Проект по ПРПСП

Многонишково умножение на матрици

Реализиран от

Красимир Христов Кръстев

Софтуерно Инженерство, втори курс

Факултетен номер 61163

Условие на задачата:

**Зад. 6 (Умножение на матрици)**

Разглеждаме матриците A с размерност (m, n) и B с размерност (n, k). Матрицата C =

A.B, равна на произведението на A и B ще има размерност (m, k). Да се напише програма

която пресмята матрицата C. Работата на програмата по умножението на матриците да се

раздели по подходящ начин на две или повече нишки (задачи).

Изискванията към програмата са следните:

(o) Размерността на матриците се задава от подходящо избрани командни параметри;

Елементите на матриците генерираме произволно с помощта на **Math.random()** или класа

**java.util.Random**; (Тоест матриците може да имат float или double елементи)

(о) Друг команден параметър задава максималния брой нишки (задачи) на които

разделяме работата по пресмятането елементите на C;

(о) Извежда подходящи съобщения на различните етапи от работата си, както и

времето отделено за изчисление;

(o) Да се осигури възможност за „quiet“ режим на работа на програмата, при който се

извежда само времето отделено за изчисление на резултатната матрица (чрез подходящо

избран друг команден параметър);

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**;

(о) Задачата може да се реши и с помощта на RMI (**java.rmi**). За целта трябва да се

помисли за разпределения достъп до общия ресурс в случая матриците A, B и C.

Алгоритъм:

Data decomposition

В началото програмата се инициализират входните данни – размрностите на матриците и броя на нишките, които ще осъществяват умножението







След това се инициализират матриците А и Б с произволни стойности и резултатната матрица получава начални стойности на елементите 0.0

Програмата се стартира в класа Program.java и това е master процеса. По – късно в него се пускат последователно – зададения брой нишки. Изпълнението зависи от броя на независимите процесори на машината. При четириядрен процесор и 4 нишки – всички нишки ще работят паралелно и няма да се изчакват, тъй като пишат на различни места в резултатната матрица.

Програмата реализира алгоритъма за умножение на матрици еднонишково и многонишково. В конзолата се изкарват подходящи съобщения за времето на работа.

Работата се разделя на подходящи порции и се изпълнява едновременно.

За жалост на машината, на която пиша документацията, нямам Java и не мога да предоставя резултати от работата на програмата, но алгоритъма е ясен и недълъг и всъщност доста тривиален ☺

Следните тестови данни са получени от изпълнението на програмата:

матрица 2000-200, 2000-200

При работа на само една нишка, времето за изпълнение е:

1 нишка – 28026

При работа на 2 нишки, времето за изпълнение е:

2 нишки – 27459

При работа на 3 нишки, времето за изпълнение е:

3 нишки – 27413

При работа на 4 нишки, времето за изпълнение е:

4 нишки – 23800

При работа на 5 нишки, времето за изпълнение е:

5 нишки – 15165

При работа на 6 нишки, времето за изпълнение е:

6 нишки – 15765

От тези тестове ясно се вижда, ползата от използването на много нишки. Разликата между стандартното еднонишково изпълнение и използването на 5 нишки за изпълнение на задачата е очевидна:

 1 нишка – 28026 5 нишки – 15165

Ускорението е почти двойно.

Но все пак има една точка (5 нишки) от която нататък, времето спира да намалява и започва да се увеличава. Следния пример показва точно това:

5 нишки – 15165 6 нишки – 15765

Извода е, че не можем да делим работата до безкрай. 6 нишки в случая работят по-бавно от 5 нишки – с цели 600.