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

Практикум на ЭВМ

Отчёт № 1

Параллельная программа на OpenMP, реализующая однокубитное квантовое преобразование

Работу выполнил
Чепурнов А. В.

Москва 2019

Постановка задачи и формат данных

1) Реализовать параллельную программу на C++ с использованием OpenMP, 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.
2) Определить максимальное количество кубитов, для которых возможна работа программы на системе Polus. Выполнить теоретический расчет и проверить его экспериментально.
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.

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 omp_get_wtime().

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

Результаты выполнения

Количество кубитов (n)	Количество нитей	Время работы (сек)			Ускорение		
		k = 1	k = 13	k = n	k = 1	k = 13	k = n
20	1	0,102394	0,112211	0,111905	1	1	1
20	2	0,0609302	0,0687141	0,0784312	1,6805131	1,633013	1,426792
20	4	0,0700737	0,0895678	0,0769511	1,4612329	1,252805	1,454235
20	8	0,119248	0,119231	0,0919334	0,8586642	0,941123	1,21724
24	1	1,65515	1,64981	1,65927	1	1	1
24	2	0,980038	1,02679	1,01497	1,6888630	1,606765	1,634797
24	4	0,768599	0,73215	0,681765	2,1534636	2,253377	2,433786
24	8	0,83507	0,979701	0,639323	1,9820494	1,683993	2,595355
28	1	27,2458	27,1541	26,7541	1	1	1
28	2	16,403	16,4615	17,9908	1,6610254	1,649552	1,487099
28	4	14,2189	14,9035	13,5485	1,9161679	1,821995	1,974691
28	8	15,0742	14,5989	12,4576	1,8074458	1,86001	2,147613
30	1	111,568	104,729	108,283	1	1	1
30	2	67,2744	61,6917	68,2649	1,6584020	1,697619	1,586218
30	4	46,5785	49,161	53,2195	2,3952682	2,130327	2,034649
30	8	37,9913	48,1652	30,9121	2,9366723	2,174371	3,502933

Основные выводы

Теоретически в оперативной памяти IBM Polus объёмом 256 Гбайт может находиться один вектор размерности 2^{34} (одно комплексное число занимает 16 байт) или 2 вектора размерности 2^{33}. На практике при запуске программы с n > 30 система возвращала ошибку выделения памяти.

Распараллеливание ускоряет выполнение программы, однако эффективность этого ускорения оказалась низкой. Скорее всего это связано с большими накладными расходами OpenMP.