Université de Versailles Saint-Quentin-en-Yvelines
Institut des sciences et techniques des Yvelines
M1 Calcul Haute Performance et Simulation

NOM et PRÉNOM: KEDDAD YOUCEF

N° ÉTUDIANT: 22205416

Rapport de Projet

Le but de ce Projet est la mesure des performances lors de l'exécution des plusieurs programmes en utilisant différents compilateurs, Pour se faire on passe par:

.Les prérequis:

1.Pour assurer que le CPU tourne a une fréquence stable:

```
youcef@youcef-HP-Laptop-15s-eq2xxx:~$ sudo cpupower -c 3 frequency-set -g performance
Setting cpu: 3
```

2.Pour piner le processus sur un cœur de calcul:

```
youcef@youcef-HP-Laptop-15s-eq2xxx:~$ taskset -c 3 <PROG>
```

3. le PC est connecté au secteur.

0.CPU INFOS:

les informations sur le CPU sont stockées dans le dossier **CPU_Info**

1.

On choisit un compilateur et on exécute le Makefile par:

```
make CC=$Compiler
```

Après on lance notre programme:

```
taskset -c 3 ./dgemm 100 100 > perf_${compiler}.dat

sed -n '1p;2p' perf_${compiler}.dat > perf_IJK_${compiler}.dat
sed -i '1d' perf_IJK_${compiler}.dat
cut -d';' -f-1,11 perf_IJK_${compiler}.dat > plot_${compiler}.dat

sed -n '1p;3p' perf_${compiler}.dat > perf_IKJ_${compiler}.dat
sed -i '1d' perf_IKJ_${compiler}.dat
cut -d';' -f-1,11 perf_IKJ_${compiler}.dat >> plot_${compiler}.dat

sed -n '1p;4p' perf_${compiler}.dat > perf_IEX_${compiler}.dat
sed -i '1d' perf_IEX_${compiler}.dat
cut -d';' -f-1,11 perf_IEX_${compiler}.dat >> plot_${compiler}.dat

sed -n '1p;5p' perf_${compiler}.dat > perf_UNROLL_${compiler}.dat
sed -i '1d' perf_UNROLL_${compiler}.dat
cut -d';' -f-1,11 perf_UNROLL_${compiler}.dat >> plot_${compiler}.dat

sed -n '1p;6p' perf_${compiler}.dat > perf_UNROLL8_${compiler}.dat
sed -i '1d' perf_UNROLL8_${compiler}.dat
cut -d';' -f-1,11 perf_UNROLL8_${compiler}.dat >> plot_${compiler}.dat
```

les commandes ci dessus permettent de lancer le programme sur le cœur de calcul 3 et récupérer les mesures de performances dans un fichier data. Ensuite, séparer chaque version de calcul (IJK,IKJ,IEX,UNROLL,UNROLL8,CBLAS) dans un fichier a part et créer un fichier data pour ploter des histogrammes de comparaison.

On répète la même procédure en changeant le compilateur(gcc,clang) et en testant a chaque fois un autre flag d'optimisation O1,O2,O3.

L'étape suivante sera de tracer les histogrammes à partir des fichiers .dat pour comparer entre les versions de calcul, les flags d'optimisation et les compilateurs en utilisant cette commande.

gnuplot plot.gp

Exemple d'un fichier .gp:

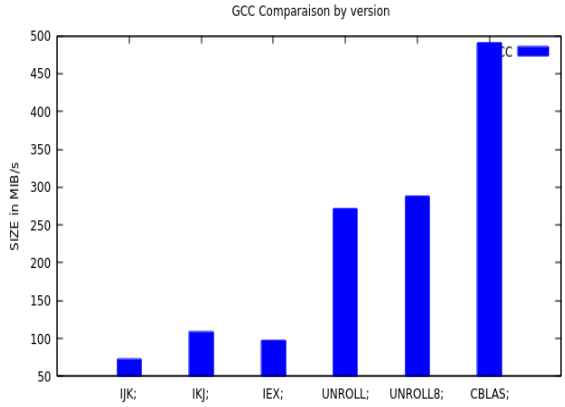
```
set style data histograms
set title "GCC Comparaison by version"
set ylabel 'SIZE in MIB/s'
set style fill solid
plot './plot_gcc.dat' using 2:xtic(1) linecolor 'blue' title 'GCC'
pause -1
```

Résultats et discussion des mesures

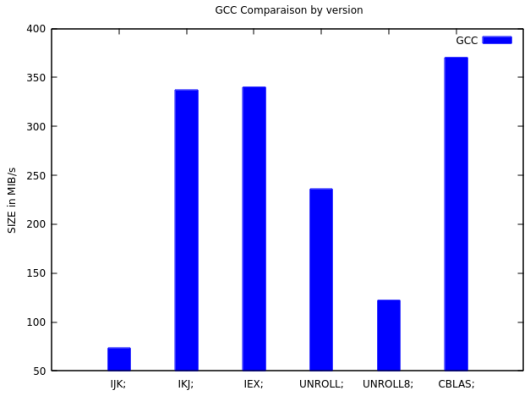
Dans cette comparaison on compare toujours la vitesse d'exécution (Mib/s) en fonction de (version de calcul)

I)-Dgemm:

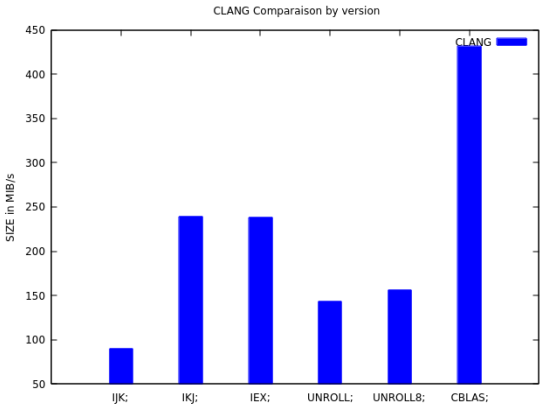
1. Par version:



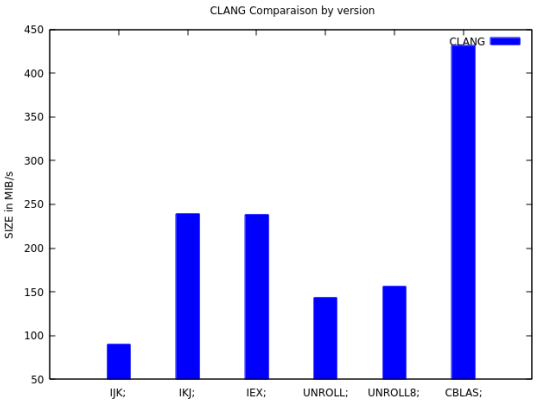
gcc_o1



gcc_o2



clang_o3



clang_o2

On remarque :

1.'CBLAS' est toujours la meilleur version et 'IJK' toujours en dernière place.

C'est logique parce-que CBLAS est une bibliothèque qui contient des fonctions réalisant des opérations de base de l'algèbre linéaire et ils sont développées de manière très optimisée.

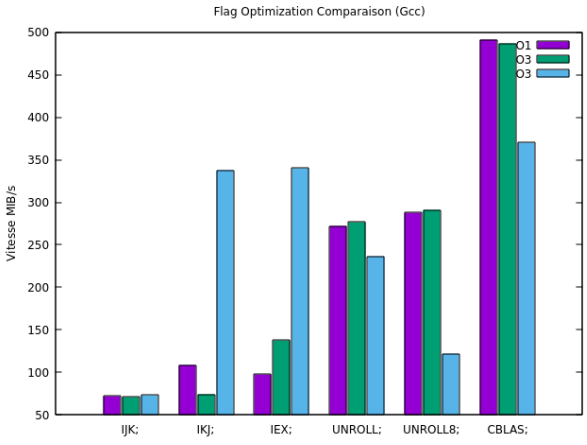
Par contre la version IJK est la version basique qui ne contient aucune optimisation.

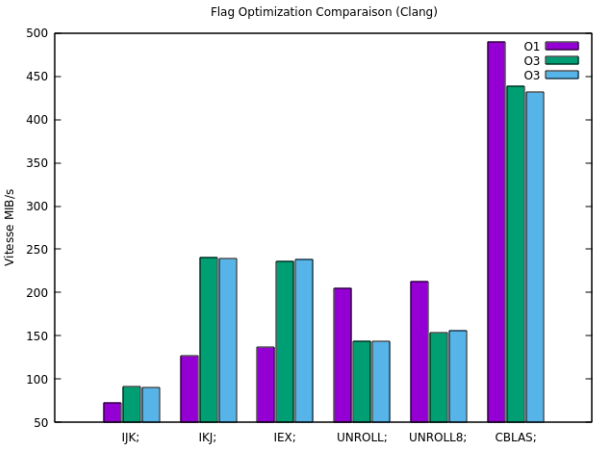
2.'IKJ' et 'IEX' sont mieux que 'UNROLL'.

Les versions IKJ et IEX ont des optimisations comme 'changement du boucle interne' et 'extraction invariante'.

3.'UNROLL8' est légèrement mieux que 'UNROLL4'.

2. Par flag d'optimisation:



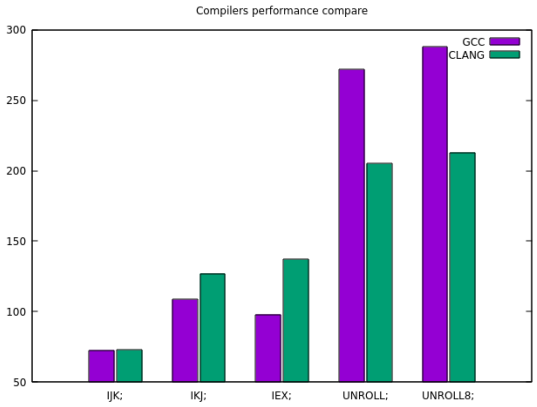


On remarque:

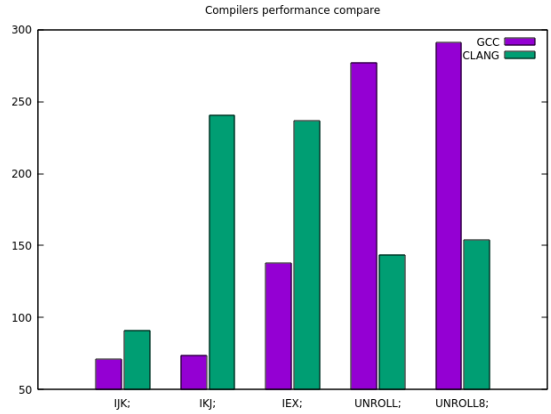
1.Pour IJK,IKJ,IEX: O3 est légèrement mieux que O2 mieux que O1.

2.Pour UNROLL4,UNROLL8,CBLAS: O1 est légèrement mieux que O2 mieux que O3.

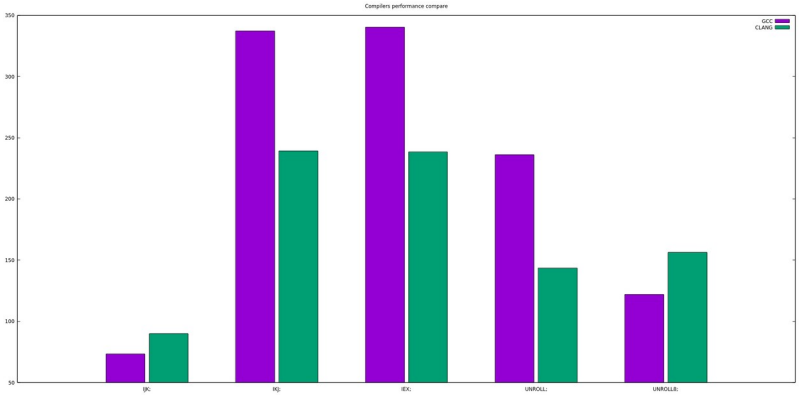
3. Par Compilateur:



o1



o2



o3

On remarque:

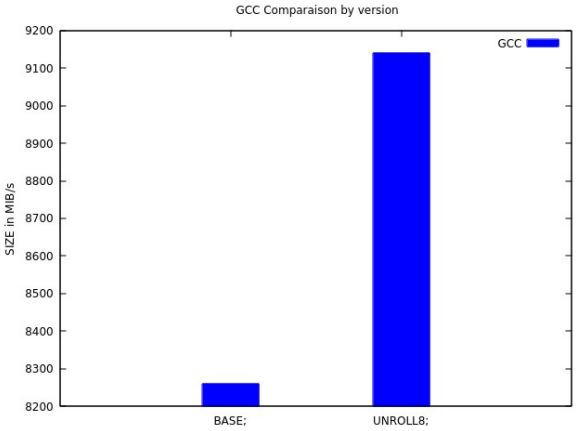
1.Pour IJK,IKJ,IEX: Clang est mieux que Gcc.

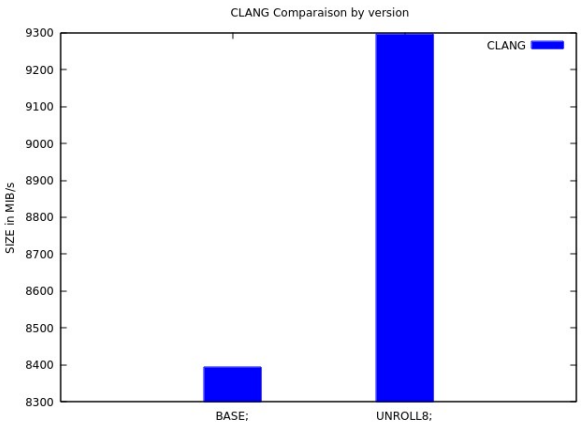
2.Pour UNROLL4,UNROLL8: Gcc est mieux que Clang.

En général Gcc donne des bons résultats avec le flag d'optimisation O3 et Clang donne des bons résultats avec O1 et O2.

II)-Dotprod:

1. Par version:



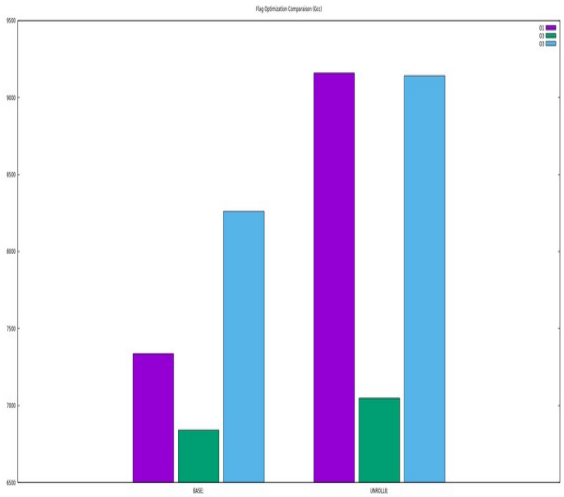


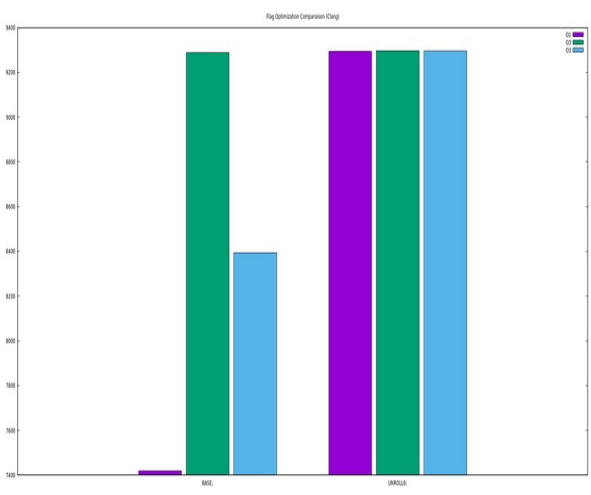
On remarque :

1.UNROLL8 est mieux que BASE.

La version UNROLL8 contient des optimisations contrairement au version BASE.

2. Par flag d'optimisation:



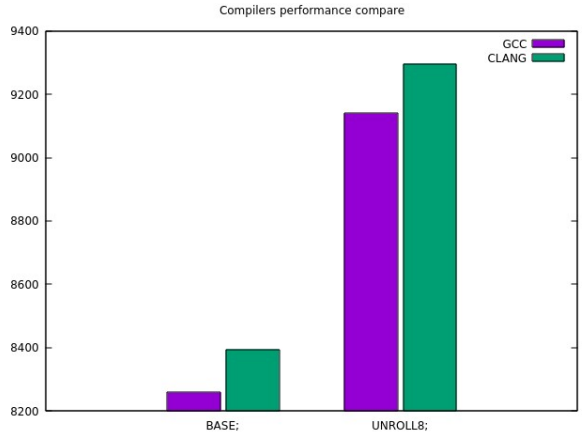


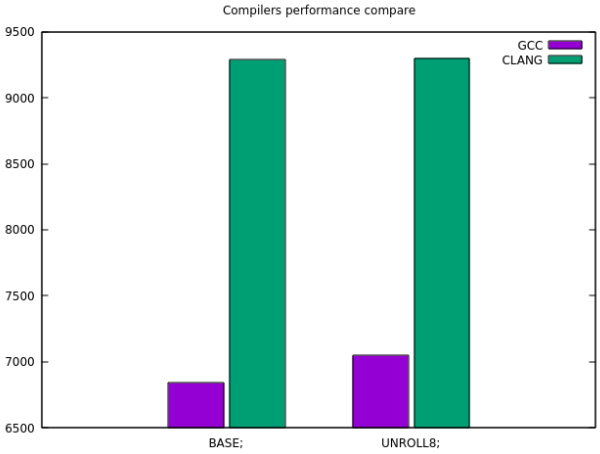
On remarque:

1.Pour Gcc: O3 est le meilleur flag.

2.Pour Clang: le meilleur flag est O2.

3. Par Compilateur:





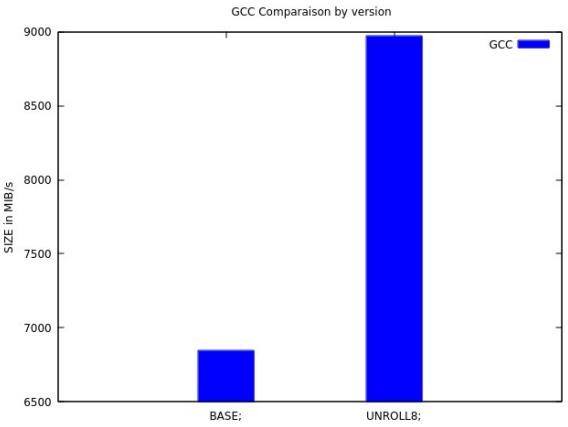
On remarque :

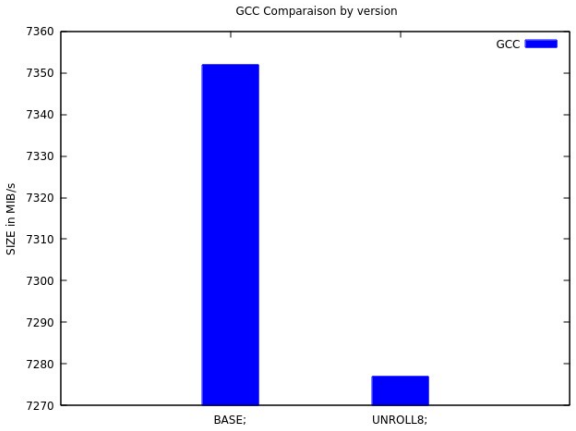
1.Clang est clairement mieux que Gcc.

Pour Dotprod Clang est le meilleur compilateur.

II)-Reduc:

1. Par version:



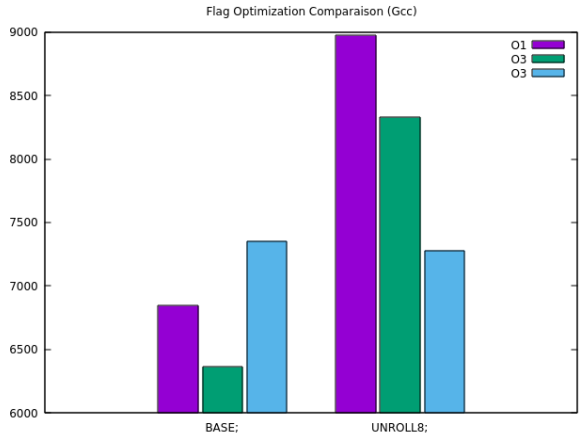


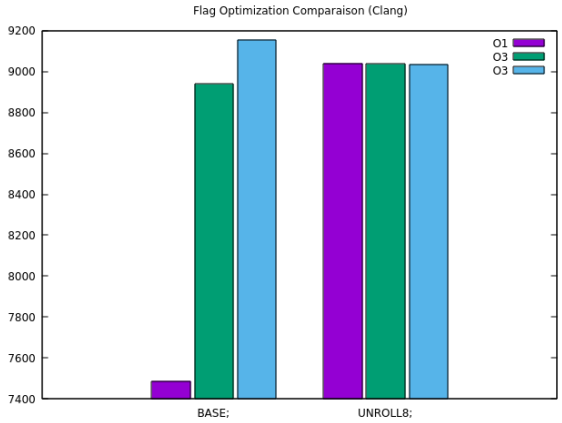
On remarque:

1.UNROLL8 mieux que BASE avec le flag O1.

2.BASE est mieux que UNROLL8 avec le flag O3.

2. Par flag d'optimisation:



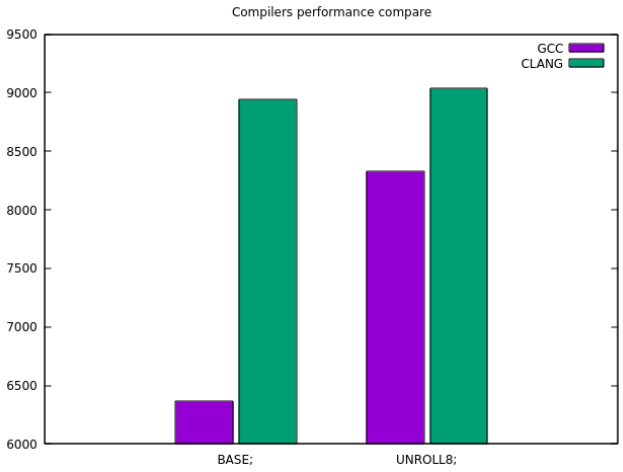


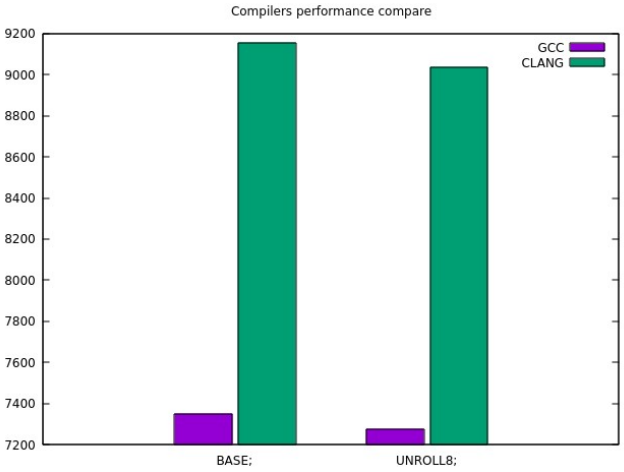
On remarque:

1.'BASE': O3 est le meilleur flag d'optimisation.

2.'UNROLL8': O1 est légèrement mieux que O2.

3. Par Compilateur:





On remarque :

1.Clang est toujours mieux que Gcc pour **le Reduc.**