## Лабораторная работа №3

## РАЗРАБОТКА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА БАЗЕ ТЕХНОЛОГИИ MPI

## Рабочее задание

1. Разработать параллельную программу на языке C++ в соответствии с индивидуальным вариантом задания (таблица 2) с использованием технологии MPI. Ввод массивов исходных данных производить из текстового файла, формируемого вручную или с помощью специально написанной программы с использованием датчика случайных чисел. Ввод размерности (сложности) задачи и другие скалярные данные вводить по запросу программы или через командную строку. Обеспечить вывод исходных данных и вывод полученного результата на экран. Производить раздельно измерение времени выполнения параллельного участка кода и времени выполнения всего вычислительного алгоритма.
2. Произвести запуск разработанной программы в режимах А, В, С c занесением результатов в соответствующую таблицу.

Режим А: запуск на одном узле с 1 процессом (последовательный запуск);

Режим В: запуск на одном узле с несколькими параллельными процессами;

Режим С: запуск на двух узлах с несколькими параллельными процессами;

Число процессов в режимах B и C выбрать равным числу задействованных логических ядер.

Для каждого режима произвести запуск программы не менее 5 раз, затем определить среднее значение времени вычислений. Повторить измерения для других значений размерности (сложности) задачи (не менее 5 значений).

1. Рассчитать коэффициент ускорения вычислений, коэффициент эффективности и стоимость вычислений. Результаты занести в таблицу 1.
2. Определить долю параллельных операций в разработанном программном коде. Рассчитать максимальное значение коэффициента ускорения вычислений по закону Амдала для разработанной программы.

Рекомендованное число процессов в режимах В и С – общее число логических ядер на локальных машинах

Таблица 1 – Результаты вычислительного эксперимента

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Размерность  (сложность)  задачи | Среднее время вычислений | | | | Показатели эффективности вычислений | | |
| A | | B | C | S | E | C |
| … |  |  | |  |  |  |  |
| … |  |  | |  |  |  |  |

Таблица 2 – Варианты заданий

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Отсортировать элементы каждой строки двумерного массива NxM по возрастанию |
| 2 | Отсортировать элементы каждого столбца двумерного массива NxM по возрастанию |
| 3 | Переставить элементы каждой строки двумерного массива NxM в обратном порядке |
| 4 | Переставить элементы каждого столбца двумерного NxM массива в обратном порядке |
| 5 | Транспонировать матрицу NxM |
| 6 | Найти среднее арифметическое каждой строки двумерного массива |
| 7 | Найти максимальный элемент одномерного массива |
| 8 | Определить долю четных чисел в одномерном числовом массиве |
| 9 | Произведение вектора на матрицу NxM |
| 10 | Произведение матрицы NxM на вектор |
| 11 | Скалярное произведение двух векторов |
| 12 | Рассчитать площадь круга радиуса R методом Монте-карло |
| 13 | Вычислить определенный интеграл заданной функции методом Монте-карло |
| 14 | Вычислить определенный интеграл заданной функции методом трапеций |
| 15 | Определить частоту использования заданного слова в тексте |

## Приложение

Для задания размерности (сложности) задачи предусмотрен ввод желаемого количества элементов или числа интервалов в главном процессе (процесс с идентификатором 0). Главный процесс, в свою очередь, передаёт это значение всем остальным процессам с помощью функции:

**MPI\_Bcast(buffer, count, datatype, root, comm)**

Где:

- buffer – адрес (указатель) исходного буфера, содержащего передаваемые данные;

- count - количество записей в буфере (целое);

- datatype - тип данных в буфере (MPI\_CHAR, MPI\_INT, MPI\_LONG, MPI\_DOUBLE и др.);

- root - номер главного процесса (целое);

- comm - коммуникатор (дескриптор).

Для синхронизации параллельных процессов используются функции MPI\_Reduce и MPI\_Barrier.

Функция редукции MPI\_Reduce служит для объединения частных результатов, сформированных отдельными параллельными процессами. Формат:

**MPI\_Reduce (sendbuf, recvbuf, count, datatype, op, root, comm)**

- sendbuf - адрес исходного буфера, содержащего передаваемые данные;

- recvbuf - адрес принимающего буфера (используется только корневым процессом);

- count - количество элементов в исходном буфере (целое);

- datatype - тип данных элементов исходного буфера (дескриптор);

- op - операция редукции (MPI\_MIN, MPI\_MAX, MPI\_SUM, MPI\_PROD);

- root - номер главного(принимающего) процесса (целое);

- comm - коммуникатор (дескриптор).

Задаётся соответствующая операция редукции.

Функция принимает значения sendbuf от всех процессов, совершает над ними операцию редукции, возвращает результат в главный процесс (переменная recvbuf)

Функция MPI\_Barrier служит для блокирования (остановки) всех процессов заданного коммуникатора до тех пор, пока каждый из процессов не достигнет этой функции.

Формат:

**MPI\_Barrier(comm)**

- comm - коммуникатор.

Дополнительную информацию по использованию функций MPI можно получить на сайте <https://parallel.ru/> или здесь: <https://www.opennet.ru/docs/RUS/MPI_intro/>