Московский Государственный Университет

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


# Практикум на ЭВМ.
# Отчет №1: Параллельная программа на OpenMP, которая реализует однокубитное квантовое преобразование.

Шахворостов Дмитрий 323

2021

# Постановка задачи

Реализовать параллельную программу на C++ с использованием OpenMP, которая выполняет однокубитное квантовое преобразование над вектором состояний длины $2^n$, где n – количество кубитов, по указанному номеру кубита k.

Формат командной строки:

./main <n> <k> <numtreads>

Сборка:
make

# Листинг программы

Инициализация матрицы U.

```
vector<complexd> a(num_qubits);
vector<vector<complexd>> u(2);
for (size_t i = 0; i < 2; ++i) {
    u[i].resize(2);
}
for (size_t i = 0; i < 2; ++i) {
    for (size_t j = 0; j < 2; ++j) {
            u[i][j] = 1.0 / sqrt(2);
    }
}
u[1][1] *= -1;

a_start_time = omp_get_wtime();

srand(omp_get_wtime());
int temp_seed = rand();
```

Инициализация вектора a.

```
#pragma omp parallel
{
    unsigned int seed = temp_seed * (omp_get_thread_num() + 1);
    #pragma omp for
    for (size_t i = 0; i < num_qubits; ++i) {
        a[i] = complexd(((double) rand_r(&seed)) / RAND_MAX,
                        ((double) rand_r(&seed)) / RAND_MAX);
    }
}

a_end_time = omp_get_wtime();
b_start_time = omp_get_wtime();

vector<complexd> b = single_qubit_transform(a, u, n, k);
b_end_time = omp_get_wtime();
```

Однокубитное квантовое преобразование.

```
vector<complexd> single_qubit_transform(vector<complexd>& a, vector<vector<complexd>>& u,
                                        size_t n, size_t k) {
    size_t num_qubits = 1 << n, temp = 1 << (n - k);
    vector<complexd> b(num_qubits);

    #pragma omp parallel
    {
        #pragma omp for
        for (size_t i = 0; i < num_qubits; ++i) {
            b[i] = u[(i & temp) >> (n - k)][0] * a[(i | temp) ^ temp]
            + u[(i & temp) >> (n - k)][1] * a[i | temp];
        }
    }

    return b;
}
```

# Результаты

А) 18 позиция.

| Количество кубитов | Количество потоков | Время работы программы(сек) | Ускорение |
|---|---|---|---|
| 20 | 1 | 0,036 | 1,000 |
| | 2 | 0,019 | 1,885 |
| | 4 | 0,011 | 3,259 |
| | 8 | 0,007 | 5,310 |
| 24 | 1 | 0,577 | 1,000 |
| | 2 | 0,305 | 1,895 |
| | 4 | 0,168 | 3,430 |
| | 8 | 0,103 | 5,613 |
| 28 | 1 | 10,189 | 1,000 |
| | 2 | 4,956 | 2,056 |
| | 4 | 2,869 | 3,551 |
| | 8 | 1,678 | 6,072 |
| 30 | 1 | 38,702 | 1,000 |
| | 2 | 20,799 | 1,861 |
| | 4 | 11,521 | 3,359 |
| | 8 | 6,901 | 5,608 |

Б) 1 позиция

| Количество кубитов | Количество потоков | Время работы программы(сек) | Ускорение |
|---|---|---|---|
| 20 | 1 | 0,036 | 1,000 |
| | 2 | 0,019 | 1,880 |
| | 4 | 0,011 | 3,369 |
| | 8 | 0,007 | 5,486 |
| 24 | 1 | 0,578 | 1,000 |
| | 2 | 0,306 | 1,888 |
| | 4 | 0,171 | 3,388 |
| | 8 | 0,105 | 5,530 |
| 28 | 1 | 9,722 | 1,000 |
| | 2 | 5,141 | 1,891 |
| | 4 | 2,831 | 3,434 |
| | 8 | 1,770 | 5,492 |
| 30 | 1 | 50,448 | 1,000 |
| | 2 | 26,661 | 1,892 |
| | 4 | 14,654 | 3,443 |
| | 8 | 8,552 | 5,899 |

В) Последняя позиция.

| Количество кубитов | Количество потоков | Время работы программы(сек) | Ускорение |
|---|---|---|---|
| 20 | 1 | 0,036 | 1,000 |
| | 2 | 0,019 | 1,885 |
| | 4 | 0,011 | 3,347 |
| | 8 | 0,007 | 5,292 |
| 24 | 1 | 0,578 | 1,000 |
| | 2 | 0,306 | 1,891 |
| | 4 | 0,170 | 3,404 |
| | 8 | 0,104 | 5,560 |
| 28 | 1 | 9,364 | 1,000 |
| | 2 | 6,715 | 1,395 |
| | 4 | 2,869 | 3,264 |
| | 8 | 1,686 | 5,555 |
| 30 | 1 | 39,797 | 1,000 |
| | 2 | 28,107 | 1,416 |
| | 4 | 11,420 | 3,485 |
| | 8 | 7,074 | 5,626 |

## Основные выводы.

Исследования показывают, что изменение количества запущенных процессов оказывает значительное влияние на время выполнения программы. Другими словами, алгоритм хорошо масштабируется, но не стоит забывать про накладные расходы на поддержку параллельной программы тем более в случае OpenMP – накладные расходы на создание и завершение легковесных потоков. С увеличением номера кубита, к которому применяется преобразование, получаемое ускорение уменьшается.