ALGORITMI PARALELI ȘI DISTRIBUIȚI : TEMA 1

Parallel Multiplayer Tron

Responsabili: Cătălin LEORDEANU, Andrei POȘTOACĂ, Dorinel FILIP

Cuprins

Obiectivele Temei

Tema propune implementarea unei simulări a jocului Snake cu mai mulți jucători. Programul va primi un input format dintr-o planșă de joc (o matrice) ce reprezintă starea și poziția inițială a jucătorilor alături de o listă de mutări alese de aceștia și va avea ca output mai multe matrici reprezentând starea planșei de joc la diferite momente de timp (discrete) ale simulării. Simularea se va încheia la încheierea unui număr de pași sau la realizarea primei coleziuni între jucători sau între două segmente ale aceluiași șerpișor.

În ceea ce privește intersectarea cu marginile planșei, aceasta se va considera a fi sub formă de thorus, jucătorul continuându-și deplasarea din direcția opusă, plecând de la punctul cel mai apropiat de latura opusă a marginii intersectate.

Tema se va realiza în limbajul C și va folosi OpenMP pentru soluționarea folosind mai multe fire de execuție a problemei.

În urma realizării acestei teme, studentul va fi capabil să:

- Facă descompunerea unui algoritm pentru exploatarea eficientă a oportunităților de paralelizare;
- Implementeze (folosind OpenMP) un algoritm paralel pentru rezolvarea problemei propuse;
- Ofere o evaluare obiectivă a scalabilității implementării făcute, pe baza unor experimente ce includ măsurarea timpilor de execuție a soluției pentru diferite scenarii / input-uri.

Un obiectiv important al temei este obținerea unui algoritm cu performanțe ce scalează în raport cu numărul de procesoare/fire de execuție folosite și demonstrarea (prin conținutul unui fișier de tip README cu

experimentări și observații) a acestei proprietăți. **O temă care nu folosește tehnologiile indicate sau este trimisă fără README (sau cu README necorespunzător) va fi notată cu 0 puncte.**

Rularea temei

Dupa compilare, tema va primi urmatoarele argumente la rulare: fisierul de intrare, fisierul de iesire, numarul de runde

./tema1 input_file output_file nr_runde

(modificat la 26.10.2017)

Formatul fisierelor de intrare

Fisierele de intrare vor avea urmatorul format text. Pe prima linie numarul n de jucatori, apoi pe urmatoarele n linii fiecare snake cu pozitia varfului Ox si Oy, un numar intreg cu care este codificat in harta si directia de deplasare. Dupa aceasta urmeaza dimensiunea hartii (numarul de linii si numarul de coloane), apoi harta in sine.

Exemplu:

```
3
1 2 1 N
5 7 2 E
3 4 3 S
9 9
0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 0 1 0 3 3 3 3 0
0 1 1 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 2 2 2 0
0 0 0 0 0 2 0 0 0
0 0 0 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0
```

Restricție/Dezambiguizare: Pentru orice punct aparținând unui șarpe se garantează că cel mult alte două puncte vecine vor aparține aceluiași șarpe.

Se consideră puncte vecine acelea ale căror coordonate se pot obține prin varierea (alternativă sau simultană) cu o unitate (prin adunare sau scădere) a oricăruia dintre indicii de coloană sau de linie.

Formatul fisierelor de iesire

Fisierul de iesire va avea acelasi format ca si fisierul de intrare si va contine pozitia jucatorilor dupa cele n runde. De exemplu, dupa o singura runda, pozitiile din fisierul de intrare va fi:

```
3
0 2 1 N
5 8 2 E
3 4 3 S
9 9
```

```
0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 0 1 0 3 3 3 0 0
0 0 1 0 3 0 0 0 0
0 0 0 0 3 0 0 0 0
0 0 0 0 0 2 2 2 2
0 0 0 0 0 2 0 0 0
0 0 0 0 2 2 0 0 0
0 0 0 0 0 0 0 0 0
```

Detalii de implementare

Tema va trebui sa citeasca fisierul de intrare si sa simuleze deplasarea jucatorilor pe harta data. Nu exista obstacole pe harta in afara altor jucatori. Pentru fiecare pas al algoritmului fiecare jucator se va deplasa in directia precizata cu o pozitie, pastrand aceeasi lungime.

Marginile hartii sunt conectate. Astfel, daca un jucator deplasandu-se spre Nord depaseste marginea de sus a hartii, el va aparea pe aceeasi coloana in partea de jos a hartii si va continua sa se deplaseze.

Simularea se incheie daca s-au efectuat cele n runde specificate la rulare, sau daca doi dintre jucatori s-au ciocnit. In acest caz se va afisa in fisierul de iesire harta in urma ciocnirii jucatorilor.

Paralelizarea implementarii se va face in OpenMP. Calculul pozitiei jucatorilor pentru fiecare runda se va face in paralel.

Rularea temei se va face folosind numărul de fire de execuție specificat de variabila de mediu OMP_NUM_THREADS.

Scalabilitate si readme

O parte importanta a acestei teme este obtinere scalabilitatii. Desigur, pentru a obtine scalabilitate trebuie intai scris programului serial, si acesta va avea o parte mare din punctaj. Pentru a demonstra scalabilitatea trebuie sa masurati timpul de executie al programului vostru sub aceleasi configuratii, folosind 2, 4, 6 si 8 thread-uri. Readme-ul trebuie sa contina, descrierea solutiei voastre, metodologia de testare si descrierea sistemelor pe care ati efectuat testele alaturi de rezultatele prin care demonstrati (sau nu) scalabilitatea.

Implementarea, trimiterea și testarea temei

Implementarea temei se va face **obligatoriu** plecând de la scheletul de cod oferit, fără modificarea fișierelor *main.c* sau *Makefile*.

Mai mult decât atât, pentru a păstra relevanța testelor cât și funcționalitatea checker-ului automat, implementarea voastră nu trebuie să facă alte afișări/operații cu fișiere în afara celor deja incluse în scheletul de cod. Nerespectarea acestei restricții poate conduce la nepunctarea temei (indiferent de rezultatul oferit de către checker).

Testarea finlă a funcționalității temei se va face pe *vmchecker* folosind o serie de teste ce vor fi făcute publice cu cel puțin 5 zile înainte de deadline-ul temei. Execuția testelor automate se va face pe coada ibm-nehalem.q din clusterul facultății.

Vă rugăm să rețineți că vmchecker este doar o unealtă care face mai transparentă/rapidă evaluarea finală a temelor. Pentru testarea pe parcurs a temelor puteți folosi în mod direct resursele din clusterul facultății (prin intermediul fep.grid.pub.ro).

În acest context, vă sfătuim să vă testați temele cât mai din timp și menționăm că nu ne asumăm responsabilitatea pentru eventuale cozi de așteptare în rularea (pe vmchecker) a testelor automate ce pot apărea, mai ales în ultimele zile înainte de deadline.

Tentativă de manipulare artificială a timpilor de execuție a programului (prin adăugarea de I/O, sleep-uri etc.), precum și falsificarea datelor din README va fi penalizată drastic (până la maximul permis de regulamentul materiei).