

Лекція 4. Декомпозиція потоків за даними

Розбиття масиву за діапазоном. Вирівнювання навантаження на процесори за допомогою кругового алгоритму. Збирання результатів роботи окремих процесорів.

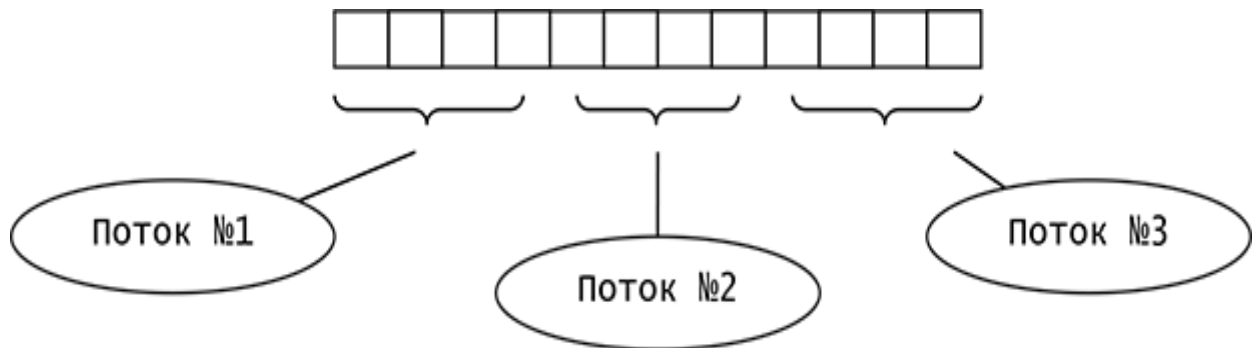
1. Розподіл даних по різних потоках

Декомпозиція за даними. При декомпозиції задач за даними кожна під задача (потік) виконує одні і ті самі дії, але з різними даними, наприклад, обробку великого масиву (колекції) даних.

Для цього у функції потоку необхідно передбачити можливість розбиття діапазону 0, (N - 1) на потрібну кількість потоків. При запуску потоку йому потрібно передати як аргумент деякі дані про діапазон.

Це може бути або "індекс потоку", що визначає область масиву, який обробляється в цьому потоці, або початковий і кінцевий індекси масиву.

Це розбиття визначає сам розробник.



Приклад 1. Обчислення суми елементів масиву. Послідовна реалізація.

В програмі створюється клас `MyThread` з двома методами. В методі `GenMas` створюється масив і заповнюється даними. Можна використати випадкові числа, але в даному випадку це утруднює порівняння результатів обчислення. Числа генеруються як ряд послідовних цілих чисел 1,2,...,n. Кількість чисел передається як параметр.

В методі `RunMas` обчислюється сума. Методи викликаються послідовно. Масив визначається на рівні класу.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
namespace Lab3
{
    //Генерація масиву чисел та їх послідовна обробка
    class MyThread
    {
        int[] Data; // масив чисел
        long count; //сума чисел

        public void GenMas(int n)
```

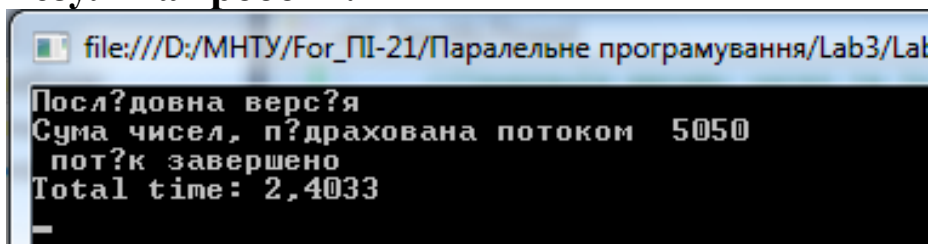
```

{
    // заповнення масиву.
    Data = new int[n]; // масив чисел
    for (int i = 0; i < Data.Length; i++)
        Data[i] = i+1;
//    Console.WriteLine("Генерація чисел і заповнення масиву");
//    for (int i = 0; i < Data.Length; i++)
//        Console.WriteLine("Data[" + i + "] = " + Data[i]);
}
public void RunMas1(object num)
{
    // num кількість чисел масиву, які обчислюються
    // Зверніть увагу на те, що в цій формі методу RunMas()
    // вказується параметр типа object.
    int n=(int)num;
    count = 0;
    for (int i=0;i<n;i++)
        count += Data[i];
    Console.WriteLine("Сума чисел, підрахована потоком " + count);
    Console.WriteLine(" потік завершено");
}
}

class Program
{
    static void Main()
    {
        Stopwatch sw = new Stopwatch();
        sw.Start();
        Console.WriteLine("Послідовна версія");
        MyThread mt = new MyThread(); // об'єкт
        //заповнення масиву
        mt.GenMas(100);
        mt.RunMas1(100);
        sw.Stop();
        TimeSpan ts = sw.Elapsed;
        Console.WriteLine("Total time: {0}", ts.TotalMilliseconds);
        Console.ReadKey();
    }
}
}

```

Результат роботи:



Приклад 2. Паралельна версія. Розбиття масиву за діапазоном.

Модифікуємо програму таким чином. Спочатку викликається метод заповнення масиву числами. Потім створюється 2 вторинні потоки, які викликають метод RunMas для обчислення часткових сум. У методі Start для методу RunMas потрібно передати параметри. Враховуємо те, що в метод Start можна передати тільки 1 параметр, тому передаємо параметри через масив типу object. У якості параметрів передаємо нижню і верхню границі підмасивів, а також ознаку частини масиву (1 і 2).

```
thrd1.Start(new object[] { 0, 50, 1 }); //1- перша частина масиву
thrd2.Start(new object[] { 50, 100, 2 }); //2 - друга частина масиву
```

Результати обчислення кожного потоку зберігаються у властивостях класу

```
public long Count1 {get;set;}
public long Count2 {get;set;}
```

В методі обчислення суми RunMas розпаковуємо параметри, обчислюємо часткову суму. Далі, в залежності від номера під масиву призначаємо обчислену часткову суму відповідній властивості Count1 чи Count2.

```
public void RunMas(object obj)
{
    // num - частина масиву, яка обчислюється
    // Зверніть увагу на те, що в цій формі методу RunMas()
    // вказується параметр типа object.
    int min = (int)((object[])obj)[0];
    int max = (int)((object[])obj)[1];
    int nomtrd=(int)((object[])obj)[2];
    count = 0;
    for (int i=min;i<max;i++)
        count += Data[i];
    if (nomtrd == 1)
    {
        Count1 = count;
        Console.WriteLine("Сума чисел, підрахована першим потоком " + count);
    }
    else
    {
        Count2 = count;
        Console.WriteLine("Сума чисел, підрахована другим потоком " + count);
    }
    Console.WriteLine("потік завершено ");
}
}
```

В основній програмі в методі main() призупиняється виконання основного потоку до завершення вторинних потоків.

Після цього значення полів класу призначаються змінним і обчислюється загальна сума.

```
totalSumma = mt.Count1 + mt.Count2;
```

Повний код програми:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Diagnostics;
namespace Lab3
{
    //Генерація масиву чисел та їх послідовна обробка
    class MyThread
    {
        int[] Data; // масив чисел
        long count; //сума чисел
        public long Count1 {get;set;}
        public long Count2 {get;set;}
        public void GenMas(int n)
        {
            // заповнення масиву.
            Data = new int[n]; // масив чисел
            for (int i = 0; i < Data.Length; i++)
                Data[i] = i+1;
            Console.WriteLine("Генерація чисел і заповнення масиву");
        }
        public void RunMas(object obj)
        {
            // num - частина масиву, яка обчислюється
            // Зверніть увагу на те, що в цій формі методу RunMas()
            // вказується параметр типа object.
            int min = (int)((object[])obj)[0];
            int max = (int)((object[])obj)[1];
            int nomtrd=(int)((object[])obj)[2];
            count = 0;
            for (int i=min;i<max;i++)
                count += Data[i];
            if (nomtrd == 1)
            {
                Count1 = count;
                Console.WriteLine("Сума чисел, підрахована першим потоком " + count);
            }
            else
            {
                Count2 = count;
                Console.WriteLine("Сума чисел, підрахована другим потоком " + count);
            }
            Console.WriteLine("потік завершено ");
        }
    }

    class Program
    {
        static void Main()
        {
            Stopwatch sw = new Stopwatch();
            sw.Start();
            Thread thrd1, thrd2;
            Console.WriteLine("Паралельна версія - розбиття масиву на 2 частини ");
            MyThread mt = new MyThread(); // об'єкт
            //заповнення масиву
            mt.GenMas(100);
            //створення 2 потоків для парал.виконання RunMas
            long totalSumma;
            // створення потоків
            thrd1 = new Thread(mt.RunMas);

```

```

        thrd2 = new Thread(mt.RunMas);
        // Тут змінна num передається методу Start ()
        // як аргумент.
        thrd1.Start(new object[] { 0, 50, 1 }); //1- перша частина масиву
        thrd2.Start(new object[] { 50, 100, 2 }); //2 - друга частина масиву
        thrd1.Join();
        thrd2.Join();
        totalSumma = mt.Count1 + mt.Count2;
        Console.WriteLine("Сума першої частини масиву = " + mt.Count1);
        Console.WriteLine("Сума другої частини масиву = " + mt.Count2);
        Console.WriteLine("Загальна сума. = " + totalSumma);
        Console.WriteLine("Основний потік завершено.");
        sw.Stop();
        TimeSpan ts = sw.Elapsed;
        Console.WriteLine("Total time: {0}", ts.TotalMilliseconds);
        Console.ReadKey();
    }
}
}

```

Результати роботи:

```

file:///D:/МНТУ/For_ПІ-21/Паралельне програмування/Lab3/Lab3/bin/Debug/Lab3.EXE
Паралельна версія - розбиття масиву на 2 частини
Генерація чисел? заповнення масиву
Сума чисел, підрахована другим потоком 3775
потік завершено
Сума чисел, підрахована першим потоком 1275
потік завершено
Сума першої частини масиву = 1275
Сума другої частини масиву = 3775
Загальна сума. = 5050
Основний потік завершено.
Total time: 26,2298

```

Результати обчислення співпадають з результатами послідовної версії.

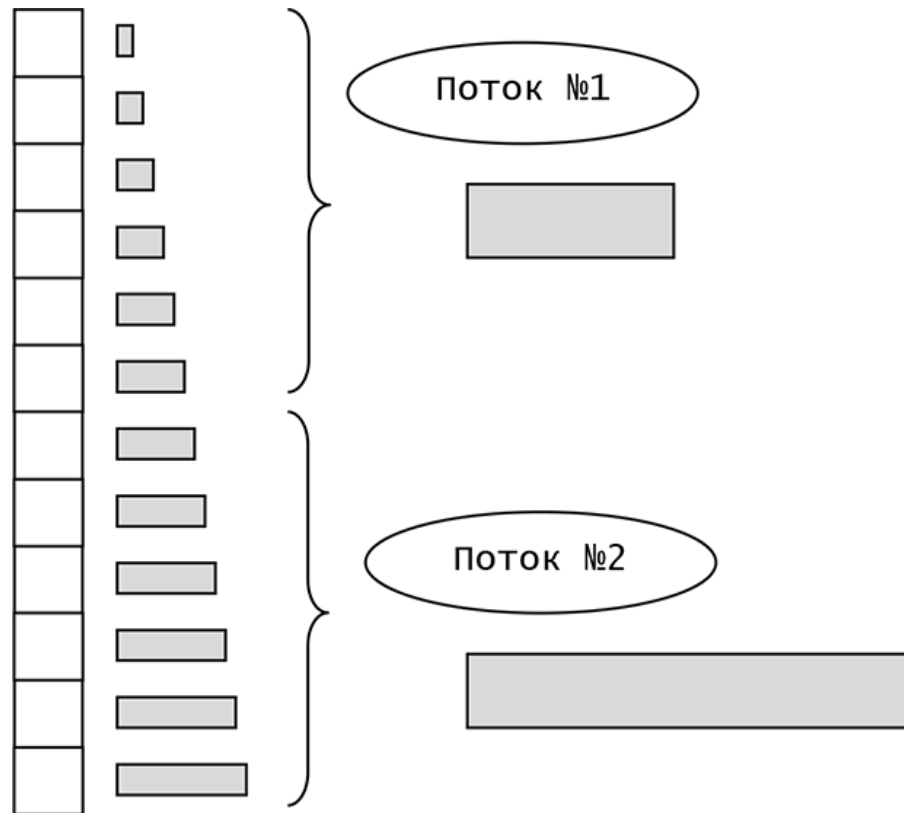
А час роботи послідовної версії менший ніж паралельної. Це можна пояснити накладними витратами на розпаралелювання. Отже, для невеликих розмірів масиву, розпаралелювання не дає переваг у продуктивності.

2. Вирівнювання навантаження на процесори

Приклад 3. Паралельна версія. Використання кругового алгоритму.

Наведений алгоритм розбиття масиву на частини за діапазоном при великих розмірах масиву може бути неефективним за часом навантаження на процесори.

Проблема полягає в тому, що обчислювальне навантаження при обробці кожного елемента залежить від індексу масиву. Обробка початкових елементів масиву займає менший час в порівнянні з обробкою останніх елементів. Розподіл даних по діапазону призводить до незбалансованого завантаження потоків і зниження ефективності розпаралелювання.



Тому для кращого балансування навантаження рекомендується використовувати кругову декомпозицію. В цьому прикладі перший потік обробляє всі парні елементи масиву (0,2,4, і т.д), другий – непарні (1,3,5, і т.д.).

У якості параметра в потоки передаються початкові елементи масиву

```
thrd1.Start(0); //1- парні номери елементів масиву
thrd2.Start(1); //2 - непарні номери елементів масиву
```

Круговий алгоритм реалізовано в методі RunMas. Ми змінили оператор циклу для перебору елементів через 1 і аналіз ознаки потоку.

```
public void RunMas(object obj)
{
    //Круговий алгоритм обчислення суми елементів масиву
    // Зверніть увагу на те, що в цій формі методу RunMas()
    // вказується параметр типа object.
    int min = (int)obj; //номер першого елементу масиву
    count = 0;
    int i = min;
    while (i < Data.Length)
    {
        count += Data[i];
        i += 2;
    }
    if (min == 0)
    {
        Count1 = count;
        Console.WriteLine("Сума чисел, підрахована потоком 1 = " + Count1);
    }
    else
    {
        Count2 = count;
        Console.WriteLine("Сума чисел, підрахована потоком 2 = " + Count2);
    }
}
```

```

        Console.WriteLine("потік завершено ");
    }

```

Повний код програми

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Diagnostics;
namespace Lab3_Circle
{
    //Генерація масиву чисел та їх послідовна обробка
    class MyThread
    {
        int[] Data; // масив чисел
        long count; //сума чисел
        public long Count1 { get; set; } //сума парних чисел масиву
        public long Count2 { get; set; } //сума непарних чисел масиву
        public void GenMas(int n)
        {
            // заповнення масиву числами 1,2,...n.
            Data = new int[n]; // масив чисел
            for (int i = 0; i < Data.Length; i++)
                Data[i] = i + 1;
            Console.WriteLine("Генерація чисел і заповнення масиву");
        }
        public void RunMas(object obj)
        {
            //Круговий алгоритм обчислення суми елементів масиву
            // Зверніть увагу на те, що в цій формі методу RunMas()
            // вказується параметр типу object.
            int min = (int)obj; //номер першого елементу масиву
            count = 0;
            int i = min;
            while (i < Data.Length)
            {
                count += Data[i];
                i += 2;
            }
            if (min == 0)
            {
                Count1 = count;
                Console.WriteLine("Сума чисел, підрахована потоком 1 = " + Count1);
            }
            else
            {
                Count2 = count;
                Console.WriteLine("Сума чисел, підрахована потоком 2 = " + Count2);
            }

            Console.WriteLine("потік завершено ");
        }
    }

    class Program
    {
        static void Main()
        {
            Stopwatch sw = new Stopwatch();
            sw.Start();
            Thread thrd1, thrd2;
            Console.WriteLine("Паралельна версія - круговий алгоритм ");

```

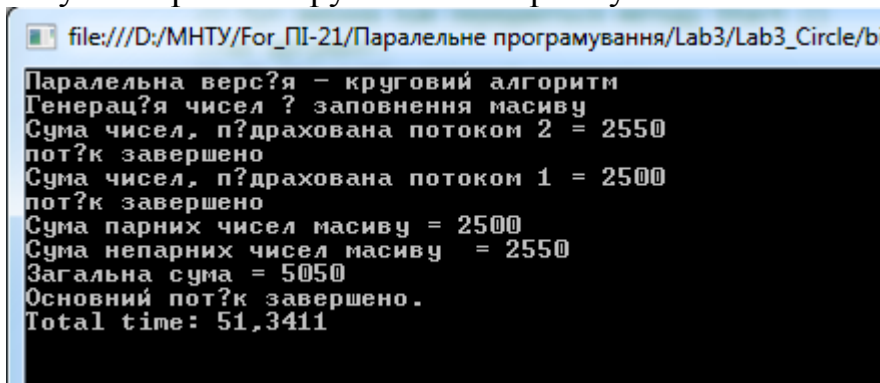
```

MyThread mt = new MyThread(); // об'єкт класу
//заповнення масиву
mt.GenMas(100);
//створення 2 потоків для парал.виконання RunMas
thrd1 = new Thread(mt.RunMas);
thrd2 = new Thread(mt.RunMas);
// Тут змінна num передається методу Start ()
//як аргумент.
thrd1.Start(0); //парні елементи
thrd2.Start(1); //непарні елементи
thrd1.Join();
thrd2.Join();
long totalSumma; //сума елементів масиву
totalSumma = mt.Count1 + mt.Count2;
sw.Stop();
Console.WriteLine("Сума парних чисел масиву = " + mt.Count1);
Console.WriteLine("Сума непарних чисел масиву = " + mt.Count2);
Console.WriteLine("Загальна сума = " + totalSumma);
Console.WriteLine("Основний потік завершено.");
TimeSpan ts = sw.Elapsed;
Console.WriteLine("Total time: {0}", ts.TotalMilliseconds);
Console.ReadKey();
    }
}
}

```

Круговий алгоритм реалізувати простіше і він потребує меншої кількості параметрів.

Результат роботи кругового алгоритму:



```

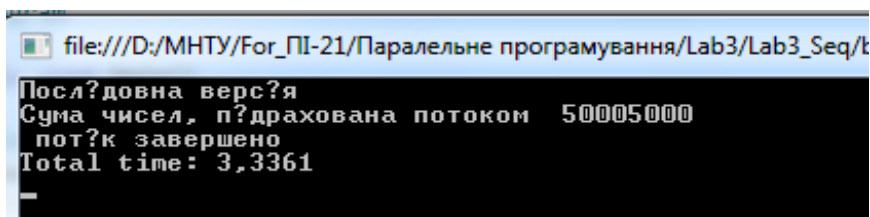
file:///D:/МНТУ/For_ПІ-21/Паралельне програмування/Lab3/Lab3_Circle/b
Паралельна версія - круговий алгоритм
Генерація чисел ? заповнення масиву
Сума чисел, підрахована потоком 2 = 2550
потік завершено
Сума чисел, підрахована потоком 1 = 2500
потік завершено
Сума парних чисел масиву = 2500
Сума непарних чисел масиву = 2550
Загальна сума = 5050
Основний потік завершено.
Total time: 51,3411

```

3. Дослідження часу виконання алгоритмів

Збільшимо розмір масиву Data [10000] і виконаємо для цього масиву всі 3 версії програми.

1. Послідовна версія



```

file:///D:/МНТУ/For_ПІ-21/Паралельне програмування/Lab3/Lab3_Seq/t
Послідовна версія
Сума чисел, підрахована потоком 50005000
потік завершено
Total time: 3,3361

```

2. Паралельна версія. Розбиття масиву на 2 частини


```

file:///D:/МНТУ/For_ПІ-21/Паралельне програмування/Lab3/Lab3/bin/Debug/
Паралельна версія – розбиття масиву на 2 частини
Генерація чисел ? заповнення масиву
Сума чисел, п?драхована першим потоком 12502500
пот?к завершено
Сума чисел, п?драхована другим потоком 37502500
пот?к завершено
Сума першої частини масиву = 12502500
Сума другої частини масиву = 37502500
Загальна сума. = 50005000
Основний пот?к завершено.
Total time: 53,5704

```

3. Круговий алгоритм

```

file:///D:/МНТУ/For_ПІ-21/Паралельне програмування/Lab3/Lab3_Circle/
Паралельна версія – круговий алгоритм
Генерація чисел ? заповнення масиву
Сума чисел, п?драхована потоком 2 = 25005000
пот?к завершено
Сума чисел, п?драхована потоком 1 = 25000000
пот?к завершено
Сума парних чисел масиву = 25000000
Сума непарних чисел масиву = 25005000
Загальна сума = 50005000
Основний пот?к завершено.
Total time: 30,5628

```

Бачимо, що час роботи кругового алгоритму трохи кращий за розбиття на 2 частини. Але послідовна версія все ще працює швидше.

Висновки.

При декомпозиції задач за даними кожна підзадача (потік) виконує одні і ті самі дії, але з різними даними, наприклад, обробку великого масиву (колекції) даних. Для цього необхідно певним чином розділити елементи масиву по окремих потоках. Рекомендується для цього використовувати круговий алгоритм. Переваги паралелізму досягаються при великих об'ємах даних.

Контрольні запитання і завдання для самостійного виконання

1. Як реалізувати паралельну обробку великого масиву даних?
2. Яка ідея лежить в основі кругової декомпозиції для роботи з даними?
3. Чому ефект від розпаралелювання спостерігається тільки при великій кількості елементів?
4. Чому збільшення складності обробки підвищує ефективність багатопотокової обробки?
5. Яка кількість потоків є оптимальною для конкретної обчислювальної системи?
6. Чому нерівномірність завантаження потоків призводить до зниження ефективності багатопотокової обробки?
7. Які варіанти декомпозиції дозволяють збільшити рівномірність завантаження потоків?

8. У якій ситуації кругова декомпозиція не забезпечує рівномірне завантаження потоків?
9. Дослідити роботу версій при збільшенні розміру масиву.