Byng - Documentation technique

Introduction

Byng est le nom donné à un moteur de recherche programmé em C++, il est constitué de 3 parties. Une interface web de recherche, une api de recherche retournant les résultats et un crawler navigant sur le web de page en page.

Architecture

Base de données

Nous avons décidé d'utiliser PostgreSQL à cause de ses performances et fonctionnalitées beaucoup plus intéressantes que MySQL mais aussi par préférence personnelle.

Crawler

Le crawler est constitué de deux parties.

Une partie "Scheduler" qui requête régulièrement la base de donnée pour des liens expirés toutes les 5 minutes.

Et une partie "Crawler" attends de recevoir une URL à récupérer. Le Crawler utilise la libraire Gumbo de Google, testée sur leur index, il a paru évident de l'utiliser.

Le Crawler récupère la page HTML en utilisant CURL, parse avec Gumbo et récupère l'URL, le titre de la page, son favicon et les h1, h2, h3, h4, h5, h6.

Ces éléments sont ensuite insérés dans la base de donnée avec l'heure actuelle.

Le Crawler a également récupéré toutes les balises "a" pour ajouter leurs urls à la queue.

La communication entre les crawlers s'effectue avec rabbit-mq afin de pouvoir communiquer depuis l'extérieur et ajouter facilement de nouvelles URLs.

Le crawler accepte également une variable d'environnement afin de lancer plus d'un thread de crawling.

API

L'API Byng est en somme toute simple. Son rôle est d'exposer un accès à l'interface de recherche et de redonner les résultats en format JSON afin d'être digestible par l'interface web et même une éventuelle application mobile.

C'est cette API qui fait appel à la base de donnée afin d'effectuer la recherche.

```
SELECT url, title, favicon
FROM (
      SELECT url as url, title as title, favicon as favicon
,
      setweight(to_tsvector(array_to_string(string_to_array
(url, '-'), ',')), 'A') || ' ' ||
      setweight(to_tsvector(title), 'B') || ' ' ||
      setweight(to_tsvector(array_to_string(h1, ',')), 'B')
 || ' ' ||
      setweight(to_tsvector(array_to_string(h2, ',')), 'C')
 || ' ' ||
      setweight(to_tsvector(array_to_string(h3, ',')), 'C')
 || ' ' ||
      setweight(to_tsvector(array_to_string(h4, ',')), 'D')
 || ' ' ||
      setweight(to_tsvector(array_to_string(h5, ',')), 'D')
 || ' ' ||
      setweight(to_tsvector(array_to_string(h6, ',')), 'D')

      as document from crawled_sites
) search
WHERE search.document @@ to_tsquery($QUERY)
ORDER BY ts_rank(search.document, to_tsquery($QUERY)) DESC
LIMIT $LIMIT OFFSET $OFFSET;
```

C'est dans cette requête que PostgreSQL démontre son utilité par rapport à MySQL. Nous créons un document normalisant le contenu de nos colonnes

(faisant la moyenne des mots dans un format spécial) en attribuant une note si une occurrence est trouvée (par exemple, une occurrence trouvée dans le titre est plus importante que trouvée dans un h6). Nous récupérons ainsi l'url, le titre et le favicon rangé par score.

Les résultats sont ensuite sérialisés et envoyés.

L'API expose également un endpoint afin de pouvoir ajouter une URL à la queue du crawler.

Libmongoose est utilisé pour servir les requêtes HTTP.

Interface web

L'interface web est une simple page html avec un formulaire de recherche qui une fois soumis, envoi une requête POST vers l'API. Le résultat est ensuite affiché sur la page web.

Un autre formulaire est également disponible afin de soumettre une URL au crawler.

Cluster

Byng a été déployé avec succès dans un cluster CoreOS utilisant Docker et provisionné avec Ansible.

CoreOS

CoreOS est une distribution linux destinée aux déploiements et permet de deployer un cluster de plusieurs serveurs facilement. CoreOS fourni également un magasin de configuration distribué (etcd) et un outil en ligne de commande pour configurer son cluster (fleet).

Docker

Docker est une plateforme logicielle permettant de fabriquer, tester, deployer et lancer des applications distribuées. Chaque "container" est en fait une machine virtuelle légère avec une très faible empreinte contrairement à des

machines virtuelles classiques émulant tout un stack matériel. Docker permet la distribution de ces "containers" sur le réseau afin d'être deployable facilement à travers un cluster.

Serveur maître

Le serveur maître consiste en une instance d'Ubuntu 14.04 LTS avec une copie configurée d'etcd et de fleet.

Tout d'abord, il y a le serveur DNS. Byng utilise SkyDNS qui est un serveur dynamique où sa configuration est stockée dans etcd. Cela permet de mettre à jour en temps réel sa configuration et d'ajouter ou de retirer des éléments en fonction des serveurs et services lancés. Il est configuré pour répondre au domaine .byng.

Nginx expose deux éléments. L'API publiquement accessible depuis **query.byng** et l'interface web accessible publiquement depuis **search.byng**.

Les requêtes envoyées à ces domaines sont "reverse-proxy" vers l'intérieur du cluster sur le domain **web.byng** et **api.byng**. Nginx résous ces domaines à chaque requête afin de bénéficier du load balancing offert par le serveur DNS qui renvoie à chaque fois un serveur différent.

Sur ce serveur maître, PgPool2 est installé. PgPool2 est un "middleware" parlant le protocol de PostgreSQL, c'est à dire que n'importe quel client peut s'y connecter et exécuter des requêtes comme si c'était un serveur normal. Sauf que PgPool2 permet de pooler ces connexions, de paralléliser ces requêtes et d'assurer la réplication des données.

Chaque requête modifiant la base de donnée (INSERT, CREATE TABLE, etc...) est envoyée uniquement sur le serveur maître, puis sont répliquées après réussite sur l'ensemble du cluster. Cependant, une requête de lecture comme un SELECT est envoyée sur un serveur aléatoire. PgPool2 assurant ainsi le load balancing.

Ce serveur a également Docker d'installer afin de pouvoir compiler les différents éléments de Byng.

Cluster

Le cluster CoreOS est par défaut constitué de 3 serveurs où la base de donnée, l'api, l'interface web et le crawler sont déployés. A chaque déploiement réussi d'un service, le magasin de configuration etcd est mis à jour et ainsi, le serveur DNS est conscient automatiquement des changements d'adresses IPs.

Schéma