LINFO1252 – Systèmes informatiques

Travail Dirigé (TD) Semaine 7 : Initiation à la mesure de performance

Objectifs de ce TD : Vous aurez besoin de mesurer la performance d'applications multi-threadées dans le cadre du projet. Dans ce TD, vous apprendrez à mesurer la performance d'une application et à automatiser le traitement et la présentation des résultats.

Compétences travaillées : écriture de script `bash`, consultation de la documentation (`man`), réalisation d'un script de post-traitement en python, utilisation de la librairie `matplotlib`

Pré-requis : Ce TD nécessite l'accès à une machine physique ou virtuelle avec plusieurs cœurs (minimum 2, idéalement 4 ou plus). Si vous n'avez pas accès à une machine avec plusieurs cœurs il est acceptable de réaliser le TD avec un binôme. Le TD n'est pas noté mais les compétences acquises sont nécessaires pour réaliser le projet.

Étape 1 : Mise en place (5 minutes)

Nous allons utiliser comme charge de travail la compilation d'un logiciel de complexité moyenne, un moteur de rendu HTML écrit en C appelé Modest.

Faites un clone du dépôt git suivant : https://github.com/lexborisov/Modest

Le dépôt contient un `Makefile` que vous pouvez inspecter. La cible par défaut (`make`) et le nettoyage (`make clean`) seront les deux cibles qui nous seront utiles.

Étape 2 : Paralléliser la compilation avec `make` (10 minutes)

En consultant la documentation de `make`, déterminez l'option permettant de lancer X instances de compilation simultanément. Cette option vous permettra de contrôler le niveau de parallélisme de thread lors des exécutions. Testez-la avec X={1, 2, 4, 8, 16}. Pensez à utiliser `make clean` entre deux tests, sinon la compilation ne sera pas relancée !

Étape 3 : Automatisez la récolte de données de performance (50 minutes)

On souhaite évaluer et ensuite représenter sous forme graphique le temps de compilation avec un nombre croissant de threads. Pour chaque configuration (nombre de threads), une seule mesure est insuffisante : il peut y avoir des variations dû à des facteurs externes (les autres threads présents sur votre machine). Il faut donc obtenir plusieurs (par ex. 5) mesures. Lors du post-traitement on calculera la moyenne et l'écart type de ces mesures (ou d'autres agrégats comme, par exemple, la médiane).

Écrivez un script `bash` qui, de façon autonome, pour chaque configuration entre 1 et 8 threads, exécute 5 fois la compilation avec le nombre de threads et mesure le temps passé.

Le script doit afficher sur `STDOUT` les résultats sous forme d'un tableau au format CSV (comma-separated-values) : https://fr.wikipedia.org/wiki/Comma-separated_values

Voici quelques conseils pour vous aider dans votre tâche :

- La première ligne du CSV en sortie doit indiquer les titres des colonnes.
- Pour mesurer le temps, vous pouvez utiliser l'utilitaire UNIX `time(1)` :
    - Utilisez directement le chemin absolu `/usr/bin/time` pour ne pas risquer d'utiliser la fonction intégrée de `bash` de même nom.
    - `time(1)` rapporte plusieurs temps : à vous de sélectionner celui qui nous intéresse ici !
    - Il existe une option de make qui permet de ne pas afficher les commandes sur STDOUT.
    - `/usr/bin/time` sort par défaut les mesures de temps sur STDERR, il existe une option qui permet de rediriger sa sortie vers un fichier. Attention que par défaut, le fichier est écrasé à chaque appel. Par ailleurs, cette option va également ajouter un retour à la ligne à chaque appel.
    - Si vous obtenez le warning "`ar: `u' modifier ignored since `D' is the default (see `U')`" lors de l'utilisation de `make`, vous pouvez rediriger la sortie STDERR de `make` vers `/dev/null`.
    - Il peut être utile de diviser le problème en plusieurs scripts pour faciliter les redirections des différentes commandes
- On pourrait être tenté d'agréger dès à présent les données récoltées pour n'afficher sur `STDOUT` que la moyenne et l'écart type. C'est une mauvaise idée ! Si on souhaite par exemple calculer par la suite la médiane, l'information est perdue et il faudrait relancer tous les tests. La bonne pratique est donc de conserver l'ensemble des données mesurées et de réaliser le calcul des valeurs statistiques en post-traitement.

Une fois votre script réalisé, collectez les performances finales en redirigeant la sortie `STDOUT` de votre script vers un fichier « `mesures.csv` ». Il est préférable de réaliser cette étape en limitant le nombre de programmes en cours d'exécution risquant d'influencer les résultats, comme, par exemple, votre navigateur web, Discord, etc.

Étape 4 : Écrire le script de post-traitement et de présentation (45 minutes)

Votre objectif dans cette dernière étape est de produire des courbes représentant graphiquement vos mesures. Vous utiliserez à cette fin un script en python. Il n'est pas souhaitable, en effet, d'utiliser C ou un script `bash` pour cela : un langage de haut niveau est bien plus adéquat. Le script de post-traitement réalise trois opérations :

- Il lit et interprète le contenu du fichier csv donné en entrée ;
- Il calcule les données statistiques suivantes :
    - Le temps moyen de compilation avec X threads, avec X = {1..8} ;
    - L'écart type de ce temps pour les 5 (ou +) mesures correspondant à chacune de ces configurations.
- Il utilise la librairie `matplotlib` pour produire les graphes au format `png` ET `pdf`.

Vous trouverez un script d'exemple en python sur Moodle. Pour installer `matplotlib` (et/ou `numpy`) vous pouvez utiliser la commande : `pip3 install matplotlib` (resp. `numpy`).

Les graphes produits respecteront les contraintes suivantes :

- Ils auront un titre général et des titres des axes explicites ;
- L'origine sera toujours (0,0) ;
- Ils présenteront la moyenne ainsi que l'écart type avec des barres d'erreur (vous pouvez réaliser une première version avec seulement la moyenne dans un premier temps, et ajouter les barres d'erreur ensuite).