HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Alberto Dassatti, Bruno
Da Rocha Carvalho

Laboratoire de High Performance Coding

semestre printemps 2024

Laboratoire 5 : Profiling

Temps à disposition : 6 périodes (3 séances de laboratoire).

1 Objectifs de ce laboratoire

L'objectif de ce laboratoire est de se familiariser avec des outils de profiling afin d'analyser les performances d'un programme. Pour cela, vous aller utiliser les outls proposé pour profiler des programmes.

2 Perf

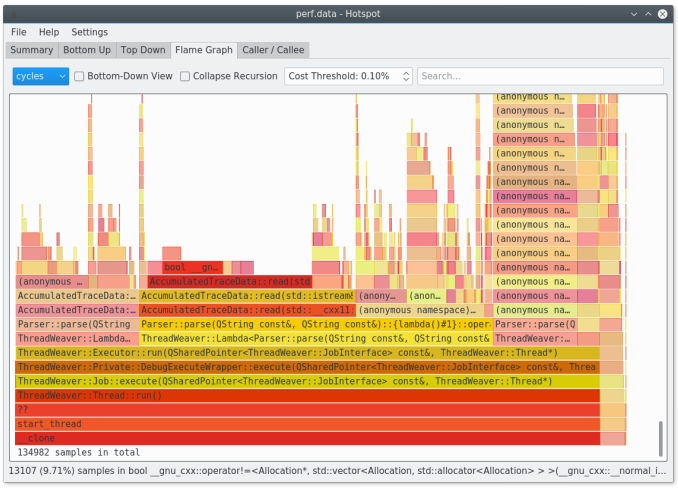
Le kernel Linux implémente des moyens de profiling, accessibles depuis l'outil `perf`. Beaucoup de ressources en ligne décrivent l'architecture et l'utilisation de `perf`. Pour vous familiariser avec cet outil, vous pouvez suivre le tutoriel en trois étapes qui décrit son utilisation, ainsi que consulter cette source d'informations.

Vous pouvez utiliser les commandes `perf stat` et `perf top` pour commencer votre analyse, puis utiliser `perf record` et `perf report` pour approfondir l'analyse.

Pour plus d'informations techniques sur `perf` (facultatif), vous pouvez consulter sa documentation officielle.

3 Hotspot

Vous l'aurez remarqué, `perf` ne permet pas de visualiser facilement les "hotspots" d'un programme qui a été profilé. Ces hotspots peuvent être visualisés sous la forme d'un "flame graph", comme celui proposé par le programme "Hotspot" ci-dessous :



Pour vous familiariser avec Hotspot, vous pouvez lire la page README de son dépôt Git, puis essayer le programme. Vous pouvez effectuer des mesures de cycles mais également de cache misses.

4 Valgrind

Valgrind est un outil sous Linux initialement conçu pour le débogage de l'allocation dynamique de mémoire sur le heap (memcheck). Valgrind a depuis évolué en une suite d'outils (helgrind, cachegrind, callgrind, ...) permettant également de faire du profiling. Pour utiliser les outils cachegrind et callgrind, vous pouvez suivre ce tutoriel : Guide to Callgrind.

Si vous souhaitez utiliser l'outil memcheck, vous pouvez consulter la documentation officielle à cette adresse : Manual for Memcheck (facultatif).

5 Travail à effectuer

5.1 Familiarisation

Afin de vous familiariser avec les outils proposés, nous vous invitons à les tester sur un code que vous connaissez bien : celui de la détection de contour. Amusez-vous à profiler le code et à observer les résultats.

Aucune livraison n'est requise pour cette partie, veuillez donc ne pas y consacrer trop de temps.

5.2 Profiling

Maintenant que vous avez pris en main les outils, nous vous demandons de choisir un programme open source en C qui suscite votre intérêt. L'objectif sera de profiler ce projet avec les outils que vous avez testés.

Vous devrez analyser le projet et, en le profilant, identifier des possibilités d'amélioration ou de

modification.

Pour éviter de vous heurter à des difficultés, choisissez soigneusement le projet avant de commencer à le profiler. Par example, consultez les issues du projet pour avoir une idée de la faisabilité de l'analyse. Venez discutez avec nous, par mail ou directement pendant les séances de labo pour qu'on vous donne notre avis.

Les seules contraintes sont que le programme choisi doit être en C, open source et compatible avec Linux.

6 Travail à rendre

Un rapport devra contenir :

- Une introduction sur le projet choisi
- La description détaillé de l'utilisation que vous allez profiler (baseline).
- Si vous avez dû effectuer des manipulations spéciales pour l'installation (non décrites dans le README du projet), expliquez-les.
- Une analyse approfondie des résultats obtenus avec les outils. Appuyé par des screenshots, des graphiques ou autres.
- Une conclusion sur les résultats
- Enfin, proposer une amélioration et pouvoir la démontrer.