TP 5 (2025-11-28) — Profilage (noté)

Rendu papier à la fin de la séance

Fichiers/liens ⟶ john.gliksberg@uvsq.fr

Sujet du mail: `"TOI/TP5 - <Nom complet>"`

Décrivez votre réflexion — rédigez pour être lisible par un humain

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 `mandelbrot.pack` contient plusieurs branches de développement proposant des modifications pour tenter d'accélérer le code. Je vous fait confiance pour le cloner d'une manière adéquate.

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

1. En explorant le repo, expliquez l'intention des modifications effectuées dans les branches autres que `master` et `zoom`.
2. Potentiellement, émettez des hypothèses sur leur impact réel ; en terme de performance, 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 tableau 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.
   1. Quels sont les impacts négatifs potentiels de cette technique ? Peut-on calculer le surcoût associé ?