СУ „Св. Климент Охридски“

Факултет по математика и информатика

Курсов проект

Тема 13: Пресмятане на π – Chudnovsky

по

Разпределени софтуерни архитектури

Летен семестър, 2018/2019 год.

Изготвил:

ФН: 61999,

Силвия Тодорова Иванова

Проверил:

……

/ас. Христо Христов/

Дата: 14.06.2019

юни, 2019

СЪДЪРЖАНИЕ

Пресмятане на **π** – Chudnovsky

1. Въведение

1.1. Условие на задачата

Числото (стойността на) **π** може да бъде изчислено по различни начини. Използвайки сходящи редове, можем да пресметнем стойността на **π** с произволно висока точност. Един от бързо сходящите към **π** редове е този, открит от индийския математик Srinivasa Ramanujan през 1910-1914 година. Пред 1987 братята Chudnovsky откриват ред с още по голяма сходимост.

Редът има вида:

$$\frac{1}{\pi} = 12 \sum_{k=0}^{\infty} \frac{(-1)^k (6k)!(13591409 + 545140134k)}{(3k)!(k!)^3 640320^{3k+3/2}}$$

Фиг. 1

Вашата задача е да напишете програма за изчисление на числото **π** използвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на **π** със зададена от потребителя точност. Програмата също така трябва да извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление на стойността на **π**. Резултатите от пресмятането трябва да се намират във посочения от потребителя файл.

Параметри:

-p - точност на пресмятанията, задава се в брой цифри след десетичната запетая (пример: -p 10240)

-t - максималния брой нишки на които разделяме работата по пресмятането на **π** (пример: –t 1)

-o - име на изходен файл (пример: -o result.txt)

ЗАБЕЛЕЖКА: Параметърът не е задължителен, ако не е подаден, то се избира име по подразбиране определено от програмата.

-q - програмата се стартира в „quiet“ режим на работа, при който се извежда само времето отделено за изчисление на **π**.

ЗАБЕЛЕЖКА: Параметърът не е задължителен.

1.2. Алгоритъм за пресмятане на числото **π**

Съществуват различни алгоритми за пресмятане на стойността на числото **π**, с цел намаляване на времето за изчисление, тъй като стойностите, с които работим са доста големи. Обикновено се извежда константата пред сумата и се изчислява всеки член на реда по отделно. Накрая се събират резултатите от частните сумите и се умножават по константата. Това може да бъде приложено тъй като всеки член на реда е самостоятелен. Този алгоритъм е подходящ за паралелни процеси, но е сравнително бавен. Друг алгоритъм е така наречия “Chudnovsky binary splitting”. Двоичното разделяне е техника с общо предназначение за ускоряване на този вид изчисления. Това, което прави алгоритъмът, е да раздели сумата на отделните части, които са по-лесни за пресмятане.

$$\begin{aligned}
\frac{1}{\pi} &= 12\sum_{k=0}^{\infty}\frac{(-1)^k(6k)!(13591409+545140134k)}{(3k)!(k!)^3 640320^{3k+3/2}} \\
&= \frac{12}{640320\sqrt{640320}}\sum_{k=0}^{\infty}\frac{(-1)^k(6k)!(13591409+545140134k)}{(3k)!(k!)^3 640320^{3k}} \\
&= \frac{1}{426880\sqrt{10005}}\sum_{k=0}^{\infty}\frac{(-1)^k(6k)!(13591409+545140134k)}{(3k)!(k!)^3 640320^{3k}}
\end{aligned}$$

Фиг.1 Извеждане на константата пред сумата

Броя на итерациите се получават на база броя на цифрите на числото **π**.

$$\text{брой итерации} = \frac{\text{брой цифри} \,.\, ln(10)}{ln(1519313730560000)}$$

Паралелното пресмятане на **π** се получава като всяка нишка пресмята определен интервал от сумата. Така работата по пресмятането на числото е разделена на броя на нишките. След което събираме получените резултати и умножаване по константата. Евентуално може да се направи оптимизация, при която отново имаме разделяне на интервала на броя на използваните нишки, но интервалите не са равни помежду си. Всеки следващ е приблизително наполовина от предишния.

$$S(a,b) = \frac{T(a,b)}{B(a,b).Q(a,b)}$$

Пълното описание на използвания алгоритъма може да бъде открито в източник под номер [1].

1.3. Обзор на източниците

[1] Pi - Chudnovsky, craig-wood,
https://www.craig-wood.com/nick/articles/pi-chudnovsky/

[2] Parallel Algorithm Models, Parallelcomp,
http://parallelcomp.uw.hu/ch03lev1sec6.html

[3] Executor Service in Java, HowToDoInJava,
https://howtodoinjava.com/java/multi-threading/executor-service-example/

Посочените уеб сайтове са в основата на разработката на проекта. В [1] е описана не само формулата, по която се пресмята стойността на числото, а също и различни имплементации на метода на езика Python. Източници под номер [2, 3] спомагат за определяне на модела на декомпозиция и реализирането на многонишковото изпълнение на програмата.

1.4. Технологии

Приложението е разработено на Java SE 8, средата е Intellij,. Използваните външни библиотеки са Apfloat[5] , пресмята числа с голяма точност, и commons-lang3-3.9, съдържа класа Pair, който ни позволява да пазим два свързани елемента в един обект.

2. Проектиране

2.1. Архитектура на приложението

Входната точка на програмата се намира в класа App. Единствената задача, която изпълнява той е да стартира програмата чрез създаване на инстанция на класа Starter и да изпълни публично достъпния метод start. Starter класът се грижи за парсването на параметрите и стартиране на Worker-а, където реално се случва изпълнението на алгоритъма, измерването на времето за пресмятане, както и писането във файл. Изчислението на числото **π** заедно с цялата математика се осъществят в класа ChudnovskyAlgorithm, метод calculatePi.

2.2. Модел на паралелизма и декомпозиция на данните

Реализиран е паралелизъм по данни SPMD, като нишките работят асинхронно, независимо една от друга. Използваната архитектура е Master-Slaves. Класът ChudnovskyAlgorithm и по-конкретно метода calculatePi играе ролята на Master. При всяко извикване на програмата се определя броя на нишките и броя на цифрите на числото **π**. Спрямо тези две стойности се извършва разпределението на подзадачите между различните нишки. Всяка една задача се състои в пресмятане на определен интервал от сумата, като всеки интервал е с приблизително равна големина[4]. Реализирането на този подход в Java се осъществява чрез класа ExecutorService и негови методи[3].

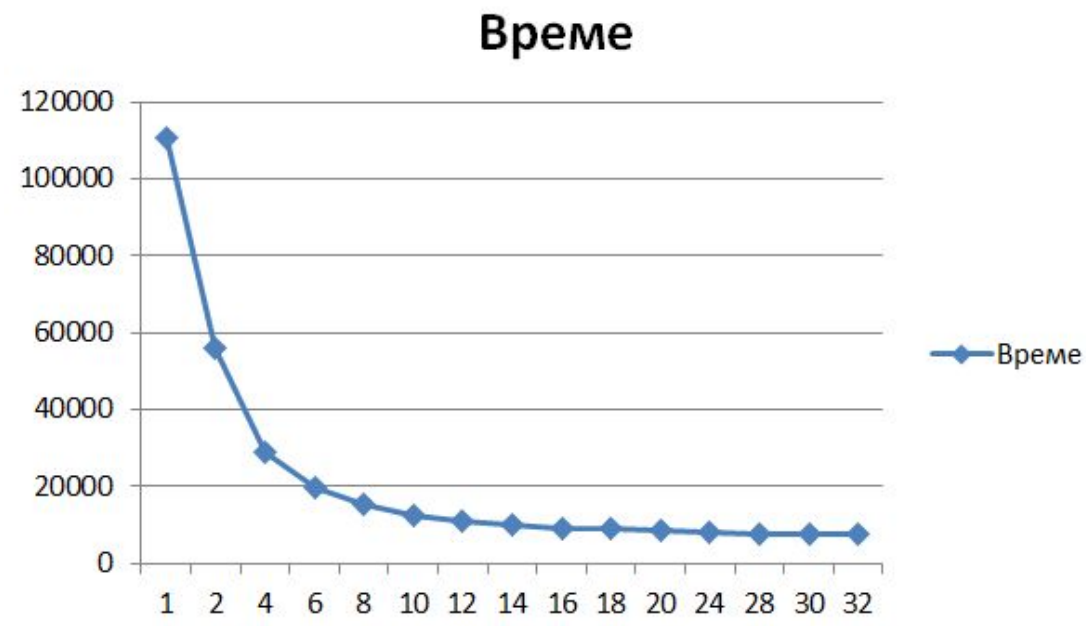
3. Проведени тестове и измервания

На предоставена машина за извършване на тестовете са проведени тестове за намиране на стойността на числото **π** с точност 80 000 знака на числото. Стойността се записва във файл под име pi.txt или друго, зададено от потребителя.

На следващата таблица са показани времето, ускорението и ефективността при различен брой нишки.

Нишки	Време (ms)	Ускорение	Ефективност
1	110712	1	1
2	56127	1,972526591	0,986263296
4	28856	3,836706404	0,959176601
6	19641	5,636780205	0,939463367
8	15427	7,176508718	0,89706359
10	12546	8,824485892	0,882448589
12	10769	10,2806203	0,856718358
14	9863	11,22498226	0,801784447
16	9106	12,15813749	0,759883593
18	8833	12,53390694	0,696328163
20	8602	12,87049523	0,643524762
24	7998	13,84246062	0,576769192
28	7431	14,89866774	0,532095277
30	7800	14,19384615	0,473128205
32	7696	14,38565489	0,449551715

Таблица 1 Резултати



Диаграма 1 Време за изпълнение

На Диаграма 1 е посочено времето за изпълнение на програмата. По У координатата се намира времето за изпълнение в милисекунди, а по Х координатата - броя зададени нишки от потребителя.

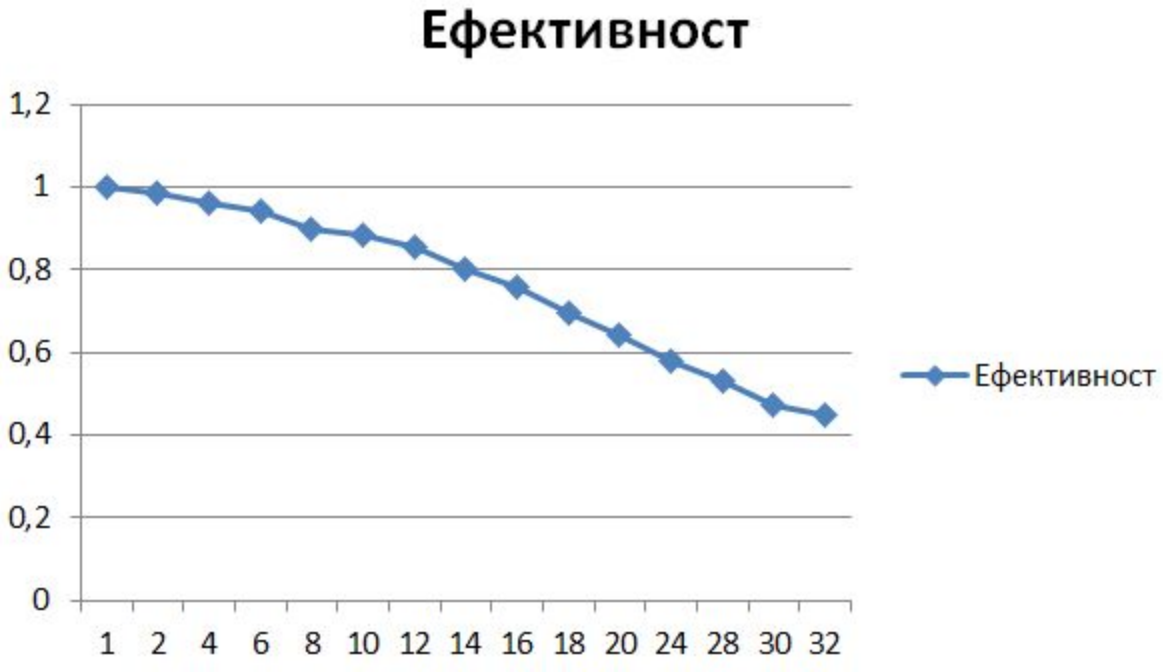
Tp = време за изпълнение на програмата с p на брой нишки

Диаграма 2 Ускорение

На втората диаграма виждаме ускорението на база данните от Таблица 1. За пресмятането му е използвана формулата

$$Sp = \frac{T1}{Tp},$$

където T е времето за изпълнение на p на брой нишки



Диаграма 3 Ефективност

На диаграма под номер 3 е илюстрирана ефективността.

Формула за изчисление:

$$Ep = \frac{Sp}{p}$$

Sp - ускорението за p на брой нишки

4. Използвана литература

[1] Pi - Chudnovsky, craig-wood, https://www.craig-wood.com/nick/articles/pi-chudnovsky/

[2] Parallel Algorithm Models, Parallelcomp, http://parallelcomp.uw.hu/ch03lev1sec6.html

[3] Executor Service in Java, HowToDoInJava, https://howtodoinjava.com/java/multi-threading/executor-service-example/

[4] Aman Agnihotr, StackOverflow, https://stackoverflow.com/questions/22051345/breaking-a-mathematical-range-in-equal-parts

[5] Apfloat for Java, apfloat, http://www.apfloat.org/apfloat_java/