|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

по дисциплине:

**Архитектура параллельных вычислительных систем**

|  |  |  |
| --- | --- | --- |
| Студент |  | Абидоков Рашид Ширамбиевич |
| Группа |  | РК6-11М |
| Тип задания |  | лабораторная работа |
| Тема лабораторной работы |  | Применение технологий OpenMP и CUDA |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Абидоков Р. Ш.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Спасенов А. Ю.**

*подпись, дата фамилия, и.о.*

*Москва, 2020 г.*

Оглавление

[Цель выполнения лабораторной работы 3](#_Toc60092232)

[Задание на лабораторную работу 3](#_Toc60092233)

[1. Реализация с помощью OpenMP 4](#_Toc60092234)

[2. Реализация с помощью CUDA 5](#_Toc60092235)

[3. Сравнение времени работы 7](#_Toc60092236)

[Заключение 10](#_Toc60092237)

# Цель выполнения лабораторной работы

Цель выполнения лабораторной работы– получение практических навыков применения технологий OpenMP и CUDA, оценка времени работы программы.

# Задание на лабораторную работу

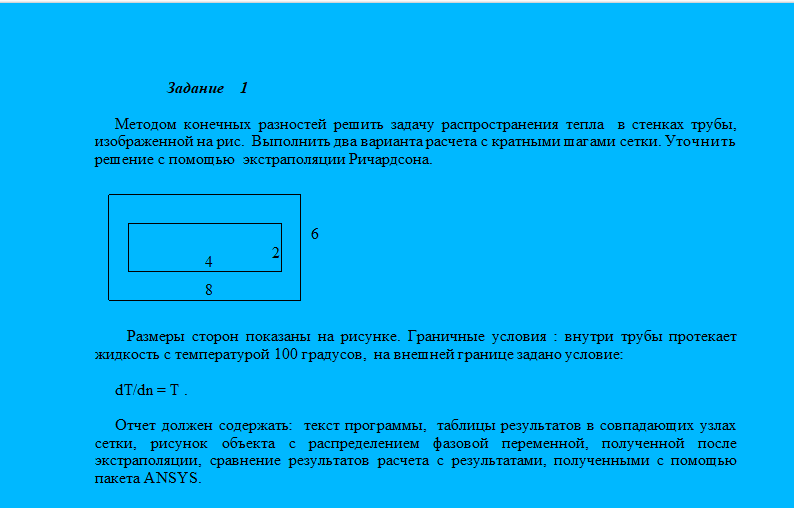


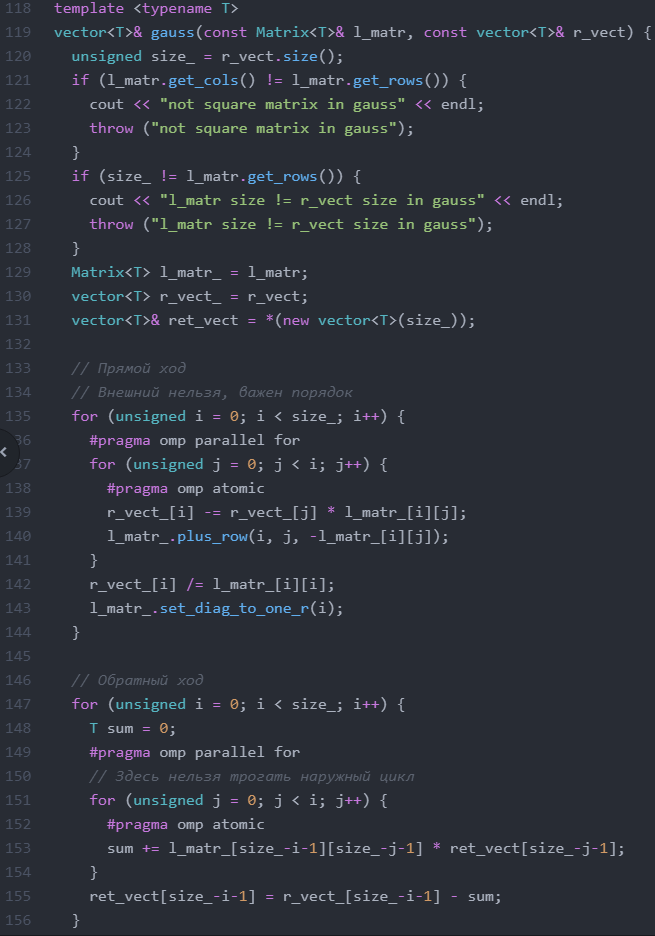
Рис. 1 Исходное задание

Однако для большей наглядности применения методов распараллеливания условие задачи было изменено – требуется решить нестационарную задачу, т.е. получить распределение температуры в зависимости от времени. Начальной температурой пластины считается 20 градусов Цельсия.

# Реализация с помощью OpenMP

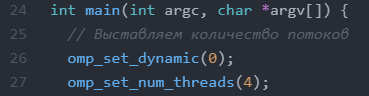
Поскольку практически всё время работы программы занимает решение СЛАУ методом Гаусса, распараллеливанию как в данном пункте, так и при использовании CUDA подвергался именно метод Гаусса. Исходный код реализующей его функции приведен в Листинге 1.

Листинг 1.



В явном виде количество потоков задается в функции main, фрагмент которой приведен в Листинге 2.

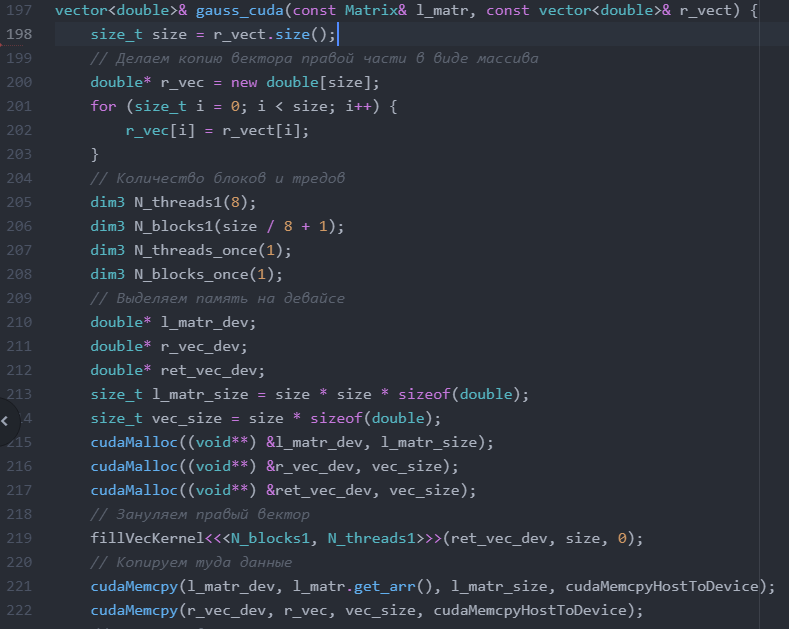
Листинг 2.



# Реализация с помощью CUDA

Поскольку прямой ход метода Гаусса имеет сложность , а обратный, в свою очередь, , при большом количестве узлов (когда распараллеливание является эффективным) наибольшее влияние на время выполнения оказывает именно прямой ход. Исходный код функции, реализующей метод Гаусса, приведен в Листинге 3.

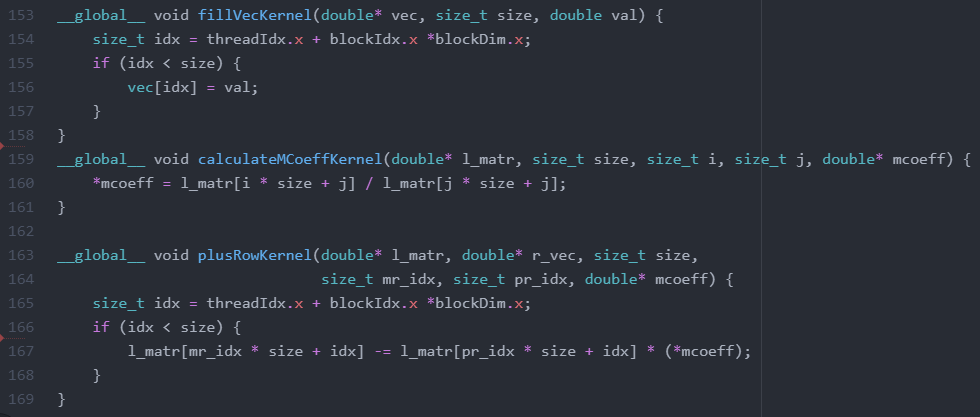
Листинг 3.



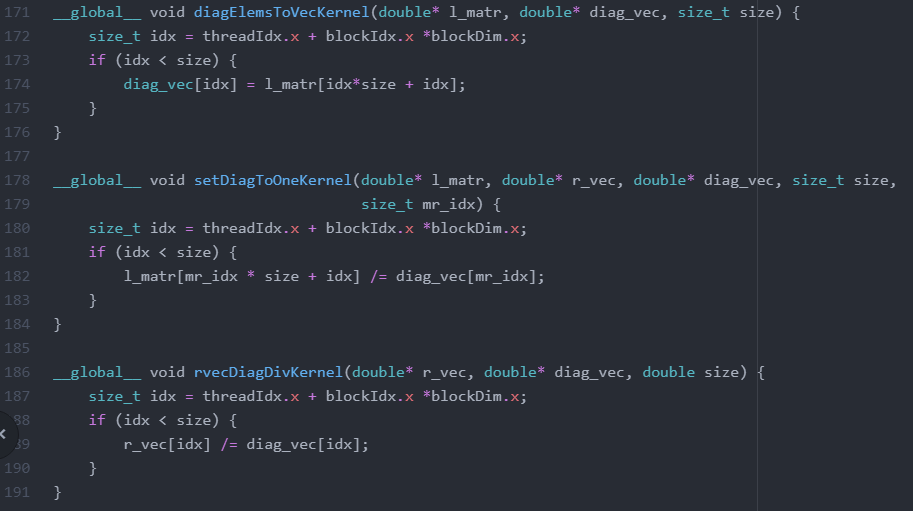


Используемые Kernel-функции в Листингах 4 и 5.

Листинг 4.



Листинг 5.

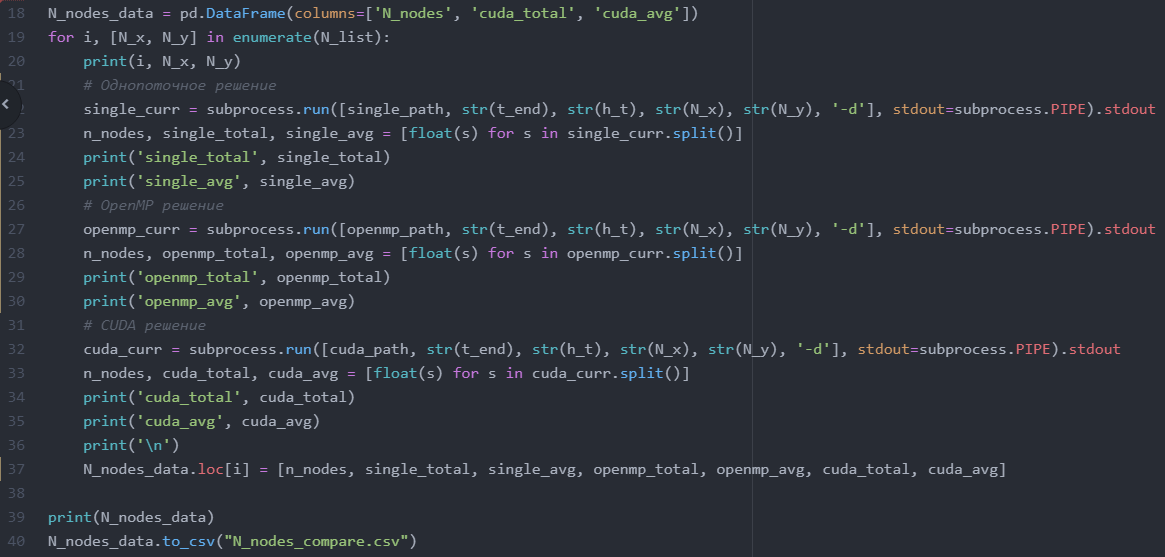


# Сравнение времени работы

Сравнение времени работы однопоточной и многопоточных реализаций проводилось на компьютере с процессором Intel Core i7-6500u, имеющем 2 ядра с 4 потоками, и видеокартой gtx-960m. Для автоматизации использовался скрипт, написанный на языке python, код которого приведен в Листинге 6.

Листинг 6.





Всё время указано в мс, столбцы "шаг" – общее время, деленное на количество шагов интегрирования

Таблица 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N узлов | Однопоточн. всего | Однопоточн.  шаг | OpenMP всего | OpenMP шаг | CUDA всего | CUDA шаг |
| 40 | 31.0 | 0.6 | 299.0 | 6.0 | 130.0 | 2.6 |
| 160 | 1354.0 | 27.08 | 1794.0 | 35.88 | 594.0 | 9.2 |
| 360 | 14783.0 | 295.66 | 10096.0 | 201.92 | 1975.0 | 35.22 |
| 640 | 82487.0 | 1649.74 | 48314.0 | 966.28 | 4955.0 | 92.08 |
| 1000 | 287670.0 | 5753.4 | 179644.0 | 3592.88 | 11382.0 | 214.68 |

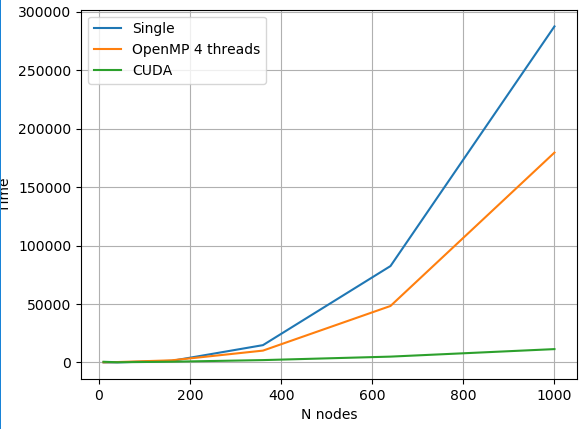


Рис 2.

Более наглядным является график кубического корня времени от количества узлов. Т.к. сложность алгоритма , графики похожи на прямые линии. Видно, что при малом количестве узлов (и, как следствие, малой сложности) параллельные реализации проигрывают в скорости однопоточной, что связано с накладными расходами при распараллеливании.

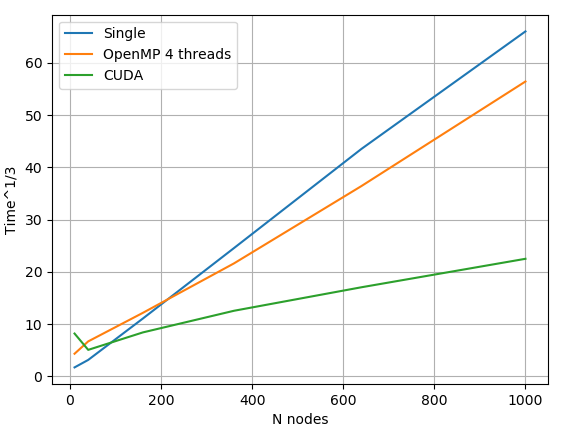


Рис 3.

Графики отношения времени однопоточной реализации к времени многопоточных

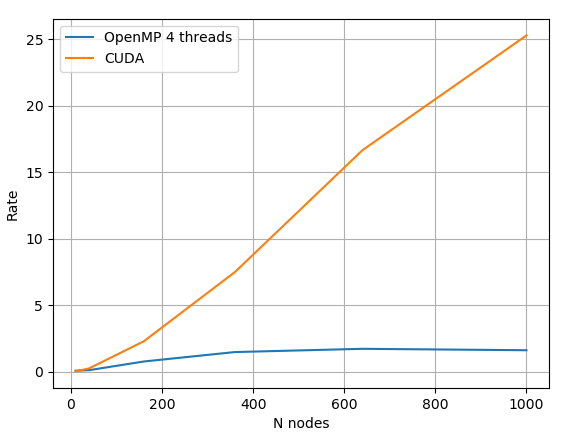


Рис 4.

# Заключение

Было реализовано распараллеливание метода Гаусса с использованием технологий OpenMP и CUDA. Проведено сравнение времени работы однопоточной и многопоточных реализаций, показано, что даже при локальном применении данных технологий прирост производительности является существенным.

При этом в случае, если вычислительная сложность задачи не очень велика, распараллеливание может не только не привести к ускорению работы программы, но даже замедлить её, что связано с возникающими накладными расходами, связанными с управлением памятью и синхронизацией потоков.