Concurrency

เสนอ

ผศ.วัจนพงศ์ เกษมศิริ

จัดทำโดย

64010462	บุริศ เสรีวัตตนะ
64010850	ศิวกร สุริยะ
64010926	สุพิชญา พรหมปาลิต
64010936	สุรางคนางค์ เกตุยั่งยืนวงศ์
64010994	อรัชฌา ปิ่นประยูร
64011131	ธนกฤต ธรรมภาณพินิจ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 01076011

OPERATING SYSTEMS

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

รายละเอียดอุปกรณ์ที่ใช้ในการทดสอบ

Spec อุปกรณ์ที่ใช้ทดลอง :

CPU : 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz, 2688 Mhz

จำนวน Core : 6

จำนวน Thread : 12

Ram : 32 GB

Execute : release mode

dotNet Version : 5.0

Version เริ่มต้น

คำอธิบาย :

ใน Source code เริ่มต้น เป็นโปรแกรมที่อ่านข้อมูลจากไฟล์ Problem01.dat และนำข้อมูลในรูปแบบของอาร์เรย์ไบต์ (byte array) มาประมวลผลใน Function sum() และแสดงผลรวมและเวลาที่ใช้ในการประมวลผลข้อมูลทั้งหมดในตอนสุดท้ายใน Function sum()

```
static void sum()
{
    if (Data_Global[G_index] % 2 == 0)
    {
        Sum_Global -= Data_Global[G_index];
    }
    else if (Data_Global[G_index] % 3 == 0)
    {
        Sum_Global += (Data_Global[G_index] * 2);
    }
    else if (Data_Global[G_index] % 5 == 0)
    {
        Sum_Global += (Data_Global[G_index] / 2);
    }
    else if (Data_Global[G_index] % 7 == 0)
    {
        Sum_Global += (Data_Global[G_index] / 3);
    }
    Data_Global[G_index] = 0;
    G_index++;
}
```

ใน Function Main()

```
static void Main(string[] args)
{
    Stopwatch sw = new Stopwatch();
    int i, y;

    /* Read data from file */
    Console.Write("Data read...");
    y = ReadData();
    if (y == 0)
    {
        Console.WriteLine("Complete.");
    }
    else
    {
        Console.WriteLine("Read Failed!");
    }

    /* Start */
    Console.Write("\n\nWorking...");
    sw.Start();
    for (i = 0; i < 1000000000; i++)
        sum();
    sw.Stop();
    Console.WriteLine("Done.");

    /* Result */
    Console.WriteLine("Summation result: {0}", Sum_Global);
    Console.WriteLine("Time used: " + sw.ElapsedMilliseconds.ToString() + "ms");
}
```

ผลลัพธ์ :

```
Data read...Complete.


Working...Done.
Summation result: 888701676
Time used: 11316ms
```

Summation result : 888701676

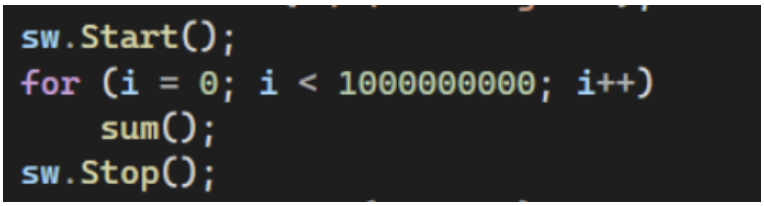
ใช้เวลา : 11.316 วินาที

ปัญหาที่พบ :

จากผลลัพธ์จะพบว่าโปรแกรมนี้ใช้เวลาทำงานมากพอสมควร

ข้อสันนิษฐาน :

ใน Function Main() มีการใช้ Loop เรียก Function sum() อยู่ 1 พันล้านรอบ ทำให้มีการ pop และ push ของ instruction address ใน stack เป็นจำนวนมาก อาจส่งผลให้โปรแกรมทำงานใช้เวลามากขึ้นเล็กน้อย



Version 1

คำอธิบาย :

แก้ไขจาก Version 0 โดยเปลี่ยนจากการใช้ Loop ใน Function Main() เป็นการใช้ Loop ใน Function sum()

```
static void sum()
{
    for (int i = 0; i < 1000000000; i++)
    {
        if (Data_Global[G_index] % 2 == 0)
        {
            Sum_Global -= Data_Global[G_index];
        }
        else if (Data_Global[G_index] % 3 == 0)
        {
            Sum_Global += (Data_Global[G_index] * 2);
        }
        else if (Data_Global[G_index] % 5 == 0)
        {
            Sum_Global += (Data_Global[G_index] / 2);
        }
        else if (Data_Global[G_index] % 7 == 0)
        {
            Sum_Global += (Data_Global[G_index] / 3);
        }
        Data_Global[G_index] = 0;
        G_index++;
    }
}
```

```
static void Main(string[] args)
{
    Stopwatch sw = new Stopwatch();
    int i, y;

    /* Read data from file */
    Console.Write("Data read...");
    y = ReadData();
    if (y == 0)
    {
        Console.WriteLine("Complete.");
    }
    else
    {
        Console.WriteLine("Read Failed!");
    }

    /* Start */
    Console.Write("\n\nWorking...");
    sw.Start();
    //for (i = 0; i < 1000000000; i++)
        sum();
    sw.Stop();
    Console.WriteLine("Done.");

    /* Result */
    Console.WriteLine("Summation result: {0}", Sum_Global);
    Console.WriteLine("Time used: " + sw.ElapsedMilliseconds.ToString() + "ms");
}
```

ผลลัพธ์ :

```
Data read...Complete.


Working...Done.
Summation result: 888701676
Time used: 10645ms
```

Summation result : 888701676 ถูกต้อง

ใช้เวลา : 10.645 วินาที

ปัญหาที่พบ :

โปรแกรมนี้ใช้เวลาทำงานลดลงไปเพียงเล็กน้อย แค่ระดับ 0.1 วินาที โปรแกรมยังคงทำงานได้ไม่เร็วเท่าที่ควร

ข้อสันนิษฐาน :

โปรแกรมนี้มีการทำงานแบบ Single Thread หมายความว่าใช้เพียง Thread เดียวในการงานวน Loop จำนวน 1 พันล้านรอบจึงยังทำใช้เวลาทำงานค่อนข้างมากอยู่

Version 2

คำอธิบาย :

แก้ไขจาก Version 1 โดยมีการเพิ่ม Thread มาช่วยในการทำงานอีก 1 Thread

ขั้นตอน

1. ลบ Function sum() ทิ้ง แล้วสร้าง sum1 sum2 โดยให้แบ่งครึ่งช่วงในการคำนวณ
2. เปลี่ยน index ของ Data_Global ให้ใช้ค่า i จาก loop ในการชี้ index แทน ค่า G_index ที่เป็นตัวแปร เพราะเนื่องจากมีการใช้ G_index++ จากทั้ง 2 Function อาจทำให้ค่า index ของ Data_Global ไม่ถูกต้อง

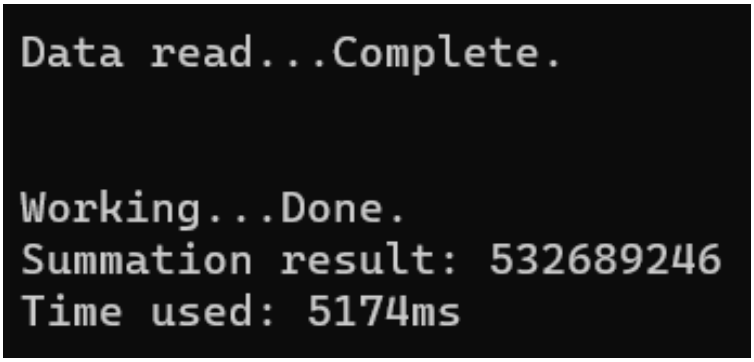
```
static void sum1()
{
    for (int i = 0; i < 500000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            Sum_Global -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            Sum_Global += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            Sum_Global += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            Sum_Global += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }
}
```

```
static void sum2()
{
    for (int i = 500000000; i < 1000000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            Sum_Global -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            Sum_Global += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            Sum_Global += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            Sum_Global += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }
}
```

3. ใน Main() สร้าง Thread มา 2 Thread ให้ทำงานพร้อมกัน และกำหนด Function ขึ้นมาใหม่สำหรับการทำงานของทั้ง 2 threads คือ function sum1() โดยมีเงื่อนไข คือ for (int i = 0; i < 500000000; i++) สำหรับการคำนวณช่วงครึ่งแรก และ sum2() โดยมีเงื่อนไข คือ for (int i = 500000000; i < 1000000000; i++) สำหรับการคำนวณช่วงครึ่งหลัง

```
static void Main(string[] args)
{
    Stopwatch sw = new Stopwatch();
    int i, y;

    /* Read data from file */
    Console.Write("Data read...");
    y = ReadData();
    if (y == 0)
    {
        Console.WriteLine("Complete.");
    }
    else
    {
        Console.WriteLine("Read Failed!");
    }

    /* Start */
    Console.Write("\n\nWorking...");

    sw.Start();
    //for (i = 0; i < 1000000000; i++)

    Thread th1 = new Thread(sum1);
    Thread th2 = new Thread(sum2);

    th1.Start();
    th2.Start();

    th1.Join();
    th2.Join();

    sw.Stop();
    Console.WriteLine("Done.");

    /* Result */
    Console.WriteLine("Summation result: {0}", Sum_Global);
    Console.WriteLine("Time used: " + sw.ElapsedMilliseconds.ToString() + "ms");
}
```

ผลลัพธ์ :



Summation result : 532689246 ไม่ถูกต้อง

ใช้เวลา : 5.174 วินาที

ปัญหาที่พบ :

ค่า Sum_Gobal น้อยกว่าที่ควรจะเป็น

ข้อสันนิษฐาน :

โปรแกรมนี้มี Thread มากกว่า 1 ทำงานพร้อม ๆ กัน ดังนั้นค่า Sum_Global จึงอาจจะมีการเข้าถึงหรืออัพเดตค่ามากกว่า 1 thread ทำให้เมื่อถูกอัพเดตค่าพร้อมกันจากหลาย ๆ thread ทำให้ค่า Sum_Global ผิด

Version 3

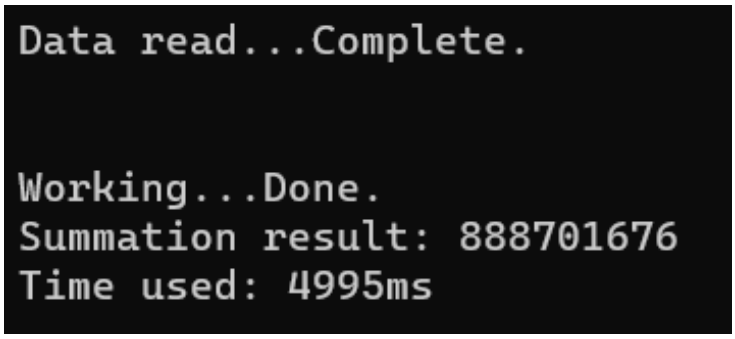
คำอธิบาย :

1. เนื่องจากค่า Sum_Global ที่เป็น ตัวแปรชนิด Global รวมถึงผลจากการที่ทำงานแบบ Multi Thread ทำให้ Function sum1 และ sum2 คำนวณผลลัพธ์ออกมาได้ไม่ถูกต้อง
2. จึงหลีกเลี่ยงการใช้ตัวแปร Sum_Global เปลี่ยนมาเป็นการใช้ตัวแปร Sum_local แทนใน Function sum1 และ sum2 ซึ่งเป็น ตัวแปร Local เพื่อให้แต่ละ Thread ใช้งานตัวแปร Sum_local ของตัวเองในการรวบรวมผลลัพธ์สำหรับช่วงการคำนวณใน Data_Global ในช่วงของตัวเองที่กำหนดไว้
3. เมื่อรวบรวมค่าผลลัพธ์ดังกล่าวเสร็จ จึงค่อยนำไปรวมกันกับตัวแปร Sum_Global ในขั้นตอนสุดท้ายของ Function

```
static void sum1()
{
    long sum_local = 0;
    for (int i = 0; i < 500000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            sum_local -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            sum_local += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            sum_local += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            sum_local += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }

    Sum_Global += sum_local;
}
```

```
static void sum2()
{
    long sum_local = 0;
    for (int i = 500000000; i < 1000000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            sum_local -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            sum_local += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            sum_local += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            sum_local += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }
    Sum_Global += sum_local;
}
```

ผลลัพธ์:



Summation result : 888701676 ถูกต้อง

ใช้เวลา : 4.995 วินาที

ปัญหาที่พบ :

ผลลัพธ์ของโปรแกรมถูกต้องแต่ความเร็วดีขึ้นเพียงเล็กน้อย จึงอยากแก้ไขโปรแกรมให้คำนวณได้เร็วขึ้นกว่าเดิม

ข้อสันนิษฐาน :

จากการทดลองแก้ไขโปรแกรมทำให้เราได้ทราบว่า จำนวน Thread มีผลต่อเวลาในการคำนวณ ดังนั้น จึงสันนิษฐานได้ว่า หากจำนวน Thread มีมากกว่า 2 จะทำให้โปรแกรมสามารถประมวลผลได้เร็วกว่าเดิม

Version 4

คำอธิบาย :

จาก Version 3 นั้นพบว่า การใช้งานแบบ Multi Thread จำนวน 2 Thread ให้ช่วยกันทำงาน ทำให้โปรแกรมสามารถทำงานได้เร็วมากขึ้น

จึงต้องการแก้ไขให้สามารถใช้จำนวน Thread ได้มากที่สุดเท่าที่อุปกรณ์ที่ใช้ Run โปรแกรมจะมี โดยใช้

1. Environment.ProcessorCount คือ คำสั่งในการเข้าถึงจำนวน thread ที่คอมพิวเตอร์เครื่องนั้น ๆ มี
2. มีการเพิ่มการคำนวณจำนวน thread ใหม่ เนื่องจากอาจเกิดปัญหา Thread แบ่งช่วงการคำนวณแล้วหารไม่ลงตัว จึงใช้สมการนี้ในการแบ่ง

PREFERED_DATA_SIZE = MAX_DATA_SIZE + number_of_thread - (MAX_DATA_SIZE % number_of_thread);

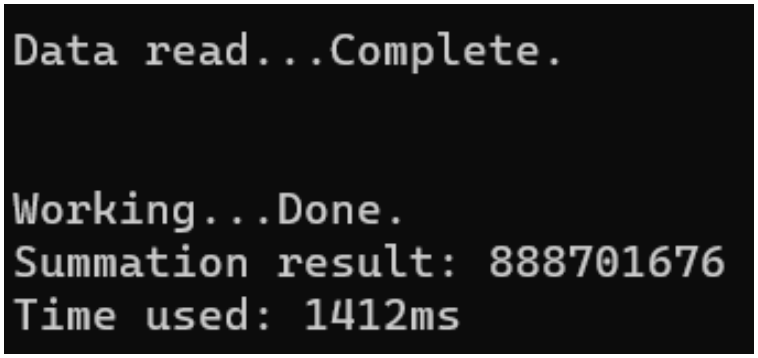
```
static int ReadData()
{
    int returnData = 0;
    FileStream fs = new FileStream("C:\\Users\\jj\\source\\repos\\OS_LAB\\OS_LAB\\Problem01.dat", FileMode.Open);
    BinaryFormatter bf = new BinaryFormatter();

    number_of_thread = Environment.ProcessorCount;
    PREFERED_DATA_SIZE = MAX_DATA_SIZE + number_of_thread - (MAX_DATA_SIZE % number_of_thread);

    try
    {
        Data_Global = (byte[])bf.Deserialize(fs);
    }
    catch (SerializationException se)
    {
        Console.WriteLine("Read Failed:" + se.Message);
        returnData = 1;
    }
    finally
    {
        fs.Close();
    }

    return returnData;
}
```

```
static void sum(int start, int end)
{
    int Sum_Local = 0;
    for (int i = start; i < end; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            Sum_Local -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            Sum_Local += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            Sum_Local += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            Sum_Local += (Data_Global[i] / 3);
        }
    }
    Sum_Global += Sum_Local;
}
```

3. ใช้ for loop ในการกำหนดช่วงการทำงานของแต่ละ Thread ผ่าน function sum โดยมีการระบุพารามิเตอร์ start และ end ซึ่งใช้เป็นตัวแปรในการกำหนดช่วงของ index เริ่มต้นและสิ้นสุดในการทำงานของแต่ละ Thread ตามลำดับ

```
static void Main(string[] args)
{
    Stopwatch sw = new Stopwatch();
    int i, y;

    /* Read data from file */
    Console.Write("Data read...");
    y = ReadData();
    if (y == 0)
    {
        Console.WriteLine("Complete.");
    }
    else
    {
        Console.WriteLine("Read Failed!");
    }

    /* Start */
    Console.Write("\n\nWorking...");
    sw.Start();

    // สร้าง Thread ตามจำนวน Thread ที่อุปกรณ์มี
    Thread[] th = new Thread[number_of_thread];
    int dataSizePerThread = PREFERED_DATA_SIZE / number_of_thread;

    for (i = 0; i < th.Length; i++)
    {
        int start = i * dataSizePerThread;
        int end;
        if (i != th.Length - 1)
        {
            end = ((i + 1) * dataSizePerThread);
        }
        else {
            end = MAX_DATA_SIZE;
        }
        // ทำให้ Thread ทำงานในบริบทของฟังก์ชัน โดยใส่ parameter บอกจุดเริ่มและจุดจบ ของช่วง Data ของแต่ละ Thread
        th[i] = new Thread(() => sum(start, end));
    }
    // ให้ทุก Thread เริ่มทำงาน
    for (i = 0; i < th.Length; i++)
        th[i].Start();

    for (i = 0; i < th.Length; i++)
        th[i].Join();

    sw.Stop();
    Console.WriteLine("Done.");

    /* Result */
    Console.WriteLine("Summation result: {0}", Sum_Global);
    Console.WriteLine("Time used: " + sw.ElapsedMilliseconds.ToString() + "ms");
}
```

ผลลัพธ์ :



Summation result : 888701676 ถูกต้อง

ใช้เวลา : 1.412 วินาที

ตารางเปรียบเทียบ

Version	0	1	2	3	4
Summation result	888701676	888701676	532689246	888701676	888701676
Time used (s)	11.316	10.645	5.174	4.995	1.412
Thread used (s)	1	1	2	2	ตามจำนวน thread ในเครื่อง