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

Практикум на ЭВМ

Отчёт № 2

Параллельная программа на MPI, реализующая однокубитное квантовое преобразование

Работу выполнил
Чепурнов А. В.

Москва 2019

Постановка задачи и формат данных

1) Разработать схему распределенного хранения данных и параллельный алгоритм для реализации однокубитного квантового преобразования на кластерной системе.
2) Реализовать параллельную программу на C++ с использованием MPI, которая выполняет однокубитное квантовое преобразование над вектором состояний длины 2^n, где n – количество кубитов, по указанному номеру кубита k. Для работы с комплексными числами использовать стандартную библиотеку шаблонов.
3) Протестировать программу на системе Polus. В качестве теста использовать преобразование Адамара по номеру кубита:
   a) Который соответствует номеру в списке группы плюс 1
   b) 1
   c) n

Начальное состояние вектора генерируется случайным образом и нормируется (тоже параллельно).

Формат командной строки: <Число кубитов *n*> <Номер кубита *k*> [<имя файла исходного вектора>] [<имя файла полученного вектора>]
Формат файла-вектора: Вектор представляется в виде бинарного файла следующего формата:

Тип	*Значение*	*Описание*
Число типа int	n – натуральное число	Число кубитов
Массив чисел типа complex<double>	2^n – комплексных чисел	Элементы вектора

Описание алгоритма

Однокубитная операция над комплексным входным вектором $\{a_i\}$ размерности 2^n задается двумя параметрами: комплексной матрицей $\{u_{ij}\}$ размера 2x2 и числом k от 1 до n (номер кубита, по которому проводится операция). Такая операция преобразует вектор $\{a_i\}$ в $\{b_i\}$ размерности 2^n, где все элементы вычисляются по следующей формуле:

$$b_{i_1 i_2 \dots i_k \dots i_n} = \sum_{j_k=0}^{1} u_{i_k j_k} a_{i_1 i_2 \dots j_k \dots i_n} = u_{i_k 0} a_{i_1 i_2 \dots 0_k \dots i_n} + u_{i_k 1} a_{i_1 i_2 \dots 1_k \dots i_n}$$

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

Преобразование Адамара задается следующей матрицей:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Аппаратное обеспечение: Исследования проводились на вычислительном комплексе IBM Polus.

Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция MPI_Wtime().

Анализ ускорения: Ускорение, получаемое при использовании параллельного алгоритма для p процессов, высчитывалось как отношение времени выполнения программы без распараллеливания к времени параллельного выполнения программы.

Результаты выполнения

Количество кубитов (n)	Количество процессов	Время работы (сек)			Ускорение		
		k = 1	k = 13	k = n	k = 1	k = 13	k = n
25	1	3,668287	3,74194	3,617166	1	1	1
	2	1,738048	1,938538	2,40423	2,110579	1,93029	1,504501
	4	0,888508	0,953233	0,928816	4,128592	3,925525	3,894384
	8	0,632259	0,541721	0,424204	5,801874	6,907504	8,526949
26	1	7,182884	7,1183	7,149479	1	1	1
	2	3,668784	3,775942	3,867757	1,957838	1,885172	1,848482
	4	1,918187	3,51985	1,957058	3,744621	2,02233	3,653177
	8	1,034416	1,066542	0,983245	6,943903	6,674186	7,27131
27	1	14,41311	17,53888	16,06032	1	1	1
	2	7,879647	8,281645	7,227281	1,829157	2,117801	2,22218
	4	4,70703	3,764529	3,695418	3,062039	4,658984	4,346009
	8	1,975873	1,788797	1,952147	7,294552	9,804846	8,227003
28	1	32,4515	29,39813	28,88402	1	1	1
	2	14,79652	13,65161	13,96279	2,193185	2,153455	2,068643
	4	10,00601	9,000058	7,627221	3,243201	3,266437	3,786965
	8	4,103183	3,960069	4,129057	7,90886	7,42364	6,995307