Пресмятане на числото Pi

Проект по Разпределени софтуерни архитектури

Изготвил: Иван Младенов, ф.н 61950

Проверил:…………………..................

/ас. Христо Христов/

1

Съдържание

[Условие на задачата](#page3) [3](#page3)

Описание на алгоритъма [3](#page3)

[Архитектура на приложението и реализация на избрания алгоритъм](#page4) [3](#page4)

[Проведени тестове и измервания](#page7) [4](#page7)

2

Условие на задачата

Числото (стойността на) Pi може да бъде изчислено по различни начини. Използвайки сходящи редове, можем да сметнем стойността на Pi с произволно висока точност. Един от бързо сходящите към Pi редове е този, открит от индийския математик Srinivasa Ramanujan през 1910-1914 година. За стойността на Pi имаме.



Вашата задача е да напишете програма за изчисление на числото Pi изпозлвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на Pi със зададена от потребителя точност.

* Програмата трябва да използва паралелни процеси (нишки) за да разпредели работата по търсенето на елементите от редицата
* Програмата трябва да получава параметър, който задава точността на пресмятанията (точността се изразява в брой членове на реда). Например **“-p 10240”**
* Команден параметър, който задава максималния брой нишки, на които да се разделя работата по пресмятането. Например **“-t 1”** или **“-tasks 4”.**
* Опционален параметър, който определя името на изходен файл, в който да се запише резултата от работата на програмата (стойността на Pi). Например **“-o output.txt”**. Ако параметъра не е подаден записва във файл по подразбиране “result.txt”.
* Опционален параметър за “чист” режим на работа. Например **“-q”**. Ако този параметър е подаден, програмата извежда само времето за изчисление на Pi.
* Програмата трябва да извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление.

Описание на алгоритъма

За целта, програмата ще пресметне сумата от елементите на редицата на Ramanujan с избраната точност (брой елементи). Всеки елемент ще бъде пресметнат на отделна нишка с цел паралелизиране на работата и използването на цялата мощност на машината (в рамките на подаденият параметър за максимален брой нишки).

Програмата е написана на езика C# в средата .NET Core 2.1, която е мултиплатформена и позволява изпълнение под Linux операционна система.

Архитектура на приложението и реализация на избрания алгоритъм

Изваден е отделен клас, който представлява параметрите, подадени на програмата:

public class Parameters

{

public const string DefaultOutputFile = "result.txt";

// -p

public int ElementsCount { get; set; }

// -t or -tasks

public int MaxTasks { get; set; }

// -q

public bool IsQuiet { get; set; }

// -o

public string OutputFile { get; set; }

}

3

След началото на програмата, се създава инстанция от този клас, в която се попълват стойностите на подадените параметри.

За начало имаме акумулаторна променлива **Sum,** която е инициализирана със стойност 0 и ще представлява сборът от елементите на редицата.

За самите изчисления се използва метода ***For*** на вградения клас ***Parallel***. Параметрите на този метод са както следва:

* Начална стойност (в нашият случай редицата започва от 0)
* Крайна стойност (подаденият параметър **”–p“)**
* Опции, които включват пропърти за максимално ниво на паралелизъм (максимален брой нишки, които да се използват. Параметъра **“-t”**
* Функция, която да се изпълнява за всяка стойност (в случая се извиква метода **Calculate**, на който се подава текущият **n** от редицата и резултата се добавя към стойността на акумулатора **Sum**)

Метода Calculate получава като параметър номера на елемента от редицата и пресмята стойността на самия елемент.

След намирането на резултата от сумата на редицата, за получаването на стойността на Pi трябва да се извърши следната операция

Получената стойност бива записана във файла, подаден като параметър (или файла по подразбиране, ако липсва параметър) чрез метода **WriteToFile**, който използва вграденият клас **System.IO.File**

За измерването на времето се използва вграденият клас **Stopwatch** и на екрана излиза броя милисекунди, необходими за извършването на операцията. Измерването започва след прочитането и обработването на входните параметри.

4

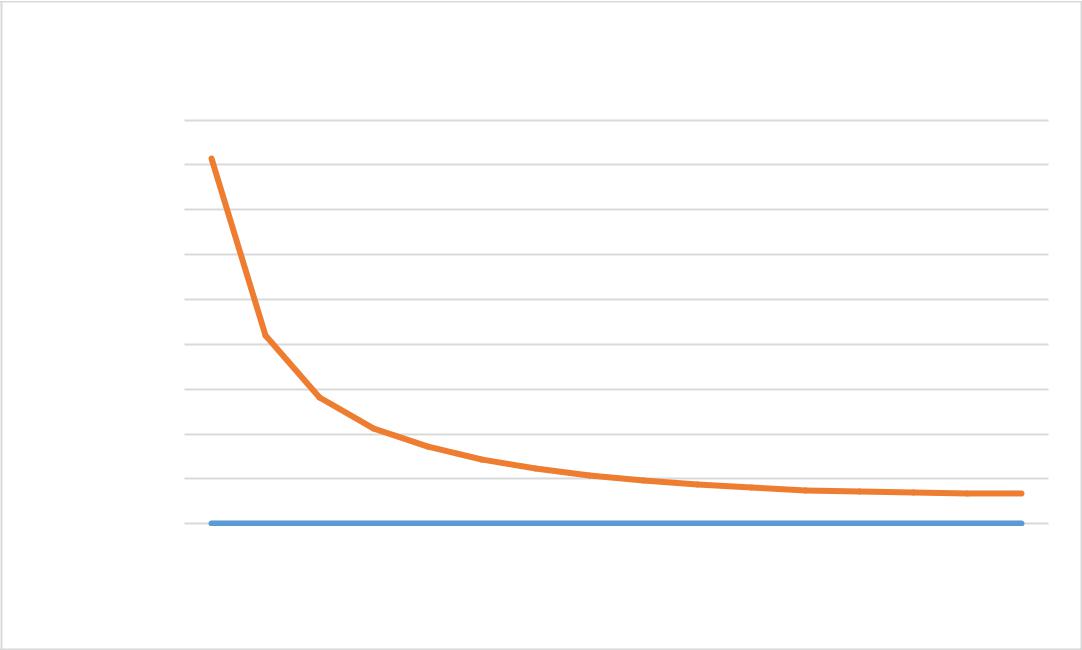
Проведени тестове и измервания

На предоставена машина за извършване на тестовете са проведени тестове за намиране на стойността на Pi с точност 50000 елемента от редицата. В таблицата са представени данните получени при извършване на тестовете:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Брой*** |  |  | ***Ефективност*** |  |
| ***процесорни*** | ***Време в ms*** | ***Ускорение*** |  |
| ***(ефикасност)*** |  |
| ***ядра*** |  |  |  |
|  |  |  |  |
| **1** |  | 1 | 1 |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
| **8** |  |  |  |  |
| **9** |  |  |  |  |
| **10** |  |  |  |  |
| **11** |  |  |  |  |
| **12** |  |  |  |  |
| **13** |  |  |  |  |
| **14** |  |  |  |  |
| **15** |  |  |  |  |
| **16** |  |  |  |  |

Данните в горната таблица са представени графично, посредством следните диаграми:

* На диаграмата е показано времето, необходимо за намиране на стойността на Pi с точност 50000 елемента от редицата, при употребата на различен брой процесорни ядра, започвайки от 1 ядро (серийната версия на програмата) до всичките 16 ядра на предоставената машина:



Време за определяне на минималната описана, около множество от 4000 точки

окръжност

|  |
| --- |
| време в ms |

900000

800000

700000

600000

500000

400000

300000

200000

100000

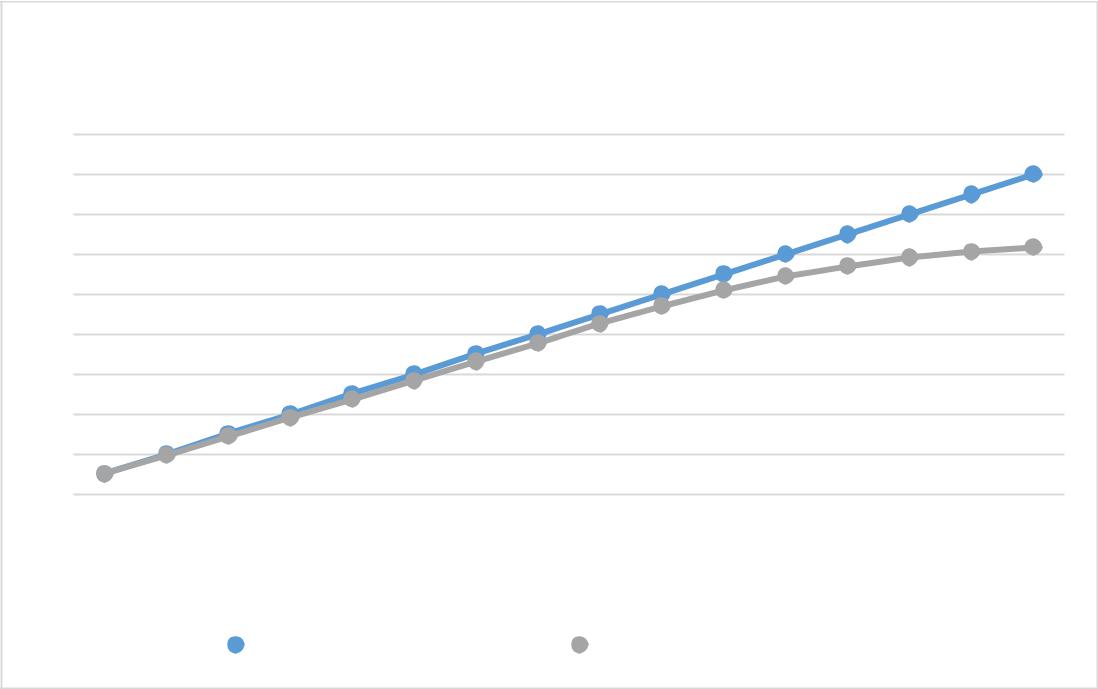
0

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

брой процесорни ядра

5

* На диаграмата е показано постигнатото ускорение при намиране на стойността на Pi с точност 50000 елемента от редицата, при употребата на различен брой процесорни ядра, започвайки от 1 ядро (серийната версия на програмата) до всичките 16 ядра на предоставената машина:



18

16

14

12

10

8

6

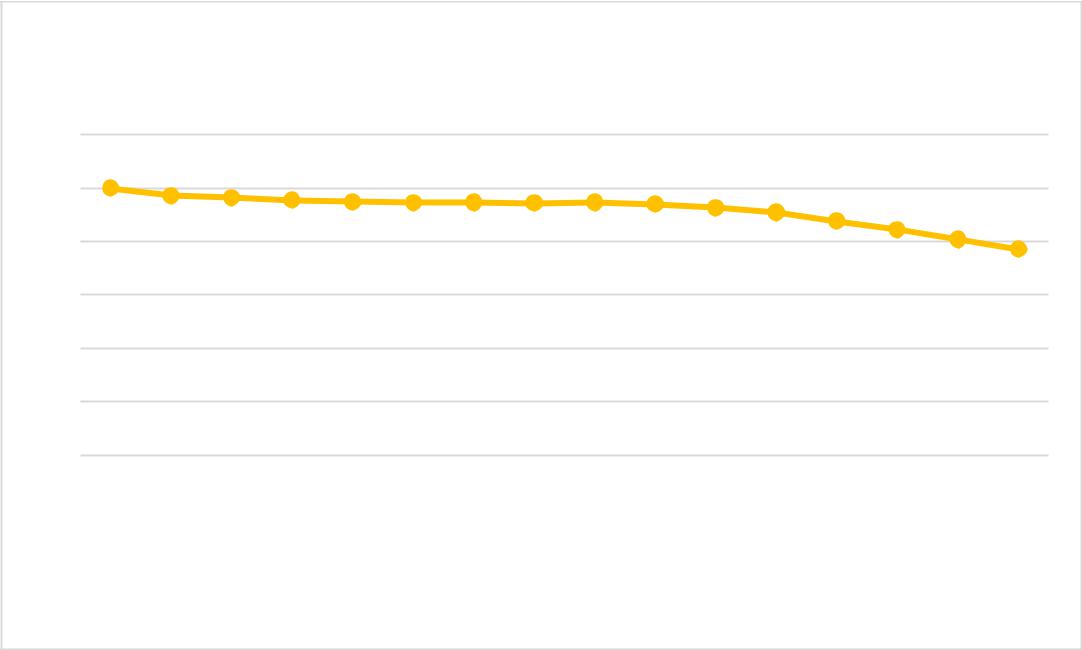
4

2

0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |  |
|  |  |  |  |  |  | брой процесорни ядра | | | |  |  |  |  |  |  |  |
|  |  |  | Идеално ускорение | | | |  |  | Действително ускорение | | | | |  |  |  |
|  |  |  |  |  |  |  |  |

* На диаграмата е показана постигнатата ефективност (ефикасност) при намиране на стойността на Pi с точност 50000 елемента от редицата, при употребата на различен брой процесорни ядра, започвайки от 1 ядро (серийната версия на програмата) до всичките 16 ядра на предоставената машина:



Ефективност (ефикасност) при определяне на минималната

описана, около множество от 4000 точки окръжност

1,2

1

0,8

0,6

0,4

0,2

0

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

брой процесорни ядра

 Ефективност (ефикасност)

6