

Tema 3 - GPU Accessible Population



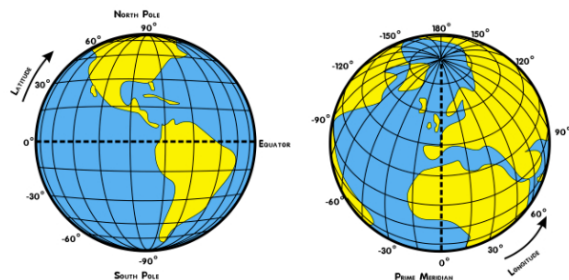
- **Soft deadline:** 24 Mai 26 Mai 28 Mai 2022.
- **Hard deadline:** 31 Mai 2022. Veți primi o depunere de 10% din punctajul maxim al temei pentru fiecare zi de întârziere (după 24 Mai 28 Mai, maxim 3 zile de întârziere).
- **Responsabil:** [Grigore Lupescu](#), [Stefan-Dan Ciocirlan](#), [Costin Carabas](#)



- **Data publicare:** 10 Mai 2022
- **Data actualizare:** 27 Mai 2022

Intro

Ion Popescu, un antreprenor roman, observa ca lantul sau de tonete mobile are succes si se decide sa-si extinda afacerea in alte orase din lume. El isi da seama ca are nevoie de o metoda prin care sa calculeze in timp util zonele cu populatia cea mai mare, prin care sa selecteze ulterior modul in care isi redistribuie tonetele pe fiecare zona in parte. Se consulta cu alte persoane din zona IT si isi da seama ca are nevoie de un algoritim special care sa foloseasca tot hardware-ul disponibil din workstation-ul sau in special GPU-ul.



Enunț

Realizati o implementare a algoritmului "Accessible Population". Este obligatoriu ca implementarea sa fie realizata pe GPU folosind CUDA si orice implementare care nu respecta aceasta cerinta duce automat la pierderea intregului punctaj pe tema.



Tema se poate dezvolta pe orice sistem cu suport CUDA, dar punctarea se face exclusiv pe cozile asc-test si hpsl.

Fisierele din arhiva de tema:

- .devcontainer/devcontainer.json
- checker - rulați scripturile din acest director
 - utils - scripturi ajutatoare
 - run_fep_checker.sh - script care poate fi rulat local si va incarca tema pe fep si o va rula pe cluster
- run_cluster_checker.sh - script care poate fi rulat pe fep si va rula tema pe cluster (necesar sa fie incarcat tot directorul pe fep intr-un director astema3)
- run_local_checker.sh - script care ruleaza tema local
- sol
 - Makefile - binarul rezultat in urma comenzii "make" sa fie numit **gpu_sol**.
 - fisiere de cod
 - README.md
- tests - testele pe care se face evaluarea
 - X.in fisiere de intrare
 - X.ref fisiere de referinta pentru rezultate

Binarul rezultat **./gpu_sol** va accepta urmatoarele argumente:

```
./gpu_sol <kmrange1> <file1_in> <file1_out> <kmrange2> <file2_in> <file2_out>...
```



unde:

- ARGV[3X] kmrange ca intreg in km - distanta maxima la care se va considera populatia unui oras
- ARGV[3X+1] nume fisier intrare orase/locatie/populatie.
- ARGV[3X+2] nume fisier pentru iesire rezultat populatie accesibila.



Pentru a folosi utilitarele din directorul checker, trebuie să aveți activată autentificare pe fep prin chei.

Example rulare:

```
./gpu_sol 127 ../tests/B0.in ../tests/B0.out  
./gpu_sol 345 ../tests/B0.in ../tests/B0.out 456 ../tests/E1.in ../tests/E1.out
```

Exemplu format fisier intrare:

```
< ID oras> <lat> <lon> <pop>
```

Navigare

- Regulament
- Echipă
- Orar
- Catalog

Laboratoare

- Laboratorul 01 - Introducere in limbajul Python
- Laboratorul 02 - Fire de executie in Python
- Laboratorul 03 - Programare concurentă in Python (continuare)
- Laboratorul 04 - Arhitecturi de Microprocesoare si Sisteme de Calcul
- Laboratorul 05 - Tehnici de Optimizare de Cod - Inmultirea Matricelor
- Laboratorul 06 - Analiza Performantelor Programelor
- Laboratorul 07 - Arhitecturi de tip GPGPU
- Laboratorul 08 - Arhitectura GPU NVIDIA CUDA
- Laboratorul 09 - Advanced CUDA
- Exerciții din alți ani

Teme

- Tema 1
- Tema 2
- Tema 3

Resurse

- Folosire X11 în WSL2
- Ghid folosire cluster

GPU related

- CUDA C Programming
- CUDA NVCC compiler
- Visual Profiler
- CUDA 9.1 Toolkit
- NVIDIA Tesla K40M
- NVIDIA Tesla C2070
- Nvidia Tesla 2050/2070
- Nvidia CUDA Fermi/Tesla

Lecture related

- Computatoare
- Taxonomia Flynn
- Single Board Computers
- Explicitly Parallel Instruction Computing
- Intel Parallel Studio

Utilitare

- Dinero cache simulator
- Python Visual Interpreter

Older Labs & Resources

- Cell - Rulare pe CLUSTER
- GDB on Cell BE
- Mailbox Hands-On
- Arhitectura Cell BE
- Kickstart Cell BE
- DMA 101
- Reference Manuals
- Folosirea simulatorului
- Branch Prediction
- Cell Profiler
- Tutorial Cell - Eclipse
- Software Managed Cache
- Liste DMA
- Continut

Table of Contents

- Tema 3 - GPU Accessible Population
 - Intro
 - Enunț
 - Notare
 - Resurse

XEE6994	77.723	79.685	1031
E8509	2.811	3.006	2166
EE4858	77.388	79.455	536

Exemplu format fisier iesire:

```
24853
353830
55650
2438
```

- Pentru fiecare linie citita din fisierul de input trebuie sa afisati la stdout populatia accesibila a orasului reprezentat de linia citita (in aceeaasi ordine).
- Cand distanta intre 2 orase A si B este mai mica sau egala cu $kmrange$ atunci avem $accpopA += popB$, respectiv $accpopB += popA$.
- Daca $kmrange$ este 0, output ar trebui sa fie identic cu populatia fiecarui oras de la input (presupunand ca orasele nu se suprapun).

Notare

Testarea va fi facuta manual folosind testele din arhiva temei. Testele pot suferi modificari dpdv al continutului dar numarul/dimensiunea/complexitatea lor se va pastra.

1. Toate teste au timeout total 15 sec (Tesla P100), pentru Tesla A100 timeout 18 sec, și pentru K40M timeout 30 sec. In conditii mentionate de catre student, se va rula si pe Tesla A100 cu timeout 18 sec. In principiu rulara o sa fie pe Tesla P100 cu timeout 15 sec. 2. Rezolvarea se va face folosind CUDA avand ca device unitatea GPU Tesla P100 (sau Tesla A100) disponibila pe coada asc-test sau K40M pe coada hpsl.

Punctajul maxim este de 100 pct distribuite astfel:

- [**90 pct**] Punctaj dat prin rulara testelor. A nu se optimiza solutia pentru trecerea testelor!
- [**10 pct**] Implementare, README, lipsa probleme conexe.

Arhiva zip va cuprinde obligatoriu fisierele cu solutia și alte fisiere aditionale folosite, dar minim (fisierele din directorul sol):

- Makefile
- Fisier CUDA *.cu
- Readme, unde se explică:
 - Cum s-a implementat solutia ?
 - Output la performantele obtinute și discutie rezultate.



Pentru a folosi utilitarele de mai jos, trebuie să aveți activată autentificare pe fep prin chei.

Exemplu rulare folosind bash script din arhiva teste:

```
[local-checker]$ ./run_fep_checker.sh
INFO: Using cached SIF image
make: Entering directory `../sol'
nvcc -g kernel.cu helper.cpp -o gpu_sol
make: Leaving directory `../sol'

-----
Executing tests with timeout: 15s
-----
B0 Passed .... 10p
E1 Passed .... 20p
M1 Passed .... 20p
M2 Passed .... 20p
H1 Passed .... 20p
-----
Final score: 90/90
```

sau din fep:

```
[fep]$ ~/asctema3/checker/run_cluster_checker.sh
INFO: Using cached SIF image
make: Entering directory `../sol'
nvcc -g kernel.cu helper.cpp -o gpu_sol
make: Leaving directory `../sol'

-----
Executing tests with timeout: 15s
-----
B0 Passed .... 10p
E1 Passed .... 20p
M1 Passed .... 20p
M2 Passed .... 20p
H1 Passed .... 20p
-----
Final score: 90/90
```

sau local:

```
[local-checker]$ ./run_local_checker.sh
make: Entering directory `../sol'
nvcc -g kernel.cu helper.cpp -o gpu_sol
make: Leaving directory `../sol'

-----
Executing tests with timeout: 15s
-----
B0 Passed .... 10p
E1 Passed .... 20p
M1 Passed .... 20p
M2 Passed .... 20p
H1 Passed .... 20p
-----
Final score: 90/90
```



Sesiunile interactive nu sunt permise pe coada asc-test. Va trebui să utilizați scripturile din arhiva teste pentru a folosi această coadă. (sau echivalentul lor)





În cazul în care întâmpinați probleme cu coada sa va adresați pe forum

NOTE


in case of any unexpected problems or errors do not hesitate to return




Resurse

-  Tema 3 GPU
-  Schelet de cod (16 MAI 2022)

Logged in as: Bogdan Mihai TUDORACHE (108609) (bogdan.tudorache99)

asc/tema/tema3.txt · Last modified: 2022/05/27 22:21 by costin.carabas

 Old revisions

 Media Manager  Manage Subscriptions  Back to top

