УДК 004.272.2, 530.145 **DOI: 10.14529/cmse210104**

СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПАКЕТОВ СИМУЛЯЦИИ КВАНТОВЫХ ВЫЧИСЛЕНИЙ QUEST И INTEL-QS*

© 2021 А.В. Линев[1], П.Е. Ведруков[1], Д.С. Куландин[1], И.Б. Мееров[1], С. Денисов[1,2]

[1]*Нижегородский государственный университет им. Н.И. Лобачевского (603950 Нижний Новгород, пр. Гагарина, д. 23),*

[2]*Столичный университет Осло (NO-0130 Осло, Норвегия, ул. Olavs plass, P.O. Box 4)*

E-mail: alin@unn.ru, pavelvedrukov@mail.ru, kulandin08@gmail.com, meerov@vmk.unn.ru, sergiyde@oslomet.no

Поступила в редакцию: 14.09.2020

В ближайшем будущем появятся квантовые компьютеры, пригодные для практического использования. Разработка квантовых алгоритмов может проводиться с использованием классических компьютеров и специализированного программного обеспечения, позволяющего симулировать работу квантовой схемы. Результаты моделирования могут использоваться для анализа алгоритма, а также способствуют ко-дизайну при разработке квантовых архитектур. Однако при планировании и выполнении численных экспериментов необходимо понимать возможности симуляторов и ограничения на параметры квантовой схемы, накладываемые характеристиками доступных классических вычислительных ресурсов. В работе представлены результаты вычислительных экспериментов по симуляции работы квантовых схем на идеальном квантовом компьютере с использованием пакетов QuEST и Intel-QS, а также собственной «наивной» реализации. Показаны ограничения на размер моделируемой квантовой системы N при использовании вычислительных систем различного класса — виртуальной машины, вычислительного сервера, вычислительного сервера с графическим ускорителем, суперкомпьютера (маскимальный достигнутый размер $N = 33$). Приведены характеристики производительности и масштабируемости рассматриваемых реализаций на общей и распределенной памяти (наблюдаемая эффективность масштабирования — 30 % и 70 % соответственно). Для пакета QuEST и собственной реализации представлена производительность при использовании графических сопроцессоров.

Ключевые слова: вычислительная квантовая физика, квантовые алгоритмы, высокопроизводительные вычисления, GPGPU, QuEST, Intel-QS.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Линев А.В., Ведруков П.Е., Куландин Д.С., Мееров И.Б., Денисов С. Сравнение производительности пакетов симуляции квантовых вычислений QuEST и Intel-QS // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 49–61. DOI: 10.14529/cmse210104.

Введение

Важной составляющей разработки и анализа квантовых алгоритмов [1] является симуляция их выполнения на классических компьютерах [2]. Разработаны десятки приложений [3], которые позволяют описывать квантовые схемы с использованием специализированных языков квантовых вычислений [4–7], выполнять симуляцию работы квантового алгоритма [8–11] и даже поддерживать полный цикл разработки, включающий описание алгоритма, его оптимизацию, отображение на конкретную архитектуру кван-

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2020».

тового компьютера, верификацию, симуляцию работы полученной квантовой схемы и оценку ее производительности [12].

Симуляция возможна только для квантовых систем, состоящих из малого числа кубитов. Даже использование суперкомпьютеров позволяет симулировать работу квантовых систем, состоящих всего лишь из 38 [8], 42 [9], 45 [13], 49 [14], возможно, 53 [15], а в специальных случаях — до 64 кубитов [16], в то время как Google уже представляет результаты экспериментов на реальном квантовом компьютере общего назначения из 53 кубитов [17], анонсирует квантовый процессор из 72 кубитов [18], а D-Wave systems — специализированную систему из 2000 кубитов, реализующую алгоритм квантового отжига [19].

Несмотря на различную демонстрируемую производительность [20], разнообразие приложений позволяет выбрать наиболее подходящее для конкретной задачи и имеющихся вычислительных ресурсов. При выборе программных средств моделирования квантовых вычислений мы провели ряд экспериментов для определения производительности различных пакетов по выполнению симуляции работы квантовой схемы на идеальном квантовом компьютере [2] на системах различного класса — от настольного компьютера до вычислительного кластера. Мы рассматривали пакеты, которые, с одной стороны, позволяют оценить потребление ресурсов при выполнении отдельных шагов моделирования, с другой стороны — имеют параллельные версии для общей и распределенной памяти. Множество пакетов не удовлетворяют одному из данных требований (IBM Qiskit, LIQ$Ui|\rangle$, ProjectQ, DG-M и т.д.) или недоступны для свободного использования [21]. Мы выбрали для анализа два пакета, удовлетворяющие нашим требованиям и положительно характеризуемые в публикациях — QuEST и Intel-QS. Полученные результаты будут использованы для формирования набора инструментов, которые будут использоваться для разработки и анализа квантовых алгоритмов и обучения студентов и аспирантов ННГУ им. Н.И. Лобачевского.

Работа построена следующим образом. В разделе 1 приведено описание тестовой задачи. В разделе 2 — описание проведенных экспериментов. Раздел 3 содержит характеристики вычислительных систем. Раздел 4 — результаты экспериментов и комментарии к ним. В последнем разделе представлено обобщение результатов.

1. Тестовая задача

Квантовый компьютер хранит информацию в виде кубитов, которые можно считать квантовой версией битов. Базовые операции над кубитами включают квантовые гейты — аналоги классических битовых операций — и операцию измерения кубита.

Мы будем рассматривать симуляцию идеального квантового компьютера, в котором отсутствуют шумы и декогеренция, присущие реальным устройствам. В этом случае состояние квантового компьютера с N кубитами описывается квантовым состоянием (также называемым «волновой функцией») системы кубитов, представленным в виде вектора комплексных чисел размером 2^N (a_i), $i \in \{0,1\}^N$, $a_i \in \mathbb{C}$, удовлетворяющего условию нормировки $\sum_{i\in\{0,1\}^N}|a_i|^2 = 1$.

Действие квантового гейта в общем случае вызывает изменение всех 2^N элементов вектора, согласно природе физического процесса описывается унитарным преобразованием и может быть представлено унитарной матрицей. Квантовые компьютеры реализуют набор базовых гейтов, управляющих состоянием одного–трех кубитов и представляемых сильно разреженной матрицей преобразования системы в целом (комбинация кронекеровских произведений матрицы воздействия на кубиты и единичных матриц). Для повышения производительности симуляция квантовых гейтов обычно выполняется

не универсальным умножением разреженной матрицы на плотный вектор, а реализацией специализированного алгоритма, уникального для каждого типа квантового гейта.

В тестовом примере мы использовали однокубитовый гейт преобразования Адамара (H), двухкубитовый гейт контролируемого отрицания ($CNOT$) и вычисление вероятности измерения нулевого кубита в состояниях $|0>$ и $|1>$ (P_0 и P_1 соответственно), которые описываются следующими матрицами и формулами:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, C\text{--}NOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}, P_0 = \sum_{i_1,\ldots,i_{N-1}} a_{0,i_1,\ldots,i_{N-1}}, P_1 = \sum_{i_1,\ldots,i_{N-1}} a_{1,i_1,\ldots,i_{N-1}}.$$

Первый из них воздействует на один кубит и требует вычислений с использованием и обновлением всех элементов квантового состояния, второй — воздействует на два кубита и не предполагает выполнения вычислений. Для выполнения одной операции Адамара для кубитной системы требуется выполнение 2^N умножений и 2^N сложений комплексных чисел, т.е. $8 * 2^N$ FLOP. При этом требуется считать из памяти $16 * 2^N$ байт и записать в память $16 * 2^N$ байт (например, для $N = 30$ и 50 операций — 400 GFLOP и 32 ГБ). Для выполнения одной операции контролируемого отрицания требуется выполнить 0 FLOP, считать из памяти $8 * 2^N$ байт и записать $8 * 2^N$ байт (для $N = 30$ и 50 операций — 0 GFLOP и 16 ГБ). Фактическое число операций и объем передаваемых данных зависят от конкретной архитектуры вычислительной системы.

При выполнении экспериментов производилась симуляция выполнения на системе из N кубитов квантовой схемы, состоящей из 50 операций Адамара и 50 операций контролируемого отрицания. Рассматривались следующие этапы симуляции:

- инициализация программного окружения ПО симуляции;
- создание системы из N кубитов;
- последовательное применение оператора Адамара к кубитам 0, 1,..., $N-1$, 0, 1,... (далее по циклу); общее число операций — 50;
- последовательное применение оператора CNOT к парам кубитов (control=0, target=1), (1,2),..., $(N-2, N-1)$, $(N-1, 0)$, (0,1),... (далее по циклу); общее число операций — 50;
- вычисление вероятностей нахождения нулевого кубита в состояниях $|0>$ и $|1>$;
- удаление системы кубитов;
- завершение программного окружения ПО симуляции.

Оценивались время работы и потребление оперативной памяти на всех этапах.

2. Описание экспериментов

Для оценки характеристик программных средств симуляции работы квантовой схемы мы использовали 3 реализации:

- QuEST — симулятор квантовых компьютеров с поддержкой параллельного выполнения на общей и распределенной памяти [22], при работе на одном узле способен использовать один графический ускоритель;
- Intel Quantum Simulator (Intel-QS) — симулятор, оптимизированный для многоядерных архитектур и вычислительных кластеров [23];
- «наивная» реализация, выполненная нами (не использует векторизацию, оптимизацию использования кеша и т.д.); используется для оценки производительности качественных реализаций.

Все пакеты компилировались и запускались с параметрами по умолчанию.

Проводились следующие виды экспериментов.

1. Определение зависимости времени работы и объема используемой оперативной памяти от числа кубитов. Выполнялись тесты для числа кубитов от 2 до максимально возможного на вычислительной системе. На многоядерных системах использовались все ядра. Измерялись время работы каждого этапа и объем памяти, используемый процессом в конце этапа.

2. Масштабируемость на общей памяти. Использовалось 30 кубитов и число потоков OpenMP от 1 до максимально возможного на вычислительной системе. Измерялись время работы каждого этапа и максимальный объем памяти, используемый процессом.

3. Масштабируемость на распределенной памяти. Использовалось 30 кубитов и число MPI-процессов от 1 до максимально возможного на вычислительной системе. Измерялись время работы каждого этапа и объем памяти, используемый одним процессом в конце этапа.

4. Определение зависимости времени работы от числа кубитов при использовании графического ускорителя. Измерялось полное время симуляции.

На различных вычислительных системах выполнялся различный состав экспериментов. На системах 1–3 (см. раздел 3) проводились все возможные для них виды экспериментов. На системе 4 проводился только четвертый эксперимент.

3. Вычислительные системы

Для оценки возможности использования программных средств симуляции работы квантовой схемы мы использовали 4 вычислительные системы, существенно отличающиеся с точки зрения производительности.

1. Виртуальная машина (VM), запущенная на рабочей станции.

– Конфигурация: Intel Core i3-7100 (Kaby Lake), 3,90 ГГц, 1 ядро, 1 ГБ DDR4-2132 (1066 МГц).
– Производительность: 62,4 GFLOPS/ядро, 17 064 МБ/с * 2 канала.
– Интерконнект: предоставлямый средством виртуализации.
– Операционная система: CentOS 7.4.1708.
– Компилятор: gcc 4.8.5-39, gcc 7.3.1-5.
– MPI: mpich-3.0.

2. Dell PowerEdge R815 (64-ядерный сервер).

– Конфигурация: 4xAMD Opteron 6366 HECPU, 3,60 ГГц, 4x16 ядер, 256 ГБ DDR3-1600.
– Производительность: 921,6 GFLOPS (14,4/ядро), 12 800 МБ/с * 4 канала* 4 ЦП (До 1600 МТ/с, 140 ГБ/с согласно описанию платформы).
– Интерконнект: 4xBroadcom 5709C (Gigabit Ethernet).
– Операционная система: Ubuntu 18.04.1 LTS.
– Компилятор: gcc7.3.0-27.
– MPI: OpenRTE 3.0.0 (openmpi-based)

3. Intel Endeavour (вычислительный кластер)

– Конфигурация: 2xIntel Xeon Platinum 8260L (Cascade Lake), 2,40 ГГц, 2x24 физических ядер, 256 ГБ DDR4-2933.
– Производительность: 2 304 GFLOPS (48/ядро), 23 466 МБ/с * 6 каналов * 2 ЦП.
– Интерконнект: Mellanox Technologies MT28908 Family (HDR Infiniband).
– Операционная система: CentOS Linux 7 (Core).
– Компилятор: icc 19.0.5.281 20190815.
– MPI: Intel(R) MPI Library for Linux OS, Version 2019 Update 5 Build 20190806.

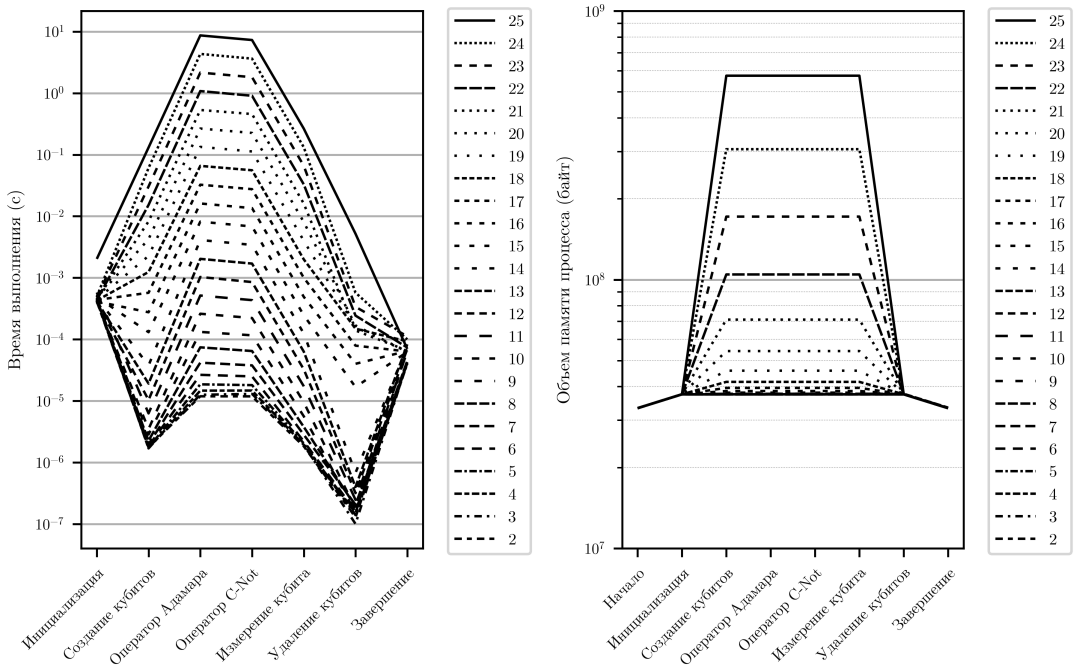
4. DGX-2 (сервер с графическими ускорителями).

- Конфигурация: 2xIntel Xeon Platinum 8168 (Skylake), 2,70 ГГц, 2x24 физических ядер, 1,5 ТБ DDR4-2666.
- Производительность: 2 918,4 GFLOPS (60,8/ядро), 21 333 МБ/с * 6 каналов * 2 ЦП.
- Графические сопроцессоры: 16xNVIDIA Tesla V100-SXM3-32GB, 16*32 ГБ HBM2.
- Производительность графических сопроцессоров: 124,8 TFLOPS DP (7,8/GPU), 900 ГБ/с * 16 GPU.
- Интерконнект: 8x Mellanox Technologies MT27800 Family (100 Гб/с Infiniband).
- Операционная система: Ubuntu 18.04.5 LTS (Bionic Beaver).
- Компилятор: gcc 7.5.0, Cuda compilation tools, release 10.1, V10.1.243.
- MPI: нет.

4. Результаты экспериментов

4.1. Виртуальная машина

В виртуальной машине проводился только первый эксперимент для квантовых систем размером от 2 до 25 кубитов. На рис. 1 приведены результаты для пакета QuEST. Здесь и далее все графики зависимости измеряемых значений от числа кубитов приведены в логарифмической системе координат.



а) Время выполнения этапов б) Объем памяти на этапах

Рис. 1. Зависимость времени работы этапов алгоритма и объема используемой оперативной памяти от числа кубитов; пакет QuEST; виртуальная машина

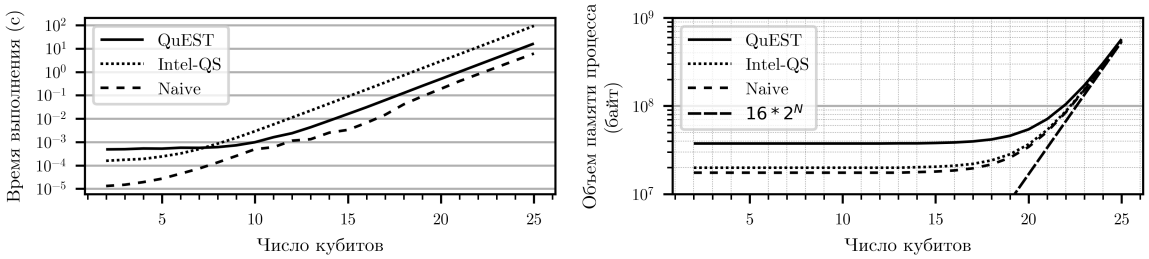
Наблюдается экспоненциальная зависимость времени выполнения содержательных этапов от числа кубитов (отличия по времени для малого числа кубитов могут быть связаны с особенностями архитектуры системы, например, с размещением всех данных в кеше). Объем потребляемой памяти приближенно определяется как сумма константы (код и данные за исключением квантового состояния) и $16 * 2^N$ байт (размер вектора квантового состояния).

Время выполнения операций инициализации и завершения не зависит от числа кубитов. Удаление кубита требует на 2–4 порядка меньше времени, чем остальные содержательные операции. Создание и измерение кубита требует в несколько десятков раз

меньше времени, чем выполнение операторов. Оператор *CNot* выполняется в 2–3 раза быстрее оператора Адамара.

Для других экспериментов и типов архитектур в большинстве случаев эти соотношения качественно сохраняются, поэтому далее мы их приводить не будем.

На рис. 2 приведены результаты экспериментов для различных пакетов. Используется суммарное время выполнения всех этапов и максимальное потребление оперативной памяти.



а) Общее время расчета б) Необходимый объем памяти

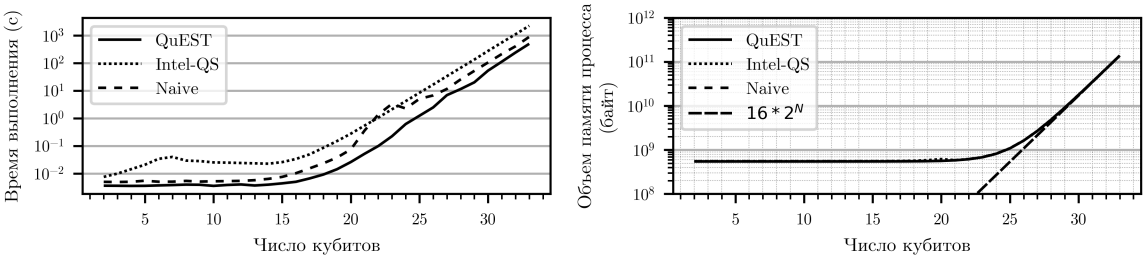
Рис. 2. Выполнение расчетов для различного числа кубитов; виртуальная машина

Экспоненциальная зависимость времени выполнения и объема потребляемой памяти от числа кубитов сохраняется, причем объем памяти фактически определяется размером вектора квантового состояния.

«Наивная» реализация показала более высокую производительность, чем другие пакеты. Таким образом, при моделировании простых квантовых схем собственные реализации могут показывать производительность, сравнимую с универсальными профессиональными пакетами.

4.2. Сервер Dell PowerEdge R815

На этом 64-ядерном сервере проводились эксперименты 1 и 2. На рис. 3 приведены результаты первого эксперимента для квантовых систем размером от 2 до 33 кубитов.



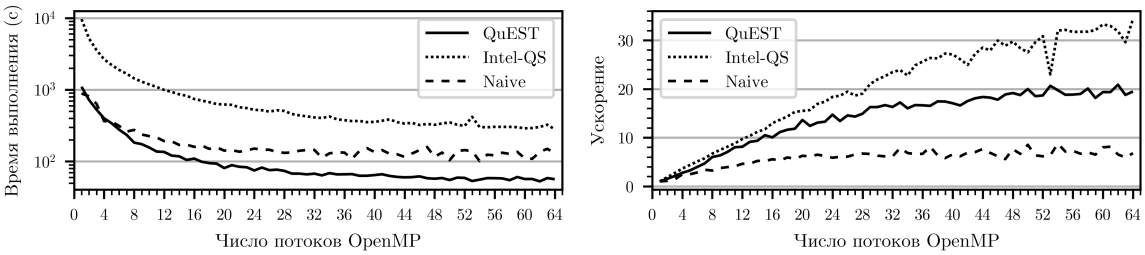
а) Общее время расчета б) Необходимый объем памяти

Рис. 3. Выполнение расчетов для различного числа кубитов; Dell PowerEdge R815

Скорость работы пакетов может отличаться в 4–5 раз. Низкая производительность Intel-QS по-видимому объясняется оптимизацией его кода для архитектур и компиляторов Intel.

Результаты эксперимента 2, оценивающего сильную масштабируемость при распараллеливании на общей памяти, приведены на рис. 4.

Эффективность распараллеливания пакета QuEST, показавшего лучшие результаты на данной системе, составляет около 30 %. Базовые квантовые гейты реализуются алгоритмами с низкой арифметической интенсивностью, поэтому их производительность ограничивается возможностями оперативной памяти, а не числом одновременно работающих потоков. Для более сложных схем продемонстрирована возможность достижения эффективности распараллеливания более 80 % и производительности в 47 % от теоретической [14].

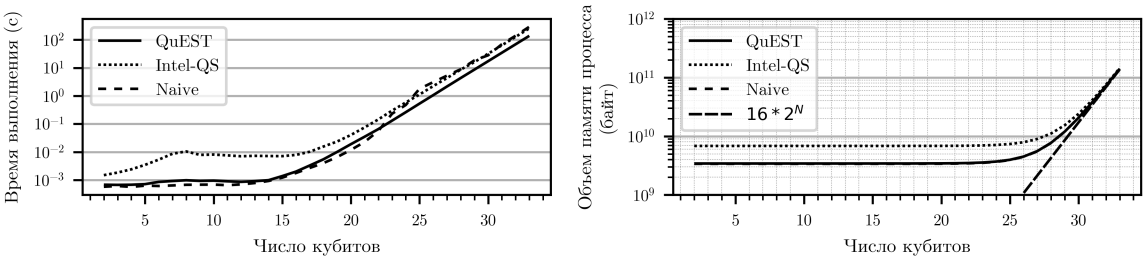


а) Общее время расчета б) Ускорение на общей памяти

Рис. 4. Производительность пакетов на системе с общей памятью; Dell PowerEdge R815

4.3. Вычислительный кластер Intel Endeavour

На вычислительном кластере Intel Endeavour проводились эксперименты 1–3. На рис. 5 приведены результаты первого эксперимента для квантовых систем размером от 2 до 33 кубитов.



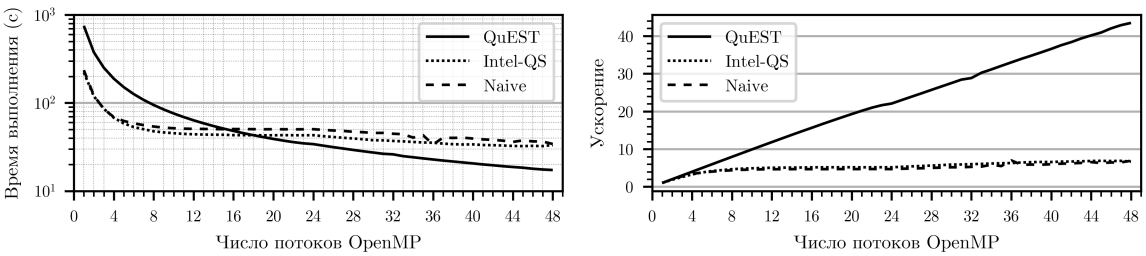
а) Общее время расчета б) Необходимый объем памяти

Рис. 5. Выполнение расчетов для различного числа кубитов; 1 узел Intel Endeavour

Для числа кубитов больше 20 скорость работы различных реализаций отличается не более чем в 2 раза.

На рис. 6 показаны результаты эксперимента 2, оценивающего сильную масштабируемость при распараллеливании на общей памяти.

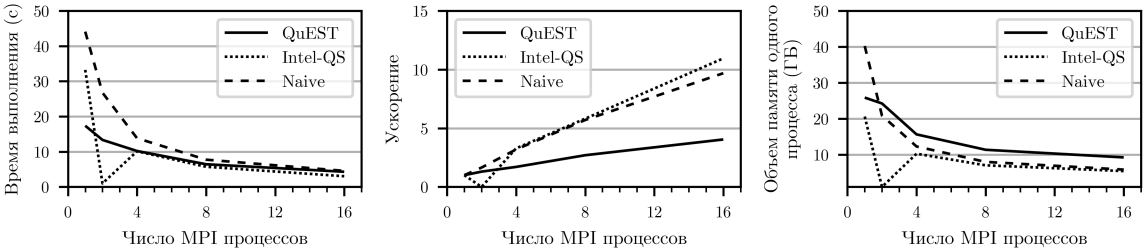
QuEST показывает отличную эффективность распараллеливания (≈90 %), остальные пакеты — очень низкую (≈15 %). Для «наивной» реализации это объясняется отсутствием оптимизаций. Для Intel-QS — высокой производительностью однопоточной реализации и однопоточной реализацией операции вычисления вероятности нахождения кубита в одном из состояний. Без учета этой операции эффективность масштабирования Intel-QS составляет ≈23 %, а время выполнения при максимальном числе потоков больше чем у QuEST всего на 9 %.



а) Общее время расчета б) Ускорение на общей памяти

Рис. 6. Производительность пакетов на системах с общей памятью; 1 узел Intel Endeavour

На рис. 7 показаны результаты эксперимента 3 по изучению сильной масштабируемости при распараллеливании на распределенной памяти. Intel-QS не отработал при запуске с одним и двумя MPI-процессами. В целях оценки масштабируемости, для одного процесса приведены данные версии для общей памяти.



а) Общее время расчета б) Ускорение в) Объем памяти

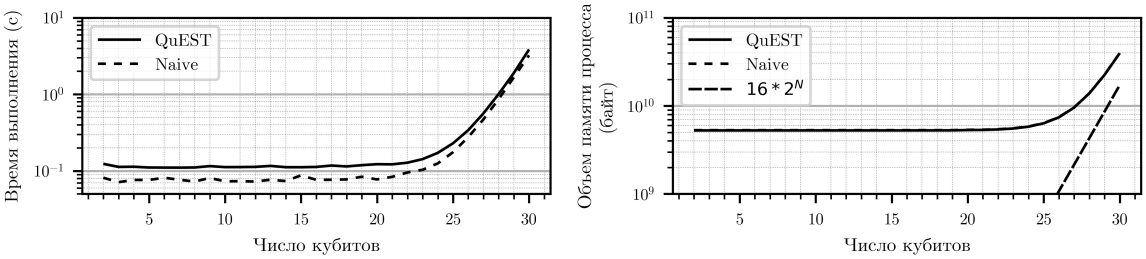
Рис. 7. Характеристики выполнения пакетов на системе с распределенной памятью; Intel Endeavour

Все реализации показывают сравнимую производительность, но QuEST показывает в 2,5 раза худшую масштабируемость. Время выполнения эксперимента слишком мало для точной оценки производительности, но при увеличении числа кубитов пакеты на доступных нам ресурсах перестают отрабатывать корректно для многих сочетаний параметров эксперимента. Эффективность QuEST существенно отличается от заявляемой авторами; возможно, необходимо подбирать параметры его запуска или использовать более сложные квантовые схемы.

Объем памяти, используемой одним MPI-процессом, уменьшается примерно пропорционально их числу. Удвоение числа процессов позволяет сократить объем памяти каждого процесса примерно в два раза или использовать тот же объем для моделирования системы размером на один кубит больше.

4.4. Сервер с графическими ускорителями DGX-2

На данном сервере проводился эксперимент 4. Из рассматриваемых пакетов только QuEST поддерживает использование графических ускорителей и способен использовать только один ускоритель, поэтому мы выполнили «наивную» реализацию для одного GPGPU и сравнили производительность. На рис. 8 приведены результаты четвертого эксперимента для квантовых систем размером от 2 до 30 кубитов (30 — максимальный размер системы кубитов, состояние которой можно разместить в памяти одного GPU).



а) Общее время расчета б) Необходимый объем памяти ЦП

Рис. 8. Выполнение расчетов для различного числа кубитов; 1 GPU DGX-2

Производительность QuEST и «наивной» реализации сопоставимы, объем используемой памяти ЦП отличается менее чем на 1 %.

Время работы QuEST для 27–30 кубитов примерно в 4 раза меньше, чем на одном узле Intel Endeavour при использовании максимального числа потоков. В нашем экспе-

рименте скорость вычислений ограничена производительностью работы с памятью, и отличие показателей ЦП и GPU примерно соответствует отношению теоретической пропускной способности памяти использованных систем, а полученные фактические значения говорят о возможности дополнительной оптимизации работы с памятью. Время работы наивной реализации для GPU меньше в 9–11 раз, но это объясняется низкой производительностью параллельной версии для ЦП. Можно предположить, что использование нескольких GPU будет показывать масштабируемость, сравнимую с MPI-версиями, а в случае использования нескольких GPU одного узла — даже более высокую, что позволит моделировать на сервере DGX-2 системы из 32–33 кубитов с производительностью, сопоставимой с производительностью более 50 двухпроцессорных узлов без GPU.

Заключение

Результаты проведенных экспериментов позволяют сделать следующие выводы относительно возможностей симуляции работы квантовой схемы на идеальном квантовом компьютере с использованием различных вычислительных систем.

- Время моделирования экспоненциально зависит от числа кубитов.
- Объем необходимой памяти экспоненциально зависит от числа кубитов и будет являться основным ограничивающим фактором при его увеличении.
- Масштабируемость на общей памяти имеет среднюю эффективность (≈30 %), но известно, что на более сложных квантовых схемах можно достичь эффективности более 80 %.
- Эффективность масштабируемости на распределенной памяти достигает 70 %.
- Использование распределенной памяти позволяет увеличить размер моделируемых систем. При удвоении числа узлов можно симулировать систему размером на 1 кубит больше.
- Графические сопроцессоры могут обеспечить значительный рост производительности (8–9 раз по сравнению с ЦП), но критически ограничены объемом установленной на них оперативной памяти.
- Переход от использования виртуальной машины с 1 Гб оперативной памяти к суперкомпьютеру «Ломоносов-2» [24] позволит увеличить размер моделируемой системы с 25 до 40 кубитов, что не имеет принципиального значения при разработке, отладке и тестировании квантовых схем.
- «Наивные» реализации могут быть использованы для моделирования простых квантовых схем и в целях обучения. Исследование больших и сложных схем может потребовать профессиональных пакетов.
- В наших экспериментах Intel-QS и QuEST показали сравнимую производительность, лучший результат на распределенной памяти достигнут Intel-QS.

Полученные результаты будут использованы для формирования набора инструментов, которые будут использоваться для разработки и анализа квантовых алгоритмов и обучения студентов и аспирантов ННГУ им. Н.И. Лобачевского.

Исследования выполнены при поддержке гранта РНФ № 19-72-20086 с использованием вычислительных ресурсов СК Intel Endeavor и Simula Research Laboratory.

Литература

1. Mermin N.D. Quantum computer science: an introduction. Cambridge University Press, 2007. 233 p.
2. Trieu D.B. Large-scale simulations of error prone quantum computation devices. Forschungszentrum Jülich, 2010. Vol. 2. 173 p.
3. Wiki Q. List of QC simulators. 2015. URL: https://quantiki.org/wiki/list-qc-simulators (дата обращения: 01.09.2020).
4. Green A.S., Lumsdaine P.L., Ross N.J., et al. Quipper: a scalable quantum programming language // Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation. 2013. P. 333–342. DOI: 10.1145/2491956.2462177.
5. Cross A.W., Bishop L.S., Smolin J.A., et al. Open quantum assembly language // arXiv preprint. 2017. arXiv:1707.03429.
6. Svore K., Geller A., Troyer M., et al. Q# Enabling Scalable Quantum Computing and Development with a High-level DSL // Proceedings of the Real World Domain Specific Languages Workshop 2018. 2018. P. 1–10. DOI: 10.1145/3183895.3183901.
7. Abhari A.J., Faruque A., Dousti M.J., et al. Scaffold: Quantum programming language. Princeton Univ NJ Dept of Computer Science, 2012. 43 p.
8. Jones T., Brown A., Bush I., et al. Quest and high performance simulation of quantum computers // Scientific reports. 2019. Vol. 9, no. 1. P. 1–11. DOI: 10.1038/s41598-019-47174-9.
9. Guerreschi G.G., Hogaboam J., Baruffa F., et al. Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits // Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034007. DOI: 10.1088/2058-9565/ab8505.
10. Smelyanskiy M., Sawaya N.P.D., Aspuru-Guzik A. qHiPSTER: the quantum high performance software testing environment // arXiv preprint. 2016. arXiv:1601.07195.
11. Aleksandrowicz G., Alexander T., Barkoutsos P., et al. Qiskit: An open-source framework for quantum computing. 2019. DOI: 10.5281/zenodo.2562110.
12. Amy M., Gheorghiu V. staq–A full-stack quantum processing toolkit // Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034016. DOI: 10.1088/2058-9565/ab9359.
13. Häner T., Steiger D.S. 5 petabyte simulation of a 45-qubit quantum circuit // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2017. P. 1–10. DOI: 10.1145/3126908.3126947.
14. Pednault E., Gunnels J.A., Nannicini G., et al. Breaking the 49-qubit barrier in the simulation of quantum circuits // arXiv preprint. 2017. arXiv:1710.05867.
15. Pednault E., Gunnels J., Maslov D., et al. On "Quantum Supremacy". 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy (дата обращения: 01.09.2020).
16. Chen Z.Y., Zhou Q., Xue C., et al. 64-qubit quantum circuit simulation // Science Bulletin. 2018. Vol. 63, no. 15. P. 964–971. DOI: 10.1016/j.scib.2018.06.007.
17. Arute F., Arya K., Babbush R., et al. Quantum supremacy using a programmable superconducting processor // Nature. 2019. Vol. 574, no. 7779. P. 505–510. DOI: 10.1038/s41586-019-1666-5.
18. Google AI Blog: A Preview of Bristlecone, Google's New Quantum Processor. URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (дата обращения: 01.09.2020).
19. The D-Wave 2000Q™ System. URL: https://www.dwavesys.com/d-wave-two-system (дата обращения: 01.09.2020).

20. de Avila A.B., Reiser R.H., Pilla M.L., et al. State-of-the-art quantum computing simulators: Features, optimizations, and improvements for D-GM // Neurocomputing. 2020. Vol. 393. P. 223–233. DOI: 10.1016/j.neucom.2019.01.118.
21. Häner T., Steiger D.S., Smelyanskiy M., et al. High performance emulation of quantum circuits // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC'16. IEEE, 2016. P. 866–874. DOI: 10.1109/SC.2016.73.
22. GitHub — QuEST-Kit/QuEST: A multithreaded, distributed, GPU-accelerated simulator of quantum computers. URL: https://github.com/QuEST-Kit/QuEST (дата обращения: 01.09.2020).
23. GitHub — iqusoft/intel-qs: High-performance simulator of quantum circuits. URL: https://github.com/iqusoft/intel-qs (дата обращения: 01.09.2020).
24. Voevodin Vl.V., Antonov A.S., Nikitenko D.A., et al. Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

Линев Алексей Владимирович, заведующий лабораторией, кафедра программной инженерии, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Ведруков Павел Евгеньевич, студент, Институт информационных технологий, математики и механики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Куландин Денис Сергеевич, студент, Институт информационных технологий, математики и механики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Мееров Иосиф Борисович, к.т.н., доцент, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Денисов Сергей, к.ф.-м.н., профессор, кафедра прикладной математики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация), доцент, факультет компьютерных наук, Столичный университет Осло (Осло, Норвегия)

DOI: 10.14529/cmse210104

QUEST AND INTEL-QS QUANTUM COMPUTATION SIMULATION PACKAGES PERFORMANCE COMPARISON

© 2021 A.V. Liniov[1], P.E. Vedrukov[1], D.S. Kulandin[1], I.B. Meyerov[1], S. Denisov[1,2]

[1]*Lobachevsky State University of Nizhny Novgorod (pr. Gagarina 23, Nizhnij Novgorod, 603950 Russia),*

[2]*Oslo Metropolitan University (P.O. Box 4, st. Olavs plass, NO-0130 Oslo, Norway)*

E-mail: alin@unn.ru, pavelvedrukov@mail.ru, kulandin08@gmail.com, meerov@vmk.unn.ru, sergiyde@oslomet.no

Received: 14.09.2020

In the nearest future quantum computers will be suitable for practical use. The development of quantum algorithms can be carried out using classical computers and specialized software that allows simulating of a quantum circuit functioning. Simulation results can be used to analyze the algorithm and also contribute to co-design when developing quantum architectures. However, when planning and performing numerical experiments, it is necessary to understand the capabilities of simulators and the limitations on the parameters of the quantum circuit imposed by the characteristics of the available classical computational resources (computers). This paper presents the results of computational experiments on simulating the operation of quantum circuits on an ideal quantum computer using the QuEST and Intel-QS packages, as well as our own "naïve" implementation. Restrictions on the size of a simulated quantum system N are shown when using computing systems of various classes — a virtual machine, a computing server, a computing server with a graphics accelerator (GPU), a supercomputer (the maximum achieved size is $N = 33$). The performance and scalability characteristics of the considered implementations on shared and distributed memory are given (the observed scaling efficiency is 30 % and 70 %, respectively). For the QuEST package and our own implementation the performance is presented for systems with graphics accelerator (GPU).

Keywords: computational quantum physics, quantum algorithms, high-performance computing, GPGPU, QuEST, Intel-QS.

FOR CITATION

Liniov A.V., Vedrukov P.E., Kulandin D.S., Meyerov I.B., Denisov S. QuEST and Intel-QS Quantum Computation Simulation Packages Performance Comparison. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 49–61. (in Russian) DOI: 10.14529/cmse210104.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cites.

References

1. Mermin N.D. Quantum computer science: an introduction. Cambridge University Press, 2007. 233 p.
2. Trieu D.B. Large-scale simulations of error prone quantum computation devices. Forschungszentrum Jülich, 2010. Vol. 2. 173 p.
3. Wiki Q. List of QC simulators. 2015. URL: https://quantiki.org/wiki/list-qc-simulators (accessed: 01.09.2020).
4. Green A.S., Lumsdaine P.L., Ross N.J., et al. Quipper: a scalable quantum programming language. Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation. 2013. P. 333–342. DOI: 10.1145/2491956.2462177.

5. Cross A.W., Bishop L.S., Smolin J.A., et al. Open quantum assembly language. arXiv preprint. 2017. arXiv:1707.03429.
6. Svore K., Geller A., Troyer M., et al. Q# Enabling Scalable Quantum Computing and Development with a High-level DSL. Proceedings of the Real World Domain Specific Languages Workshop 2018. 2018. P. 1–10. DOI: 10.1145/3183895.3183901.
7. Abhari A.J., Faruque A., Dousti M.J., et al. Scaffold: Quantum programming language. Princeton Univ NJ Dept of Computer Science, 2012. 43 p.
8. Jones T., Brown A., Bush I., et al. Quest and high performance simulation of quantum computers. Scientific reports. 2019. Vol. 9, no. 1. P. 1–11. DOI: 10.1038/s41598-019-47174-9.
9. Guerreschi G.G., Hogaboam J., Baruffa F., et al. Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits. Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034007. DOI: 10.1088/2058-9565/ab8505.
10. Smelyanskiy M., Sawaya N.P.D., Aspuru-Guzik A. qHiPSTER: the quantum high performance software testing environment. arXiv preprint. 2016. arXiv:1601.07195.
11. Aleksandrowicz G., Alexander T., Barkoutsos P., et al. Qiskit: An open-source framework for quantum computing. 2019. DOI: 10.5281/zenodo.2562110.
12. Amy M., Gheorghiu V. staq–A full-stack quantum processing toolkit. Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034016. DOI: 10.1088/2058-9565/ab9359.
13. Häner T., Steiger D.S. 5 petabyte simulation of a 45-qubit quantum circuit. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2017. P. 1–10. DOI: 10.1145/3126908.3126947.
14. Pednault E., Gunnels J.A., Nannicini G., et al. Breaking the 49-qubit barrier in the simulation of quantum circuits. arXiv preprint arXiv:1710.05867. 2017.
15. Pednault E., Gunnels J., Maslov D., et al. On "Quantum Supremacy". 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/ (accessed: 01.09.2020).
16. Chen Z.Y., Zhou Q., Xue C., et al. 64-qubit quantum circuit simulation. Science Bulletin. 2018. Vol. 63, no. 15. P. 964–971. DOI: 10.1016/j.scib.2018.06.007.
17. Arute F., Arya K., Babbush R., et al. Quantum supremacy using a programmable superconducting processor. Nature. 2019. Vol. 574, no. 7779. P. 505–510. DOI: 10.1038/s41586-019-1666-5.
18. Google AI Blog: A Preview of Bristlecone, Google's New Quantum Processor. URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (accessed: 01.09.2020).
19. The D-Wave 2000Q™ System. URL: https://www.dwavesys.com/d-wave-two-system (accessed: 01.09.2020).
20. de Avila A.B., Reiser R.H., Pilla M.L., et al. State-of-the-art quantum computing simulators: Features, optimizations, and improvements for D-GM. Neurocomputing. 2020. Vol. 393. P. 223–233. DOI: 10.1016/j.neucom.2019.01.118.
21. Häner T., Steiger D.S., Smelyanskiy M., et al. High performance emulation of quantum circuits. SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2016. P. 866–874. DOI: 10.1109/SC.2016.73.
22. GitHub – QuEST-Kit/QuEST: A multithreaded, distributed, GPU-accelerated simulator of quantum computers. URL: https://github.com/QuEST-Kit/QuEST (accessed: 01.09.2020).
23. GitHub – iqusoft/intel-qs: High-performance simulator of quantum circuits. URL: https://github.com/iqusoft/intel-qs (accessed: 01.09.2020).
24. Voevodin Vl.V., Antonov A.S., Nikitenko D.A., et al. Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.