Проект

по Системи за Паралелна Обработка

Паралелен Алгоритъм за

Изчисляване на ПИ

Изготвил:

 Стефан Василев

Ръководител:

проф. д-р Васил Георгиев Цунижев

1. Постановка на задачата

 Числото (стойността на) Pi може да бъде изчислена по различни начини. Разглеждаме следната формула за приближено пресмятане на Pi:



Бележка: Колкото повече събираеми генерираме толкова по-добра ще бъде точността с която пресмятаме Pi.

1. Начин на употреба на програмата

За да си пуснем програмата, трябва да отворим команден прозорец на операционната система, да се навигираме до папката “…\CalculatePiMultithreaded\bin”, след което да изпълним командата:

java Main [<key> <value>] [<key > <value>] [<key > <value>] [-q]

Където параметрите заградени в скоби са опционални, а пък key принадлежи на { “-p”, “-t”, “-o” }. Ако key = “-o”, то value трябва да бъде стринг. В противен случай value принадлежи на множеството на целите положителни числа по-големи от 0.

Пояснения:

* Когато key = “-p” съответното му value, трябва да бъде положително цяло число. То ще указва броя на знаците на Pi, след десетичната запетая. Например командата “java Main –p 100” означава, че ще пресмятаме Pi с точност до 100 знака.
* Когато key = “-t” съответното му value, трябва да бъде положително цяло число. То ще указва броя на използваните нишки при пресмятане на Pi. Например командата “java Main –t 3” ще означава че Pi ще бъде пресмятано само с 3 нишки. Ако аргументът value е по голям от броя на ядрата, с които разполага процесора то програмата ще се изпълни с възможно най-големия брой ядра.
* Когато key = “-o” съответното му value, текст. То ще указва мястото в операционната система, в което да се запише пресметнатата стойност за Pi. Ако не бъде указана стойност на този аргумент, то стойността на Pi се записва в “result.txt”, който ще се намира в текущата папка, от която изпълняваме командата.
* Ако някъде измежду параметрите попаднем на key = “-q”, то това означава, че стойността на Pi няма да се запише във файл.
1. Описание на реализирания алгоритъм

 За паралелното решаване на задачата е избран подходът на многонишковото програмиране. Тъй като всяко събираемо от сумата не зависи от предходни изчисления, то работата по изчисленията на всяко събираемо може да бъде разпределена в различни нишки. Тъй като с нарастването на к пресмятането на всяко следващо събираемо става все по бавно трябва да разпределим работата на нишките на определени интервали. Например ако ще имаме 100 итерации, (тоест к = 0,1,2, …, 100), и четири нишки, то първата нишка ще смята сумите за к принадлежащо на [0, 50]. Втората нишка ще смята за к принадлежащо на [51, 75]. Третата за к принадлежащо на [76, 89]. И последната за к принадлежащо на [90, 100). Дължината на всеки следващ интервал е равна на дължината на предходния интервал, разделена на 2.

1. *Тестови замервания и анализ на метричните показатели*

 Разработено е приложение на JAVA и е тествано на 4 ядрен мултипроцесор с цел да се оценят ускорението S (забързване, speed-up) и ефективността Е (efficiency) на описания алгоритъм. Броят на итерациите е 50. Броят на знаците след десетичната запетая е 100. Тук T(p) е времето необходимо за завършване на работата на алгоритъм с p на брой нишки то:

S(p) = T(1)/T(p)

E(p) = S(p)/p

|  |  |  |
| --- | --- | --- |
| p | S(p) | E(p) |
| 1 | 1.0000 | 1.0000 |
| 2 | 1.1087 | 0.5544 |
| 3 | 1.4521 | 0.4840 |
| 4 | 1.8234 | 0.4559 |