УДК 004.67

ТЕХНОЛОГИИ УСКОРЕНИЯ ВЫЧИСЛЕНИЙ

Артамонов В.В. – магистрант, Маслов Д.Б. – магистрант,
Скрябин А.А. – магистрант,
Славолюбова Я.В. – к.ф.-м.н., доцент кафедры ПИТ
Кузбасский государственный технический университет имени Т.Ф. Горбачева
Россия, г. Кемерово

Современный мир стремительно развивается благодаря развитию информационных технологий, которые обеспечивают быструю и эффективную обработку больших объемов данных. В этой связи распределенные системы обработки данных приобретают особое значение, так как они позволяют использовать ресурсы множества вычислительных узлов для решения сложных задач. Технологии ускорения вычислений, такие как параллельные ядра центральных процессоров, а также программные платформы CUDA и OpenCL, играют ключевую роль в повышении производительности распределенных систем. Внедрение этих технологий способствует значительному сокращению времени обработки данных и расширению возможностей современных вычислительных систем.

Цель настоящего исследования заключается в анализе и обосновании применения современных методов параллельных вычислений в распределенных системах обработки данных, включая многоядерные процессоры, графические ускорители и платформы CUDA и OpenCL

Для ускорения вычислений через центральный процессор (ЦП) важно понимать ограничение роста производительности, которое описывает закон Амдала. Этот закон формулируется следующим образом:

$$S_p = \frac{1}{\alpha - \frac{1-\alpha}{p}},$$

где S_p - ускорение вычислений при использовании p вычислителей, а α - доля последовательного кода, который нельзя распараллелить. Закон демонстрирует, что даже при увеличении количества процессоров прирост производительности ограничен из-за неизбежной последовательной части задачи (рисунок 1). Например, если α=0.25, то максимальное ускорение не превысит 4 раза, независимо от количества процессоров.

Современные центральные процессоры используют широкий набор архитектурных и аппаратных технологий, направленных на повышение производительности. По мере роста требований к вычислительной мощности увеличение быстродействия одного ядра остается критически важным. Ключевые технологии, применяемые для ускорения вычислений на центральном процессоре, включают:

- конвейеризацию;

- суперскалярную архитектуру;
- векторные инструкции (SIMD);
- технологию одновременной многопоточности (Hyper-Threading / SMT);
- повышение тактовой частоты (Turbo Boost).

Кроме того, ускорение зависит не только от количества ядер, но и от особенностей языка программирования. Например, при реализации параллельного цикла (parallel for) на языке Rust можно добиться небольшого ускорения даже при использовании одного потока (около 1.01 раза) за счет оптимизаций компилятора, отличающихся от поведения аналогичных конструкций на других языках, таких как C# или python. Существенное влияние также оказывают архитектура процессора, особенности аппаратного обеспечения и операционной системы, которые взаимодействуют с кодом и влияют на итоговую производительность.

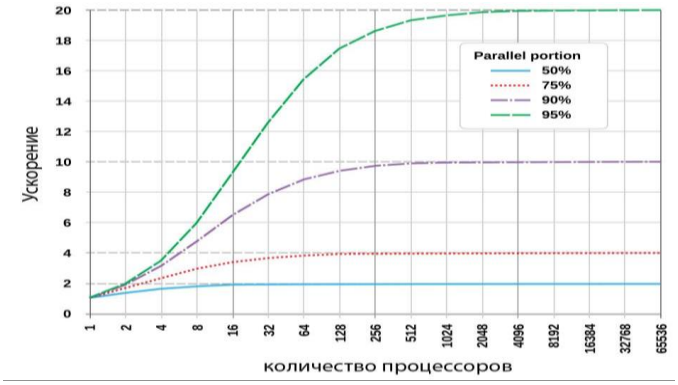

Рисунок 1 - Ускорение программы при параллельных вычислениях на нескольких процессорах

Для преодоления ограничений центрального процессора применяются графические ускорители, в частности технология NVIDIA CUDA. Графические процессоры (GPU - graphics processing unit) используют иную архитектуру: тысячи параллельных потоков и высокая пропускная способность памяти позволяют обрабатывать большие массивы данных значительно эффективнее, чем ЦП. Программная модель CUDA разделяет код на часть, выполняемую на хосте, и ядра, запускаемые на устройстве; последние компилируются в промежуточное представление PTX либо в машинный код под конкретную архитектуру графического процессора.

В операциях линейной алгебры и других вычислительно плотных алгоритмах, где структура данных позволяет равномерно распределять работу между потоками, ускорение достигает многократных значений. Практические эксперименты подтверждают, что при корректной организации вычислений графический ускоритель способен существенно опережать как последовательный, так и многопоточный вариант исполнения на центральном процессоре.

Однако использование графических ускорителей требует учета ряда особенностей, которые существенно влияют на итоговую производительность. Во-первых, данные необходимо передавать между оперативной памятью

центрального процессора и памятью видеоускорителя, что сопровождается заметными накладными расходами. По этой причине слишком малые задачи не позволяют раскрыть потенциал параллельной архитектуры и могут работать медленнее, чем их аналоги на ЦП. Во-вторых, эффективность вычислений на GPU зависит от структуры доступа к памяти: для достижения высоких скоростей необходимо обеспечивать последовательный и согласованный доступ потоков к данным, а также активно использовать локальные области памяти ускорителя для временных блоков вычислений. В противном случае ограниченная пропускная способность глобальной памяти станет определяющим фактором замедления.

Кроме того, необходимо учитывать специфику численных методов в условиях массового параллелизма. Параллельная перестановка операций приводит к небольшим различиям в результатах вычислений по сравнению с последовательной реализацией, особенно при работе с арифметикой с плавающей точкой. В задачах решения систем линейных уравнений важным становится использование устойчивых схем, например частичного выбора главного элемента или блочных вариантов LU-разложения, иначе высокое ускорение может сопровождаться потерей точности.

На практике все это приводит к необходимости тщательно подбирать алгоритмы, хорошо масштабируемые на большое число потоков, минимизировать количество обменов данными между ЦП и GPU, а также использовать средства высокоточного измерения времени выполнения и анализа производительности.

Еще одной известной технологией ускорения вычислений является OpenCL (Open Computing Language) - это открытый стандарт и фреймворк для разработки параллельных программ, которые могут выполняться на разнородных вычислительных платформах, включая ЦП, GPU, процессоры цифровой обработки сигналов (DSP), а также другие вычислительные модули и программируемые логические интегральные схемы (FPGA). Основное назначение OpenCL - максимально эффективно использовать всю вычислительную мощность системы, распределяя ресурсоемкие задачи между устройствами, которые смогут выполнить их наиболее эффективно.

Архитектура OpenCL включает несколько ключевых компонентов. Платформа состоит из одного или нескольких устройств (CPU, GPU и т.п.), на которых выполняются вычисления. Хост-программа работает на CPU и управляет процессом: создает контексты, очереди команд, буферы данных и запускает вычисления на устройствах. Вычислительные функции, называемые ядрами (kernels), пишутся на диалекте языка C и компилируются во время выполнения. Ядра исполняются параллельно множеством рабочих элементов, организованных в многомерную структуру NDRange. Важную роль играет модель памяти OpenCL с иерархией глобальной, локальной и приватной памяти, что позволяет оптимизировать эффективность вычислений.

Преимущества OpenCL включают кроссплатформенность, гетерогенность, т.е. использование всех видов вычислительных ресурсов системы, а

также высокую производительность при правильной оптимизации за счет одновременного выполнения большого количества потоков. OpenCL широко применяется в задачах обработки изображений и видео, научных вычислениях, финансовом моделировании, машинном обучении и в игровых технологиях.

Для оценки эффективности различных технологий был проведен вычислительный эксперимент, в рамках которого измерялось время решения системы линейных уравнений размером 5000. Полученные результаты представлены в таблице 1.

Таблица 1 - Решение СЛАУ-матрицы размером 5000 с использование различных технологий ускорения

Последовательное время	Параллельное (8 потоков)	NVIDIA CUDA (общее время)	OpenCL
~70,5 сек.	~66,9 сек.	~8,3 сек	~114,4 сек

Проведенный вычислительный эксперимент показывает, что эффективность технологий параллельных вычислений существенно зависит не только от аппаратных возможностей, но и от особенностей применяемых алгоритмов и программных моделей. Хотя графические ускорители демонстрируют значительный прирост производительности при использовании CUDA, вариант реализации на OpenCL оказался заметно медленнее не только GPU-версии, но и многопоточного CPU-исполнения. Это связано со спецификой выбранного алгоритма решения системы линейных уравнений, который характеризуется большим числом обращений к памяти и сложной схемой зависимости данных. Для таких задач пропускная способность памяти и характер доступа к ней становятся определяющим фактором, а недостаточная оптимизация ядра приводит к тому, что ресурс массового параллелизма OpenCL не раскрывается. Дополнительное влияние оказывают накладные расходы абстрактного рантайма OpenCL, управление очередями и синхронизация событий, что увеличивает общее время выполнения. Отдельно стоит отметить, что без специализированной оптимизации использование локальной памяти и других механизмов архитектуры GPU остается ограниченным, что также снижает эффективность. Полученные результаты подтверждают, что повышение производительности вычислений требует не только применения современных ускорителей, но и тщательного подбора алгоритмов, их адаптации под конкретную архитектуру и оптимизации доступа к памяти. Простое использование более мощного оборудования не гарантирует ускорения без учёта этих факторов.

Список литературы:

1. Дороганов, В. С. Современные проблемы распределенных информационных систем: конспект лекций [Электронный ресурс] // URL:

https://library.kuzstu.ru/dl.php?n=91198&type=utchposob:common (дата обращения: 20.11.2025).

2. Таненбаум Э. Архитектура компьютера. – 6-е изд. [Текст] / Таненбаум Э., Остин Т. – СПб.: Питер, 2013. – 816 с.: ил.

3. Малявко, А.А. Параллельное программирование на основе технологий openmp, cuda, opencl, mpi [Электронный ресурс] // URL: https://urait.ru/viewer/parallelnoe-programmirovanie-na-osnove-tehnologiy-openmp-cuda-opencl-mpi-562821 (дата обращения: 20.11.2025).