|  |  |
| --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| **PARALELNÉ PROGRAMOVANIE**  **Zadanie II** | |
|  | |
| **2024** | **Bc. Peter Hovanec, Bc. Ivan Kušlita, Bc. Timotej Sobota** |

**Obsah**

[Zoznam obrázkov 3](#_Toc152538204)

[Zoznam tabuliek 4](#_Toc152538205)

[Formulácia úlohy 5](#_Toc152538206)

[1. Riešenie s využitím sekvenčného algoritmu 6](#_Toc152538207)

[1.1. Algoritmus a programová implementácia navrhnutého riešenia 6](#_Toc152538208)

[1.2. Časová a pamäťová zložitosť navrhnutého riešenia 6](#_Toc152538209)

[1.3. Záver 6](#_Toc152538210)

[2. Riešenie s využitím paralelného algoritmu a POSIX vlákien 7](#_Toc152538211)

[2.1. Algoritmus navrhnutého riešenia 7](#_Toc152538212)

[2.2. Dekompozícia problému a dát 7](#_Toc152538213)

[2.3. Časová a pamäťová zložitosť navrhnutého riešenia 7](#_Toc152538214)

[2.4. Záver 7](#_Toc152538215)

[3. Riešenie s využitím paralelného algoritmu a MPI 8](#_Toc152538216)

[3.1. Algoritmus navrhnutého riešenia 8](#_Toc152538217)

[3.2. Dekompozícia problému a dát 8](#_Toc152538218)

[3.3. Časová a pamäťová zložitosť navrhnutého riešenia 8](#_Toc152538219)

[3.4. Záver 8](#_Toc152538220)

[4. Záver 9](#_Toc152538221)

Zoznam obrázkov

[Obr. 1 Vkladanie popisu pre obrázok 9](#_Toc152536255)

Zoznam tabuliek

[Tab. 1 Štatistické zhodnotenie percenta zhody za rok 2013 9](#_Toc152536256)

Formulácia úlohy

V jazyku C realizujte výpočet, ktorý:

1. Načíta CT snímok s rozlíšením 1024 x 1024 x 314 voxelov zo súboru c8.raw, pričom z neho načíta iba prvých 64 rezov (teda 1024 x 1024 x 64). Každý voxel je zaznamenaný na 8 bitoch ako uint8. Hodnoty voxelov sa pohybujú v rozsahu <0;255>. Voxely sú v súbore usporiadané tak, že predstavujú dump poľa v[x][y][z]. Najskôr sa inkrementuje súradnica X, potom Y a nakoniec Z.
2. Aplikujte prahovanie na každý z voxelov a to tak, že prahovou bude hodnota 25. Vzniknú tak metadáta s 1b/voxel, kde táto hodnota metadát bude
   1. rovná 0 pre voxel, ktorého hodnota je menšia alebo rovná prahovej hodnote a voxel bude považovaný za pasívny
   2. rovná 1 pre voxel, ktorého hodnota je väčšia ako prahová hodnota a voxel bude považovaný za aktívny
3. Vyhľadajte v rámci CT všetky dvojice vzor (A) – kópia (B), kde vzor aj kópia majú rozmer 4 x 4 x 4 voxely. Vzor sa posúva vždy o 4 pixely. Vzor sa zhoduje so svojou kópiou vtedy, keď pre každý z jeho 64 voxelov platí, že ak je príslušný voxel vo vzore aktívny, tak je aktívny aj v kópii a naopak, ak je pasívny vo vzore, tak je rovnako pasívny aj v kópii. Ak bol A vzor a B kópia, tak situáciu, keď je B vzor a A kópia budeme brať do úvahy ako výskyt ďalšej dvojice vzor-kópia.
4. Spočítajte, koľko dvojíc vzor-kópia sa nachádza v načítaných dátach.
5. Výpočet realizujte ako sekvenčný a odmerajte čas potrebný pre realizáciu výpočtu. Meranie začnite po načítaní dát z disku.
6. Výpočet realizujte ako paralelný, s využitím knižnice pthread.h. Zvoľte rôzny počet vlákien, dekomponujte problém a s využitím príslušného paralelizmu realizujte paralelný výpočet. Odmerajte čas potrebný pre realizáciu výpočtu pre rôzny počet vlákien. Meranie začnite vždy po načítaní dát z disku.
7. Výpočet realizujte ako paralelný, s využitím knižnice MPI. Zvoľte rôzny počet procesov, dekomponujte problém a s využitím príslušného paralelizmu realizujte paralelný výpočet. Odmerajte čas potrebný pre realizáciu výpočtu pre rôzny počet procesov. Meranie začnite vždy po načítaní dát z disku.

Do dokumentácie uveďte úlohu, načrtnite spôsob riešenia, ďalej vo forme tabuľky a grafu uveďte čas realizácie výpočtu pri sekvenčnom ako aj paralelnom vykonávaní s rôznym počtom vlákien resp. procesov. Okomentujte namerané časy. Vyjadrite sa k časovej zložitosti použitých problémov resp. algoritmov. Uveďte zdrojový kód programu.

1. Riešenie s využitím sekvenčného algoritmu

Cieľom je analyzovať CT dáta a identifikovať všetky dvojice 4x4x4 vzorov, ktoré sú kópiami. Vzor a kópia sa považujú za rovnaké, ak sú všetky ich voxely buď aktívne, alebo pasívne v rovnakých pozíciách. Tento výpočet sa realizuje sekvenčne, pričom sa meria čas potrebný na spracovanie.

* 1. Algoritmus a programová implementácia navrhnutého riešenia
     1. Načítanie dát:

Program číta vstupné dáta zo súboru c8.raw. Predpokladá sa, že dáta sú v surovom binárnom formáte.

* + 1. Prahovanie voxelov a tvorba metadát:

Na každý voxel sa aplikuje prahová hodnota 25.

Všetky voxely nad prahovou hodnotou (aktivované) sú uložené v poli metadát ako bity (1 pre aktivovaný, 0 pre neaktivovaný voxel).

* + 1. Extrahovanie 3D vzorov:

Trojrozmerný priestor je rozdelený na bloky 4×4×4. Každý takýto blok je reprezentovaný ako 64-bitový vzor (1 bit na voxel).

Tieto vzory sú uložené v poli.

* + 1. Triedenie vzorov a počítanie kópií:

Vzory sú usporiadané pomocou **qsort**, čo umožňuje efektívnejšie vyhľadávanie opakujúcich sa vzorov.

Počet párov identických vzorov je vypočítaný ako kombinácia **N \* (N - 1)** pre každý skupinový vzor.

* + 1. Výstup:

Program vypíše počet aktívnych a neaktívnych voxelov, počet dvojíc identických vzorov a čas spracovania.

**Rámcová implementácia v jazyku C**

**1. Načítanie a inicializácia:**

Súbor **c8.raw** je otvorený na spracovanie:

1. openFile("c8.raw");

2.

3. start = clock();

4.

5. uint8\_t \*metadata = (uint8\_t \*)calloc((WIDTH \* HEIGHT \* DEPTH) / 8, sizeof(uint8\_t));

6. if (metadata == NULL)

7. {

8. perror("Failed to allocate metadata");

9. terminate(1);

10. }

11.

**2. Prahovanie a tvorba bitového poľa metadát:**

**Každý voxel je prepočítaný a uložený ako bit do poľa metadata:**

1. for (int z = 0; z < DEPTH; z++)

2.     {

3.         for (int y = 0; y < HEIGHT; y++)

4.         {

5.             for (int x = 0; x < WIDTH; x++)

6.             {

7.                 int index = (z \* WIDTH \* HEIGHT) + (y \* WIDTH) + x;

8.                 int bitIndex = index % 8;

9.                 int byteIndex = index / 8;

10.

11.                 if (voxels[z][y][x] > 25)

12.                 {

13.                     metadata[byteIndex] |= (1 << bitIndex);

14.                     count\_one\_voxels++;

15.                 } else {

16.                     count\_zero\_voxels++;

17.                 }

18.             }

19.         }

20.     }

21.

**3. Extrahovanie 4×4×4 vzorov:**

**Pre každú 4×4×4 bunku sa vytvorí 64-bitový vzor:**

1. uint64\_t \*patterns = (uint64\_t \*)malloc(total\_patterns \* sizeof(uint64\_t));

2. if (patterns == NULL)

3. {

4. perror("Failed to allocate patterns array");

5. terminate(1);

6. }

7.

8. int pattern\_index = 0;

9. for (int z = 0; z < DEPTH; z += PATTERN\_SIZE)

10. {

11. for (int y = 0; y < HEIGHT; y += PATTERN\_SIZE)

12. {

13. for (int x = 0; x < WIDTH; x += PATTERN\_SIZE)

14. {

15. uint64\_t pattern = 0;

16. int bit = 0;

17. for (int dz = 0; dz < PATTERN\_SIZE; dz++) // 4

18. {

19. for (int dy = 0; dy < PATTERN\_SIZE; dy++) // 4

20. {

21. for (int dx = 0; dx < PATTERN\_SIZE; dx++) // 4

22. {

23. int voxel\_x = x + dx;

24. int voxel\_y = y + dy;

25. int voxel\_z = z + dz;

26.

27. int index = (voxel\_z \* WIDTH \* HEIGHT) + (voxel\_y \* WIDTH) + voxel\_x;

28. int bitIndex = index % 8;

29. int byteIndex = index / 8;

30.

31. uint8\_t voxel\_bit = (metadata[byteIndex] >> bitIndex) & 1;

32.

33. if (voxel\_bit)

34. {

35. pattern |= ((uint64\_t)1 << bit);

36. }

37. bit++;

38. }

39. }

40. }

41. patterns[pattern\_index++] = pattern;

42. }

43. }

44. }

45.

**4. Triedenie vzorov:**

**Vzory sú utriedené pre efektívne vyhľadávanie:**

1. qsort(patterns, total\_patterns, sizeof(uint64\_t), compare\_uint64);

**5. Spočítanie identických vzorov:**

**Pre každý vzor sa počíta počet identických výskytov:**

1.     uint64\_t current\_pattern = patterns[0];

2.     uint64\_t count = 1;

3.     uint64\_t total\_pairs = 0;

4.

5.     for (int i = 1; i < total\_patterns; i++)

6.     {

7.         printf("%lu\n", patterns[i]);

8.         if (patterns[i] == current\_pattern)

9.         {

10.             count++;

11.         }

12.         else

13.         {

14.             total\_pairs += count \* (count - 1); // N \* (N - 1)

15.             current\_pattern = patterns[i];

16.             count = 1;

17.         }

18.     }

19.

20.     total\_pairs += count \* (count - 1);

21.

**6. Výstup výsledkov:**

**Na konci program vypíše štatistiky:**

1. printf("Total matching pattern-copy pairs: %'lu\n", total\_pairs);

2.     printf("0 VOXELS: %ld 1 VOXELS: %ld\n", count\_zero\_voxels, count\_one\_voxels);

3.     end = clock();

4.     function\_time = ((double)(end - start)) / CLOCKS\_PER\_SEC;

5.

6.     printf("Sequential computation time: %f seconds\n", function\_time);

7.

8.

* 1. Časová a pamäťová zložitosť navrhnutého riešenia

Popíšte časovú a pamäťovú zložitosť vami navrhnutého riešenia, popíšte, aké parametre mal počítač, na ktorom bolo realizované testovanie a namerané časy vykonávania programu (urobte 12 meraní, z ktorých vylúčite najkratší a najdlhší čas realizácie programu a urobte priemer 10 zostávajúcich časov). Výsledky meraní uveďte vo forme tabuľky.

* 1. Záver

Zhodnoťte dosiahnuté riešenie.

1. Riešenie s využitím paralelného algoritmu a POSIX vlákien

Sem vložte minimálny úvod, stačí dva až tri riadky aby medzi nadpismi bol vložený text.

* 1. Algoritmus navrhnutého riešenia

Popíšte vami navrhnuté riešenie v podobe paralelného algoritmu a rámcovo jeho riešenie v jazyku C, s využitím POSIX vlákien.

* 1. Dekompozícia problému a dát

Popíšte, ako ste dekomponovali problém na podproblémy, aby bolo možné navrhnúť paralelný algoritmus, aký typ paralelizmu bol použitý a ako boli využité POSIX vlákna.

* 1. Časová a pamäťová zložitosť navrhnutého riešenia

Popíšte časovú a pamäťovú zložitosť vami navrhnutého riešenia, popíšte, aké parametre mal počítač, na ktorom bolo realizované testovanie a namerané časy vykonávania programu (urobte 12 meraní, z ktorých vylúčite najkratší a najdlhší čas realizácie programu a urobte priemer 10 zostávajúcich časov). Výsledky meraní uveďte vo forme tabuľky a grafu. Urobte pritom merania pri využití rôzneho počtu vlákien, pričom uveďte graf, pri ktorom na ose x bude uvedený počet použitých vlákien a na ose y nameraný čas vykonávania programu.

* 1. Záver

Zhodnoťte dosiahnuté riešenie.

1. Riešenie s využitím paralelného algoritmu a MPI

Sem vložte minimálny úvod , stačí dva až tri riadky aby medzi nadpismi bol vložený text.

* 1. Algoritmus navrhnutého riešenia

Popíšte vami navrhnuté riešenie v podobe paralelného algoritmu a rámcovo jeho riešenie v jazyku C s využitím MPI.

* 1. Dekompozícia problému a dát

Popíšte, ako ste dekomponovali problém na podproblémy, aby bolo možné navrhnúť paralelný algoritmus, aký typ paralelizmu bol použitý a ako boli využité funkcie MPI.

* 1. Časová a pamäťová zložitosť navrhnutého riešenia

Popíšte časovú a pamäťovú zložitosť vami navrhnutého riešenia, popíšte, aké parametre mal počítač, na ktorom bolo realizované testovanie a namerané časy vykonávania programu (urobte 12 meraní, z ktorých vylúčite najkratší a najdlhší čas realizácie programu a urobte priemer 10 zostávajúcich časov). Výsledky meraní uveďte vo forme tabuľky a grafu. Urobte pritom merania pri využití rôzneho počtu procesných elementov, pričom uveďte graf, pri ktorom na ose x bude uvedený počet použitých procesných elementov a na ose y nameraný čas vykonávania programu.

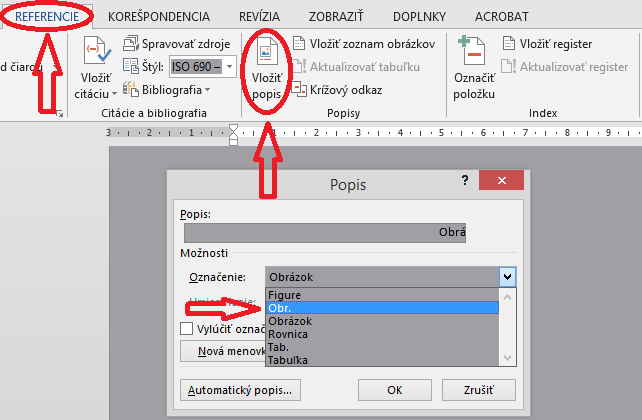
* 1. Záver

Zhodnoťte dosiahnuté riešenie.

1. Záver

Sem uveďte celkový záver, v ktorom zhodnotíte aká najlepšia časová a pamäťová zložitosť bola dosiahnutá, ktorým riešením, aká časť zadania bola úspešne realizovaná a aká časť zadania nebola úspešne realizovaná.

*Nasleduje príklad uvedenia popisu obrázka a tabuľky, aby bolo možné zabezpečiť automatické generovanie zoznamu obrázkov a tabuliek, vo finálnej verzii dokumentácie zadania túto časť zmažte.*



Obr. 1 Vkladanie popisu pre obrázok

Tab. 1 Štatistické zhodnotenie percenta zhody za rok 2013

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **% zhoda** | **FBERG** | | | **HF** | | | **SJF** | | | **FEI** | | | **SVF** | | | **FVT** | | | **EKF** | | | **FU** | | | **LF** | | |
| **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** | **BC** | **ING** | **PHD** |
| 0 - 10% | 281 | 321 | 27 | 110 | 54 | 27 | 377 | 359 | 28 | 431 | 382 | 23 | 93 | 94 | 24 | 150 | 168 | 8 | 181 | 173 | 14 | 39 | 27 |  | 85 | 200 | 4 |
| 11 - 20% | 55 | 78 | 5 | 12 | 14 | 1 | 37 | 72 | 3 | 17 | 31 |  | 35 | 32 | 3 | 33 | 51 | 1 | 24 | 19 |  | 1 | 3 |  | 22 | 30 | 1 |
| 21 - 30% | 26 | 40 | 4 | 6 | 8 |  | 21 | 24 | 1 | 6 | 11 |  | 16 | 15 | 1 | 19 | 31 | 2 | 4 | 4 | 1 | 2 | 1 |  | 7 | 20 | 1 |
| 31 - 40% | 10 | 20 |  | 1 | 2 |  | 7 | 9 |  | 5 | 2 |  | 6 | 10 |  | 7 | 13 | 1 |  | 1 |  |  |  |  | 2 | 13 |  |
| 41 - 50% | 14 | 10 |  |  |  |  | 4 | 1 |  | 1 | 1 |  | 4 | 3 |  |  | 3 |  |  |  |  |  |  |  | 1 | 7 |  |
| 51 - 60% | 2 | 1 | 1 | 2 |  |  | 4 | 2 |  | 3 |  |  | 1 |  |  | 2 | 2 |  | 2 | 2 |  |  |  |  | 2 | 2 |  |
| 61 - 70% | 1 | 1 |  |  |  |  | 5 |  |  | 3 |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |