Anto59290

Zeste de savoir

Django et performance

1er février 2020

Table des matières

Les frameworks tel que Django ont parfois la réputation de devenir assez lents quand ils sont utilisés sur des gros projets. En effet, Django est souvent utilisé pour tester rapidement des idées, cela est peut-être dû à sa philosophie "livré avec les piles". C'est aussi souvent un framework choisi par les fondateurs de start-ups pour livrer un prototype fonctionnel le plus rapidement possible.

Il arrive parfois qu'après quelques années du code "naïf" de la première heure soit toujours en production et reçoivent de plus en plus de charge chaque jour. À un moment une partie de ce code deviendra tellement lent que cela dégrade l'expérience des utilisateurs. Prenons par exemple le nombre d'objets enregistrés dans la base de données, il est probable que ce nombre grandisse rapidement lorsque votre projet se développe. Le code naïf du début n'aura probablement pas été pensé pour gérer des millions d'objets en base, et sera donc assez inefficient.

Dans cette série de billets nous verrons comment identifier et résoudre les problèmes de performances les plus courants avec Django. Commençons par voir comment mesurer et identifier les problèmes de performances. Dans cette série de billets nous nous baserons sur des applications utilisant Python / Django et une base de données SQL, certains exemples utiliseront Django REST Framework.

1. Observabilité

Avant de tacler un problème de performance la première étape de s'assurer d'avoir un monitoring correct. En effet, comment vérifier les résultats des modifications apportées si vous n'avez aucune idée de ce qui se passe en production en termes de performances. Dans le cas d'une application web on considère généralement la latence comme la métrique la plus importante quand il s'agit de performance. De ce fait pouvoir mesurer la latence moyenne, P75, P90, P95 et P99 semble être un bon point de départ. Dans certains cas plus poussés il pourra aussi être intéressant de mesurer le CPU, la RAM et le réseau utilisé par les serveurs, on entre alors plus dans le monitoring de l'infrastructure que de l'application. En effet une modification de code peut très bien rendre l'application plus rapide mais beaucoup plus gourmande en RAM, ce qui peut parfois causer des gros dégâts.

Il est aussi important de garder l'environnement de test aussi stable que possible lorsque l'on teste des changements de code dans le but d'améliorer la performance. En effet ne changer qu'un seul paramètre de l'expérience à la fois est une bonne pratique scientifique à utiliser ici. Par exemple veiller à ne pas changer les ressources du serveur ou le type de base de données entre plusieurs tests afin de garder les résultats comparables.

FIGURE 1.1. – Le logo de DataDog

Pour être capable de monitorer les performances de votre application en production (ou dans n'importe quel environnement) on utilise un APM. Il en existe de différentes sorte sur le marché, une solution en SaaS a l'avantage d'être assez rapide à mettre en place et ne nécessite que très peu de modification sur votre code et votre infrastructure. En revanche beaucoup de données (parfois sensibles) sont passées à ces outils, ce qui peut être gênant dans certains contextes. Le but de cet article n'est pas de comparer l'ensemble des solutions disponibles. après avoir longuement utilisé Newrelic je suis passé depuis deux ans à Datadog, dont l'utilisation est à mon sens plus simple. L'installation se fait simplement en ajoutant un petit agent sur le serveur et en ajoutant un middleware dans votre configuration Django. Une fois que la solution de votre choix est installée, voici le genre d'information que vous devriez avoir:

— La latence de votre application Django (avec de l'historique)
— Les routes les plus utilisées dans l'application
— Les routes les plus lentes dans l'application
— Les routes pour lesquelles votre serveur a passé le plus de temps
— Des traces / détails pour chaque route spécifique, les outils échantillonnent généralement un certain pourcentage. Cela est normal, mais veillez à avoir assez de matière pour pouvoir tirer des conclusions.

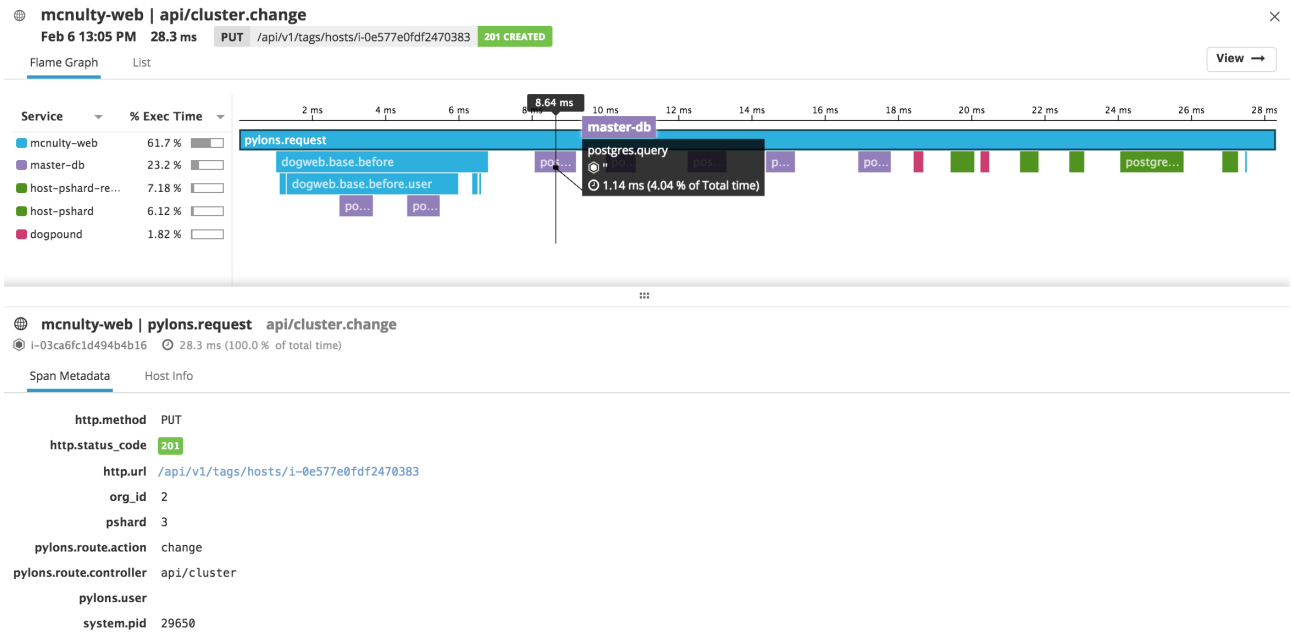

FIGURE 1.2. – Un exemple de traces dans un APM

Il est important de comprendre que chaque métrique apporte des informations différentes. Par exemple si vous vous intéressez à la performance pour des raisons de coûts (payer moins de serveurs par exemple) alors la métrique "temps passé par le serveur" est probablement celle qui vous intéresse le plus. En revanche si vous essayez d'optimiser le taux de conversion de votre site vous allez probablement plus vous concentrer sur la performance des pages les plus critiques de votre site en termes de ventes.

Au plus vous creuserez dans l'APM au plus vous trouverez des détails intéressants sur comment se comporte votre application en production. Par exemple: est ce que le site est toujours aussi rapide quelle que soit la langue d'affichage (dans le cas d'un site en plusieurs langues)? Est-ce que le site est plus rapide pour un utilisateur anonyme ou pour un utilisateur connecté? En général la latence moyenne est un bon point de départ mais ne vous emmènera pas très loin en termes d'optimisation. Comme dirait mon frère: "Avec la tête dans le frigo et les pieds dans le four, en moyenne tout va bien"

Ce billet fait partie d'une série qui entrera en détails sur l'optimisation des performances web avec Django. Le but sera de compiler tous les billets dans un article. N'hésitez à réagir où à poser vos questions