**ЛАБОРАТОРНА РОБОТА #1**

З дисципліни «Комп’ютерні системи»

На тему «Багатопроцесорні обчислювальні системи із роздільною (спільною) пам’яттю»

|  |  |
| --- | --- |
| **Виконав**  Студент групи КВ-21  Комарницький О.Б. | **Перевірила**  Доцент кафедри СП і СКС  Потапова К.Р. |

|  |  |  |
| --- | --- | --- |
| **Оцінка** | **Дата** | **Підпис** |
|  |  |  |

**Варіант #8 -> 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **# варіант** | **Кількість процесорів** | **Коефіцієнт**  **α** | **Коефіцієнт**  **β** | **Обчислювальний**  **алгоритм** |
| 2 | 2 | 3 | 4 | Додавання двох матриць |

**Завдання:**

обчислювальні системи із роздільною пам’яттю:

1. За заданим алгоритмом побудувати граф, в якому вершинам відповідають обчислення розгалужень алгоритму, а дугам – зв’язки (інформаційні або через пам'ять) між гілками алгоритму.
2. Визначити значення T0 (в тактах) для кожної гілки алгоритму враховуючи те, що t+=1такт, t\*= α\* t+, t/= β\* t+;
3. Виконати задачу оптимального тактового розподілу (планування) обчислювальних розгалужень алгоритму із процесорами ОС.
4. Перевірити правильність виконання задачі планування на програмній моделі.
5. Порівняти показники ефективності роботи багатопроцесорної ОС при ручному та автоматичному режимі управління планування обчислень.
6. Проаналізувати ефективність функціонування даної ОС за результатами роботи програмної моделі.

обчислювальні системи із спільною пам’яттю:

1. За заданим алгоритмом побудувати граф, в якому вершинам відповідають обчислення розгалужень алгоритму, а дугам – зв’язки (інформаційні або через пам'ять) між гілками алгоритму.
2. , 3), 4), 5) – ті ж, що 3), 4), 5), 6) для ОС типу М1

**Обчислювальні системи із роздільним управлінням:**

1. Граф алгоритму

j++

Read j

i++

Read A[i][j]

Read i

Read M

i >= M - 1

Write C[i][j]=A[i][j]+B[i][j]

Read N

j >= N - 1

Read B[i][j]

1. Значення T0

Кожна незалежна гілка процесу додавання матриці містить лише одну дію: додавання двох комірок пам’яті. Тому T0 = 1 такт.

1. Оптимальне планування розгалуження обчислювального процесу

Оскільки пам’ять у обчислювальній системі М1 є спільною для усіх процесорів, і необхідний розподілений доступ до спільного ресурсу, тому вирішення задачі неможливе без методів синхронізації. В даній моделі використовуються критичні секції.

Як відомо, поки один процес використовує спільний ресурс, інший змушений чекати для доступу до нього. Тому задача оптимального розподілу вимагає якомога меншу кількість команд у критичній секції.

Даний алгоритм містить в критичній секції лише зчитування та інкрементацію лічильників i, j. Натомість, дії додавання і запис готового результату реалізовуються паралельно, адже запис здійснюється у дві незалежні комірки пам’яті.

Код мовою С++ процесу, що здійснює додавання комірок матриці:

void process\_M1(int id)

{

int i, j;

while (true)

{

if (mem.i == mem.M)

break;

EnterCriticalSection(&cs);

i = mem.i, j = mem.j;

if (mem.j == mem.N - 1)

{

mem.j = 0;

mem.i++;

}

else

mem.j++;

LeaveCriticalSection(&cs);

mem.C[i][j] = mem.A[i][j] + mem.B[i][j];

}

}

1. Перевірка правильності роботи на програмній моделі

Оскільки поставлена задача – реалізувати додавання комірок матриці, сформувати нову матрицю *С*, де кожен елемент матриці *С* є сумою відповідних елементів матриць *А* і *В*.

Виведемо на екран консолі запис у вигляді:

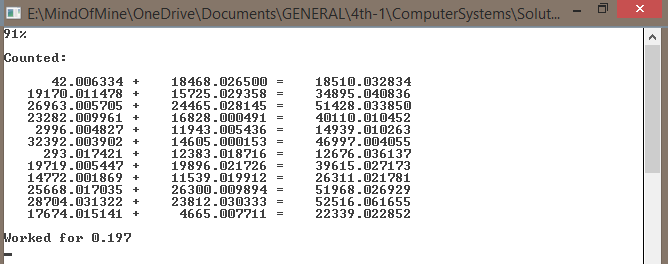
void output()

{

for (int i = 0; i < mem.N; ++i)

for (int j = 0; j < mem.M; ++j)

cout << mem.A[i][j] << " + " << mem.B[i][j] << " = " << mem.C[i][j] << endl;

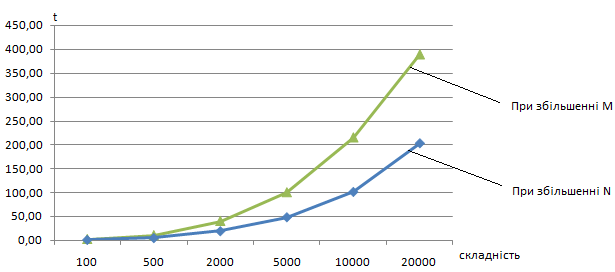
}

1. Проаналізувати ефективність функціонування даної ОС за результатами роботи програмної моделі.

**Показники ОС М1 Показники синхронної роботи ОС**

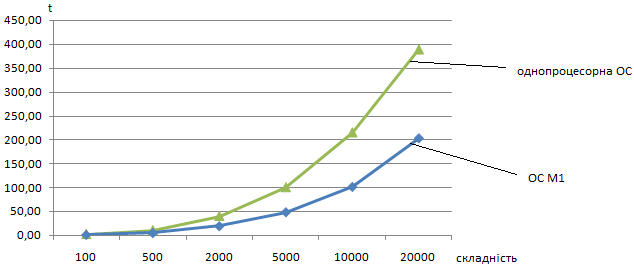
|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 1.172 |
| 10\*50 | | 5.017 |
| 10\*200 | | 19.401 |
| 10\*500 | | 48.561 |
| 10\*1000 | | 101.591 |
| 10\*2000 | | 203.375 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 1.202 | |
| 50\*10 | 5.296 | |
| 200\*10 | 21.752 | |
| 500\*10 | 54.444 | |
| 1000\*10 | 117.705 | |
| 2000\*10 | 216.734 | |

|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 1.91 |
| 10\*50 | | 10.146 |
| 10\*200 | | 39.859 |
| 10\*500 | | 100.062 |
| 10\*1000 | | 215.26 |
| 10\*2000 | | 389.652 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 2.03 | |
| 50\*10 | 9.651 | |
| 200\*10 | 38.744 | |
| 500\*10 | 95.019 | |
| 1000\*10 | 195.738 | |
| 2000\*10 | 390.746 | |

Графіки залежності часу виконання обчислень від складності задачі (кількості елементів масивів) на ОС М1:

**Висновок:**

Графіки залежності відрізняються через більшу кількість операцій при збільшенні кількості рядків, адже при перемотуванні лічильника *і* більш часто обнуляється лічильник *j* та інкрементуємо лічильник *і*, коли ж менша кількість рядків – відбувається більш часто лише операція інкрементації лише лічильника *j.* Оскільки ці команди знаходяться в критичній секції, тому цей фактор сильно впливає на роботу системи.

Графіки для порівняння швидкодії ОС М1 та звичайної одно процесорної системи:

**Висновок:**

Із графіку помітно, що із ростом складності (кількості елементів матриць) краще справляється 2-процесорна ОС із роздільним управлінням. Це пояснюється завдяки паралельному обчисленні незалежних комірок матриць.

Також варто відзначити, що графік ОС М1 має гіперболічний характер, а не лінійний, що пояснюється збільшенням кількості проміжної арифметичної дії - переведення каретки із кінця строки на початок

**Обчислювальні системи із спільним управлінням:**

1. Граф алгоритму

Read i, j

Read A[i][j]

read B[i][j]

Read M, N

j >= N - 1

i >= M - 1

Write C[i][j]=A[i][j]+B[i][j]

j++

i++

1. Оптимальне планування розгалуження обчислювального процесу

Оскільки доступ спільної пам’яті є тільки у пристрою управління, тому механізм синхронізації для обчислювальної системи М2 не потрібен.

Для даної системи не потрібна така кількість ручних перевірок лічильників, на відміну від ОС М1. Тому виконання додавання матриць для пристрою управління можна подати як звичайний цикл, що проходить по всім елементам матриці. Проте, необхідний ще один цикл, який моделює “очікування” вільного процесора.

Кожен процесор слугує лише арифметично-логічним пристроєм, що виконує дію додавання двох елементів, які вказує пристрій управління. Тому код процесора виглядає так:

void process\_M2(int i, int j, int id) {

mem2.C[i][j] = mem2.A[i][j] + mem2.B[i][j];

(id == 1) ? (p1 = FREE) : (p2 = FREE);

}

Кожен процесор виконує паралельно такі дії: отримує задачу, стає зайнятим, здійснює обчислення та знову стає вільним.

Так виглядає код головного циклу пристрою управління:

void processor\_M2()

{

for (int i = 0; i < mem2.M; ++i)

{

for (int j = 0; j < mem2.N; ++j)

{

while (true)

{

if (p1 == FREE)

{

p1 = BUSY;

async(process\_M2, i, j, 1);

break;

}

else if (p2 == FREE)

{

p2 = BUSY;

async(process\_M2, i, j, 2);

break;

}

}

}

}

}

1. Перевірка правильності роботи на програмній моделі:

Оскільки поставлена задача – реалізувати додавання комірок матриці, сформувати нову матрицю *С*, де кожен елемент матриці *С* є сумою відповідних елементів матриць *А* і *В*.

Виведемо на екран консолі запис у вигляді:

void output()

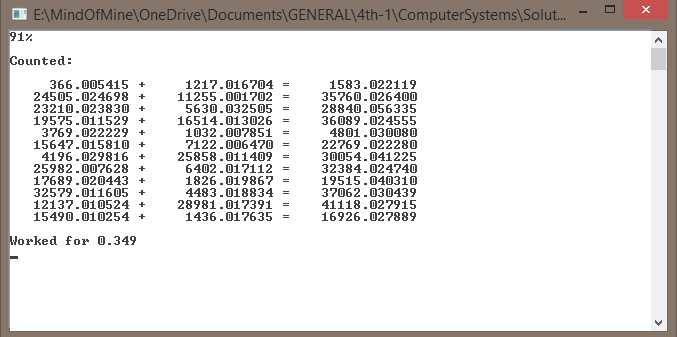
{

for (int i = 0; i < mem.N; ++i)

for (int j = 0; j < mem.M; ++j)

cout << mem.A[i][j] << " + " << mem.B[i][j] << " = " << mem.C[i][j] << endl;

}



1. Проаналізувати ефективність функціонування даної ОС за результатами роботи програмної моделі.

**Показники ОС М2 Показники синхронної роботи ОС**

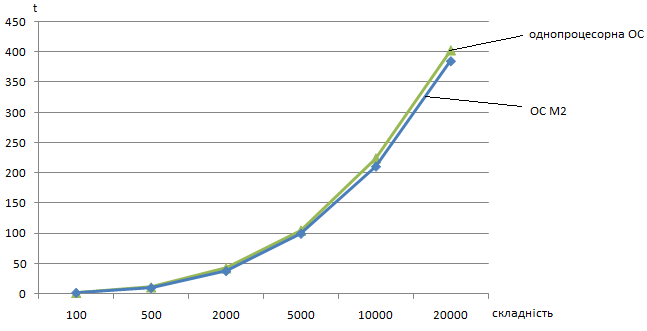
|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 1.974 |
| 10\*50 | | 10.089 |
| 10\*200 | | 38 |
| 10\*500 | | 99.852 |
| 10\*1000 | | 210.87 |
| 10\*2000 | | 385.294 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 2.479 | |
| 50\*10 | 10.661 | |
| 200\*10 | 48.288 | |
| 500\*10 | 94.916 | |
| 1000\*10 | 213.802 | |
| 2000\*10 | 388.581 | |

|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 2.0 |
| 10\*50 | | 11.146 |
| 10\*200 | | 42.859 |
| 10\*500 | | 105.062 |
| 10\*1000 | | 220.26 |
| 10\*2000 | | 394.652 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 2.03 | |
| 50\*10 | 9.651 | |
| 200\*10 | 38.744 | |
| 500\*10 | 95.019 | |
| 1000\*10 | 215.738 | |
| 2000\*10 | 390.746 | |

**Висновок:**

Як помітно із показників швидкодії системи М2, графіки залежності часу виконання від кількості елементів матриці в двох випадках (при збільшенні стовпців та рядків) практично не відрізняються. Це пояснюється тим, що кількість дій, виконаних в обох випадках, однакова, бо інкрементація лічильників відбувається лише в пристрою управління.

Графіки для порівняння швидкодії ОС М2 та звичайної одно процесорної системи:



**Висновок:**

Із графіку помітно, що обчислювальна система типу М2 завдяки паралелізму має дещо кращу швидкодію, ніж однопроцесорна ОС. З іншого боку, різниця дуже незначна, що пояснюється своєрідною реалізацією моделі обчислювальної системи із спільним управлінням: кожна дія додавання елементів матриць, що розпаралелюється, виконується через виклики асинхронних функцій, що займає багато часу, на відміну від ОС М1, де функції процесорів запускаються лише по одному разу.

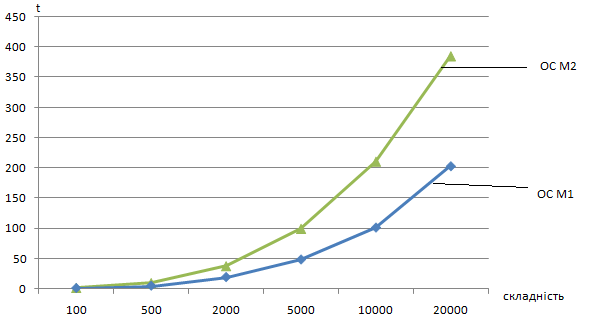
**Порівняння ОС М1 та М2:**

**Показники ОС М1 Показники ОС М2**

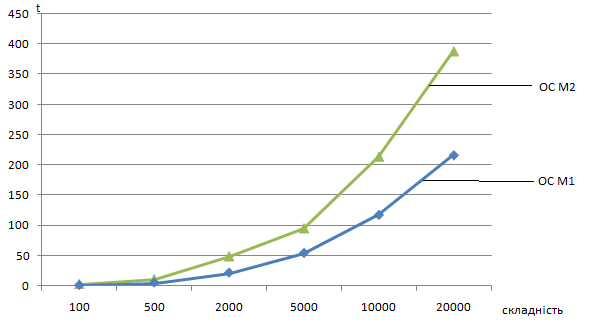
|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 1.172 |
| 10\*50 | | 5.017 |
| 10\*200 | | 19.401 |
| 10\*500 | | 48.561 |
| 10\*1000 | | 101.591 |
| 10\*2000 | | 203.375 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 1.202 | |
| 50\*10 | 5.296 | |
| 200\*10 | 21.752 | |
| 500\*10 | 54.444 | |
| 1000\*10 | 117.705 | |
| 2000\*10 | 216.734 | |

|  |  |  |
| --- | --- | --- |
| **Кількість елементів** | | **Час виконання** |
| Із збільшенням к-сті стовпців | | |
| 10\*10 | | 1.974 |
| 10\*50 | | 10.089 |
| 10\*200 | | 38 |
| 10\*500 | | 99.852 |
| 10\*1000 | | 210.87 |
| 10\*2000 | | 385.294 |
| Із збільшенням к-сті рядків | | |
| 10\*10 | 2.479 | |
| 50\*10 | 10.661 | |
| 200\*10 | 48.288 | |
| 500\*10 | 94.916 | |
| 1000\*10 | 213.802 | |
| 2000\*10 | 388.581 | |

Графіки для порівняння швидкодії ОС М1 та ОС М2 при збільшенні к-сті стовпців:



Графіки для порівняння швидкодії ОС М1 та ОС М2 при збільшенні к-сті рядків:



**Висновок:**

Як помітно із обох рисунків, система типу М1 значно переважає швидкодією ОС М2.

Це пояснюється тим, що:

* в ОС М2 у випадку зайнятості відразу обох процесорів відбуваються зайві ітерації запитів всередині циклу;
* велика кількість викликів функції в системі типу М2, що при дизасемблюванні програмного коду С++ впливають на швидкодію;
* елементарність поставленої задачі – додавання матриць, адже виходить так, що для додавання двох чисел по-новому запускається нова функція;

Отже, для реалізації для додавання двох матриць більше підходить обчислювальна система із роздільним управлінням.