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	0,120763	0,209777	0,210239	1	1	1
	2	0,053365	0,103794	0,120547	2,262958	2,0211	1,744043
	4	0,139883	0,119823	0,131597	0,863319	1,750723	1,597596
	8	0,110672	0,124045	0,133593	1,091182	1,691143	1,57373
24	1	1,533961	1,631469	1,70056	1	1	1
	2	0,800141	0,95075	1,134122	1,917113	1,715981	1,499451
	4	0,619112	0,634571	0,771134	2,47768	2,570978	2,205272
	8	0,380299	0,264412	0,463577	4,033561	6,170171	3,668344
28	1	25,87947	28,32331	30,38038	1	1	1
	2	17,71262	16,74308	17,26878	1,461075	1,691643	1,759266
	4	11,49104	11,35999	9,98362	2,252143	2,493251	3,043022
	8	7,0688	5,34561	7,46391	3,661084	5,298424	4,070304
30	1	120,3835	114,5307	105,3514	1	1	1
	2	70,4821	59,8814	66,4787	1,708001	1,912626	1,584739
	4	46,89035	42,55462	42,12026	2,567341	2,691381	2,501205
	8	28,34057	24,43183	21,11211	4,247744	4,687766	4,990093

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.