UNIVERSITATEA DE STAT DIN MOLDOVA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALITATEA INFORMATICA

LUCRARE DE LABORATOR

la disciplina „Programarea paralelă și distribuită”

Lucrarea de laborator nr. 5: Un algoritm paralel pentru soluționarea jocurilor bimatriceale în strategii pure, utilizând algoritmul 2D-ciclic

Verificat: Hâncu Boris, dr., conf. univ.
Efectuat: Cemîrtan Cristian, studentul grupei I2101

CHIȘINĂU – 2023

Cuprins

Formularea problemei

Având condițiile din lucrările de laborator nr. 1, 3 și 4, de realizat paralelizarea la nivel de date utilizând algoritmul 2D-ciclic, pentru a determina situațiile de echilibru Nash.

Scopul principal a programului este de a minimiza utilizarea memoriei operative. Această realizare a fost posibilă utilizând colecțiile/structurile de date dinamice STL din C++, de ex. masive cu lungimi dinamice (std::vector), seturi/mulțimi (std::set) și hărți cu perechi cheie-valoare (std::map). De exemplu, acest lucru ne permite cu ușurință să determinăm elementele mulțimii LineGr fără a transpune matricea B (nu facem o „copie” a matricei).

Cum are loc distribuirea submatricelor pentru fiecare proces? Acest lucru se realizează în combinație cu paralelizarea la nivel de date stipulată în punctul 1.b), utilizând operațiile cu fișiere (de ex. MPI_File_open).

Pentru a lucra cu o matrice cu dimensiuni foarte mari, fiecare proces, din topologia 2D, își citește elementele pas cu pas din fișier (grație vederilor de fișier MPI) și nu se necesită copiat submatricea în memoria operativă.

Paralelizarea la nivel de operații este conform punctului 2.b), utilizând funcția ProcessMatrixSerial. Cum am demonstrat în raportul pentru lucrarea de laborator nr. 1, utilizarea funcției MPI_Reduce cu operația MPI_ALLMAXLOC implică „umflarea” submatricelor, și conform scopului stabilit, nu-l implementez în această lucrare de laborator, deoarece nu dorim să creștem utilizarea memoriei operative.

Cum se lansează programul:

```
mpirun -host ... lab4.exe mtx1.txt gM gN bM bN silent time print
```

1. mtx1.txt (sau altfel) – fișierul cu dimensiunile pentru două matrici și conținuturile lor;
2. gM, gN - Dimensiunile grilei de procese. Pentru calcularea automată, se specifică ca 0x0;
3. bM, bN - Dimensiunile blocului/submatricei;
4. silent (opțional) – nu afișează situațiile de echilibru;
5. time (opțional) – afișează timpul de execuție a programului;
6. print (opțional) – afișează submatricele de la fiecare proces.

Algoritmul

1. Procesul root citește fișierul cu matricile A și B, și apoi le înscrie, în format nativ, într-un fișier aparte „array.dat”, pentru a realiza paralelizarea la nivel de date prin fișier;
2. Se creează un comunicator cu topologie carteziană 2D ce reprezintă grila de procese;
3. Se creează un nou tip de date ce reprezintă o secvență de blocuri (distribuite) prevăzute în cadrul algoritmului 2D-ciclic, utilizând funcția **MPI_Type_create_darray** unde se pasează ca parametrii – aria grilei de procese, rankul procesului din comunicatorul nou, nr. de dimensiuni care este 2, dimensiunile întregii matrici, un tablou cu elementul special MPI_DISTRIBUTE_CYCLIC (distribuit ciclic) repetat de două ori, dimensiunile submatricei, dimensiunile grilei de procese, valoarea specială MPI_ORDER_C (reprezentarea pe linii), tipul de date de la care se derivă – MPI_INT și pointerul spre tipul nou de date;
4. Fiecare proces își deschide fișierul „array.dat” utilizând MPI_File_open și își setează vederea, utilizând MPI_File_set_view, unde e-tipul este MPI_BYTE și f-tipul este cel stabilit din punctul 3;
5. Urmează algoritmul 1:
    - Algoritmul se repetă de 2 ori, unde prima iterație este pentru matricea A, și a doua iterație este pentru matricea B.

    a. Fiecare proces își creează o hartă (o mulțime din perechi cheie-valoare) unde cheia este indicele coloanei/liniei și valoarea este un tablou format din cortegiile compuse din trei valori (valoarea, indicele liniei, indicele coloanei). Tabloul va stoca cortegiile pentru valorile maxime de pe coloană/linie;
        - Pentru matricea A, cheia este indicele coloanei, iar pentru matricea B, cheia este indicele liniei. Această caracteristică este necesară pentru a evita transpunerea matricii B.

    b. Prin iterarea submatricelor obținute de la citirea fișierului, fiecare proces își completează harta cu cortegiile corespunzătoare;

    c. Procesele cu rankul diferit de 0 (care nu sunt root), își expediază cortegiile procesului root prin intermediul funcției MPI_Gather, și apoi termină algoritmul;

    d. Procesul root, prin intermediul funcției MPI_Gatherv, obține toate cortegiile culese de celelalte procese;

    e. Pentru fiecare iterație a algoritmului 1, procesul root își determină indicii globali pentru matricea A sau B.

- Procesul root are două mulțimi (ColGr și LineGr) pentru a stoca indicii globali pentru matricile A și B.

6. Prin iterarea tuturor cortegiilor (din algoritmului 1 asupra matricelor A și B) obținute de la toate procesele din grila de procese, procesul root își determină mulțimea NE. Altfel spus, se face intersecția mulțimilor ColGr și LineGr după coordonatele a fiecărui cortegiu.

Exemplificarea algoritmului

De exemplu, avem grila de procese 2x2 cu blocuri 3x2 și următoarele matrici:

$$A = \begin{pmatrix} 400 & 0 & 0 & 0 & 0 & 0 \\ 300 & 300 & 0 & 0 & 0 & 0 \\ 200 & 200 & 200 & 0 & 0 & 0 \\ 100 & 100 & 100 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -100 & -100 & -100 & -100 & -100 & -100 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 200 & 100 & 0 & -100 & -200 \\ 0 & 0 & 100 & 0 & -100 & -200 \\ 0 & 0 & 0 & 0 & -100 & -200 \\ 0 & 0 & 0 & 0 & -100 & -200 \\ 0 & 0 & 0 & 0 & 0 & -200 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Avem următoarea mapare:

P0	P0	P1	P1	P0	P0
P0	P0	P1	P1	P0	P0
P0	P0	P1	P1	P0	P0
P2	P2	P3	P3	P2	P2
P2	P2	P3	P3	P2	P2
P2	P2	P3	P3	P2	P2

Submatricele locale (din matricea A) distribuite pe procese:

400 (0,0)	0 (0,1)	0 (0,4)	0 (0,5)	0 (0,2)	0 (0,3)
300 (1,0)	300 (1,1)	0 (1,4)	0 (1,5)	0 (1,2)	0 (1,3)
200 (2,0)	200 (2,1)	0 (2,4)	0 (2,5)	200 (2,2)	0 (2,3)
100 (3,0)	100 (3,1)	0 (3,4)	0 (3,5)	100 (3,2)	100 (3,3)
0 (4,0)	0 (4,1)	0 (4,4)	0 (4,5)	0 (4,2)	0 (4,3)
-100 (5,0)	-100 (5,1)	-100 (5,4)	-100 (5,5)	-100 (5,2)	-100 (5,3)

Submatricele locale din matricea B distribuite pe procese:

0 (0,0)	200 (0,1)	-100 (0,4)	-200 (0,5)	100 (0,2)	0 (0,3)
0 (1,0)	0 (1,1)	-100 (1,4)	-200 (1,5)	100 (1,2)	0 (1,3)
0 (2,0)	0 (2,1)	-100 (2,4)	-200 (2,5)	0 (2,2)	0 (2,3)
0 (3,0)	0 (3,1)	-100 (3,4)	-200 (3,5)	0 (3,2)	0 (3,3)
0 (4,0)	0 (4,1)	0 (4,4)	-200 (4,5)	0 (4,2)	0 (4,3)
0 (5,0)	0 (5,1)	0 (5,4)	0 (5,5)	0 (5,2)	0 (5,3)

Pentru procesele cu rankul (care au indicii locali):

0. Pe coloane/ColGr0 (matricea A) = {(0,0), (1,1), (0,4), (1,4), (2,4), (0,5), (1,5), (2,5)}.
   Pe linii/LineGr0 (matricea B) = {(0,1), (1,0), (1,1), (2,0), (2,1)}.
1. ColGr1 = {(2,2), (0,3), (1,3), (2,3)}.
   LineGr1 = {(0,2), (1,2), (2,2), (2,3)}.
2. ColGr2 = {(3,0), (3,1), (3,4), (4,4), (3,5), (4,5)}.
   LineGr2 = {(3,0), (3,1), (4,0), (4,1), (4,4), (5,0), (5,1), (5,4), (5,5)}.
3. ColGr3 = {(3,2), (3,3)}.
   LineGr3 = {(3,2), (3,3), (4,2), (4,3), (5,2), (5,3)}.

Determinarea indicilor globali:

Procesul root selectează doar indicii elementelor maxime de pe coloană/linie.

ColGr = {(0,0), (0,4), (0,5), (1,1), (1,4), (1,5), (2,2), (2,4), (2,5), (3,3), (3,4), (3,5), (4,4), (4,5)} = Col0Gr U Col1Gr U Col2Gr U Col3Gr U Col4Gr U Col5Gr.

0. Col0Gr = {(0,0)}.
1. Col1Gr = {(1,1)}.
2. Col2Gr = {(2,2)}.
3. Col3Gr = {(3,3)}.
4. Col4Gr = {(0,4), (1,4), (2,4), (3,4), (4,4)}.
5. Col5Gr = {(0,5), (1,5), (2,5), (3,5), (4,5)}.

LineGr = {(0,1), (1,2), (2,0), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3), (4,0), (4,1), (4,2), (4,3), (4,4), (5,0), (5,1), (5,2), (5,3), (5,4), (5,5)} = Line0Gr U Line1Gr U Line2Gr U Line3Gr U Line4Gr U Line5Gr.

0. Line0Gr = {(0,1)}.
1. Line1Gr = {(1,2)}.
2. Line2Gr = {(2,0), (2,1), (2,2), (2,3)}.
3. Line3Gr = {(3,0), (3,1), (3,2), (3,3)}.
4. Line4Gr = {(4,0), (4,1), (4,2), (4,3), (4,4)}.
5. Line5Gr = {(5,0), (5,1), (5,2), (5,3), (5,4), (5,5)}.

NE = ColGr ∩ LineGr = {(2,2), (3,3), (4,4)}.

Rezultatele rulării programului

Fișierele pentru testare sunt stocate pe cluster.

```
/home/I01/CemirtanCristian/lucrari_de_laborator/lab5$ mpiexec -host compute-1-1:10 lab5.exe
mtx5.txt 0 0 100 100 time
Situatiile Nash de echilibru:
(35, 489), (37, 132), (40, 229), (61, 284), (62, 41), (67, 424), (69, 321), (77, 243), (97,
42), (106, 313), (107, 44), (116, 425), (120, 429), (127, 297), (128, 388), (133, 362),
(147, 217), (153, 381), (155, 166), (156, 361), (161, 169), (163, 64), (165, 110), (200,
318), (225, 119), (232, 322), (246, 160), (247, 97), (261, 206), (263, 326), (266, 297),
(276, 364), (293, 365), (301, 245), (317, 383), (318, 185), (321, 307), (340, 81), (352,
141), (354, 407), (358, 210), (362, 10), (362, 103), (397, 154), (407, 390), (416, 27),
(418, 319), (423, 95), (428, 487), (434, 378), (440, 190), (444, 31), (457, 260), (493,
211), (498, 28),
Total: 55
Crearea fisierului: 0.11058
Media: 0.278377, Max: 0.298423, Min: 0.258419

/home/I01/CemirtanCristian/lucrari_de_laborator/lab5$ mpiexec -host compute-1-1:4 lab5.exe
mtx1.txt 2 2 3 2 print
Rankul 0@(0,0):
400 0 0 0
300 300 0 0
200 200 0 0
Coloana 0: 400@(0,0),
Coloana 1: 300@(1,1),
Coloana 4: 0@(0,4), 0@(1,4), 0@(2,4),
Coloana 5: 0@(0,5), 0@(1,5), 0@(2,5),

Rankul 1@(0,1):
0 0
0 0
200 0
Coloana 2: 200@(2,2),
Coloana 3: 0@(0,3), 0@(1,3), 0@(2,3),

Rankul 2@(1,0):
100 100 0 0
0 0 0 0
-100 -100 -100 -100
Coloana 0: 100@(3,0),
Coloana 1: 100@(3,1),
Coloana 4: 0@(3,4), 0@(4,4),
Coloana 5: 0@(3,5), 0@(4,5),

Rankul 3@(1,1):
100 100
0 0
-100 -100
Coloana 2: 100@(3,2),
Coloana 3: 100@(3,3),

----
Rankul 0@(0,0):
0 200 -100 -200
0 0 -100 -200
0 0 -100 -200
Randul 0: 200@(0,1),
Randul 1: 0@(1,0), 0@(1,1),
Randul 2: 0@(2,0), 0@(2,1),

Rankul 1@(0,1):
```

```
100 0
100 0
0 0
Randul 0: 100@(0,2),
Randul 1: 100@(1,2),
Randul 2: 0@(2,2), 0@(2,3),

Rankul 2@(1,0):
0 0 -100 -200
0 0 0 -200
0 0 0 0
Randul 3: 0@(3,0), 0@(3,1),
Randul 4: 0@(4,0), 0@(4,1), 0@(4,4),
Randul 5: 0@(5,0), 0@(5,1), 0@(5,4), 0@(5,5),

Rankul 3@(1,1):
0 0
0 0
0 0
Randul 3: 0@(3,2), 0@(3,3),
Randul 4: 0@(4,2), 0@(4,3),
Randul 5: 0@(5,2), 0@(5,3),

Situatiile Nash de echilibru:
(2, 2), (3, 3), (4, 4),
Total: 3

/home/I01/CemirtanCristian/lucrari_de_laborator/lab5$ mpiexec -host compute-1-1:8 lab5.exe
mtx6.txt 2 4 1 3
Situatiile Nash de echilibru:
(0, 4), (1, 4),
Total: 2
```