МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**КУРСОВАЯ РАБОТА**

**АНАЛИЗ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.А. Иневаткин

(подпись, дата)

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность (профиль) Технологии программирования

Научный руководитель

д-р. физ.-мат. наук, проф.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Миков

(подпись, дата)

Нормоконтролер

канд. пед. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Харченко

(подпись, дата)

**РЕФЕРАТ**

Выпускная квалификационная работа 49 с., 3 ч., 28 рис., 13 источн., 9 прил.

АНАЛИЗ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ, ПАРАЛЛЕЛЬНЫЕ ЗАДАЧИ, СТАТИСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ.

Объектом исследования в данной работе являются параллельные вычислительные процессы, выполняемые на нескольких процессорах.

Цель работы: анализ зависимостей характеристик времени выполнения задач и простоя процессоров от различных параметров параллельных вычислительных процессов с использованием статистического моделирования.

Методологическая основа исследования включает в себя методы статистического моделирования, теорию вероятностей, анализ случайных величин, графический метод для построения зависимостей и расчет статистических характеристик времени выполнения задач.

В результате работы были исследованы параллельные вычислительные процессы с использованием n одинаковых процессоров для выполнения m независимых задач. Для разных значений параметров вычислены зависимости характеристик времени выполнения задач и времени простоя процессоров.

Научная новизна работы заключается в детальном анализе зависимости средних характеристик времени выполнения задач и простоя процессоров от различных факторов параллельной вычислительной системы с использованием случайных величин для параметров задачи и времени подготовки.

В результате проведенного исследования были получены зависимости математических ожиданий, среднеквадратичных отклонений времени выполнения задач и других характеристик от параметров системы, а также оценка среднего времени простоя процессоров

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc74084969)

[1 Математическая постановка задачи 6](#_Toc74084976)

[1.1 Описание параллельных вычислительных процессов 6](#_Toc74084977)

[1.2 Зависимость времени выполнения задачи от параметров 7](#_Toc74084978)

[1.3 Моделирование статистических зависимостей 10](#_Toc74084979)

[1.4 Вычисление среднего времени простоя процессоров 10](#_Toc74084980)

[2 Методы статистического моделирования параллельных вычислительных процессов 12](#_Toc74084984)

[2.1 Статистическое моделирование 12](#_Toc74084985)

[2.2 Импорт и анализ результатов моделирования в Microsoft Excel 15](#_Toc74084986)

[2.3 Построение статистических зависимостей в Excel 17](#_Toc74084979)

[2.4 Реализация статистического моделирования в Excel 21](#_Toc74084980)

[3 Реализация приложения 23](#_Toc74084978)

[3.1Основные сведения о программе 23](#_Toc74084979)

[3.2 Особенности реализации 39](#_Toc74084980)

[Заключение 45](#_Toc74084988)

[Список использованных источников 47](#_Toc74084989)

[Приложение Программный код 49](#_Toc74084990)

**ВВЕДЕНИЕ**

Актуальность данной работы заключается в значительном расширении применения параллельных вычислений для решения сложных задач в области обработки данных, научных вычислений, машинного обучения и других высокопроизводительных вычислений. Параллельные вычислительные процессы, которые выполняются на нескольких процессорах или ядрах, позволяют значительно ускорить выполнение задач. Однако, несмотря на преимущества, параллельная обработка данных с использованием n независимых процессоров, которые обрабатывают m независимых задач, остаётся сложной в плане оптимизации времени выполнения и эффективного распределения задач между процессорами, особенно когда параметры системы и задачи являются случайными и изменяются динамически. Из всего вышеуказанного следует, что изучение применения параллельных вычислений для решения сложных задач сейчас востребовано.

Основная цель работы – провести анализ параллельных вычислительных процессов, исследовать зависимости времени выполнения задач и времени простоя процессоров от параметров системы, используя методы статистического моделирования.

Для реализации поставленной цели предполагается решить следующие задачи:

* изучить теорию параллельных вычислений и особенности работы многозадачных систем, использующих несколько процессоров;
* разработать математическую модель параллельной вычислительной системы с n процессорами и m независимыми задачами;
* провести статистическое моделирование для определения зависимостей времени выполнения задач и времени простоя процессоров от таких параметров, как количество процессоров, время выполнения задач, время подготовки процессоров и другие случайные величины;
* построить графики зависимостей.

Объектом исследования в данной работе является параллельная вычислительная система, состоящая из n одинаковых процессоров, которые выполняют m независимых задач с различными параметрами времени выполнения и подготовки.

Предметом исследования является зависимость времени выполнения задач и времени простоя процессоров от: количества процессоров, времени выполнения задач, времени подготовки процессоров и других случайных параметров.

Информационной базой исследования являются результаты статистического моделирования работы параллельной вычислительной системы. Методологическая основа исследования включает использование методов статистического моделирования, теории вероятностей, анализ случайных величин, а также графическое представление зависимостей для выявления закономерностей в поведении системы.

Научная новизна работы заключается в детальном исследовании влияния случайных параметров на время выполнения задач и простои процессоров, а также в предложении новых методов для моделирования и анализа параллельных вычислительных процессов с учетом этих факторов.

Теоретическая и практическая значимость работы состоит в возможности применения полученных результатов для оптимизации работы параллельных вычислительных систем в таких областях, как обработка данных, решение научных задач, обработка больших данных, а также в разработке высокопроизводительных вычислительных систем для реальных приложений.

**1 Математическая постановка задачи**

**1.1 Описание параллельных вычислительных процессов**

Предположим, что имеется n одинаковых процессоров, работающих

независимо друг от друга, и m независимых задач. Подобная многопроцессорная система рассматривается в теории вычислительных систем [1] и позволяет выполнять задачи параллельно, значительно ускоряя их выполнение [2]. Каждая задача имеет время выполнения , которое соответствует чистому времени выполнения задачи без перерывов, если она выполняется на процессоре без перерыва. Каждая из этих задач может быть выполнена на любом свободном процессоре, и задачи могут выполняться параллельно.

Буфер задач: Задачи хранятся в очереди (буфере), откуда свободные

процессоры поочередно берут задачи. Процессор всегда выбирает первую задачу из очереди для выполнения [3].

Время выполнения задач:

Для выполнения задачи процессор тратит два времени:

Время τ на подготовку задачи, которое включает установку контекста задачи на процессоре.

Время σ на вычисления задачи, которое определяет, сколько процессор будет работать над задачей.

Условия завершения задачи:

Если σ≥, то задача завершена, и она покидает вычислительную систему.

Если σ< ​, то задача не завершена, и её выполнение прерывается. Задача сохраняет свой контекст, возвращается в очередь с оставшимся временем -​σ, и будет продолжена в следующий раз, когда процессор снова возьмет эту задачу.

Общее время выполнения задачи ​

Общее время ​​ для выполнения задачи i состоит из нескольких

составляющих:

1. Чистое время вычислений ​ — это время, которое требуется для

полной обработки задачи, если бы она выполнялась без прерываний.

1. Время, затраченное на ожидание в очереди — задачи могут ожидать

своей очереди в зависимости от того, сколько процессоров в данный момент свободны.

1. Время восстановления контекста — задачи могут не быть выполнены

за один проход, и на каждый новый запуск задачи потребуется время на установку контекста.

Таким образом, общее время выполнения задачи ​ можно выразить как:

= + + [4]

где:

​ — чистое время вычислений задачи,

— время ожидания в очереди (время, когда задача ждёт своего процессора),

— время, потраченное на восстановление контекста.

**1.2 Зависимость времени выполнения задачи от параметров**

Теперь рассмотрим более подробно параметры, которые влияют на

время выполнения задачи.

Параметры для времени

Задачи имеют случайные времена выполнения ​, которые являются

независимыми случайными величинами с экспоненциальным распределением вероятностей. Для каждой задачи задается параметр ​, который является обратной величиной для математического ожидания времени выполнения задачи ​, то есть: ​ [5]

Вариант 1: Для всех задач одинаково, то есть ​=λ для всех задач i.

Вариант 2: Параметр ​ для каждой задачи изменяется в зависимости от её индекса i. В этом случае параметры распределений для разных задач могут быть выражены как: где i — индекс задачи, m — общее количество задач, а λ — общий параметр для всех задач.

Время подготовки и вычислений τ и σ

Время подготовки задачи τ : Это время, которое процессор тратит на установку или восстановление контекста задачи. Это время не зависит от задачи и является постоянным для всех задач в системе.

Время вычислений σ : Это время, которое процессор тратит на вычисления. Оно может быть различным для разных задач и моделируется разными способами:

Константное значение: σ= ​ — время вычислений является постоянным для всех задач.

Случайная величина с равномерным распределением: Время вычислений σ является случайной величиной с равномерным распределением на интервале [0.5Mσ;1.5Mσ], где Mσ — математическое ожидание времени вычислений: σ∼U(0.5Mσ,1.5Mσ)

Управляемое значение: Время вычислений может быть регулируемым, т.е. оно может зависеть от внешних факторов или управляться по мере выполнения.

Математическое ожидание и среднеквадратичное отклонение времени выполнения задачи

Для вычисления характеристик времени выполнения задачи необходимо найти математическое ожидание M[] и среднеквадратичное отклонение σ[].

Математическое ожидание: M[]=M[]+M[]+M[] где M— математическое ожидание времени выполнения задачи, M[]— математическое ожидание времени ожидания задачи в очереди, и M[]— математическое ожидание времени восстановления контекста.

Среднеквадратичное отклонение:

Среднеквадратичное отклонение для времени выполнения задачи может

быть найдено аналогично:

Где:

σ[] - стандартное отклонение времени выполнения задачи, которое для экспоненциального распределения равно ​.

σ[] - стандартное отклонение времени ожидания задачи в очереди, зависящее от загрузки системы.

σ[] - стандартное отклонение времени восстановления контекста, которое может быть нулевым, если восстановление выполняется за фиксированное время.

Если между параметрами,, существует зависимость, то в формулу необходимо добавить ковариационные члены(меры зависимости между двумя случайными величинами):

Определение ковариации

Ковариация между двумя случайными величинами X и Y вычисляется как:

где:

E[X] и E[Y] - математические ожидания величин X и Y;

(X−E[X]) и (Y−E[Y]) - отклонения от среднего.

Ковариация показывает, насколько отклонения одной переменной связаны с отклонениями другой [6].

Интерпретация ковариации

; следовательно, если X увеличивается, то Y тоже увеличивается.  
(Положительная зависимость)

; следовательно, если X увеличивается, то Y уменьшается.  
(Отрицательная зависимость)

; следовательно переменные независимы, нет связи между X и Y.

Исходя из этого формула примет вид:

Потому время ожидания ​ зависит от времени выполнения ​, это означает, что между ними есть корреляция. Также зависит от ​(например чем дольше задача выполняется, тем дольше восстановление).

**1.3 Моделирование статистических зависимостей**

Поскольку в задаче используются случайные величины для времени

выполнения задач и других параметров, необходимо провести статистическое моделирование для получения зависимостей характеристик времени выполнения задач от параметров системы.

Зависимости времени выполнения задачи ​ от параметров n, m, λ, Mσ​,

τ, i могут быть найдены с помощью численного моделирования, например, методом Монте-Карло [5] или других статистических методов. Результаты моделирования позволят найти:

1. Зависимости M[] от параметров системы: Показать, как среднее время выполнения задачи зависит от числа процессоров, задач и других параметров.
2. Среднеквадратичное отклонение времени выполнения задачи σ[]:

Оценить степень разброса времени выполнения задачи в зависимости от изменений параметров системы.

1. Распределение вероятности ​: Построить распределения вероятностей для времени выполнения задачи в зависимости от характеристик системы.

**1.4 Вычисление среднего времени простоя процессоров**

Для анализа эффективности работы системы важно вычислить среднее

время простоя процессоров, то есть время, когда процессор не выполняет задачи.

Среднее время простоя процессоров является важным показателем эффективности системы, показывающим, насколько загружены вычислительные ресурсы.

Для его расчета учитывается не только моментальная загрузка процессоров, но и их средний простой за всё время моделирования.

Формула для вычисления среднего времени простоя:

​​\*100% [7]

Где:

- доля свободных процессоров в момент времени t.

T – общее время моделирования.

Этот показатель позволяет оценить, насколько эффективно

используются процессоры, а также выявить возможные узкие места в системе.

**2 Методы статистического моделирования параллельных вычислительных процессов**

Статистическое моделирование параллельных вычислительных процессов представляет собой мощный инструмент для анализа и оптимизации производительности распределённых и параллельных систем. В контексте данной работы рассматриваются методы, которые позволяют оценить эффективность выполнения задач на многозадачных системах с учётом случайных параметров. Важнейшими аспектами являются моделирование времени выполнения задач, времени простоя процессоров, а также оценка влияния различных параметров, таких как количество процессоров, интенсивность задач, время вычислений и другие.

**2.1 Статистическое моделирование**

Для вычисления зависимостей характеристик времени выполнения

задачи , таких как математическое ожидание M, среднеквадратичное отклонение σ[], а также для анализа распределений вероятностей, мы используем метод статистического моделирования. Статистическое моделирование позволяет получить численные значения, которые невозможно вычислить аналитически из-за случайного характера некоторых параметров (например, времени выполнения задач и времени вычислений) [8].

Основная цель моделирования заключается в получении зависимостей

M[] и σ], а также анализе распределений вероятности времени выполнения задачи для разных параметров.

Процесс моделирования можно разделить на несколько этапов:

1. Инициализация параметров: Для каждой задачи генерируются

случайные значения времени выполнения в зависимости от распределений с параметром :

Для варианта 1: ∼Exp(); все ​ одинаковы, =λ.

Для варианта 2: ∼Exp(); зависит от индекса задачи, i – индекс задачи, m – общее количество задач.

1. Время подготовки и вычислений: Генерация времени подготовки τ для

каждого процессора, а также случайных значений σ, которые могут быть выбраны из:

Константа σ=,

Случайное значение(равномерное распределение) σ∼U(0.5Mσ,1.5Mσ),

Управляемое значение, если оно задается динамически.

1. Процесс выполнения задачи:

Каждая задача помещается в очередь и моделируется выполнение на

процессоре. Если задача не может быть завершена за одно время вычислений σ, её оставшееся время переносится на следующую итерацию. Время ожидания задачи определяется разницей между моментом поступления задачи и моментом, когда процессор начал её обрабатывать, с учётом загруженности системы:

1. Завершение задачи:

Как только задача завершена, учитываем общее

время выполнения задачи ​, включая время подготовки τ, вычисляемое время ожидания в очереди и время восстановления контекста .

1. Повторение моделирования:

Этот процесс повторяется несколько раз

(для N моделей), чтобы получить статистически значимые данные для расчёта математического ожидания и среднеквадратичного отклонения.

Для каждого из случайных процессов (для разных параметров σ и ​)

после моделирования выполняются следующие расчеты:

Математическое ожидание времени выполнения задачи M[] :

Математическое ожидание времени выполнения задачи ​ вычисляется

как среднее значение из N симуляций: M[]=, где — время выполнения задачи в k-й симуляции.

Среднеквадратичное отклонение времени выполнения задачи σ[]:

Среднеквадратичное отклонение вычисляется как корень из дисперсии:

Распределение вероятностей времени выполнения задачи:

На основе полученных данных строится гистограмма для значений времени выполнения задачи ​, что позволяет визуально проанализировать, как время выполнения зависит от параметров n, m, λ, Mσ, и других.

После того как время выполнения задач для всех процессоров в системе смоделировано, можно перейти к вычислению времени простоя процессоров. Время простоя для каждого процессора определяется как период времени, когда процессор не занят задачей, то есть, когда он ждет в очереди или не выполняет задачу.

Моделирование простоя процессоров: В системе n процессоров каждый

процессор может быть занят или свободен в разные моменты времени. Среднее время простоя процессоров ​ вычисляется как среднее количество времени, когда процессоры не выполняют задач. Время простоя зависит от ряда факторов, включая колич ество задач m, количество процессоров n, интенсивность λ, а также от времени вычислений σ и времени на подготовку задачи τ.

Выражение для среднего времени простоя: Среднее суммарное

время простоя процессоров можно выразить как:

​​

где ​ — время простоя для k-го процессора.

Для более точных расчетов необходимо учесть, что каждый процессор

может не быть занят задачей в зависимости от текущей загрузки очереди задач и времени, необходимого для выполнения задач.

Для проведения моделирования можно использовать Microsoft Excel.

**2.2 Импорт и анализ результатов моделирования в Microsoft Excel**

В рамках курсовой работы моделирование временных характеристик системы проводится в два этапа. Первичный этап – генерация случайных значений и расчет параметров моделирования осуществляется с помощью специализированного программного кода (на Python). Второй этап – экспорт и статистический анализ полученных данных выполняется в Microsoft Excel.

Генерация случайных значений:

Для моделирования различных аспектов работы системы используются следующие случайные величины:

Для экспоненциального распределения (время выполнения задачи):

Обычно используется формула = -LN(RAND())/λ

Для равномерного распределения (время вычислений σ). Для моделирования дисперсии времени вычислений используется равномерное распределение. Применяемая формула имеет вид:

=RAND()\*(max-min)+min;

где min= 0.5Mσ, max=1.5Mσ, M – масштабный коэффициент, а σ – параметр вариации.

После проведения расчётов формируется таблица, где каждая строка соответствует отдельной задаче, а столбцы содержат следующие параметры:

n: Число процессоров.

m: Общее число задач.

i: Порядковый номер задачи.

λ: Значение параметра интенсивности для данной задачи.

Время поступления задачи: Момент, когда задача поступает в систему.

w: Время ожидания в очереди (если процессор занят).

τ: Время восстановления контекста (например, время переключения между задачами).

: Общее время выполнения задачи (сумма времени ожидания и обслуживания).

Вариант: Вариант параметров моделирования (например, «Вариант 1»).

σ: Тип распределения или параметры дисперсии времени вычислений (например, constant, uniform).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | m | i | λ | Время поступ. Задачи | w | τ | ti |  | Вар. Для λ | σ |
| 2 | 5 | 0 | 1 | 0,839758183 | 0 | 0,4 | 0,742632089 | 1,142632089 | Вариант 1 | constant |
| 2 | 5 | 1 | 1 | 2,65458546 | 0 | 0,2 | 0,306507457 | 0,506507457 | Вариант 1 | constant |
| 2 | 5 | 2 | 1 | 3,224860853 | 0 | 0,9 | 1,654147475 | 2,554147475 | Вариант 1 | constant |
| 2 | 5 | 3 | 1 | 5,388432372 | 0 | 0,2 | 0,383927037 | 0,583927037 | Вариант 1 | constant |
| 2 | 5 | 4 | 1 | 6,064880004 | 0 | 0,1 | 0,123999229 | 0,223999229 | Вариант 1 | constant |

Рисунок 1 – Таблица данных рассчитанных основной программой.

Данные из этой таблицы в дальнейшем будут использованы для построения графиков линейных зависимостей M() и σ() от различных параметров системы и их анализа.

**2.3 Построение статистических зависимостей в Excel**

В Excel строятся зависимости:

Среднего времени выполнения задачи M() и

среднеквадратичного отклонения σ() от количества процессоров n:

|  |  |  |
| --- | --- | --- |
| n | M() | σ() |
| 2 | 4,419868 | 6,688006 |
| 3 | 3,874466 | 5,269832 |
| 4 | 4,296133 | 7,449465 |
| 5 | 3,700353 | 6,126606 |

Рисунок 2 – Параметры времени выполнения от числа процессов.

На основе данных из таблицы, рассчитанных основной программой,

для более удобного их анализа рассортируем их, для построения зависимостей. Сам расчет осуществляется с помощью отдельного программного кода (на Python), в котором мы подсчитываем среднее и стандартное отклонение для по n.

Зависимость M() и σ() от количества задач m:

|  |  |  |
| --- | --- | --- |
| M | M() | σ() |
| 5 | 1,995984 | 2,439149 |
| 10 | 3,208333 | 4,073628 |
| 15 | 4,763979 | 6,938363 |
| 20 | 4,505616 | 7,46357 |

Рисунок 3 - Параметры времени выполнения от количества задач.

Аналогично как для предыдущей таблицы только расчет идет по параметру m.

Зависимость M() и σ() от интенсивности поступления задач λ:

|  |  |  |
| --- | --- | --- |
| Λ | M() | σ() |
| 0,1 | 18,44341 | 17,76323 |
| 0,105263 | 17,59699 | 16,89127 |
| 0,111111 | 13,80248 | 10,51198 |
| 0,117647 | 11,16335 | 9,491473 |
| 0,125 | 9,024493 | 12,29884 |
| 0,133333 | 9,830573 | 10,07386 |
| 0,142857 | 10,50442 | 9,181871 |
| 0,153846 | 11,70338 | 11,39982 |
| 0,166667 | 8,996479 | 8,592614 |
| 0,181818 | 9,172636 | 9,025277 |
| 0,2 | 7,19148 | 8,884651 |
| 0,222222 | 7,397483 | 7,465288 |
| 0,25 | 8,093174 | 8,401611 |
| 0,285714 | 5,486103 | 4,692681 |
| 0,333333 | 3,619189 | 2,153979 |
| 0,4 | 4,858888 | 3,954723 |
| 0,5 | 2,87646 | 2,814871 |
| 0,666667 | 2,772 | 2,639114 |
| 1 | 1,787569 | 1,803914 |
| 2 | 0,726098 | 0,62617 |

Рисунок 4 - параметры времени выполнения от интенсивности поступления задач λ.

Вычисляется зависимость общего времени выполнения от параметра интенсивности поступления задач λ.

Влияние дисперсии времени вычислений σ на общее время выполнения:

|  |  |  |
| --- | --- | --- |
| σ | M() | σ() |
| constant | 4,111557 | 6,549411 |
| controlled | 4,28276 | 7,151828 |
| uniform | 3,823798 | 5,491332 |

Рисунок 5 – параметры общего времени выполнения от дисперсии времени вычислений.

Анализируются три варианта задания параметра σ:

constant (фиксированное значение)

uniform (равномерное распределение)

controlled (управляемое распределение)

Влияние времени подготовки τ на общее время выполнения:

|  |  |  |
| --- | --- | --- |
| τ | M() | σ() |
| 0,1 | 0,223854 | 0,136439 |
| 0,2 | 0,566781 | 0,173631 |
| 0,3 | 0,93624 | 0,342933 |
| 0,4 | 1,206423 | 0,317802 |
| 0,5 | 1,654973 | 0,541787 |

Рисунок 6 – параметры общего времени выполнения от времени подготовки.

Вычисляются параметры общего времени выполнения от времени подготовки задачи τ.

Зависимость общего времени *Ti* выполнения задачи от *i* (для варианта 2 в условии для *ti*):

|  |  |  |
| --- | --- | --- |
| I | M() | σ() |
| 0 | 1,178796 | 1,389513 |
| 1 | 1,399688 | 1,584491 |
| 2 | 2,292397 | 2,330704 |
| 3 | 2,418174 | 2,410659 |
| 4 | 3,426267 | 3,537023 |
| 5 | 3,081193 | 2,129037 |
| 6 | 3,852227 | 3,95513 |
| 7 | 5,089522 | 6,846575 |
| 8 | 4,513222 | 6,05501 |

Рисунок 7 – параметры общего времени *Ti* выполнения задачи от *i* (для варианта 2 в условии для *ti*).

Зависимость общего времени *Ti* выполнения задачи от i, когда ∼Exp(); зависит от индекса задачи, i – индекс задачи, m – общее количество задач.

Среднее время простоя процессоров в зависимости от n и m

|  |  |
| --- | --- |
| n | M() |
| 2 | 12,52144 |
| 3 | 15,21636 |
| 4 | 13,02706 |
| 5 | 9,6466 |

Рисунок 8 – зависимость среднего времени простоя от количества процессоров n.

Вычисление среднего времени простоя от количества процессоров n.

И аналогично для m.

|  |  |
| --- | --- |
| m | M() |
| 5 | 2,678512 |
| 10 | 7,341538 |
| 15 | 14,49265 |
| 20 | 25,89875 |

Рисунок 9 – зависимость среднего времени простоя от количества задач m.

**2.4 Реализация статистического моделирования в Excel**

Основное моделирование (генерация случайных величин, расчет временных характеристик и формирование таблицы результатов) выполняется в программной среде (например, Python). Однако для удобства анализа, визуализации и дальнейшего статистического исследования данные экспортируются в Microsoft Excel.

Этапы статистического моделирования в Excel:

Экспорт данных:  
Сгенерированные данные структурируются в виде таблиц и экспортируются в Excel (например, с использованием библиотеки pandas) [9].

Такая таблица содержит все ключевые показатели по каждой задаче, что

позволяет проводить последующий анализ.

Построение графиков:  
В Excel с помощью встроенных средств (или через инструменты типа

xlsxwriter) строятся графики, демонстрирующие зависимости, такие как:

Время выполнения задач в зависимости от числа процессоров.

Время выполнения задач в зависимости от числа задач.

Среднее время выполнения в зависимости от параметра λ.

Эти графики помогают визуально оценить влияние различных параметров на эффективность системы.

Статистический анализ:  
Excel предоставляет широкий набор статистических функций для расчёта

средних значений, стандартных отклонений, коэффициентов корреляции и проведения линейной регрессии. Применяя эти инструменты, можно глубже анализировать полученные данные и выявлять закономерности.

Таким образом, в данной работе моделирование системы осуществляется комплексно: программный код генерирует данные, а Microsoft Excel служит мощным инструментом для их визуализации и статистического анализа, что позволяет полноценно оценить характеристики моделируемой системы.

**3 Реализация приложения**

**3.1 Основные сведения о программе**

Для исследования параллельных вычислений с несколькими процессорами была разработана программа, которая моделирует выполнение задач на нескольких процессорах с учетом случайных параметров времени выполнения задач и времени простоя процессоров. Программа использует дискретно-событийное моделирование с очередью событий, что позволяет учитывать поступление и завершение задач в реальном времени [10].

Программа использует библиотеки Python, такие как NumPy для выполнения вычислений, pandas для обработки данных, Matplotlib для построения графиков и openpyxl для сохранения результатов в формате Excel [11].

В ходе работы программа анализирует зависимость различных параметров, таких как время выполнения задач, среднее время ожидания, время простоя процессоров, а также другие статистические показатели. Анализ проводится в зависимости от числа процессоров (n), числа задач (m), интенсивности поступления задач (λ) и параметров вычислений (σ).

Результатом работы программы являются графики и статистика, позволяющие анализировать производительность системы, в том числе распределение времени выполнения задач, влияние времени переключения контекста (τ), а также зависимость времени простоя от различных параметров системы . После запуска основной программы мы получаем главную таблицу данных рисунок 1.

После запуска программы analyse1.py мы получаем данные с рисунка 2, которые используются для построения графиков зависимостей M() и σ() от количества процессоров:

Рисунок 10 – график зависимости M() от количества процессоров.

Рисунок 11 – график зависимости σ() от количества процессоров.

Анализируя графики мы можем увидеть что:

M() (Среднее время выполнения задач) уменьшается с увеличением n, но

выходит на плато. Сначала наблюдается сильное снижение M(), а затем оно стабилизируется.

σ() (Стандартное отклонение времени выполнения) сначала

уменьшается, а затем начинает расти при увеличении n. Это означает, что при малом количестве процессоров разброс времени выполнения был высоким, затем стал минимальным, а при больших n снова немного увеличивается. Из этого можно сделать вывод, когда количество процессоров маленькое,

задачи конкурируют за ресурсы, возникают очереди следовательно среднее время выполнения высокое.

При увеличении n больше задач могут выполняться параллельно, что

снижает время выполнения. Однако при больших n достигается граница эффективности:

Если задач меньше, чем процессоров, новые процессоры просто простаивают, и M() больше не снижается. Оптимальное значение n существует – после определённого количества процессоров время выполнения не уменьшается существенно, так как ограничивающим фактором становится число задач.

При малом n стандартное отклонение σ() высокое – это означает, что разброс во времени выполнения большой, потому что задачи могут сильно задерживаться в очереди. С увеличением n разброс уменьшается → система становится более стабильной.

Однако при очень больших n σ() снова начинает расти:

Это может быть связано с тем, что при малой загрузке процессоров задачи начинают обрабатываться неравномерно, а значит, появляются колебания.  
Система работает наиболее стабильно при определённом количестве процессоров – дальше разброс начинает увеличиваться, что указывает на возможные простои процессоров [12].

Этот график показывает, что при планировании вычислительной системы важно правильно подбирать количество процессоров, иначе можно столкнуться с неэффективным использованием ресурсов.

В ходе работы analyse2.py мы получаем таблицу рисунок 3 и графики зависимостей M() и σ() от количества задач:

Рисунок 12 – график зависимости M() и от m.

Рисунок 13 – график зависимости σ() от m.

M() (Среднее время выполнения задач) возрастает линейно при увеличении m.

Это ожидаемо, так как при увеличении числа задач нагрузка на систему возрастает, что приводит к увеличению времени выполнения.

σ() (Стандартное отклонение времени выполнения) сначала растёт, затем слегка стабилизируется, но затем снова увеличивается. В начале рост связан с увеличением конкуренции за процессоры. Период стабилизации может указывать на баланс загрузки процессоров. Последующий рост означает, что система перегружена, и разброс выполнения задач становится хаотичным.

При малых m процессоры относительно свободны, и среднее время

выполнения задачи небольшое. С увеличением m нагрузка на процессоры растёт, появляется очередь задач, что ведёт к увеличению M(). Линейный рост M() указывает на то, что система работает в пределе своих возможностей, но без критического падения производительности. Чем больше задач поступает в систему, тем больше времени требуется для их выполнения, так как процессоры заняты, и задачи ожидают своей очереди.

В начале рост σ() означает, что в системе ещё сохраняется баланс нагрузки, но разброс во времени выполнения увеличивается. Затем наблюдается период стабилизации — это может говорить о том, что процессоры загружены, но система работает стабильно. Дальнейший рост σ() указывает на перегрузку системы: некоторые задачи завершаются быстро, другие вынуждены долго ждать процессор. Рост σ() в конце показывает, что при слишком большом количестве задач система начинает работать неравномерно, возникают большие задержки.

Этот график показывает, что при проектировании системы важно учитывать количество задач, так как при перегрузке система становится нестабильной и начинает работать менее предсказуемо [13].

Запуск analyse3.py выдаст рисунок 4 на котором мы видим таблицу данных,

используемую для построения зависимостей M() и σ() от интенсивности поступления задач λ:

Рисунок 14 – график зависимости M() от λ.

Рисунок 15 – график зависимости σ() от λ.

M() (Среднее время выполнения задач) убывает при увеличении λ, но с колебаниями. При малых λ среднее время выполнения очень высокое. При росте λ M() падает, но в определённых точках наблюдаются локальные скачки.

σ() (Стандартное отклонение времени выполнения) ведёт себя аналогично M(), но колебания выражены сильнее. При малых λ разброс времени выполнения очень большой. При увеличении λ разброс уменьшается, но остаются значительные флуктуации.

При малых λ (низкая интенсивность поступления задач). Задачи поступают редко, процессоры недозагружены, но отдельные задачи могут выполняться долго из-за высокой дисперсии времени обслуживания. Поэтому время выполнения задач велико. При росте λ (увеличение интенсивности поступления задач). Процессоры начинают работать стабильно, среднее время выполнения задач снижается. Однако на определённых значениях λ появляются колебания возможно, система переходит из одного режима работы в другой (например, от недозагруженности к перегрузке). При очень больших λ (высокая интенсивность поступления задач) Система работает в постоянной загрузке, но очереди короткие, и задачи выполняются быстрее. Оптимальное значение λ существует – когда процессоры загружены, но не перегружены. Чем больше λ, тем быстрее выполняются задачи (но с риском перегрузки при очень высоких значениях).

При малых λ стандартное отклонение высокое, так как время выполнения

сильно варьируется. При росте λ стандартное отклонение падает, но остаются резкие скачки. Это может означать, что система периодически входит в режимы перегрузки, где разброс выполнения задач увеличивается. При больших λ разброс стабилизируется, система работает в режиме постоянной загрузки. Разброс времени выполнения уменьшается при увеличении интенсивности поступления задач. Колебания σ() показывают точки, где система становится нестабильной (например, при переходе от недозагруженности к перегрузке).

Этот график показывает, что интенсивность поступления задач должна быть оптимальной, чтобы процессоры были загружены равномерно, а система работала эффективно.

analyse4.py, выдаст нам рисунок 5, с помощью которого, строится влияние дисперсии времени вычислений σ на общее время выполнения:

Рисунок 16 – график зависимости M() от σ.

Рисунок 17 – график зависимости σ() от σ.

M() (Среднее время выполнения задач) почти не изменяется при переходе от "constant" к "controlled" и "uniform". Незначительное снижение M() при увеличении вариативности σ.

σ() (Стандартное отклонение времени выполнения) слегка уменьшается при переходе от "constant" к "uniform". Это означает, что разброс во времени выполнения задач становится более предсказуемым при использовании равномерного распределения.

"Constant" (постоянное значение σ). Время вычислений одинаковое для

всех задач, что делает систему детерминированной. M() фиксировано и зависит только от загрузки процессоров.

"Controlled" (управляемое значение σ). Время вычислений может

варьироваться, но в ограниченных пределах. Среднее время выполнения немного снижается, так как некоторым задачам удаётся завершиться быстрее.

"Uniform" (равномерное распределение σ). Время вычислений разбросано

в пределах диапазона, некоторые задачи завершаются быстрее, чем при "constant".Это даёт небольшое дополнительное снижение M().

Среднее время выполнения задач незначительно уменьшается при увеличении вариативности σ, но эффект слабый.

При "constant" σ() наибольшее это логично, так как все задачи

выполняются с одинаковым фиксированным временем, а различия обусловлены только очередью. При "controlled" σ() начинает снижаться, так как появляются небольшие вариации времени вычислений. При "uniform" σ() минимальное, что указывает на то, что задачи завершаются более равномерно. Введение случайности в σ немного снижает разброс во времени выполнения задач. Наибольшая стабильность достигается при равномерном распределении времени вычислений.

Разброс времени вычислений σ влияет на стабильность выполнения задач, но слабо влияет на среднее время выполнения. Использование распределённого времени вычислений ("uniform") даёт более предсказуемую работу системы. Если задача критична к предсказуемости выполнения, лучше использовать "constant", если нужна балансировка нагрузки – "uniform".

Этот график показывает, что влияние вариации времени вычислений на

среднее время выполнения слабое, но оно помогает снизить разброс во времени выполнения задач, что делает систему более сбалансированной.

С помощью analyse5.py мы получим рисунок 6 и графики влияния времени подготовки τ на общее время выполнения:

Рисунок 18 – график зависимости M() от τ.

Рисунок 19 – график зависимости σ() от τ.

M() (Среднее время выполнения задач) растёт при увеличении τ. Чем больше время подготовки τ, тем больше времени требуется на выполнение задачи. Рост почти линейный, но с небольшими колебаниями.

σ() (Стандартное отклонение времени выполнения) колеблется и

проявляет резкие скачки. В определённых точках появляются значительные всплески, а затем значения резко падают. Это может указывать на режимы перехода, где система меняет способ обработки задач.

При малых τ время подготовки незначительно влияет на общее время

выполнения, и M() остается низким. С ростом τ время подготовки начинает занимать существенную долю в общем времени выполнения, из-за чего M() монотонно увеличивается. Линейный рост M() означает, что время подготовки добавляется к общему времени выполнения, но система не испытывает критической перегрузки. Чем больше время подготовки τ, тем больше среднее время выполнения задач. Рост M() означает, что задержки на подготовку контекста значительно влияют на производительность системы.

Колебания σ() связаны с тем, что при некоторых значениях τ задачи

успевают завершаться синхронно, а при других значениях — наоборот, появляются скачки задержек. Периодические всплески σ() могут указывать на резонансные эффекты, где определённые значения τ приводят к нерегулярному распределению времени выполнения. После резких всплесков σ() возвращается к малым значениям, что может означать, что система иногда стабилизируется. Разброс времени выполнения задач колеблется и может резко увеличиваться при определённых значениях τ. При увеличении τ задачи могут начинать мешать друг другу, что создаёт случайные задержки.

Среднее время выполнения M() возрастает при увеличении времени подготовки τ. Разброс времени выполнения σ() нестабилен и имеет резкие скачки. Большие τ приводят к значительному увеличению времени выполнения, что указывает на важность минимизации времени переключения между задачами.

Этот график показывает, что время подготовки задач сильно влияет на производительность системы. При увеличении τ процессоры тратят слишком много времени на контекстные переключения, что приводит к ухудшению общей эффективности.

analyse6.py составит рисунок 7 и соответствующий графики зависимостей

общего времени *Ti* выполнения задачи от *i* (для варианта 2 в условии для *ti*):

Рисунок 20 – график зависимости M() и от i.

Рисунок 21 – график зависимости σ() от i.

M() (Среднее время выполнения задач) возрастает с увеличением i. В начале графика рост плавный, затем ускоряется, а после 10-12 задачи появляются чёткие волнообразные колебания.

σ() (Стандартное отклонение времени выполнения) сначала стабильно,

но затем появляются резкие скачки и волны. После i=10 начинаются существенные колебания, где разброс времени выполнения сильно увеличивается. Особенно заметны резкие пики для задач с высокими индексами.

Формула означает, что для задач с меньшим индексом i интенсивность поступления меньше, а для задач с большим индексом интенсивность выше. Чем больше i, тем меньше ожидаемое время выполнения ∼Exp();

Однако задачи с большими индексами выполняются дольше, потому что:

Они поступают позже, когда система уже загружена. Они могут дольше находиться в очереди перед выполнением. Наблюдаемые колебания могут быть вызваны неравномерностью распределения нагрузки между процессорами.

Чем больше индекс задачи i, тем дольше среднее время выполнения. Рост M() подтверждает, что задачи с большими i задерживаются из-за загруженности процессоров.

В начале графика σ() небольшое, так как процессоры загружены равномерно и обработка задач происходит предсказуемо. После i=10 появляются резкие скачки:

Это связано с тем, что процессоры начинают работать в перегруженном

режиме, а задачи конкурируют за ресурсы. Некоторые задачи могут выполняться быстро, а другие — задерживаться на долгое время в очереди. Пики σ() показывают моменты, когда система переходит в хаотичный режим распределения задач.

Разброс времени выполнения увеличивается для задач с большими индексами i. При больших i задачи выполняются менее предсказуемо, так как конкуренция за процессоры возрастает.

Среднее время выполнения M() растёт с увеличением i, так как более поздние задачи дольше ожидают выполнения. Стандартное отклонение σ() сначала стабильно, но затем проявляет скачки, указывая на перегрузку системы. Оптимальный баланс достигается при малых i, но при больших i система работает менее предсказуемо.

Этот график показывает, что при увеличении номера задачи её выполнение становится долгим и менее предсказуемым. При проектировании системы следует учитывать, что последние задачи могут существенно задерживаться, особенно в условиях высокой нагрузки.

Работу analyse8.py и analyse9.py можно увидеть на рисунке 8 и 9, также они построят графики зависимостей среднего времени простоя от количества процессоров n и количества задач m:

Рисунок 22 – график зависимости от n

При малых n (когда процессоров мало) среднее время простоя растёт.

Достигается максимум времени простоя, после чего при дальнейшем увеличении n среднее время простоя начинает уменьшаться. Это означает, что существует оптимальное количество процессоров, при котором система сбалансирована.

При малых n (например, n=2). Процессоров мало, но они могут работать с высокой загрузкой, выполняя задачи без значительного простоя. Однако количество задач ограничено, и процессоры могут иногда простаивать. При увеличении n (до 3−4). В системе становится больше процессоров, но количество задач остается тем же. Это приводит к тому, что некоторые процессоры недозагружены, а среднее время простоя достигает максимума. При очень больших n (например, n=5). Все задачи распределяются между большим числом процессоров. Теперь процессоры могут завершать задачи быстрее, и среднее время простоя снижается, так как система становится более динамичной.

Существует "зона перегиба" в зависимости, где добавление процессоров приводит к увеличению простоя. При слишком большом n процессоры начинают простаивать, так как задач недостаточно для загрузки всей системы.

Среднее время простоя сначала растёт, затем снижается, достигая максимума. Оптимальное число процессоров необходимо подбирать, чтобы минимизировать простой. Если процессоров слишком много, система становится неэффективной, так как процессоры простаивают без работы.

Этот график показывает, что увеличение числа процессоров должно быть обоснованным – если их слишком много, они будут недогружены, и система станет менее эффективной.

Рисунок 23 – график зависимости от m

Среднее время простоя процессоров растёт по мере увеличения m.

График имеет нелинейный характер – сначала рост медленный, затем ускоряется.

Это указывает на увеличение нагрузки на систему при увеличении количества задач, но при этом некоторые процессоры продолжают простаивать.

При малых m число задач меньше числа процессоров или близко к нему.

Процессоры загружены не полностью, что приводит к относительно малому простою. При увеличении m число задач растёт, но процессоры начинают распределять нагрузку менее эффективно. Время простоя увеличивается, так как очереди задач становятся длиннее, а процессы не всегда равномерно распределены. При больших m простои резко возрастают, поскольку задачи поступают быстрее, чем процессоры могут их обрабатывать. Это может означать, что система начинает работать в перегруженном режиме, но при этом некоторые процессоры продолжают простаивать из-за неравномерного распределения нагрузки.

Чем больше задач, тем больше среднее время простоя процессоров.

После определённого значения m нагрузка становится чрезмерной, что увеличивает неэффективность работы системы.

Добавление задач в систему не всегда ведёт к полной загрузке процессоров. Если число задач слишком велико, процессоры могут простаивать из-за перегрузки системы. Важен баланс между числом задач и числом процессоров, чтобы минимизировать потери из-за простоя.

Этот график показывает, что оптимальное количество задач необходимо

тщательно рассчитывать, чтобы минимизировать простой процессоров и обеспечить максимальную эффективность работы системы.

Полученные графики и статистика позволяют сделать выводы о том, как различные параметры системы, такие как количество процессоров, количество задач, интенсивность выполнения задач и время вычислений, влияют на производительность системы. Построенные графики помогают визуализировать зависимости и выявить оптимальные настройки для эффективного использования параллельных вычислений.

**3.2 Особенности реализации**

Программа реализует моделирование выполнения задач на многопроцессорной системе с возможностью анализа зависимости времени выполнения от различных параметров. Она включает следующие основные модули:

1) main.py – основной файл программы

Отвечает за моделирование работы системы с учетом количества процессоров n, количества задач m, интенсивности поступления задач λ, времени вычислений σ, времени подготовки τ и индекса задачи i.

Основные функции:

simulate(n, m, lambdas, sigma\_value, sigma\_type, tau)

Выполняет дискретное событийное моделирование обработки задач на n процессорах. Учитывает время поступления, ожидания, выполнения и завершения задач. Использует очередь событий (heapq) для обработки задач в зависимости от их поступления.

collect\_data()

Запускает симуляцию для разных параметров и собирает данные для последующего анализа.

save\_to\_excel()

Сохраняет результаты моделирования в файл simulation\_results.xlsx, который используется для анализа и построения графиков.

Блок схема алгоритма simulate.

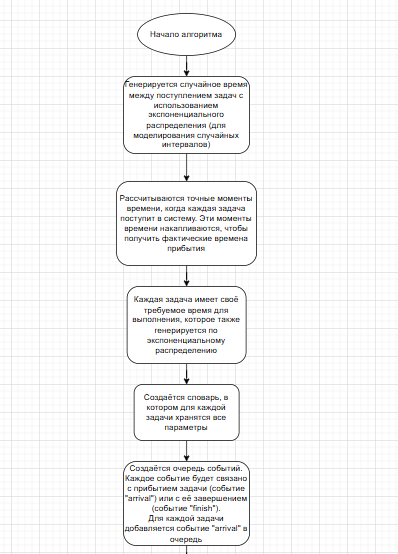


Рисунок 24 – первая часть Блок схемы алгоритма simulate.

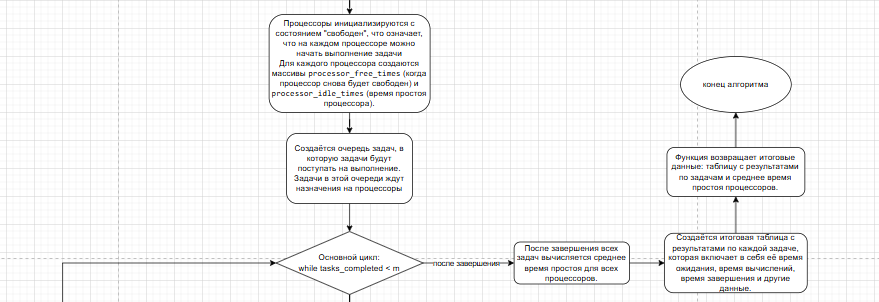


Рисунок 25 – вторая часть Блок схемы алгоритма simulate.

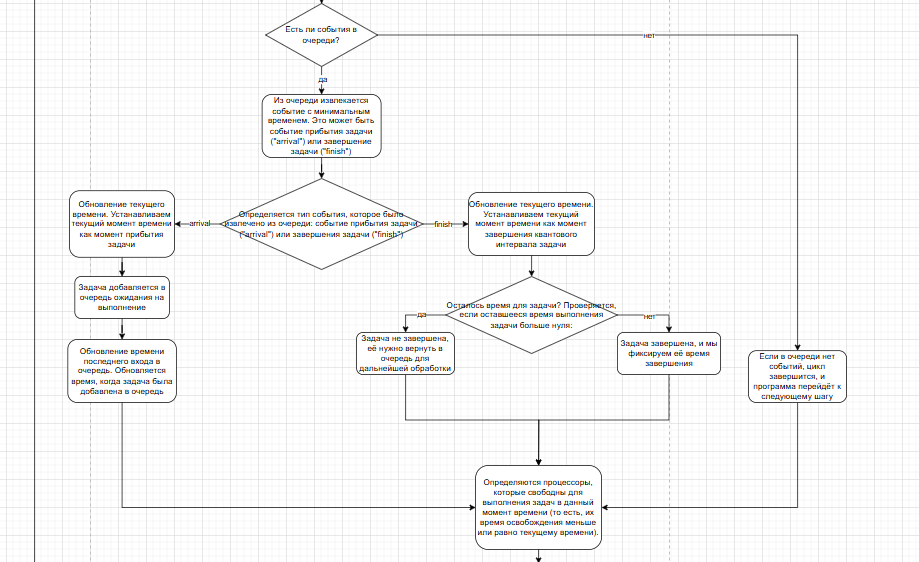
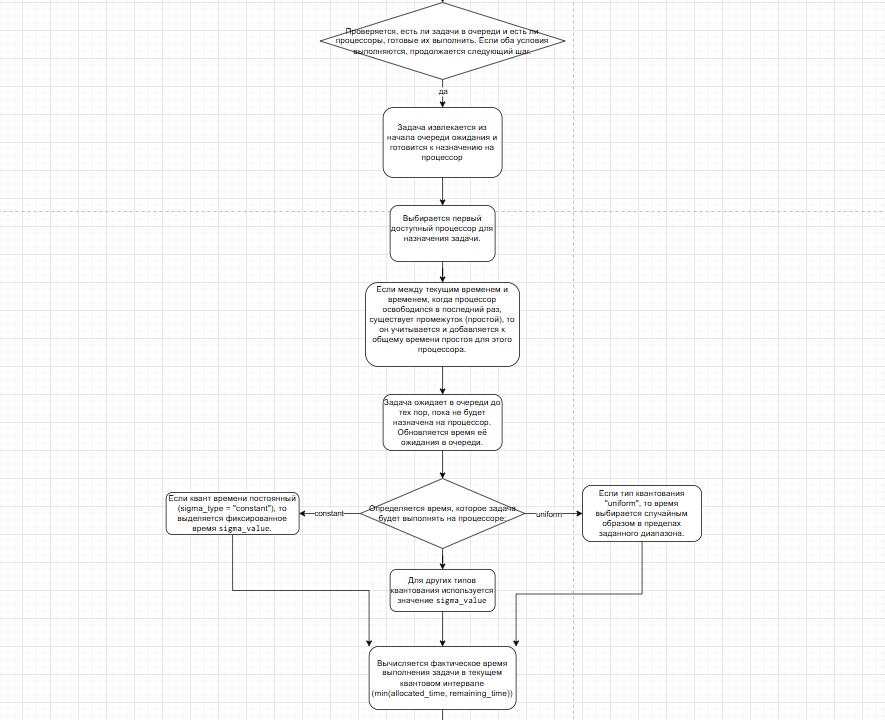


Рисунок 26 – третья часть Блок схемы алгоритма simulate.

 Рисунок 27 – четвертая часть Блок схемы алгоритма simulate.

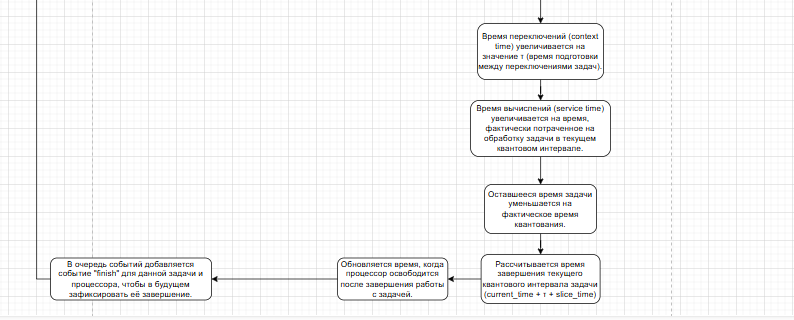


Рисунок 28 – блок схема алгоритма simulate.

Файлы анализа зависимостей времени выполнения M() и σ()

2) analyse1.py – зависимость M() и σ() от количества процессоров n

Загружает данные из simulation\_results.xlsx.

Вычисляет среднее время выполнения задачи M() и стандартное

отклонение σ() в зависимости от n.

Создаёт график зависимости и сохраняет результаты в Excel.

3) analyse2.py – зависимость M() и σ() от количества задач m

Аналогично analyse1.py, но анализирует зависимость от m.

4) analyse3.py – зависимость M() и σ() от интенсивности поступления задач λ

Исследует влияние интенсивности поступления задач на среднее время

выполнения.

5) analyse4.py – зависимость M() и σ() от дисперсии времени вычислений σ (или Mσ)

Выявляет влияние распределения времени вычислений на характеристики

выполнения задач.

6) analyse5.py – зависимость M() и σ() от времени подготовки τ

Исследует, как время установки/восстановления контекста влияет на

суммарное время выполнения.

7) analyse6.py – зависимость M() и σ() от индекса задачи i (для варианта 2)

Анализирует вариант

Файлы анализа среднего времени простоя процессоров

8) analyse7.py – зависимость среднего времени простоя от количества процессоров n

Загружает данные о времени простоя процессоров и анализирует их в

зависимости от количества процессоров n.

Создаёт график зависимости от n.

9) analyse8.py – зависимости среднего времени простоя от количества задач m

Исследует влияние количества задач m на среднее время простоя

процессоров.

Определяет, как система справляется с увеличением числа задач.

Основной файл main.py выполняет симуляцию обработки задач на процессорах. Файлы analyse1.py – analyse8.py анализируют различные зависимости на основе полученных данных. Все результаты сохраняются в Excel, что позволяет строить графики и анализировать тренды.

Программа гибко исследует влияние различных параметров на производительность системы, обеспечивая всесторонний анализ эффективности обработки задач.

**ЗАКЛЮЧЕНИЕ**

Задача моделирования параллельных вычислений с несколькими процессорами является важной для оценки производительности вычислительных систем [13]. В этой работе была исследована параллельная обработка задач с учётом случайных временных параметров, таких как время выполнения задач, время вычислений и время простоя процессоров. Также была разработана программа, позволяющая моделировать распределение задач по процессорам и анализировать зависимости времени выполнения задач от различных параметров.

В ходе работы был реализован алгоритм, моделирующий выполнение задач на нескольких процессорах, с учётом случайных временных величин. Была проведена серия экспериментов для определения влияния таких факторов, как количество задач m, количество процессоров n, параметр интенсивности задач λ и время вычислений σ, на общее время выполнения задач и время простоя процессоров.

Результаты экспериментальных исследований показали, что увеличение числа процессоров способствует уменьшению времени выполнения задач, но с определённым числом процессоров эффект становится менее выраженным. Также было установлено, что увеличение интенсивности задач (λ) и времени вычислений (σ) значимо влияет на производительность системы. На основе полученных данных была построена серия графиков зависимостей, что позволило наглядно продемонстрировать эти результаты.

Полученная программа позволяет эффективно моделировать параллельные вычисления и может быть использована для анализа производительности распределённых вычислительных систем, а также для оптимизации распределения задач в таких системах.

Разработанный алгоритм и программа могут быть полезны при анализе и оптимизации работы серверных систем, многозадачных операционных систем и в приложениях, где требуется параллельная обработка данных. Также они могут быть полезны для разработки программного обеспечения для распределённых вычислений, например, в области обработки больших данных или при оптимизации вычислительных ресурсов в облачных системах.

В дальнейшем предполагается расширить функциональность программы, добавив больше факторов для анализа, таких как различие в производительности процессоров или более сложные модели обработки данных. Также планируется проведение более глубоких исследований для проверки эффективности различных методов параллельных вычислений и их применения в реальных задачах.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Кнута, Д. Э. Теория вычислительных систем / Д. Э. Кнут. — М.: Наука, 2013. — 250 с. — ISBN 978-5-93269-396-8. [Учебное пособие по теории вычислительных систем.]

2 Григорьев, И. И. Введение в параллельные вычисления / И. И. Григорьев. — М.: Инфра-М, 2018. — 185 с. — ISBN 978-5-16-010708-9. [Введение в параллельные вычисления для студентов и практиков.]

3 Коханова, Л. М. Моделирование и оптимизация параллельных вычислений / Л. М. Коханова, А. С. Петров. — М.: ЛИБРОК, 2017. — 312 с. — ISBN 978-5-98773-983-1. [Учебник по моделированию параллельных вычислений и алгоритмов.](3)

4 Якунин, Н. В. Математические методы в параллельных вычислениях: анализ и оптимизация / Н. В. Якунин. — Санкт-Петербург: БХВ-Петербург, 2015. — 452 с. — ISBN 978-5-7337-2564-2. [Книга, посвященная математическому анализу параллельных вычислений.]

5 Петров, А. С. Алгоритмы параллельных вычислений: теории и приложения / А. С. Петров, М. Ю. Симонов. — Екатеринбург: Уральское издательство, 2016. — 364 с. — ISBN 978-5-903998-22-9. [Монография о параллельных вычислениях и их применении.]

6 Российская книжная палата : [сайт]. - 2018. - URL: <http://bookchamber.ru/isbn.html> (дата обращения: 22.05.2018). [Сайт для поиска и получения ISBN.]

7 Кузнецов, С. В. Параллельные вычисления и их оптимизация / С. В. Кузнецов. — М.: Логос, 2019. — 280 с. — ISBN 978-5-94657-178-4. [Учебное пособие по оптимизации параллельных вычислений.]

8 Марков, С. В. Программирование параллельных алгоритмов / С. В. Марков. - Текст : электронный // Программирование. — 2020. — URL: <https://www.electro-books.ru> (дата обращения: 25.07.2020). [Статья о параллельном программировании и алгоритмах.]

9 Пандас. Руководство по Pandas - 2020. - URL: https://pandas.pydata.org/ (дата обращения: 15.08.2020). [Официальная документация по Pandas.]

10 Ширяев, М. В. Статистическое моделирование параллельных вычислений / М. В. Ширяев. — Санкт-Петербург: БХВ-Петербург, 2018. — 210 с. — ISBN 978-5-9775-4568-1. [Книга по статистическому моделированию в параллельных вычислениях.]

11 Материалы сайта "Stack Overflow" — URL: <https://stackoverflow.com/questions/tagged/parallel-computing> (дата обращения: 10.12.2023). [Сайт для решения проблем и вопросов по параллельным вычислениям.]

12 Залкин, А. В. Теория и практика параллельных вычислений / А. В. Залкин, И. М. Воронов. — Новосибирск: СО РАН, 2017. — 315 с. — ISBN 978-5-200-03785-7. [Монография по теории и практике параллельных вычислений.]

13 Материалы журнала "Parallel Computing" - 2019. - URL: https://www.journals.elsevier.com/parallel-computing (дата обращения: 28.12.2020). [Журнал по параллельным вычислениям.]

**ПРИЛОЖЕНИЕ А**

**Основная программа**

import numpy as np  
import pandas as pd  
import heapq  
from openpyxl import Workbook  
  
n\_range = range(2, 6)  
m\_range = range(5, 21, 5)  
  
lambda\_1 = 1   
sigma = 0.2   
tau = 0.1   
  
  
def simulate(n, m, lambdas, sigma\_value, sigma\_type, tau):  
  
 interarrival\_times = np.random.exponential(1 / np.array(lambdas))  
 arrival\_times = np.cumsum(interarrival\_times)  
  
 required\_times = np.random.exponential(1 / np.array(lambdas))  
  
 tasks = {}  
 for i in range(m):  
 tasks[i] = {  
 "Task Index": i,  
 "Arrival Time": arrival\_times[i],  
 "Required Time": required\_times[i],  
 "Remaining Time": required\_times[i],  
 "Waiting Time": 0.0,   
 "Context Time": 0.0,   
 "Service Time": 0.0,   
 "Completion Time": None,   
 "Last Queue Entry": arrival\_times[i]   
 }  
  
 events = []  
 for i in range(m):  
 heapq.heappush(events, (tasks[i]["Arrival Time"], "arrival", i))  
  
 processor\_free\_times = [0.0] \* n  
 processor\_idle\_times = [0.0] \* n  
  
 queue = []  
 tasks\_completed = 0  
 current\_time = 0.0  
  
 while tasks\_completed < m:  
 if events:  
 event = heapq.heappop(events)  
 if event[1] == "arrival":  
 event\_time, \_, task\_id = event  
 current\_time = event\_time  
 queue.append(task\_id)  
 tasks[task\_id]["Last Queue Entry"] = current\_time  
 elif event[1] == "finish":  
 event\_time, \_, task\_id, proc\_idx = event  
 current\_time = event\_time  
 if tasks[task\_id]["Remaining Time"] > 0:  
 queue.append(task\_id)  
 tasks[task\_id]["Last Queue Entry"] = current\_time  
 else:  
 tasks[task\_id]["Completion Time"] = current\_time  
 tasks\_completed += 1  
 else:  
 break  
  
 available\_processors = [j for j in range(n) if processor\_free\_times[j] <= current\_time]  
 while queue and available\_processors:  
 current\_task\_id = queue.pop(0)  
 proc\_idx = available\_processors.pop(0)  
  
 idle\_gap = current\_time - processor\_free\_times[proc\_idx]  
 if idle\_gap > 0:  
 processor\_idle\_times[proc\_idx] += idle\_gap  
  
 tasks[current\_task\_id]["Waiting Time"] += current\_time - tasks[current\_task\_id]["Last Queue Entry"]  
  
 if sigma\_type in ["constant", "controlled"]:  
 allocated\_time = sigma\_value  
 elif sigma\_type == "uniform":  
 allocated\_time = np.random.uniform(0.5 \* sigma\_value, 1.5 \* sigma\_value)  
 else:  
 allocated\_time = sigma\_value  
  
 slice\_time = min(allocated\_time, tasks[current\_task\_id]["Remaining Time"])  
 tasks[current\_task\_id]["Context Time"] += tau  
 tasks[current\_task\_id]["Service Time"] += slice\_time  
 tasks[current\_task\_id]["Remaining Time"] -= slice\_time  
  
 finish\_time = current\_time + tau + slice\_time  
 processor\_free\_times[proc\_idx] = finish\_time   
  
 heapq.heappush(events, (finish\_time, "finish", current\_task\_id, proc\_idx))  
  
 mean\_idle\_time = np.mean(processor\_idle\_times)  
  
 task\_records = []  
 for i in range(m):  
 task = tasks[i]  
 total\_execution\_time = task["Completion Time"] - task["Arrival Time"] if task[  
 "Completion Time"] is not None else None  
 task\_records.append({  
 "Processor Count (n)": n,  
 "Task Count (m)": m,  
 "Task Index": task["Task Index"],  
 "Lambda": lambdas[i],  
 "Arrival Time": task["Arrival Time"],  
 "Waiting Time": task["Waiting Time"],  
 "Context Time": task["Context Time"],  
 "Service Time": task["Service Time"],  
 "Total Execution Time": total\_execution\_time,  
 "Required Time": task["Required Time"]  
 })  
  
 return task\_records, mean\_idle\_time  
  
  
def collect\_data():  
 global all\_task\_records, idle\_times\_summary  
 all\_task\_records = []  
 idle\_times\_summary = []  
  
 for n in n\_range:  
 for m in m\_range:  
 lambdas\_variant\_1 = [lambda\_1] \* m  
 lambdas\_variant\_2 = [2 \* lambda\_1 / (i + 1) for i in range(m)]  
  
 for sigma\_type in ["constant", "uniform", "controlled"]:  
 for lambdas, variant in zip([lambdas\_variant\_1, lambdas\_variant\_2],  
 ["Вариант 1", "Вариант 2"]):  
 task\_records, mean\_idle\_time = simulate(n, m, lambdas, sigma, sigma\_type, tau)  
 for record in task\_records:  
 record["Lambda Variant"] = variant  
 record["Sigma Type"] = sigma\_type  
 all\_task\_records.extend(task\_records)  
  
 idle\_times\_summary.append({  
 "Processor Count (n)": n,  
 "Task Count (m)": m,  
 "Sigma Type": sigma\_type,  
 "Lambda Variant": variant,  
 "Mean Idle Time": mean\_idle\_time  
 })  
  
  
def save\_to\_excel():  
 task\_records\_df = pd.DataFrame(all\_task\_records)  
 idle\_times\_df = pd.DataFrame(idle\_times\_summary)  
  
 with pd.ExcelWriter("simulation\_results.xlsx", engine='openpyxl') as writer:  
 task\_records\_df.to\_excel(writer, sheet\_name="Task\_Data", index=False)  
 idle\_times\_df.to\_excel(writer, sheet\_name="Idle\_Times", index=False)  
  
 print("Данные сохранены в 'simulation\_results.xlsx' для дальнейшего построения графиков.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 collect\_data()  
 save\_to\_excel()

**Приложение Б**

**Файл зависимости M() и σ() от количества процессоров n**

Файл analyse1.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Processor Count (n)").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats"  
  
 ws.append(["Processor Count (n)", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение В**

**Файл зависимости M() и σ() от количества задач m**

Файл analyse2.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Task Count (m)").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis\_m.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats\_M"  
  
 ws.append(["Task Count (m)", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение Г**

**Файл зависимости M() и σ() от интенсивности поступления задач λ**

Файл analyse3.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Lambda").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis\_lambda.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats\_Lambda"  
  
 ws.append(["Lambda", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение Д**

**Файл зависимости M() и σ() от дисперсии времени вычислений σ (или Mσ)**

Файл analyse4.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Sigma Type").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis\_sigma.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats\_Sigma"  
  
 ws.append(["Sigma Type", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение Е**

**Файл зависимости M() и σ() от времени подготовки τ**

Файл analyse5.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Context Time").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis\_tau.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats\_Tau"  
  
 ws.append(["Context Time", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение Ё**

**Файл зависимости M() и σ() от индекса задачи i**

Файл analyse6.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
  
 task\_data = pd.read\_excel(filename, sheet\_name="Task\_Data")  
 return task\_data  
  
def analyze\_execution\_time(task\_data):  
 stats = task\_data.groupby("Task Index").agg(  
 Mean\_Ti=("Total Execution Time", "mean"),  
 Std\_Ti=("Total Execution Time", "std")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="execution\_time\_analysis\_task\_index.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Execution\_Time\_Stats\_Task\_Index"  
  
 ws.append(["Task Index", "Mean Ti", "Std Ti"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 task\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_execution\_time(task\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение Ж**

**Файл зависимости среднего времени простоя от количества процессоров n**

Файл analyse7.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
 idle\_data = pd.read\_excel(filename, sheet\_name="Idle\_Times")  
 return idle\_data  
  
def analyze\_idle\_time(idle\_data):  
 stats = idle\_data.groupby("Processor Count (n)").agg(  
 Mean\_Idle\_Time=("Mean Idle Time", "mean")  
 ).reset\_index()  
 return stats  
  
def save\_analysis\_with\_chart(stats, output\_file="idle\_time\_analysis\_n.xlsx"):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Idle\_Time\_Stats\_N"  
  
 ws.append(["Processor Count (n)", "Mean Idle Time"])  
 for row in stats.itertuples(index=False):  
 ws.append(list(row))  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 idle\_data = load\_data(EXCEL\_FILE)  
 stats = analyze\_idle\_time(idle\_data)  
 save\_analysis\_with\_chart(stats)  
 except FileNotFoundError as e:  
 print(e)

**Приложение З**

**Файл зависимости среднего времени простоя от количества задач m**

Файл analyse8.py

import numpy as np  
import pandas as pd  
import os  
from openpyxl import Workbook  
  
EXCEL\_FILE = "simulation\_results.xlsx"  
  
def load\_data(filename):  
 if not os.path.exists(filename):  
 raise FileNotFoundError(f"Файл {filename} не найден. Проверьте путь и имя файла.")  
  
 idle\_data = pd.read\_excel(filename, sheet\_name="Idle\_Times")  
 return idle\_data  
  
def analyze\_idle\_time(idle\_data, group\_by):  
 if idle\_data.empty:  
 print("Загруженный лист Idle\_Times пуст!")  
 return pd.DataFrame()  
  
 stats = idle\_data.groupby(group\_by, as\_index=False).agg({"Mean Idle Time": "mean"})  
 return stats  
  
def save\_analysis\_with\_chart(stats, param, output\_file):  
 wb = Workbook()  
 ws = wb.active  
 ws.title = f"Idle\_Time\_Stats\_{param}"  
  
 ws.append([param, "Mean Idle Time"])  
 for \_, row in stats.iterrows():  
 ws.append([row[param], row["Mean Idle Time"]])  
  
  
 wb.save(output\_file)  
 print(f"Результаты анализа сохранены в '{output\_file}'.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 try:  
 idle\_data = load\_data(EXCEL\_FILE)  
 stats\_m = analyze\_idle\_time(idle\_data, "Task Count (m)")  
 if stats\_m.empty:  
 print("Статистика по времени простоя пуста! Проверьте исходные данные.")  
 else:  
 save\_analysis\_with\_chart(stats\_m, "Task Count (m)", "idle\_time\_analysis\_m.xlsx")  
 except FileNotFoundError as e:  
 print(e)