Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
_____Кафедра Суперкомпьютеров и Квантовой Информатики_____

Отчёт № 5
Реализация квантового преобразования Фурье с помощью программы на основе MPI

Работу выполнила
Домрачева Д. А.

Москва 2019

Постановка задачи и формат данных

Задача:

Разработать параллельную программу с использованием MPI и OpenMP, реализующую алгоритм квантового преобразования Фурье, протестировать на системе Ломоносов 2. Тип данных – complex<double>.

Формат командной строки:

Режим генерации:

<число кубитов n> <файл вывода вектора b> <файл вывода вектора a>

Алгоритм и хранение данных

По аргументам командной строки происходит определение режима работы программы:

- Режим генерации вектора: каждый процесс генерирует свою часть вектора с помощью функции rand_r. Для избежания повтора данных в качестве seed используется текущее время, умноженное на ранг процесса плюс один.
- Режим считывания вектора: из указанного входного файла происходит параллельное считывание средствами MPI (каждому процессу задается смещение в файле, начиная с которого он считывает свою порцию данных).
- Если указана необходимость вывода полученного вектора в файл, то вывод производится средствами MPI по аналогии с чтением вектора из файла.

После генерации/чтения вектора выполняется вычисление выходного вектора. В программе определяются индексы элементов исходного вектора, необходимых для вычисления текущего элемента b, затем происходит определение необходимости обмена данными между процессами. Если обмен необходим, то он симметричен, поэтому два процесса выполнят пересылку своего элемента и прием элемента от другого процесса. Если необходимости в обмене нет (искомый элемент находится в памяти текущего процесса), то вычисление произойдет в обычном порядке. Вычисление выполняется последовательно n раз, при этом кубит, по которому проводится преобразование изменяется от 1 до n.

На каждом процессе хранится лишь части векторов a и b (кроме случая, когда запущен только один процесс).

Тестирование и результаты

Количество кубитов	Количество вычислительных узлов	Количество используемых потоков на узел	Время работы программы (сек)
28	1	1	970.656
		2	542.122
		4	311.298
		8	271.263
	2	1	516.998
		2	267.121
		4	153.353
		8	124.662

	4	1	267.866
		2	176.002
		4	101.365
		8	71.404

Вывод

Программа начинает работать быстрее при увеличении числа процессов, но с увеличением числа нитей для каждого количества используемых вычислительных узлов происходит падение скорости снижения времени из-за увеличения количества пересылок в программе.