|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Univerzitet u Novom Sadu**  **Fakultet Tehničkih Naunka**  **NOVI SAD**  **Projekat**  **Kandidat: Mijat Miletić**  **Broj indeksa: sw66-17**  **Predmet: Paralelno programiranje**  **Tema rada: Množenje matrica**  **Novi Sad, Maj 2019** | |  | |

**Sadržaj**

[1.](#_2r0uhxc) Zadatak 3

2. Analiza 3

3[.](#_1jlao46) Koncept rešenja 4

4[.](#_43ky6rz) Opis rešenja 5

5[.](#_2iq8gzs) Testiranje 9

# 

1. **Zadatak**

Potrebno je realizovati program za množenje kvadratnih matrica. Dimenzije i same matrice se nalaze u jednoj tekstualnoj datoteci, gde je prva linija dimenzija matrice a u ostalim linijama su vrednosti po vrsti matrice. Ukoliko imamo dve matrice sa jedinicama dimenzija 2x2, tekstualna datoteka će izgeldati ovako:

2

1

1

1

1

1

1

1

1

Rezultat množenja matrica se smešta u prvu matricu (matricu A). Potrebno je rezultujuću matricu upisati u tekstualnu datoteku u istom formatu u kojem su zapisane ulazne matrice. Potrebno je realizovati serijsko i paralelno rešenje.

**2. Analiza problema**

Zbog toga što se u zadatku traži da se rezultat množenja upisuje u matricu A, pri računanju nekog reda rezultujuće matrice potrebno je da se rezultati smeštaju u bafer da ne bi došlo do netačnog rezultata. Nakon što se zadati red iz matrice A izmnoži sa svim kolonama matrice B bezbedno je da se bafer presipa u matricu A, u odgovarajući red. Zbog ovoga se javlja problem sinhronizacije u paralelnom rešenju.

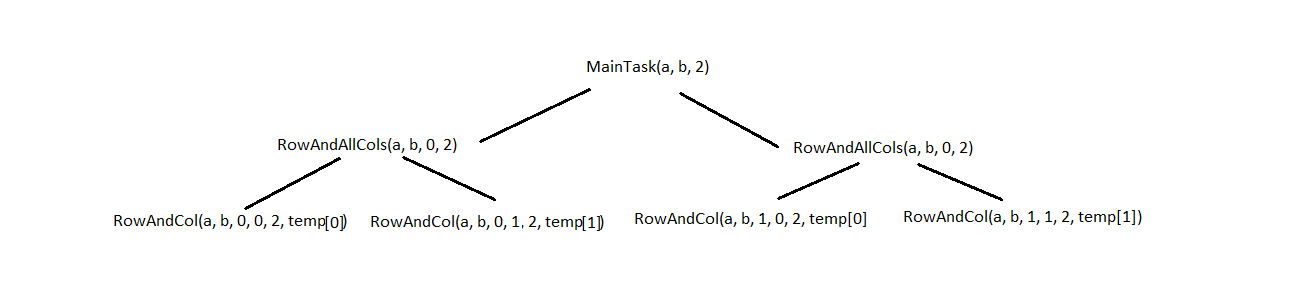
**3. Koncept rešenja**

**3.1 Serijsko rešenje**

Za serijsko rešenje je potrebna jedna funkcija koja ce u sebi imati bafer veličine broja kolona(vrsta) matrice A, koji se puni rezultatima množenja odgovarajuće vrste iz matrice A i odgovarajućih kolona iz matrice B. Nakon što se završi monženje vrste iz A sa svim kolonama iz B sledi presipanje bafera u A u odgovarajuću vrstu. Ovo se radi za svaku vrstu iz matrice A.

**3.2 Paralelno rešenje**

Za paraleno rešenje potrebno je korišćenje tbb zadataka. U rešenju postoje tri tipa zadataka. MainTask koji čeka sve njegove podzadatke da zavrse svoji posao. Zadatak RowAndAllCols (kreira ga MainTask za svaku vrstu iz A) koji ima zadat red iz A i bafer veličine dim koji se presipa u zadat red matrice A nakon što se završe njegovi podzadaci, čime se rešava problem sinhronizacije. Kreira se dim ovakivih zadataka. Zadatak RowAndCol (kreira ga RowAndAllCols za svaku kolonu iz matrice B) koji ima zadat red iz A, vrstu iz B i mesto u baferu na koje upisuje rezultat množenja datog reda i date vrste. Za svaki RowAndAllCols se kreira po dim zadataka, što znači da se na kraju dobije (dim\*dim) RowAndAllCols zadataka + dim RowAndAllCols zadataka + 1 MainTask zadatak. Na kraju se dobije Sledi primer kreiranja zadataka za matricu 2x2.



slika 2.2.1 Primer zadataka za matrixu 2x2

**3.3 Dodatno ubrzanje u oba slučaja**

Dodatno ubrzanje za paralelno i serijsko rešenje se može postići ukolko se matrica B transponuje jer su matrice kvadratne. Uz to mora se promeniti i sam algoritam mnozenja, da bi rezultat ostao tačan. Množejne se obavlja brže jer se kolona matrice B nalazi u jednom nizi, a deo tog niza se nalazi u keš memoriji pa se do elemenata dolazi brže. Matrica se može transponovati prilikom učitavanja u program.

**4. Opis rešenja**

**4.1 Moduli**

**4.1.1 MainTask klasa**

Nasleđuje klasu tbb:task.

Privatna polja:

1. *int* dim - dimenzija matrice
2. *int*\*\* a - matrica A
3. *int*\*\* b - matrica B
4. *task\_list* tasks - lista podzadaka koje mravi ova klasa

Javne metode

1. *MainTask(*int\*\* \_a, int\*\* \_b, int \_dim)
2. *task*\* execute() je nasledjena metoda iz klase tbb::task koja se redefiniše.

Opis metode: Za svaku vrstu iz A kreira objekat tipa *RowAndAllCols* i dodaje ga u listu *tasks,* nakon toga postavlja *ref\_count* na dim *+ 1* i poziva metodu *spawn\_and\_wait\_for\_all* nad listom *tasks.*

**4.1.2 RowAndAllCols klasa**

Nasledjuje klasu tbb::task.

Privatna polja:

1. *int*\*\* a- matrica A
2. *int*\*\* b- matrica B
3. *int dim -* dimenzija
4. *int* row- redni broj reda u matrici A
5. *int*\* temp- bafer
6. *task\_list* tasks *-* lista zadataka

Javne metode:

1. *RowAndAllCols(*int\*\* a, int\*\* b, int row, int dim)

Opis metode: Konstuktor koji inicijalizuje polja klase i dinamički zauzima porstor za bafer od *dim* elemenata i svaki od tih elemenata postavlja na 0.

1. *task*\* execute()

Opis metode: Implementacija nasleđenu metodu koja za svaku kolonu u matrici B kreira objekat tipa *RowAndCol*, zatim ga smešta u listu *tasks* i kada to uradi za sve kolone postavlja polje *ref\_count* na *dim + 1* i poziva metodu *spawn\_and\_wait\_for\_all* nad listom *tasks.*

1. *~RowAndAllCols()*

Opis metode: Destruktor koji poziva *delete[]* nad *temp* jer je taj niz dinamički zauzet.

**4.1.3 RowAndCol klasa**

Nasleđuje klasu tbb::task

Privatna polja:

1. *int*\*\* a - matrica A
2. *int*\*\* b- matrica B
3. *int* row\_a *-* redni broj reda u matrici A
4. *int col\_b* - redni broj kolone u matrici B
5. *int dim -*  dimenzija
6. *int*& temp *-*  metso u baferu

Javne metode:

1. *RowAndCol(*int\*\* a, int\*\* b, int row\_a, int col\_b, int dim, int& temp)
2. *task*\* execute()

Opis metode: mnozi korespodentne elemente row\_avrste matrice A i col\_bkolone matrice B i smesta ih u *temp*.

**4.1.4 Util modul**

Sadrži slobodne funkcije za čitanje i pisanje iz datoteka, kreiranje matrica i kod za serijsko množenje matrica.

Funkcije:

1. *void* save\_results(vector<double>& sts, vector<double>& pts, vector<double>& speedup)

Opis: čuva ubrzanja, vreme u sekundama za paralelno i serijsko rešenje za sve dimenzije zadate u nizu DIMENSIONS

1. *void* open\_file(string file\_name, ifstream& file)

Opis: otvara fajl za čitanje

1. *void* load\_matrices(ifstream& file, int\*\* a, int\*\* a1, int\*\* b, int dim);

Opis: učitava matrice, stim što matricu B ucitava transponovano

1. *void* print\_matrix(int\*\* m, int dim)

Opis: štampa matricu na ekran

1. *void* create\_matrix(string file\_name, int dim)

Opis: kreira matircu datih dimenzija i upisuje je u datoteku zadatu u Defines zaglavlju

1. *void* create\_matrices()

Opis: kreira vise matrica za dimenzije zadate u nizu DIMENSIONS i piše ih u datoteku zadatu u Defines zaglavlju

1. *void* write\_matrix\_to\_file(string path, int\*\* a, int dim)

Opis: zapisuje rezultat mnozenja u zadatu datoteku.

1. *void* multiply\_matrices\_serial(int\*\* a, int\*\* b, int dim)

Opis: serijsko množenja matrica.

**4.1.5 Test modul**

Sadrži slobodne funkcije koje testiraju rad programa.

Funkcije:

1. *bool* compare\_files(string python\_txt\_file, int\*\* test\_matrix, int dim)

Opis: Poredi test matricu sa referentnom matricom koja se nalazi u datoteci na putanji zadatoj parametrom python\_txt\_file radi utvrđivanja tačnosti rešenja.

1. *void* solve(string in\_file\_name, string out\_file\_name, vector<double>& sts, vector<double>& pts, vector<double>& speedup)

Opis: čita, množi, testira, beleži ubrzanje, beleži rezultate

1. *void* solve\_all()

Opis: poziva metodu *solve(string, string)* za sve dimenzije zadate u nizu DIMENSIONS

**4.1.6 Defines modul**

Sadrži konstante za putanje do foldera ulaznih i izlaznih datoteka, dimenzije matrica itd.

**4.1.7 main modul**

Odavde kreće program tako što poziva metodu *void* solve\_all().

**5. Testiranje**

**5.1 Testiranje programa**

Program se izvšava tako što se za svaku dimenziju iz globalnog niza DIMENSIONSpoziva funkcija void solve\_all()koja radi sledece:

1. Učitava matrice ( matricu A učitava dva puta u dva niza, zbog serijskog rešenja )
2. Množi paralelnim rešenjem
3. Dodaje vreme u vektor *vector*<double>& sts
4. Testira tačnost rezultata množenja funkcijom *compare\_files(*string python\_txt\_file, int\*\* test\_matrix, int dim)
5. Množi serijskim rešenjem
6. Dodaje vreme u vektor *vector*<double>& pts
7. Testira tačnost rezultatata množenja funkcijom *compare\_files(*string python\_txt\_file, int\*\* test\_matrix, int dim)
8. Poredi vreme serijskog i paralelnog rešenja i prikazuje koliko je paralelno brže od serijskog
9. Dodaje izračunato ubzanje u vektor vector<double>& speedup
10. Piše rezultujuću matricu u datoteku

Kad uradi ovo za sve dimenzije, vrednosti iz vektora *sts, pts, speedup* zapisuje u 3 različite datoteke čije su putanje zadate u zaglavlju Defines. Ulazne matrice su kreirane nezavisno od ovog testa pozivanjem funkcije void create\_matrices(). Referenta matrica se kreira pomoću pajton skripte koja učitava matrice iz istih fajlova kao i C++ program, množi ih pomoću numpy biblioteke i rešenje zapisuje u datoteku koja se koristi za kontrolu rešenja.

**5.2 Primeri**

U primeru ispod prikazani su rezultati za matrice dimenzija: 10, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 3500, 4000. Brojevi na x osi predstavljaju dimenzije, a brojevi na y osi predstavljaju količinu ubrzanja.

Graf ubrzanja

U ovom primeru se vidi da je ubrzanje u proseku 2.7 (paraleno je 2.7 puta brže od serijskog) izuzev u slučajevima malih dimenzija gde je serijsko rešenje bolje od paralelnog. Na sledećoj stranici se nalazi tablea vremena za serijsko i paralelno rešenje za iste dimenzije.

|  |  |  |  |
| --- | --- | --- | --- |
| Dimenzija | Vreme serijskog u sekundama | Vreme paralelnog u sekundama | Paralelno / serijsko (ubrzanje) |
| 10 | 1.38528e-06 | 0.0023813 | 0.000581734 |
| 100 | 0.000818425 | 0.000510061 | 1.60456 |
| 200 | 0.00627893 | 0.00259159 | 2.42281 |
| 300 | 0.0208867 | 0.0103026 | 2.02732 |
| 400 | 0.0493989 | 0.0177128 | 2.78889 |
| 500 | 0.0934872 | 0.034375 | 2.71963 |
| 600 | 0.177194 | 0.0590164 | 3.00246 |
| 700 | 0.265218 | 0.110897 | 2.39157 |
| 800 | 0.400795 | 0.140948 | 2.84357 |
| 900 | 0.578715 | 0.194464 | 2.97595 |
| 1000 | 0.772774 | 0.267404 | 2.88991 |
| 1500 | 2.71055 | 0.905257 | 2.99423 |
| 2000 | 6.26201 | 2.2617 | 2.76872 |
| 2500 | 12.3107 | 4.4873 | 2.74344 |
| 3000 | 21.0311 | 7.53929 | 2.78954 |
| 3500 | 33.2035 | 11.9473 | 2.77916 |
| 4000 | 49.2398 | 18.2258 | 2.70166 |

Tabela 4.2.1