Московский государственный университет
Кафедра суперкомпьютеров и квантовой информатики

Отчет по первому практическому заданию. Однокубитное преобразование вектора-состояния

Выполнил: Плужников Иван, студент 323 группы

Москва 2021

Задание

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. Описание однокубитного преобразования дано ниже в разделе методические рекомендации. Для работы с комплексными числами возможно использование стандартной библиотеки 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 должно генерироваться случайным образом. Заполнить таблицу и построить график зависимости ускорения параллельной программы от числа процессоров для каждого из случаев a)-c):

Описание алгоритма

Однокубитная операция задается двумя параметрами: комплексной матрицей размера 2*x*2 и числом от 1 до *n* (данный параметр обозначает номер кубита, по которому проводится операция).

Итак, дана комплексная матрица:

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

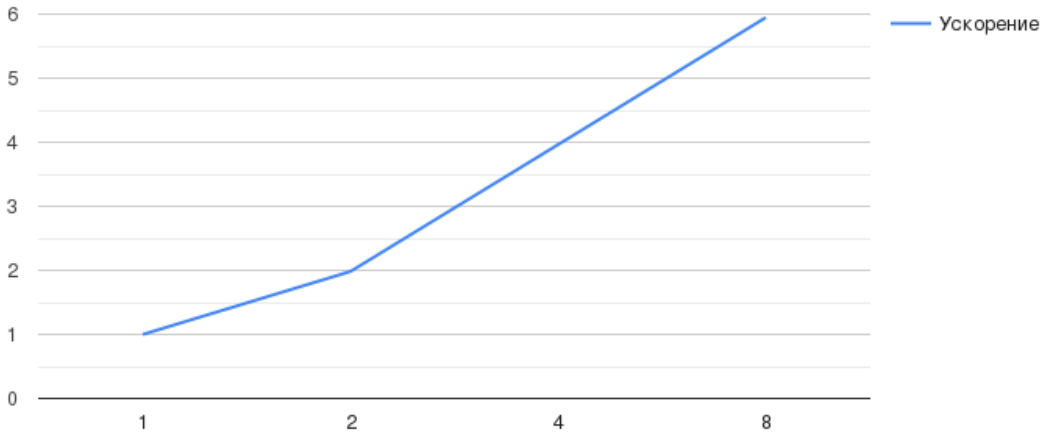
и *k* - номер индекса от 1 до *n* (номер кубита).

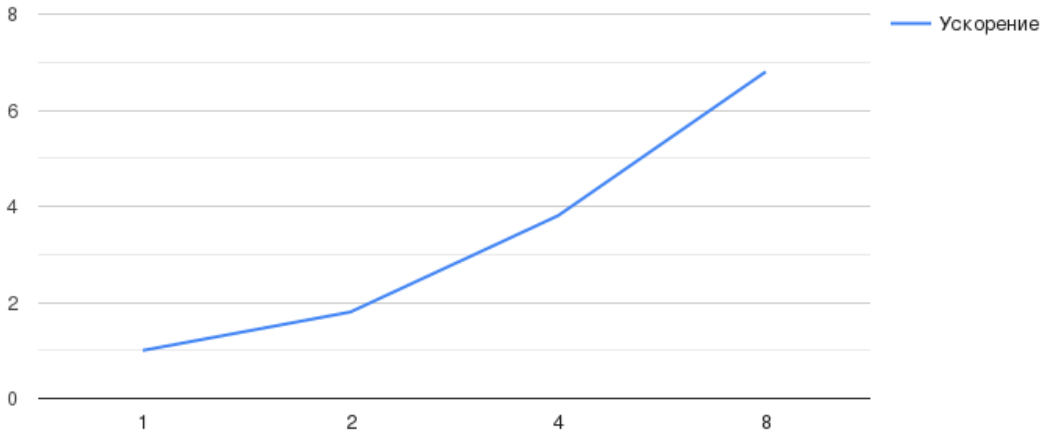
Такая операция преобразует вектор $\{a_{i_1 i_2 \ldots i_n}\}$ в $\{b_{i_1 i_2 \ldots i_n}\}$, где все 2^n элементов нового вектора вычисляются по следующей формуле:

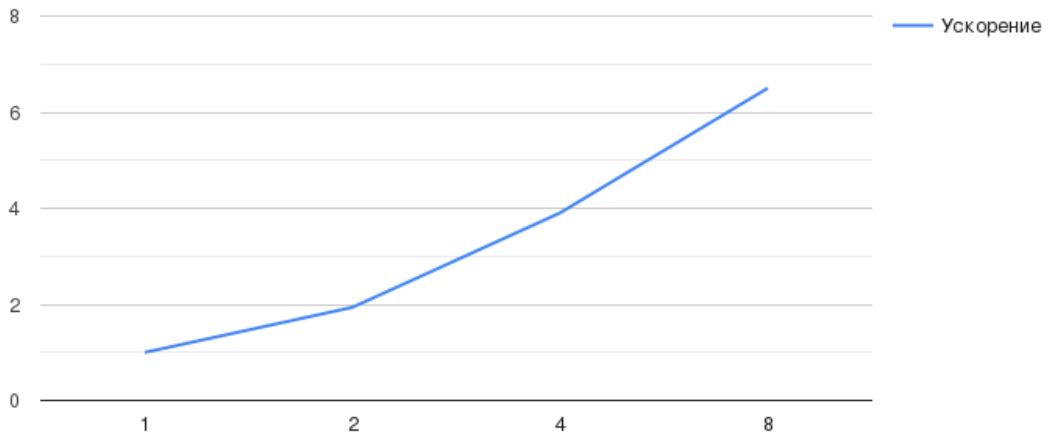
$$b_{i_1 i_2 \ldots i_k \ldots i_n} = \sum_{j_k=0}^{1} u_{i_k j_k} a_{i_1 i_2 \ldots j_k \ldots i_n} = u_{i_k 0} a_{i_1 i_2 \ldots 0_k \ldots i_n} + u_{i_k 1} a_{i_1 i_2 \ldots 1_k \ldots i_n}$$

Количество кубитов	Количество процессоров	Время работы программы(сек)			Ускорение		
		k =1	k = 13	k = n	k = 1	k = 13	k = n
20	1	0.093749	0.093747	0.092983	1	1	1
	2	0.047281	0.046958	0.047197	1,9828	1,996401039	1,970104032
	4	0.023980	0.026129	0.024177	3,909466222	3,587852578	3,845927948
	8	0.018742	0.017482	0.016043	5,002080888	5,36248713	5,795861123
	160	0.007175	0.007114	0.007632	13,084507	13,1408451	12,1833071
24	1	1.495206	1.568541	1.494971	1	1	1
	2	0.761997	0.752969	0.783435	1,962220324	1,68	1,91025641
	4	0.387888	0.395535	0.382917	3,92105263	3,80487805	3,90416461
	8	0.251466	0.248947	0.253769	5,945956909	6,300702559	5,891070225
	160	0.057123	0.058982	0.061346	28,2641509	26,593553966	24,4918033
28	1	24.901079	24.637257	24.216557	1	1	1
	2	12.183924	12.397169	12.461979	2,05785124	1,98387097	1,9516129
	4	6.200570	6.263630	6.191769	4,01612903	3,9047619	3,96721311
	8	3.507420	3.288214	3.486420	7,11428571	7,6875	7,11764706
	160	0.692365	0.80408	0.714187	36,0869565	30,640305691	34,0985915
максимальное	1	105.996422	95.797493	95.897324	1	1	1
	2	51.216545	49.733566	49.900890	2,05882353	1,93877551	1,93877551
	4	26.745071	25.223854	24.811956	3,963213334	4,13043478	3,95833333
	8	14.689852	13.178022	12.992264	7,215622186	7,30769231	7,91666667
	160	5.221394	6.961027	5.972296	20,3653846	13,761976932	16,2372881

K = 1, 13, n:


6
5
4
3
2
1
0
1
2
4
8
Ускорение


8
6
4
2
0
1
2
4
8
Ускорение



Вычисления программы проводились на системе IBM Polus

Для оценки времени выполнения однокубитного преобразования использовалась функция gettimeofday(), возвращающая системное время в микросекундах

Ускорение выполнения многопоточной программы = отношение времени выполнения однопоточного преобразования к времени выполнения многопоточного однокубитного преобразования

Выводы

При росте числа кубитов наблюдается быстрый рост времени выполнения преобразования. Увеличение числа процессоров уменьшало время выполнения программы. Однако, номер преобразуемого кубита не повлиял на время выполнения преобразования. Максиамльный размер вектора, при котором не возникало ошибки - 2^{30} элементов.