Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
_____Кафедра Суперкомпьютеров и Квантовой Информатики_____

Отчёт № 2
Однокубитное квантовое преобразование, реализуемое с помощью параллельной программы с использованием MPI

Работу выполнила
Домрачева Д. А.

Москва 2018

Постановка задачи и формат данных

Задача:

Разработать параллельную программу с использованием MPI, реализующую алгоритм однокубитного квантового преобразования. Тип данных – complex<double>.

- Определить максимальное количество кубитов, для которых возможна работа программы на системе Polus. Выполнить теоретический расчет и проверить его экспериментально.
- Протестировать программу на системе Polus для преобразования Адамара и трех различных номеров кубита k.

Формат командной строки:

Режим генерации:

<число кубитов n> <номеркубита k> <флаг генерации начального вектора (gen)> <файл вывода или “no”, если вывод не нужен>

Режим чтения вектора из файла:

<входной файл с вектором a > <файл вывода>

Математическая постановка задачи:

Однокубитное преобразование – преобразование вектора $\{ a_{i_1 i_2 \dots i_n} \}$ в вектор $\{ b_{i_1 i_2 \dots i_n} \}$, задающееся комплексной матрицей 2×2 (в частности, преобразование Адамара задается матрицей $U = \frac{1}{\sqrt{(2)}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$) и числом 1≤k≤n – номером кубита, где элементы нового вектора вычисляются по формуле: $b_{i_1 i_2 \dots i_k \dots i_n} = \sum_{j_k=0}^{1} u_{i_k j_k} a_{i_1 i_2 \dots j_k \dots i_n}$.

Алгоритм и хранение данных

По аргументам командной строки происходит определение режима работы программы:

- Режим генерации вектора: каждый процесс генерирует свою часть вектора с помощью функции rand_r. Для избежания повтора данных в качестве seed используется текущее время, умноженное на ранг процесса плюс один.
- Режим считывания вектора: из указанного входного файла происходит параллельное считывание средствами MPI (каждому процессу задается смещение в файле, начиная с которого он считывает свою порцию данных).
- Если указана необходимость вывода полученного вектора в файл, то вывод производится средствами MPI по аналогии с чтением вектора из файла.

После генерации/чтения вектора выполняется вычисление выходного вектора. В программе определяются индексы элементов исходного вектора, необходимых для вычисления текущего элемента b , затем происходит определение необходимости обмена данными между процессами. Если обмен необходим, то он симметричен, поэтому два процесса выполнят пересылку своего элемента и прием элемента от другого процесса. Если необходимости в обмене нет (искомый элемент находится в памяти текущего процесса), то вычисление произойдет в обычном порядке.

На каждом процессе хранится лишь части векторов a и b (кроме случая, когда запущен только один процесс).

Тестирование и результаты

a) k = 4

Количество кубитов	**Количество процессов**	**Время работы программы (сек)**	**Ускорение**
25	1	4.1454	1
	2	2.13361	1.961
	4	1.12951	3.671
	8	0.67553	6.136
26	1	8.34621	1
	2	4.30904	1.937
	4	2.55035	3.273
	8	1.27393	6.571
27	1	16.8175	1
	2	8.62175	1.951
	4	4.53488	3.713
	8	2.58928	6.497

b) k = 1

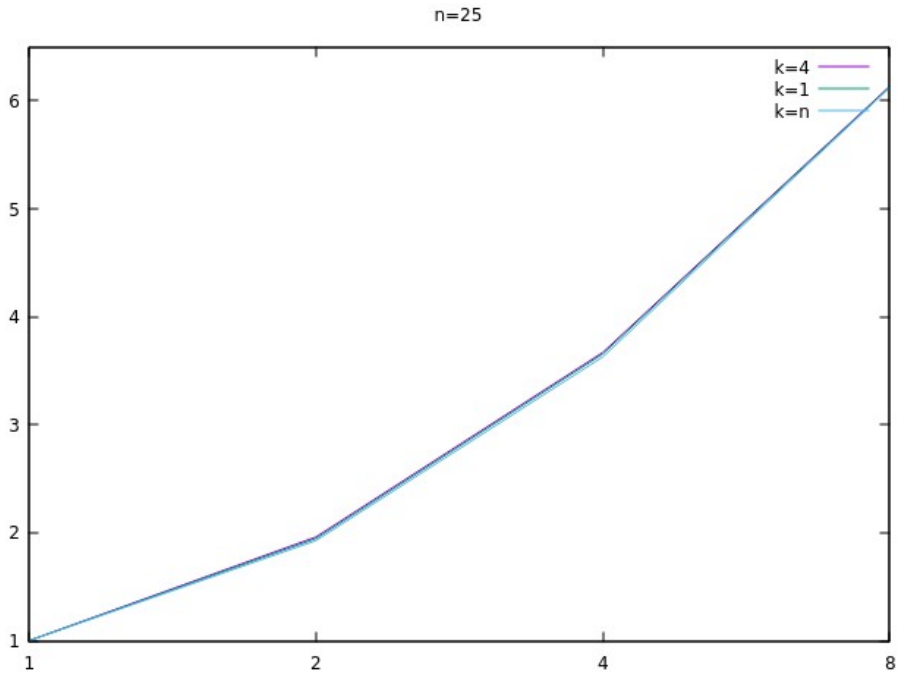
Количество кубитов	**Количество процессов**	**Время работы программы (сек)**	**Ускорение**
25	1	4.15852	1
	2	2.13781	1.945
	4	1.13762	3.655
	8	0.67819	6.131
26	1	8.76331	1
	2	4.53352	1.933
	4	2.53421	3.458
	8	1.40685	6.229
27	1	16.6313	1
	2	8.54201	1.947
	4	4.43028	3.754
	8	2.64871	6.279

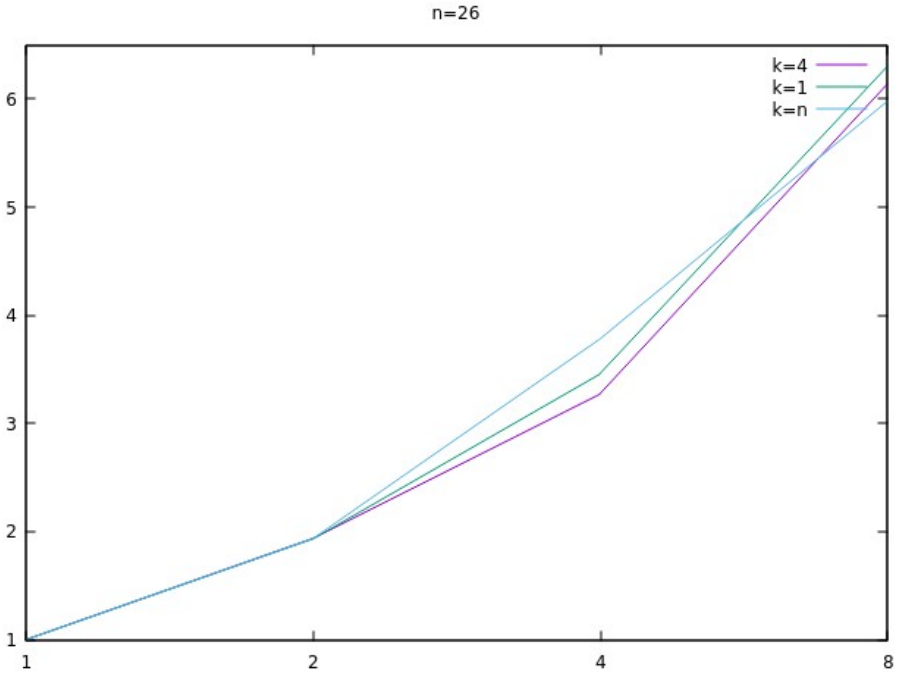
c) k = n

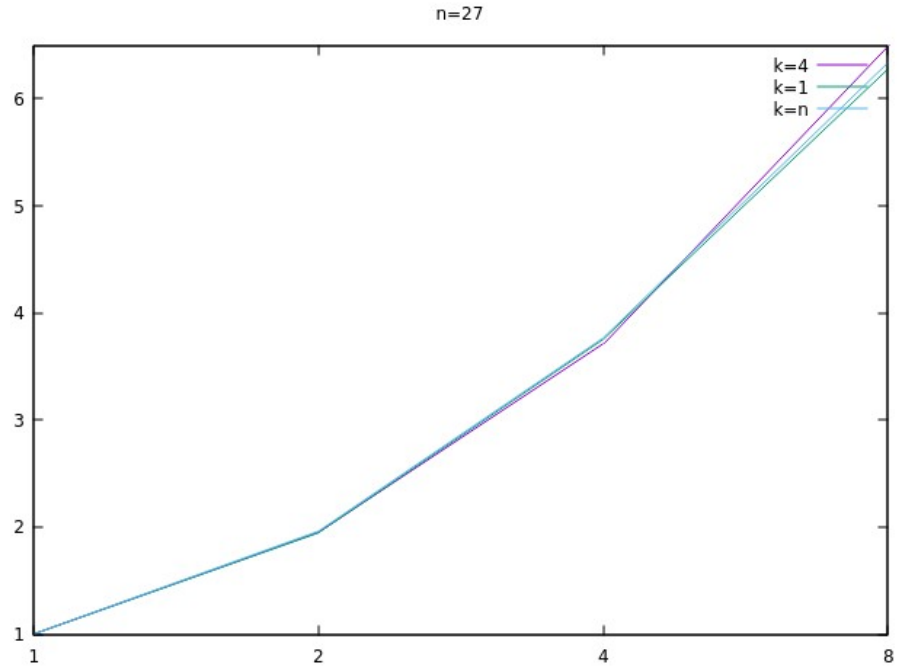
Количество кубитов	**Количество процессов**	**Время работы программы (сек)**	**Ускорение**
25	1	4.14326	1
	2	2.15391	1.927
	4	1.14184	3.628
	8	0.67667	6.123
26	1	8.31708	1
	2	4.29621	1.936
	4	2.19898	3.784
	8	1.39114	5.983
27	1	16.7612	1
	2	8.54867	1.961
	4	4.45142	3.766
	8	2.64482	6.339

Для тестирования корректности работы программы были проведены запуски программы в режиме считывания вектора размера 2^{16} из файла на 1, 2, 4 и 8 процессах. Результирующие векторы данных запусков совпали, следовательно, стоит полагать, что программа работает корректно.

Графики ускорения







Таким образом, можно видеть, что с увеличением количества процессов происходит уменьшение времени работы, а вместе с тем и рост эффективности.