Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Практикум на ЭВМ: 6 семестр.

Отчёт № 3.

Анализ параллельной программы на MPI, реализующей зашумленное преобразование n-Адамар

Работу выполнил

Федоров В. В.

Москва 2021

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм квантового преобразования n-Адамар с зашумлением с использованием MPI, проанализировать зависимость времени выполнения программы от числа кубитов, потери точности преобразованного вектора от уровня шума и числа кубитов при помощи системы BlueGene.

Формат командной строки: Первый аргумент — единственный символ, означающий один из двух режимов работы:

- `g` — вектор состояния инициализируется случайными значениями. Дальнейшие аргументы командной строки: <число кубитов n> <уровень шума e> <имя файла для вывода времени выполнения> <имя файла для вывода потери точности>
- `f` — вектор состояния считывается из файла, вектор-результат также записывается в файл. Дальнейшие аргументы командной строки: <имя файла для считывания вектора> <уровень шума e> <имя файла для записи вектора>
    - Вектор записывается в файл в бинарном формате — сначала идут 4 бита числа кубитов n, затем — 2^n пар восьмибайтовых чисел с плавающей точкой, соответствующих действительной и мнимой части соответствующего элемента вектора.

Описание программы

Как и в задании 2, разделим весь вектор состояния на $2p$ блоков, тогда у каждого процесса будет по 2 блока. Пусть вектор уже распределен по процессам так, чтобы соседние по k-му кубиту состояния находились у одного и того же процесса, тогда необходимо перепослать блоки так, чтобы на одном процессе находились состояния, соседние по m-му кубиту. Тогда как для k, так и для m подсчитаем расстояние между блоками, принадлежащими одному и тому же процессу, из чего можно легко подсчитать, какому процессу какой блок принадлежит. После этого каждый процесс сможет выяснить, какие блоки хранятся у него и какие ему будут нужны, а затем — номера процессов, которым нужно отправить имеющиеся блоки и у которых нужно получить новые.

	k = 1	m = 3
000	0	0
001	1	0
010	2	1
011	3	1
100	0	2
101	1	2
110	2	3
111	3	3

В примере, продемонстрированном таблицей выше, видно, что процесс с номером 1 должен отправить свои блоки процессам 0 и 2, а получить новые блоки — у процессов 2 и 3.

Для генерации шума над произвольным однокубитным оператором необходимо домножить его на случайную матрицу:

$$U(\theta)=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix};\ \theta=e\,\xi;\ \xi\sim N(0,1)$$

Тестирование

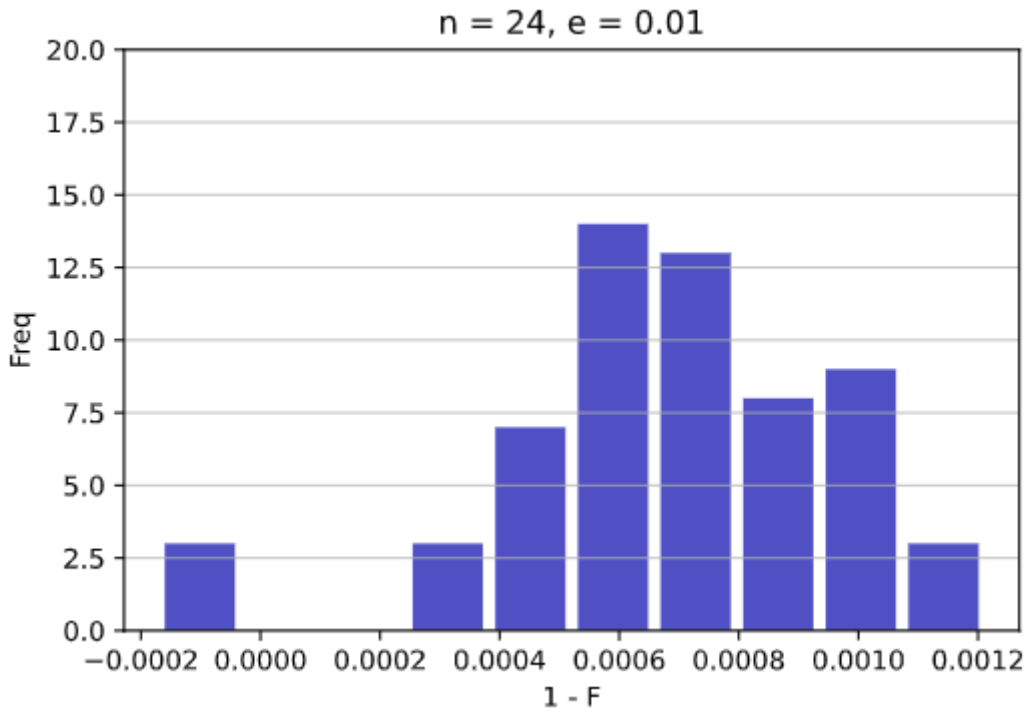
Программа тестировалась при помощи режима «f» на 5 заранее сгенерированных файлах с n = 9 всех количеств процессов от 1 до 16, равных степени двойки на нулевом уровне шума.

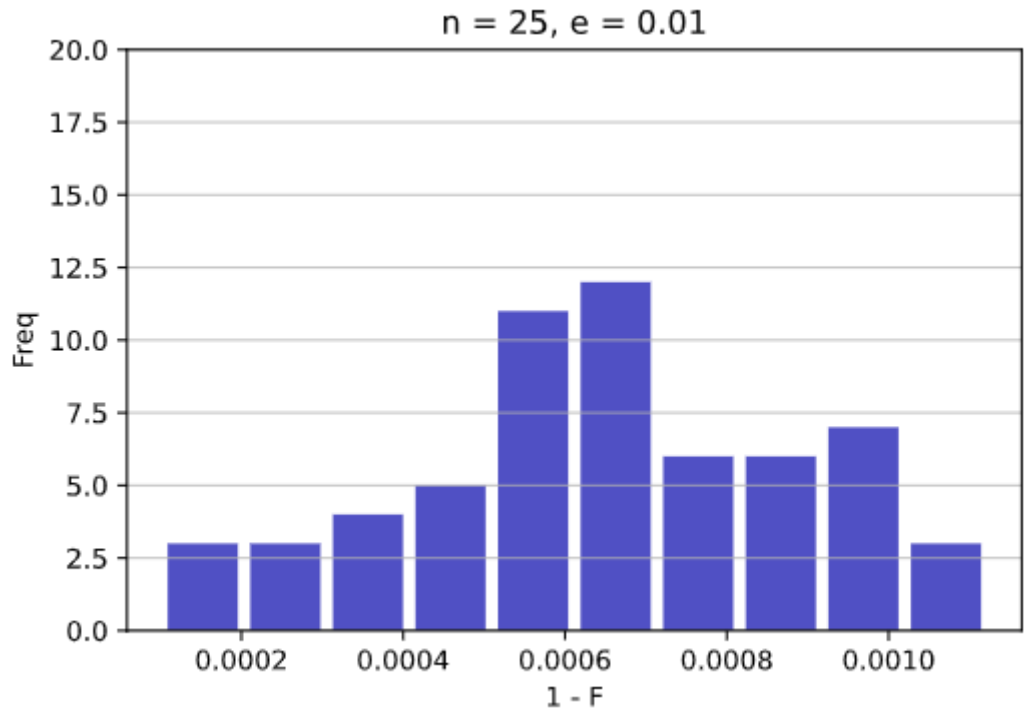
Результаты выполнения

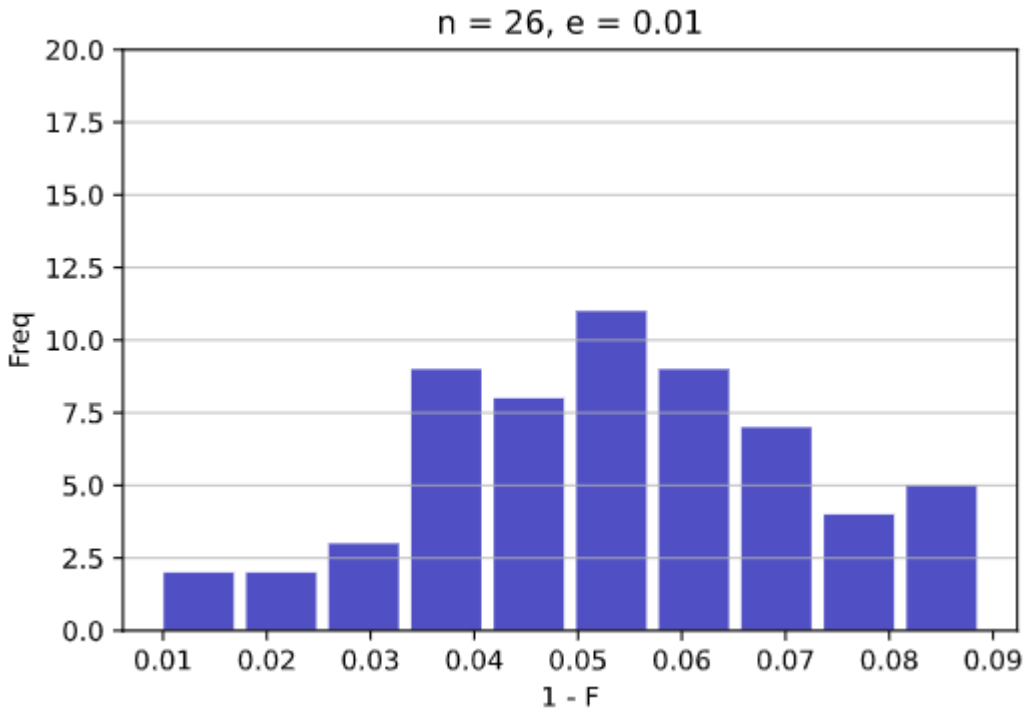
Анализ скорости работы программы

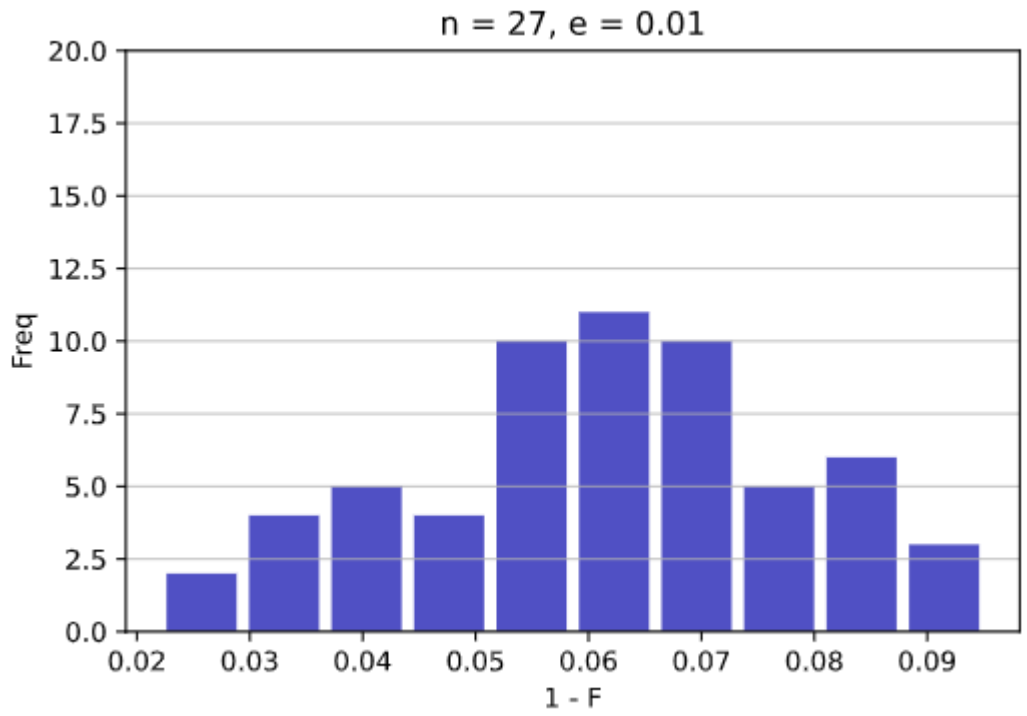
Кол-во кубитов	Кол-во проц-в	Время работы, с	Ускорение	Эффективность
28	32	9,297840	1,000000	1,000000
	64	4,862700	1,912074	0,956037
	128	2,583660	3,598709	0,899677
	256	1,368560	6,793886	0,849236
	512	0,685979	13,554118	0,847132

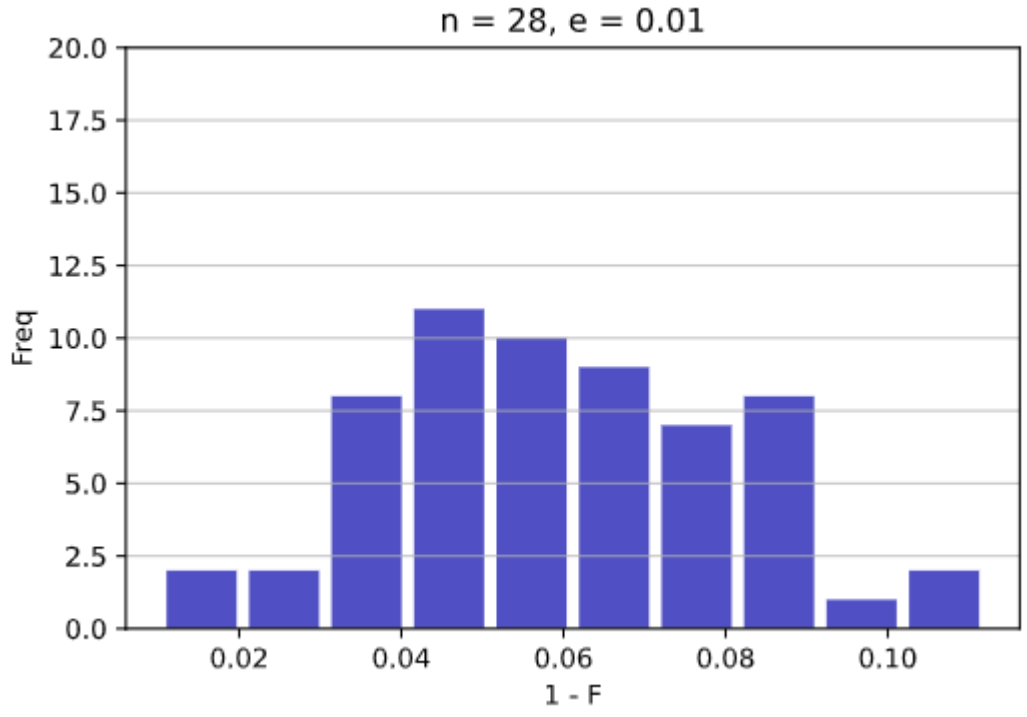
Гистограммы распределения потерь точности

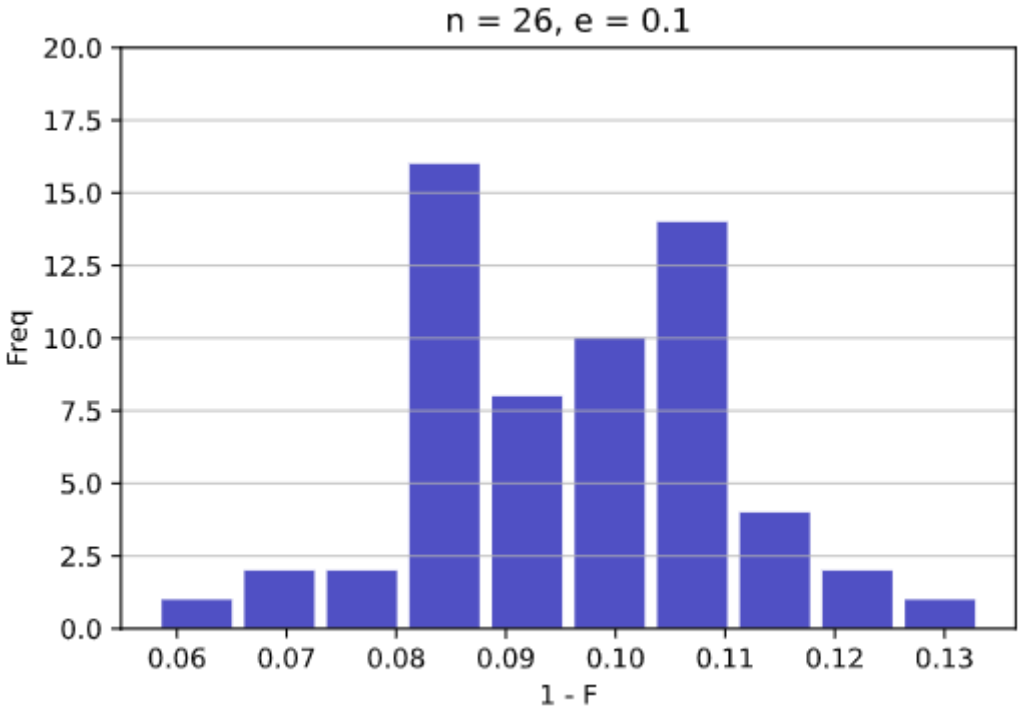


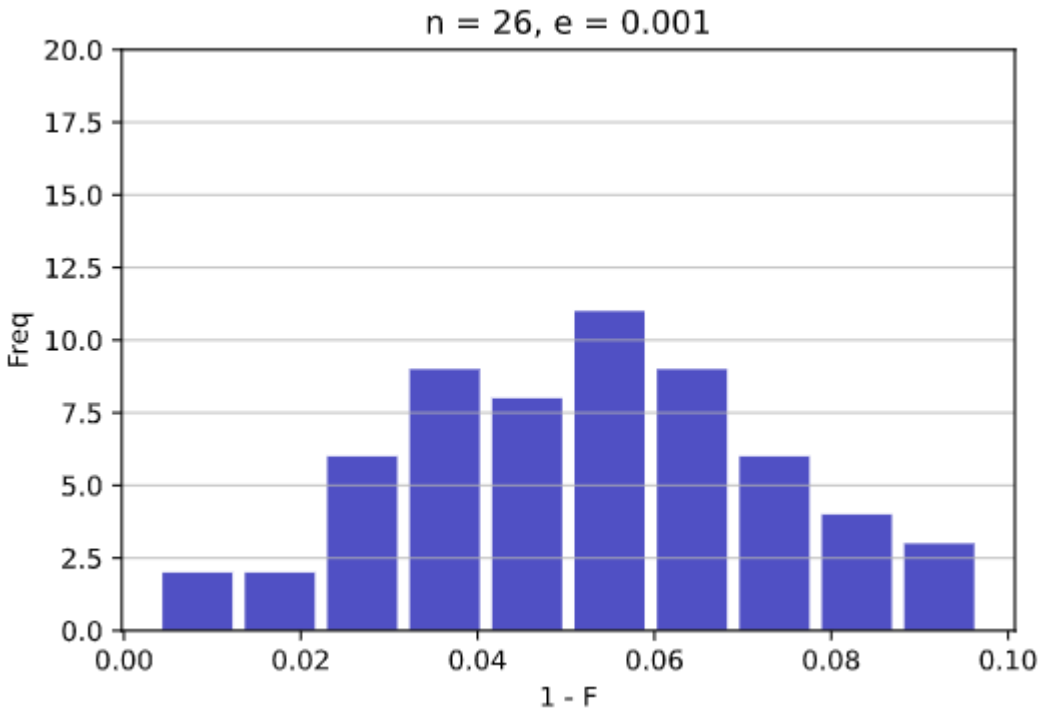

n = 25, e = 0.01
Freq
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0002
0.0004
0.0006
0.0008
0.0010
1 - F


n = 26, e = 0.01
Freq
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
1 - F


n = 27, e = 0.01
Freq
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
1 - F


n = 28, e = 0.01
Freq
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.02
0.04
0.06
0.08
0.10
1 - F





Средние значения потерь точности

24	0,000669
25	0,000644
26	0,054035
27	0,061703
28	0,058546

e = 0,01

0.1	0,096158
0.01	0,054035
0.001	0,052782

n = 26

Основные выводы.

В целом потери точности увеличиваются вместе с ростом числа кубитов, равно как и с ростом уровня шума. По графикам также можно заметить, что дисперсия тоже растет пропорционально этим величинам.