МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Параллельные алгоритмы»

Тема: Основы работы с процессами и потоками

Студентка гр. 9304 _____ Аксёнова Е.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2022

Цель работы.

Ознакомиться с работой с процессами и потоками в языке программирования C++.

Задание.

Выполнить поэлементное сложение 2х матриц M*N

Входные данные: Две матрицы вводятся из файла или генерируются.

Результат: Сумма двух матриц записывается в файл

1. Выполнить задачу, разбив её на 3 процесса. Выбрать механизм обмена данными между процессами.
    a. Процесс 1: заполняет данными входные матрицы (читает из файла или генерирует их некоторым образом).
    b. Процесс 2: выполняет сложение
    c. Процесс 3: выводит результат
2. Выполнить задачу, разбив её на 3 потока.
    a. Поток 1: заполняет данными входные матрицы (читает из файла или генерирует их некоторым образом).
    b. Поток 2: выполняет сложение
    c. Поток 3: выводит результат
3. Разбить сложение на P потоков.

Исследовать зависимость между количеством потоков, размерами входных данных и параметрами целевой вычислительной системы.

Выполнение работы.

Сложение матриц с помощью 3 процессов.

С помощью fork() порождаются процессы-потомки, которые почти идентичны потоку-родителю (не наследуются pid процесса, израсходованное время ЦП и т.п.). Весь код после fork() выполняется дважды, как в процессе-потомке, так и в процессе-родителе (чтобы не было дублирования, используется switch по pid процесса).

Для передачи данными между процессами использовалась разделяемая память, так как он самый быстрый (не приводит к переключению контекста между процессом и ядром).

Для управления разделяемой памятью использовались:

1. shmget(ключ, размер сегмента, идентификатор) - возвращает идентификатор разделяемому сегменту памяти, соответствующий значению аргумента key.
2. shmat(id разделяемой памяти, место появления разделяемой памяти в памяти процесса, режим чтения/записи) - подключает сегмент общей памяти идентификатором shmid к адресному пространству вызывающего процесса.
3. snmdt(адрес памяти) - отключение разделяемой память от процесса.

Сложение матриц с помощью 3 потоков.

Создание потоков происходит с помощью конструктора thread(), в который передается функция и необходимые параметры.

Синхронизация между процессами происходит с помощью join(), который блокирует вызывающий поток до завершения потока, представленного экземпляром.

Сложение матриц с помощью N процессов.

Для сложения с помощью N потоков используется массив потоков. Каждый поток суммирует i-ую ячейку каждой матрицы. Каждый из потоков присоединяется с помощью join().

Исследование зависимости между количеством потоком, размерами входных данных и параметрами вычислительной системы.

В таблице 1 представлено сравнение размера входных данных и времени вычисления при одинаковом количестве потоков - 10 и поэлементном сложении, т.е. число потоков - N x M.

Таблица 1 - Сравнение размера входных данных и времени вычисления

Время вычисления	Размер входных данных
0.134978	100 x 50
13.007267	1000 x 500
1298.894570	10000 x 5000
2609.200688	100000 x 10000

В таблице 2 представлено сравнение размера входных данных и времени вычисления при одинаковом количестве потоков - 10 и постолбцовом сложении, т.е. число потоков - N.

Таблица 2 - Сравнение размера входных данных и времени вычисления

Время вычисления	Размер входных данных
0.006043	100 x 50
0.053381	1000 x 500
2.233303	10000 x 5000
6.591545	100000 x 10000

Выводы.

В ходе выполнения лабораторной работы было написанно 3 программы на языке программирования C++:

1. Для сложения матриц с помощью 3 процессов
2. Для сложения матриц с помощью 3 потоков
3. Для сложения матриц с помощью N потоков

Также было установлено, что с ростом размера входных данных, время выполнения расчетов растет.

С увеличением числа потоков, все зависит от того, насколько сложные операции выполняет поток. Иногда время на создание потока превышает время на выполняемые в нем операции.

Сложение посимвольно в многопоточной среде проигрывает в производительности постолбцовому сложению, как раз из-за затрат на создание процесса.