Tema ASC – OpenCL

Accessible Population

Ultima actualizare 13 Mai 2017

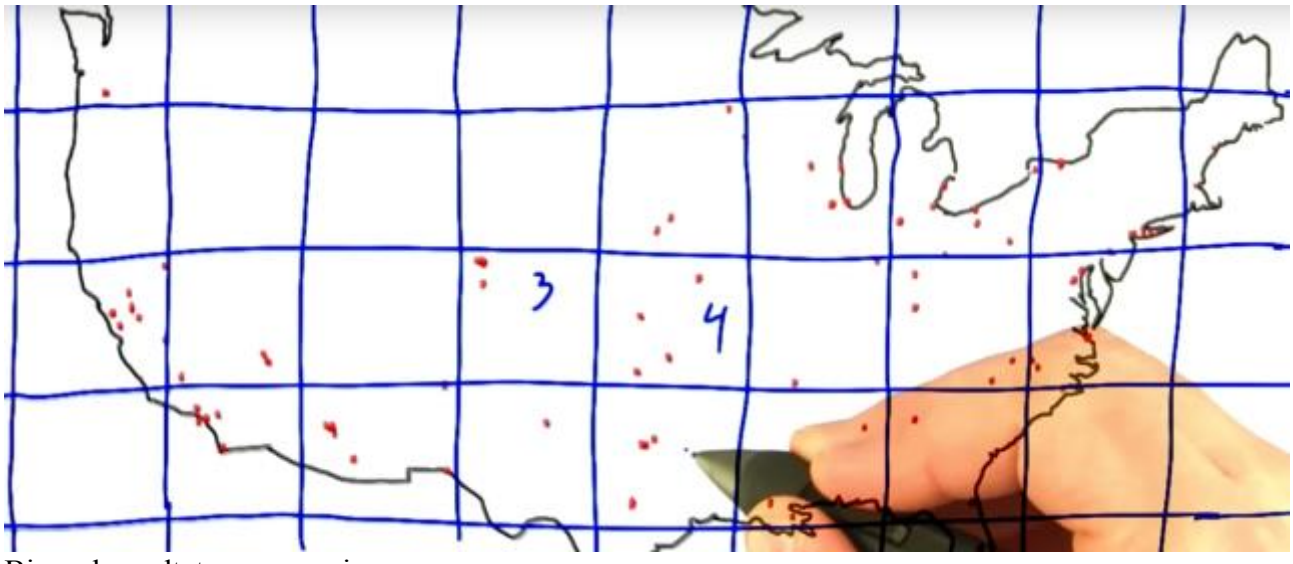
Publicare – **27 aprilie 2017**
Deadline soft - **19 mai 2017**
Deadline hard (-50% punctaj) - **24 mai 2017**

CERINTA:

Realizați o implementare a algoritmului "Accessible Population" prin care alcătuiți o lista al orașelor cu populația accesibila pe o anume distanta. Este **obligatoriu** ca implementarea sa fie realizata folosind OpenCL. Orice implementare care nu respecta aceasta cerință duce automat la pierderea întregului punctaj pe tema.

Orice filtrare a orașelor unde are sens calcularea distantei este permisa si încurajata. Condiția de baza este ca OpenCL sa fie folosit pentru calcularea mai multor distante decât partea non-OpenCL. O varianta brute force optimizata care folosește toate device-urile are șanse mici sa treacă testele hard sau extreme – dar nu este descurajata, cat timp testele trec e OK.

Pentru a evalua performanta monitorizați execuția de kernel si transferurile de memorie. In cazul problemei de fata mai problematic este execuția de kernel – transferurile de memorie cel mai probabil au timp neglijabil. Experimental trebuie găsit un echilibru intre complexitatea unui kernel (variabile folosite = registri, memorie locala folosita, bariere, instrucțiuni control flux precum if/while/for) cat si maparea problemei pe hardware (global/local vs numărul efectiv de thread-uri hardware). Spre exemplu lansarea unui număr mic de thread-uri (global work < 1000) va lasă hardware-ul neutilizat. Definirea unui kernel foarte complex va limita numărul de thread-uri active si iarăși va lasa hardware-ul neutilizat. Pentru Nvidia Tesla se poate folosi utilitarul „nvidia-smi" pentru a vedea încărcarea unității GPU intr-un anume moment.



Binarul rezultat se va numi **accpop**.
Arhiva va cuprinde un **Makefile**. Comenzile "make", "make all" sau "make accpop" trebuie sa rezulte in binarul "./accpop".

Tema va fi trimisa si testata folosind VMChecker

https://vmchecker.cs.pub.ro/ui/#ASC

INPUT

./accpop <kmrange> <file_in> <file_out>

Example:
./accpop 100 easy1.in easy1.out
./accpop 150 hard1.in hard1.out

ARGV[1] - kmrange
Distanta ca întreg – reprezintă distanta *maxima* la care se va considera populația unui oraș.

ARGV[2] – filename_in
Un fișier cu următorul format (fara header, direct date):

<lat:float, lon:float>	<population:int>
-48.034,79.9116	70937
47.9252,111.898	76284

ARGV[3] – filename_out
Un fișier cu următorul format:
70937
76284

OUTPUT

Pentru fiecare linie citita din fișierul de input trebuie sa scrieți in fișierul de output populația accesibila a orașului reprezentat de linia citita (in aceeași ordine). Când distanta intre 2 orașe A si B este *mai mica sau egala* cu kmrange atunci avem accpopA += popB, respectiv accpopB += popA. Avem tot timpul accpopA >= popA, indiferent de kmrange>=0.

Exemplu input:

0.1,0.2	1000
0.1,0.3	1000
45.3,45.5	100

La kmrange = 0, o sa avem ca output doar populația inițiala a orașelor
1000
1000
100
La kmrange = 50, o sa avem diferențe date doar de orașe apropiate:
2000
2000
100
La kmrange = 20000, o sa avem suma tuturor:
2100
2100
2100

Urmăriți formatul fișierelor de input si de referința (output). Orice diferență (output vs reference) va rezulta in invalidarea rezultatului.

TESTARE

Testarea va fi făcută folosind vmchecker: https://vmchecker.cs.pub.ro/ui/#ASC
Suita de teste este descrisa mai jos. Testele sunt disponibile si offline.

No	VMChecker test	Număr orașe	Timeout	Punctaj
1	easy1.in	1444	15 sec	15 pct
2	easy2.in	5776	15 sec	15 pct
3	easy3.in	35344	15 sec	15 pct
4	medium1.in	128164	20 sec	10 pct
5	medium2.in	156025	20 sec	10 pct
6	medium3.in	173889	20 sec	10 pct
7	hard1.in	219961	25 sec	10 pct
8	hard2.in	251001	25 sec	10 pct
9	hard3.in	287296	25 sec	10 pct
	BONUS			
10	extreme1.in	564001	25 sec	10 pct
	Alte considerente	**Criteriu 1**	**Criteriu 2**	**Punctaj**
	Readme	Dimensiune 50 – 500 cuvinte	Claritate, structura, discutie rezultate	5
	Soluție / Coding	Organizare cod	Comentarii	5
			TOTAL	**125 pct**

1. Fiecare test are **timeout la rulare,** orice depășire rezulta in terminarea execuției acelui test.
2. Orice diferență in output vs reference, rezulta in 0 pct pentru testul rulat. Diferența se face de către vmchecker folosind diff.
3. Tema se poate rezolva folosind orice device OpenCL de pe coada ibm-dp.q (CPU sau GPU) cat timp interfața folosita este OpenCL: http://cs.curs.pub.ro/wiki/asc/_media/asc:lab10:dp-wn0x.pdf
4. Este permisa folosirea mai multor device-uri simultan din aceeași platforma sau din platforme diferite (exemplu Nvidia Tesla + Intel Xeon). Pentru a trece toate testele este totuși suficient folosirea unui singur device - de exemplu Tesla M2070.
5. Execuția vmchecker se face pe cozile hp-sl.q sau ibm-dp.q respectiv ibm-dp48.q.
6. Dezvoltarea temei se va face folosind qsub pentru rulare. Vmchecker de asemenea folosește qsub către hp-sl.q (fallback qsub catre ibm-dp48.q) pentru execuție.

Vmchecker rulează folosind qsub un script ce conține următoarele:

```
timeout 15 ./accpop 400 input/easy1.in output/easy1.out | tail -n 50
timeout 15 ./accpop 400 input/easy2.in output/easy2.out | tail -n 50
timeout 15 ./accpop 400 input/easy3.in output/easy3.out | tail -n 50
timeout 20 ./accpop 200 input/medium1.in output/medium1.out | tail -n 50
timeout 20 ./accpop 200 input/medium2.in output/medium2.out | tail -n 50
timeout 20 ./accpop 200 input/medium3.in output/medium3.out | tail -n 50
timeout 25 ./accpop 150 input/hard1.in output/hard1.out | tail -n 50
timeout 25 ./accpop 150 input/hard2.in output/hard2.out | tail -n 50
timeout 25 ./accpop 100 input/hard3.in output/hard3.out | tail -n 50
timeout 25 ./accpop 100 input/extreme1.in output/extreme1.out | tail -n 50
```

OBSERVATII

1. Scheletul de cod reprezintă un punct de plecare ce conține atât rutine pentru input/output cat si funcție de calcul a distantei. Nu este obligatoriu folosirea sa.
2. Următoarea funcție C/C++ va fi folosita pentru calculul distantei intre 2 puncte geografice date de longitudine si latitudine:

```
#define EARTH_RADIUS 6371.0f
#define DEGREE_TO_RADIANS 0.0174533f

float get_distance(float lat1, float lon1, float lat2, float lon2)
{
  float dist_lat = (lat2 - lat1) * DEGREE_TO_RADIANS;
  float dist_lon = (lon2 - lon1) * DEGREE_TO_RADIANS;
  float inter = sin(dist_lat / 2) * sin(dist_lat / 2) +
    cos(lat1 * DEGREE_TO_RADIANS) * cos(lat2 * DEGREE_TO_RADIANS) * sin(dist_lon / 2) *
sin(dist_lon / 2);
  return 2 * EARTH_RADIUS * atan2(sqrt(inter), sqrt(1 - inter));
}
```

3. Pentru calculul distantelor se va folosi **FLOAT** (32 bit). Pentru calculul populației accesibile se va folosi **INT** (32 bit).

Link-uri utile

[1] Descrierea algoritmului de Accessible Population:
https://www.youtube.com/watch?v=wKMzqPufv7E

[2] Acceleratoare OpenCL ibm-dp.q:
http://cs.curs.pub.ro/wiki/asc/_media/asc:lab10:dp-wn0x.pdf

[3] VMChecker
https://elf.cs.pub.ro/vmchecker/ui/#ASC

[4] Documentatie OpenCL:
https://www.khronos.org/registry/OpenCL/sdk/1.1/docs/man/xhtml/
https://www.khronos.org/files/opencl-1-1-quick-reference-card.pdf
https://www.khronos.org/registry/OpenCL/specs/opencl-1.1.pdf

[5] Laboratoare OpenCL:
http://cs.curs.pub.ro/wiki/asc/asc:lab10:index
http://cs.curs.pub.ro/wiki/asc/asc:lab11:index

[6] Latitude/longitude distance calculator
http://www.nhc.noaa.gov/gccalc.shtml