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 2023

Laboratoire 2 : Profiling

Temps à disposition: 4 périodes (2 séances de laboratoire)

Récupération du laboratoire dans l'archive sur Cyberlearn

1 Objectifs de ce laboratoire

Dans ce laboratoire, vous récupérerez le code mis en place dans le premier laboratoire afin d'analyser votre implémentation et de pouvoir commencer à l'améliorer.

2 Cahier des charges

Dans ce laboratoire, vous devrez utiliser l'outil likwid. Likwid est un outil open source de profiling et de monitoring de performance pour les applications exécutées sur des processeurs multi-cœurs. Il permet aux développeurs de comprendre et d'optimiser la performance de leurs applications en fournissant des informations détaillées sur l'utilisation du matériel informatique, telles que la consommation de cache, la latence de la mémoire et la bande passante, ainsi que l'utilisation du processeur et des threads. Likwid peut être utilisé pour mesurer des applications tels que l'affinité des threads, la vectorisation et l'optimisation de la bande passante mémoire.

Avec cet outil, vous allez devoir construire le roofline de votre processeur en fonction de deux facteurs : la capacité de calcul et la bande passante mémoire. Une fois, ce dernier construit, vous pourrez profiler l'exécution de votre programme et analyser où il se situe dans votre roofline. Ainsi vous pouvez exploiter au maximum les performances offertes par votre architecture et saurez où investir pour les améiorer.

3 Installation de likwid

Pour l'installation de likwid :

- Pour les utilisateurs natifs de linux vous pouvez installer le paquet `likwid`. Veuillez vous référer à la documentation qui vous explique comment l'installer.
- Pour les utilisateurs de la VM REDS, `sudo apt install likwid`.
- Si vous avez un autre environnement, on vous invite à utiliser les machines de laboratoires et d'installer likwid en suivant le lien du premier point.

On vous conseille de jeter un œil à leur git pour toute information supplémentaire si vous avez une architecture spéciale, et de manière générale, pour trouver des informations sur tous les outils qu'ils proposent.

4 Roofline

Exploitez les outils fournis par likwid afin de pouvoir dessiner votre roofline. Vous devez faire un banc de test pour trouver le maximum d'opérations à virgule flottante que peut exécuter votre processeur. En lisant les pages man des outils likwid, la documentation de Likwid et avec des recherches sur Internet, vous pouvez trouver les options et les arguments nécessaires pour exécuter ce test de performance (pas nécessaire d'écrire du code).

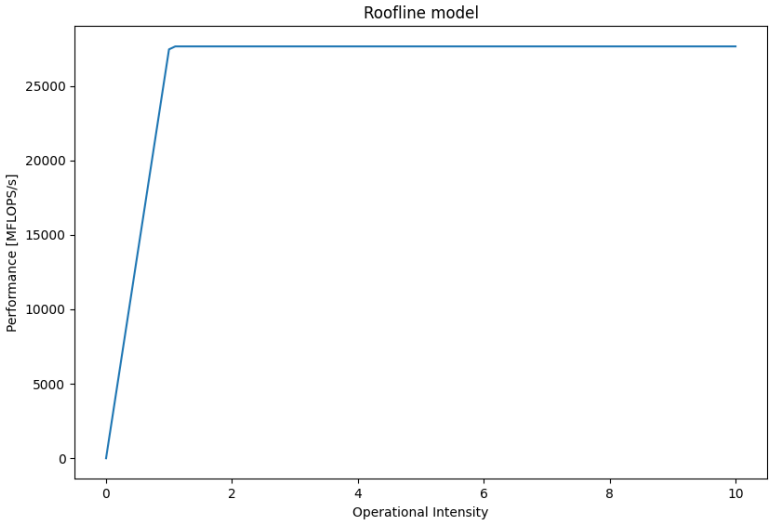
Quelques informations. La taille de votre problème de banc de test n'est pas forcément pertinente. Exécutez le banc de test plusieurs fois afin d'avoir un résultat plus stable.

Une fois votre valeur obtenue, vous allez être intéressé par trouver la bande passante mémoire de votre processeur. Il vous faudra relancer un banc de test afin de pouvoir retrouver cette valeur, mais, cette fois-ci, il vous est recommandé d'observer les capacités des mémoires L1, L2 et L3 afin de pouvoir au mieux les remplir avec la taille de votre banc de test (si vous mettez une valeur trop élevée, le banc de test refusera l'exécution.).

Une fois vos valeurs obtenues, capacité de calcul qu'on appellera `maxperf` et votre bande passante qu'on appellera `maxband`, vous allez pouvoir construire votre graphique du roofline de votre processeur.

4.1 Construction du graphique de performance.

Voici un exemple typique d'un roofline :



Ce graphique est défini tel que:

$$f(x) = x * maxband,$$

$$ssi : x * maxband < maxperf$$

$$sinon f(x) = maxperf$$

Il vous est fourni dans le laboratoire un script python gérant déjà la construction du graphique. Vous devez simplement référer dans un fichier texte `input.txt` les valeurs de maxperf et maxband tel que:

```
maxperf
maxband
```

Ensuite lancez le script et vous devriez voir votre roofline.

5 Profiling de votre code

Maintenant que vous possédez un graphique de référence, il est intéressant de pouvoir y placer des codes afin de voir si l'on exploite au mieux la capacité de votre architecture. La librairie likwid offre la possibilité d'analyser du code, à des endroits précis, pour nous permettre de nous rendre compte si l'on exploite efficacement notre architecture. Votre but et de récupérer le code que vous avez fait pour le laboratoire n°1, de le recompiler avec le nouveau Makefile qui vous est fourni et d'exploiter les marqueurs de likwid pour cibler les parties de code à mesurer.

Pour utiliser les marqueurs, il vous suffit de rajouter un include `<likwid-marker.h>` Et ensuite d'englober les parties de code que vous souhaitez analyser avec les marqueurs :

```
LIKWID_MARKER_INIT;
...
LIKWID_MARKER_START("name");
... <- code a analyser
LIKWID_MARKER_STOP("name");
...
LIKWID_MARKER_CLOSE;
```

Un tutoriel plus détaillé est disponible ici

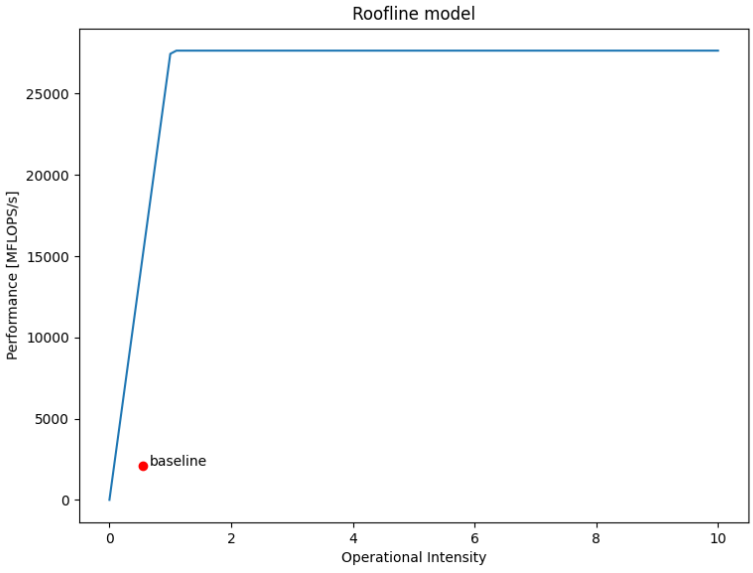
Une fois votre parcelle de code encadrée par les marqueurs, vous allez utiliser l'outil d'analyse de performance de likwid. Faites bien attation d'utiliser le bon groupe de surveillance ainsi que de bien spécifier un identifiant de processeur. Comme précédemment lisez bien le `man` de l'analyseur de performance pour retrouver facilement toutes les informations que vous avez besoin.

Une fois l'analyse terminée, vous devrez reporter vos valeurs dans le graphique afin de situer où se situe votre algorithme. Rajoutez au fichier `input.txt` les valeurs d'operational intensity (abscisse) et le nombre d'opérations à virgule flottante(ordonnée) afin de pouvoir les visualiser, nommez cette mesure "baseline".

Vous devrez typiquement vous retrouver avec un fichier comme :

```
27648.63
27452.33
5.522620e-1 2127.4496 baseline
```

Et une fois lancé, vous pourrez apercevoir où se situe votre programme. À noter que vous pouvez ajouter plusieurs points en même temps en continuant d'en ajouter à la suite dans le fichier `input.txt`.



6 Travail à rendre

Maintenant, que vous avez un environnement prêt pour profiler votre programme, vous devrez développer deux ou trois versions optimisées de A* à l'aide des informations que vous pouvez obtenir du graphique et de likwid. Vous devrez reporter toutes vos analyses et vos essais dans un rapport et rendre votre code (optimisé) et vos explications.

Le rapport doit être court et concis et au format markdown ou pdf.