Софийски университет „Св. Климент Охридски“, Факултет по математика и информатика

Курсов проект

по Разпределени софтуерни архитектури

Задача 13:
„Пресмятане на Pi – Chudonovsky“

Изготвил:
Симеон Станимиров Александров, 61916, Софтуерно инженерство, курс 3, група 5

Научен ръководител:
ас. Христо Христов

Дата: 23.06.2018 г. Подпис:……

Съдържание

Цел на проекта

Целта на проекта е пресмятането на числото Pi чрез паралелен алгоритъм. Използвайки сходящи редове, е възможно да се пресметне стойността на числото Pi с произволно висока точност. Този проект представлява имплементация на реда на братята Chudonovsky за изчисление на числото Pi, цитиран по-долу:

(3) *Chudonovsky*, 1987

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

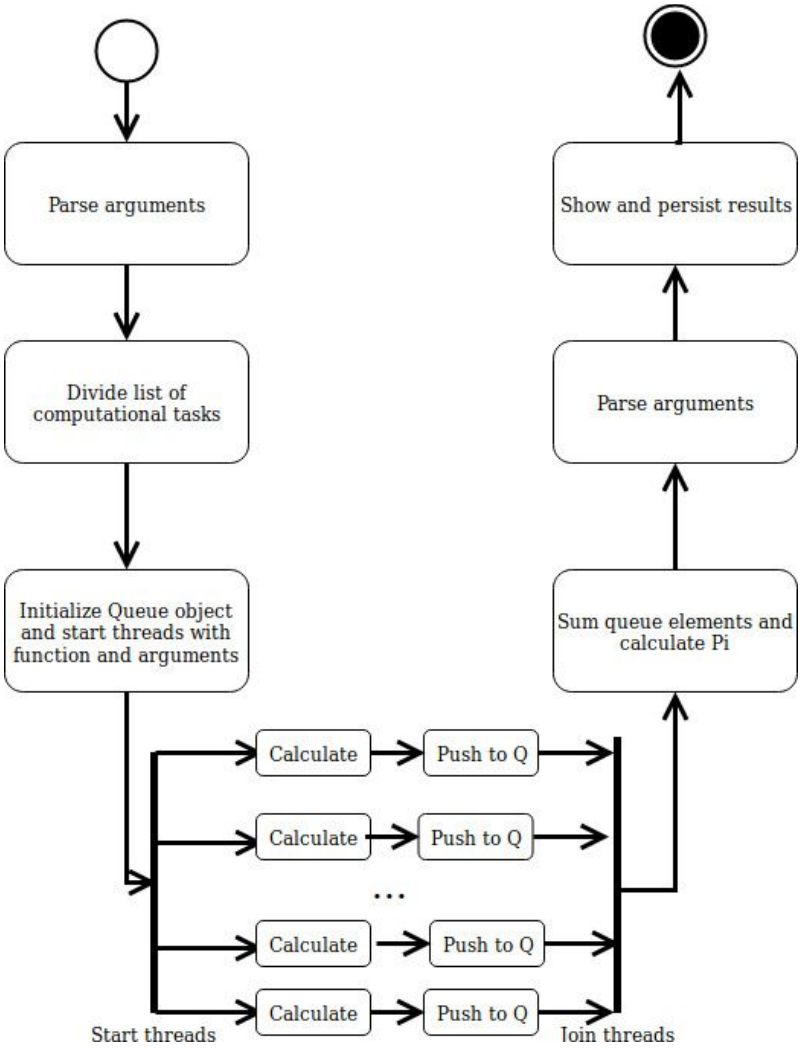
Изискванията към програмата са следните:

- Програмата трябва да използва паралелни процеси (нишки), за да разпредели работата по пресмятането на реда на повече от един процесор;
- Стойностите, които са необходими за изчисление на реда, могат да бъдат подадени като командни параметри и програмата трябва да може да ги разпознава и използва;
- Броят на членовете на реда като команден параметър трябва да има вида **“-p 2500”**;
- Заявка за максималния брой нишки трябва да изглежда по този начин: **“-t 32”**, ако отсъства, процесът по подразбиране е един;
- Програмата трябва да извежда подходящи съобщения на различните етапи от работата си;

- Резултатът от работата си (стойността на π) програмата трябва да записва в изходен файл, който може да е зададен като команден параметър **“-o pi.txt”**, ако липсва, по подразбиране името му е *result.txt*;
- Програмата трябва да осигури възможност за „тих“ режим на работа, при който се извежда само времето, отделено за изчисление на π, отново чрез подходящо избран команден параметър **“-q”**.

Описание и реализация на алгоритъма

Реализацията на алгоритъма и използването на паралелизъм при пресмятането на числото Pi е постигната на езика Python. Следната диаграма представя абстрактен поглед върху изпълнението на програмата:



След обработката на аргументите, списъкът със задачи се разделя в зависимост от броя на нишките по принципа на Дирихле, за да бъдат нишките оптимално натоварени(тоест да не свършва една нишка с изпълнението си много преди друга). Разделянето е илюстрирано по следния начин:

```
>>> l = [1,2,3,4,5,6,7,8,9,10]
>>> threads=3
>>> print [ l[i::threads] for i in xrange(threads) ]
[[1, 4, 7, 10], [2, 5, 8], [3, 6, 9]]
>>>
```

По-долу е представена най-важната част от проекта. Инициализира се обект от класа Queue, който ще служи за споделен между нишките контейнер. След това последователно се създават нишките, като всяка от тях приема функция и

аргументи към нея. Всяка една нишка прави изчисленията за част от реда и в края на изпълнението си записва резултатът си в опашката *q.*

```
q = Queue(int(args.t)) # Max size is the number of threads

for i in xrange(1,int(args.t) + 1):
    thread = Process(target=calculate_subset, args=(q, i, divided[i - 1], args.q, ))
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
```

Методите *calculate_subsetl(n)* и *calculate_all(q, thread_id, list_to_calc, quiet_mode* пресмятат съответно един елемент от реда и всички елементи от списъка, с който е стартирана нишката.

След приключване на всички нишки, стойностите от опашката *q* се сумират и се достига до стойността на числото Pi. На конзолата се извежда времето за изпълнение, а стойността на числото Pi се запазва в тесктов файл.

Всички библиотеки, които програмата използва са вградени в езика Python и не предполагат предварително инсталиране:

- Argparse - за обработка на аргументи подадени от командния ред
- Datetime - библиотека за представяне на време в различни формати
- Класа Decimal от decimal модулът - представяне на числа с плаваща запетая
- Класовете Process, Queue от multiprocessing модулът - предоставят

  възможност за паралелизъм

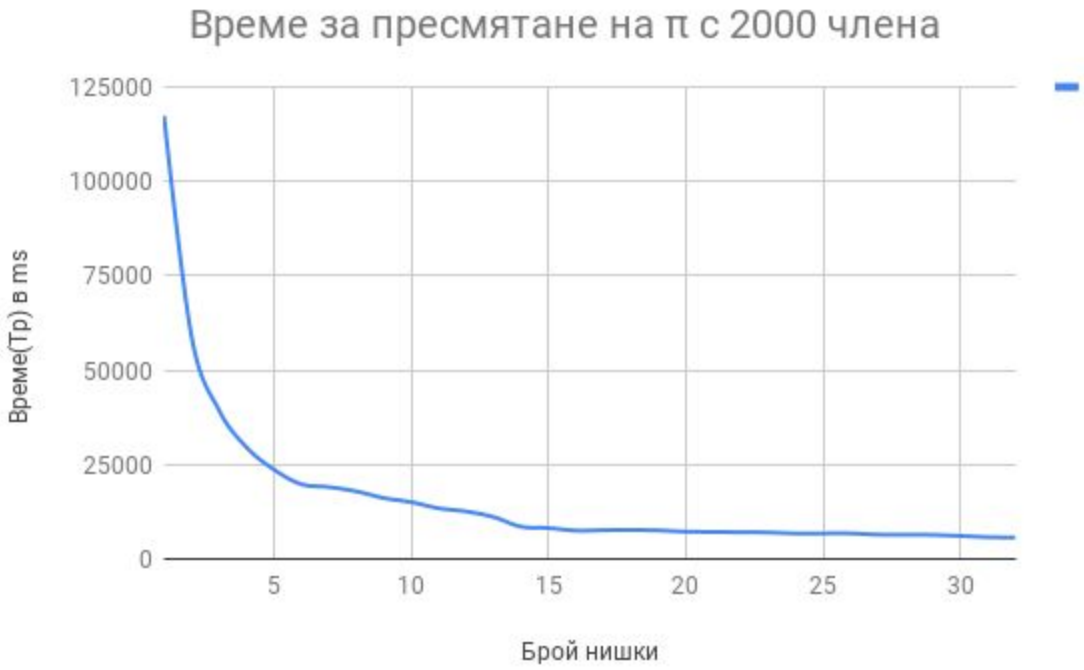
Резултати

Ще покажем резултати от изпълнението на програмата при стойност на броя членове на реда 2000 (*-p 2000*). Програмата е тествана машината t5600.rmi.yaht.net, която разполага с 32 ядра. Таблицата и графиките по-долу представят визуално сравнение между резултатите на изпълнението на една и съща програма с единствената разлика в подадените параметри - броя на нишките.
За смятането на изчислението и ускорението са използвани следните формулите:

S(p) = T(1)/T(p) – за ускорение
E(p) = S(p)/p – за ефективност

Получените резултати са представени в табличен и графичен вид.

n threads	T(p) in ms	S(p)	E(p)
1	117362	1	1
2	59081	1.986459268	0.9932296339
3	39459	2.974277098	0.9914256992
4	29633	3.960516991	0.9901292478
5	23716	4.948642267	0.9897284534
6	19811	5.92408258	0.9873470967
7	19052	6.16008818	0.8800125971
8	17924	6.547757197	0.8184696496
9	16149	7.267446901	0.8074941001
10	15061	7.792444061	0.7792444061
11	13435	8.735541496	0.794140136
12	12598	9.315923162	0.7763269302
13	11126	10.54844508	0.8114188526
14	8568	13.69771242	0.9784080299
15	8201	14.31069382	0.9540462545
16	7534	15.577648	0.9736029997
17	7632	15.37762055	0.9045659144
18	7691	15.25965414	0.8477585634
19	7569	15.50561501	0.8160850005
20	7244	16.20127002	0.8100635008
21	7171	16.36619718	0.779342723
22	7098	16.53451677	0.7515689439
23	7033	16.68733115	0.7255361371
24	6777	17.31769219	0.7215705081
25	6789	17.28708204	0.6914832818
26	6782	17.3049248	0.6655740308
27	6468	18.14502165	0.6720378387
28	6433	18.2437432	0.6515622571
29	6374	18.41261374	0.6349177153
30	6113	19.19875675	0.6399585583
31	5793	20.25927844	0.653525111
32	5690	20.62601054	0.6445628295


Време за пресмятане на π с 2000 члена
125000
100000
75000
50000
25000
0
Време(Tp) в ms
5
10
15
20
25
30
Брой нишки


Ускорение при пресмятане на π с 2000 члена
25
20
15
10
5
0
Ускорение(Sp)
5
10
15
20
25
30
Брой нишки

Източници

https://docs.python.org/2/library/multiprocessing.html
https://www.encyclopediaofmath.org/index.php/Dirichlet_box_principle
https://timber.io/blog/multiprocessing-vs-multithreading-in-python-what-you-need-to-know/