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

Студент гр. 0304 _____ Нагибин И.С.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучить основы работы с процессами и потоками.

Здание.

Лабораторная состоит из 3х подзадач, которые выполняют одинаковую задачу с использованием процессов или потоков.

Выполнить умножение 2х матриц.

Входные матрицы вводятся из файла (или генерируются).

Результат записывается в файл.

1.1. Выполнить задачу, разбив её на 3 процесса. Выбрать механизм обмена данными между процессами.

Процесс 1: заполняет данными входные матрицы (читает из файла или генерирует их некоторым образом).

Процесс 2: выполняет умножение

Процесс 3: выводит результат

1.2.1. Аналогично 1.1, используя потоки (std::threads)

1.2.2. Разбить умножение на P потоков (“наивным” способом по по строкам-столбцам).

В отчёте:

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

Выполнение работы.

1. Умножение матриц с помощью 3 процессов. tmp.cpp

Для передачи данных между процессами была использована библиотека Boost, а конкретно её раздел Boos.Interprocess. Библиотека позволяет создавать именованные участки памяти и обращаться к ним по названию из других процессов.

Для успешного доступа к памяти из других процессов необходимо создать специальный аллокатор, также для удобства был создан тип

MyVectorVector, который является вектором векторов с необходимым аллокатором.

```
    typedef boost::interprocess::allocator<int,ë
boost::interprocess::managed_shared_memory::segment_manag
er> ShmemAllocator;
    typedef boost::interprocess::vector<int,
ShmemAllocator> MyVector;
    typedef boost::interprocess::allocator<MyVector,
boost::interprocess::managed_shared_memory::segment_manag
er> VectorAllocator;
    typedef boost::interprocess::vector<MyVector,
VectorAllocator> MyVectorVector;
```

Далее в функции main средствами библиотеки создаются все необходимые поля для работы с матрицами.

```
   segment.construct<MyVectorVector>("Matrix1")
(vector_alloc_inst);
    segment.construct<MyVectorVector>("Matrix2")
(vector_alloc_inst);
    segment.construct<bool>("Ready flag")(false);
```

Для доступа к памяти из другого процесса необходимо обратиться к сегмента памяти и найти именнованное поле.

```
auto *matrix1 =
segment.find<MyVectorVector>("Matrix1").first;
auto *matrix2 =
segment.find<MyVectorVector>("Matrix2").first;
```

Таким образом, происходит доступ к матрицами в каждом процессе. Сначала считывание из файла, потом умножение, а после вывод в консоль. Для считывания матрицы из файла, её преобразования, умножения и вывода написаны отдельные функции.

2. Умножение матриц с помощью 3 потоков. lb1.cpp

Для умножения матриц с помощью 3 потоков была создана функция threadsTest. В ней инициализируются необходимые потоки.

```
std::thread matrixReading(readMatrixes);
matrixReading.join();
std::thread matrixMult(multiplyVectorMatrix);
matrixMult.join();
printMatrix(GLOB_matrix1);
```

3. Выполнение параллельного умножения матриц. lb1.cpp

Для умножения матриц с использованием *P* потоков на основе программы *thread_proc* была создана функция paralletMatrixMultiply. В ней поток P потоков по очереди вычисляют свои столбцы итоговой матрицы

4. Измерения производительности.

Для измерения производительности была использована утилита *time*, так как для корректного сравнения скорости работы потоков и процессов необходимо учитывать время запуска и остановки программ. Измерялось реальное, системное и пользовательское время. Результаты приведены в табл. 1-2.

Таблица 1. Умножение матриц с помощью 3 потоков.

Размер матриц	Real time	User time	System time
5x5	0m0,002s	0m0,002s	0m0,000s
10x10	0m0,002s	0m0,000s	0m0,002s
20x20	0m0,002s	0m0,000s	0m0,001s
50x50	0m0,005s	0m0,005s	0m0,000s

Таблица 2. Умножение матриц с помощью 3 процессов.

Размер матриц	Real time	User time	System time
5x5	0m0,003s	0m0,001s	0m0,002s
10x10	0m0,003s	0m0,003s	0m0,000s
20x20	0m0,004s	0m0,002s	0m0,002s
50x50	0m0,009s	0m0,011s	0m0,000s

```
Есть 8 потоков
Исследуем зависимость количества потоков от времени перемножения квадр
атной матрицы 720x720
КОЛ-ВО ПОТОКОВ: 1 ||| ИТОГОВОЕ ВРЕМЯ: 4.34717 с.
КОЛ-ВО ПОТОКОВ: 2 ||| ИТОГОВОЕ ВРЕМЯ: 2.05042 с.
КОЛ-ВО ПОТОКОВ: 3 ||| ИТОГОВОЕ ВРЕМЯ: 1.43001 с.
КОЛ-ВО ПОТОКОВ: 4 ||| ИТОГОВОЕ ВРЕМЯ: 1.14551 с.
КОЛ-ВО ПОТОКОВ: 5 ||| ИТОГОВОЕ ВРЕМЯ: 1.31832 с.
КОЛ-ВО ПОТОКОВ: 6 ||| ИТОГОВОЕ ВРЕМЯ: 1.15704 с.
КОЛ-ВО ПОТОКОВ: 7 ||| ИТОГОВОЕ ВРЕМЯ: 1.12385 с.
КОЛ-ВО ПОТОКОВ: 8 ||| ИТОГОВОЕ ВРЕМЯ: 1.09334 с.
КОЛ-ВО ПОТОКОВ: 9 ||| ИТОГОВОЕ ВРЕМЯ: 1.14308 с.
КОЛ-ВО ПОТОКОВ: 10 ||| ИТОГОВОЕ ВРЕМЯ: 1.15244 с.
КОЛ-ВО ПОТОКОВ: 11 ||| ИТОГОВОЕ ВРЕМЯ: 1.17126 с.
КОЛ-ВО ПОТОКОВ: 12 ||| ИТОГОВОЕ ВРЕМЯ: 1.139 с.
КОЛ-ВО ПОТОКОВ: 13 ||| ИТОГОВОЕ ВРЕМЯ: 1.14333 с.
КОЛ-ВО ПОТОКОВ: 14 ||| ИТОГОВОЕ ВРЕМЯ: 1.15992 с.
КОЛ-ВО ПОТОКОВ: 15 ||| ИТОГОВОЕ ВРЕМЯ: 1.15136 с.
```

Рисунок 1. Результат работы программы lb1.cpp

```
Есть 8 потоков
Исследуем зависимость количества потоков от времени перемножения квадр
атной матрицы 1000x1000
КОЛ-ВО ПОТОКОВ: 1 ||| ИТОГОВОЕ ВРЕМЯ: 12.5493 с.
КОЛ-ВО ПОТОКОВ: 2 ||| ИТОГОВОЕ ВРЕМЯ: 5.73517 с.
КОЛ-ВО ПОТОКОВ: 3 ||| ИТОГОВОЕ ВРЕМЯ: 4.27588 с.
КОЛ-ВО ПОТОКОВ: 4 ||| ИТОГОВОЕ ВРЕМЯ: 3.21357 с.
КОЛ-ВО ПОТОКОВ: 5 ||| ИТОГОВОЕ ВРЕМЯ: 3.84387 с.
КОЛ-ВО ПОТОКОВ: 6 ||| ИТОГОВОЕ ВРЕМЯ: 3.34502 с.
КОЛ-ВО ПОТОКОВ: 7 ||| ИТОГОВОЕ ВРЕМЯ: 3.51539 с.
КОЛ-ВО ПОТОКОВ: 8 ||| ИТОГОВОЕ ВРЕМЯ: 3.52066 с.
КОЛ-ВО ПОТОКОВ: 9 ||| ИТОГОВОЕ ВРЕМЯ: 3.59744 с.
КОЛ-ВО ПОТОКОВ: 10 ||| ИТОГОВОЕ ВРЕМЯ: 3.56315 с.
КОЛ-ВО ПОТОКОВ: 11 ||| ИТОГОВОЕ ВРЕМЯ: 3.7502 с.
КОЛ-ВО ПОТОКОВ: 12 ||| ИТОГОВОЕ ВРЕМЯ: 3.65562 с.
КОЛ-ВО ПОТОКОВ: 13 ||| ИТОГОВОЕ ВРЕМЯ: 3.70261 с.
КОЛ-ВО ПОТОКОВ: 14 ||| ИТОГОВОЕ ВРЕМЯ: 3.59313 с.
КОЛ-ВО ПОТОКОВ: 15 ||| ИТОГОВОЕ ВРЕМЯ: 3.66121 с.
```

Рисунок 2. Результат работы программы lb1.cpp

Выводы.

Были изучены основы работы с процессами и потоками. В ходе работы было измерено время работы программы для перемножения матриц. Таким образом, при перемножении матриц тремя потоками и тремя процессами не было найдено значительных отличий в скорости работы программы. Исходя из полученных данных можно сказать, что на обслуживание процессов и потоков необходимо примерно одинаковое время с точностью до тысячных секунды.

При сравнении количества поток для перемножения разных матриц можно выявить, что линейное увеличение количества потоков необязательно уменьшает время исполнения программы. Это связано с тем, что необходимо время для обслуживания потоков. Так, например, при использовании 1-4 потоков время будет уменьшаться, 5 и 6 вызывают неожиданный скачок времени перемножения матриц. 7 и 8 ожидаемое замедление перемножения матриц, связанное с отсутствием потоков для работы операционной системы. Дальнейшее увеличение количества потоков не приносит прироста производительности, так как физических потоков больше нет.