Изчисление на π с произволно висока точност

# Проект по СПО Георги Ангелов, ф.н.80699 Компютърни науки, 4. група

# Въведение

π е една от най-известните математически константи. Пресмятането на π с висока точност се е превърнало в нещо подобно на спорт или предизвикателство. Текущият рекорд е 12,100,000,000,050 изчислени позиции след десетичната запетая.  
От практическа гледна точка, точност от 39 десетични цифри би била достатъчна, за да се определи дължината на вселената с точност до атом.

# Формула

За бързо пресмятане на π най-често се сходящи редове, които имат граница въпросната константа. Има десетки такива редове, като разликата е в бързината на сходимост. В този проект се използва ред открит от индийския математик Srinivasa Ramanujan:

Това е един от по-бързо сходящите редове, като за всяко следващ член на сумата се получават около 6 десетични цифри от π. Има и по-бързо сходящи редове, като този на братята Chudnovsky, който пресмята 14 цифри на събираемо.

Формулата на Chudnovsky също може да бъде използвана в този проект с леки модификации.

# Серийна имплементация

В класическия, последователен, вариант за пресмятане на горната формула би било добра идея да се пази временната стойност на частите от формулата, които могат да се преизползват от следващата итерация на сумата. Такива натрупващи се стойности са: . Така, за получаване на (n+1)-вия член от n-тия тези запомнени стойности от предишната итерация се умножават съответно с

Тук f е следната функция: , която може да се имплементира рекурсивно като   
Това позволява тя да се пресметне с O(log(b-a)) на брой необходими умножения, което е по-добре при големи n заради времето, което е необходимо за умножение на големи числа.

След измерване на необходимото време за всеки вид операция се оказва, че доминиращата операция е делението на числа с голяма точност, което е 99,9% от цялостното време за пресмятане на π с дадена точност. Фигурата по-долу илюстрира това съотношение. Стойностите са в милисекунди за фиксиран брой членове на сумата.

# Паралелна имплементация

След наблюдение на горните времеви съотношения става ясно, че делението е доминиращата операция, което означава, че то трябва да е фокуса на паралелизацията.

Тъй като умножението е толкова бързо, то дори умноженията да не се правят паралелно, а чрез натрупване, както при последователния алгоритъм, то няма да има никакво ускорение. Дори би се наблюдавало забавяне, защото времето, необходимо за преизчисление и синхронизация би било повече от самото умножение.

Паралелният алгоритъм е следния (k = брой нишки, m = желан брой събираеми):

1. Числителите и знаменателите на m-те члена на сумата се пресмятат последователно от една нишка, която ги запазва в масив/опашка terms.
2. Паралелно k на брой нишки взимат елемент от масива/опашката, като изчакват ако няма пресметнат такъв, и извършват делението. След това добавят резултата към променлива, локална за съответната нишка.
3. След приключване на всички нишки с цикъл по всички тях се сумират частичните суми от локалните променливи, резултата се дели на 4\*882 и се взима реципрочната му стойност.

Тъй като времето за генериране на тези числители и знаменатели е ~1000 пъти по-малко от времето за останалите операции, то то може да се извърши и преди стартирането на нишките, като се попълни масив с m на брой полета от по 2 големи цели числа.

Друг вариант е да се използва Producer-Consumer модел, като така няма нужда да се съхраняват всички събираеми предварително в паметта, но е необходима синхронизация на достъпа до опашката.

И двата варианта имат практически едно и също бързодействие, заради доминиращото време за изпълнение на деленията – нишките прекарват много повече време при пресмятането, отколкото върху критичната област (достъпа до опашката). В първият вариант за 10000 събираеми предварителното попълване на масива става за 92ms (в сравнение с 21000ms цялостно изчисление на 4 нишки).