Московский государственный университет
Кафедра суперкомпьютеров и квантовой информатики

Отчет по второму практическому заданию.

Плужников Иван, 323

Москва 2021

Задание

1. 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. 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.

Описание алгоритма

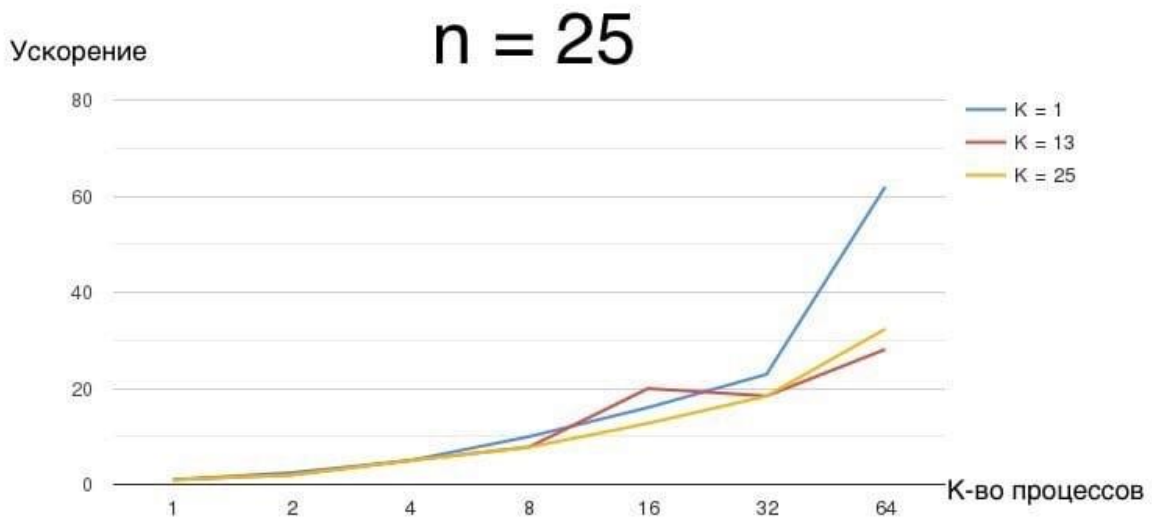
Однокубитная операция задается двумя параметрами: комплексной матрицей размера 2*x*2 и числом от 1 до *n* (данный параметр обозначает номер кубита, по которому проводится операция).

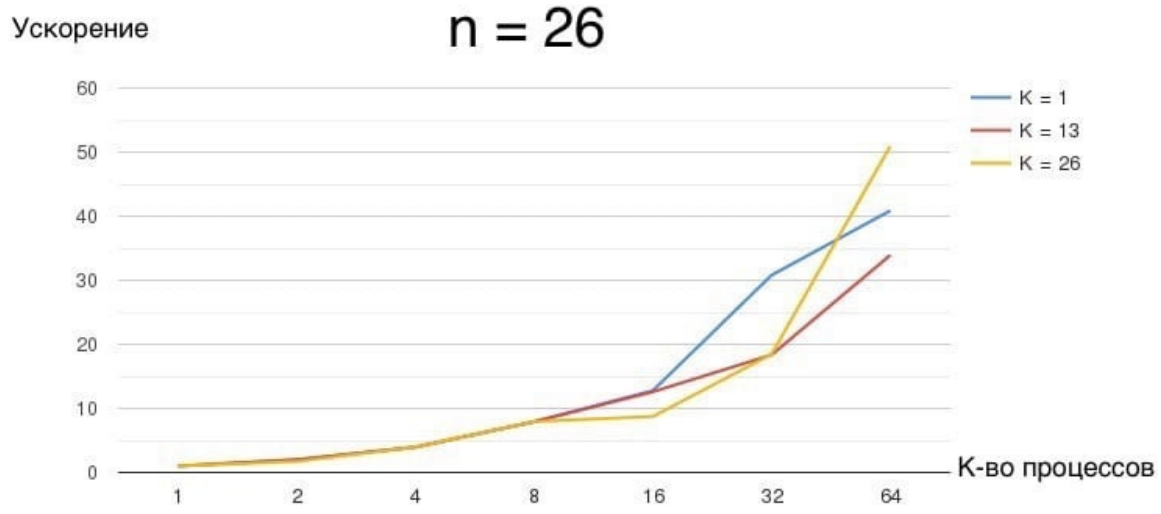
Итак, дана комплексная матрица:

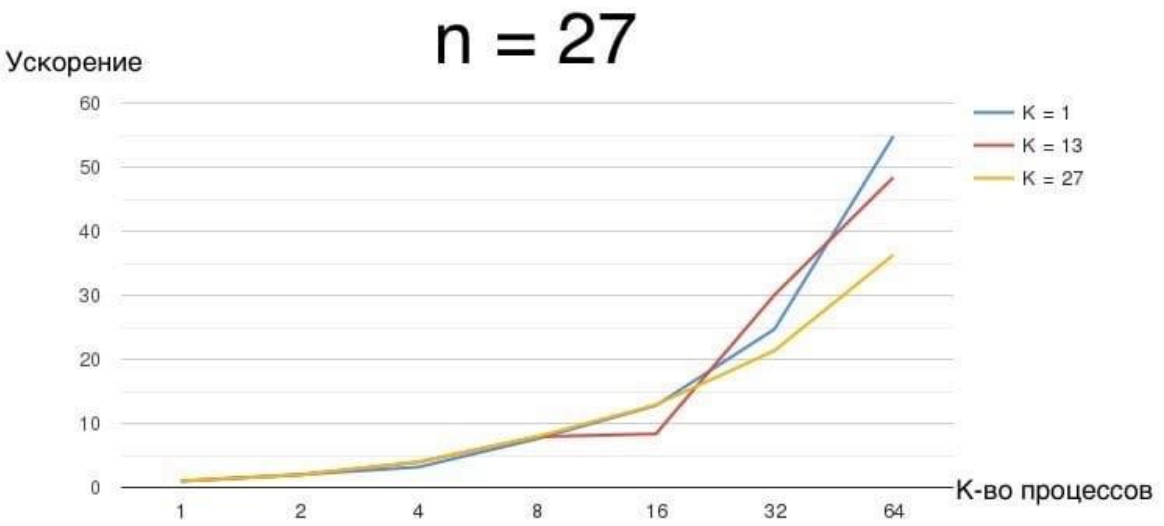
и *k* - номер индекса от 1 до *n* (номер кубита).

Такая операция преобразует вектор в , где все 2^n элементов нового вектора вычисляются по следующей формуле:

Количество кубитов	Количество MPI процессов	Время работы программы(сек)			УСКОРЕНИЕ		
		K =1	K = 13	k = n	1	13	n
25	1	3.9051	3.10304	3.10034	1	1	1
	2	1.58929	1.55745	1.55647	2,4571	1,9924	1,9919
	4	0.77879	0.7781	0.777356	5,0143	5,0188	5,0132
	8	0.39163	0.397777	0.39238	9,9714	7,801	7,751
	16	0.243975	0.195631	0.243738	16,0061	19,9616	12,72
	32	0.170308	0.124488	0.168452	22,9296	18,4209	18,404
	64	0.0630382	0.110386	0.09583	61,9482	28,1108	32,352
26	1	6.2148	6.20987	6.20477	1	1	1
	2	3.11228	3.11939	3.6058	1,9969	1,9907	1,7208
	4	1.5627	1.56257	1.5591	3,977	3,957	3,9797
	8	0.782229	0.782485	0.780756	7,945	7,925	7,9471
	16	0.485852	0.452893	0.709768	12,7915	12,5915	8,742
	32	0.201761	0.338331	0.336172	30,8028	18,3544	18,457
	64	0.152079	0.183095	0.121852	40,8656	33,9161	50,92
27	1	12.4259	12.4103	12.405	1	1	1
	2	6.22521	6.2439	6.21907	1,9961	1,993	1,991
	4	3.89908	3.1193	3.1238	3,1869	3,9786	3,9711
	8	1.95683	1.56716	1.5617	6,35	7,919	7,9433
	16	0.974516	1.48431	0.96127	12,7508	8,361	12,905
	32	0.502525	0.412549	0.58047	24,7269	30,082	21,371
	64	0.2267	0.256452	0.34182	54,8121	48,3923	36,291




Ускорение
n = 26
60
50
40
30
20
10
0
1
2
4
8
16
32
64
K = 1
K = 13
K = 26
K-во процессов


Ускорение
n = 27
60
50
40
30
20
10
0
1
2
4
8
16
32
64
K = 1
K = 13
K = 27
K-во процессов

Вычисления программы проводились на системе IBM Polus.

Для оценки времени выполнения однокубитного преобразования

использовалась функция MPI_Wtime

Ускорением параллельного алгоритма называют отношение времени выполнения лучшего последовательного алгоритмам к времени выполнения параллельного алгоритма.

Метод параллелизма: 1) Производится считывание из файла (каждый процесс считывает свой участок исходя из номера) или генерация вектора (на каждом процессе генерируется свой участок, после этого пересылает на 0 процесс сумму модулей квадратов амплитуд, на 0 процессе считается их сумма и рассылается всем процессам для нормировки их участков вектора).

2) После происходит преобразование, далее тестирование или запись из каждого процесса своего участка.

Выводы

При росте числа кубитов наблюдается экспоненциальный рост времени выполнения преобразования. Ускорение при K = 1 в большинстве своем выше, чем при K = n или 13, хотя случаются выбросы. Это может объясняться как большим числом пересылок, так и меньшей локальностью использующихся каждым процессом данных в памяти.