Université Saint Quentin en Yvelines
Université Paris Saclay


Techniques d'optimisation

M1 Calcul Haute Haute Performance, Simulation

# Projet Optimisation d'une simulation

***Étudiant:***
Anis Mehidi

***Responsables:***
Hugo Taboada
Julien Jaeger

Avril 2022

# Table des matières

# 1 Compilation et Exécution du programme

## 1.1 Erreur lors de la compilation

Le programme est non fonctionnel pour le début car il y a un problème dans le Makefile.

Pour commencer, le code a un problème de lien au moment de la compilation car nous définissons 3 variables dans lbm_phys.c et nous les redéfinissons dans lbm_phys.h. On a alors une définition multiple.

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDI
ANT/simu_simple_LBM$ make
mpicc -Wall -g -c -o main.o main.c
mpicc -Wall -g -c -o lbm_phys.o lbm_phys.c
mpicc -Wall -g -c -o lbm_init.o lbm_init.c
mpicc -Wall -g -c -o lbm_struct.o lbm_struct.c
mpicc -Wall -g -c -o lbm_comm.o lbm_comm.c
mpicc -Wall -g -c -o lbm_config.o lbm_config.c
mpicc -Wall -g -o lbm main.o lbm_phys.o lbm_init.o lbm_struct.o lbm_comm.o lbm_config.o -lm
/usr/bin/ld : lbm_phys.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:9 : définitions multiples de « opposite_of »; main.o:/home/anism/Téléchargements/TOP_PRO
JET_ETUDIANT/simu_simple_LBM/lbm_phys.h:9 : défini pour la première fois ici
/usr/bin/ld : lbm_phys.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:10 : définitions multiples de « equil_weight »; main.o:/home/anism/Téléchargements/TOP_P
ROJET_ETUDIANT/simu_simple_LBM/lbm_phys.h:10 : défini pour la première fois ici
/usr/bin/ld : lbm_phys.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:11 : définitions multiples de « direction_matrix »; main.o:/home/anism/Téléchargements/T
OP_PROJET_ETUDIANT/simu_simple_LBM/lbm_phys.h:11 : défini pour la première fois ici
/usr/bin/ld : lbm_init.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:9 : définitions multiples de « opposite_of »; main.o:/home/anism/Téléchargements/TOP_PRO
JET_ETUDIANT/simu_simple_LBM/lbm_phys.h:9 : défini pour la première fois ici
/usr/bin/ld : lbm_init.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:10 : définitions multiples de « equil_weight »; main.o:/home/anism/Téléchargements/TOP_P
ROJET_ETUDIANT/simu_simple_LBM/lbm_phys.h:10 : défini pour la première fois ici
/usr/bin/ld : lbm_init.o:/home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm_p
hys.h:11 : définitions multiples de « direction_matrix »; main.o:/home/anism/Téléchargements/T
OP_PROJET_ETUDIANT/simu_simple_LBM/lbm_phys.h:11 : défini pour la première fois ici
collect2: error: ld returned 1 exit status
make: *** [Makefile:24 : lbm] Erreur 1
```

et pour remédier à ça, nous allons ajouter marquer que ces variables sont externes dans le fichier d'en-tête pour indiquer qu'elles sont définis dans un autre fichier.

```
1 /********************** CONSTS **********************/
2 extern const int opposite_of[DIRECTIONS];
3 extern const double equil_weight[DIRECTIONS];
4 extern const Vector direction_matrix[DIRECTIONS];
```

## 1.2 Erreur lors de l'exécution le programme

Lors de l'exécution avec la commande **mpirun -np 512** , mon pc ne supporte pas cette exécution car le nombre maximum de processus supporté par mon ordinateur est dépassé. On diminue le nombre de processus à exécuter à 8 et on exécute avec cette commande **mpirun -np 8 –oversubscribe ./lbm**

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 512 ./lbm
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 512
slots that were requested by the application:

  ./lbm

Either request fewer slots for your application, or make more slots
available for use.

A "slot" is the Open MPI term for an allocatable unit where we can
launch a process.  The number of slots available are defined by the
environment in which Open MPI processes are run:

  1. Hostfile, via "slots=N" clauses (N defaults to number of
     processor cores if not provided)
  2. The --host command line parameter, via a ":N" suffix on the
     hostname (N defaults to 1 if not provided)
  3. Resource manager (e.g., SLURM, PBS/Torque, LSF, etc.)
  4. If none of a hostfile, the --host command line parameter, or an
     RM is present, Open MPI defaults to the number of processor cores

In all the above cases, if you want Open MPI to default to the number
of hardware threads instead of the number of processor cores, use the
--use-hwthread-cpus option.

Alternatively, you can use the --oversubscribe option to ignore the
number of available slots when deciding the number of processes to
launch.
--------------------------------------------------------------------------
```

Lors de l'exécution, on a une erreur de segmentation, le signal 11, qui veut dire que le programme a accédé à un emplacement mémoire qui n'a pas été attribué.

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08302] *** Process received signal ***
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08302] Signal: Segmentation fault (11)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08302] Signal code: Address not mapped (1)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08302] Failing at address: (nil)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08303] *** Process received signal ***
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08303] Signal: Segmentation fault (11)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08303] Signal code: Address not mapped (1)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08303] Failing at address: (nil)
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16000
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08305] *** Process received signal ***
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08305] Signal: Segmentation fault (11)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08305] Signal code: Address not mapped (1)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08305] Failing at address: (nil)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08308] *** Process received signal ***
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08308] Signal: Segmentation fault (11)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08308] Signal code: Address not mapped (1)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08308] Failing at address: (nil)
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08311] *** Process received signal ***
[anism-VivoBook-ASUSLaptop-X515EA-X515EA:08311] Signal: Segmentation fault (11)
```

### 1.2.1 Débogage du programme

Afin de déterminer l'origine de cette erreur, et avoir une idée de comment la corrigé, on utilise GDB, et l'exécution avec ce dernier en utilisant le RUN, donne ça :

```
(gdb) run
Starting program: /home/anism/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 19084]
[New Thread 0x7ffff785b700 (LWP 19088)]
[New Thread 0x7ffff6ea8700 (LWP 19089)]
================== CONFIG ==================
iterations          = 16000
width               = 800
height              = 160
obstacle_r          = 17.000000
obstacle_x          = 161.000000
obstacle_y          = 83.000000
reynolds            = 100.000000
reynolds            = 100.000000
inflow_max_velocity = 0.100000
output_filename     = resultat.raw
write_interval      = 50
------------ Derived parameters --------------
kinetic_viscosity   = 0.034000
relax_parameter     = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 162 )

Thread 1 "lbm" received signal SIGSEGV, Segmentation fault.
0x0000555555556bb8 in setup_init_state_global_poiseuille_profile (mesh=0x7fffffffdc10, mesh_type=0x7fffffffdc40, mesh_comm=0x7fffffffdc50) at lbm_init.c:85
85                                  Mesh_get_cell(mesh, i, j)[k] = compute_equilibrium_profile(v,density,k);
(gdb)
```

L'erreur vient de la fonction **Mesh_get_cell()** à la ligne 85 du fichier **lbm_init.c**. Donc la ligne de l'erreur est :

**Mesh_get_cell(mesh, i, j)[k] = compute_equilibrium_profile(v,density,k) ;**

L'erreur effective vient de la structure mesh ,on fait un print afin de voir les adresses :

```
(gdb) print mesh
$1 = (Mesh *) 0x7fffffffdc10
(gdb) print (*mesh)
$2 = {cells = 0x0, width = 802, height = 162}
(gdb)
```

L'une des cellules du maillage n'est pas allouée ,c'est a dire l'adresse 0x0, une fois la fonction **Mesh_get_cell()** accède à cet emplacement, un bug aura lieu. Pour ça, on va faire un " **list Mesh_get_cell** ", et on remarque que la structure n'est pas allouée dans la fonction et on cherche le fichier.c associé pour la définition de cette structure.

```
(gdb) list Mesh_get_cell
110     /*******************  FUNCTION  *********************/
111     /**
112      * Fonction à utiliser pour récupérer une cellule du maillage en fonction de ses coordonnées.
113     **/
114     static inline lbm_mesh_cell_t Mesh_get_cell( const Mesh *mesh, int x, int y)
115     {
116             return &mesh->cells[ (x * mesh->height + y) * DIRECTIONS ];
117     }
118
119     /*******************  FUNCTION  *********************/
```

Lors de l'ouverture du fichier **lbm_struct.c**, on voit que l'erreur est claire dans la fonction **Mesh_init()**.

```
1     //alloc cells memory
2     mesh->cells = NULL;
3     //mesh->cells = malloc( width * height  * DIRECTIONS * sizeof(
    double ) );
```

Et pour corriger cette erreur, il suffit d'enlever le commentaire pour malloc et redéfinir le code de cette manière :

```
1     //alloc cells memory
2     mesh->cells = malloc( width * height  * DIRECTIONS * sizeof( double
      ) );
3     //errors
4     if( mesh->cells == NULL )
5     {
6         perror( "malloc" );
7         abort();
8     }
```

Afin d'exécuter le programme, j'ai choisi 8 processus, et j'ai réduit le nombre d'itération dans le fichier **config.txt** de 16000 à 16.

```
1     iterations            = 16 //Le nombre d'iterations diminue
2     width                 = 800
3     height                = 160
4     #obstacle_r            =
5     #obstacle_x            =
6     #obstacle_y            =
7     reynolds              = 100
8     inflow_max_velocity   = 0.100000
9     inflow_max_velocity   = 0.100000
10    output_filename       = resultat.raw
11    write_interval        = 50
```

Ensuite, je lance le programme avec un nombre réduit d'itérations à 16 et sans MPI, juste avec la commande **./lbm** . Le programme se termine sans aucun problème.

Donc, je décide de l'exécuter avec MPI , mais cette fois le programme ne veut pas terminé son exécution, et je crois qu'il s'agit d'un interblocage et cela se produit lorsque des processus concurrents s'attendent mutuellement. Un processus peut aussi s'attendre lui-même. Les processus bloqués dans cet état le sont définitivement.

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Telechargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
inflow_max_velocity  = 0.100000
output_filename      = resultat.raw
write_interval       = 50
------------ Derived parameters --------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
```

En analysant les codes, on voit que le problème est que sur la fonction principale et exactement dans le fonction **close_file**, on n'autorise que le **RANG_MASTER** pour exécuter la fonction **close_file**. Mais dans cette fonction on a une barrière MPI avec **MPI_COMM_WORLD** qui fait attendre tous les processus. Et le problème est que les autres threads n'ont pas de barrière MPI. Donc, la fonction ne peut pas fermer le fichier fp vu que le **RANK_MASTER** va attendre indéfiniment.

```
1     void close_file(FILE* fp){
2     //wait all before closing
3     MPI_Barrier(MPI_COMM_WORLD); //La barriere qui empeche notre
    programme a ferme le fichier fp
4     //close file
5     fclose(fp);
6 }
```

Pour corriger cela, je supprime la ligne de barrière MPI dans la fonction **close_file** à la ligne 62.

Maintenant notre programme finie son exécution correctement :

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDI
ANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
inflow_max_velocity  = 0.100000
output_filename      = resultat.raw
write_interval       = 50
------------ Derived parameters --------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$
```

### 1.3 valgrind

J'ai utilisé l'outil Valgrind afin de voir s'il y a des fuites mémoire en exécutant avec la commande : **valgrind ./lbm**

```
==15599==
==15599== Process terminating with default action of signal 27 (SIGPROF)
==15599==    at 0x4BF552A: __open_nocancel (open64_nocancel.c:45)
==15599==    by 0x4C0330F: write_gmon (gmon.c:370)
==15599==    by 0x4C03B6E: _mcleanup (gmon.c:444)
==15599==    by 0x4B2900D: __cxa_finalize (cxa_finalize.c:83)
==15599==    by 0x1094E6: ??? (in /home/anism/Bureau/TOP_PROJET_ETUDIANT/simu_simple_LBM/lbm)
==15599==    by 0x4011F5A: _dl_fini (dl-fini.c:138)
==15599==    by 0x4B288D6: __run_exit_handlers (exit.c:108)
==15599==    by 0x4B28A8F: exit (exit.c:139)
==15599==    by 0x4B060B9: (below main) (libc-start.c:342)
==15599==
==15599== HEAP SUMMARY:
==15599==     in use at exit: 117,067 bytes in 86 blocks
==15599==   total heap usage: 25,819 allocs, 25,733 frees, 33,940,560 bytes allocated
==15599==
==15599== LEAK SUMMARY:
==15599==    definitely lost: 8,244 bytes in 36 blocks
==15599==    indirectly lost: 599 bytes in 20 blocks
==15599==      possibly lost: 0 bytes in 0 blocks
==15599==    still reachable: 108,224 bytes in 30 blocks
==15599==         suppressed: 0 bytes in 0 blocks
==15599== Rerun with --leak-check=full to see details of leaked memory
==15599==
==15599== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Expiration de la minuterie durant l'établissement du profile
```

Il semble être normal. Nous n'avons pas perdu ou supprimé d'octets. Mais nous avons 8,244 octets définitivement perdus. Je vais donc vérifier le code pour voir où la mémoire n'est pas libérée.

## 2 Scalabilité du programme

Afin de déterminer si notre code possède une bonne Scalabilité ou non on va :

- **Cas 1 :** Augmenter la taille des données et la taille totale du problème.
- **Cas 2 :** Augmenter la taille des données pour une taille de problème fixe.

Ensuite on va faire une comparaison par rapport au temps de calcul écoulé lors du traitement.

### 2.1 Cas 1 : Augmentation de la taille des données et la taille totale du problème

Avant de commencer, on doit avoir le temps de calcul écoulé lors du traitement dans l'état normal .Pour ça, j'ai choisi **MPI_Wtime()** qui renvoie le temps écoulé sur le processeur appelant avec le code suivant :

```
    /* notre code , juste avant la boucle ou s'est commente Time steps
   dans le fichier main.c */
    double start, end;
    start = MPI_Wtime();
    // Notre boucle Time steps
    end = MPI_Wtime();
    double temps_ecoule = end - start;
    if (rank == RANK_MASTER)
    {
      fprintf(stderr, "Temps ecoule en seconde(s): %f\n", temps_ecoule)
   ;
    }

}
```

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
inflow_max_velocity  = 0.100000
output_filename      = resultat.raw
write_interval       = 50
------------ Derived parameters --------------
kinetic_viscosity    = 0.034000
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 16.363735
```

On change les paramètres de note fichier config.txt en 4 fois plus que les données initiales. On aura un résultat comme suit :

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 480 ) (WH 3202 82 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 400 ) (WH 3202 82 )
================== CONFIG ==================
iterations           = 16
width                = 3200
height               = 640
obstacle_r           = 65.000000
obstacle_x           = 641.000000
obstacle_y           = 323.000000
reynolds             = 400.000000
reynolds             = 400.000000
inflow_max_velocity  = 0.400000
output_filename      = resultat.raw
write_interval       = 200
------------ Derived parameters -------------
kinetic_viscosity    = 0.130000
relax_parameter      = 1.123596
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 3202 82 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 3202 82 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 160 ) (WH 3202 82 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 240 ) (WH 3202 82 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 320 ) (WH 3202 82 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 560 ) (WH 3202 82 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 19.301366
```

Le temps écoulé est de 19.301366 secondes, et on remarque que le temps a augmenté par rapport à notre version initiale, en augmentant la taille des données et la taille totale du problème.

### 2.2 Cas 2 : Augmentation de la taille des données pour une taille de problème fixe

On change les paramètres de notre fichier config.txt en 4 fois plus que les données initiales de ces paramètres suivants :

- Vitesse des données entrantes (**Reylonds**) : 400
- Le facteur d'échelle (**inflow_max_velocity**) : 0.400000
- L'intervalle d'écriture entre les sorties (**write_interval**) : 200

On aura un résultat comme suit :

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
================= CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 400.000000
reynolds             = 400.000000
inflow_max_velocity  = 0.400000
output_filename      = resultat.raw
write_interval       = 200
------------ Derived parameters -------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
===========================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 16.384364
```

Le temps écoulé est de 16.384364 secondes, et on remarque que le temps reste presque le même par rapport à notre version initiale, en augmentant la taille des données pour une taille de problème fixe.

D'après les deux résultats de ces deux cas, on remarque que la scalabilité de notre code est mauvaise, et pour y remédier, on doit optimiser notre programme afin que cette dernière soit plus bonne.

# 3 Optimisation du programme

## 3.1 Optimisation de la compilation

On sait toujours que nos programmes sont jamais optimisé car déjà ils manquent des flags d'optimisation et on peut justifier ça en utilisant maqao, donc j'ai décidé de rajoute ces flags d'optimisation : **-Ofast -march=native -funroll-loops -finline-functions**

- **-Ofast** : pour toutes les optimisations de base qu'apporte le compilateur, permet un niveau d'optimisation plus élevé.
- **-march=native** : pour activer la vectorisation en fonction de notre système.
- **-funroll-loops** : Déroule les boucles dont le nombre d'itérations peut être déterminé à la compilation ou à l'entrée dans la boucle.
- **-finline-functions** : pour aligner les fonctions.

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDI
ANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
inflow_max_velocity  = 0.100000
output_filename      = resultat.raw
write_interval       = 50
------------ Derived parameters --------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 15.505750
```

Le temps écoulé est de 15.505750 secondes, et on remarque que le temps a vraiment diminué par rapport à notre version initiale, en ajoutant les flags d'optimisation à la compilation.

Afin de déterminer les parties du code à optimiser, on doit faire un profilage du code afin de connaître les fonctions qui coûtent le plus en terme du temps et cela se fait avec l'outil gprof et en utilisant la commande **gprof lbm>lbm.gprof** et en ajoutant le flag **-pg** dans le Makefile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 25.66      0.10     0.10  2477592     0.00     0.00  compute_equilibrium_profile
 17.96      0.17     0.07   240000     0.00     0.00  compute_cell_collision
 14.11      0.23     0.06  4955184     0.00     0.00  get_vect_norme_2
 12.83      0.28     0.05   240000     0.00     0.00  get_cell_velocity
 10.27      0.32     0.04       15     2.67     3.58  propagation
  7.70      0.35     0.03   240000     0.00     0.00  get_cell_density
  3.85      0.36     0.02  5095680     0.00     0.00  Mesh_get_cell
  2.57      0.37     0.01   317592     0.00     0.00  helper_compute_poiseuille
  2.57      0.38     0.01       15     0.67     1.00  special_cells
  1.28      0.39     0.01   240000     0.00     0.00  lbm_cell_type_t_get_cell
  1.28      0.39     0.01                             main
  0.00      0.39     0.00   431460     0.00     0.00  Mesh_get_cell
  0.00      0.39     0.00   317672     0.00     0.00  lbm_cell_type_t_get_cell
  0.00      0.39     0.00   317592     0.00     0.00  Mesh_get_cell
  0.00      0.39     0.00      150     0.00     0.00  lbm_comm_sync_ghosts_diagonal
  0.00      0.39     0.00      105     0.00     0.00  lbm_comm_sync_ghosts_horizontal
  0.00      0.39     0.00       60     0.00     0.00  lbm_comm_sync_ghosts_vertical
  0.00      0.39     0.00       15     0.00    19.12  collision
```

### 3.2 Parallélisation OpenMP

D'après le fichier généré par gprof, nous pouvons voir que nous avons pas mal de fonctions qui prennent tout le temps de l'application. Par conséquent, si nous voulons optimiser le code, nous devons optimiser ces fonctions, peut-être en ajoutant la parallélisation avec OpenMP mais nous devons être capable de paralléliser ces fonctions en totalité ou presque. Pour une

bonne implémentation hybride de MPI/OpenMP, il faut changer toute la structure. Si nous gardons cette structure, nous ajouterons le pragma omp à chaque fonction mais nous créerons et détruirons des threads à chaque fois pour chaque fonction. Pour réduire le temps d'exécution de ces fonctions on utilisera le système de parallélisation OpenMP.

La majorité de ces fonctions, ne possèdent pas de zone à paralléliser ou y'aura un risque d'interblocage, donc pas de parallélisation nécessaire.

Comme aucune parallélisation ne peut se faire à l'intérieur de ces fonctions, on cherche les zones ou la plupart de ces fonctions sont utilisées, avec la mesure que la parallélisation pourrait être possible. Ou bien modifier les bouts de codes qui ralentissent le programme.

La fonction save_frame() qui fait appel au fonctions get_cell_density(), get_cell_velocity(), get_vect_norme2() et Mesh _get_cell peut être parallélisée.

Juste après l'exécution de notre programme, on voit une nette amélioration de 15.505750 à 15.261540 secondes

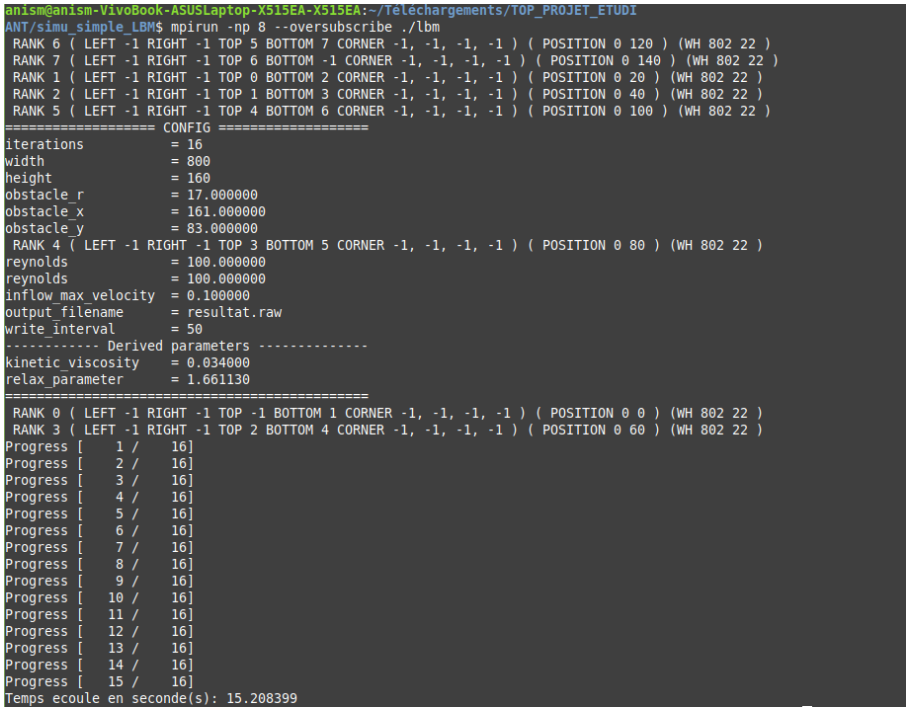
### 3.3 Barrières MPI

Une analyse du code nous informe que des barrières sont implémentées après presque chacune de ces fonctions suivantes :

- save_frame_all_domain()
- special_cells()
- collision()
- propagation()
- lbm_comm_ghost_exchange()
- lbm_comm_sync_ghosts_horizontal
- lbm_comm_sync_ghosts_vertical
- lbm_comm_sync_ghosts_diagonal

Et comme nous sommes sur une mémoire distribué non partagée, donc ces MPI_Barrier sont inutiles, ils sont la juste pour l'impression et nous pouvons les garder pour une sortie lisible, mais ça ralenti notre programme (attendre tous les processus ) Du coup, je propose qu'on supprime toutes les barrières dans la fonction principale main() et dans la fonction lbm_comm_ghost_exchange dans le fichier **lbm_comm.c**.

On exécute le programme ,et on voit que le temps a diminué encore après avoir supprimer les barrières.




Juste après l'exécution de notre programme, on voit une nette amélioration de 15.261540 à 15.208399 secondes.

### 3.4 FLUSH

Lorsque j'ai supprimé **MPI_Barrier** dans **lbm_comm.c**, je constate que nous avons la fonction flush qui fait normalement référence à la sortie. Vu que j'ai jamais vu cette fonction, j'ai lancé dans ma recherche le mot FLUSH pour voir son origine, que j'ai constaté çette déclaration sur le fichier **lbm_config.h** :

```
1 #define concat(a,b,c,d,e) a##b##c##d##e
2 |
3 |
4 |
5 #define __FLUSH_INOUT__ concat(s,l,e,e,p)(1)
6 |
7 |
8 |
9 #define FLUSH_INOUT() __FLUSH_INOUT__
```

Et c'est la que je comprends que la fonction **FLUSH_INOUT()** fait un sleep en appelant **concat(a,b,c,d,e)** et du coup elle va concaténer les lettres du mot sleep, et malgré que sur le commentaire juste avant cette fonction , on dit que cette fonction est très importante, et qu'il ne faut pas l'enlever , mais en la supprimant , le résultat est juste magnifique.

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 100.000000
reynolds             = 100.000000
inflow_max_velocity  = 0.100000
output_filename      = resultat.raw
write_interval       = 50
------------ Derived parameters --------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 0.298422
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$
```

Juste après l'exécution de notre programme, on voit un résultat juste magnifique, une rapidité d'exécution et une nette amélioration de 15.208399 à 0.298422 secondes.

# 4 Contrôle de la scalabilité à la fin de notre optimisation ?

On va refaire la même étude avec les mêmes configurations faite au début lorsqu'on avait vérifié la scalabilité pour voir ce que ça donne au niveau scalabilité.

**Cas 1 : Augmentation de la taille des données et la taille totale du problème**

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 160 ) (WH 3202 82 )
================== CONFIG ==================
iterations           = 16
width                = 3200
height               = 640
obstacle_r           = 65.000000
obstacle_x           = 641.000000
obstacle_y           = 323.000000
reynolds             = 400.000000
reynolds             = 400.000000
inflow_max_velocity  = 0.400000
output_filename      = resultat.raw
write_interval       = 200
------------ Derived parameters --------------
kinetic_viscosity    = 0.130000
relax_parameter      = 1.123596
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 3202 82 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 3202 82 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 320 ) (WH 3202 82 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 400 ) (WH 3202 82 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 480 ) (WH 3202 82 )
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 560 ) (WH 3202 82 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 240 ) (WH 3202 82 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 1.911210
```

- Le temps écoulé est de 19.301366 secondes avant optimisation.
- Le temps écoulé est de 1.911210 secondes après optimisation.

**Cas 2 : Augmentation de la taille des données pour une taille de problème fixe**

```
anism@anism-VivoBook-ASUSLaptop-X515EA-X515EA:~/Téléchargements/TOP_PROJET_ETUDIANT/simu_simple_LBM$ mpirun -np 8 --oversubscribe ./lbm
 RANK 7 ( LEFT -1 RIGHT -1 TOP 6 BOTTOM -1 CORNER -1, -1, -1, -1 ) ( POSITION 0 140 ) (WH 802 22 )
 RANK 2 ( LEFT -1 RIGHT -1 TOP 1 BOTTOM 3 CORNER -1, -1, -1, -1 ) ( POSITION 0 40 ) (WH 802 22 )
 RANK 4 ( LEFT -1 RIGHT -1 TOP 3 BOTTOM 5 CORNER -1, -1, -1, -1 ) ( POSITION 0 80 ) (WH 802 22 )
 RANK 6 ( LEFT -1 RIGHT -1 TOP 5 BOTTOM 7 CORNER -1, -1, -1, -1 ) ( POSITION 0 120 ) (WH 802 22 )
================== CONFIG ==================
iterations           = 16
width                = 800
height               = 160
obstacle_r           = 17.000000
obstacle_x           = 161.000000
obstacle_y           = 83.000000
reynolds             = 400.000000
reynolds             = 400.000000
inflow_max_velocity  = 0.400000
output_filename      = resultat.raw
write_interval       = 200
------------ Derived parameters -------------
kinetic_viscosity    = 0.034000
relax_parameter      = 1.661130
============================================
 RANK 0 ( LEFT -1 RIGHT -1 TOP -1 BOTTOM 1 CORNER -1, -1, -1, -1 ) ( POSITION 0 0 ) (WH 802 22 )
 RANK 1 ( LEFT -1 RIGHT -1 TOP 0 BOTTOM 2 CORNER -1, -1, -1, -1 ) ( POSITION 0 20 ) (WH 802 22 )
 RANK 3 ( LEFT -1 RIGHT -1 TOP 2 BOTTOM 4 CORNER -1, -1, -1, -1 ) ( POSITION 0 60 ) (WH 802 22 )
 RANK 5 ( LEFT -1 RIGHT -1 TOP 4 BOTTOM 6 CORNER -1, -1, -1, -1 ) ( POSITION 0 100 ) (WH 802 22 )
Progress [    1 /    16]
Progress [    2 /    16]
Progress [    3 /    16]
Progress [    4 /    16]
Progress [    5 /    16]
Progress [    6 /    16]
Progress [    7 /    16]
Progress [    8 /    16]
Progress [    9 /    16]
Progress [   10 /    16]
Progress [   11 /    16]
Progress [   12 /    16]
Progress [   13 /    16]
Progress [   14 /    16]
Progress [   15 /    16]
Temps ecoule en seconde(s): 0.321388
```

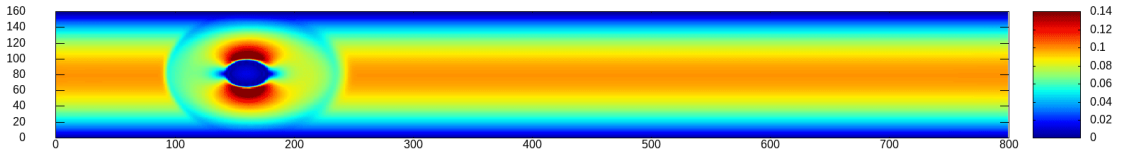
- Le temps écoulé est de 16.384364 secondes avant optimisation.
- Le temps écoulé est de 0.321388 secondes après optimisation.

Les résultats sont beaucoup mieux après l'optimisation du programme, du coup on déduit que cette optimisation appliqué, à évolué la scalabilité de notre programme, malgré je crois qu'on a pas atteint la scalabilité parfaite.

# 5 Validation des résultats

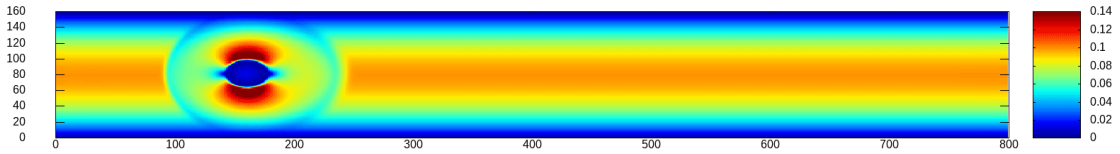
## 5.1 Résultat initiale

En utilisant la commande **./gen_animate_gif.sh ./ref_resultat_200.raw ./out.gif** et voici la dernière image générer par le gif pour notre résultat initale.




## 5.2 Résultat finale après optimisation

En utilisant la commande **./gen_animate_gif.sh ./resultat.raw ./result.gif** et voici la dernière image générer par le gif pour notre résultat finale après optimisation.




On voit clairement qu'on est sur presque les mêmes résultats après l'optimisation de notre programme donc à mon avis on peut valider nos résultats.

# 6 Conclusion

Afin d'avoir une scalabilité parfaite, il faut respecter certaines règles lors de l'optimisation du programme :

- Exécuter le code une première fois pour voir le résultat donnée.
- Déboguer le programme avec différents outils (gdb, valgring).
- Correction des erreurs dans le code.
- Utilisation de l'outil de profilage du code afin de savoir les fonctions qu'on devrait optimiser et qui consomme le plus dans notre programme.
- Étude de la scalabilité du code initial et finale.
- Ajout des flags d'optimisation nécessaires pour la compilation.
- Parallélisation OpenMP pour les boucles .
- Parallélisation MPI et éviter l'usage excessif de barrières inutiles.
- Lecture du code pour savoir si les appels sont bons (par exemple FLUSH_INOUT).
- Mesures après chaque modification du code pour voir si nous avons le même comportement que le que le code original.

# Annexe

`https://github.com/anisus07/TOP_PROJECT_MEHIDI_ANIS.git`