|  |
| --- |
| Пресмятане на Pi по формула на Ramanujan – SUPER PI |
|
| Фак. номер: *81117* |
| Кристина Георгиева Гочева |
| КН, трети курс, пета група |
| Проект по Системи за паралелна обработка, 2017г.  проверил: ............................ /ас.Христо Христов/ |

Съдържание

[1 Условие на задачата 3](#_Toc486608367)

[2 Анализ 3](#_Toc486608368)

[2.1 UML class diagram 3](#_Toc486608369)

[2.2 Функционален анализ 3](#_Toc486608370)

[2.2.1 Модел на декомпозиция 4](#_Toc486608371)

[2.2.2 Гранулярност 4](#_Toc486608372)

[2.2.3 Балансиране и разпределение 5](#_Toc486608373)

[2.3 Нефункционален анализ 5](#_Toc486608374)

[3 Проектиране 5](#_Toc486608375)

[3.1 Функционално проектиране 5](#_Toc486608376)

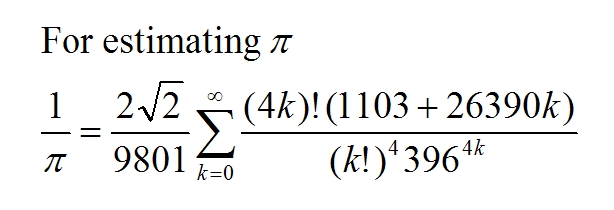
[3.2 Нефункционално проектиране 6](#_Toc486608377)

[3.3 Тестване 7](#_Toc486608378)

[4 Източници 11](#_Toc486608379)

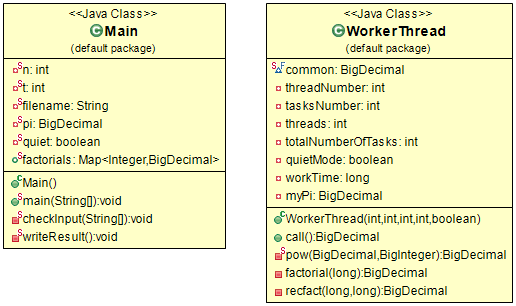
# Условие на задачата

Да напишете програма за изчисление на числото Pi изпозлвайки един от бързо сходящите към Pi редове - този, открит от индийския математик Srinivasa Ramanujan през 1910-1914 година.



# Анализ

## UML class diagram



## Функционален анализ

Съществуват различни подходи за намирането възможно повече цифри след десетичната запетая за числото pi и неговото многонишково пресмятане. Разгледаните образци представят следната оптимизация за личните им разработки :

| Образец No. | P | N | T | Тестова машина |
| --- | --- | --- | --- | --- |
| **1** | NO INFORMATION | 22,459,157,718,361 | 105days | 4 x Xeon E7-8890 v3 @ 2.50 GHz 1.25 TB DDR4 20 x 6 TB 7200 RPM Seagate |
| **2** | NO INFORMATION | 12,100,000,000,050 | 94 days | 2 x Xeon E5-2690 @ 2.9 GHz 128 GB DDR3 @ 1600 MHz 24 x 3 TB |
| **superPi** | Подробна информация в следващите таблици и графики | | | |

При всяка от тези реализации е използвано различно задаване на числото пи, така че не може да се смятат резултатите за абсолютно равностойни. Известни са няколко алгоритъма за изчисляване на пи, някои са геометрични, други са зададени със редове, подобен на този в условието за проекта SUPER PI.   
 Анализът на имплементаицията разработките на проектите използвани за образци води до заключение в двете основни решения при избора на разработка – модел на декомпозиция и избор на гранулярност.

### Модел на декомпозиция

Избраният модел на декомпозиция в този проект е Master-Slave имплементация на кода. По този начин имаме една основна нишка (Main), която извършва две основни дейности- разпределя работата между останалите нишки и следи за получаване на крайни резултати от всяка нишка. Втория тип нишки – p на брой slave- нишки, които в тази конкретна импементация разгледани като Worker Thread, които се грижат получената част от задача да бъде изпълнена.

### Гранулярност

Най- общо гранулярността може да се раздели на една и фина. Едра – когато цялата задача е разделена на няколко условно казано големи порции и фина- когато е разделена на множество под задачки и при изпълнението на една, свободната slave-нишка би получила следваща.

В имплементация на Super Pi се използва едра гранулярност. Имаме декомпозиция на данните, която пряко се отразява на ускорението. На всяка нишка още при самото начало се задават задачите, които тя трябва да свърши и те са p на брой пресмятания, които обаче не са последователни членове в редицата зададена в условието.

### Балансиране и разпределение

Проекът предоставя статично разпределяне със създаване на начен брой нишки и задаване на задачите им от самото инициализиране. При паралелизъм p всяка нишка от самото начало получава членовете на реда, които трябва да пресметне, те са за всяка i-та нишка съответно i , i+p, …i+N-p елементите от редицата.

## Нефункционален анализ

Разработеният проект използва Java JRE 8. Не се налага допълнително инсталиране на библиотеки и dependencies за изпробване функционалността на програмата. Приложението SUPER PI е конзолно и обработва списък от допълнителни опции, една, от които е печатане във файл на получения резултат.

# Проектиране

## Функционално проектиране

Входните данни се определят от следните параметри:

* **-q** – Възможност за работа на програмата в „тих“ режим. Когато се зададе този параметър се извежда само времето отделено за изчисление на *e*.
* **-t** –максималния брой нишки (задачи), на които разделяме работата по пресмятането на *e.*
* **-p** – Задава точността на пресмятанията (брой членове на реда във формулата).
* **-о** – Записва резултата от работа си (стойността на пи) във изходен файл. По подразбиране е файла „result.txt“.

Алгоритъмът разделя членовете на реда, така че всяка нишка да обработва еднакъв брой събираеми. Всеки резултат от процеса изпълнен в нишка се предава на главната нишка, която от своя страна обхожда ArrayList с получените резултати и събира отделните членове. Основните класове при разработата за WorkerThread и Main.   
 WorkerThread е имплементация на интерфейса Callable<BigDecimal>. Високата точност при изчисление на числото налага SUPER PI да използва BigDecimal за работа с факториели и повдигане на степени на цели положителни числа. Основната работа всяка нишка извършва в имплементиранията на метода call(), който и връща сума от сметнатите членове на редицата. Използва се оптимизирана имплементация на повдигането на степен – методът pow() и на факториел – factorial() .   
 С цел оптимизация е наложена и меморизация на факториелите. В ConcurrentMap<Integer,BigDecimal> се пазят като ключ числата, а като стойности съответстващите им факториели.Така при всяко извикване първо се проверява в Map-a най-близкото възможно число до търсенето и се извършва пресмятане само за разликата.

## Нефункционално проектиране

Представената по-долу работа на SUPER PI е изследнвана на база няколко сървърни устройства.

|  |  |  |  |
| --- | --- | --- | --- |
| Параметри: | ats24.rmi.yaht.net | cf.rmi.yaht.net | t5600.rmi.yaht.net |
| Architecture:   CPU op-mode(s):   CPU(s):   Thread(s) / core:   NUMA node(s):   Vendor ID:   CPU family:   Model name:  Stepping:   CPU MHz: | x86\_64  32-bit, 64-bit   24  1  4  GenuineIntel  6  Intel(R) Xeon(R) CPU X5650 @ 2.67GHz  2  2665.823 | x86\_64  32-bit, 64-bit   8  1  2  GenuineIntel  6  Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz   7  2400.148 | x86\_64  32-bit, 64-bit   32  2  2  GenuineIntel  6  Intel® Xeon® CPU E5-2660 0 @ 2.20GHz  7  2403.671 |

## Тестване

Времената са измерени в милисекунди.

Ускорението при използване на повече от една нишкасе изчислява по формулата: Sp = T1 / Tp.

Ефективността се изчислява по формулата: Ep = Sp / p.

* Тест от ats24.rmi.yaht.net

|  |  |  |  |
| --- | --- | --- | --- |
| Брой нишки | Време | Ускорение | Ефективност |
| 1 | 443622 | 1 | 1 |
| 2 | 228377 | 1.942495236 | 0.971247618 |
| 4 | 121017 | 3.66577974 | 0.916444935 |
| 8 | 68506 | 6.475653299 | 0.809456662 |
| 16 | 52310 | 8.480545052 | 0.530034066 |
| 32 | 55432 | 8.002915252 | 0.250091102 |
| 64 | 55331 | 8.017610381 | 0.125275162 |

* Тест от cf.rmi.yaht.net

|  |  |  |  |
| --- | --- | --- | --- |
| Брой нишки | Време | Ускорение | Ефективност |
| 1 | 517215.6 | 1 | 1 |
| 2 | 250838.2 | 2.061949097 | 1.030974549 |
| 4 | 139049.4 | 3.719653591 | 0.929913398 |
| 8 | 82690.2 | 6.254859705 | 0.781857463 |
| 16 | 68191 | 7.584807379 | 0.474050461 |
| 32 | 68300.6 | 7.572636258 | 0.236644883 |
| 64 | 68145.2 | 7.589905085 | 0.118592267 |

* Тест от t5600.rmi.yaht.net

|  |  |  |  |
| --- | --- | --- | --- |
| Брой нишки | Време | Ускорение | Ефективност |
| 1 | 395077.4 | 1 | 1 |
| 2 | 200547.4 | 1.96999512 | 0.984998 |
| 4 | 106972 | 3.69327861 | 0.92332 |
| 8 | 55462 | 7.12338899 | 0.890424 |
| 16 | 30658.2 | 12.8865165 | 0.805407 |
| 32 | 25720 | 15.3607076 | 0.480022 |
| 64 | 26664.4 | 14.8166619 | 0.23151 |

# Източници

[1] http://pi2e.ch/blog/ - World Record Size Computation by Peter Trueb

[2] http://www.numberworld.org/misc\_runs/pi-12t/ - 12.1 Trillion Digits of Pi By Alexander J. Yee & Shigeru Kondo