ГУАП

КАФЕДРА № 51

ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| доцент, к.т.н. | 05.12.2019 | Е.М. Линский |
| должность, уч. степень, звание | дата | инициалы, фамилия |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8

СОЗДАНИЕ ПРОГРАММЫ НА ЯЗЫКЕ JAVA

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |
| --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5723 | 05.12.2019 | А.В. Царевский |
|  |  | **дата** | **инициалы, фамилия** |

**Санкт-Петербург 2019**

Задание

Реализовать класс ParallelMatrixProduct для многопоточного умножения матриц UsualMatrix. В конструкторе класс получает число потоков, которые будут использованы для перемножения (число потоков может быть меньше, чем число строк у первой матрицы).

В функции main сравнить время перемножения больших случайных матриц обычным и многопоточным способом. Получить текущее время можно с помощью методов класса System.

Дополнительное задание

Написать многопоточную задачу о ферзях.

Инструкция

Для успешной работы программы пользователю нужно задать размеры двух матриц таким образом, чтобы они могли быть перемножены (число столбцов первой матрицы и число строк второй должно совпадать). После запуска программы пользователь увидит, на сколько многопоточное умножение матриц выигрывает по времени у однопоточного.

Для запуска дополнительного задания пользователю нужно создать класс-наследник OddNum, чтобы посчитать количество нечетных элементов в векторе.

Тестирование

1. Тест 1 – сравнение времени выполнения.

Кусок кода в main:

UsualMatrix m1 = new UsualMatrix(1000, 1000);  
UsualMatrix m2 = new UsualMatrix(1000, 1000);  
  
double startTime = System.currentTimeMillis();  
  
ParallelMatrixProduct p = new ParallelMatrixProduct(30);  
  
UsualMatrix res1 = p.prod(m1, m2);  
double middleTime = System.currentTimeMillis();  
System.out.println("Многопоточная программа выполнялась " + ((middleTime - startTime) / 1000) + " секунд");  
  
UsualMatrix res2 = m1.product(m2);  
  
double timeSpent = System.currentTimeMillis() - middleTime;  
System.out.println("Однопоточная программа выполнялась " + (timeSpent / 1000) + " секунд");

Вывод в консоль:

Многопоточная программа выполнялась 6.222 секунд

Однопоточная программа выполнялась 24.556 секунд

1. Тест 2 – проверка, что результат многопоточного и однопоточного перемножения одинаковый.

Кусок кода:

UsualMatrix m1 = new UsualMatrix(1000, 1000);  
UsualMatrix m2 = new UsualMatrix(1000, 1000);

ParallelMatrixProduct p = new ParallelMatrixProduct(30);

UsualMatrix res1 = p.prod(m1, m2);

UsualMatrix res2 = m1.product(m2);

System.*out*.println(res1.equals(res2));

Вывод в консоль:

true

1. Тест 3 – тест дополнительного задания.

Кусок кода:

Queens c = new Queens(10);  
long begin = System.*currentTimeMillis*();  
int count1 = c.calcQueenNum(1);  
long end = System.*currentTimeMillis*();  
String resStr1 = "Количество = " + count1 + " Время на " + c.threads.length + " потоке: " + (end - begin) + "ms";  
log.info(resStr1);  
  
begin = System.*currentTimeMillis*();  
int count2 = c.calcQueenNum(4);  
end = System.*currentTimeMillis*();  
String resStr2 = "Количество = " + count2 + " Время на " + c.threads.length + " потоках: " + (end - begin) + "ms";  
log.info(resStr2);

Вывод в консоль:

дек. 05, 2019 1:51:02 AM chess.Queens main

INFO: Количество = 724 Время на 1 потоках: 70ms

дек. 05, 2019 1:51:02 AM chess.Queens main

INFO: Количество = 724 Время на 4 потоках: 25ms

Из чего можно сделать вывод, что многопоточная программа выполняется быстрее