Travaux Pratiques de Programmation Orientée Matériel

Arthur Bit-Monnot

1 Installation de l'environnement de TP

1.1 Communication par Discord

Pour communiquer, nous utiliserons un serveur Discord afin que vous puissiez nous exposer vos problèmes techniques avant la séance de tp. Pendant la séance de TP, nous utiliserons le même serveur pour vous guider dans la réalisation du TP. Afin de permettre le partage d'écran, nous vous demandons de ne pas utiliser la version web mais d'installer le client desktop.

Étape 1.A *Installer un client desktop pour Discord : https://discordapp.com/*

Étape 1.B *Se connecter sur le serveur du cours : https://discord.gg/Gy3DJAE*

En cas, de problème pour la suite de l'installation, vous pouvez poster une demande d'aide dans le channel *problèmes-techniques*. Vérifiez avant que la solution à votre problème n'a pas déjà été donné à quelqu'un d'autre.

1.2 Suivi sur Google Docs

Pour nous permettre de suivre votre avancement dans le TP, nous vous demandons de remplir à chaque étape franchie du TP le document partager suivant : https://docs.google.com/spreadsheets/d/1wuIKDgLwOPMaqFLggxS9kuKyw_B3NZrFJ8F71R5jKgI/edit?usp=sharing

Étape 1.C *Dans ce document, vérifiez que vous apparaissez bien, indiquez votre login insa pseudo et votre pseudo sur Discord.*

Étape 1.D *Indiquez les étapes franchies à l'aide d'un* `X` *dans la case correspondante.* ***À chaque nouvelle étape ou question franchie, rajoutez un*** `X` ***dans le document.***

1.3 Machine Linux

Pour l'installation de l'environnement de TP vous aurez besoin d'une machine linux avec l'environnement de TP. Si vous avez déjà une machine linux fonctionnelle vous pouvez sauter l'étape suivante.

Étape 1.E ***Installation d'une machine virtuelle.***

Suivez les instructions "Installation par le biais d'une machine virtuelle" sur le WikEtud : https://wiki.etud.insa-toulouse.fr/index.php?title=Installer_l%27environnement_de_TP_d%27ADA#Installation_par_le_biais_d.27une_machine_virtuelle

- *Procédez à l'installation de virtualbox et de la machine virtuelle. La machine Lubuntu proposée pour la première année est plus légère et suffit pour nos besoins.*
- *Notez le mot de passe*
- *Optionnellement, regardez la deuxième partie du tutoriel (Outils pratique...).*

Tout le reste de la procédure doit être suivie dans la machine virtuelle (ou sur votre système linux si vous avez choisi une autre méthode).

Étape 1.F *Installez les outils de développement nécessaires :* `git` *et* `gcc`*. Dans un terminal, exécutez ;*

```
sudo apt install git build-essential
```

Étape 1.G *Récupérez les sources du TP et vérifiez que tout fonctionne :*

```
git clone https://github.com/arthur-bit-monnot/tp-archi-mat.git
cd tp-archi-mat
make
./base 10
```

La dernière ligne de cette commande doit être `Execution successfull`

Bravo, tout est opérationnel ! Indiquez votre progression dans le document partagé. Vous êtes prêt pour la séance de TP. Si vous avez eu un problème sur l'une des étapes, demandez de l'aide sur le serveur Discord.

2 Contexte du TP

2.1 Avant toute chose

Étape 2.A *Avant toute chose, assurez vous d'avoir la dernière version du TP. Dans le repository que vous avez précédemment récupéré :*

```
git pull
```

Nous vous fournissons en template pour vos réponses, sous forme de GoogleDoc. L'objectif est que vous puissiez y inscr

Étape 2.B

- *Récupérez le template pour les réponses : https://docs.google.com/spreadsheets/d/1bP3OqOgPtKQWtz5r-HK5QDSy5QGfJkPdHq1KWjwnzBg/edit?usp=sharing*
- *Créez votre propre copie :* `File -> Make a Copy`
- *Récupérez un lien partagé vers votre rapport.* `Share -> Get Shareable link ->`

Anyone with this link can edit.

- *Indiquez le lien vers votre copie dans le document d'avancement (section 1.2).*

Ce document a pour vocation à recueillir des réponses succinctes aux questions du TP. Il ne servira pas dans le cadre d'une cadre d'une évaluation mais seulement pour nous permettre de juger rapidement de votre avancement et compréhension. Il est partagé en écriture pour que nous puissions facilement vous indiquer la validité d'une réponse.

2.2 Contexte

L'objectif du TP est d'optimiser l'implémentation d'une base de données.

L'interface (API) que doit avoir la base de données est spécifiée dans le fichier `api.h` et le fichier `base.c` contient l'implémentation actuelle que vous êtes chargés d'optimiser.

Afin de mesurer les performances de votre implémentation, il vous est fourni un programme de test `main.c`. Le projet est fourni avec un `Makefile`, celui-ci vous permet de compiler le programme de test (et votre implémentation) à l'aide la commande `make`.

3 Gestion de la mémoire

3.1 Familiarisation avec le code

Ouvrez le fichier `api.h` et analysez la structure de données `struct Person`. Celle-ci représente un élément de votre base de données. Regardez également les différentes fonctions déclarées dans le fichier. Ces fonctions correspondent à des requêtes faîtes sur une base de données que vous devrez implémenter ou améliorer.

Question 3.A *Identifiez la taille de chacun des champs composant une structure personne. Quelle est la somme de la taille de tout ces champs ?*

Question 3.B *La taille d'une structure `Person` est de 32 octets. Pourquoi est-ce différent de la somme des tailles des champs ?*

Ouvrez le fichier `base.c` qui contient l'implémentation actuelle de la base de donnée.

Question 3.C *Quelle est la structure en mémoire de la base de données ?*

Question 3.D *Quelle est la mémoire occupée par la base de données pour X éléments ? Quel est le nombre maximum d'élément que vous pouvez garder en RAM sur votre machine ? (vous pouvez trouver la taille de la RAM avec la commande `free`)*

Ouvrez le fichier `main.c`. Celui-ci contient du code pour générer des données et faire des tests de performance sur la base de données. Vous pouvez ignorer la plupart du code dans ce fichier et vous concentrer uniquement sur le contenu de la fonction `main`.

Une fois le projet compilé (avec `make`) vous pouvez lancer le test de performance avec la commande

```
./base N
```

où `N` est en entier entre 6 et 27. Le nombre d'éléments dans la base de données sera de 2^N.

3.2 Passage à l'échelle

Les benchmarks contiennent deux tests qui cherche à accéder à grand nombre d'éléments de la base de données : `sequential_lookup` et `random_lookup`. Regardez dans le fichier `main.c` à quoi correspondent ces tests.

On souhaite étudier le passage à l'échelle de ces benchmarks en fonction du nombre d'éléments dans la base de données.

Étape 3.E *Tracez les courbes du temps par opération pour ces deux benchmarks pour N allant de 6 à 24.*

Pour `sequential_lookup` comme pour `random_lookup`, chaque opération consiste uniquement à récupérer en mémoire une structure `Person`.

Question 3.F *À votre avis, pourquoi les temps augmentent pour les accès aléatoires ?*

Trouvez la taille des caches sur votre machine (commande `lscpu`).

Question 3.G *Expliquez les résultats à partir de la proportion de données pouvant entrer dans chaque niveau de cache. Pouvez vous identifier des décrochages dans les courbes et les faire correspondre à des tailles de cache.*

Question 3.H *Comment expliquez-vous les résultats différents pour l'accès séquentiel ?*

3.3 Alignement mémoire

On rappelle que sur les processeurs Intel modernes une ligne de cache fait 64 octets et que, par conséquent toutes les lignes de cache commencent à une adresse multiple de 64.

Question 3.I *Combien de lignes de cache sont nécessaires pour contenir une personne ? Au minimum ? Au maximum ?*

Question 3.J *En modifiant la fonction `db_init()` dans `base.c`, vérifiez l'adresse des 2 premiers éléments de la base de données. Tiennent-t'ils bien une ligne de cache ?*

Vérifiez ceci pour différents N.

Note : vous pouvez lire et modifier la valeur d'une adresse en la convertissant en entier, puis en la reconvertissant en adresse :

```
Person *p = ...;
unsigned long address = (unsigned long) p;
printf("address: %lu\n", address);
// can change address here
p = (Person *) address;
```

Question 3.K *Modifiez la fonction* `db_init()` *pour s'assurer que tous les éléments tiennent sur une seule ligne de cache.*

Note : si besoin, vous pouvez augmenter la mémoire allouée dans le malloc pour être sur de ne pas dépasser.

Question 3.L *Mesurez l'impact en performance sur les accès séquentiels et aléatoires pour une taille de base de donnée assez grosse (e.g. $N = 23$). Indiquez le gains moyen en pourcentage.*

4 Optimisation de requêtes

4.1 Counting

On considère ici la fonction `db_count_male()` dans `base.c`

Question 4.A *En ne prenant en compte que les opérations arithmétiques (addition, comparaison...), donnez une estimation du nombre de cycle par itération de la boucle.*

Question 4.B *En vous rapportant à la fréquence de votre CPU, combien de temps devrait prendre ces opérations ?*

Question 4.C *Quelle est la quantité de mémoire parcourue par* `db_count_male` *(pour un N donné) ? Comparez cette valeur et le temps d'exécution pour avec celle des accès séquentiels. Quelle semble être le facteur limitant dans cette procédure ?*

4.2 Distances

On cherche à analyser les différentes opérations dans la boucle de `db_closest` et en particulier de cette sous partie :

```
float dx = lat - db->persons[i].latitude;
float dy = lon - db->persons[i].longitude;
float dist = sqrtf(dx * dx + dy * dy);
```

Question 4.D *Quelles sont les dépendances de données entres les opérations ?*

Pour vous aider à identifier les différentes instructions et trouver leur coût, vous pourrez regarder l'assembleur généré pendant la compilation (`base.o.asm`) et sur la table des instructions : https://www.agner.org/optimize/instruction_tables.pdf .

Pour sélectionner votre processeur, vous pouvez trouver son identifiant (par exemple i7-10510U) avec la commande `lscpu`. Une recherche google avec cet identifiant devrait vous permettre d'identifier sa génération.

Question 4.E *Identifiez l'instruction assembleur des opérations utilisées ici (addition, soustraction...). Pour chacune d'elle, indiquez sa latence et sa* reciprocal throughput.

Question 4.F *Quelle est l'instruction la plus coûteuse ? Comment pourrait-on s'en passer ?*