Проект

по Системи за Паралелна Обработка

Паралелен алгоритъм за умножение на матрици

Изготвил:

Емануела Миланова

Ръководител:

ас. Христо Христов

Проверил: ........................

(ас. Христо Христов)

1. **Въведение**

Умножението на матрици е една от най – основните операции в линейната алгебра. Умножението на матрици служи в като основен компонент в много различни алгоритми, включително решения за системи линейни уравнения на детерминанта на матрица и транзитивно закриването на графика. Умножението на големи матрици е обемна операция и изисква множество изчисления. Съвременните изчислителни системи работят паралелно и при наличието на подходящ алгоритъм, който използва всички налични ресурси, може да се постигне голямо ускорение на изчислението.

1. **Постановка**

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:

* Размерността на матриците се задава от подходящо избрани командни параметри   – например „­m 1024 ­n 512 ­k 2048“;
* Елементите на матриците генерираме произволно с помощта на Math.random() или класа java.util.Random; (Тоест матриците може да имат float или double елементи)
* Команден параметър указващ входен текстов файл,   съдържащ матриците,   които ще умножаваме – например „­i m3x­data.in“. Който и да е от параметрите „­m“ „­n“ „­k“ и параметърът „­i“ са взаимно­изключващи се; Ако все пак бъдат зададени и четирите (или някой от „­m“ „­n“ „­k“, заедно с „­i“), решението как да реагира програмата е Ваше.  (Идеята? Или генерираме произволни матрици или използваме отнапред даден вход). Форматът на файла m3x­data.in е следният:
  + 1вият ред съдържа числата m, n и k, разделени с интервал;
  + На оставащите m+n реда във файла са разположени редовете на матриците A и B. Първо тези на A, след което тези на B. Елементите на всеки ред от матрицата са разделени със интервали (един или повече).
* Команден параметър указващ изходен файл,   съдържащ резултата от пресмятането   – например „­о m3x­data.out“. Форматът на изходният файл е следният:

m  k

c11 c12 c13 … c1k

c21 c22 c23 … c2k

…

cm1 cm2 cm3 … cmk

При липса на този команден параметър не се записва във файл резултата от умножението на матриците;

* Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането елементите на C – например “–t 1” или “–tasks 3”;
* Извежда подходящи съобщения на различните етапи от работата си,   както и времето отделено за изчисление;
* Да се осигури възможност за „quiet“ режим на работа на програмата, при който се извежда само времето отделено за изчисление на резултантната матрица, отново чрез подходящо избран друг команден параметър – например “­q”;
* Командните аргументи на терминална Java програма, получаваме във масива String args[] на main() метода, намиращ се в стартовият клас. За „разбирането“ им (анализирането им) може да ползвате и външни библиотеки писани специално за тази цел . Един добър пример за това е: Apache Commons CLI.

1. **Описание на избрания алгоритъм**

Алгоритъмът има няколко основни стъпки.

Първата стъпка е четенето на аргументите и анализирането им. В случай, че са зададени стойности за числата “m”, “n” и “k: генерираме две случайни матрици, а в случай, че е зададена стойност за параметъра “i” създаваме матриците от файл. Ако и двата параметъра имат стойност то генерираме случайни матрици. Следва проверка за наличието на параметър за брой нишки и ако лисва такъв задаваме броя на нишките да бъде броя на ядрата на текущата тестова машина. Прави се проверка и за параметъра “q” и при наличието му се изкарва само съобщение за времето, което е било нужно да се умножат матриците.

Втората и най – основна стъпка е умножението на матриците. Първо изчисляваме на колко парчета можем да разделим работата по следната формула:

*Парчета работа = броя редове на изходната матрица / броя нишки*

Следва създаването на „бъдещите“ задачи. Всяка нишка получава даден брой „бъдещи“ задачи, които трябва да изпълни. При получаването на резултат от някоя от нишките се попълват сметнатите от тази нишка стойности в изходната матрица.

Третата и последна стъпка е обработката на получената матрица. В случай, че е зададен параметъра “o” създаваме изходен файл и записваме умножената матрица там. Ако сме в тих режим на работа не правим нищо, иначе изкарваме на конзолата получената матрица. С това завършва програмата.

1. **Тестови замервания и анализ на метричните показатели**

Разработеното приложение е написано на Java по горе описания алгоритъм и е тествано на 8 ядрен мултипроцесор. Замерванията са направени чрез произволно генериране на две матрици с размерно (1024, 512) и (512, 2048).

Ще оценим ускорението S (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*

Ще тестваме на 1, 2, 4 и 8 нишки времето на изпълнение на програмата, ускорението и ефективността.