МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«Программирование многопоточных приложений. POSIX Threads.»

Студента 2 курса, группы 21209

Панас Матвея Алексеевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Мичуров Михаил Антонович

Новосибирск 2023 г.

Цель

Освоить разработку многопоточных программ с использованием POSIX Threads API. Познакомиться с задачей динамического распределения работы между процессорами.

Задание

Есть список неделимых заданий, каждое из которых может быть выполнено независимо от другого. Задания могут иметь различный вычислительный вес, т.е. требовать при одних и тех же вычислительных ресурсах различного времени для выполнения. Считается, что этот вес нельзя узнать, пока задание не выполнено. После того, как все задания из списка выполнены, появляется новый список заданий. Необходимо организовать параллельную обработку заданий на нескольких компьютерах. Количество заданий существенно превосходит количество процессоров. Программа не должна зависеть от числа компьютеров. За компьютеры будем принимать MPI процессы.

Для решения задачи балансировки вычислительной нагрузки между процессами создаем на каждом из процессов по еще одному дополнительному потоку receiver_thread, который будет заниматься приемом запросов о передаче части задач. На главном потоке процесса будут проходить все вычисления. При завершении вычислений главный поток отправляет запросы о добавке на другие процессы. Дополнительный созданный поток на фоне вычислений главного потока принимает запросы о добавке от других процессов и отправляет им определенное количество задач. При получении добавки главным потоком начинаются вычисления снова проводятся, а дополнительный поток так и продолжает принимать запросы и отправлять задания. Когда все задания закончатся, потоки завершают свою работу.

Команды компиляции и запуска:

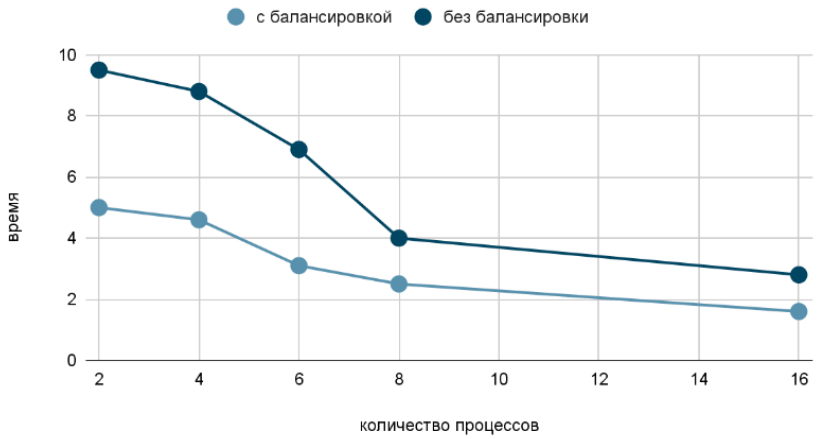
```
mpicc -std=gnu99 -O3 -Wpedantic -Wall -Werror task5.c -o task
```

```
mpiexec -machinefile $PBS_NODEFILE -np $MPI_NP ./task
```

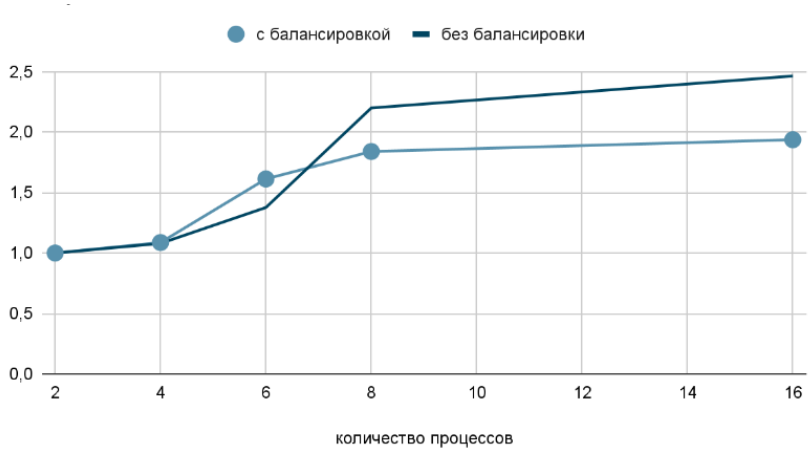
Исходный код: https://github.com/phestrix/OPP/blob/main/lab5/task5.c

Замеры работы программы

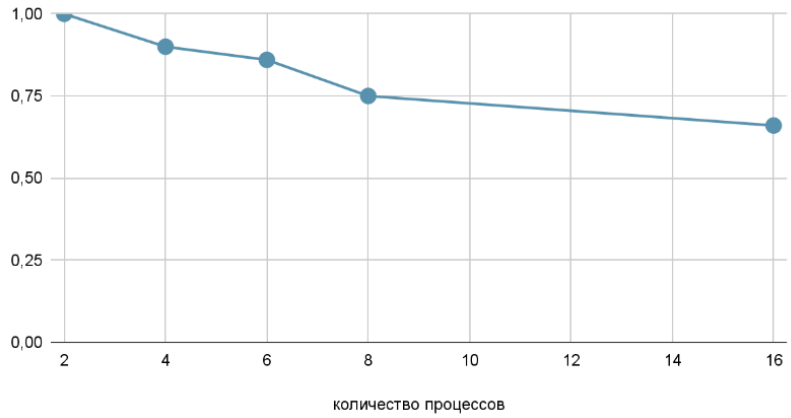
Время работы программы:



Ускорение:



Эффективность:



Заключение

Я освоил разработку многопоточных программ с использованием POSIX Threads API. Также познакомился с задачей динамического распределения нагрузки между процессорами. Распределение нагрузки с помощью балансировки дает ощутимый прирост производительности.