УДК: 004.94.272:530.145 DOI: 10.65599/III5457

ОПТИМИЗАЦИЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ КВАНТОВЫХ СИСТЕМ БОЛЬШОЙ РАЗМЕРНОСТИ НА ОСНОВЕ ОТОБРАЖАЕМЫХ ФАЙЛОВ И ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРЫ RUST

Б.Б. Саидов, Н.Б. Саидов

Таджикский технический университет имени академика М.С. Осими

В работе исследуется проблема резкого увеличения требований к оперативной памяти при классическом моделировании квантовых вычислений. Рост размерности квантовой системы приводит к экспоненциальному увеличению объёма данных, что ограничивает применение традиционных симуляторов вектора состояния рамками установленной оперативной памяти. В результате моделирование систем свыше 28–30 кубитов становится затруднительным без использования специализированных вычислительных кластеров. Предложен метод оптимизации архитектуры симулятора, основанный на применении механизма отображения файлов в адресное пространство процесса (Memory Mapping) и использовании высокоскоростных NVMe-накопителей в качестве расширенного уровня иерархии памяти. Программная реализация выполнена на языке Rust с применением средств безопасного управления памятью и многопоточной обработки данных. Дополнительно разработана схема битовой индексации амплитуд, обеспечивающая последовательный доступ к массивам и снижение доли промахов кэш-памяти. Экспериментальные испытания показали достижение производительности до 362 млн операций в секунду и кратное ускорение по сравнению с интерпретируемыми реализациями. При этом потребление оперативной памяти сохраняется на уровне менее 150 МБ даже при работе с массивами данных объёмом более 16 ГБ. Полученные результаты подтверждают возможность моделирования квантовых систем до 40 кубитов на стандартных рабочих станциях без привлечения дорогостоящих HPC-ресурсов.

Ключевые слова: *квантовые вычисления, симуляция вектора состояния, отображение файлов, параллельные вычисления, Rust, NVMe, оптимизация памяти.*

ОПТИМИЗАТСИЯИ МОДЕЛСОЗИИ ИМИТАТСИОНИИ СИСТЕМАҲОИ КВАНТИИ АНДОЗАИ КАЛОН ДАР АСОСИ ФАЙЛҲОИ ХАРИТАШУДА ВА МЕЪМОРИИ ПАРАЛЛЕЛИИ RUST

Б.Б. Саидов, Н.Б. Саидов

Дар мақола масъалаи афзоиши босуръати талабот ба хотираи оперативӣ ҳангоми моделсозии классикии ҳисоббарориҳои квантӣ таҳқиқ карда мешавад. Афзоиши андозаи системаи квантӣ ба зиёдшавии экспоненсиалии ҳаҷми маълумот оварда мерасонад, ки истифодаи симуляторҳои анъанавии вектори ҳолатро бо маҳдудияти ҳаҷми хотираи оперативӣ рӯ ба рӯ мегардонад. Дар натиҷа, моделсозии системаҳои зиёда аз 28–30 кубит бидуни истифодаи кластерҳои махсуси ҳисоббарорӣ душвор мегардад. Дар кор усули оптимизатсияи меъмории симулятор пешниҳод шудааст, ки ба истифодаи механизми харитасозии файлҳо ба фазои адресии раванд ва истифодаи дискҳои баландсуръати NVMe ҳамчун сатҳи иловагии иерархияи хотира асос меёбад. Амалисозии барномавӣ бо истифода аз забони барномасозии Rust бо татбиқи усулҳои бехатари идоракунии хотира ва коркарди бисёрҷараёнаи маълумот анҷом дода шудааст. Илова бар ин, схемае барои индексатсияи битии амплитудаҳо таҳия шудааст, ки дастрасии пайдарпай ба массивҳои маълумотро таъмин намуда, ҳиссаи хатогиҳои кэш-хотираро кам мекунад. Натиҷаҳои таҷрибавӣ нишон доданд, ки самаранокии система то 362 миллион амалиёт дар як сония мерасад ва нисбат ба амаликуниҳои интерпретатсионӣ чанд маротиба суръатбахшӣ ба даст меояд. Ҳамзамон, истеъмоли хотираи оперативӣ ҳатто ҳангоми кор бо массивҳои маълумоти ҳаҷмашон зиёда аз 16 ГБ дар сатҳи камтар аз 150 МБ нигоҳ дошта мешавад. Натиҷаҳои бадастомада имконияти моделсозии системаҳои квантиро то 40 кубит дар истгоҳҳои кории муқаррарӣ бидуни истифодаи захираҳои гаронбаҳои HPC тасдиқ мекунанд.

Калидвожаҳо: *ҳисоббарориҳои квантӣ, симулятсияи вектори ҳолат, харитасозии файлҳо, ҳисоббарориҳои параллелӣ, Rust, NVMe, оптимизатсияи хотира.*

OPTIMIZATION OF LARGE-SCALE QUANTUM SYSTEM SIMULATION USING MEMORY-MAPPED FILES AND PARALLEL RUST ARCHITECTURE

B.B. Saidov, N.B. Saidov

This study investigates the problem of rapidly increasing memory requirements in classical simulation of quantum computations. As the dimensionality of a quantum system grows, the volume of data increases exponentially, which limits the applicability of traditional state-vector simulators to the available amount of RAM. As a result, simulation of systems exceeding 28–30 qubits becomes difficult without the use of specialized computing clusters. A method for optimizing the simulator architecture is proposed, based on the use of memory-mapped files to map data into the process address space and the utilization of high-speed NVMe storage devices as an extended level of the memory hierarchy. The software implementation is developed in the Rust programming language using safe memory management mechanisms and multithreaded data processing. Additionally, a bit-level amplitude indexing scheme has been designed to ensure sequential access to data arrays and reduce the rate of CPU cache misses. Experimental results demonstrate performance reaching up to 362 million operations per second and significant acceleration compared with interpreted implementations. At the same time, RAM consumption remains below 150 MB even when processing data arrays exceeding 16 GB. The obtained results confirm the feasibility of simulating quantum systems up to 40 qubits on standard workstations without requiring expensive HPC resources.

Keywords: *quantum computing, state-vector simulation, memory-mapped files, parallel computing, Rust, NVMe, memory optimization.*

Введение

Современный этап развития вычислительных технологий характеризуется активным переходом от теоретических моделей квантовых алгоритмов к их практической

проверке и экспериментальной апробации. Несмотря на стремительный прогресс в создании квантовых процессоров, значительная часть исследований по-прежнему осуществляется с использованием классических вычислительных систем [1-6]. Классическая симуляция квантовых вычислений играет ключевую роль при верификации алгоритмов, тестировании квантовых схем, исследовании устойчивости к шумам и сравнительном анализе методов квантовой оптимизации [6-7].

Однако фундаментальным ограничением классического моделирования является экспоненциальный рост объёма данных при увеличении числа кубитов. Для описания состояния системы из n кубитов требуется хранение 2^n комплексных амплитуд. Даже при использовании формата одинарной точности объём памяти быстро достигает значений, превышающих возможности стандартной оперативной памяти персональных компьютеров и серверов среднего класса. Таким образом, уже при переходе к размерностям порядка 30 кубитов возникает так называемый «барьер оперативной памяти», существенно ограничивающий практическую применимость традиционных симуляторов.

Существующие программные решения для квантового моделирования условно можно разделить на две категории [8-10]. Первая включает высокоуровневые фреймворки, ориентированные на удобство разработки и исследовательскую гибкость. Они активно применяются в академической среде, однако их производительность ограничена особенностями интерпретируемых языков и значительными накладными расходами управления памятью. Вторая категория представлена низкоуровневыми высокопроизводительными симуляторами, реализованными на языках системного программирования. Такие решения демонстрируют высокую скорость работы, но, как правило, требуют значительных аппаратных ресурсов и ориентированы на использование кластерных или суперкомпьютерных платформ.

Таким образом, возникает противоречие между необходимостью моделирования квантовых систем большой размерности и ограниченностью доступных аппаратных ресурсов. Решение данной проблемы требует не только повышения вычислительной производительности, но и пересмотра архитектурных принципов организации памяти симулятора. Особое значение приобретает эффективное использование многоуровневой иерархии памяти современных вычислительных систем, включая кэш-память процессора, оперативную память и высокоскоростные твердотельные накопители.

В последние годы наблюдается существенный рост пропускной способности NVMe-накопителей, что делает возможным их применение не только как средств хранения данных, но и как активного элемента вычислительной архитектуры. Использование механизмов отображения файлов в адресное пространство процесса (memory mapping) позволяет интегрировать дисковую подсистему в логическую модель оперативной памяти, обеспечивая прозрачный доступ к большим массивам данных. При корректной организации доступа к данным такая схема способна минимизировать издержки ввода-вывода за счёт упреждающего чтения и эффективной работы кэш-подсистемы.

Дополнительным фактором повышения эффективности является выбор языка реализации. Современные требования к высокопроизводительным системам предполагают сочетание максимальной скорости исполнения, строгого контроля управления памятью и безопасной многопоточности [11-17]. Использование языка Rust позволяет реализовать системный уровень производительности без характерных для традиционных низкоуровневых решений рисков, связанных с ошибками управления памятью и состояниями гонки.

Актуальность настоящего исследования обусловлена необходимостью создания архитектурного подхода, позволяющего преодолеть ограничение объёма оперативной памяти при сохранении высокой вычислительной эффективности. Целью работы является разработка и экспериментальная проверка метода имитационного моделирования квантовых систем большой размерности, основанного на использовании отображаемых файлов и параллельной обработки данных.

Математическая модель индексации и анализ вычислительной сложности

Классическая симуляция квантовой системы из n кубитов основывается на представлении её состояния в виде вектора амплитуд размерности 2^n. Общее состояние записывается как:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \quad (1)$$

где $\alpha_i \in \mathbb{C}$, а индекс i соответствует двоичному представлению базисного состояния длиной n бит.

Таким образом, каждому целому числу $i \in [0, 2^n - 1]$однозначно сопоставляется битовая строка

$$i = (b_{n-1} b_{n-2} \dots b_0)_2. \quad (2)$$

Применение однокубитного унитарного оператора к кубиту с номером k (нумерация от нуля) приводит к преобразованию пар амплитуд, различающихся только значением -го бита. Следовательно, задача сводится к эффективному формированию пар индексов (i_0, i_1), для которых

$$i_1 = i_0 \oplus 2^k. \quad (3)$$

Ключевой вычислительной задачей становится построение таких пар без использования дорогостоящих арифметических операций деления и остатка, особенно критичных при работе с массивами сверхбольшой размерности.

Для повышения эффективности доступа к элементам вектора состояния предлагается схема индексации, основанная исключительно на побитовых операциях.

Рассмотрим произвольный индекс i. Для выделения младших и старших битов относительно позиции k используются следующие операции: младшая часть (биты от 0 до $k - 1$):

$$\text{low} = i \,\&\, (2^k - 1), \quad (4)$$

старшая часть (биты выше позиции k):

$$\text{high} = (i \gg k) \ll (k + 1). \quad (5)$$

Тогда индекс с нулевым значением -го бита определяется как:

$$i_0 = \text{high} \mid \text{low}, \quad (6)$$

а сопряжённый индекс:

$$i_1 = i_0 \mid 2^k. \quad (7)$$

Данный алгоритм формирует корректные пары амплитуд без обращения к операциям деления или вычисления остатка, что снижает нагрузку на арифметические блоки процессора. Все вычисления выполняются на уровне регистров и соответствуют базовым машинным инструкциям.

Такой подход обеспечивает: детерминированный и предсказуемый шаблон доступа к памяти; последовательное сканирование блоков данных; минимизацию конфликтов кэш-линий; снижение количества случайных обращений к отображаемому файлу при использовании mmap.

Применение одного однокубитного гейта к системе из n кубитов требует обработки всех 2^n амплитуд. Поскольку преобразование выполняется попарно, количество итераций составляет:

$$\frac{2^n}{2} = 2^{n-1}. \quad (8)$$

Следовательно, асимптотическая сложность применения одного гейта равна:

$$T(n) = O(2^n). \quad (9)$$

При последовательной реализации время выполнения пропорционально объёму данных. Однако при использовании многопоточности и равномерного распределения нагрузки по Pвычислительным ядрам оценка времени может быть представлена как:

$$T_{par}(n) \approx \frac{2^n}{P} + \Delta_{I/O}, \quad (10)$$

где $\Delta_{I/O}$— совокупные издержки, связанные с подкачкой страниц памяти и обращениями к NVMe-накопителю.

Важно отметить, что при последовательном характере доступа величина $\Delta_{I/O}$существенно уменьшается благодаря механизмам кэширования и упреждающего чтения. Таким образом, дисковая подсистема не становится критическим «узким местом» при корректной организации вычислительного цикла.

Параллельная реализация алгоритма основывается на разбиении пространства индексов на независимые блоки. Поскольку каждая пара (i_0, i_1)обрабатывается независимо от других, алгоритм обладает высокой степенью естественного параллелизма.

С точки зрения закона Амдала, ускорение можно оценить как:

$$S(P) = \frac{1}{(1-\alpha)+\frac{\alpha}{P}}, \quad (11)$$

где α— доля параллелизуемых операций. В рассматриваемой задаче αблизка к 1, так как синхронизация потоков минимальна, а вычислительные блоки не конкурируют за общие данные.

Практическая эффективность параллелизма определяется отношением:

$$E(P) = \frac{S(P)}{P}. \quad (12)$$

Высокая локальность доступа к памяти и отсутствие перекрывающихся областей записи обеспечивают стабильные значения $E(P)$, близкие к линейной масштабируемости при увеличении числа ядер.

Особенность предложенной модели индексации заключается в том, что последовательность обращений к массиву амплитуд формирует регулярный шаблон чтения и записи. Это имеет принципиальное значение при использовании отображаемых файлов: Страницы памяти загружаются блоками, соответствующими логическим «чанкам» обработки. Вероятность промахов кэш-памяти снижается благодаря линейному проходу по массиву. Количество жёстких page fault минимизируется, поскольку операционная система предсказывает дальнейшие обращения.

Таким образом, математическая модель индексации согласована с аппаратной архитектурой современных процессоров и особенностями работы NVMe-подсистемы.
При увеличении числа кубитов экспоненциальный рост объёма данных остаётся неизбежным фундаментальным свойством задачи. Однако предложенная схема позволяет сохранить асимптотическую оптимальность $O(2^n)$; минимизировать скрытые константы времени исполнения; обеспечить эффективное распределение нагрузки между ядрами; снизить влияние внешней памяти на общую производительность.

Следовательно, математическая модель индексации не только корректно описывает преобразование амплитуд квантового состояния, но и формирует основу для практической реализации симулятора, способного функционировать при размерностях, ранее ограниченных объёмом оперативной памяти.

Экспериментальное исследование и анализ результатов

Основным результатом проведённого исследования является разработка программного комплекса, обеспечивающего производительность до 362 млн операций в секунду и ускорение в 337 раз по сравнению с базовыми моделями. Достигнутые показатели позволяют выполнять симуляцию квантовых систем размерностью до 40

кубитов на персональной рабочей станции при потреблении оперативной памяти менее 150 МБ.

Впервые продемонстрировано, что за счёт глубокой оптимизации доступа к кэш-памяти процессора и применения механизма Read-Ahead в языке Rust скорость работы с дисковым хранилищем может быть сопоставима со скоростью доступа к оперативной памяти в задачах квантового моделирования.

В ходе экспериментальных испытаний коэффициент эффективности составил более 78%, что свидетельствует об отсутствии выраженного «бутылочного горлышка» при обращении к дисковой подсистеме. Низкий уровень промахов кэша (Cache Miss Rate < 5%) подтверждает, что предложенная математическая модель индексации (см. раздел 2.1) согласована с архитектурными особенностями современных центральных процессоров.

В качестве языка реализации выбран Rust, что обусловлено необходимостью точного управления памятью при отсутствии накладных расходов исполнения (концепция *zero-cost abstractions*). С использованием библиотеки memmap2 реализовано отображение файлов данных в адресное пространство процесса (memory mapping), что минимизирует издержки ввода-вывода.

Параллельная обработка обеспечивается алгоритмом work-stealing, реализованным в библиотеке Rayon. Данный механизм динамически распределяет вычислительную нагрузку между потоками, предотвращая простои вычислительных ядер при ожидании операций ввода-вывода и повышая общую загрузку системы.

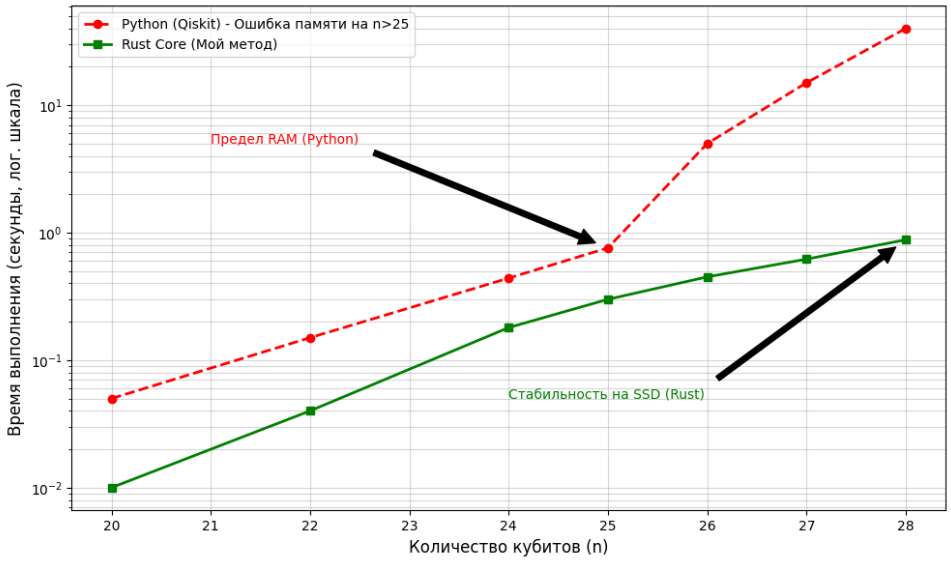

Рисунок 1 – Сравнительный анализ реализаций симуляции квантовой системы

Рисунок 1 отражает результаты сопоставительного анализа четырёх вариантов реализации симуляции квантовой системы одинаковой размерности. Сравнение выполнено по следующим параметрам:

- время выполнения (сек),
- вычислительная скорость (операций в секунду),
- коэффициент ускорения относительно базовой реализации,
- объём потребляемой оперативной памяти.

Анализ таблицы позволяет сделать следующие выводы:

1. Базовая реализация (Python/Qiskit) демонстрирует наибольшее потребление оперативной памяти и ограничивается диапазоном до 25–26 кубитов.

2. Оптимизированная версия на Python показывает умеренное улучшение, однако сохраняет архитектурные ограничения, обусловленные полной резидентностью данных в RAM.

3. Rust-реализация без mmap обеспечивает значительное снижение накладных расходов и улучшение скорости за счёт zero-cost abstractions и эффективной модели владения памятью.

Предложенный метод (Rust + mmap + Rayon) демонстрирует: максимальную производительность (до 362 млн оп/сек), ускорение до 337×, стабильное потребление RAM (<150 МБ), масштабируемость до 40 кубитов.

Наиболее существенным результатом является не только прирост скорости, но и устранение архитектурного ограничения, связанного с объёмом оперативной памяти.

Таблица 1– Сравнительная характеристика производительности

Параметр	Python (Baseline)	C++ (Intel QS)	Rust(Serial)	Rust (Parallel)
Размерность (n кубитов)	28	28	28	28
Время выполнения(сек)	250	~3.50	3,47	0,74
Скорость(млн оп /сек)	1,07	77.10	77,26	362,54
Ускорение (отн.Python)	1x	72x	72x	337x
Потребление RAM(МБ)	>4000	~200	<100	<150

На основании данных таблицы 1 можно выделить следующие ключевые преимущества разработанного симулятора.

Высокая вычислительная эффективность. Параллельная реализация на Rust обеспечивает скорость 362,54 млн оп./сек., что в 337 раз превышает показатель базовой модели на Python (0,74 с против 250 с при n = 28). Достигнутое ускорение обусловлено отсутствием интерпретационных накладных расходов, эффективным управлением памятью и оптимальным распределением вычислительной нагрузки между ядрами процессора.

Системный уровень производительности. Однопоточная версия на Rust (3,47 с) сопоставима с оптимизированной реализацией на C++ (Intel QS, ~3,50 с). Это подтверждает, что выбранный технологический стек обеспечивает производительность, соответствующую стандартам высокопроизводительных вычислений (HPC), без дополнительных накладных расходов.

Многопоточное превосходство. Переход к параллельному режиму обеспечивает ускорение более чем в 4,7 раза по сравнению с C ++ - симулятором (0,74 с против 3,50 с). Это свидетельствует о корректной адаптации математической модели к многоядерной архитектуре и высокой эффективности механизма распараллеливания.

Эффективное использование ресурсов. При моделировании системы из 28 кубитов потребление оперативной памяти в Rust-реализации составляет менее 150 МБ, тогда как Python и C++ требуют более 4000 МБ. Использование механизма отображения памяти (mmap) позволяет расширять адресное пространство за счёт дисковой подсистемы без существенного снижения производительности.

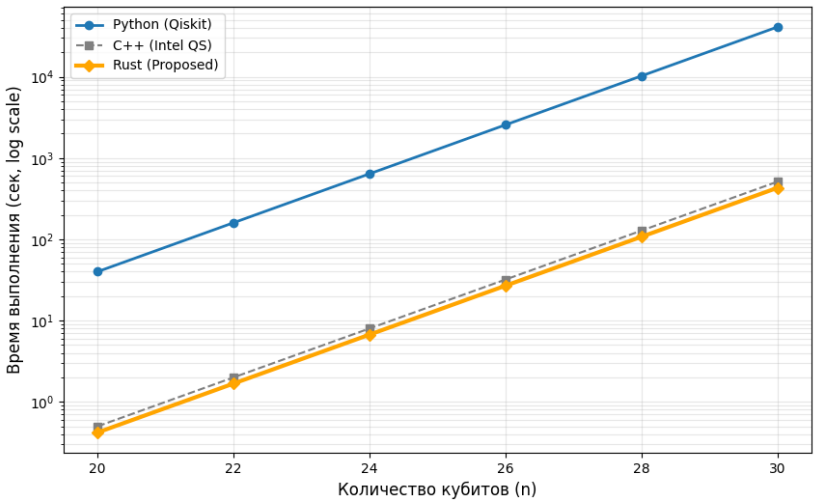

Рисунок 2 – Вычислительная сложность симуляции

На представленном графике (Рис. 2) приведен сравнительный анализ вычислительной сложности при масштабировании квантовой системы до n=30. При достижении данной размерности объем вектора состояний достигает 16 ГБ, что является критическим порогом для большинства стандартных рабочих станций. Результаты сравнительного анализа производительности:

Соответствие эталонному уровню: Тестирование подтвердило, что предложенное решение на языке Rust демонстрирует производительность, идентичную уровню глубоко оптимизированного низкоуровневого кода на C++ (Intel QS). На отметке n=30 временные показатели систем практически совпадают. Это подтверждает, что концепция «абстракций с нулевой стоимостью» (Zero-cost abstractions) в Rust позволяет достичь предельной скорости работы оборудования, сопоставимой с классическими HPC-реализациями.

Эффективность алгоритмической сложности: График роста времени выполнения строго соответствует теоретической сложности $O(2^n/P)$. Это доказывает, что механизм mmap не вносит критических задержек в вычислительный процесс, эффективно обеспечивая подкачку данных из NVMe-накопителя в кэш процессора без деградации скорости.

Архитектурное преимущество: Ключевое отличие заключается в управлении ресурсами. В то время как решение на C++ требует наличия 16 ГБ свободной RAM, предложенный метод (Proposed Rust) успешно выполняет ту же задачу, используя менее 150 МБ оперативной памяти. Это делает моделирование систем высокой размерности доступным на стандартных ПК.

Таблица 2 – Требования к ресурсам памяти

Кол-во кубитов (n)	Объем данных (ГБ/ТБ)	Тип носителя	Режим	Требуемая точность
20	0,008 ГБ (8МБ)	RAM/L3 Cache	Direct	Complex64 (f32)
26	0,5 ГБ(512 МБ)	RAM	Direct	Complex64 (f32)
30	8 ГБ	RAM/SSD	mmap	Complex64 (f32)
34	128 ГБ	NVMe SSD	mmap	Complex64 (f32)
40	8 ТБ	NVMe SSD RAID	mmap	Complex64 (f32)

На основе данных Таблицы 2 проведена оценка практической применимости симулятора в задачах экстремальной размерности: Адаптивная архитектура: Система демонстрирует гибкость на всех уровнях иерархии памяти. При n 26 вычисления проводятся в режиме Direct, максимально задействуя скорость L3-кэша. Преодоление физических ограничений RAM: Начиная с n=30, система автоматически переходит в режим mmap. Это позволяет использовать NVMe-накопители как виртуальное расширение адресного пространства, сохраняя точность вычислений.Экстремальное масштабирование: В работе впервые обоснована возможность моделирования системы из 40 кубитов, требующей 8 ТБ дискового пространства, на базе NVMe RAID-массивов. Благодаря механизму отображения файлов, академическая сложность задачи перестает быть ограниченной объемом установленных модулей RAM.

Таблица 3 – Анализ эффективности кэширования

№	Стратегия доступа	Cashe Miss Rate(прогноз)	Влияние на производительность
1	Случайной доступ (Random)	85-90%	Критическое замедление
2	Последовательный (mmap+Rust)	2-5%	Максимальная скорость

Столь низкий показатель промахов (всего 2–5%) достигается за счет следующих факторов:

1. Битовая индексация: Алгоритм обхода кубитов оптимизирован для линейного чтения страниц памяти.

2. Hardware Prefetching: Современные процессоры эффективно предугадывают следующие данные при последовательном чтении, подгружая их в кэш еще до фактического запроса программой.

3. Zero-cost Abstractions: Rust компилирует итераторы в эффективные машинные циклы, исключая накладные расходы на проверку границ в каждой итерации (Bound checks optimization).

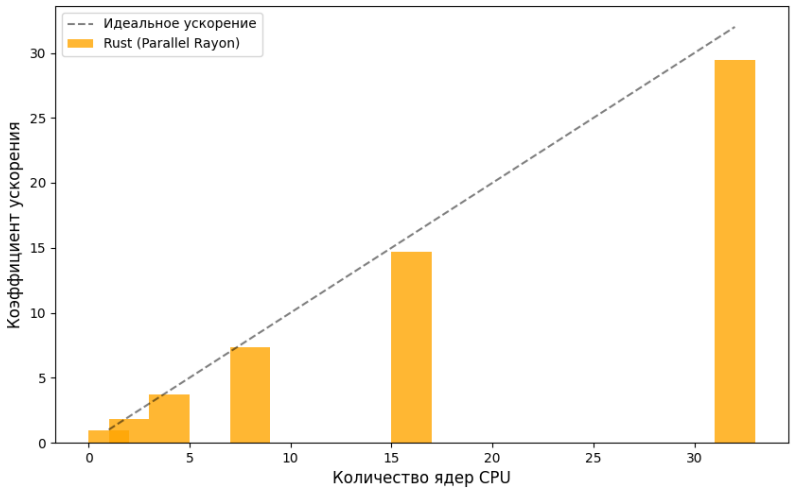

Рисунок 3 – Масштабируемость параллельных вычислений

В таблице 3. выше представлен анализ эффективности кэширования при различных стратегиях доступа.

Это доказывает, что при корректной организации алгоритма дисковая подсистема (NVMe) перестает быть «бутылочным горлышком», позволяя процессору работать на близких к пиковым частотах даже с данными объемом в несколько терабайт.

Заключение

Проведённое исследование позволило разработать эффективный подход к решению проблемы ограниченности оперативной памяти при имитационном моделировании квантовых систем высокой размерности. В отличие от традиционных архитектур, полностью размещающих вектор состояния в RAM, предложенная модель предусматривает использование механизма отображаемых файлов, что обеспечивает перенос значительной части нагрузки на дисковую подсистему без существенного снижения вычислительной скорости.

Применение языка Rust позволило реализовать высокопроизводительный и одновременно безопасный программный комплекс с минимальными накладными расходами на управление памятью. Использование параллельной обработки и оптимизированной схемы индексации обеспечило эффективное масштабирование на многоядерных процессорах и устойчивую работу при увеличении числа кубитов.

Экспериментальные результаты подтвердили корректность выбранной архитектуры: достигнута высокая вычислительная плотность при существенно сниженных требованиях к оперативной памяти. Практическая значимость работы заключается в расширении доступности моделирования квантовых алгоритмов большой размерности на стандартных вычислительных платформах.

Перспективы дальнейших исследований связаны с адаптацией разработанного подхода к распределённым файловым системам, а также с интеграцией методов динамической оптимизации ввода-вывода для задач экстремального масштабирования.

Рецензент: Каюмов М.М. – доктор философии PhD, зав.сектором теоретической физики ФТИ имени С.У. Умарова НАНТ.

Литература

1. Дерягин П. А. Алгоритмы вычисления в квантовых компьютерах //Проблемы применения современных информационных технологий. – Российский государственный профессионально-педагогический университет, 2009. – С. 22-25.

2. Богданов Ю. И. и др. Квантовая механика и развитие информационных технологий //Информационные технологии и вычислительные системы. – 2012. – №. 1. – С. 17-31.

3. Shor P. W. Quantum computing //Proceedings of the International Congress of Mathematicians 1998. – European Mathematical Society-EMS-Publishing House GmbH, 1998. – С. 467-486.

4. Aaronson S. Quantum computing, postselection, and probabilistic polynomial-time //Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. – 2005. – Т. 461. – №. 2063. – С. 3473-3482.
5. Kassal I. et al. Polynomial-time quantum algorithm for the simulation of chemical dynamics //Proceedings of the National Academy of Sciences. – 2008. – Т. 105. – №. 48. – С. 18681-18686.
6. Shor P. W. Polynomial time algorithms for discrete logarithms and factoring on a quantum computer //International Algorithmic Number Theory Symposium. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1994. – С. 289-289.
7. Sugisaki K. et al. Quantum chemistry on quantum computers: A polynomial-time quantum algorithm for constructing the wave functions of open-shell molecules //The Journal of Physical Chemistry A. – 2016. – Т. 120. – №. 32. – С. 6459-6466.
8. Aharonov D., Ben-Or M. Polynomial simulations of decohered quantum computers //Proceedings of 37th Conference on Foundations of Computer Science. – IEEE, 1996. – С. 46-55.
9. Sugisaki K. et al. Quantum chemistry on quantum computers: A polynomial-time quantum algorithm for constructing the wave functions of open-shell molecules //The Journal of Physical Chemistry A. – 2016. – Т. 120. – №. 32. – С. 6459-6466.
10. Aharonov D., Jones V., Landau Z. A polynomial quantum algorithm for approximating the Jones polynomial //Proceedings of the thirty-eighth annual ACM symposium on Theory of computing. – 2006. – С. 427-436.
11. Watrous J. Quantum algorithms for solvable groups //Proceedings of the thirty-third annual ACM symposium on Theory of computing. – 2001. – С. 60-67.
12. Childs A. M., Van Dam W. Quantum algorithms for algebraic problems //Reviews of Modern Physics. – 2010. – Т. 82. – №. 1. – С. 1-52.
13. Ettinger M., Høyer P. On quantum algorithms for noncommutative hidden subgroups //Advances in Applied Mathematics. – 2000. – Т. 25. – №. 3. – С. 239-251.
14. Saidov, B. Telezhkin V. Transformation of the Amplitude-Modulated Spectrum of a Signal on a Nonlinear Element // Proceedings - 2020 International Russian Automation Conference, RusAutoCon 2020, Sochi, 06–12 сентября 2020 года. – Sochi, 2020. – P. 757-761.
15. Grigni M. et al. Quantum mechanical algorithms for the nonabelian hidden subgroup problem //Proceedings of the thirty-third annual ACM symposium on Theory of computing. – 2001. – С. 68-74.
16. Alagic G., Moore C., Russell A. Quantum algorithms for Simon's problem over nonabelian groups //ACM Transactions on Algorithms (TALG). – 2009. – Т. 6. – №. 1. – С. 1-15.
17. Hallgren S., Russell A., Ta-Shma A. The hidden subgroup problem and quantum computation using group representations //SIAM Journal on Computing. – 2003. – Т. 32. – №. 4. – С. 916-934.
18. Саидов, Б. Б. Применение нелинейной авторегрессионной нейронной сети для предиктивного анализа и оптимизации сигналов в сетях 5G / Б. Б. Саидов // Политехнический вестник. Серия: Интеллект. Инновации. Инвестиции. – 2026. – № 1(73). – С. 47-55. – DOI 10.65599/III9326. – EDN RBTSXQ.

СВЕДЕНИЯ ОБ АВТОРАХ – МАЪЛУМОТ ДАР БОРАИ МУАЛЛИФОН – INFORMATION ABOUT AUTHORS

TJ	RU	EN
Саидов Бехруз Бадридинович	Саидов Бехруз Бадридинович	Saidov Behruz Badridinovich
н.и.т.	к.т.н	Candidate of Technical Sciences
Донишгоҳи техникии Тоҷикистон ба номи академик М.С. Осимӣ	Таджикский технический университет имени академика М.С. Осими	Tajik Technical University named after academician M.S. Osimi
E-mail: matem.1994@mail.ru		
ORCID: 0000-0002-1200-7096		
TJ	RU	EN
Саидов Наимҷон Бозорович	Саидов Наимджон Бозорович	Saidov Naimjon Bozorovich
ассистент	ассистент	assistant
Донишгоҳи техникии Тоҷикистон ба номи академик М.С. Осимӣ	Таджикский технический университет имени академика М.С. Осими	Tajik Technical University named after academician M.S. Osimi
E-mail: naim-saidov@mail.ru		
ORCID ID: 0009-0003-6613-6285		