Моделирование квантового преобразования Фурье с шумами на суперкомпьютере Ломоносов

Корж Оксана, Чернявский Андрей, Корж Антон

Московский государственный университет имени М.В.Ломоносова

Кафедра суперкомпьютеров и квантовой информатики

Введение

- Квантовая информатика является относительно молодой и очень бурно развивающейся областью современной науки.
- Из важнейших направлений квантовой информатики можно выделить квантовые вычисления, 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.
- Важно моделирование на основе реальных физических систем с учетом квантового шума, который является основным препятствием на пути реализации квантовых битов (кубитов) и вентилей.

О чем этот доклад?

- 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 состояний (фактически всей задействованной оперативной памяти). При этом доступ к данным осуществляется в произвольном порядке: порядок зависит от последовательности номеров кубитов, для которых выполняется преобразование.
- Моделирование квантовых вычислений – хорошая задача для анализа вычислительных возможностей суперкомпьютера
- Предлагается рассмотреть моделирование квантовых вычислений с шумами на суперкомпьютере с распределенным хранением данных. Использован метод активных сообщений

Квантовые вычисления

- Однокубитная операция

$$\{a_i\} = \{a_0, a_1, \ldots, a_{2^n-1}\} \qquad U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

$$b_{i_1 i_2 \ldots i_k \ldots i_n} = \sum_{j_k=0}^{1} u_{i_k j_k} a_{i_1 i_2 \ldots j_k \ldots i_n} = u_{00} a_{i_1 i_2 \ldots 0_k \ldots i_n} + u_{01} a_{i_1 i_2 \ldots 1_k \ldots i_n}$$

$$\begin{aligned} b_0 &= b_{\mathbf{00}} = u_{00} a_{\mathbf{00}} + u_{01} a_{\mathbf{10}} \\ b_1 &= b_{\mathbf{01}} = u_{00} a_{\mathbf{01}} + u_{01} a_{\mathbf{11}} \\ b_2 &= b_{\mathbf{10}} = u_{10} a_{\mathbf{00}} + u_{11} a_{\mathbf{10}} \\ b_3 &= b_{\mathbf{11}} = u_{10} a_{\mathbf{01}} + u_{11} a_{\mathbf{11}} \end{aligned}$$

Предыдущие работы

- *J. Robert Burger, “New approachs to quantum computer simulaton in a classical supercomputer”, CoRR, Vol. Quant-ph/0308158 (2003)*

- В статье 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.

Предыдущие работы

- *Guido Arnold, Thomas Lippert, Nikolas Pomplun, Marcus Richter, “Large Scale Simulation of Ideal Quantum Computers on SMP-Clusters”, PARCO , pp. 447-454, 2005*

- В статье 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.

- *World record: German supercomputer simulates quantum computer http:// phys.org/news189231849.html (дата обращения 01.12.2012г.)*

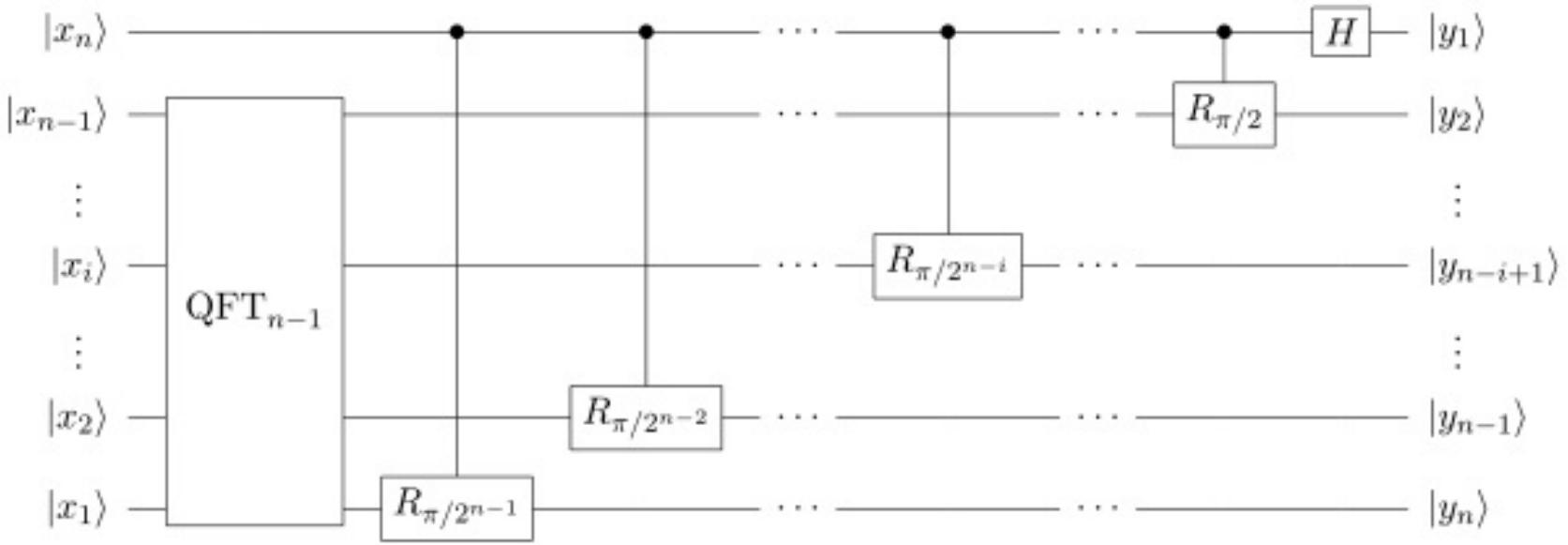
- В 2010 году было выполнено моделирование алгоритма Шора на суперкомпьютере Jugene в Суперкомпьютерном центре Юлиха. Пиковая производительность этого суперкомпьютера на тот момент составляла порядка одного петафлопса, объем оперативной памяти составлял порядка 140 терабайт. Удалось выполнить моделирование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

Предыдущие работы

- *Ю.И. Богданов, Н.А. Богданова, В.Ф. Лукичев, А.А. Орликовский, И.А. Семенихин, А.С. Холево , А.Ю. Чернявский Математическое моделирование влияния квантовых шумов на точность реализации квантовых алгоритмов // International Conference "Parallel and Distributed Computing Systems" PDCS 2013 (Ukraine, Kharkiv, March 13-14, 2013) , cc. 50-57*

- В статье описывается моделирование преобразования Фурье и алгоритма Гровера с учетом квантовых шумов. Рассмотрено моделирование на суперкомпьютере МВС-100К. При использовании 256 вычислительных узлов время выполнения преобразования Фурье составило 68 секунд при моделировании 28 кубитов. Предложенная реализация не обладает достаточных уровнем масштабируемости, что не позволяет провести эксперименты для более чем 32 кубитов.

Схема квантового преобразования Фурье



Модель шума

- Зашумленный вентиль Адамара H_e определяется следующими формулами:

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

- где e — уровень ошибки, N(0,1) — нормальная случайная величина с нулевым средним и единичной дисперсией.

- Зашумленное фазовое преобразование:

$$R_{ije} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \exp\left(\frac{2\pi i}{2^{j-i}} + ie\xi\right) \end{pmatrix}$$

Реализация параллельной версии алгоритма с помощью библиотеки DISLIB

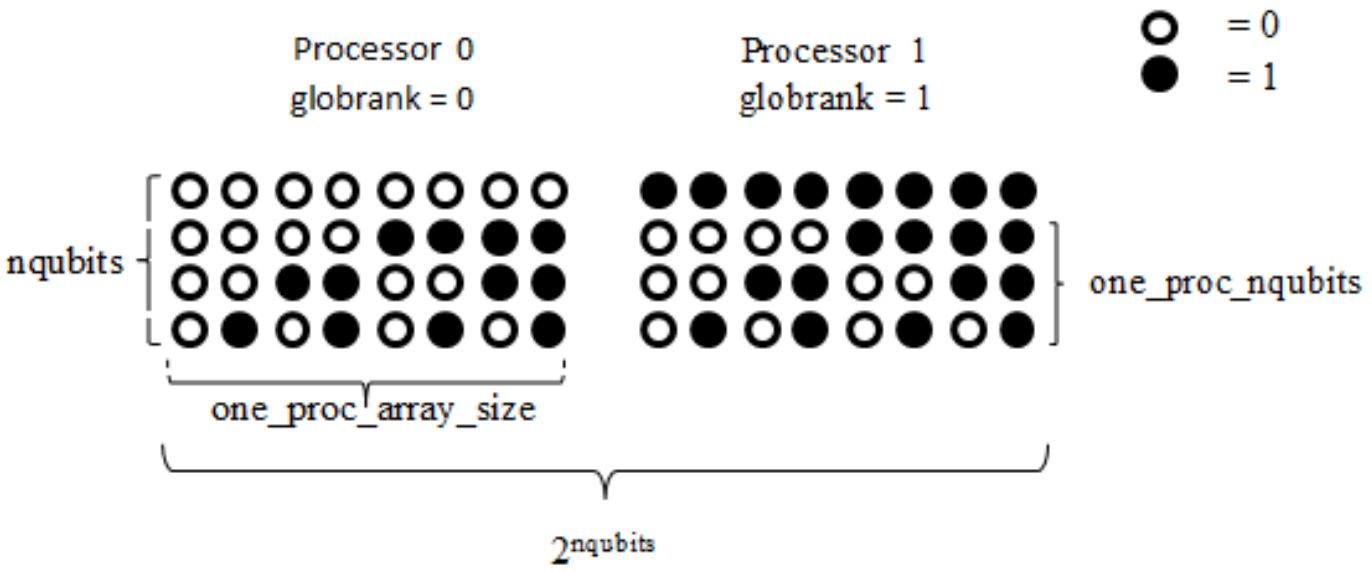
- Особенность квантовых вычислений с точки зрения реализации для параллельной вычислительной системы – необходимость постоянного нерегулярного доступа к данным, которые хранятся распределенно. Поэтому использование механизма односторонних активных сообщений представляется перспективным.

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

- Реализация на персональном компьютере – максимум 28 кубитов, реализация на суперкомпьютере Ломоносов – максимум 40-42 кубитов.

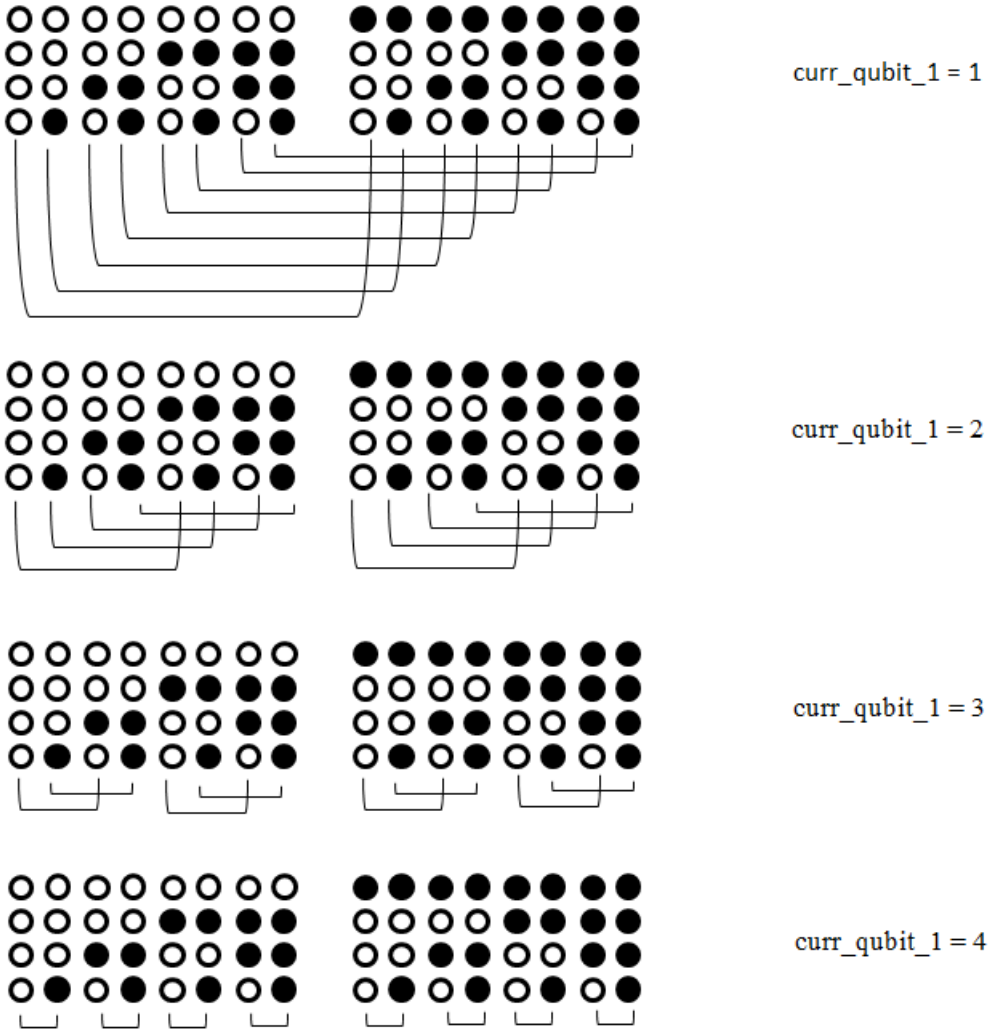
Реализация параллельной версии алгоритма с помощью библиотеки DISLIB

- Показан пример хранения вектора состояний. В примере показано хранение данных для случая двух процессоров и 4 кубитов.



Расположение данных по процессорам

- 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.



Реализация параллельной версии алгоритма с помощью библиотеки DISLIB

- 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 элемента.
- 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.

```
// 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];
                    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];
                    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();
}
```

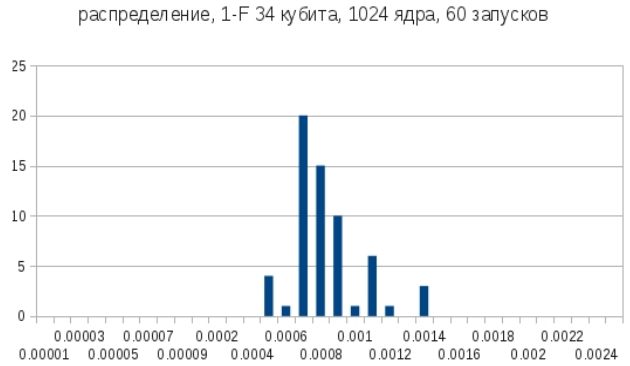
```
typedef struct elem_ind {
    complexd elem;
    long long ind;
}e_i;

// 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;
}
```

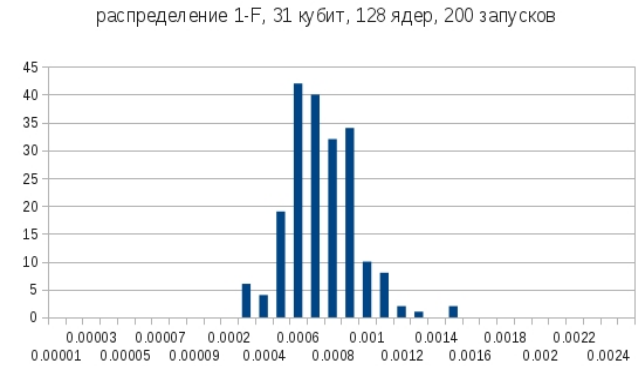
Тестирование на системе *Ломоносов*

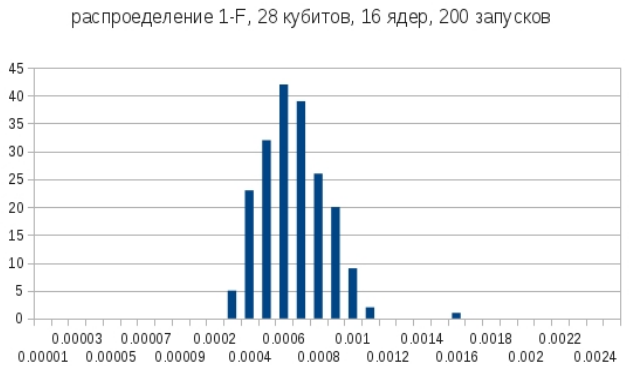
- Моделирование проводилось для различного числа кубитов на различном количестве процессоров. Максимально использовалось 8192 ядра, на которых выполнялось моделирование 37 кубитов. В качестве меры точности выбрана вероятность совпадения F(Fidelity) между идеальным и зашумленным векторами состояний. В качестве меры потери точности использовалась 1-F.

Тестирование на системе Ломоносов

- Приведены результаты моделирования на суперкомпьютере Ломоносов. Для 37 кубитов статистика построена по результатам 90 запусков, 34 кубита — 60 запусков, 31 и 28 кубитов — 200 запусков.
- Приведено распределение потерь точности 1-F для алгоритма квантового преобразования Фурье с различным числом кубитов . Точность e=0.01.







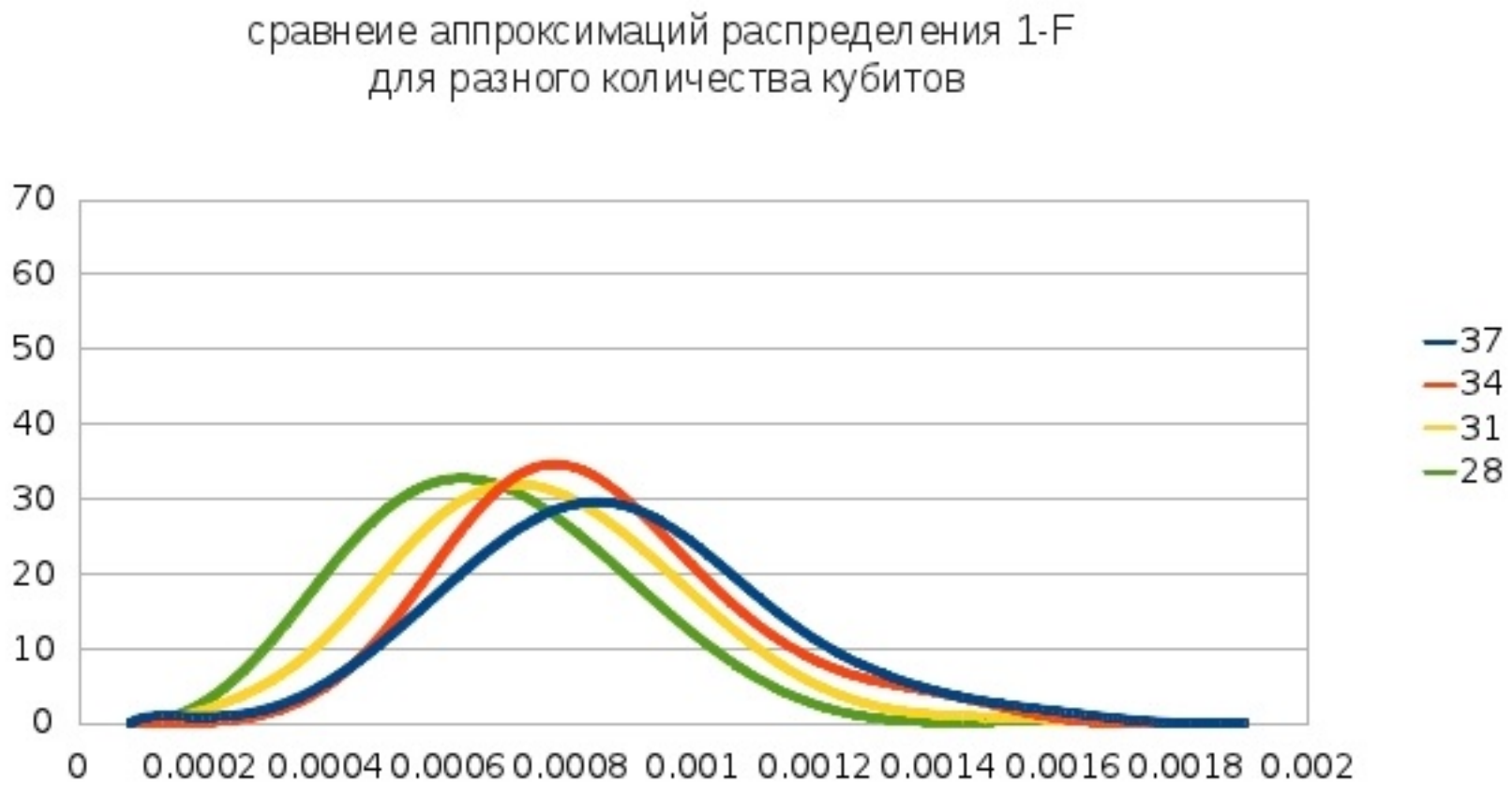
Тестирование на системе Ломоносов

- Приведены значения средних потерь для различного числа кубитов

Количество кубитов	Количество ядер	Средние потери точности
28	16	0.00070762
31	128	0.00079261
32	256	0.00082287
34	1024	0.00088478
37	8192	0.00090877

Тестирование на системе Ломоносов

- Сравнение аппроксимаций 1-F б-сплайнами для различного количества кубитов.



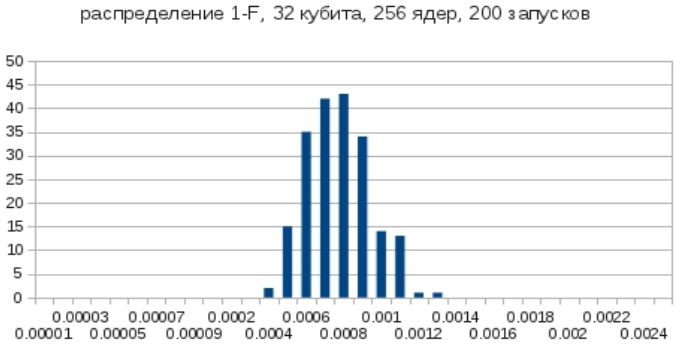
Тестирование на системе Ломоносов

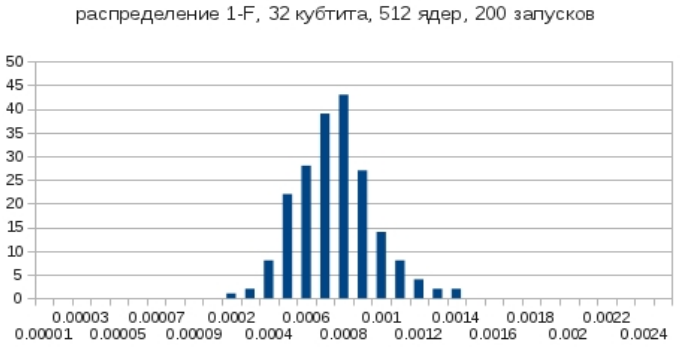
- Приведено среднее время работы алгоритма для различного числа кубитов. Количество процессоров бралось минимальным с условием, что задача помещалась в память.

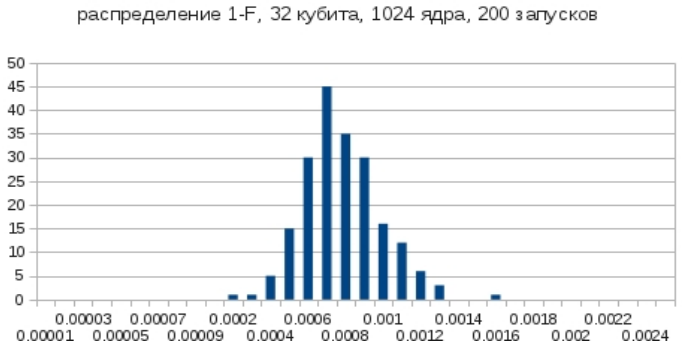
Количество кубитов	Количество ядер	Среднее время работы (сек)
28	16	61
31	128	71
32	256	81
34	1024	112
37	8192	*161

Тестирование на системе Ломоносов

- Показаны результаты моделирования для фиксированного количества кубитов (32) и различного количества процессоров
- Приведено распределение потерь точности 1-F для алгоритма квантового преобразования Фурье с различным количеством процессоров. Точность e=0.01.

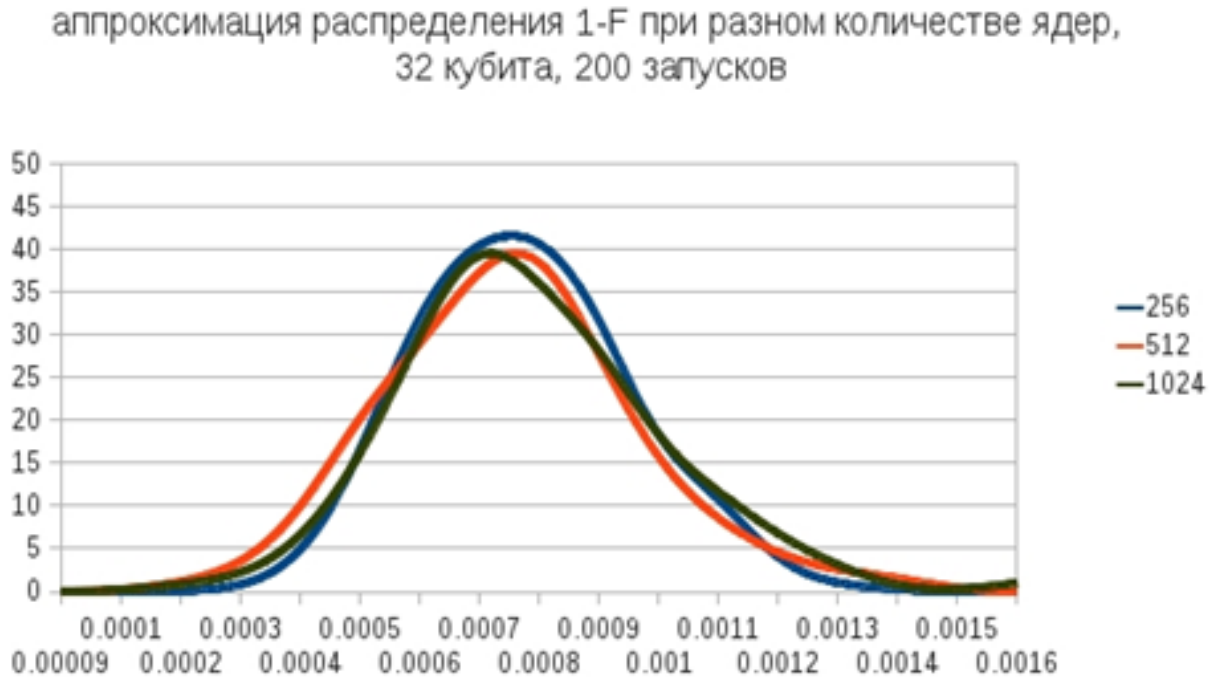






Тестирование на системе Ломоносов

- Сравнение аппроксимаций 1-F б-сплайнами для различного количества процессоров.



Тестирование на системе Ломоносов

- Приведены значения средних потерь для 32 кубитов при расчёта на разном количестве процессоров.

Количество кубитов	Количество ядер	Средние потери точности
32	256	0.00082287
32	512	0.00080265
32	1024	0.00083367

Тестирование на системе Ломоносов

- Время работы зашумленного квантового преобразования Фурье для фиксированного размера задачи (32 кубита) и разного количества процессоров

Количество ядер	Среднее время работы (сек)
256	81
512	42
1024	22

Заключение и дальнейшие планы

- В работе представлены результаты моделирования зашумленного квантового преобразования Фурье на суперкомпьютере Ломоносов. Показано, что распределение потерь точности близко к нормальному. Так же показано, что точность в среднем уменьшается при увеличении количества кубитов. Изменение количества процессоров при фиксированном количестве кубитов в среднем не влияет на точность. Проведены результаты моделирования квантового преобразования Фурье с зашумленными вентилями для 37 кубитов.

- Работа выполнена при финансовой поддержке РФФИ (гранты 11–07–00756, 12–07–31229, 12–01–31274). Авторы выражают благодарность сотрудникам лаборатории физики квантовых компьютеров ФТИАН за плодотворное обсуждение вопросов, связанных с тематикой работы.

Спасибо за внимание

sqi.cs.msu.ru