Cahier des charges du projet Benchtrack

Alexis Lenoir -Xin He -Zhuangzhuang Yang

Jeudi 4 février 2021

1 Contexte et présentation

L'origine de ce projet part du constat qu'il est difficile de comparer des implémentations différentes d'algorithmes avancées. En effet, celles-ci dépendent de nombreux facteurs comme une grande diversité dans les formats d'entrée et de sortie, des conditions d'utilisation et plus simplement du langage de programmation utilisé.
Il existe déjà plusieurs frameworks qui s'occupe de gèrer des tests sur des implémentations, comme le module pytest et la solution logicielle Jenkins. Mais ils s'attachent à tester pour s'assurer de la validité du code, ils interviennent dans son élaboration, notre objectif se place en aval : nous ne voulons pas développer des implémentations, mais en comparer des préexistantes. Le maître mot est donc comparaison.
Notre objectif est donc d'élaborer un framework python, le plus générique possible, de comparaison d'impléméntation d'algorithme complexe pour générer un benchmark, c'est à dire pouvoir observer et comparer les perfomances des différentes implémentations. Les résultats seront affichés à l'aide d'un site statique. Ce site de comparaison devra pouvoir être mis à jours automatiquement, régulièrement.

2 Besoins et contraintes

Nous voulons que notre infrastructure de comparaison soit la plus générique possible. A l'aide d'un ensemble de fichiers contenant des implémentations (targets) et d'autres sur lesquels on va tester nos targets (benchmark) afficher des résultats de comparaison, impérativement dans un site statique. Nous souhaitons lancer ces comparaison avec un script python.

2.1 Description générales des entrées :

Un ensemble de target, un ensemble de benchmark. Nous devons spécifier pour chaque benchmarks (bout de code) :

— Les targets (implémentations) que l'on peut tester (dans une arborescence)

— Les données d'entrées nécessaire pour le test (output)
— Les résultats attendus du tests (expected)

3 Objectifs d'une première version

Fonctionnalités générales :
— Les fichiers de configurations en json
— Un script run.py qui permet de lancer les tests
— Létude porte sur deux critères : la vitesse d'éxécution et l'exactitude des réponses
— Les résultats doivent s'afficher sur un site statique (un fichier html)

3.1 Description des inputs :

Des differents benchmark dans differents répertoires, pour des differents benchmark(implémentations) :
"target.json" : inclus les noms des fichiers du code(les fichier doivent contenus dans le dossier des codes),des parametres des differents TESTs "expected.json" :le resultat expected config.ini : pour configuration.

3.2 Description des outputs :

Au début, il faut créer des fichers de resultats sous format txt. A partir des resultats,donner les resultats en format HTML stastique. Afficher temps d'exécution des différents implémentations en graphe et en tableau Aussi les codes des TESTs.

4 Améliorations futures

On pourra ensuite implémenter les améliorations suivantes :
— Augmenter le nombre de critère dans l'évaluation de la comparaison
— Prendre en compte plus de langages : R, Java, C++, Julia ...
— Améliorer l'affichage du site statique
— Conserver en mémoire les résultats, pour les comparer entre eux, montrer ainsi l'évolution

5 Délai

Une première version fonctionnelle doit être achever avant le lundi 1er mars 2021.
La dernière version de notre projet doit être réaliser avant le lundi 17 mai 2021.