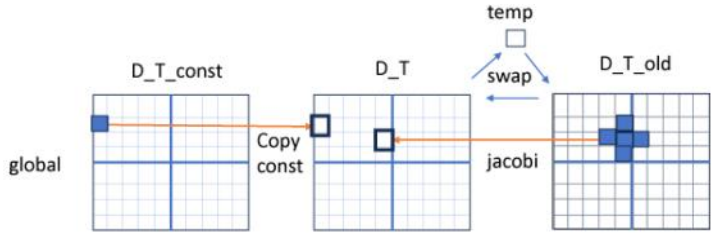
Lab 7b - Heat su GPU

https://elly2023.smfi.unipr.it/mod/page/view.php?id=5654

OBIETTIVO
Sviluppare ed eseguire la parallelizzazione tramite GPU di Heat con CUDA.

Nota:
Utilizziamo 2 kernel: il primo copia i dati costanti, il secondo calcola l'iterazione Jacobi.



Attività svolte

Dopo aver creato la directory di lavoro "mkdir ~/HPC2324/cuda/heat/", ho copiato al suo interno i file da "cp /hpc/home/roberto.alfieri/SHARE/cuda/heat/* ."

Ho creato lo script heat_gpu_scaling.slurm per la compilazione ed esecuzione del programma per diverse dimensioni della matrice, salvando tutti i dati sul file heat_gpu_scaling.csv

heat_gpu_scaling.slurm

```
#!/bin/sh

#SBATCH --output=%x.o%j
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --gres=gpu:p100:1
#SBATCH --mem=4G
#SBATCH --time=0-00:10:00


#stampa il nome del nodo assegnato e argomenti
echo "#SLURM_JOB_NODELIST   : $SLURM_JOB_NODELIST"
echo "#CUDA_VISIBLE_DEVICES : $CUDA_VISIBLE_DEVICES"

module load cuda

rm heat_gpu_scaling.csv

nvcc -O2  heat_gpu.cu -o heat_gpu

for N in 512 1024 2048 4096 8192
do
    heat_gpu -x 32 -y 32  -c $N -r $N  -s 1000 2>> heat_gpu_scaling.csv
Done
```

heat_gpu_scaling.csv

```
[martina.genovese@ui01 heat]$ more heat_gpu_scaling.csv
# NX=256, NY=256,  block_size_x=32, block_size_y=32, grid_size_x=8, grid_size_y=8, MAX_ITER=1000
GPU, 256, 256, 32, 32, 8, 8, 1000,  0.023719
# NX=256, NY=256,  block_size_x=64, block_size_y=64, grid_size_x=4, grid_size_y=4, MAX_ITER=1000
GPU, 256, 256, 64, 64, 4, 4, 1000,  0.001979
# NX=256, NY=256,  block_size_x=128, block_size_y=128, grid_size_x=2, grid_size_y=2, MAX_ITER=1000
GPU, 256, 256, 128, 128, 2, 2, 1000,  0.001987
# NX=256, NY=256,  block_size_x=256, block_size_y=256, grid_size_x=1, grid_size_y=1, MAX_ITER=1000
GPU, 256, 256, 256, 256, 1, 1, 1000,  0.001975
# NX=256, NY=256,  block_size_x=512, block_size_y=512, grid_size_x=1, grid_size_y=1, MAX_ITER=1000
```

```
GPU, 256, 256, 512, 512, 1, 1, 1000,  0.001972
# NX=512, NY=512,  block_size_x=32, block_size_y=32, grid_size_x=16, grid_size_y=16, MAX_ITER=1000
GPU, 512, 512, 32, 32, 16, 16, 1000,  0.025582
# NX=512, NY=512,  block_size_x=64, block_size_y=64, grid_size_x=8, grid_size_y=8, MAX_ITER=1000
GPU, 512, 512, 64, 64, 8, 8, 1000,  0.002037
# NX=512, NY=512,  block_size_x=128, block_size_y=128, grid_size_x=4, grid_size_y=4, MAX_ITER=1000
GPU, 512, 512, 128, 128, 4, 4, 1000,  0.001954
# NX=512, NY=512,  block_size_x=256, block_size_y=256, grid_size_x=2, grid_size_y=2, MAX_ITER=1000
GPU, 512, 512, 256, 256, 2, 2, 1000,  0.001945
# NX=512, NY=512,  block_size_x=512, block_size_y=512, grid_size_x=1, grid_size_y=1, MAX_ITER=1000
GPU, 512, 512, 512, 512, 1, 1, 1000,  0.002001
# NX=1024, NY=1024,  block_size_x=32, block_size_y=32, grid_size_x=32, grid_size_y=32, MAX_ITER=1000
GPU, 1024, 1024, 32, 32, 32, 32, 1000,  0.065576
# NX=1024, NY=1024,  block_size_x=64, block_size_y=64, grid_size_x=16, grid_size_y=16, MAX_ITER=1000
GPU, 1024, 1024, 64, 64, 16, 16, 1000,  0.001541
# NX=1024, NY=1024,  block_size_x=128, block_size_y=128, grid_size_x=8, grid_size_y=8, MAX_ITER=1000
GPU, 1024, 1024, 128, 128, 8, 8, 1000,  0.001605
# NX=1024, NY=1024,  block_size_x=256, block_size_y=256, grid_size_x=4, grid_size_y=4, MAX_ITER=1000
GPU, 1024, 1024, 256, 256, 4, 4, 1000,  0.002016
# NX=1024, NY=1024,  block_size_x=512, block_size_y=512, grid_size_x=2, grid_size_y=2, MAX_ITER=1000
GPU, 1024, 1024, 512, 512, 2, 2, 1000,  0.001593
# NX=2048, NY=2048,  block_size_x=32, block_size_y=32, grid_size_x=64, grid_size_y=64, MAX_ITER=1000
GPU, 2048, 2048, 32, 32, 64, 64, 1000,  0.198530
# NX=2048, NY=2048,  block_size_x=64, block_size_y=64, grid_size_x=32, grid_size_y=32, MAX_ITER=1000
GPU, 2048, 2048, 64, 64, 32, 32, 1000,  0.002165
# NX=2048, NY=2048,  block_size_x=128, block_size_y=128, grid_size_x=16, grid_size_y=16, MAX_ITER=1000

GPU, 2048, 2048, 128, 128, 16, 16, 1000,  0.001581
# NX=2048, NY=2048,  block_size_x=256, block_size_y=256, grid_size_x=8, grid_size_y=8, MAX_ITER=1000
GPU, 2048, 2048, 256, 256, 8, 8, 1000,  0.001730
# NX=2048, NY=2048,  block_size_x=512, block_size_y=512, grid_size_x=4, grid_size_y=4, MAX_ITER=1000
GPU, 2048, 2048, 512, 512, 4, 4, 1000,  0.001550
# NX=4096, NY=4096,  block_size_x=32, block_size_y=32, grid_size_x=128, grid_size_y=128, MAX_ITER=1000

GPU, 4096, 4096, 32, 32, 128, 128, 1000,  0.730810
# NX=4096, NY=4096,  block_size_x=64, block_size_y=64, grid_size_x=64, grid_size_y=64, MAX_ITER=1000
GPU, 4096, 4096, 64, 64, 64, 64, 1000,  0.002071
# NX=4096, NY=4096,  block_size_x=128, block_size_y=128, grid_size_x=32, grid_size_y=32, MAX_ITER=1000

GPU, 4096, 4096, 128, 128, 32, 32, 1000,  0.002134
# NX=4096, NY=4096,  block_size_x=256, block_size_y=256, grid_size_x=16, grid_size_y=16, MAX_ITER=1000

GPU, 4096, 4096, 256, 256, 16, 16, 1000,  0.002236
# NX=4096, NY=4096,  block_size_x=512, block_size_y=512, grid_size_x=8, grid_size_y=8, MAX_ITER=1000
GPU, 4096, 4096, 512, 512, 8, 8, 1000,  0.002101
# NX=8192, NY=8192,  block_size_x=32, block_size_y=32, grid_size_x=256, grid_size_y=256, MAX_ITER=1000

GPU, 8192, 8192, 32, 32, 256, 256, 1000,  2.810191
# NX=8192, NY=8192,  block_size_x=64, block_size_y=64, grid_size_x=128, grid_size_y=128, MAX_ITER=1000

GPU, 8192, 8192, 64, 64, 128, 128, 1000,  0.004143
# NX=8192, NY=8192,  block_size_x=128, block_size_y=128, grid_size_x=64, grid_size_y=64, MAX_ITER=1000

GPU, 8192, 8192, 128, 128, 64, 64, 1000,  0.004127
# NX=8192, NY=8192,  block_size_x=256, block_size_y=256, grid_size_x=32, grid_size_y=32, MAX_ITER=1000

GPU, 8192, 8192, 256, 256, 32, 32, 1000,  0.004224
# NX=8192, NY=8192,  block_size_x=512, block_size_y=512, grid_size_x=16, grid_size_y=16, MAX_ITER=1000

GPU, 8192, 8192, 512, 512, 16, 16, 1000,  0.004283
```

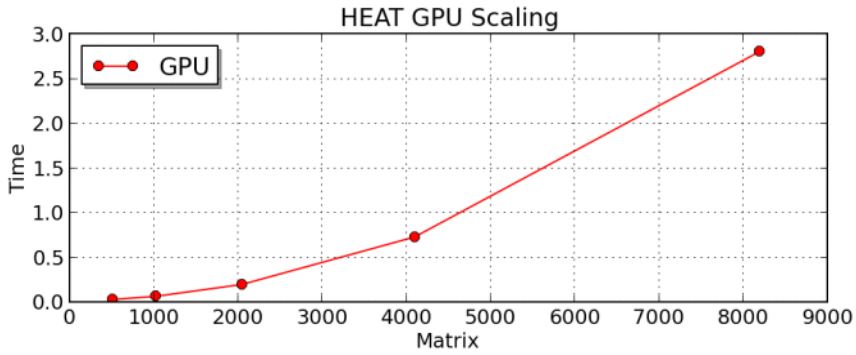
Infine ho creato il programma in python heat_gpu_scaling.py per fare un plot con i dati generati precedentemente.

heat_gpu_scaling.py

```
#!/usr/bin/env python2


import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import pandas as pd


df = pd.read_csv("heat_gpu_scaling.csv", comment='#',
names=["NX","NY","block_size_x","block_sixe_y","grid_size_x","grid_size_y","iter","time"])


print(df)
```

```
plt.subplot(2,1,1)

plt.title('HEAT GPU Scaling')
plt.grid()
plt.xlabel('Matrix')
plt.ylabel('Time')
#plt.yscale('log')
plt.plot(df.NX, df.time, 'r-o', label='GPU', color='red')
plt.legend(shadow=True,loc="best")
plt.savefig('heat_gpu_scaling.png', bbox_inches='tight', dpi=150)
plt.close()
```

Plot



Dopodichè ho copiato le altre versioni del programma heat parallelizzato all'interno della cartella e creato lo script heat_gpu_scaling_confronto.slurm per confrontare tutte e quattro le versioni.

heat_gpu_scaling_confronto.slurm

```
#!/bin/sh

#SBATCH --output=%x.o%j
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --gres=gpu:p100:1
#SBATCH --mem=4G
#SBATCH --time=0-01:00:00

#stampa il nome del nodo assegnato e argomenti
echo "#SLURM_JOB_NODELIST   : $SLURM_JOB_NODELIST"
echo "#CUDA_VISIBLE_DEVICES : $CUDA_VISIBLE_DEVICES"

module load cuda
module load gnu openmpi
module load intel

rm heat_gpu_scaling.csv
rm heat_mpi_scaling.csv
rm heat_omp_scaling.csv
rm heat_mpi_omp_scaling.csv



iter=1000

nvcc -O2 heat_gpu.cu -o heat_gpu
mpicc -O2 mpi_heat.c -o mpi_heat -fopenmp
gcc -O2 omp_heat.c -o omp_heat -fopenmp
mpicc -O2 mpi+omp_heat.c -o mpi+omp_heat -fopenmp

for N in 512 1024 2048 4096 8192
do
    heat_gpu -x 32 -y 32 -c $N -r $N -s $iter 2>> heat_gpu_scaling.csv
     mpirun mpi_heat -r $N -c $N -s $iter 2>> heat_mpi_scaling.csv
     omp_heat -r $N -c $N -s $iter 2>> heat_omp_scaling.csv
     mpirun mpi+omp_heat -r $N -c $N -s $iter 2>> heat_mpi_omp_scaling.csv
done
```

Infine ho creato il programma in python heat_gpu_scaling_confronto.py per confrontare le diverse versioni di Heat al crescere della dimensione del problema.

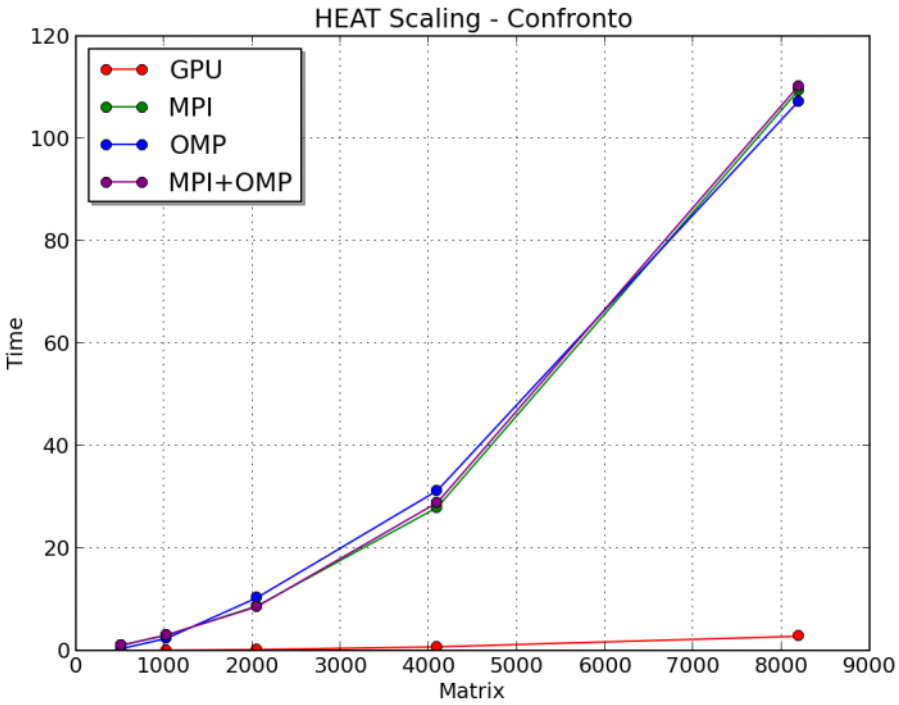
heat_gpu_scaling_confronto.py

```
#!/usr/bin/env python2

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import pandas as pd

df_gpu = pd.read_csv("heat_gpu_scaling.csv", comment='#',
names=["NX","NY","block_size_x","block_sixe_y","grid_size_x","grid_size_y","iter","time"])
df_mpi = pd.read_csv("heat_mpi_scaling.csv" , comment="#", names=["nt", "r", "c", "iter", "time"])
df_omp = pd.read_csv("heat_omp_scaling.csv", comment="#", names=["nt", "r", "c", "iter", "time"])
df_mpi_omp = pd.read_csv("heat_mpi_omp_scaling.csv", comment="#",  names=["r","c","iter", "time"])



print (df_gpu)
print (df_mpi)
print (df_omp)
print (df_mpi_omp)

plt.title('HEAT Scaling - Confronto')
plt.grid()
plt.xlabel('Matrix')
plt.ylabel('Time')
plt.plot(df_gpu.NX, df_gpu.time, 'r-o', label='GPU', color='red')
plt.plot(df_mpi.r, df_mpi.time, 'r-o', label='MPI', color='green')
plt.plot(df_omp.r, df_omp.time, 'r-o', label='OMP', color='blue')
plt.plot(df_mpi_omp.r, df_mpi_omp.time, 'r-o', label='MPI+OMP', color='purple')
plt.legend(shadow=True,loc="best")

plt.savefig('heat_gpu_scaling_confronto.png')
plt.close()
```

Heat_gpu_scaling.csv

```
[martina.genovese@ui01 heat]$ more heat_gpu_scaling.csv
# NX=512, NY=512,  block_size_x=32, block_size_y=32, grid_size_x=16, grid_size_y=16, MAX_ITER=1000
GPU, 512, 512, 32, 32, 16, 16, 1000,  0.029564
# NX=1024, NY=1024,  block_size_x=32, block_size_y=32, grid_size_x=32, grid_size_y=32, MAX_ITER=1000
GPU, 1024, 1024, 32, 32, 32, 32, 1000,  0.064640
# NX=2048, NY=2048,  block_size_x=32, block_size_y=32, grid_size_x=64, grid_size_y=64, MAX_ITER=1000
GPU, 2048, 2048, 32, 32, 64, 64, 1000,  0.198096
# NX=4096, NY=4096,  block_size_x=32, block_size_y=32, grid_size_x=128, grid_size_y=128, MAX_ITER=1000

GPU, 4096, 4096, 32, 32, 128, 128, 1000,  0.729599
# NX=8192, NY=8192,  block_size_x=32, block_size_y=32, grid_size_x=256, grid_size_y=256, MAX_ITER=1000

GPU, 8192, 8192, 32, 32, 256, 256, 1000,  2.805219
```

Plot


HEAT Scaling - Confronto
GPU
MPI
OMP
MPI+OMP
Time
Matrix
0
20
40
60
80
100
120
0
1000
2000
3000
4000
5000
6000
7000
8000
9000