Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Практикум на ЭВМ: 6 семестр.

Отчёт № 2.

Анализ параллельной программы на MPI, реализующей однокубитное квантовое преобразование

Работу выполнил

Федоров В. В.

Москва 2021

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм однокубитного квантового преобразования с использованием MPI, проанализировать зависимость времени выполнения программы от числа кубитов, номера преобразуемого кубита и числа нитей на системе Polus.

Формат командной строки: Первый аргумент — единственный символ, означающий один из двух режимов работы:

- `g` — вектор состояния инициализируется случайными значениями. Дальнейшие аргументы командной строки: <число кубитов n> <номер преобразовываемого кубита k> <имя файла для вывода времени выполнения>
- `f` — вектор состояния считывается из файла, вектор-результат также записывается в файл. Дальнейшие аргументы командной строки: <номер преобразовываемого кубита k> <имя файла для считывания вектора> <имя файла для записи вектора>
    - Вектор записывается в файл в бинарном формате — сначала идут 4 бита числа кубитов n, затем — 2^n пар восьмибайтовых чисел с плавающей точкой, соответствующих действительной и мнимой части соответствующего элемента вектора.

Описание программы

В предположении что число процессов $p=2^s$, где $s\in\mathbb{N}, s\leq\frac{n}{2}$ разделим вектор состояния на 2^s+1 блоков одного размера, тогда каждому процессу достается 2 блока. В таком случае либо блоки можно будет разделить между процессами так, чтобы элементы двух блоков конкретного процесса были попарно соседними по k-му кубиту, либо соседние элементы уже находятся в одном и том же блоке, и тогда блоки можно распределить между процессорами как угодно, но для простоты будем выдавать процессам пары соседних блоков. Рассмотрим пример с n = 4 и s = 2:

Номер блока	Номер элемента
0	0000
	0001
1	0010
	0011
2	0100
	0101
3	0110
	0111
4	1000
	1001
5	1010
	1011
6	1100
	1101

7	1110
	1111

k\pr	0	1	2	3
1	0,4	1,5	2,6	3,7
2	0,2	1,3	4,6	5,7
3	0,1	2,3	4,5	6,7
4	0,1	2,3	4,5	6,7

Таким образом, составив пользовательский тип MPI и заполнив массив сдвигов, можно распределить вектор состояния между процессами.

Тестирование

Программа тестировалась при помощи режима «f» на 5 заранее сгенерированных файлах с n = 16 для всех k от 1 до 16 и для всех количеств процессов от 1 до 16, равных степени двойки.

Результаты выполнения

Результаты для k = 1

Кол-во кубитов	Кол-во нитей	Время работы, с	Ускорение	Эффективность
25	1	3,025620	1,000000	1,000000
	2	1,503760	2,012036	1,006018
	4	0,748487	4,042315	1,010579
	8	0,377164	8,022028	1,002753
	16	0,268534	11,267177	0,704199
26	1	6,033910	1,000000	1,000000
	2	2,982360	2,023200	1,011600
	4	1,479220	4,079116	1,019779
	8	0,748852	8,057547	1,007193
	16	0,441528	13,665974	0,854123
27	1	11,958900	1,000000	1,000000
	2	5,933640	2,015441	1,007720
	4	3,053950	3,915879	0,978970
	8	1,616530	7,397883	0,924735
	16	0,792449	15,091066	0,943192

Результаты для k = 13

Кол-во кубитов	Кол-во нитей	Время работы, с	Ускорение	Эффективность
25	1	3,003670	1,000000	1,000000
	2	1,506010	1,994456	0,997228

	4	0,745098	4,031242	1,007810
	8	0,375484	7,999462	0,999933
	16	0,243091	12,356155	0,772260
26	1	5,958410	1,000000	1,000000
	2	2,961370	2,012045	1,006023
	4	1,546090	3,853857	0,963464
	8	0,867919	6,865168	0,858146
	16	0,414547	14,373304	0,898331
27	1	11,917500	1,000000	1,000000
	2	6,144880	1,939419	0,969710
	4	3,179890	3,747771	0,936943
	8	1,642660	7,255001	0,906875
	16	0,896486	13,293571	0,830840

Результаты для k = n

Кол-во кубитов	Кол-во нитей	Время работы, с	Ускорение	Эффективность
25	1	2,981840	1,000000	1,000000
	2	1,506620	1,979159	0,989579
	4	0,743982	4,007946	1,001987
	8	0,389813	7,649411	0,956176
	16	0,196743	15,156016	0,947251
26	1	6,009670	1,000000	1,000000
	2	3,035680	1,979678	0,989839
	4	1,534610	3,916089	0,979022
	8	0,791027	7,597301	0,949663
	16	0,450249	13,347437	0,834215
27	1	11,913400	1,000000	1,000000
	2	5,997440	1,986414	0,993207
	4	3,116760	3,822367	0,955592
	8	1,606160	7,417318	0,927165
	16	0,797980	14,929447	0,933090

Основные выводы.

В большинстве случаев эффективность работы близка к 1, поскольку по условию задачи в замеры времени включалась лишь та часть кода, в которой процессы не взаимодействовали друг с другом и работали независимо. Наименьшую эффективность программа показала на k = 13.