МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра системного анализа и автоматического управления

ИССЛЕДОВАНИЕ МАТЕМАТИЧЕСКОЙ МОДЕЛИ СИСТЕМЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Миронова Максима Андреевича

Научный руководитель
к. ф.-м. н., доцент _____ И. Е. Тананко

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. В современных условиях экспоненциального роста объёмов данных, генерируемых в научных исследованиях, промышленном производстве, финансовых системах и социальных сетях, традиционные последовательные архитектуры вычислительных систем перестают удовлетворять требованиям к производительности и времени отклика. Физические ограничения закона Мура и переход индустрии к многоядерным процессорам, гетерогенным вычислительным комплексам и распределённым кластерам делают параллельную обработку данных не просто альтернативой, а необходимым стандартом проектирования информационных систем. Однако практическое применение параллельных вычислений сопряжено со значительными трудностями: наличием архитектурных ограничений (например, Global Interpreter Lock в Python), нелинейным ростом накладных расходов на синхронизацию и межпроцессное взаимодействие, а также отсутствием универсальных математических моделей, способных точно прогнозировать ускорение для конкретных классов задач. Существующие теоретические оценки часто игнорируют специфику программных сред, особенности планировщиков операционных систем и характер вычислительной нагрузки, что приводит к значительному расхождению между предсказанными и фактическими показателями производительности. В этой связи исследование математических моделей систем параллельной обработки данных, их экспериментальная верификация и формирование обоснованных рекомендаций по выбору архитектурных решений и инструментов распараллеливания представляют собой актуальную научно-практическую задачу, имеющую прямое отношение к оптимизации вычислительных процессов в реальных приложениях.

Цель бакалаврской работы — исследование математической модели системы параллельной обработки данных, теоретическое обоснование её эффективности, экспериментальная верификация моделей и разработка практических рекомендаций по оптимизации вычислительных процессов.

Поставленная цель определила **следующие задачи**:

1. Провести системный анализ теоретических основ параллельных вычислений, включая классификацию архитектур, фундаментальные законы масштабирования и метрики оценки эффективности.
2. Исследовать и формализовать математические модели параллельной

обработки данных для различных классов вычислительных нагрузок (CPU-bound и I/O-bound).

3. Выполнить сравнительный анализ современных инструментов, библиотек и фреймворков для реализации параллельных систем в экосистеме Python и распределённых сред.
4. Разработать на основе теоретического исследования алгоритмические и архитектурные рекомендации по выбору стратегий параллелизма и оптимальной конфигурации вычислительных ресурсов.
5. Спроектировать, реализовать и апробировать модульный программный комплекс для экспериментального исследования математических моделей, провести серию контролируемых экспериментов и сопоставить теоретические предсказания с эмпирическими данными.

Методологические основы исследования параллельных вычислений, теории масштабирования, математического моделирования распределённых систем и оценки эффективности программных решений представлены в фундаментальных трудах Г. М. Амдала [1], Дж. Л. Густавсона [2], В. П. Гергеля [3], В. В. Воеводина [4], А. Таненбаума [5], А. Грамы [6], а также в современных исследованиях по оптимизации многопоточных и асинхронных архитектур [7, 8]. Теоретическая база работы опирается на методы математического моделирования, теорию массового обслуживания, принципы асимптотического анализа алгоритмов и эмпирические методы оценки производительности вычислительных систем.

Практическая значимость бакалаврской работы заключается в создании и верификации модульного программного комплекса на языке Python, предназначенного для автоматизированного исследования математических моделей параллелизма. Проведённая серия экспериментов позволила количественно оценить влияние накладных расходов, определить точку насыщения производительности для конкретной аппаратной конфигурации и сформулировать практические рекомендации по выбору инструментов параллелизма в зависимости от характера решаемых задач. Разработанные критерии и методики могут быть непосредственно применены при проектировании и оптимизации реальных систем обработки данных в областях машинного обучения, анализа больших данных, научных вычислений и высоконагруженных веб-сервисов, что способствует снижению эксплуатационных затрат

и повышению эффективности использования вычислительных ресурсов.

Структура и объём работы. Бакалаврская работа состоит из введения, 6 разделов, заключения, списка использованных источников и 3 приложений. Общий объём работы — 55 страниц, из них 44 страницы — основное содержание, включая 10 рисунков и 12 таблиц. Список использованных источников информации содержит 21 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы и законы параллельных вычислений» посвящен систематизации базовых концепций параллельной обработки данных, классификации вычислительных архитектур и анализу фундаментальных законов масштабирования. В подразделе 1.1 приведена классификация Флинна (SISD, SIMD, MISD, MIMD) и выполнено разделение систем по организации памяти (SMP и системы с распределённой памятью) с выделением их ключевых преимуществ и ограничений в контексте современных задач. В подразделе 1.2 исследованы законы Амдала и Густавсона-Барсиса, математически описывающие ограничения ускорения при фиксированном и масштабируемом размерах задачи соответственно. В подразделе 1.3 введена классификация задач по характеру нагрузки: CPU-bound (ограничены производительностью процессора) и I/O-bound (ограничены скоростью ввода-вывода), что позволило формализовать подходы к выбору стратегий распараллеливания. В подразделе 1.4 проведено численное сравнение двух законов масштабирования для диапазона ядер $n \in [1; 64]$ при различных долях последовательной части f. *Вывод по разделу:* установлено, что закон Амдала применим для задач с фиксированным объёмом вычислений, демонстрируя принципиальное ограничение ускорения величиной $1/f$, тогда как закон Густавсона-Барсиса описывает сценарии масштабирования задачи, позволяя достигать почти линейного роста производительности; выбор модели должен определяться характером прикладной нагрузки и архитектурными возможностями целевой платформы.

Второй раздел «Архитектурные подходы и модели параллельного программирования» посвящен анализу аппаратных платформ и программных парадигм, используемых для построения высокопроизводительных систем. В подразделе 2.1 рассмотрены многопроцессорные системы с общей

памятью (SMP) и предложена математическая модель эффективной пропускной способности с учётом конкуренции за ресурсы памяти и механизмов когерентности кэшей. В подразделе 2.2 исследованы распределённые и кластерные архитектуры, выведена оценка общего времени выполнения с учётом коммуникационных задержек и топологии сети. В подразделе 2.3 проанализированы гибридные системы (CPU/GPU, FPGA) и приведены формулы оценки их производительности для разнородных вычислительных нагрузок. В подразделе 2.4 описаны ключевые модели программирования: MPI, OpenMP, MapReduce/Spark и акторная модель, с выделением их математических особенностей и областей применения. В подразделе 2.5 выполнен сравнительный анализ моделей по критериям архитектуры, масштабируемости, применимости к типам задач и сложности реализации. *Вывод по разделу:* выбор архитектурного решения и модели программирования должен определяться характером вычислительной нагрузки, требованиями к масштабируемости и инфраструктурными ограничениями; для регулярных CPU-bound задач эффективны MPI и OpenMP, для обработки больших данных и I/O-bound нагрузок предпочтительны Spark и акторные модели, а гибридные архитектуры требуют тщательного распределения работы между компонентами системы.

Третий раздел «Математические модели и метрики параллельной обработки» посвящен разработке формального математического аппарата для оценки эффективности параллельных вычислений. В подразделе 3.1 дана формальная постановка задачи параллельной обработки и выведено общее выражение для времени выполнения с учётом разделения данных, объединения результатов и коммуникационных затрат. В подразделах 3.2 и 3.3 построены модели времени выполнения для CPU-bound и I/O-bound задач, учитывающие вычислительную сложность, пропускную способность подсистемы ввода-вывода и возможность асинхронного перекрытия операций. В подразделе 3.4 исследованы математические подходы к динамической балансировке нагрузки, включая алгоритм work-stealing и централизованное планирование, с выведением вероятностных и оптимизационных формул. В подразделе 3.5 предложен комбинированный аналитический вид функции накладных расходов $T_{overhead}(n) = \alpha \cdot n + \beta \cdot \ln(n+1)$ и проведено численное моделирование влияния линейных и логарифмических компонент на ускорение при различных сценариях коммуникаций. В подразделе 3.6 сформу-

лирована система метрик эффективности: ускорение, эффективность, масштабируемость, коэффициент использования ресурсов, накладные расходы и стоимость вычислений, доказана их взаимозависимость и необходимость комплексного применения. *Вывод по разделу:* разработанный математический аппарат позволяет количественно прогнозировать поведение параллельных систем; структура накладных расходов критически влияет на достижимое ускорение, а комплексное использование метрик обеспечивает объективную оценку архитектурных решений и предотвращает неэффективное использование ресурсов.

Четвёртый раздел «Обзор современных инструментов и фреймворков» посвящен исследованию программных средств реализации параллельных вычислений в экосистеме Python и распределённых сред. В подразделе 4.1 проанализированы встроенные библиотеки Python: `multiprocessing` (обход GIL, эффективен для CPU-bound задач), `threading` (эффективен для I/O-bound из-за освобождения GIL) и `asyncio` (оптимален для массовых асинхронных операций ввода-вывода), с приведением математических оценок времени выполнения для каждой стратегии. В подразделе 4.2 рассмотрены распределённые фреймворки Apache Spark, Ray и Celery, приведены оценки их времени отклика, особенности организации очередей задач и механизмы отказоустойчивости. В подразделе 4.3 сформулированы критерии выбора инструментов: характер задачи, масштаб системы, сложность реализации, требования к производительности, интеграция с инфраструктурой и экономические факторы. *Вывод по разделу:* на основе теоретического анализа разработаны рекомендации по выбору стека технологий: для прототипирования на одном компьютере оптимально использование `multiprocessing`, для кластерных систем – Ray или Spark, для интенсивных I/O-операций – `asyncio`; при этом выбор должен учитывать не только теоретическую производительность, но и стоимость поддержки, масштабируемость и специфику программной среды.

Пятый раздел «Реализация программного комплекса для экспериментального исследования» посвящен проектированию и разработке авторского модульного программного комплекса на языке Python 3.12 для эмпирической проверки математических моделей параллелизма. В подразделе 5.1 описана архитектура комплекса, состоящего из семи взаимосвязанных

модулей: конфигурации, теоретических моделей, метрик, нагрузочных задач, стратегий запуска, экспериментов и визуализации. Ключевой особенностью является единый интерфейс для четырёх стратегий параллелизма, обеспечивающий корректное сравнение на идентичных наборах данных без модификации кода задач. В подразделе 5.2 детализирована функциональность каждого модуля, включая реализацию функций расчёта ускорения по законам Амдала и Густавсона-Барсиса, модуль замера времени с многократными повторениями для статистической достоверности и набор эталонных CPU-bound/I/O-bound задач. В подразделе 5.3 описан стенд тестирования на базе процессора Intel Core i7-12700K и методика проведения измерений, включающая минимизацию влияния фоновых процессов и усреднение результатов. В подразделе 5.4 раскрыты особенности реализации стратегий `multiprocessing`, `threading` и `asyncio`, включая учёт влияния GIL, сериализации данных и накладных расходов на переключение контекста. *Вывод по разделу:* самостоятельно разработан и отлажен программный комплекс, обеспечивающий воспроизводимость экспериментов, автоматический сбор статистики и наглядную визуализацию результатов, что создало необходимую экспериментальную базу для верификации теоретических моделей и формирования практических рекомендаций.

Шестой раздел «Анализ результатов экспериментального исследования» посвящен экспериментальной верификации разработанных математических моделей, сравнительному анализу стратегий параллелизма и формированию практических рекомендаций. В подразделе 6.1 проведён эксперимент по исследованию зависимости ускорения от числа узлов. Экспериментально подтверждена применимость закона Амдала: кривая ускорения соответствует доле последовательной части $f \approx 15$–20%, при этом начиная с $n = 4$ наблюдается выход на плато ускорения $S \approx 3.7$, что указывает на достижение предела масштабируемости для данной конфигурации. В подразделе 6.2 выполнено сравнение стратегий: для CPU-bound задач `multiprocessing` обеспечил ускорение $\times 3.56$ относительно последовательного выполнения, тогда как `threading` показал минимальный выигрыш ($\times 1.09$) из-за блокировки GIL. Для I/O-bound задач `threading` и `asyncio` продемонстрировали ускорение $\times 3.26$ и $\times 2.85$ соответственно, подтвердив их эффективность при операциях ввода-вывода. В подразделе 6.3 определена точка на-

сыщения производительности: максимальное ускорение 3.59 достигается при $n_{opt} = 9$ процессах, что оптимально соответствует 8 физическим ядрам с учётом технологии Hyper-Threading. В подразделе 6.4 проведено сопоставление теории и практики: количественное расхождение моделей с экспериментом составляет 20–40%, что признано приемлемым для задач планирования ресурсов и объясняется специфическими накладными расходами платформы Python и планировщика ОС. В подразделе 6.5 на основе полученных данных сформулированы итоговые рекомендации по выбору инструментов и оптимальному числу процессов/потоков для различных классов задач и масштабов системы. *Вывод по разделу:* экспериментальные данные полностью подтвердили теоретические предсказания математических моделей; разработанный комплекс рекомендаций позволяет на практике выбирать оптимальные инструменты параллелизма и конфигурации оборудования, минимизируя накладные расходы и предотвращая деградацию производительности при чрезмерном распараллеливании, что обеспечивает высокую практическую ценность полученных результатов.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы достигнута поставленная цель — проведено исследование математической модели системы параллельной обработки данных, теоретически обоснована её эффективность, выполнена экспериментальная верификация моделей и сформулированы практические рекомендации по оптимизации вычислительных процессов. Все задачи работы решены в полном объёме.

Теоретическая часть исследования позволила систематизировать архитектурные подходы к параллельным вычислениям, сравнить фундаментальные законы масштабирования (Амдала и Густавсона-Барсиса) и выявить условия их применимости. Разработан математический аппарат для моделирования времени выполнения задач классов CPU-bound и I/O-bound, учитывающий вычислительную сложность, пропускную способность подсистем ввода-вывода, механизмы динамической балансировки нагрузки и структуру накладных расходов. Предложена комбинированная модель накладных расходов, включающая линейную и логарифмическую компоненты, что позволило качественно описать деградацию производительности при чрезмерном рас-

параллеливании. Сформулирована система взаимодополняющих метрик эффективности (ускорение, эффективность, масштабируемость, стоимость вычислений), обеспечивающая комплексную оценку архитектурных решений.

В практической части работы самостоятельно спроектирован и реализован модульный программный комплекс на языке Python 3.12, состоящий из семи взаимосвязанных модулей. Комплекс обеспечивает воспроизводимость экспериментов, автоматизированный сбор статистики и визуализацию результатов. Проведённая серия контролируемых экспериментов подтвердила теоретические предсказания: экспериментальные кривые ускорения соответствуют доле последовательной части $f \approx 15$–20%, а максимальное ускорение $S \approx 3.59$ достигается при $n_{opt} = 9$ процессах, что оптимально согласуется с 8-ядерной конфигурацией тестового стенда с учётом Hyper-Threading. Сравнительный анализ стратегий параллелизма показал, что модуль `multiprocessing` обеспечивает ускорение до $\times 3.56$ для CPU-bound задач, тогда как `threading` и `asyncio` эффективны для I/O-bound операций (ускорение до $\times 3.26$ и $\times 2.85$ соответственно). Количественное расхождение теоретических моделей с экспериментальными данными составило 20–40%, что признано допустимым для задач планирования ресурсов и объясняется спецификой среды выполнения Python и планировщика ОС.

Практическая значимость работы заключается в разработке критериальной базы и конкретных рекомендаций по выбору инструментов параллелизма и конфигурации вычислительных ресурсов. Полученные результаты могут быть непосредственно применены при проектировании систем обработки данных в областях машинного обучения, анализа больших данных и научных вычислений. Для одномашинных CPU-bound задач рекомендовано использование `multiprocessing` с числом процессов, равным количеству физических ядер; для I/O-bound нагрузок — `threading` или `asyncio`; при масштабировании на кластер — фреймворки Ray или Apache Spark.

Таким образом, исследование подтвердило высокую практическую ценность математического моделирования для проектирования и оптимизации систем параллельной обработки данных. Разработанные модели, программный комплекс и сформулированные рекомендации создают основу для дальнейших исследований в области адаптивных гетерогенных архитектур, интеллектуальной балансировки нагрузки и автоматизированного подбора па-

раметров параллельных приложений.

Основные источники информации:

1. Amdahl, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities / G. M. Amdahl // AFIPS Conference Proceedings. — 1967. — P. 483–485.
2. Густавсон, Дж. Л. Reevaluating Amdahl's Law / J. L. Gustafson // Communications of the ACM. — 1988. — Vol. 31, № 5. — P. 532–533.
3. Гергель, В. П. Основы параллельных вычислений для многопроцессорных вычислительных систем: учебное пособие / В. П. Гергель, Р. Г. Стронгин. — Н. Новгород: Изд-во ННГУ, 2003. — 184 с.
4. Воеводин, В. В. Математические модели и методы в параллельных процессах / В. В. Воеводин. — М.: Наука, 1986. — 296 с.
5. Tanenbaum, A. Distributed Systems: Principles and Paradigms / A. Tanenbaum, M. Van Steen. — 2nd ed. — Pearson Prentice Hall, 2007. — 686 p.
6. Grama, A. Introduction to Parallel Computing / A. Grama, A. Gupta, G. Karypis, V. Kumar. — Addison-Wesley, 2003. — 636 p.
7. Zaharia, M. Apache Spark: A Unified Engine for Big Data Processing / M. Zaharia [et al.] // Communications of the ACM. — 2016. — Vol. 59, № 11. — P. 56–65.
8. Daoud, M. High Performance Python: Practical Programming for High-Performance Computing / M. Daoud, I. Osborne. — 2nd ed. — O'Reilly Media, 2020. — 512 p.