Проект

по Системи за Паралелна Обработка

Паралелен Алгоритъм за

Изчисляване на ПИ

Изготвили:

Ива Вълова, Марияна Тодорова

Ръководител:

ас. Христо Христов

Проверил: ........................

(ас. Христо Христов)

# Постановка на задачата

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

Вашата задача е да напишете програма за изчисление на числото Pi изпозлвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на Pi със зададена от потребителя точност. Изискванията към програмата са следните:

(o) Команден параметър задава точността на пресмятанията. По

Ваше желание, точността се изразява или в брой цифри след десетичната запетая или в брой членове на реда. Командният параметър задаващ точността има вида ­ “­p 10000”;

(о)  Друг команден параметър задава максималния брой нишки  (задачи)  на които разделяме работата по пресмятането на  Pi – например “–t 1” или “–tasks 3”;

(о) Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление на стойността на Pi;

(o) Записва резултата от работа си  (стойността на  Pi) във изходен файл, зададен с подходящ параметър, например “-o result.txt”. Ако този параметър е изпуснат, се избира име по подразбиране фиксирано отнапред във Вашата програма;

(o) Да се осигури възможност за „quiet“ режим на работа на програмата, при който се извежда само времето отделено за изчисление на  Pi, отново чрез подходящо избран друг команден параметър – например“­q”;

ЗАБЕЛЕЖКА:

(о)  При желание за направата на подходящ графичен потребителски интерфейс  (GUI)   с помощта на класовете от пакета javax.swing задачата може да се изпълни от двама души;

Разработването на графичен интерфейс не отменя изискването Вашата програма да поддържа изредените командни параметри.  В този случай към функцията на параметъра параметъра „­q“ се добавя изискването да не пуска графичният интерфейс. Причината за това е,  че Вашата програма трябва да позволява отдалечено тестване, а то ще се извършва в terminal.

# 2.Описание на алгоритъма и реализацията

Приемаме точността на пресмятане, като броят знаци след десетичната запетая, което се задава от потребителя. Както и задава броят нишки, който да изпълняват задачата.

Класът Calculator съдържа логиката по пресмятането на числото Пи. За паралелното му изчисляване сме използвали подхода Task decomposition, който е реализиран чрез многонишково програмиране.

Чрез него отделните процеси извършват изчисленията едновременно, използвайки едни и същи входни данни и пресмятат частични крайни резултати. Тези крайни резултати ги сумираме и по този начин дават желания резултат. По описаният подход се намалява времето, необходимо за пресмятането на числото  (стойността на ) Pi .

Отделна логическа единица в програмата ще е пресмятане на един член на реда за Pi. Тези единици са задачите, които ще бъдат изпълнени паралелно и независимо една от друга. Затова използваме класът Task, който да пресмята един член на реда. Всяка подзадача връща един резултат, затова и класа, имплементира Java интерфейса Callable.

*private class Task implements Callable<Apfloat> {*

private int index;

public Task(int index) {

this.index = index;

}

@Override

public Apfloat call() throws Exception {

return termFunction(index);

}

*}*

Действията, които всяка една задача извършва са независими от действията на другите задачи. За механизма по паралелното изпълнение на подзадачите класа Calculator използва java.util.concurrent.ExecutorService и java.util.concurrent.CompletionService. Последният използва механизъм, чрез който да отделим метода по създаването на нови задачи от начина, по-който консумираме резултати им. Задачите създаваме и стартираме чрез следния програмен фрагмент:

*for (int i = 0; i <= this.precision / 14; i++) {*

completionService.submit(new Task(i));

}

Наблюденията ни показват, че пресмятането на 1 член на реда ни дава приблизителна точност след десетичната запетая на 14 знака. Затова и се нуждаем от пресмятането на (брой знаци) / 14 на брой задачи. Резултатите от изпълнените задачи консумираме чрез:

for (int i = 0; i <= this.precision/14; i++) {

Future<Apfloat> workerResult = completionService.take();

sum = sum.add(workerResult.get());

}.

Приложението също реализира прост графичен интерфейс.

# 3.Тестови замервания и анализ на метричните показатели

Разработено е приложение на JAVA и е тествано на 4 ядрен процесор с цел да се оценят ускорението S (забързване, speed-up) и ефективността Е (efficiency) на описания алгоритъм, където ако 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. Замерванията направени за 10 000 знака точност след десетичната запетая.