Travaux Pratiques de Programmation Orientée Matériel

2019–2020 / 3 MIC/IMACS

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*

***Utilisez votre prénom et nom comme pseudo** (ça nous facilitera beaucoup la tâche).*

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, dans la feuille de votre groupe, créez une colonne pour vous, indiquez votre Nom Prénom et login insa. Veillez à les inscrire dans un ordre alphabétique.*

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. Il est également possible de le faire sous MacOS (indications dans le chan *problèmes-techniques*)

Étape 1.E ***OPTION 1: Installation du Windows Subsystem for Linux (WSL)***

Suivre les instructions ci-dessous (nécessite d'être sur Windows 10) : https://docs.microsoft.com/fr-fr/windows/wsl/install-win10

Étape 1.F ***OPTION 2: 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...).*

N'utilisez pas la connexion distante à l'INSA, (VPN + X2GO) sauf en dernier recours. Si c'est le cas, demandez-nous dans le chan *problèmes techniques*.

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

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

Note : pas nécessaire si vous êtes connecté sur les serveurs de l'INSA.

Étape 1.H *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 successful`

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/1TJ0wy9gWRDQApvYI6HaQzuVaE9qZ4ipcgqkHzyYY8cU/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.`
- *Renommez votre fichier pour qu'il commence par votre Prénom et Nom.*
- *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
- *Pour chacun des champs, indiquez sur combien d'octets il doit être aligné pour permettre son traitement par le processeur.*
- *En supposant qu'il n'y a pas d'octets de bourrage, quels sont les champs qui ne sont pas alignés ?*
- *Combien d'octets de bourrage doit on insérer pour permettre l'alignement de ces champs ? Où ça ?*

Note : on vérifiera que la taille de la structure fait bien 32 octets après prise en compte du bourrage.

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 en octets de la RAM avec la commande* `free --bytes`*)*

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`. On mesure donc le temps (en nanosecondes) pour aller récupérer une structure personne en mémoire.

Étape 3.F *Trouvez la taille des caches sur votre machine (commande* `lscpu`*).*

Si vous êtes sur Windows Subsystem for Linux (WSL) : les informations se trouvent dans le gestionnaire des tâches Windows. Clic droit sur la barre windows > Gestionnaire des tâches > Onglet "performances" > "Processeurs" et vous verrez "Cache de niveau [1, 2 ,3]". Ce qui correspond à L1, L2, L3

Question 3.G *Pour le cas* `random_lookup`*, pourquoi le temps d'une opération augmente avec la taille de la base de données ?*

Essayez de faire correspondre les décrochages dans la courbe avec le moment où la base de données ne tient plus dans les caches L2 et L3.

Question 3.H *Contrairement, aux accès aléatoires, le temps par opération des accès séquentiels augmente très peu avec la croissance de la base de données.*

Qu'est ce que le processeur peut faire ici pour éviter les fautes de caches ? Pourquoi n'est-ce pas possible pour les accès aléatoires ?

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 ?*
- *Dans quelles conditions est-ce qu'une structure* `Person` *sera à cheval sur deux lignes de cache ?*

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 données assez grosse (e.g. N = 24). 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...) que l'on suppose nécessiter de l'ordre d'un cycle CPU chacune, donnez une estimation grossière 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 ? (estimation grossière)*

Le temps de traitement de ces opérations arithmétiques semble-t-il être le facteur limitant pour la performance de cette méthode ?

On notera ici que pour accéder au champ `male` de la structure, il est nécessaire de charger dans le cache L1 l'ensemble de la ligne de cache le contenant (dont les 64 octets contiennent 2 `Person`). L'ensemble de la base de données sera donc chargé en cache L1 ce qui explique des performances du même ordre de grandeur que pour les accès séquentiels qui eux aussi vont charger l'ensemble de la base dans le cache : la performance est limitée par les accès mémoires.

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.C *Quelles sont les dépendances de données entres les opérations ?*

Pour les représenter graphiquement, vous pouvez vous inspirer de cette représentation : http://homepages.laas.fr/abitmonnot/files/teaching/prog-mat/data-deps.jpg

Vous trouverez ci-dessous un extrait du fichier `base.o.asm` qui donne la représentation en assembleur (syntaxe AT&T) du code C ci-dessus (il se peut que cet assembleur soit différent sur votre machine).

```
movl    %ebx, %r12d
movq    %rbx, %rdx
salq    $5, %rdx
addq    0(%r13), %rdx
movss   8(%rsp), %xmm0
subss   24(%rdx), %xmm0
movss   12(%rsp), %xmm1
subss   28(%rdx), %xmm1
```

```
mulss   %xmm0, %xmm0
mulss   %xmm1, %xmm1
addss   %xmm1, %xmm0
sqrtss  %xmm0, %xmm5
```

Pour vous aider à identifier les différentes instructions et trouver leur coût, vous pourrez utiliser la table de coût des instructions pour différents processeurs : https://www.agner.org/optimize/instruction_tables.pdf.

Question 4.D *Identifiez l'instruction assembleur des opérations arithmétiques utilisées ici (addition, soustraction, multiplication et racine carrée). Pour chacune d'elle, indiquez sa latence et sa* reciprocal throughput *pour un processeur Sandy Bridge (votre processeur aura vraisemblablement des caractéristiques très proches)*

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

5 Struct of Arrays

Partant du constat que l'on arrive pas à compter les individus masculins suffisamment rapidement, un collègue a pris sur lui de restructurer la base de données. Il a créé une nouvelle structure de données interne pour la base de données et a mis son implémentation dans `arrays.c`. Malheureusement il est parti avant d'avoir pu implémenter la moindre de requête.

Le `Makefile` contient aussi les spécifications pour compiler ce programme et produit l'exécutable `./arrays`. Pour en activer la compilation, ajoutez la cible `arrays` dans le champ `TARGETS`.

Question 5.A
- *Analysez sa structure de données et mesurez les performances sur les accès séquentiels et aléatoires.*
- *Qu'est ce qui explique les performances de cette structure ?*
- *À votre avis, pourquoi a-t-il fait ce choix là ?*

5.1 Implémentation des requêtes

On cherche à implémenter les requêtes manquantes, d'abord dans une première version qui correspond à l'implémentation initialement trouvée dans `base.c`.

Quand vous faites tourner les programmes `./arrays` et `./base`, l'avant-dernière ligne affiche le résultat des requêtes. Vous pouvez l'utiliser pour comparer que votre implémentation donne bien le même résultat.

Question 5.B *Faîtes une première implémentation de la requête* `db_count_male()`*, calquée sur celle de base.c. Que constatez vous en terme de temps de calcul.*

Question 5.C *L'implémentation naïve contient un aléas de contrôle dans la boucle. À quoi correspond-il ? Comment pourrait-on l'éliminer ? Proposez une optimisation et mesurez les performances.*

Question 5.D *On rappelle que la supposition peut-être faite que le nombre d'éléments dans la base de donnée est un multiple de 64. Appliquez maintenant l'optimisation du déroulage de boucle vue en cours.*

Cherchez maintenant à optimiser au maximum chacune des fonctions présentes. Vous chercherez en particulier à limiter au maximum les différents aléas du pipeline (structurels, de données et de contrôle).

5.2 Pour aller plus loin : SIMD

Avant toute chose, ouvrez le Makefile et ajoutez l'option `-march=native`. Cette option spécifie au compilateur de spécialiser le code généré pour votre machine en particulier.

En commençant par la request `db_max_age()`, utiliser les instruction SIMD pour paralléliser vos requêtes.

Vous pourrez pour cela vous appuyer sur les pages suivantes :

- http://0x80.pl/notesen/2018-10-03-simd-index-of-min.html
- https://www.cs.virginia.edu/~cr4bd/3330/F2018/simdref.html

Un peu d'aide :

- l'utilisation d'instruction SIMD requiert typiquement que la mémoire accédee soit alignée sur la taille du bus de données (qui peut être 128, 256 ou 512 bits selon les processeurs). Vous pouvez utiliser la fonction `memalign` à la place de `malloc` pour allouer de la mémoire alignée. pour plus d'infos : `man memalign`
- Vous aurez besoins des *intrinsics* pour les instructions SIMD sur les processeurs Intel se trouvent dans les headers suivants. (d'autre headers peuvent être nécessaires pour des instructions flottantes ou d'autres jeux d'isntruction).

```
#include <smmintrin.h>
#include <immintrin.h>
```