TOI — TP Profilage (noté)

Rendu à john.gliksberg@uvsq.fr

Travail attendu

Dans ce TP j'attends la production d'un rapport au format PDF. Le rapport doit refléter la prise en main et la compréhension de l'élève. J'attends donc un rapport par élève.

Assurez-vous d'avoir bien compris les questions, vérifiez vos analyses, montrez votre reflexion. Soyez généreux dans votre compte-rendu sans pour autant écrire un roman.

Exercice 0 : prise en main du profileur

- Mettez-vous à l'aise avec plusieurs profileur :
  - *perf*(1) (avec `perf record -g`),
  - ainsi que Callgrind (avec `valgrind --tool=callgrind`).
- Analysez les profils à l'aide de leur outils correspondants :
  - `perf report`,
  - Hotspot,
  - ou bien KCacheGrind

1. Écrivez un « Hello World » en C et en C++. Profilez leurs compilations respectives via `gcc` et `g++`. Explorez rapidement les données, et faites une comparaison générales des deux profils. Justifiez vos propos à l'aide de chiffres ou des captures d'écran.

   (Bonus) Comparez avec un autre compilateur et/ou un autre langage compilé.

Sujet du TP : Kevin veut optimiser son code

Kevin a encore frappé ! Après avoir accepté vos modifications sur son repo, il a enfin un « Mandelbrot » fonctionnel. Les résultats sont éblouissants, surtout grâce à la couleur.

Ça lui a donné plein d'idées de fonctionalités à ajouter, mais le code met environ 5s pour générer une image 1500x1500. Du 1K à 0.2 fps, ce n'est pas glorieux.

Le repo suivant contient plusieurs branches de développement proposant des modifications pour tenter d'accélérer le code :

`git clone` https://github.com/trosh/ATOI24_TP_Profilage.git

Exercice 1 : qu'a-t-on cherché à faire ?

1. En explorant le repo, expliquez l'intention des modifications effectuées.
2. Potentiellement, émettez des hypothèses sur leur impact réel :
   a. en terme de performance,
   b. en terme de génie logiciel, ou autre.

Exercice 2 : qu'a-t-on vraiment fait ?

- Pour chaque version du code, mesurez le temps horloge d'exécution du programme sur votre machine (par exemple, à l'aide de *hyperfine*(1) ou bien juste de *time*(1)). Dans votre rapport, mettez ces résultats en forme ; à l'aide d'un tableur par exemple.
- Pour chaque version du code, générez le profil d'une exécution, puis déterminez la nature des changements.
- Commentez les résultats en contraste de vos hypothèses, à l'aide des données réelles.
  - Notez si certaines valeurs fournies par l'outil de profilage sont à prendre avec précaution.

Exercice 3 : c'est embarassant

1. Y a-t-il des dépendances entre itérations de la double boucle principale (sur *i*/*j*) ?
2. Utilisez OpenMP pour paralléliser la boucle extérieur (sur *i*) :

   ```
   #pragma omp parallel for
   for (...) {
   ```

   Mettez à jour le Makefile, et suivez vos modifications avec Git. Un indice : les pragmas sont ignorés sans la bonne incantation . . .
3. À l'aide de Hyperfine et Hotspot, décrivez l'impact sur les performances en général, ainsi que l'utilisation des threads au cours du temps. Expliquez ce comportement.
4. (Bonus) Peut-on faire encore mieux ? Voir la clause Schedule.
   a. Quels sont les impacts négatifs potentiels de cette technique ? Peut-on calculer le surcoût associé ?