**Паралелно пресмятане на неперовото число е**

Изготвил: Кристина Попова, факултетен номер: 81933, специалност: Компютърни науки, 3 курс, 6-та група Ръководител: проф. д-р Васил Цунижев, ас. Христо Христов

Съдържание:

1.Увод**1**

1.1. Какво представлява Ойлеровото число?2

1.2. Цел на проекта?3

1.3. Функционалности..............................................................................................................................

1.4. Решения на задачата4

1.4.1. Първо решение.....................................................................................................................

1.4.2. Второ решение......................................................................................................................

1.4.3. Трето решение.......................................................................................................................

1.4.4. Сравнение на решенията......................................................................................................

2.Проектиране и реализация5

2.1.Елементи на програмата...................................................................................................................

2.2.Описание на архитектурата на решението.....................................................................................

2.3.Тестов план......................................................................................................................................

2.3.1 Тестов план на параметрите...............................................................................................

2.3.2. Архитектура на машината...................................................................................................

3. Тестови резултати...................................................................................................................................

3.1. Таблици на измерванията...............................................................................................................

3.2.Графики на ускорението...................................................................................................................

3.3.Графики на ефективността.6

3.4.Прегред на резултатите....................................................................................................................

4.Източници................................................................................................................................................

1. Увод
   1. Какво представлява Ойлеровото число ***е***?

Едно важно за математика число е Ойлеровото число ***е***. Използвайки сходящи редове, можем да сметнем стойността на ***е*** с произволно висока точност. Един от сравнително бързо сходящите към ***е*** редове е:

* 1. Цел на проекта

Нашата цел е да пресметнем Неперовото число с голяма точност, използвайки цитираният сходящ ред, и да изследваме получилото се ускорение. Не е възможно да пресметнем всички цифри на Ойлеровото число, поради което се осигурява възможността за пресмятането на ***е*** със зададена от потребителя точност. Този брой ще е голям, затова проектът използва многонишково програмиране, за да може да се получи ускорение при пресмятането.

* 1. Функционалности

Решенията, които ще разгледаме, реализират следните функционалности, зададени с параметри:

-p <precision> – указва броя на членовете на реда

-t <numberOfThreads> – указва броя на нишките

-o <outputFile> - стойността на числото е се записва

* 1. Решения на задачата

Задачата за пресмятане на числото ***е*** може да се реши по няколко начина. Нека ги разгледаме.

* + 1. Първо решение

Първото решение, което ще разгледаме, е според източник [1]. То разделя броя на членовете n, на сходящия ред, на броя нишки p. Така всяка нишка ще трябва да пресметне сумата на n/p на брой члена. Всяка нишка пресмята по един член през p на брой члена. След това събираме отделните суми, сметнати от всяка нишка, и получаваме приближение на числото е.

За да вникнем в идеята на решение 1, нека разгледаме случая, когато n=10 и p=2. При тези параметри първата нишка ще пресметне всички нечетни членове на реда, а втората нишка ще пресметне всички четни членове.

Недостатък на решение 1 е, че всяка нишка трябва да пресмята самостоятелно факториела, който се намира в знаменателя й, което за много голям факториел би довело до значително забавяне на програмата.

* + 1. Второ решение

Второто решение, което ще разгледаме, е според източник [2]. Идеята е да разделим броя на членовете n, на реда, на броя на нишките p. Така всяка нишка ще пресметне n/p на брой последователни члена от сходящия ред. Тоест разделяме реда на p на брой парциални суми. Накрая сумираме получените резултати от всяка нишка, за да получим приближение на числото е.

Нека разгледаме един прост пример, в който n=9, а p=2. В този пример първата нишка ще пресметне първите 5 последователни члена от реда, докато втората нишка ще пресметне 9%4 на брой члена или последните 4 члена от реда.

Тук, отново, недостатък е, че нямаме преизползване на вече пресметнатите факториели, което забавя програмата.

* + 1. Трето решение

Третото решение, което ще разгледаме оптимизира решение 1 и решение 2, като за пресмятането на факториела използваме масив, който пази вече изчислените факториели (източник [3]). По този начин, вместо всеки път наново да изчисляваме факториела, ние преизползваме изчислените минали стойности. Така значително се увеличава скоростта на изчисление.

* + 1. Сравнение на решенията

|  |  |  |
| --- | --- | --- |
| Образец | Гранулярност | Преизползване на факториела |
| Решение 1 според източник [1] | финна | не |
| Решение 2 според източник [2] | едра | не |
| Решение 3 – финално решение, оптимизиращо решиния 1 и 2, според източвик [3] |  | да |

1. Проектиране на решението
   1. Елементи на програмата

Програмата е реализирана на езикът Java. Тъй като работим с големи числа използваме BigDecimal и BigInteger класовете в Java Math.

Поради това, че искаме да пресмятаме Неперовото число с различна точност и с различен брой нишки, програмата поддържа следните параметри:

* 1. Описание на архитектурата на решението

За решението на задачата използваме многонишково програмиране. Всяка нишка идвършва аналогична работа – пресмята парциалната си сума. Разпределението е статично циклично, защото спрямо подадените параметри n и p, всяка нишка пресмята n/p на брой члена.

* 1. Тестов план
     1. Тестов план на параметрите
     2. Архитектура на машината

Машините, на които беше тествана програмата, имат следните характеристики:

|  |  |  |
| --- | --- | --- |
|  | Машина 1 | Машина 2 |
| CPU | Intel® Core™ i5-7200U CPU @ 2.50GHz 2.70GHz |  |
| Sockets |  |  |
| Cores | 2 |  |
| L1 cache | 3MB |  |
| L2 cache |  |  |

1. Тестови резултати
2. Източници
3. Multi-threaded program example <https://www.codeproject.com/Questions/1216899/Multi-threaded-program-example>
4. Geeks for geeks – Sum array using pthreads

<https://www.geeksforgeeks.org/sum-array-using-pthreads/>

1. Stack overflow – Calculating factorial using dynamic programming <https://stackoverflow.com/questions/26989075/approaching-dynamic-programming>