Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра вычислительных систем

**ОТЧЕТ**

по практической работе 2

по дисциплине «**Параллельные вычислительные технологии**»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИС-242  «20» мая 2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Никулин М.К./ |
|  |  |  |
| Проверил:  Ассистент кафедры ВС  «20» мая 2024 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | // |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЕ 3](#_Toc167032266)

[ГРАФИК УСКОРЕНИЯ 4](#_Toc167032267)

[ПРИЛОЖЕНИЕ 6](#_Toc167032269)

# **ЗАДАНИЕ**

На базе директив **#pragma omp task** реализовать многопоточный рекурсивный

алгоритм быстрой сортировки (QuickSort). Опорным выбирать центральный элемент

подмассива (функция partition, см. слайды к лекции). При достижении подмассивами

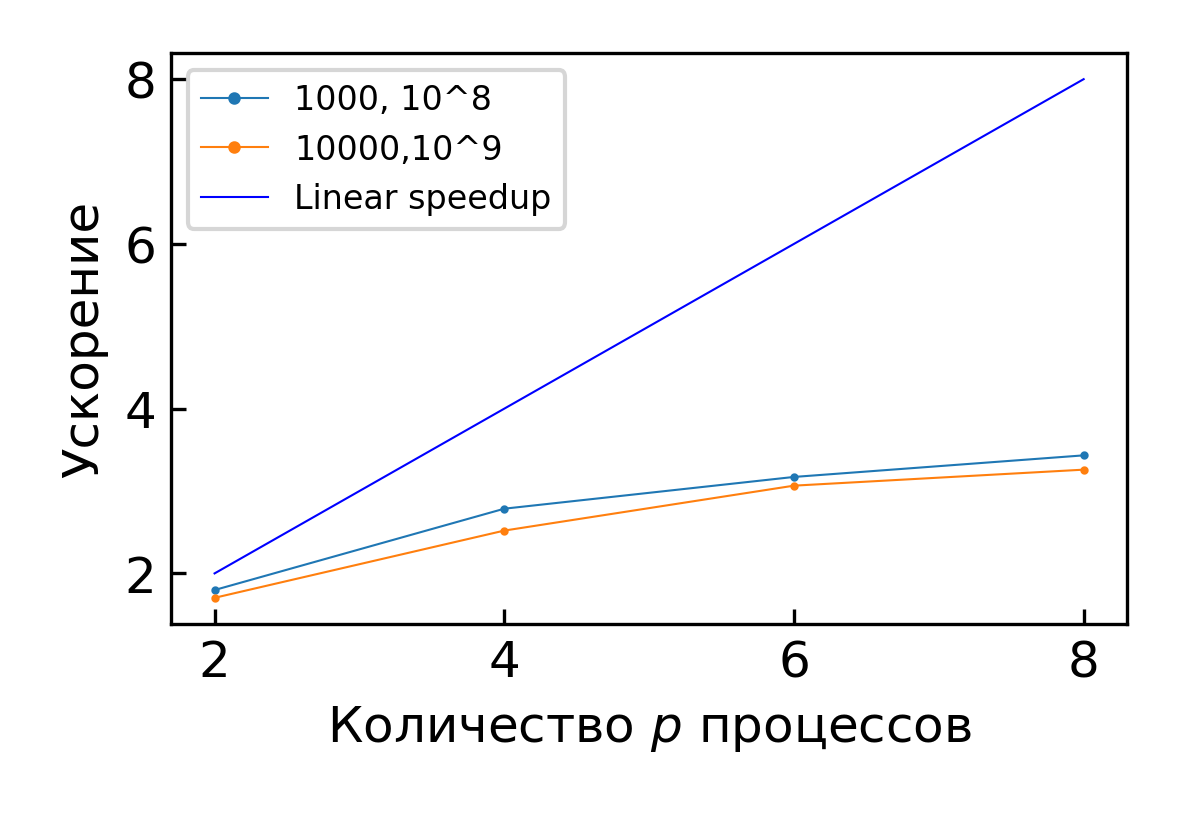
размеров THREASHOLD = 1000 элементов переключаться на последовательную версию

алгоритма.

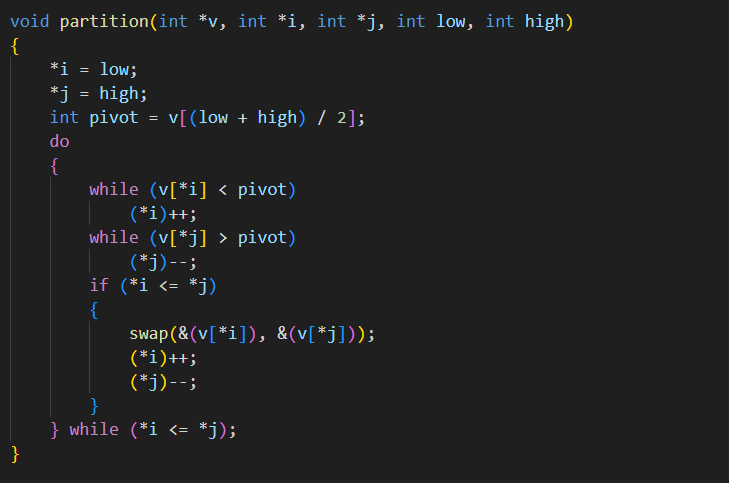
Выполнить анализ масштабируемости алгоритма для различного числа сортируемых

элементов и порогового значения THRESHOLD

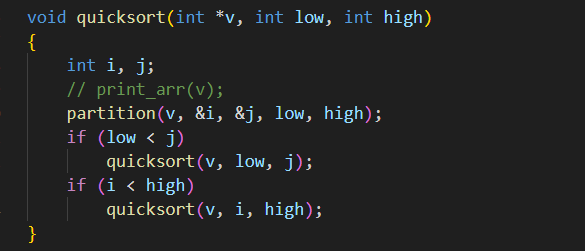
**ГРАФИК УСКОРЕНИЯ**

****

**Описание функцпй**

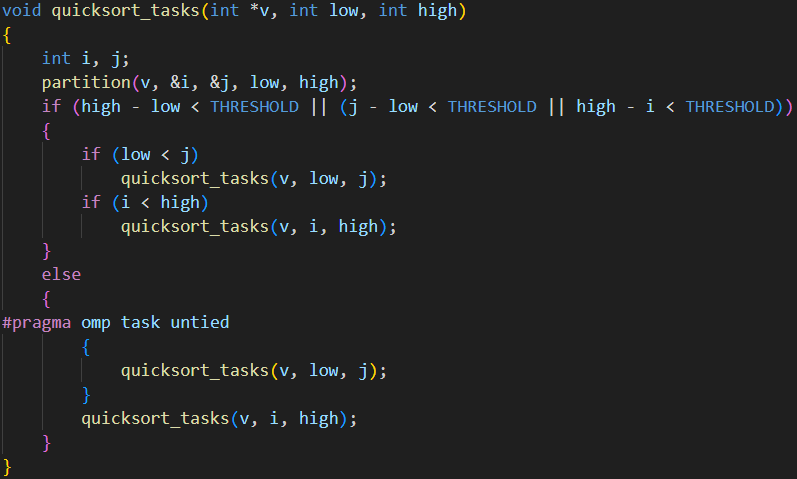
****

Функция partion делит массив на две части, относительно опорного элемента и сортирует 2 подмассива.



Рекурсивная сортировка подмассивов.

**Параллельная программа**

****

Проверяется условие, что ни подмассивы, ни весь массив не превышают граничное значение (THRESHOLD), иначе программа выполняется последовательно (не разумно тратить ресурсы процессора на параллелизм при сортировке массива <= 1000 элементов). Если пороговое значение не достигнуто, то с помощью pragma omp task untied создаём неявную задачу сортировки левого подмассива, которая pushится в дек наименее загруженного потока (потому что untied отвязывает задачу от главного потока), правый подмассив сортируется в главном потоке, т.к. он не загружен в этот момент.

Алгоритм быстрой сортировки наиболее эффективно распараллеливается за счёт параллелизма задач, потому что мы заранее не знаем, сколько будет рекурсивных вызовов, поэтому балансировать загрузку потоков получается, только используя неявные задачи (task).

# **ПРИЛОЖЕНИЕ**