Параллельный алгоритм моделирования идеального квантового алгоритма Гровера*

Д.Ю. Андреев[1,2], О.В. Корж[1], С.В. Коробков[1], А.Ю. Чернявский[1,3]

Московский государственный университет им. М.В. Ломоносова[1],
Вычислительный центр им. А. А. Дородницына РАН[2],
Физико-технологический институт РАН[3]

Одной из задач, решение которой предполагается получить с помощью экзафлопсного суперкомпьютера, является построение суперкомпьютера на новых принципах, который позволит получить существенный прогресс в скорости вычислений. В данной статье представлено моделирование работы идеального квантового компьютера на суперкомпьютере Ломоносов. Предложен эффективный алгоритм распараллеливания вычислений при одно-, двух- и трехкубитных преобразованиях с использованием библиотеки DISLIB. В качестве примера моделирования рассматривается квантовый алгоритм Гровера и квантовое преобразование Фурье.

1. Введение

Квантовая информатика является относительно молодой и очень бурно развивающейся областью современной науки. Из важнейших направлений квантовой информатики можно выделить квантовые вычисления (с которыми связана данная работа), квантовую криптографию и моделирование квантовых систем. При успешном создании квантового компьютера реализуемые на нем алгоритмы позволят решать некоторые важные задачи существенно быстрее, нежели на классических компьютерах. Так алгоритм Шора[1] позволяет раскладывать числа на простые множители за полиномиальное время, а алгоритм Гровера[2] позволяет решать любые переборные задачи за корень из классического времени. Для создания полномасштабного квантового компьютера, несомненно, необходимо моделирование его работы. Особенно важно моделирование на основе реальных физических систем с учетом квантового шума, который является основным препятствием на пути реализации квантовых битов (кубитов) и вентилей. Отметим, что с ростом числа кубитов, необходимые вычислительные ресурсы растут экспоненциально. В связи с этим фактом для решения многих задач даже с не очень большим числом кубитов невозможно обойтись без использования суперкомпьютеров. Данная работа посвящена моделированию идеальных квантовых схем и является основополагающим шагом к решению важных практических задач квантовой информатики. В работе рассматривается моделирование алгоритма Гровера, а также важнейшей подпрограммы квантовых алгоритмов – квантового преобразования Фурье, используемого в алгоритме Шора и других алгоритмах, основанных на выделении периода функций[3,4].

С точки зрения параллельных вычислений базовые алгоритмы квантовых вычислений относятся к классу DIC (Data Intensive Computing). Главным свойством задач этого класса является существенное преобладание чтения-записи данных по сравнению с количеством вычислений. При проведении каждой одно-, двух- и т.д. кубитных операции происходит изменение всего вектора состояния. При этом доступ к данным осуществляется в произвольном порядке: порядок зависит от последовательности номеров кубитов, для которых выполняется преобразование. Квантовые вычисления могут быть эффективно смоделированы на машинах с общей памятью, однако очевидно, что размер современных машин с общей памятью не позволит выполнить моделирование существенного количества кубитов. В данной статье предлагается рассмотреть моделирование квантовых вычислений на суперкомпьютере с распределенным хранением данных. Для реализации параллельной версии программы предлагается использовать метод активных сообщений, и в частности, библиотеку DISLIB[5]. Преимуществом такого подхода является возможность оптимизации пересылок данных между процессорами за счет исполь-

* Работа выполнена про поддержке грантов РФФИ 12-07-31229 и 12-01-31274.

зования низкоуровневых протоколов передачи данных в коммуникационных сетях. Высокоуровневый интерфейс библиотеки позволяет скрыть от пользователя особенности протоколов нижнего уровня, используемых на той или иной суперкомпьютерной архитектуре. Использование стандартного протокола MPI для такого рода задач представляется неэффективным, поскольку этот протокол является неэффективным для коммуникационно-сложных задач[6].

Статья включает в себя краткий обзор существующих подходов к моделированию квантовых вычислений на суперкомпьютерах, краткое описание квантовых алгоритмов, для которых проводится моделирование (алгоритма Гровера и квантового преобразования Фурье), описание программной реализации с использованием протокола активных сообщений. В заключительном разделе приводится описание результатов экспериментов для суперкомпьютера Ломоносов. Приводится анализ масштабируемости разработанных методов.

2. Обзор существующих подходов

При моделировании квантовых вычислений на суперкомпьютере количество необходимой памяти растет как степень от числа кубитов. Соответственно, возможности симуляции квантового компьютера на классических устройствах весьма ограничены. Так на современных персональных компьютерах доступное число кубитов составляет порядка 25-28, а для моделирования 35 идеальных кубитов уже требуется более 1 терабайта оперативной памяти.

Использование суперкомпьютера для моделирования квантовых вычислений требует специализированного программного обеспечения, которое позволит легко эмулировать выбранную квантовую схему. Естественно, для эффективного использования такое программное обеспечение должно быть параллельным. В статье [7] представлен алгоритм для моделирования на компьютере с общей памятью. В работе используется специальные структуры данных для хранения разреженных матриц, при помощи которых представляется эволюция квантовых состояний в ходе вычислений. В результате такого подхода было достигнуто моделирование 32 кубитов на довольно небольших ресурсах: 16 гигабайт памяти и 64 процессора. Но данный метод применим только для систем с общей памятью, где нет необходимости в пересылках между различными вычислительными узлами.

Для использования больших кластеров с распределенной памятью требуется осуществлять обмены между вычислительными узлами. Для этого наиболее часто используется протокол MPI. Примером реализации программного обеспечения удовлетворяющего данным условиям является набор программ QCMPI [8], написанный на языке Fortran с использованием MPI. Эта программа имеет большой набор операторов: стандартный оператор Паули, преобразование Адамара, контролируемое отрицание, фазовый сдвиг, и квантовое преобразование Фурье. Программная система использует библиотеки линейной алгебры BLAS[9], LAPACK[10], SCALAPACK[11]. В случае моделирования квантовых вычислений целесообразно использовать более частные алгоритмы для разреженных матриц.

В статье [12] описывается аналогичный подход к реализации модели квантового компьютера. В этой статье показано, что при моделировании 37 кубитного квантового компьютера можно добиться эффективности в 1 квантовую операцию за 10 секунд.

Также возможно использование реконфигурируемых вычислителей [13] для моделирования квантовых алгоритмов. Но и в этом случае размерность моделируемой системы ограничена памятью вычислителя.

В 2010 году было выполнено моделирование алгоритма Шора на суперкомпьютере Jugene в Суперкомпьютерном центре Юлиха [14]. Пиковая производительность этого суперкомпьютера на тот момент составляла порядка одного петафлопса, объем оперативной памяти составлял порядка 140 терабайт. Удалось выполнить моделирование 42 кубитов.

В случае же реализации подхода с хранением разреженных матриц и работой с такими матрицами с использованием специализированных протоколов обмена данными можно добиться увеличения размерности моделируемого квантового компьютера.

3. Квантовый алгоритм Гровера и квантовое преобразование Фурье

Задачей данной работы является суперкомпьютерное моделирование многокубитных квантовых схем для алгоритма Гровера и квантового преобразование Фурье. Кратко опишем формализм квантовых вычислений и рассматриваемые схемы.

3.1 Общая схема квантовых вычислений

Подробное описание формализма квантовых вычислений можно найти, например, в книгах [3,4].

Состояние одного кубита квантового компьютера определяется нормированным на единицу вектором пространства двумерного комплексного пространства $\mathbb{C}^2$:

$$a|0\rangle + b|1\rangle, |a|^2 + |b|^2 = 1,$$

где через вектора $|0\rangle$ и $|1\rangle$ обозначаются базисные вектора рассматриваемого пространства.

Состояние n-кубитной квантовой системы лежит тензорном произведении однокубитных пространств $(\mathbb{C}^2)^{\otimes n} = \underbrace{\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n}$, соответствующие вектора имеют вид

$$\sum_{i_1, i_2, \ldots, i_n = 0}^{1} c_{i_1, i_2, \ldots, i_n} |i_1 i_2 \ldots i_n\rangle. \quad (1)$$

В каждый момент времени состояние меняется под действием некоторого унитарного преобразования. Обычно, преобразования действуют лишь на малое число кубитов (в данной работе рассматриваются одно- и двухкубитные преобразования). С формальной точки зрения такие преобразования имеют вид $U \otimes I$, где U – матрица, действующая на выбранные кубиты или кубит, I - единичная матрица, действующая на остальные кубиты. Квантовый алгоритм представляет собой последовательное применение таких преобразований. Входные данные подаются либо в виде начального состояния (вектора) системы, либо в виде выбора применяемых преобразований. После окончания работы алгоритма производится так называемое измерение, дающее для состояний вида (1) n-битный ответ $\{i_1 i_2 \ldots i_n\}$ с вероятностью $|c_{i_1, i_2, \ldots, i_n}|$. Таким образом квантовый алгоритм должен увеличивать коэффициент $c_{i_1, i_2, \ldots, i_n}$ при правильном ответе задачи.

Последовательность квантовых операций, которую и представляет из себя алгоритм, удобно представлять графически в виде так называемых квантовых схем, пример которой приведен на рис. 1.

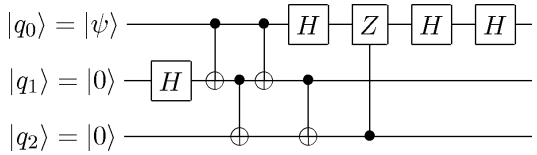

Рис. 1. Пример квантовой схемы. Каждая линия соответствует одному кубиту

3.2 Алгоритм Гровера

Пусть задана булева функция $f(x), x \in \{0,1\}^n$. Необходимо найти x, такой что $f(x) = 1$. Очевидно, что классический алгоритм в среднем решает задачу за 2^{n-1} обращений к функции f (оракулу), причем улучшить порядок $O(2^n)$ невозможно. Алгоритм Гровера позволяет решить задачу за время $O(2^{n/2})$. Важно отметить, что для работы алгоритма Гровера необходим так называемый квантовый оракул, переводящий базисное квантовое состояние $|x\rangle$ в

$(-1)^{f(x)} | x\rangle$. Несложно показать, что такой оракул может быть легко создан в виде квантовой схемы, для любой булевой функции, заданной в виде схемы из функциональных элементов. В силу ограниченности статьи мы не будем приводить квантовую схему алгоритма Гровера и объяснение действия его работы - соответствующую информацию можно найти в [3,4]. Приведем лишь особенности реализации алгоритма.

Важным подпрограммой (или подсхемой) алгоритма Гровера является инверсия относительно нулевого состояния $2|0^n\rangle\langle 0^n| - I$. В работе данное преобразование реализуется на основе методов раздела 4.3 книги [4] при помощи преобразований Тоффоли и контролируемого преобразования $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Само преобразование Тоффоли реализуется в виде схемы на рис. 2, где $V = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ - корень из операции NOT.

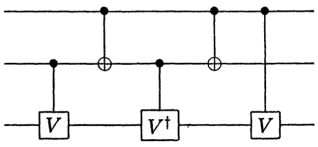

Рис. 2. Реализация элемента Тоффоли

Оракул в представленных тестовых задачах реализуется в виде обращения знака элемента вектора, соответствующего правильному ответу, однако, при необходимости может быть заменен на произвольную квантовую схему, соответствующую классической схеме вычисления некоторой булевой функции. Также в большинстве тестов инверсия производится аналогично оракулу (обращением знака при соответствующих амплитудах).

3.3 Квантовое преобразование Фурье

Квантовое преобразование Фурье является квантовым аналогом дискретного преобразования Фурье. Оно играет ключевую роль в алгоритме Шора, который позволяет разложить число N на простые множители за время $O(\log^2 N \log^3(\log N))$, что дает экспоненциальный рост скорости относительно наилучшего известного на данный момент классического алгоритма. Кроме того на основе квантового преобразования Фурье строятся алгоритмы моделирования квантовых систем на квантовом компьютере [7].

Само преобразование имеет вид $| j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{j \cdot k} | k\rangle$, $\omega = \frac{2\pi i}{2^n}$. Для соответствия с видом (1) следует отметить, что под j и k подразумеваются двоичные записи этих чисел. Такое преобразование задается рекурсивной схемой, приведенной на рис. 3.

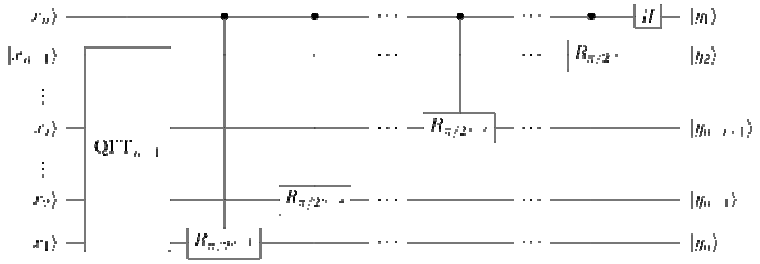

Рис. 3. Схема квантового преобразования Фурье

4. Реализация параллельной версии алгоритма с помощью библиотеки DISLIB

Особенность квантовых алгоритмов с точки зрения реализации для параллельной вычислительной системы – необходимость постоянного нерегулярного доступа к данным, которые хранятся распределенно. Поэтому использование механизма односторонних активных сообщений представляется перспективным.

Основной проблемой в реализации является размер данных, хранимых для преставления текущего вектора состояний. Для выполнение одной операции одно-, двух-, трехкубитного преобразования требуется хранить два массива комплексных чисел двойной точности. Размер каждого массива – $2^{nqubits}$ элементов, где nqubits – количество моделируемых кубитов. В случае моделирования алгоритма Гровера размер массивов $2^{2*nqubits}$ элементов.

На рис. 4 показан пример хранения вектора состояний. В примере показано хранение данных для случая двух процессоров и 4 кубитов.

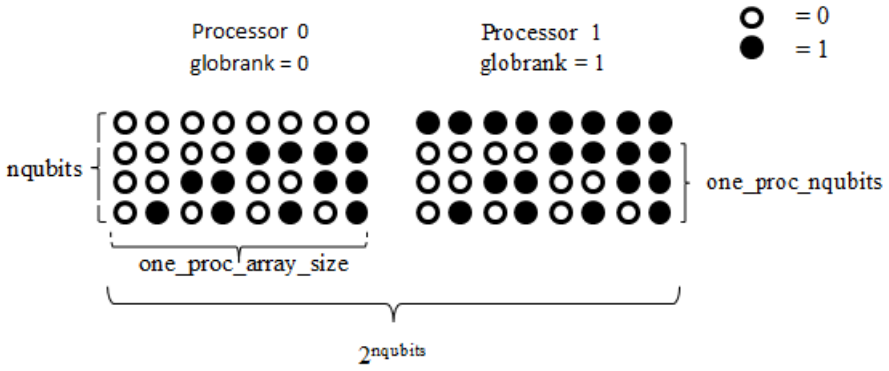

Рис. 4. Расположение данных по процессорам

Битовая длина номера элемента в массиве равна количеству используемых кубитов. При этом номер элемента можно представить как номер процессора, на котором находятся данные, и номер элемента в локальном массиве на каждом процессоре. В показанном выше примере номер процессора соответствует первому биту. Таким образом, можно определять номер элемента в глобальном массиве по номеру элемента в локальном массиве и номеру процессора. Однако это накладывает ограничение, что количество используемых процессоров должно быть степенью двойки.

При реализации однокубитной операции происходит изменение всех элементов массива состояния. При этом для расчета нового значения элемента используется текущее значение и значение элемента, индекс которого в глобальном массиве состояний отличается на один бит. Причем отличающийся бит находится в позиции, которая соответствует номеру кубита, по которому осуществляется преобразование. При этом для части номеров кубитов нужный элемент находится на текущем процессоре, а часть элементов необходимо передать по сети с других процессоров. На рис. 5 показаны необходимые обращения к данным для различных номеров кубитов. Можно видеть, что для приведённого примера обмен данными между процессорами потребуется только в случае, если номер кубита, по которому проводится преобразование, равен одному. В случае использования модели активных сообщений такая структура данных определяет, какому процессору нужно послать значение текущего обрабатываемого элемента. Соответственно идея распараллеливания однокубитной операции заключается в следующем. Для каждого элемента массива нужно посчитать сумму двух слагаемых: свое текущее значение, умноженное на некоторый коэффициент, и значение элемента с противоположным битом, также умноженное на некоторый коэффициент. Поскольку не определен порядок, в котором будет происходить суммирование, потребуется дополнительный массив, куда будут записаны результаты промежуточного суммирования. Если на процессоре не оказалось нужных данных для суммирования, то мы отправляем свой элемент на процессор, номер которого отличатся в бите,

соответствующем номеру активного кубита. И соответственно ждем от этого процессора его элемент. При этом получение элемента может произойти раньше, чем отправка. Для этого необходимо выполнять барьерную синхронизацию после отправки и получения всех элементов одной итерации. В Листинге 1 приведен код функции, реализующей отправку сообщений. В Листинге 2 показана функция-обработчик для этой функции отправки.

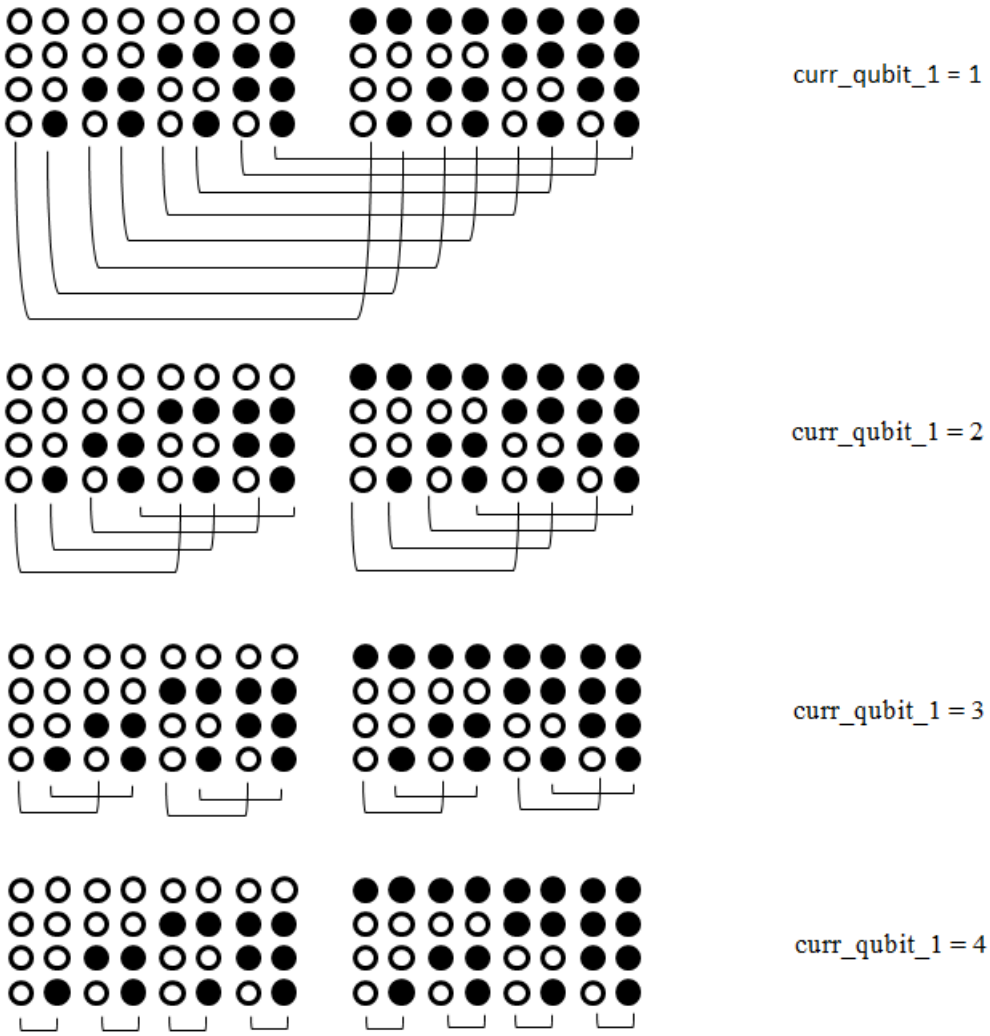

Рис. 5. Обмен данными при различных номерах активного кубита

Часто получается так, что матрица коэффициентов разреженная, таким образом можно не пересылать элементы, которые заведомо будут умножаться в итоговой сумме на ноль. Также вектор исходных состояний часто равен {1,0,0,…,0}, поэтому так же можно экономить на посылке нулевых элементов. Такая оптимизация нарушает симметричность посылок. Однако это допустимо в модели активных сообщений. И не влияет на производительность. Если в процессе не выполнилась отправка, то получение активируется вызовом барьера.

Для моделирования двух- и трехкубитных преобразований используется тот же принцип распараллеливания вычислений. Массив состояний равномерно разделяется по процессорам. Далее при изменении элементов определяется, какие являются локальными, а какие необходимо отправить / получить. Применяются аналогичные оптимизации с пересылками нулевых элементов и элементов, которые должны быть умножены на нулевые коэффициенты. С помощью определенных выше простейших функций реализуются набор операторов, таких как преобразование Адамара, фазовый сдвиг, CNOT и др.

```
// one qubit evolution function, current modification qubit is curr_qubit_1
// current matrix for transformation is in curr_U1
void one_qubit_evolution(int curr_qubit_1)
{
    long long tmp_send_proc_ind; // receiving processor number
    long long tmp_U_ind; // local index with opposite bit
    elem_ind tmp_send; // data for sending
    // output array initialization
    for(long long i=0;i<one_proc_array_size;i++) {out[i] = 0;}
    shmem_barrier_all();
    // modification of all local array elements
    for(long long i=0;i<one_proc_array_size;i++)
    {
        // required element is non-local
        if ((nqubits - curr_qubit_1) >= one_proc_nqubits)
        {
            // data for sending
            tmp_send.elem = in[i]; tmp_send.ind = i;
            // receiving processor number estimation
            tmp_send_proc_ind =
            globrank ^ (1L << (nqubits - curr_qubit_1 - one_proc_nqubits));
            // if bit in curr_qubit_1 position is 0
            if (tmp_send_proc_ind > globrank)
            {
                // current element addition
                out[i] += curr_U1[0][0] * in[i];
                // sending if element and coefficient are non-zero
                if (((abs(real(tmp_send.elem)) > 0.0000001)||(abs(imag(tmp_send.elem)) > 0.0000001))
                && (((abs(real(curr_U1[1][0]))) > 0.0000001)||(abs(imag(curr_U1[1][0]))) > 0.0000001))
                    shmem_send(&tmp_send, 1, sizeof(elem_ind), tmp_send_proc_ind);
            }
            else // if bit in curr_qubit_1 position is 1
            {
                // current element addition
                out[i] += curr_U1[1][1] * in[i];
                // sending if element and coefficient are non-zero
                if (((abs(real(tmp_send.elem)) > 0.0000001)||(abs(imag(tmp_send.elem)) > 0.0000001))
                && (((abs(real(curr_U1[1][0]))) > 0.0000001)||(abs(imag(curr_U1[1][0]))) > 0.0000001))
                    shmem_send(&tmp_send, 1, sizeof(elem_ind), tmp_send_proc_ind);
            }
        }
        else //all elements are at the current processor
        {   //local index with opposit bit
            tmp_U_ind = i ^ (1L<< (nqubits - curr_qubit_1));
            //if bit in curr_qubit_1 position is 1
            if ( i > tmp_U_ind )
                out[i] += curr_U1[1][1] * in[i] + curr_U1[1][0] * in[tmp_U_ind];
            //if bit in curr_qubit_1 position is 0
            else out[i] += curr_U1[0][0] * in[i] + curr_U1[0][1] * in[tmp_U_ind];
        }
    }
    // all processes are finished
    shmem_barrier_all();
}
```

Листинг 1. Параллельная функция для однокубитного преобразования

```
typedef struct elem_ind {
    complexd elem;
    long long ind;
}e_i;

// handler for one_qubit_evolution function
void one_qubit_evolution_hhnl(int from,void* data,int sz)
{
    // received data
    elem_ind * temp = (elem_ind *)data;

    // if current modification bit is 1
    if ((globrank - from) > 0 )
    out[temp->ind] += curr_U1[1][0] * temp->elem;
    // if current modification bit is 0
    else out[temp->ind] += curr_U1[0][1] * temp->elem;
}
```

Листинг 2. Хэндлер для функции однокубитного преобразования

В Листинге 3 приведена реализация алгоритма Гровера, последовательно вызывающая приведенные выше операторы. Особенность реализации алгоритма – массив состояний имеет размер $2^{2*nqubits}$ элементов. Это вдвое уменьшает количество доступных для моделирования кубитов. Аналогичным образом реализовано квантовое преобразование Фурье.

```
const double pi = asin(1.0)*2;
//Number of algorithm steps
double r = pi*(sqrt((double)(1<<(2*nqubits))))/4;

// Grover transformation
void Grover()
{
    Hn(); // n-qubits Hadamar transformation
    for (int i=1; i< r; i++)
    {
        Oracle(); // Oracle function
        Hn(); // n-qubits Hadamar transformation
        PhaseInverse(); // phase inverse transformation
        Hn(); // n-qubits Hadamar transformation
    }
}
```

Листинг 3. Реализация квантового алгоритма Гровера

Реализованы идеальные квантовые алгоритмы. В дальнейшем планируется реализация возможности добавления шумов. Пользователю предоставляется библиотека, реализующая базовые квантовые преобразования. Далее пользователь имеет возможность конструировать различные алгоритмы из этих преобразований. Библиотека легко расширяема. Переносимость библиотеки определяется возможностью установки библиотеки DISLIB на кластере. В дальнейшем предполагается разработка графического интерфейса для задания квантовой схемы с помощью графических нотаций.

5. Результаты тестирования алгоритмов на суперкомпьютере Ломоносов

На первом этапе тестирования реализованного алгоритма на суперкомпьютере Ломоносов требовалось провести оценку максимально возможного числа кубитов, которые получится моделировать. Это число ограничено размером данных, которые поместятся в оперативную память процессора. Теоретическая оценка показывает, что в текущей конфигурации суперкомпьютера и при использовании разработанных алгоритмов максимально возможный размер моделирования составляет 40 кубитов. На 32 узлах возможно моделирование 33 кубитов. Максимально возможное число кубитов, доступных для моделирования на одном процессоре – 29.

Далее интерес представляла оценка среднего времени выполнения одного квантового преобразования. Тестирование проводилось для преобразования Адамара. Проводилось n-кубитное преобразование и время работы делилось на количество кубитов. В таблице 1 приведено время работы программы в секундах для моделирования от 24 до 33 кубитов. Значение SF означает, что задача данного размера не может быть смоделирована на указанном количестве процессоров из-за недостаточного объема памяти.

Таблица 1. Время работы одного квантового преобразования

	24	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**
4	0.0319	0.0635	0.1275	0.2522	0.5032	1.1414	2.2330	SF	SF	SF
8	0.0102	0.0318	0.6473	0.1261	0.2518	0.5105	1.1423	2.2604	SF	SF
16	0.0083	0.0171	0.3213	0.0652	0.1293	0.2517	0.0503	1.1443	2.2896	SF
32	0.0043	0.0085	0.0160	0.0335	0.0628	0.1267	0.2504	0.5047	1.0718	2.3216

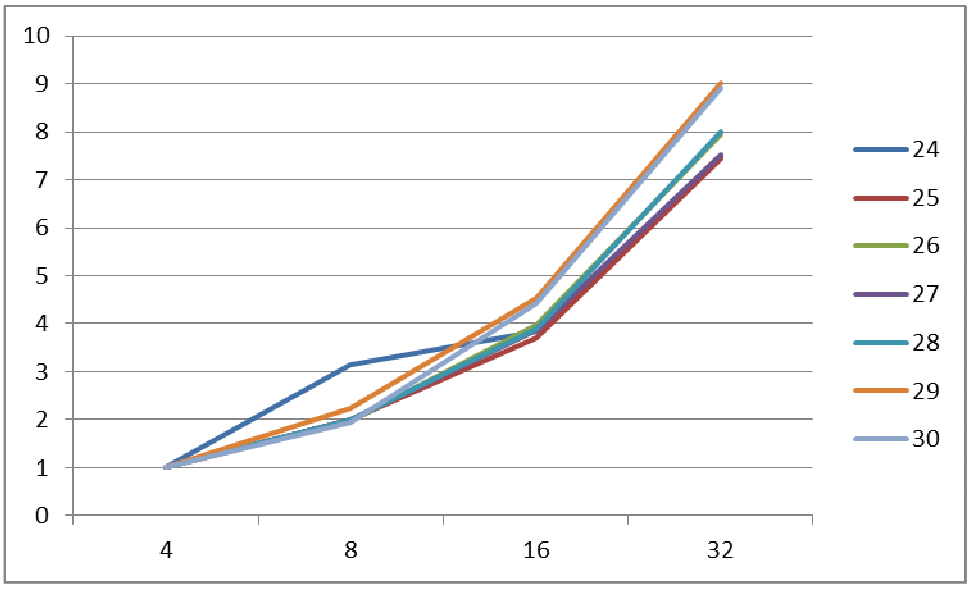

Рис. 6. График ускорения для однокубитного преобразования

На Рис. 6 показан график ускорения, полученного в этом эксперименте. Полученное ускорение практически линейное для всех рассмотренных размеров задачи. Таким образом, можно говорить о хорошей масштабируемости данной задачи как по отношению к размеру задачи, так и по отношению к увеличению количества процессоров. Эксперименты по достижению предела масштабируемости не проводились ввиду того, что для этого потребуется существенное время счета и фактически монопольный доступ на суперкомпьютер.

Далее был рассмотрен эксперимент по моделированию квантового преобразования Фурье. Результаты работы программы приведены в Таблице 2. В таблице приведено значение времени (в секундах) в зависимости от количества процессоров и количества моделируемых кубитов.

Таблица 2. Время работы квантового преобразования Фурье

QFT	**28**	**29**	**30**	**31**	**32**	**33**
8	17.4491	38.0417	90.2237	186.8465	SF	SF
16	8.5838	18.0346	40.3449	82.3250	192.9130	SF
32	4.3957	9.3806	19.5471	42.4178	97.0880	203.6334

Таблица 3. Время работы квантового алгоритма Гровера

Grover	**14**	**15**	**16**
1	10971.4132	SF	SF
2	5582.2765	SF	SF
4	2821.2334	17628.4555	SF
8	1452.7525	8876.5264	SF
16	746.7995	4499.1933	13932.3842
32	383.9903	2308.4533	6737.3245
64	198.3245	1274.4882	3367.0641

При моделировании квантового преобразования Фурье также была получена существенная масштабируемость приложения благодаря использованию механизма активных сообщений. При этом время моделирования пропорционально зависит от размера задачи.

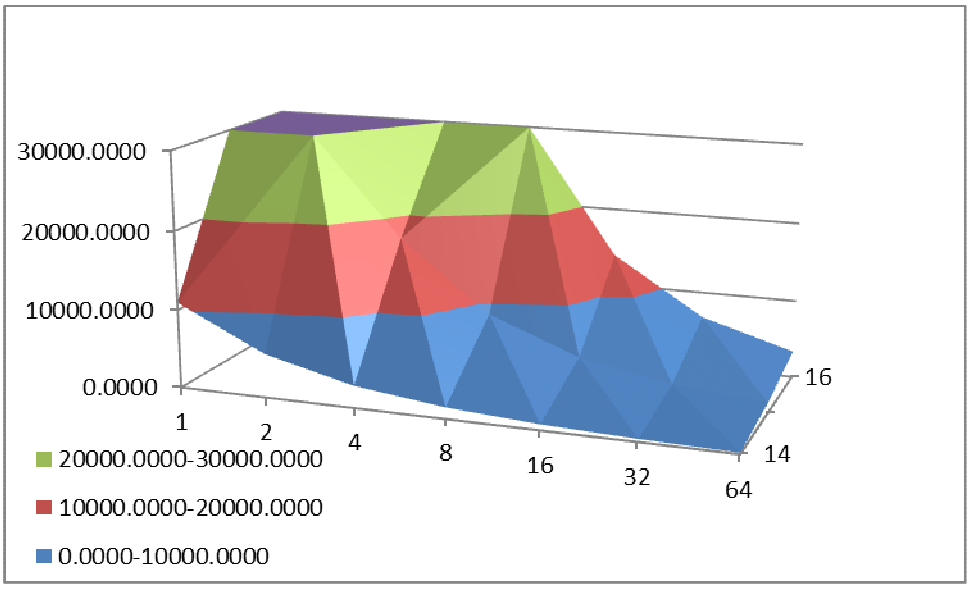

Рис. 6. График времени работы преобразования Гровера в зависимости от количества процессоров и размера задачи

В таблице 3 показано время работы программы по моделированию квантового алгоритма Гровера. Эта задача существенно более ресурсоемкая, как по времени работы, так и по количеству требуемой памяти. Максимально для моделирования использовалось 64 вычислительных узла, при этом максимально возможное количество моделируемых кубитов составило 16 (т.е. моделировалась работа 32-х кубитного квантового компьютера). Также в эксперименте наблюдается хорошая масштабируемость задачи (Рис.6).

Авторами также проводилось моделирование больших размеров задачи на большем числе вычислительных узлов. Максимально моделируемое число кубитов составило 39. Для этого использовалось 2048 вычислительных узлов суперкомпьютера Ломоносов. Однако на момент выхода этой статьи результаты эксперимента находятся в стадии верификации, поэтому численные результаты моделирования будут опубликованы позже.

6. Заключение

В работе представлено моделирование некоторых базовых алгоритмов квантовых вычислений на суперкомпьютере. Для моделирования было разработано программное обеспечение, использующее механизм передачи активных сообщений для межпроцессорных коммуникаций. Использована библиотека DISLIB. Проведено моделирование преобразования Адамара, квантового преобразования Фурье и квантового алгоритма Гровера. Во всех экспериментах показана хорошая масштабируемость разработанной параллельной программы. Дальнейшее развитие работы предполагает проведение моделирования задач большей размерности. Также предполагается моделирование квантовых алгоритмов с шумами.

Литература

1. Shor, Peter W. (1997), "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM J. Comput. 26 (5): 1484–1509
2. Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212
3. Ожигов Ю.И. Квантовые вычисления, М., Макс Пресс. 2003, -152с.
4. Нильсен М., Чанг И. Квантовые вычисления и квантовая информация: Пер с англ. МИР, 2006.

5. Корж А. А. Масштабирование Data-Intensive приложений с помощью библиотеки DISLIB на суперкомпьютерах Blue Gene/P и “Ломоносов”// Труды конференции “Научный сервис в сети Интернет-2011”, М: -изд.МГУ,сс.126-131.
6. Корж А.А. Результаты моделирования бенчмарка NBP UA на тысячи ядер суперкомпьютера BlueGene /P с помощью PGAS-расширения OpenMP // Вычислительные методы и программирование, том 11, раздел 2, 2010, сс.31-41.
7. J. Robert Burger, “New approachs to quantum computer simulaton in a classical supercomputer”, CoRR, Vol. Quant-ph/0308158 (2003)
8. Frank Tabakin, Bruno Juliá-Díaz, ”QCMPI: A parallel environment for quantum computing”, Computer Physics Communications №180, p. 948. 2009
9. Altschul S., Gish W., Miller W., Myers E., Lipman D. Basic local alignment search tool. Journal of Molecular Biology, vol. 215 (3), pp. 403–410, 1990
10. Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. LAPACK Users' Guide (Third ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.
11. ScaLAPACK web page http://www.netlib.org/scalapack/ (дата обращения 01.12.2012г.)
12. Guido Arnold, Thomas Lippert, Nikolas Pomplun, Marcus Richter, “Large Scale Simulation of Ideal Quantum Computers on SMP-Clusters”, PARCO , pp. 447-454, 2005
13. Goran Negovetic, Marek Perkowski, Martin Lukac, Andrzej Buller, “Evolving quantum circuits and an FPGA-based Quantum Computing Emulator”, International Worksho on Boolean Problems, 2002.
14. World record: German supercomputer simulates quantum computer http://phys.org/news189231849.html (дата обращения 01.12.2012г.)