017 09:10 - Frédéric Péters

> Il y a malgré tout en client/navigo des opérations qui semblent pouvoir prendre plus de 30 secondes (changement d'un workflow => remise à plat de toutes les demandes liées).

On doit arriver à identifier celles-ci et les rendre asynchrones. Sur la configuration des 30 secondes, c'était similaire avec scgi, la différence étant que ça se gérait là "hors paquet", côté nginx.

#5 - 22 septembre 2017 11:13 - Benjamin Dauvergne

Je suis d'accord avec tout, je signale juste que la solution à base signaux ALRM n'est compatible qu'avec un mode multiprocess, si on passe à des threads au niveau uwsgi/gunicorn ça plantera. Si on reste sur SIGALRM on peut très bien mettre un premier timeout à 10 secondes qui envoie juste une trace en mode warning et un deuxième 19s plus tard qui arrête la requête.

L'autre méthode c'est de lancer deux threads un pour la requête un pour des timers, celui avec les timer peut utiliser sys._current_frames() pour obtenir la trace du thread de requête ou peut l'arrêter et le thread principal fait un join des deux, le thread de requête pose sa réponse dans une variable partagée.

#6 - 25 mai 2023 08:29 - Frédéric Péters

- Statut changé de Nouveau à Fermé

- Planning mis à Non

- Club mis à Non

Un peu plus tard il y a eu #19194 assez doublon et c'est passé à 120 secondes, on vit avec ça depuis cinq ans ça me semble aller.