Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Задача №1 по практикуму

Отчёт
О выполненном задании

Выполнил:
студент 423 группы
Латыпов Ш. И.

Москва
2023

Содержание

Описание задачи

Задача состоит в том, что необходимо исследовать проблему масштабируемости при программировании с использованием технологии OpenMP.

В данном варианте рассматривается проблема синхронизации нитей из-за использования лишних барьеров. Для явной демонстрации этой проблемы написаны две программы, в одной используетя оптимальный алгоритм решения задачи, в другой "искусственно"добавлены барьеры, не меняющие логику программы, но замедляющие её работу вследствие накладных расходов на синхронизацию нитей.

Алгоритм программы

Для демонстрации этой проблемы масштабируемости была написана программа, которая в параллельном режиме транспонирует матрицу.

Матрица разбивается на блоки одинакового размера, и каждый процесс на первом шаге сохраняет себе копию этого блока, транспонирует его внутри себя, затем записывает полученный блок в общую матрицу.

В оптимальной версии программы присутствует лишь один барьер, который синхронизирует процессы перед записью полученных блоков в итоговую матрицу. В программе с демонстрацией проблемы лишних барьеров функции синхронизации добавлены после выделения памяти всеми процессами, после копирования блока в локальную память каждого процесса, и после каждого шага транспонирования.

Тестирование программы выполнялось на параллельной вычислительной системе *Polus*: 3 вычислительных узла, 2 десятиядерных процессора IBM POWER8. Использовался компилятор xlc, флаги компиляции -qsmp=omp и -std=c++20.

Запуски проводились через планировщик IBM Spectrum LSF. Файл конфигурации имеет вид, представленный ниже на фотографии. С помощью парамметра BSUB -m polus-c3-ib программа запускалась на третьем узле суперкопьютера. Вычисления проводились для матриц размером 7056×7056, 14112×14112 и 28224×28224 с использованием 1, 4, 16, 36 нитей.

```
#BSUB -m polus-c3-ib
#BSUB -W 00:15
#BSUB -o "main.%J.out"
#BSUB -e "main.%J.err"
#BSUB -R "affinity[core(8)]"
OMP_NUM_THREADS=16 ./main 7056 1764
```

Рис. 1: Файл конфигурации LSF

Работа программы

1. Матрица 7056 × 7056

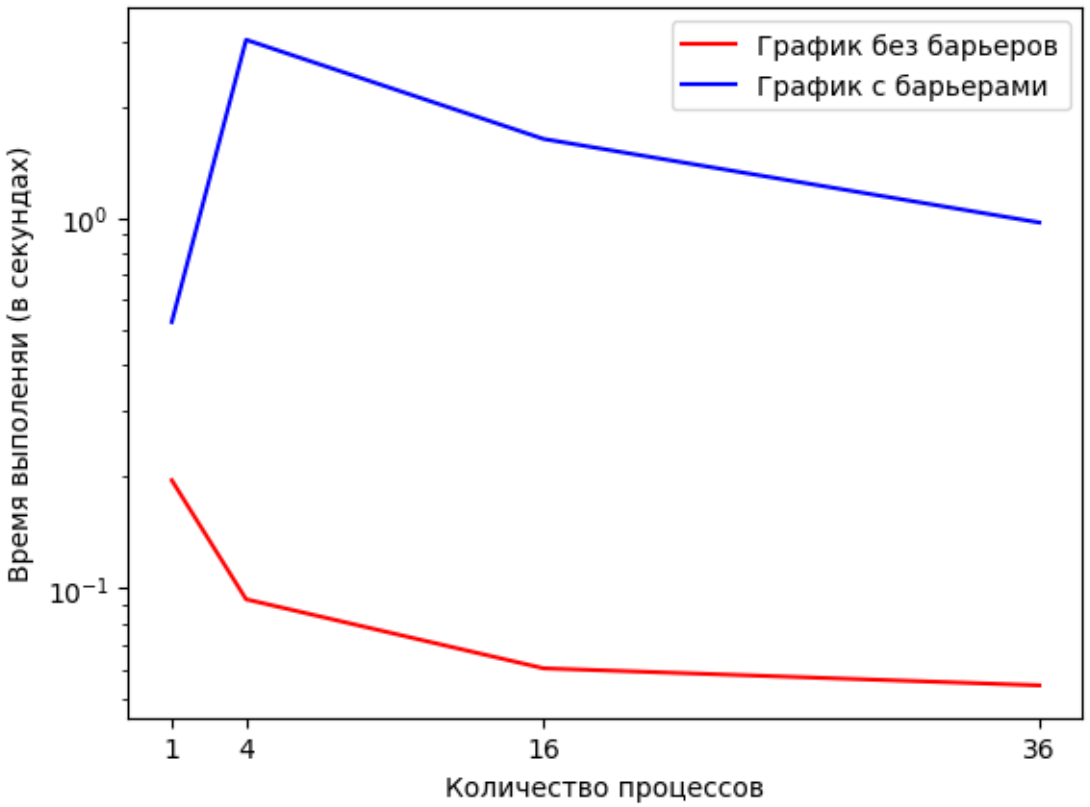

Рис. 2: График времени работы

7056	1	4	16	36
Без барьеров	0,195208	0,092985	0,060500	0,054415
С барьерами	0,523026	3,031753	1,635096	0,971561
Ускорение без барьеров	1	2,09934936	3,22657851	3,58742615
Ускорение с барьерами	1	0,17251621	0,31987507	0,53833619
Разница по времени (%)	167,932897	3160,47481	2602,63802	1685,48181

Рис. 3: Таблица результатов

2. Матрица 14112 × 14112

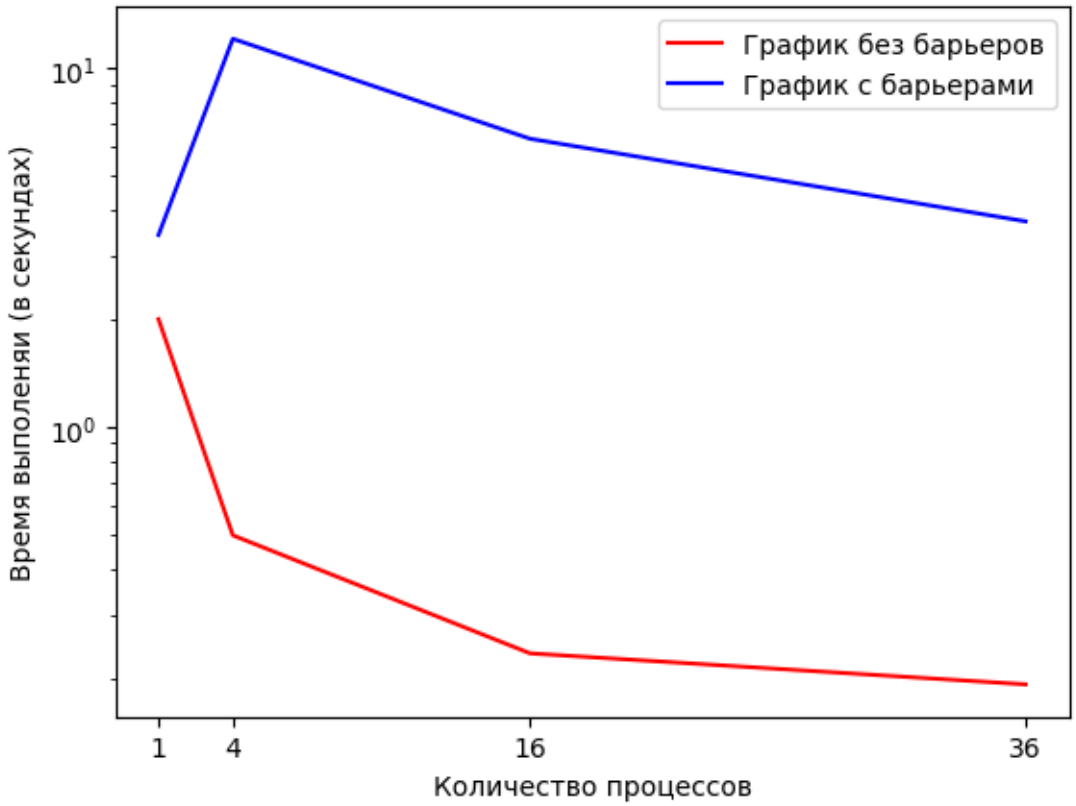

Рис. 4: График времени работы

14112	1	4	16	36
Без барьеров	1,995495	0,499165	0,234210	0,192000
С барьерами	3,422895	12,040556	6,344402	3,736901
Ускорение без барьеров	1	3,99766911	8,52010802	10,3932005
Ускорение с барьерами	1	0,28428048	0,5395142	0,91597155
Разница по времени (%)	71,5311668	2312,14189	2608,85189	1846,3026

Рис. 5: Таблица результатов

3. Матрица 28224 × 28224

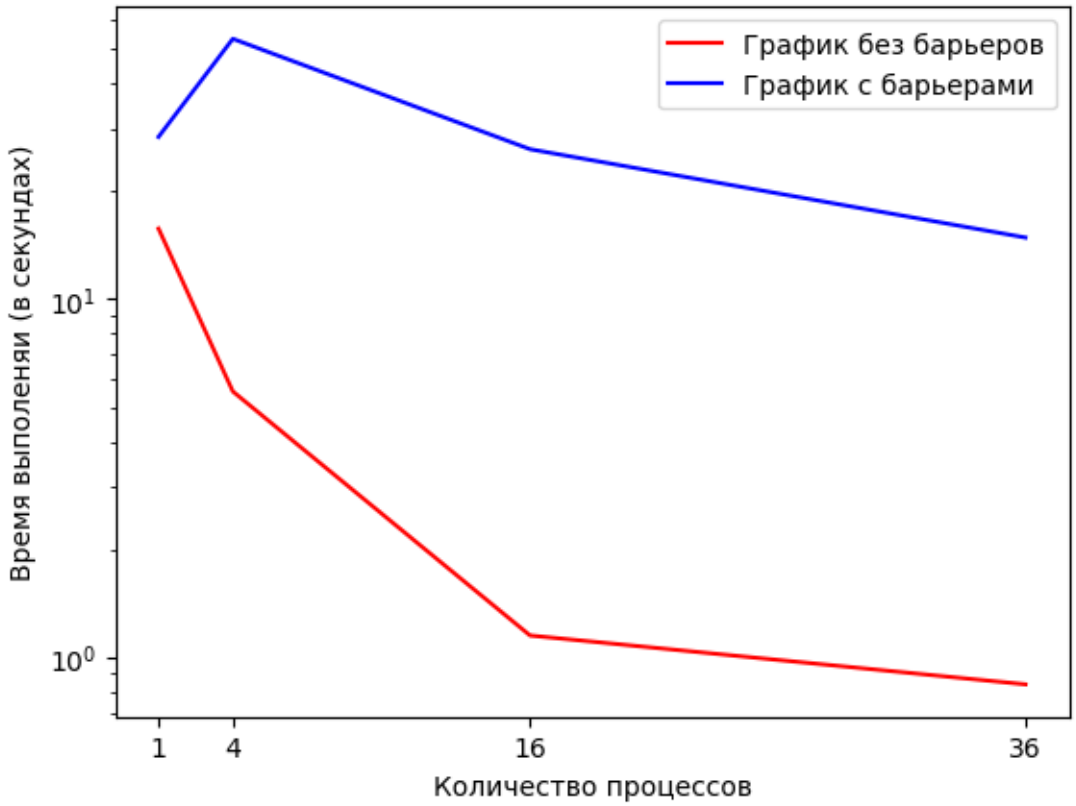

Рис. 6: График времени работы

28224	1	4	16	36
Без барьеров	15,750094	5,524481	1,149327	0,839336
С барьерами	28,402606	53,382262	26,238026	14,863091
Ускорение без барьеров	1	2,85096372	13,7037591	18,7649446
Ускорение с барьерами	1	0,53206073	1,08249782	1,91094877
Разница по времени (%)	80,3329326	866,285644	2182,90442	1670,81538

Рис. 7: Таблица результатов

Выводы

Программа дйствительно сильно замедляют свою работу из-за наличия большого количетсва лишних барьеров. Как можно заметить, даже при выполнении операции всего на 1 процессе, лишние синхронизирующие барьеры замедляют программу более чем в 2 раза, не говоря уже о параллельном режиме, где замедление заметно вплоть до 30 раз.