**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное ГОСУДАРСТВЕННОЕ бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и вычислительной техники

Кафедра ИиСП

**Лабораторная работа № 1**

**по дисциплине «Параллельное программирование»**

Выполнил:

студент группы ПС-31

Бочкарев М.А.

Проверил:

Филимонов А. A.

г. Йошкар-Ола

2017

# Постановка задачи

**Найти обратную матрицу для матрицы А**.

Входные данные: произвольная матрица А размерности n х n. Заполнение начальной матрицы реализовать в главном потоке приложения.

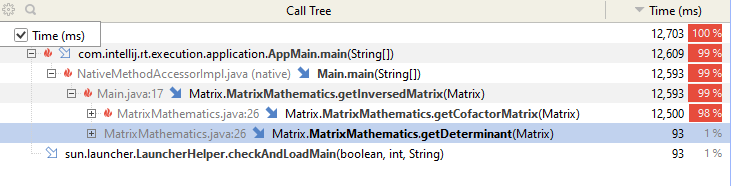
1. Реализовать последовательный вариант программы для указанного варианта.
2. Реализовать параллельный вариант программы. Количество потоков выполнения должно являться входным параметром задачи.
3. После завершения программа должна выдавать время своей работы. Подобрать размеры матриц таким образом, чтобы время работы последовательного варианта составляло не менее одной секунды.
4. Посчитать параметры качества вашей параллельной программы и построить в Excel графики для количества потоков = [1;16] и количестве задействованных ядер = [1;4] (используйте функцию SetAffinityMask).
5. Время выполнения
6. Ускорение
7. Эффективность распараллеливания.

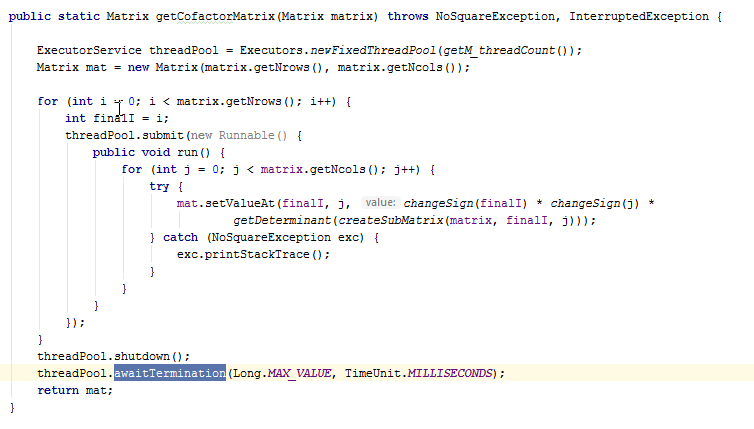
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

# Реализация

Листинг программы: <https://github.com/BochkarevMaxim/PP/tree/master/PP%20Lab%201>

Для создания эффективной параллельной программы необходимо определить участки кода в которых необходимо производить параллельные вычисления.

Для этой цели последовательная версия программы была мной проанализирована с помощью профайлера YourKit Java Profiler. Были получены следующие результаты:

Из лога работы программы видно, что 98% ее работы занимает метод getCofactorMatrix(). Именно его и следует выполнять параллельно.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Листинг параллельного метода getCofactorMatrix(): |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Графики

Так как Java не позволяет установить Affinity Mask для потоков и использует все ядра процессора на свое усмотрение, мною было решено произвести запуски приложения на двух компьютерах с различными процессорами.

Тестируемые конфигурации:

* I3-2120, 8gb RAM, Windows 10
* I3-3120M, 6gb RAM, Windows 8

# Вывод

Таким образом, исходя из полученных данных, можно сделать вывод о том, что при параллельном исполнении программы, время выполнения уменьшается.