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

Отчёт № 3
Параллельная программанаMPI, которая реализует однокубитное квантовое преобразованиес шумами

Работу выполнила
Домрачева Д. А.

Москва 2018

Постановка задачи и формат данных

Задача:

Разработать параллельную программу с использованием MPI и OpenMP, реализующую алгоритм преобразования n-Адамар, протестировать на системе Polus. Тип данных – complex<double>.

Формат командной строки:

Режим генерации:

<число кубитов n> <точность e> <количество итераций> <количество потоков> <файл вывода или “no”, если вывод не нужен> <файл вывода значений 1-F> <входной файл, если необходим>

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: $b_{i_1 i_2 \dots i_k \dots i_n} = \sum_{j_k=0}^{1} u_{i_k j_k} a_{i_1 i_2 \dots j_k \dots i_n}$.

Преобразование n-Адамар – это преобразование Адамара, выполненное последовательно n раз над вектором состояний, при этом кубит, по которому проводится преобразование изменяется от 1 до n. Зашумленный вентиль определяется формулами:

$$H_e = HU(\theta), H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, U(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \theta = e\,\xi, \xi \sim N(0,1),$$

e – уровень шума.

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. Вычисление выполняется последовательно n раз, при этом кубит, по которому проводится преобразование изменяется от 1 до n.

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

Тестирование и результаты

п.2

Количество кубитов	Количество вычислительных узлов	Количество используемых потоков на узел	Время работы программы (сек)
28	1	1	74.2878
		2	68.5362
		4	52.1107
		8	31.2767
	2	1	39.3923
		2	23.9043
		4	16.8965
		8	13.3873
	4	1	20.0627
		2	12.5622
		4	8.99562
		8	6.3345

п.3

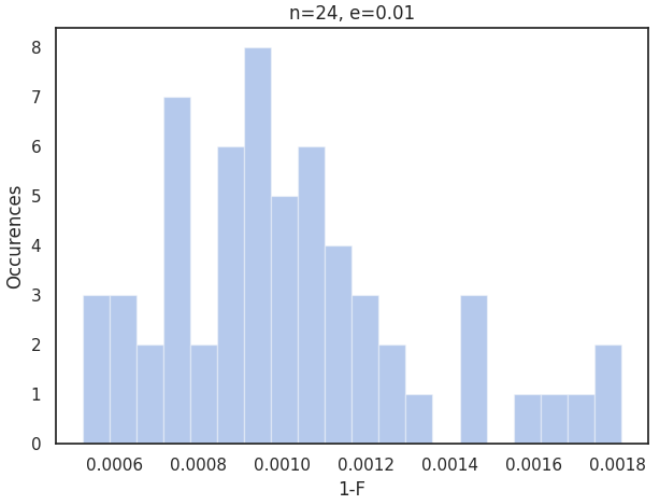
Количество кубитов	Среднее значение потерь
24	0.0010132
25	0.0010404
26	0.0011551
27	0.0011996
28	0.0012638

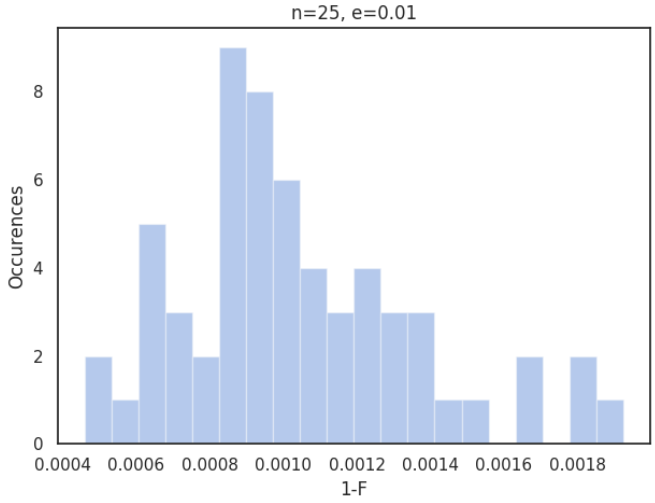
п.4

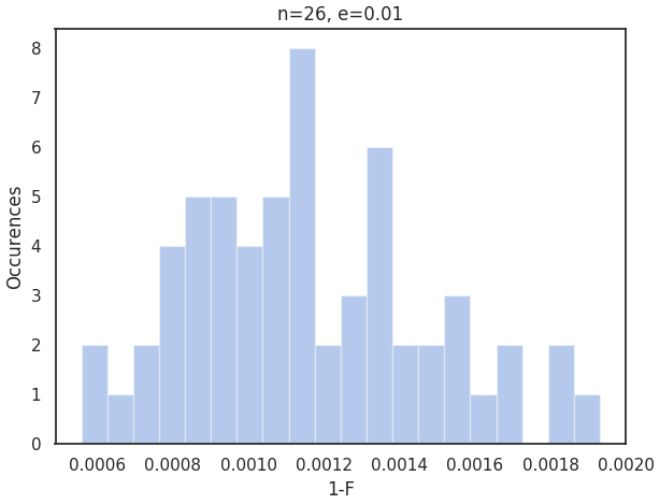
Количество кубитов	Среднее значение потерь
e=0.1	0.1071307
e=0.01	0.0011551
e=0.001	0.0000113

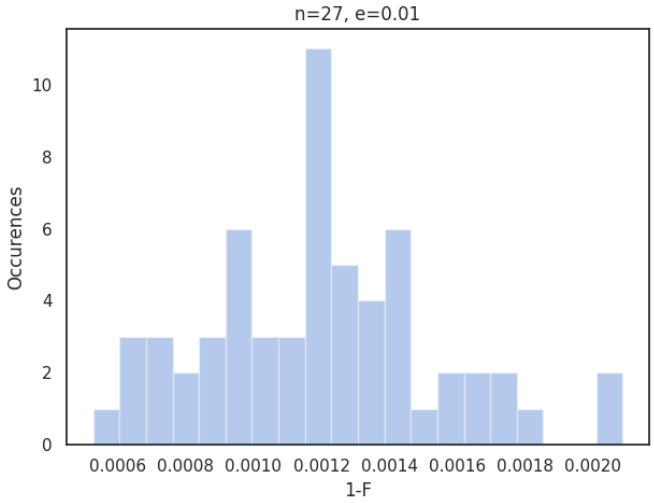
Для тестирования корректности работы программы были проведены запуски программы в режиме считывания вектора размера 2^{16} из файла на 1, 2, 4 и 8 процессах. Результирующие векторы данных запусков совпали, следовательно, стоит полагать, что программа работает корректно.

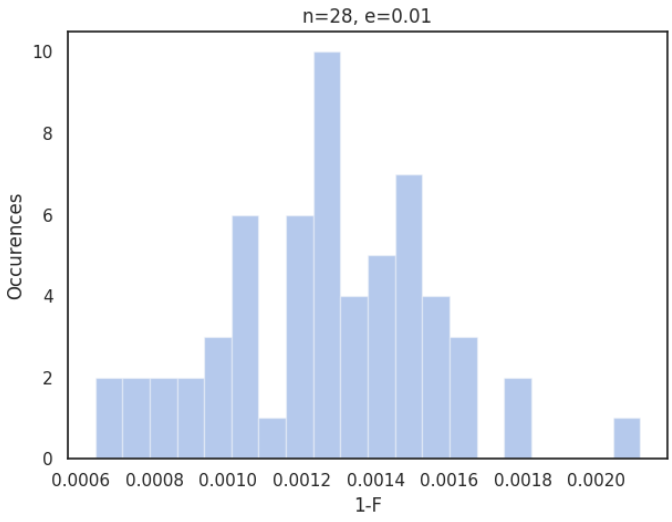
Графики
п.3


n=24, e=0.01
Occurences
1-F
0.0006 0.0008 0.0010 0.0012 0.0014 0.0016 0.0018

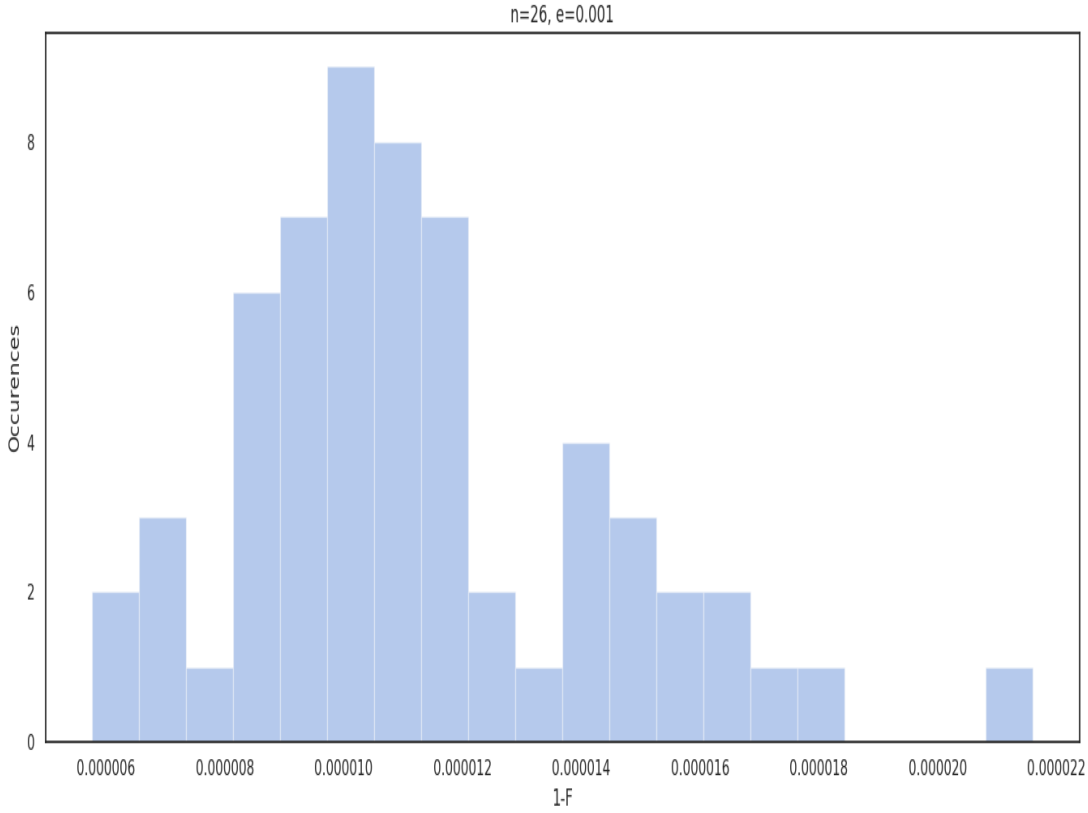

n=25, e=0.01
Occurences
1-F
0.0004 0.0006 0.0008 0.0010 0.0012 0.0014 0.0016 0.0018

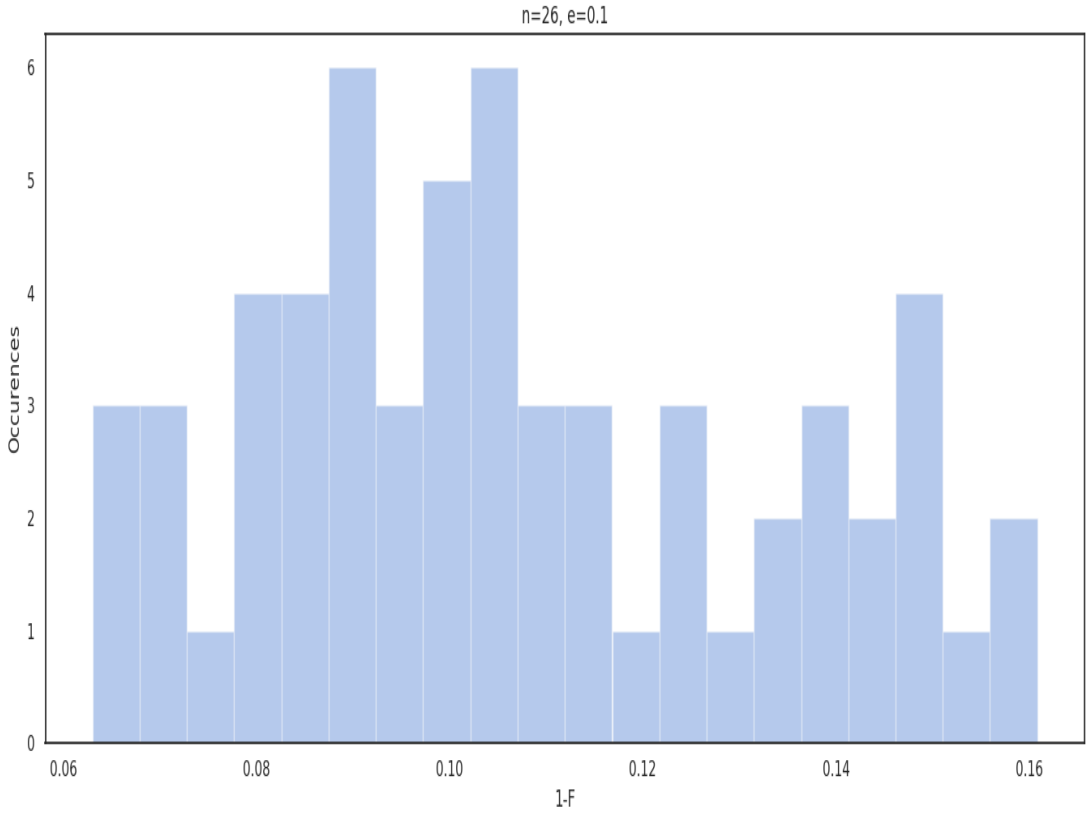

n=26, e=0.01
Occurences
1-F
0.0006 0.0008 0.0010 0.0012 0.0014 0.0016 0.0018 0.0020


n=27, e=0.01
Occurences
1-F
0.0006 0.0008 0.0010 0.0012 0.0014 0.0016 0.0018 0.0020


n=28, e=0.01
Occurences
1-F
0.0006 0.0008 0.0010 0.0012 0.0014 0.0016 0.0018 0.0020

п.4


n=26, e=0.001
Occurences
0
2
4
6
8
0.000006
0.000008
0.000010
0.000012
0.000014
0.000016
0.000018
0.000020
0.000022
1-F


n=26, e=0.1
Occurences
0
1
2
3
4
5
6
0.06
0.08
0.10
0.12
0.14
0.16
1-F