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

Сайбель Т. А.

Москва 2021

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:

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

Преобразование Адамара задается следующей матрицей:

$$H = {}^{1}/_{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

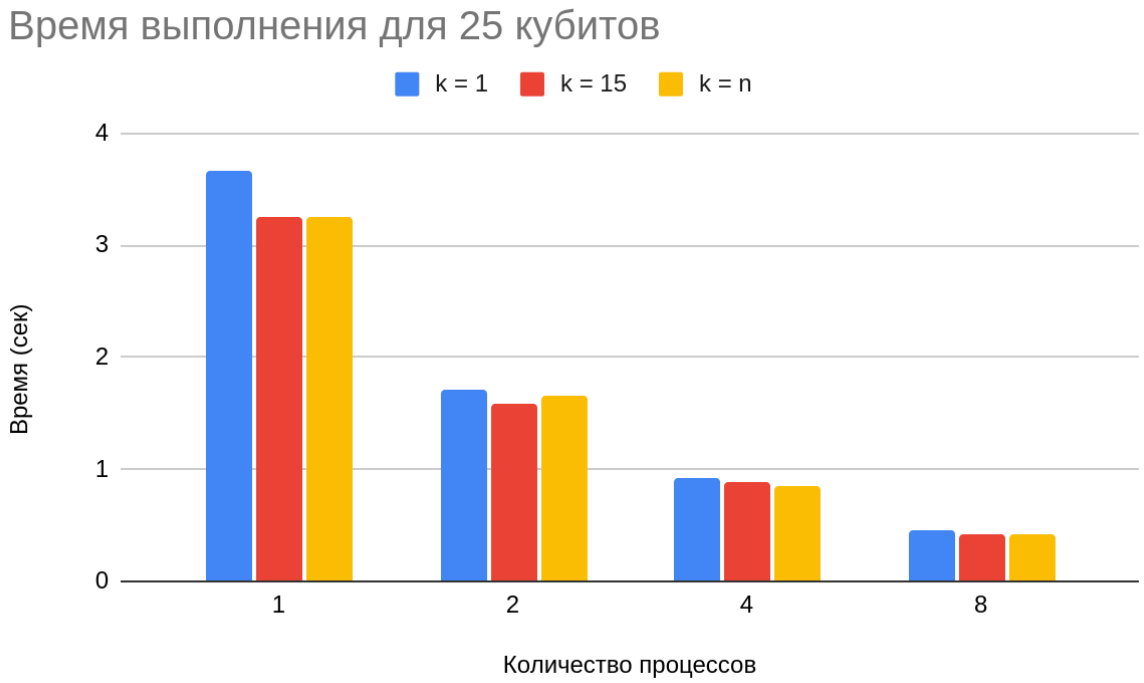
Аппаратное обеспечение:Исследования проводились на вычислительном комплексе IBM Polus.

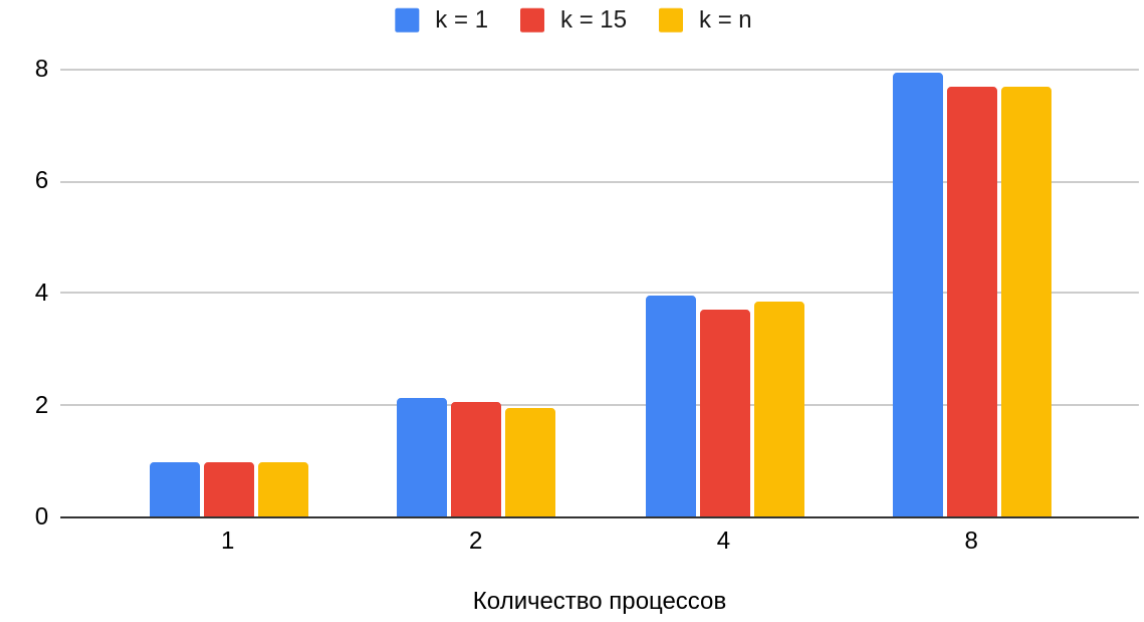
Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция MPI_Wtime().

Результаты выполнения

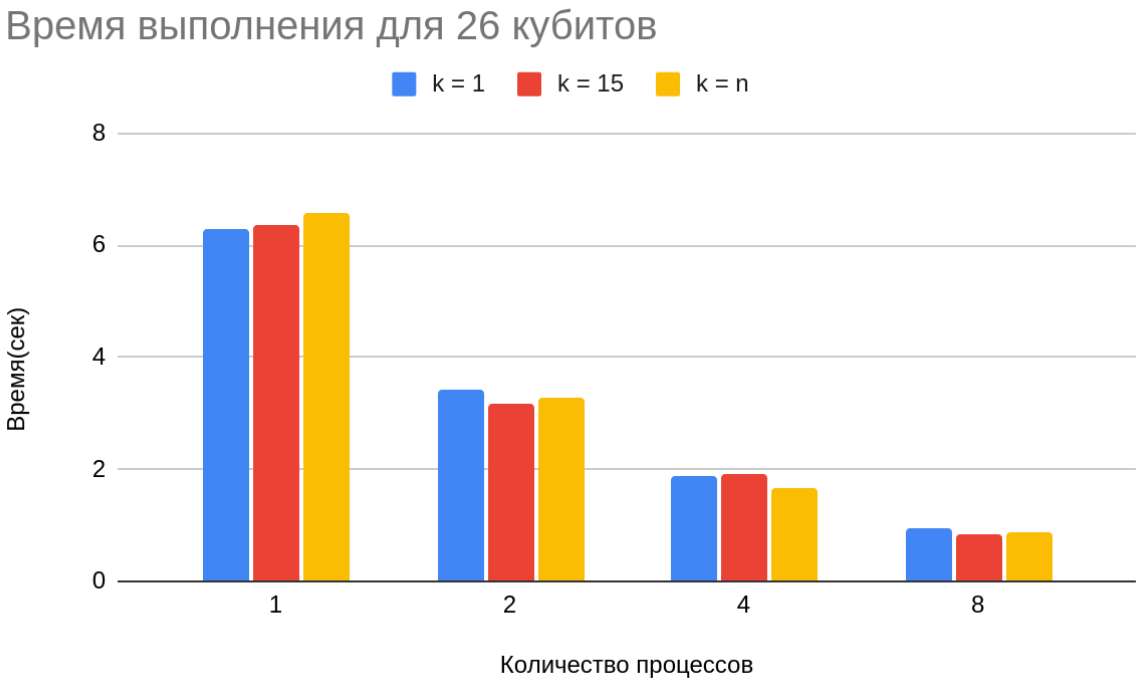
Количество кубитов (n)	Количество процессов	Время работы (сек)			Ускорение		
		k = 1	k = 15	k = n	k = 1	k = 15	k = n
25	1	3,67	3,25809	3,25809	1	1	1
	2	1,71777	1,59305	1,66339	2,135493692	2,045190044	1,958704814
	4	0,926378	0,878584	0,847635	3,959816619	3,708342059	3,843741705
	8	0,462666	0,422991	0,422991	7,928585632	7,702504309	7,702504309
26	1	6,30076	6,35526	6,57416	1	1	1
	2	3,43151	3,16936	3,28913	1,836147935	2,005218719	1,99875347
	4	1,87668	1,90364	1,6776	3,357397106	3,338477863	3,918788746
	8	0,934767	0,846246	0,877242	6,740460457	7,509943917	7,494123628
27	1	12,6124	12,7143	13,1911	1	1	1
	2	7,22634	6,5765	6,8021	1,745337197	1,933292785	1,939268755
	4	3,495	3,21219	3,99298	3,60869814	3,958140708	3,30357277
	8	1,80447	1,66237	1,66313	6,989531552	7,64829731	7,931490623

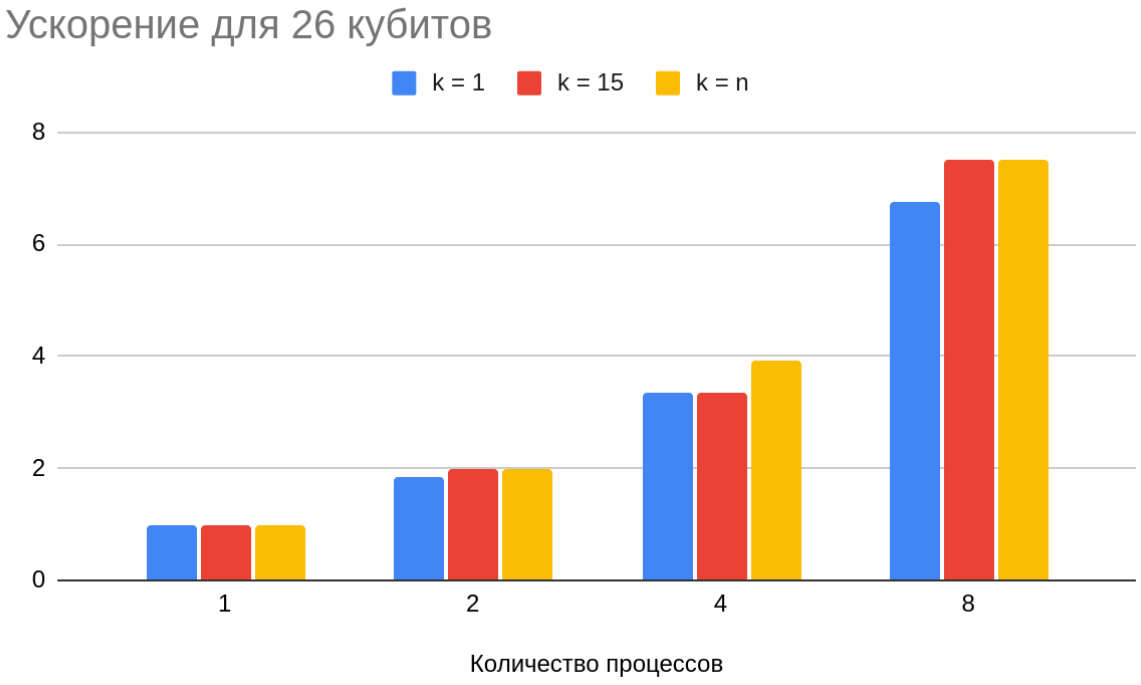
Результаты выполнения для 25 кубитов



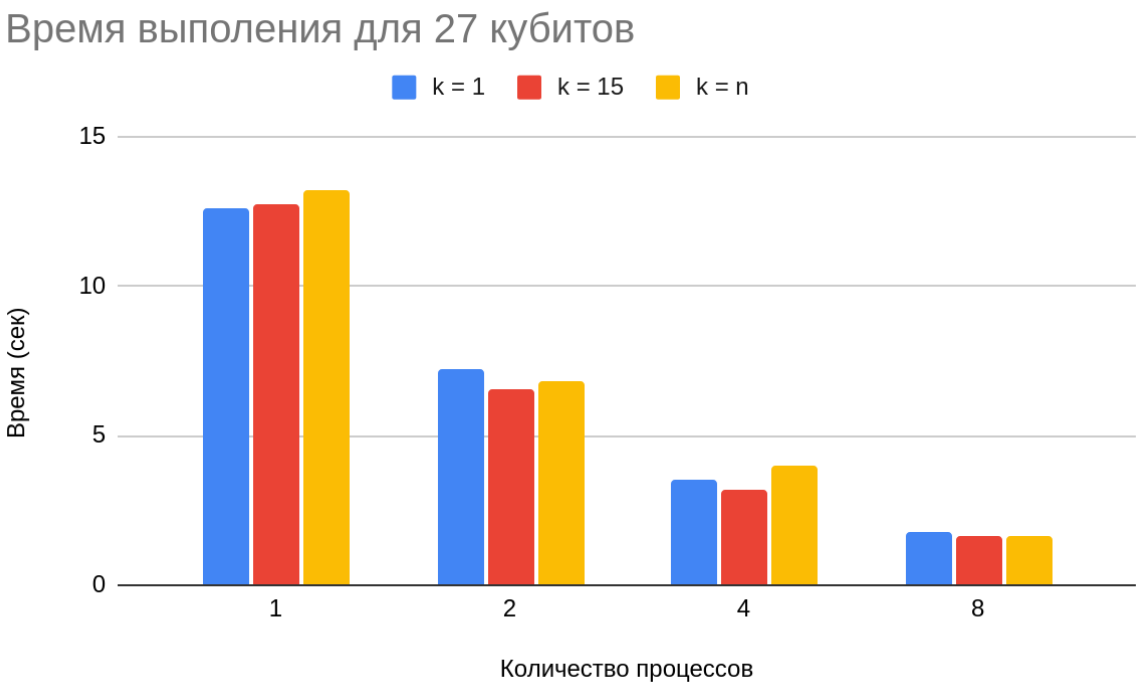


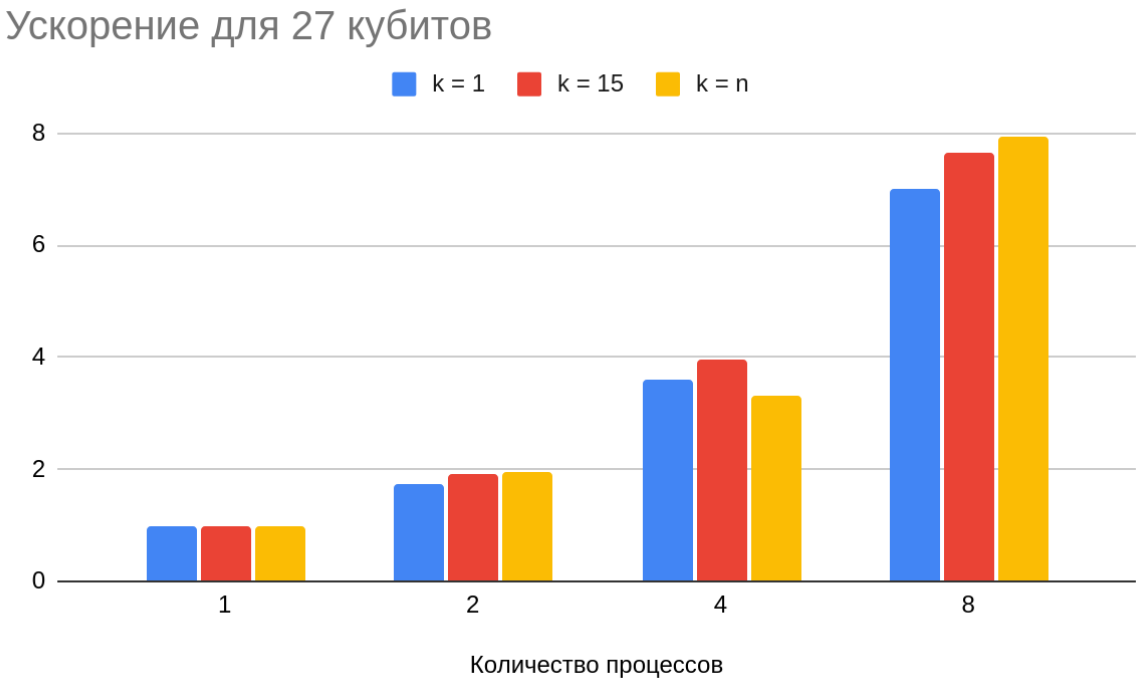
Результаты выполнения для 26 кубитов





Результаты выполнения для 27 кубитов





Основные выводы

Распараллеливание ускоряет выполнение программы, но с увеличением числа процессов эффективность снижается из-за роста количества пересылок.