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:

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

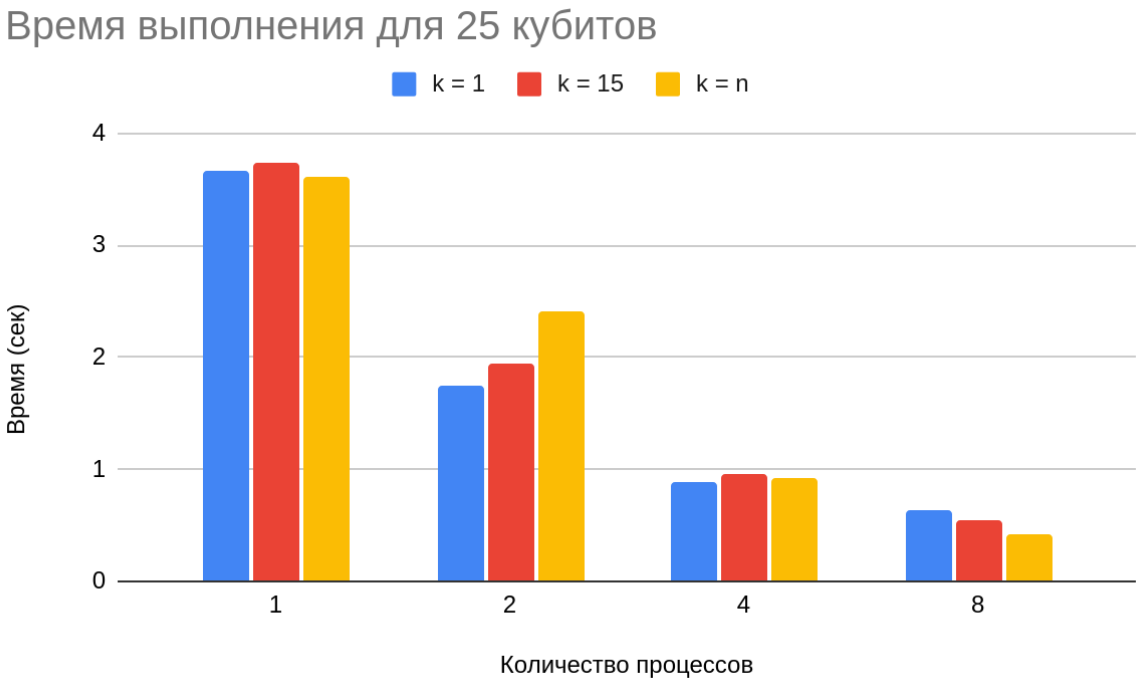
Аппаратное обеспечение:Исследования проводились на вычислительном комплексе IBM Polus.

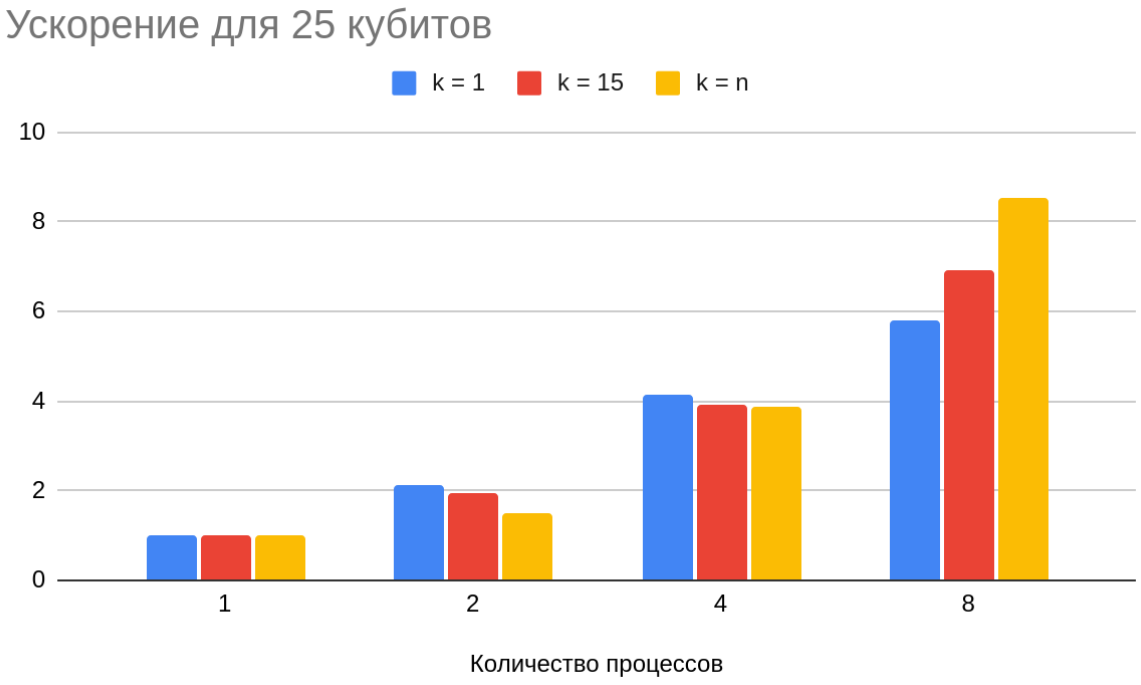
Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция MPI_Wtime().

Результаты выполнения

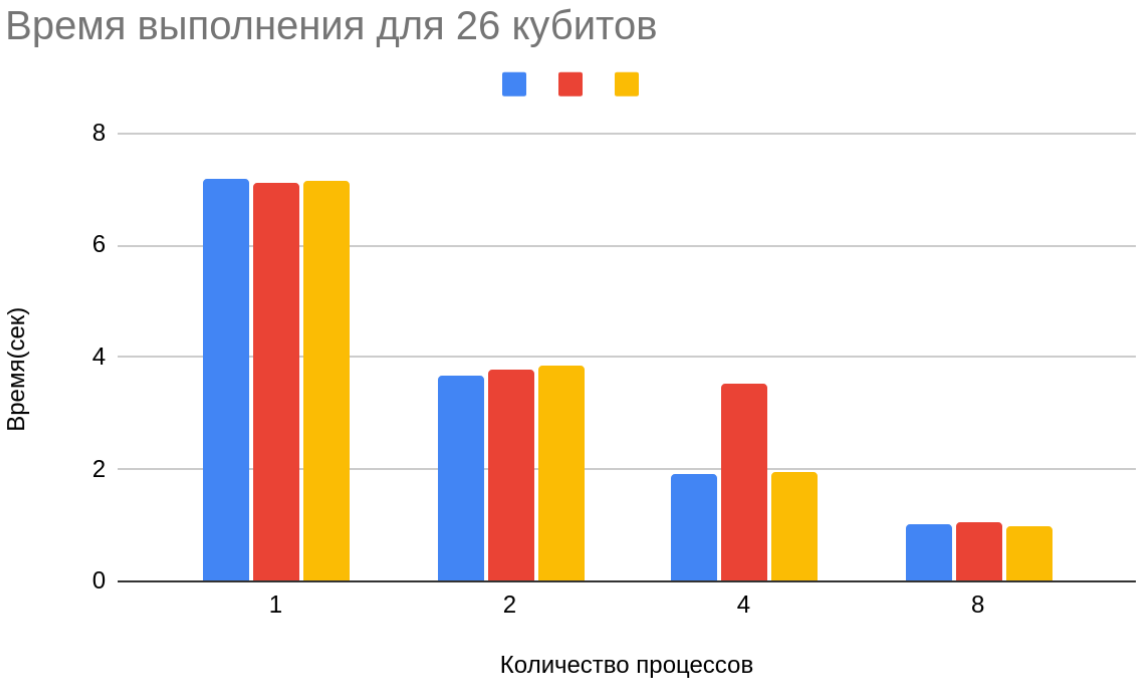
Количество кубитов (n)	Количество процессов	Время работы (сек)			Ускорение		
		k = 1	k = 15	k = n	k = 1	k = 15	k = n
25	1	3,67	3,74194	3,617166	1	1	1
	2	1,738048	1,938538	2,40423	2,110578649	1,930289734	1,504500817
	4	0,888508	0,953233	0,928816	4,128591977	3,925525029	3,894383818
	8	0,632259	0,541721	0,424204	5,801873916	6,907504047	8,526949298
26	1	7,182884	7,1183	7,149479	1	1	1
	2	3,668784	3,775942	3,867757	1,957837801	1,885171965	1,848481945
	4	1,918187	3,51985	1,957058	3,744621353	2,022330497	3,653176846
	8	1,034416	1,066542	0,983245	6,943902647	6,674186296	7,271309796
27	1	14,41311	17,53888	16,06032	1	1	1
	2	7,879647	8,281645	7,227281	1,829156814	2,117801475	2,222180098
	4	4,70703	3,764529	3,695418	3,06203912	2,585965	2,653841
	8	1,975873	1,788797	1,952147	3,5071434	3,136432	3,446798

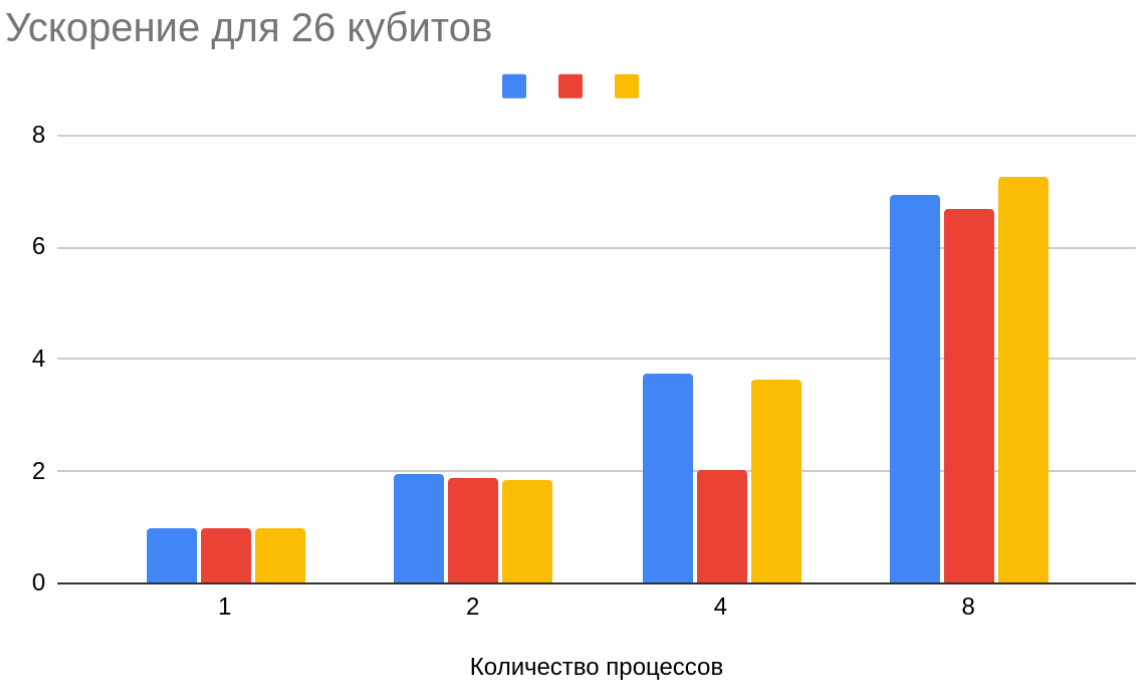
Результаты выполнения для 25 кубитов



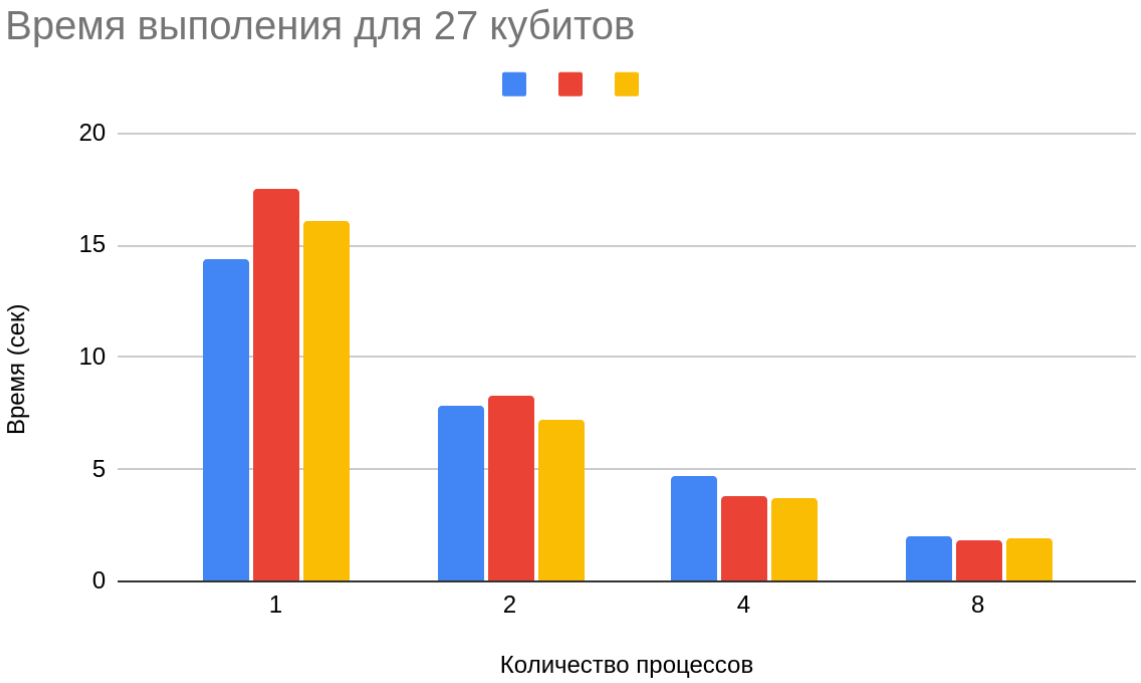


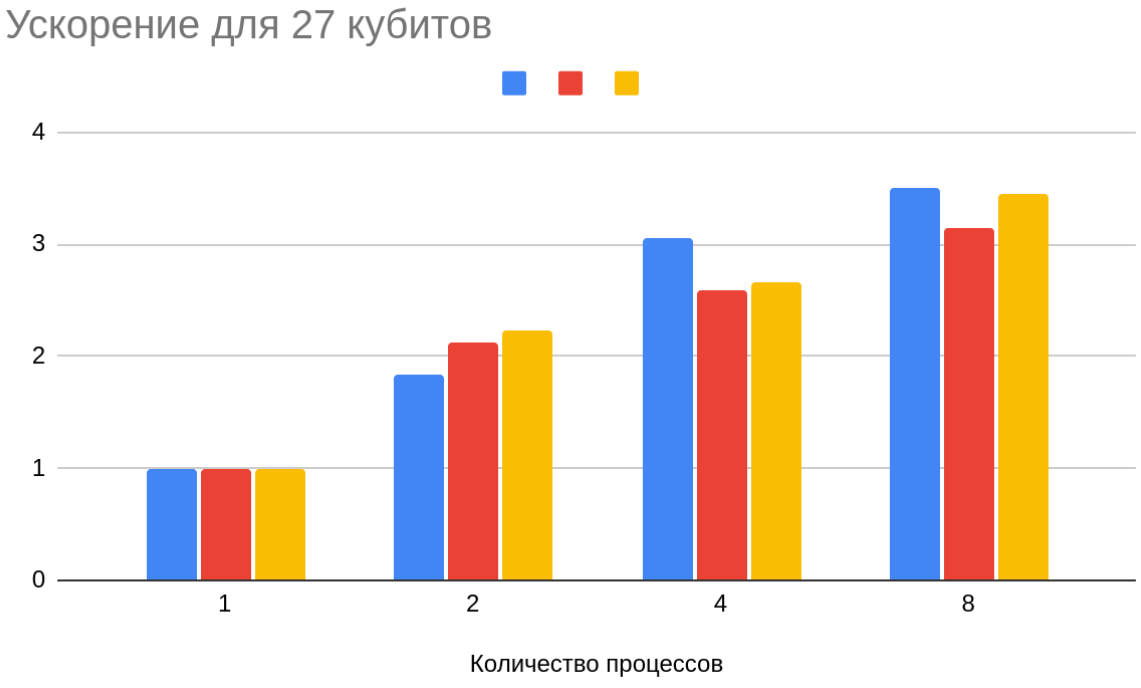
Результаты выполнения для 26 кубитов





Результаты выполнения для 27 кубитов





Основные выводы

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