BOUILLON Guillaume 17076
CAESTECKER Guillaume 17036

Rapport : Scalable architecture

Introduction

Dans le cadre de ce projet, il nous est demandé de mettre en place une architecture extensible.

> La scalabilité peut faire référence à la capacité d'un système à accroître sa capacité de calcul sous une charge accrue quand des ressources, généralement du matériel, sont ajoutées.

Nous avons ainsi mis en place l'affichage du fractal de Mandelbrot en utilisant un système de serveur permettant de calculer l'appartenance des différents pixels à l'ensemble de Mandelbrot. Chaque pixel est ainsi représenté par ses coordonnées en x et en y comme un nombre complexe.

Ainsi dans ce document nous allons vous détailler le fonctionnement du serveur API, la stratégie de répartition du load balancer et décrire les bibliothèques utilisées.

Documentation API

Nous avons utilisé nginx comme loead balancer pour gérer notre serveur. Ainsi, pour accéder à celui-ci, nous lançons un docker sur le port 8180. De ce fait, à chaque appel le serveur redirige les requêtes vers un des workers définis au préalable.

Il suffit de passer 3 paramètres lors de l'appel pour permettre le calcul :

- La valeur du réel du complexe
- La valeur de l'imaginaire du complexe
- Le nombre d'itération maximale

En retour, le server renvoie donc le nombre d'itération effectuée lors du calcul. Si le nombre maximum d'itération a été atteint, cela signifie que le pixel appartient à l'ensemble de Mandelbrot.

Type de requête	URL	Paramètres	Réponse
GET	http://ENDPOINT/	- complex_r:float - complex_r:float - Iter:int	Body: {"response":ITER_NUMBER}Status : 200

Load balancer

Par défaut, et n'ayant rajouté aucun paramètre s'y opposant, nginx utilisé comme algorithme de load balancing la méthode round robin.

> L'algorithme fonctionne de telle sorte que c'est une répartition des charges équitable entre les différents serveurs du cluster. Chaque serveur traite le même nombre de requêtes. Cela nécessite une ferme de serveur homogènes en capacité de traitement.

Bibliothèques utilisées

Python

Bottle[1]

Bottle est un micro-framework Web WSGI rapide, simple et léger pour Python. Il est distribué sous la forme d'un module de fichier unique et n'a aucune dépendance autre que la bibliothèque standard Python.

Nous avons donc utilisé Bottle pour faire tourner nos differents workers rapidement et facilement.

Cython[2]

Cython est un langage de programmation et un compilateur qui simplifient l'écriture d'extensions compilées pour Python. La syntaxe du langage est très similaire à Python mais il supporte en plus un sous-ensemble du langage C/C++ (déclarations de variables, appel de fonctions, ...).

Le premier intérêt de Cython est qu'il produit du code nettement plus performant. Dans des programmes qui nécessitent par exemple la manipulation de grands tableaux, le gain peut aller jusqu'à un facteur 1003. Par ailleurs, Cython permet d'écrire des interfaces Python à des bibliothèques externes écrites en C ou C++.

Nous avons utilisé la librairie Cython pour améliorer les performances des calculs de nos workers.

Kivy3

Kivy est une bibliothèque libre et open source pour Python, utile pour créer des applications pourvues d'une interface utilisateur naturelle. Cette bibliothèque fonctionne sur Android, iOS, GNU/Linux, OS X et Windows. Elle est distribuée gratuitement et sous licence MIT.

Le framework contient tous les éléments pour construire des applications et notamment :

- fonctionnalités de saisie étendues pour la souris, le clavier, les interfaces utilisateurs tangibles, ainsi que les évènements multi-touch générés par ces différents matériels.
- une bibliothèque graphique basée seulement sur OpenGL ES2, et utilisant les Objets Tampons Vertex et les shaders.
- une large gamme de widgets acceptant le multi-touch.
- un langage intermédiaire (le Kv3), pour construire facilement des widgets personnalisés.

Nous avons utilisé Kivy pour créer un client python permettant de représenter l'ensemble de Mandlebrot grâce aux requêtes sur vers load balancer.

Autres

Docker[4]

Le logiciel « Docker » est une technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux®.

---

[1] https://bottlepy.org/docs/dev/
[2] https://fr.wikipedia.org/wiki/Cython
[3] https://fr.wikipedia.org/wiki/Kivy
[4] https://www.redhat.com/fr/topics/containers/what-is-docker

Avec la technologie Docker, il est possible de traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs offrent une grande flexibilité : ils permettent de créer, déployer, copier et déplacer des systèmes d'un environnement à un autre.

Nous avons utilisé des des images docker pour pouvoir déployer facilement plusieurs workers sur une même machine, et par la suite sur un serveur distant.

Nginx[5]

NGINX, prononcé comme « engine-ex », est un serveur web open-source qui, depuis son succès initial en tant que serveur web, est maintenant aussi utilisé comme reverse proxy, cache HTTP, et load balancer NGINX est conçu pour offrir une **faible utilisation de la mémoire** et une grande simultanéité. Plutôt que de créer de nouveaux processus pour chaque requête Web, NGINX utilise une approche asynchrone et événementielle où les requêtes sont traitées dans un seul thread.

Avec NGINX, un processus maître peut contrôler plusieurs processus de travailleurs. Le maître gère les processus du travailleur, tandis que les travailleurs effectuent le traitement proprement dit. Comme NGINX est asynchrone, chaque requête peut être exécutée simultanément par le travailleur sans bloquer les autres requêtes.

Comme dit précédemment, nous avons utilisé nginx comme load balancer et nous avons utilisé une image docker (nginx:latest) pour créer le container. Il suffit de modifier le fichier nginx.conf dans le conteneur avec nos informations de répartition des serveurs.

Fonctionnement

Se référer au README du projet ici[6].

[5] https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-nginx/
[6] https://github.com/Guigxs/Fractal-Viewer