Université de Versailles Saint-Quentin-en-Yvelines
Master 2 : Informatique Haute Performance, Simulation

Nom : Lougani
Prénom : Faouzi
Numéro étudiant : 22003152

Rapport : GLC_TD05

Méthodologie de réalisation :

1. Dans un premier temps j'ai implémenté la fonction **dot_prod()** qui va calculer le les produit scalaires mais comme chaque nœud doit réaliser 10 produits scalaires différents notre fonction prend en plus le **rank** et **size** en paramètres qui sont utilisés afin de calculer les plages d'itérations (début et fin) de chaque nœud (ce qui évite d'avoir un nœud qui fait qu'un nœud ne pourra pas faire le même produit scalaire)
2. Afin d'éviter une écriture concurrente j'ai utiliser le concept de barrière vu au cours comme le montre la figure suivante , chaque processus écrit les produits scalaires dans un fichier (faouzi.txt) dans l'ordre.

```
for (int j = 0; j < size; j++)
{
  if(rank == j)
  {
   local_prod = dot_prod(tab, 50,size,rank);
   printf("local_prod=%d\n",local_prod);
   fprintf(f,"%d\n",local_prod);
  }

MPI_Barrier(MPI_COMM_WORLD);
```

3. Pour mesurer les temps de traitements une implémentation de la fonction **get_time_us()** est faite :

```
long get_time_us()  // return the time in the unit of us
{
    struct timeval my_time;  //us
    gettimeofday(&my_time, NULL);
    long runtime_us = my_time.tv_sec+ my_time.tv_usec; // us
    return runtime_us;
}
```

On aura les temps de traitements min et max des 5 nœuds parmi celles passées par réduction.avec : (sachant que le type ici c'est des entiers du coups MPI_INT)

```
// les temps de traitements min et max des noeuds
MPI_Allreduce(&end,&min_run,1,MPI_INT,MPI_MIN,MPI_COMM_WORLD);
MPI_Allreduce(&end,&max_run,1,MPI_INT,MPI_MAX,MPI_COMM_WORLD);
```

4. Pour l initialisation aléatoire du tableau j ai utilisé la fonction **srand()** de la bibliothèque stdlib.h les valeurs sont ajustés afin d'être dans l'intervalle [0,50] afin d avoir des valeurs de produits scalaires petites en cas de validation numérique.

Mesure des temps de traitements :

Machine 1 :

Disposant de l'implémentation MPI version :**mpich-3.4.1 ,** dans un premier temps j'ai exécuté en local mon code les résultats sont les suivants :

nb_itération =10 avec la commande **mpiexec -n 5 ./monprog.exe**

```
Min time: 1641203030  Max time: 1641214102
Min time: 1641203030  Max time: 1641214102
Min time: 1641203030  Max time: 1641214102
Min time: 1641203030  Max time: 1641214102
Min time: 1641203030  Max time: 1641214102
louganifaouzi@louganifaouzi-ThinkPad-L520:~/Bureau/abdeljalil/MPI_FAOUZI/tds/mpich-3.4.1/APP-main$
```

nb_itération =100

```
Min time: 1641297946  Max time: 1641303064
Min time: 1641297946  Max time: 1641303064
Min time: 1641297946  Max time: 1641303064
Min time: 1641297946  Max time: 1641303064
Min time: 1641297946  Max time: 1641303064
louganifaouzi@louganifaouzi-ThinkPad-L520:~/Bureau/abdeljalil/MPI_FAOUZI/tds/mpich-3.4.1/APP-main$
```

nb_itération =1000

```
Min time: 1641482824  Max time: 1641491831
Min time: 1641482824  Max time: 1641491831
Min time: 1641482824  Max time: 1641491831
Min time: 1641482824  Max time: 1641491831
Min time: 1641482824  Max time: 1641491831
louganifaouzi@louganifaouzi-ThinkPad-L520:~/Bureau/abdeljalil/MPI_FAOUZI/tds/mpich-3.4.1/APP-main$
```

Résultats :

- Les temps des traitements MIN et MAX pour une expérience restent inchangé pour les 5 nœuds
- Les temps des traitements MIN et MAX augmentent en augmentant le nombre d'itérations.

Machine 2 :

J'ai choisi d'utiliser la machine knl01 (qu'on a vue au cours 3) disposant des caractéristiques plus puissantes que celle que je dispose en local et comparer les résultats sur les nœuds.cette machine a les caractéristiques suivantes :

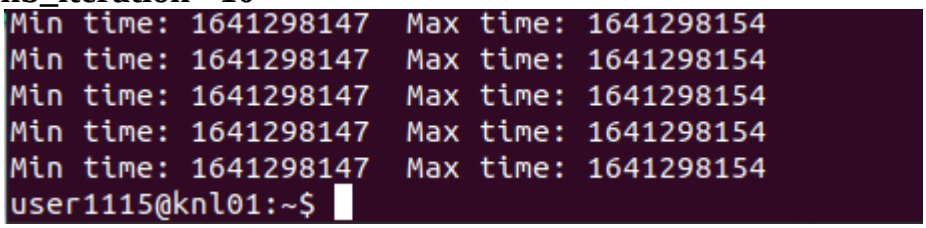
Nom du noeud	effectif nœud	Ram (en Gbyte)	Cœur par nœud	Threads	Type processeur
knl01	4 numa	24 par noeud	64	256	intel knights

Afin que nos expériences soient correctes un chargement du module mpi est necessaire avec les commandes : **module load icc/latest**

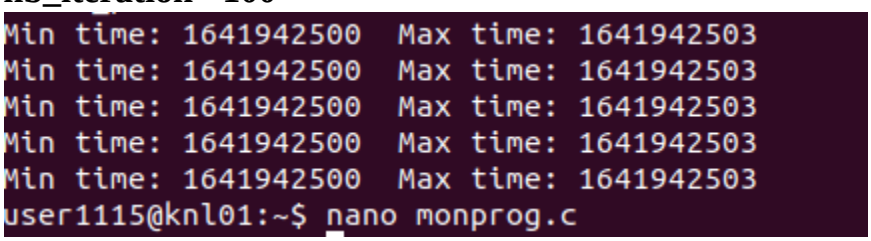
module load compiler-rt/latest

les résultats sont les suivants :

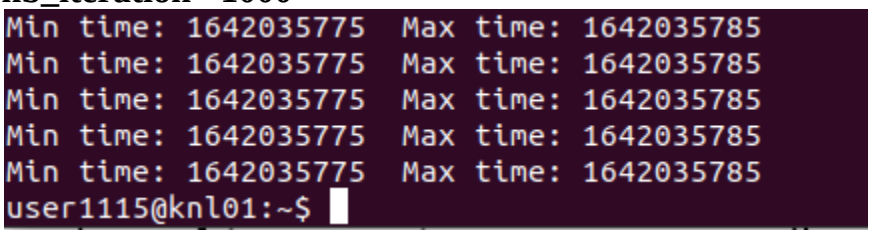
nb_itération =10



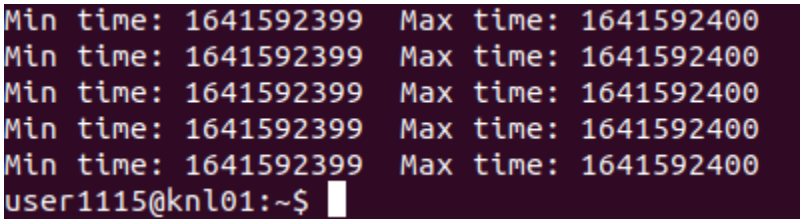
nb_itération =100



nb_itération =1000



nb_iteration =10000



<u>Résultats et conclusion :</u>

- Les temps de traitement (MIN et Max) diminuent fortement avec knl01 comparant a lorsque on exécute notre programme en local.Surtout pour le nombre d itérations important 10000 par exemple, en local pour ce même nombre d ‘itérations mon code prend des dizaines de minutes ce qui justifie l absence des mesures pour ce cas dans expérience avec la machine 1.

- Les temps des traitements MIN et MAX pour une expérience restent inchangé pour les 5 nœuds
- Les temps des traitements MIN et MAX augmentent en augmentant le nombre d’itérations.
- La version de mpi ,ainsi que la nature de la machine (architecture, nombre de coeurs ..) influencent sur les performances d’un même code parallèle.