Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РАДИОТЕХНИЧЕСКИЙ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Имитационная модель микросервисной архитектуры приложения

Студент РТ5-71Б _____ **В.М. Кабанец**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ **М.В. Черненький**
(Подпись, дата) (И.О.Фамилия)

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ5____
(Индекс)
_____ ___В.И. Терехов__
(И.О.Фамилия)
«_04_» ____сентября_____ 2024 г.

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине _Имитационное моделирование дискретных процессов_____
_____

Студент группы _РТ5-71Б__

_____Кабанец В.М._____
(Фамилия, имя, отчество)

Тема курсовой работы Имитационная модель микросервисной архитектуры приложения
_____
_____
_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
_____УЧЕБНАЯ_____
Источник тематики (кафедра, предприятие, НИР) _____КАФЕДРА_____

График выполнения работы: 25% к _3_ нед., 50% к _8_ нед., 75% к 12 нед., 100% к 15 нед.

Задание Разработать имитационную модель микросервисной архитектуры приложения
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на _37__ листах формата А4.
_____
_____
_____

Дата выдачи задания «_02_» ____октября_____ 2024 г.

Руководитель курсовой работы _____ ____М.В. Черненький__
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____В.М. Кабанец____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

Аннотация

Данная курсовая работа посвящёна моделированию микросервисной архитектуры с использованием дискретно-событийной симуляции на базе фреймворка SimPy. Основная цель работы — исследовать поведение системы при различных параметрах нагрузки, вероятностях отказа и задержках в компонентах, а также оценить влияние нескольких пользователей, одновременно генерирующих запросы, на производительность и устойчивость к сбоям.

Модель позволяет задавать различные параметры через конфигурационный файл или графический интерфейс на базе Streamlit, проводить эксперименты и визуализировать результаты в виде графиков. Изучено влияние увеличения числа пользователей, интенсивности запросов, вероятностей отказа и задержек на количество ошибок, среднее время ответа и загрузку сервисов.

Полученные результаты могут быть использованы для предварительного анализа поведения микросервисных систем, проверки гипотез о масштабируемости и устойчивости к сбоям, а также для принятия инженерных решений по настройке параметров системы и её компонентов.

ВВЕДЕНИЕ

Современные программные системы всё чаще строятся по микросервисному принципу, в котором функциональность разбивается на независимые сервисы, взаимодействующие по сетевым протоколам. Такой подход повышает масштабируемость, упрощает обновления, позволяет гибко выбирать стеки технологий для отдельных компонентов. Однако вместе с тем усложняется архитектура приложения, увеличивается количество взаимодействий между сервисами, возрастает риск сбоев, задержек и взаимовлияния при высоких нагрузках.

Актуальность задачи моделирования микросервисных архитектур обусловлена необходимостью предварительного анализа их поведения до реального развертывания. Подобный анализ позволяет оценить устойчивость к отказам, чувствительность к увеличению числа пользователей или интенсивности запросов, выяснить оптимальные параметры для балансировки нагрузки, определить достаточную пропускную способность хранилищ и кешей.

Цель данной работы — разработать модель микросервисной системы и провести серию экспериментов, чтобы исследовать влияние различных параметров (числа пользователей, интенсивности запросов, вероятностей отказа, ограничений по конкурентности операций хранилища) на ключевые показатели системы: число успешных и ошибочных ответов, среднее время обработки запросов, уровень загрузки отдельных компонентов.

Для достижения цели решаются следующие задачи:

- Разработать дискретно-событийную модель микросервисной архитектуры, включающей сервисы клиентов (P), промежуточного сервиса (Q), кеша (T) и постоянного хранилища (S).
- Обеспечить параметризацию модели через конфигурационный файл, позволяя изменять интенсивность потоков запросов, вероятности отказов, задержки и уровень параллелизма.

- Добавить возможность одновременного запуска нескольких процессов P (множество пользователей), чтобы проверить поведение системы под высокой конкурентной нагрузкой.
- Реализовать сбор и визуализацию статистики (число ошибок, время ответа, распределение операций и т.д.), а также дать пользователю инструменты для интерактивной настройки и анализа результатов (через Streamlit-интерфейс).

Практическая значимость работы заключается в том, что результаты моделирования могут использоваться разработчиками, архитекторами и специалистами по эксплуатации систем для принятия инженерных решений по настройке архитектуры, оценке масштабируемости, выявлению потенциальных узких мест и повышению надёжности микросервисных систем.

1. Дискретно-событийный подход к моделированию

При анализе распределённых систем, таких как микросервисы, важным инструментом является дискретно-событийная симуляция (Discrete-Event Simulation, DES). Данный подход предусматривает представление эволюции системы во времени через последовательность событий, происходящих в определённые дискретные моменты. Это позволяет значительно упростить и ускорить расчёты по сравнению с непрерывным моделированием, поскольку моделируются только существенные изменения состояния системы, а не каждый промежуток времени.

Основная идея DES заключается в том, что система находится в состоянии ожидания до тех пор, пока не произойдёт событие (например, поступление нового запроса, завершение обработки или отказ сервиса). Когда событие происходит, внутреннее состояние модели обновляется и планируются будущие события. Время между событиями может быть достаточно большим или переменным, и не требуется рассчитывать поведение системы в моменты, когда ничего не меняется.

Достоинства дискретно-событийного подхода:

- Простота описания логики: Внимание сосредоточено на ключевых событиях — поступлении запроса, завершении операции, отказе компонента.
- Гибкость: Можно моделировать различные сценарии, изменяя список событий и правила их генерации, планирования и обработки.
- Эффективность вычислений: Нет необходимости "пошагово" двигать время с малым шагом. Перемещение происходит от одного события к другому, что экономит вычислительные ресурсы.
- Масштабируемость: Позволяет относительно просто увеличивать сложность модели (добавлять новые компоненты, типы событий) без полного пересмотра подхода.

Применительно к задаче моделирования микросервисной архитектуры, использование DES с помощью библиотеки SimPy даёт следующие преимущества:

- Позволяет естественно описать процессы отправки запросов (ServiceP), обработки и маршрутизации (ServiceQ), работы кеша (ServiceT) и постоянного хранилища (ServiceS) в виде взаимодействующих генераторов и событий.
- Упрощает введение вероятности отказов, задержек, ограничений по конкурентности, формируя реалистичные сценарии нагрузки.
- Предоставляет возможность быстро варьировать параметры и сравнивать результаты — например, как система реагирует на увеличение числа одновременно активных пользователей или на повышение вероятности отказов.

Таким образом, дискретно-событийный подход к моделированию является ключевым методологическим решением, обеспечивающим адекватность, гибкость и эффективность создания модели микросервисной системы, а также анализа её поведения под разными условиями.

2. Анализ исходных данных

В данном разделе рассматриваются входные данные (параметры), определяющие поведение и характеристики моделируемой микросервисной системы. Исходные данные влияют на интенсивность нагрузки, вероятность сбоев, задержки в обработке запросов и другие ключевые показатели системы. Все параметры, как правило, задаются в конфигурационном файле (config.yaml) и/или устанавливаются через интерфейс визуализации, что обеспечивает гибкую настройку и воспроизводимость экспериментов.

2.1 Параметры нагрузки (P)

Сервис P имеет следующие параметры:

- Интенсивность поступления запросов

  Определяется через параметры `arrival_process` (например, "fixed_interval" или "poisson") и `mean_interarrival`. При "fixed_interval" запросы поступают через равные промежутки времени, при "poisson" интервалы экспоненциально распределены. Чем меньше средний интервал, тем выше интенсивность нагрузки на систему.

- Вероятность чтения (read_probability)

  Определяет долю запросов на чтение по сравнению с запросами на запись. При read_probability=0.5 в среднем половина запросов — чтения, половина — записи. Увеличение доли чтений может снизить нагрузку на операции записи, но в случае недоступности кеша (T) может повысить нагрузку на S.

- Количество запросов (num_requests) и число пользователей (num_users)

  Задают масштаб эксперимента. num_requests определяет, сколько запросов сгенерирует один пользователь, num_users — сколько таких

пользователей действует параллельно. Увеличение num_users приводит к одновременному поступлению множества запросов, потенциально создавая конкуренцию за ресурсы и рост очередей.

2.2 Параметры промежуточного сервиса (Q)

Время ожидания (response_timeout) определяет, как долго Q будет ждать ответа от T или S, прежде чем вернуть ошибку (timeout). Слишком маленький таймаут может привести к частым отказам, если S или T отвечают медленно. Слишком большой — к длительному ожиданию и росту среднего времени ответа.

2.3 Параметры кеша (T)

Вероятности отказа при чтении и записи (read_failure_probability, write_failure_probability) - чем выше эти вероятности, тем чаще T не сможет выполнить операцию. Отказы T приводят к большей нагрузке на S (при чтениях) и увеличению числа ошибок при записях. Понимание влияния параметров отказов T необходимо для оценки надежности кеша и его вклада в общую производительность.

2.4 Параметры постоянного хранилища (S)

Сервис S имеет следующие параметры:

- Вероятности отказа (read_failure_probability, write_failure_probability) Определяют, как часто S не может выполнить операцию. Рост вероятностей отказа в S повышает число ошибок и вынуждает P дольше ждать или получать неуспешный ответ, снижая качество обслуживания.
- Максимальное время записи и чтения (max_write_time, max_read_time) Определяют задержки при выполнении операций в S. Большие задержки напрямую влияют на среднее время ответа. При большой интенсивности запросов систематическое увеличение времени операций S может

привести к переполнению очереди и росту ошибок (в связи с таймаутами Q).

- Ограничение по конкурентности (concurrency_limit)

  Определяет максимальное число операций, которые S может обрабатывать одновременно. Небольшое значение увеличивает вероятность образования очереди запросов к S, что приводит к задержкам и повышенному количеству ошибок при высокой нагрузке. Напротив, слишком большое значение может быть неэффективно с точки зрения ресурсов.

2.5 Существующие подходы к моделированию микросервисной системы

При исследовании микросервисных архитектур существует несколько подходов к моделированию, каждый из которых имеет свои преимущества и ограничения. Основная цель такого моделирования — оценить производительность, устойчивость к сбоям, масштабируемость, а также выявить потенциальные узкие места системы до её развёртывания или изменений в инфраструктуре. Рассмотрим наиболее распространённые методы:

1. Аналитические модели и математические оценки

Марковские процессы и сетевые модели: Использование простых вероятностных моделей, описывающих систему в терминах состояний и переходов, позволяет получить аналитические формулы для времени ответа, вероятности отказа и других метрик. Однако такой подход требует существенных упрощений, абстракций и редко учитывает сложные зависимости между микросервисами.

Сетевые модели массового обслуживания (queueing theory): Теория очередей применима к системам с очередями запросов и ограниченной пропускной способностью. Можно приблизительно оценить среднее время ожидания и вероятность перегрузки. Тем не менее, построение точных аналитических моделей для сложных микросервисных систем затруднительно,

особенно если нужно учесть сложные зависимости, отказы и динамические изменения.

2. Имитационное моделирование (Simulation)

Дискретно-событийное моделирование (DES): Наиболее гибкий и широко применяемый подход. Позволяет детально смоделировать каждый микросервис как процесс, с заданной логикой обработки запросов, вероятностями отказов и задержками. Инструменты вроде SimPy дают возможность быстро и относительно просто варьировать параметры, добавлять события и анализировать реакцию системы на различные сценарии.

Агентное моделирование: Похожий на DES подход, где каждая сущность (микросервис, пользователь) рассматривается как агент с собственным поведением. Это может быть полезно при очень сложной логике взаимодействия, но реализация часто сложнее, чем при использовании DES.

3. Статистический анализ и трассировка реальных систем

Анализ логов и трассировок продакшн-систем: Если система уже существует, можно собирать данные о запросах, задержках, ошибках и анализировать их статистически. На основе реальных данных можно построить эмпирические модели. Однако этот подход не всегда применим на ранних этапах или при существенных изменениях архитектуры, когда нет исторических данных.

4. Модели на основе сетей (Network Models)

В моделях на основе сетей система видеоконференций представляется как граф, где узлами являются участники, а ребра — потоки данных между ними. Этот подход позволяет анализировать маршруты передачи видеопотоков и оптимизировать распределение ресурсов в сети, особенно в многозадачных сценариях с несколькими участниками.

- Преимущества: эффективность для анализа маршрутов и оптимизации распределения потоков данных между участниками.
- Недостатки: ограниченная возможность учета индивидуальных предпочтений пользователей и их поведения в реальном времени.

Таким образом, для комплексного исследования микросервисных архитектур DES-симуляция является одним из лучших вариантов, сочетающим достаточную детализацию, гибкость и управляемую сложность, а при необходимости результаты можно подкрепить эмпирическими данными или аналитическими оценками.

3. Дискретно-событийная модель распространения

Данный раздел посвящён описанию дискретно-событийной модели, используемой для моделирования распространения запросов и реакций микросервисной архитектуры. Под "распространением" здесь понимается процесс прохождения запросов от клиентов (P) через промежуточный сервис (Q) к кешу (T) и хранилищу (S), а также обратный поток ответов к пользователям. Дискретно-событийный подход обеспечивает удобный механизм для пошагового отслеживания событий: поступления запросов, завер

3.1 Основные процессы и события

В модели выделяются несколько ключевых процессов, реализованных в виде генераторов SimPy:

3.1.1 Процесс клиента (P)

Генерирует запросы (чтение или запись) с заданной интенсивностью и вероятностью. Каждый запрос представляет событие поступления нового задания в систему. При этом P может действовать как один пользователь, последовательно отправляющий запросы, или же моделируются несколько независимых пользователей (несколько экземпляров процесса P), действующих параллельно, что повышает конкурентность.

3.1.2 Процесс промежуточного сервиса (Q)

Обрабатывает запросы, поступающие от P. Q инициирует операции чтения или записи в T и S. Основными событиями для Q являются: поступление запроса от P, завершение операции записи в T, завершение операции записи в S, завершение операции чтения в T, завершение операции чтения в S, а также события таймаутов или сбоев. Q выступает как координатор, управляющий прохождением запроса через систему.

3.1.3 Процессы кеша (T) и хранилища (S)

Каждый из них моделируется как ресурс или набор операций с задержками и вероятностями отказов. Основными событиями являются: запрос

на чтение/запись, завершение операции, либо возникновение ошибки. Для S дополнительными событиями являются блокировки при превышении concurrency_limit и образование очереди.

3.2 Моделирование нагрузки на сеть

Нагрузка моделируется за счёт поступления запросов от одного или нескольких пользователей (процессов P):

- Интенсивность потока запросов: Определяется параметрами arrival_process и mean_interarrival. При "fixed_interval" запросы генерируются через равные промежутки времени, при "poisson" — интервалы экспоненциально распределены, имитируя пуассоновский поток.
- Число пользователей (num_users): Несколько процессов P запускаются параллельно, каждый генерирует свои запросы. Это повышает вероятность, что в один и тот же момент будет обрабатываться множество запросов, создавая условия для образования очередей в S и возникновения таймаутов в Q.
- Вероятность чтения (read_probability): Определяет долю запросов на чтение относительно запросов на запись. Если большинство запросов — чтения, и кеш T работает без сбоев, нагрузка на S может быть снижена. Однако сбои или неудачи в T приводят к повышенной нагрузке на S.
- Параметры отказов и задержек (T и S): Задержки в S (max_write_time, max_read_time) и вероятность отказов повышают среднее время ответа и число ошибок. Чем выше интенсивность запросов и дольше операции, тем выше вероятность образования очереди и перегрузки.

3.3 Входные и выходные данные модели

3.3.1 Входные данные модели

- Параметры нагрузки P: Тип процесса (arrival_process), средний интервал между запросами (mean_interarrival), read_probability, num_requests и num_users.
- Параметры Q: Время ожидания (response_timeout), определяющее таймаут при ожидании ответа от T или S.
- Параметры T: Вероятности отказа при чтении/записи и отсутствие задержки (или минимальная задержка) для операций.
- Параметры S: Вероятности отказа, максимальное время выполнения записи/чтения, concurrency_limit (ограничение по параллельным операциям).

3.3.2 Выходные данные модели

- Статистика о работе системы: Число успешно обработанных запросов, число ошибок, среднее время ответа, распределения времен ответа.
- Детали запросов: Для каждого запроса можно собрать информацию о типе (чтение/запись), результате (OK или ERROR), времени начала и окончания, продолжительности операции.
- Нагрузочные характеристики: Можно извлечь информацию о степени загрузки S (например, количество одновременных запросов), частоте возникновения таймаутов и отказов.

4. Реализация модели нагрузки на сервер с микросервисной архитектурой

В данном разделе рассматриваются практические аспекты реализации дискретно-событийной модели для исследования поведения микросервисной системы под различной нагрузкой. Описывается выбор инструментов и программной среды, а также приводится план экспериментальных сценариев, позволяющих оценить эффективность и устойчивость системы при изменении ключевых параметров.

4.1 Выбор среды моделирования

Для реализации модели выбрана экосистема языка Python, так как он обеспечивает удобство, гибкость и обширную библиотечную поддержку, включая средства для статистического анализа, визуализации и имитационного моделирования.

- Библиотека для дискретно-событийной симуляции: SimPy
  SimPy — это библиотека на Python для дискретно-событийного моделирования, которая позволяет описывать процессы в виде генераторов, использовать ресурсы и события. Она предоставляет удобный, интуитивно понятный синтаксис для задания логики работы микросервисов P, Q, S и T.
- Средства конфигурирования: YAML
  Настройки системы (интенсивность потока, вероятности отказов, времена отклика, количество пользователей) задаются в YAML-файле (config.yaml), что упрощает эксперименты без изменения кода.
- Streamlit — используется для создания простого веб-интерфейса визуализации результатов. Позволяет интерактивно менять параметры и мгновенно наблюдать за эффектами.
- Matplotlib, Seaborn — для построения графиков и наглядного представления результатов моделирования (распределение времен ответа, число ошибок, соотношения типов запросов).

4.2 Эксперимент

Для проверки и анализа работы модели планируется провести серию экспериментов, в каждом из которых параметры будут варьироваться, а результаты — сравниваться.

1. Эксперимент №1

Таб. 1 – параметры эксперимента №1

Параметр	Значение	Описание
P (пользователи)		Параметры генерации запросов
arrival_process	fixed_interval	Тип процесса поступления запросов ("fixed_interval" или "poisson")
mean_interarrival	0.2	Средний интервал между запросами
read_probability	0.5	Вероятность, что запрос будет чтением
num_requests	50	Общее количество запросов для одного пользователя
num_users	1	Число параллельных пользователей (процессов P)
Q (промежуточный сервис)		Параметры ожидания ответа от нижестоящих сервисов
response_timeout	3.0	Максимальное время ожидания ответа от T или S
T (кеш)		Параметры отказов кеша
read_failure_probability	0	Вероятность отказа при чтении в T

Параметр	Значение	Описание
write_failure_probability	0	Вероятность отказа при записи в T
S (постоянное хранилище)		Параметры отказов, задержек и конкурентности S
read_failure_probability	0	Вероятность отказа при чтении в S
write_failure_probability	0	Вероятность отказа при записи в S
max_write_time	1.0	Максимальное время на операцию записи в S
max_read_time	2.0	Максимальное время на операцию чтения в S
concurrency_limit	3	Максимальное число одновременных операций в S

Результаты:

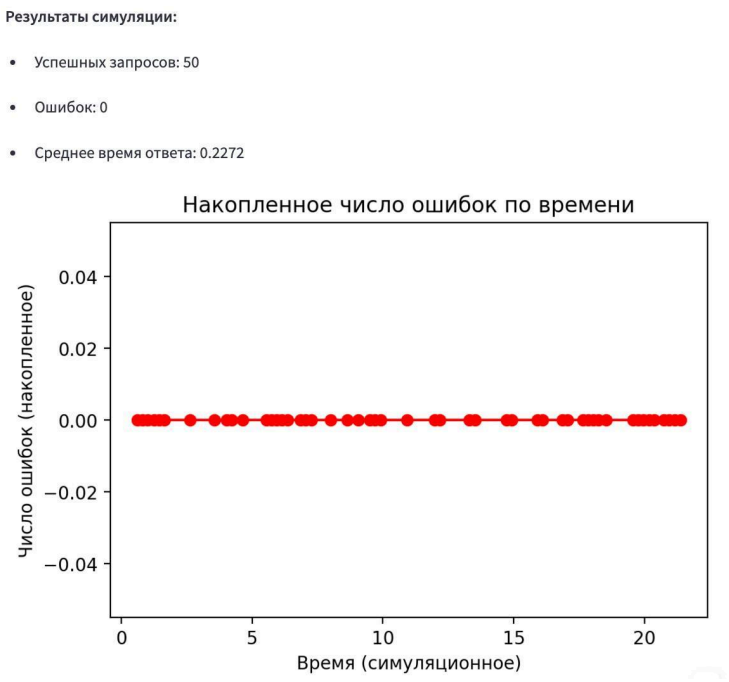

Рис. 1 – Результаты эксперимента №1. Точечный график

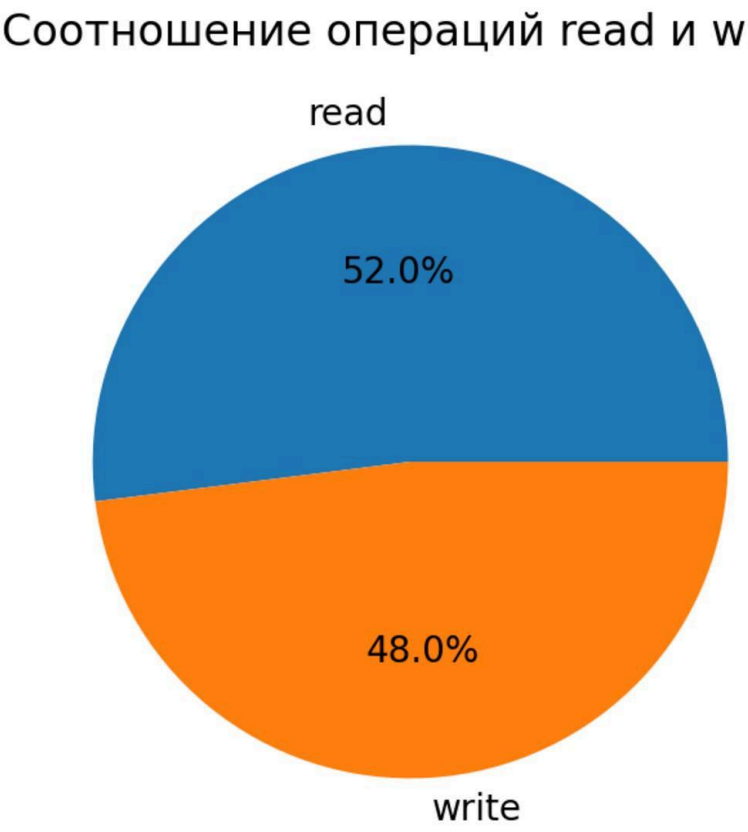

Рис. 2 – Результаты эксперимента №1. График отношения чтения и записи

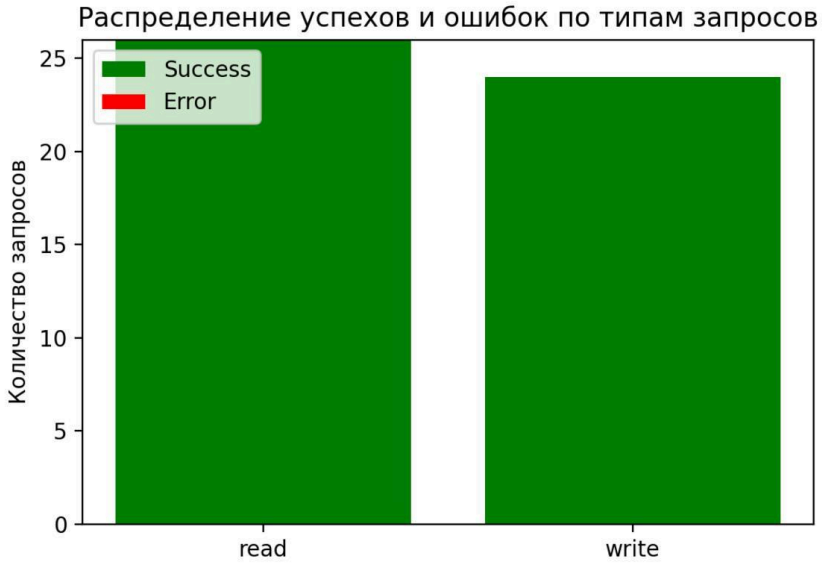

Рис. 3 – Результаты эксперимента №1. Распределение успехов и ошибок

2. Эксперимент №2

Таб. 2 – параметры эксперимента №2

Параметр	Значение	Описание
P (пользователи)		Параметры генерации запросов
arrival_process	fixed_interval	Тип процесса поступления запросов ("fixed_interval" или "poisson")
mean_interarrival	0.2	Средний интервал между запросами
read_probability	0.5	Вероятность, что запрос будет чтением
num_requests	50	Общее количество запросов для одного пользователя
num_users	1	Число параллельных пользователей (процессов P)
Q (промежуточный сервис)		Параметры ожидания ответа от нижестоящих сервисов
response_timeout	3.0	Максимальное время ожидания ответа от T или S
T (кеш)		Параметры отказов кеша
read_failure_probability	1	Вероятность отказа при чтении в T
write_failure_probability	0	Вероятность отказа при записи в T
S (постоянное хранилище)		Параметры отказов, задержек и конкурентности S
read_failure_probability	0.5	Вероятность отказа при

Параметр	Значение	Описание
		чтении в S
write_failure_probability	0	Вероятность отказа при записи в S
max_write_time	1.0	Максимальное время на операцию записи в S
max_read_time	2.0	Максимальное время на операцию чтения в S
concurrency_limit	3	Максимальное число одновременных операций в S

Результаты:

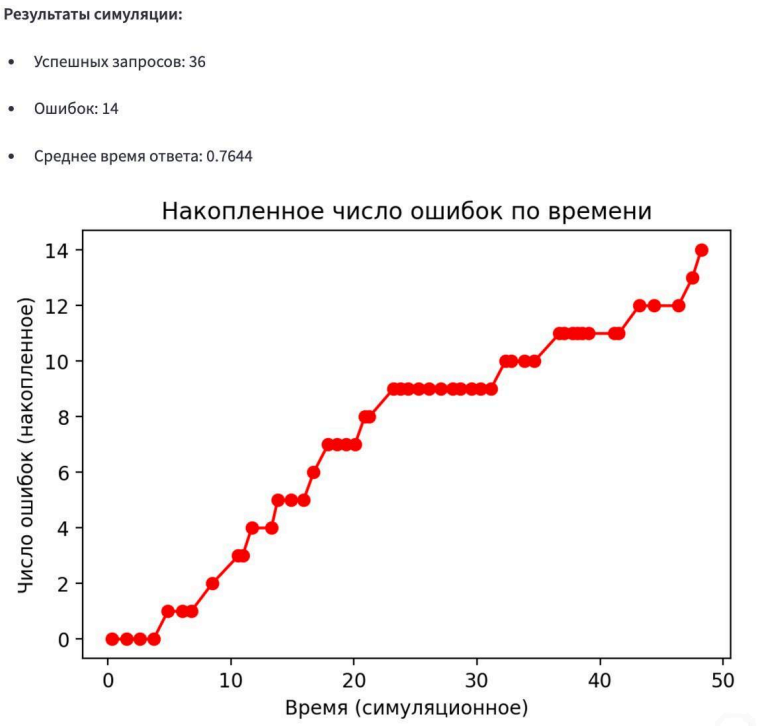

Рис. 4 – Результаты эксперимента №2. Точечный график

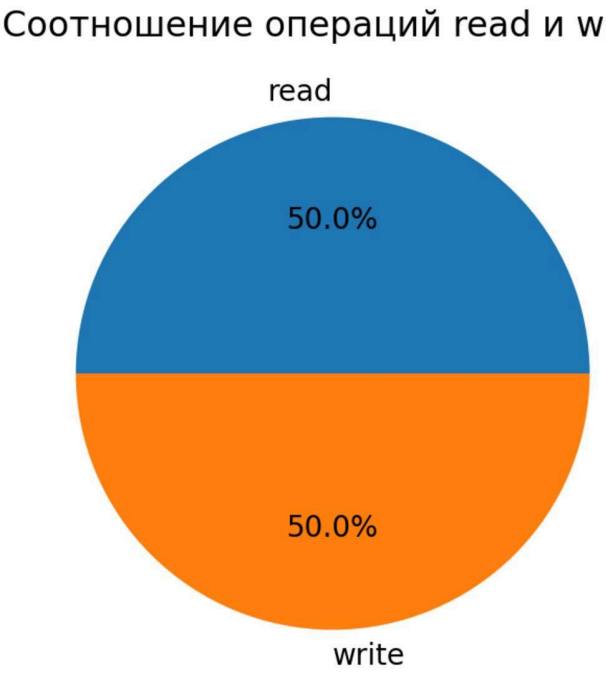

Рис. 5 – Результаты эксперимента №2. График отношения чтения и записи

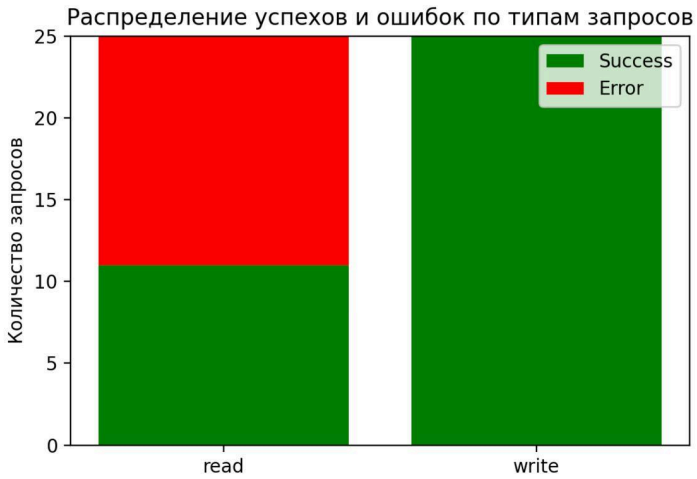

Рис. 6 – Результаты эксперимента №2. Распределение успехов и ошибок

3. Эксперимент №3

Таб. 3 – параметры эксперимента №3

Параметр	**Значение**	**Описание**
P (пользователи)		Параметры генерации запросов
arrival_process	fixed_interval	Тип процесса поступления запросов

Параметр	Значение	Описание
		("fixed_interval" или "poisson")
mean_interarrival	0.2	Средний интервал между запросами
read_probability	0.5	Вероятность, что запрос будет чтением
num_requests	50	Общее количество запросов для одного пользователя
num_users	1	Число параллельных пользователей (процессов P)
Q (промежуточный сервис)		Параметры ожидания ответа от нижестоящих сервисов
response_timeout	3.0	Максимальное время ожидания ответа от T или S
T (кеш)		Параметры отказов кеша
read_failure_probability	0	Вероятность отказа при чтении в T
write_failure_probability	0.5	Вероятность отказа при записи в T
S (постоянное хранилище)		Параметры отказов, задержек и конкурентности S
read_failure_probability	0	Вероятность отказа при чтении в S
write_failure_probability	0.5	Вероятность отказа при записи в S
max_write_time	1.0	Максимальное время на операцию записи в S
max_read_time	2.0	Максимальное время на

Параметр	Значение	Описание
		операцию чтения в S
concurrency_limit	3	Максимальное число одновременных операций в S

Результаты:

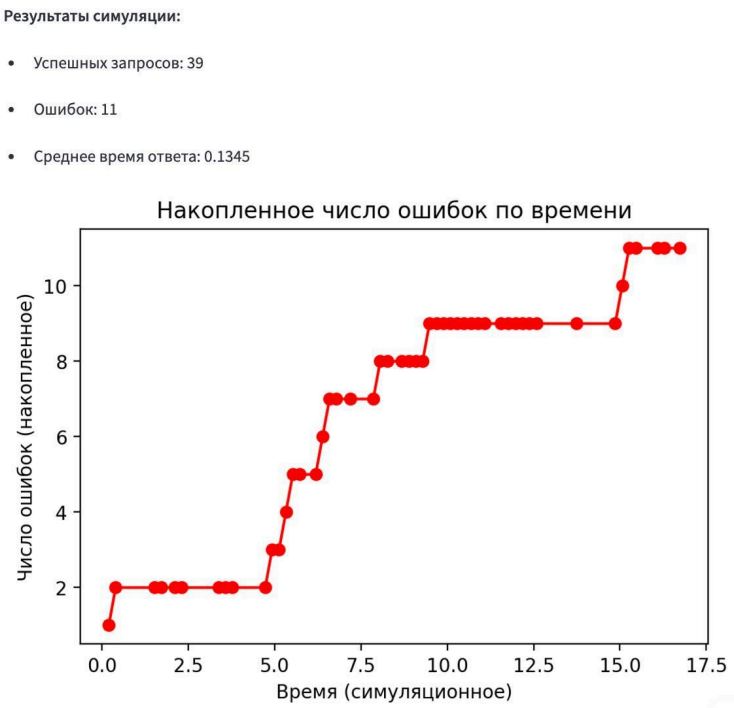

Рис. 7 – Результаты эксперимента №3. Точечный график

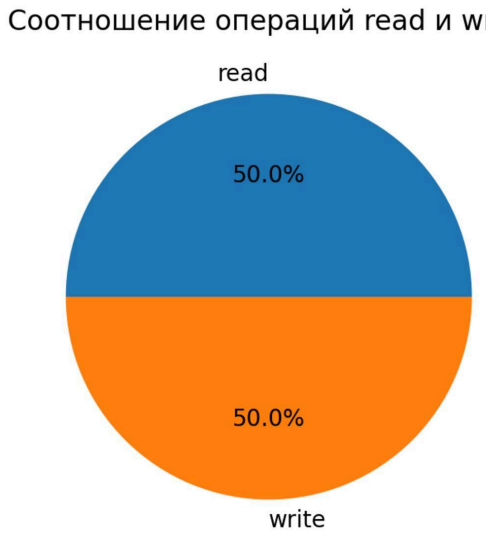

Рис. 8 – Результаты эксперимента №3. График отношения чтения и записи

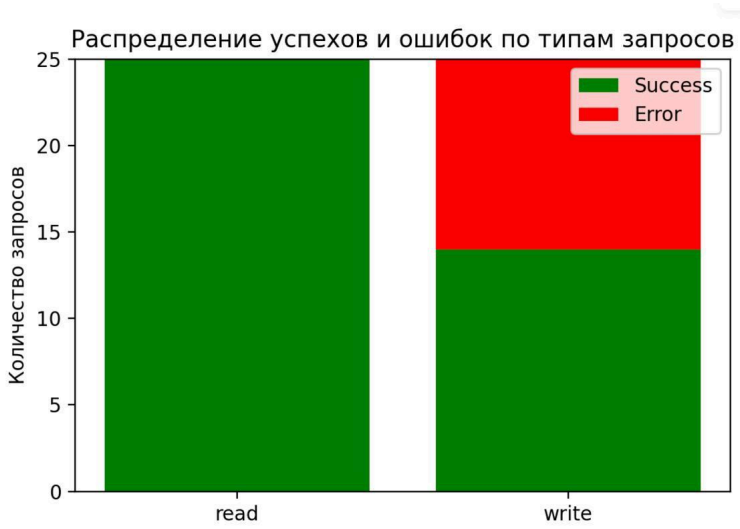

Рис. 9 – Результаты эксперимента №3. Распределение успехов и ошибок

4. Эксперимент №4

Таб. 4 – параметры эксперимента №4

Параметр	**Значение**	**Описание**
P (пользователи)		Параметры генерации запросов
arrival_process	poisson	Тип процесса поступления запросов ("fixed_interval" или "poisson")
mean_interarrival	0.2	Средний интервал между запросами
read_probability	0.8	Вероятность, что запрос будет чтением
num_requests	50	Общее количество запросов для одного пользователя
num_users	10	Число параллельных пользователей (процессов P)
Q (промежуточный		Параметры ожидания

Параметр	Значение	Описание
сервис)		ответа от нижестоящих сервисов
response_timeout	3.0	Максимальное время ожидания ответа от T или S
T (кеш)		Параметры отказов кеша
read_failure_probability	0.05	Вероятность отказа при чтении в T
write_failure_probability	0.05	Вероятность отказа при записи в T
S (постоянное хранилище)		Параметры отказов, задержек и конкурентности S
read_failure_probability	0.05	Вероятность отказа при чтении в S
write_failure_probability	0.05	Вероятность отказа при записи в S
max_write_time	1.0	Максимальное время на операцию записи в S
max_read_time	1.0	Максимальное время на операцию чтения в S
concurrency_limit	10	Максимальное число одновременных операций в S

Результаты:

Результаты симуляции:

- Успешных запросов: 489
- Ошибок: 11
- Среднее время ответа: 0.2884

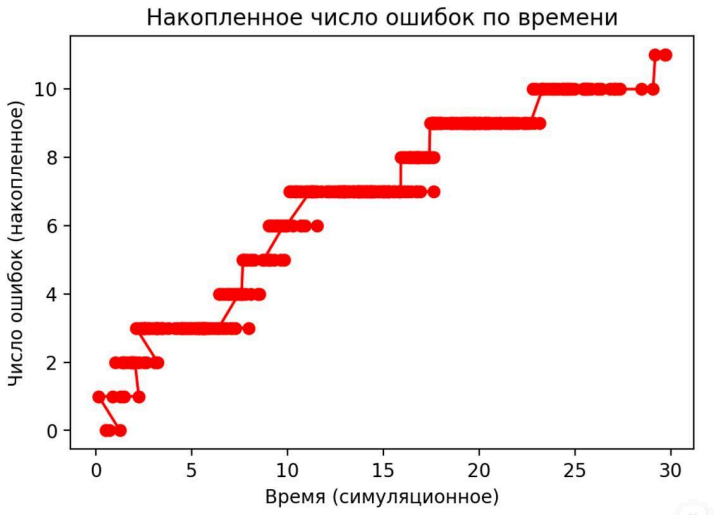

Рис. 10 – Результаты эксперимента №4. Точечный график

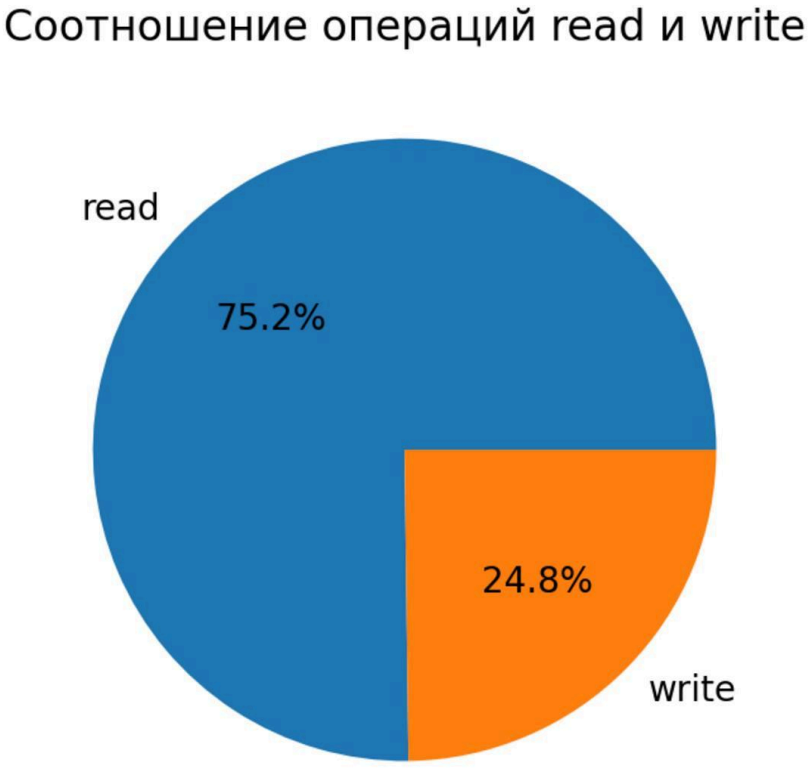

Рис. 11 – Результаты эксперимента №4. График отношения чтения и записи

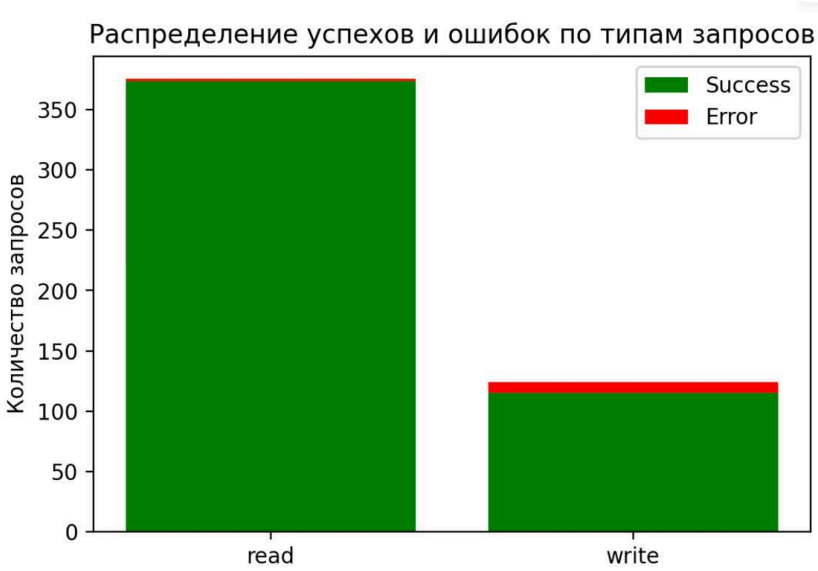

Рис. 12 – Результаты эксперимента №4. Распределение успехов и ошибок

5. Эксперимент №5

Таб. 5 – параметры эксперимента №5

Параметр	**Значение**	**Описание**
P (пользователи)		Параметры генерации запросов
arrival_process	poisson	Тип процесса поступления запросов ("fixed_interval" или "poisson")
mean_interarrival	0.2	Средний интервал между запросами
read_probability	0.8	Вероятность, что запрос будет чтением
num_requests	50	Общее количество запросов для одного пользователя
num_users	10	Число параллельных пользователей (процессов P)
Q (промежуточный сервис)		Параметры ожидания ответа от нижестоящих сервисов

Параметр	Значение	Описание
response_timeout	3.0	Максимальное время ожидания ответа от T или S
T (кеш)		Параметры отказов кеша
read_failure_probability	1.0	Вероятность отказа при чтении в T
write_failure_probability	0.05	Вероятность отказа при записи в T
S (постоянное хранилище)		Параметры отказов, задержек и конкурентности S
read_failure_probability	0.0	Вероятность отказа при чтении в S
write_failure_probability	0.05	Вероятность отказа при записи в S
max_write_time	2.5	Максимальное время на операцию записи в S
max_read_time	2.5	Максимальное время на операцию чтения в S
concurrency_limit	5	Максимальное число одновременных операций в S

Результаты:

Результаты симуляции:

- Успешных запросов: 347
- Ошибок: 153
- Среднее время ответа: 2.2791

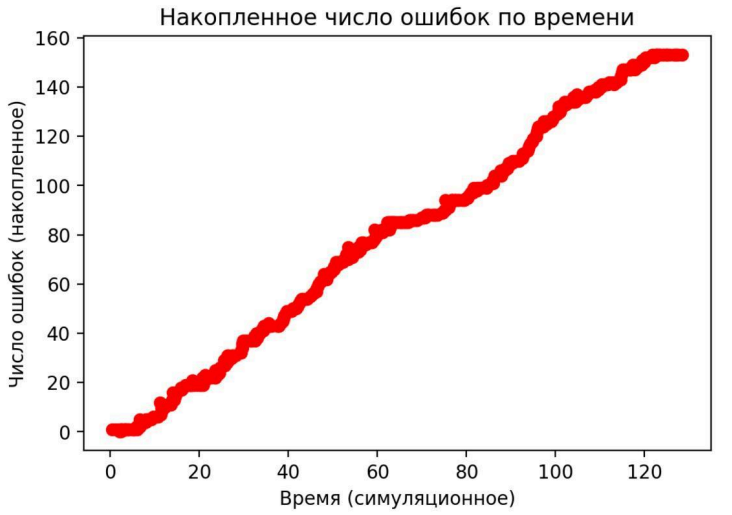

Рис. 13 – Результаты эксперимента №5. Точечный график

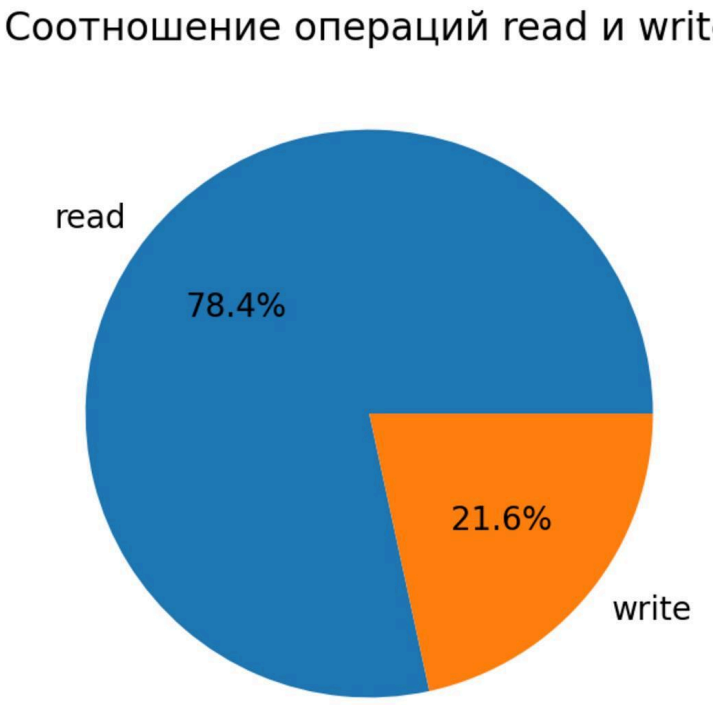

Рис. 14 – Результаты эксперимента №5. График отношения чтения и записи

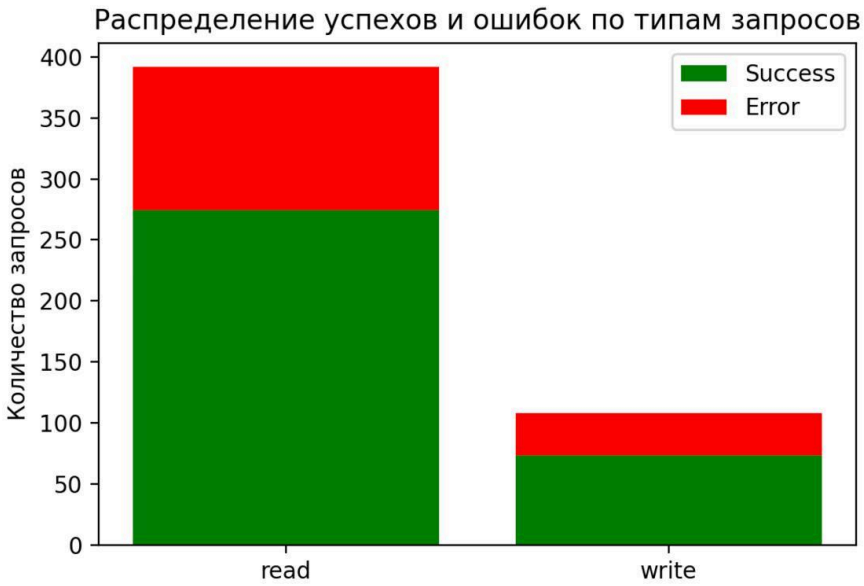

Рис. 15 – Результаты эксперимента №5. Распределение успехов и ошибок

5. Тестирование и анализ разработанной системы

Для выполнения экспериментов использовалась следующая конфигурация тестовой машины:

— Операционная система: Windows 10, 64-bit

— Процессор: Intel(R) Core(TM) i7 CPU 930 @ 2.80GHz

— Оперативная память: 14 Гб

— Среда разработки: Python 3.12.8 с библиотеками SimPy, Matplotlib, NumPy и Pandas

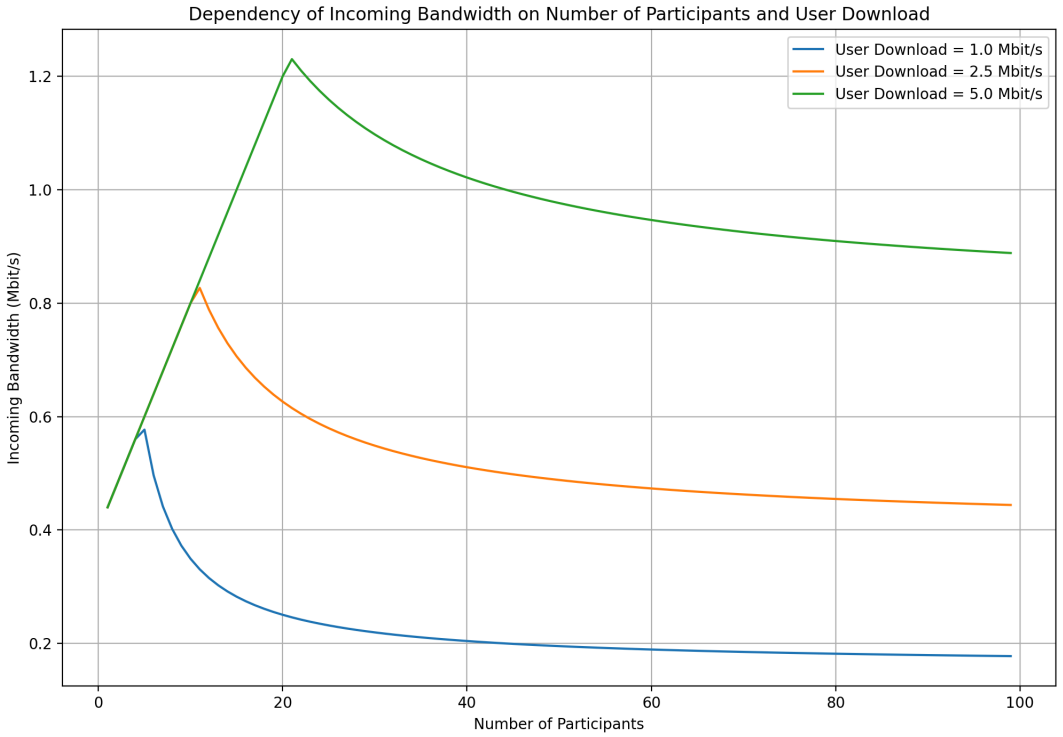

Рис. 16 – Зависимость входящей полосы пропускания от количества пользователей

График (рис. 16) зависимости исходящей полосы пропускания от числа пользователей показывает, что пропускная способность растет линейно с увеличением числа пользователей.

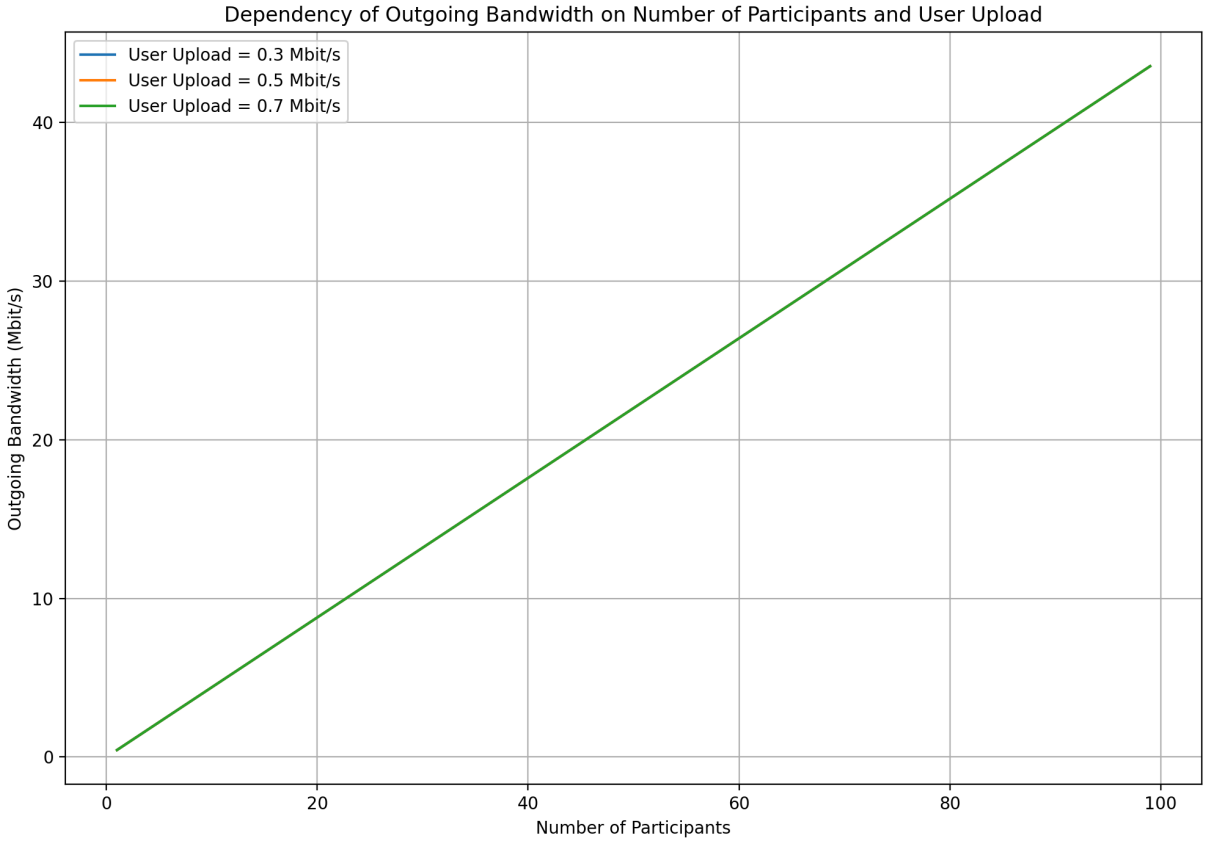

Рис. 17 – Зависимость исходящей полосы пропускания от количества пользователей

Заключение

В процессе выполнения работы создана дискретно-событийная модель микросервисной архитектуры, охватывающая взаимодействия между клиентами (P), промежуточным сервисом (Q), кешем (T) и постоянным хранилищем (S). Использование библиотеки SimPy позволило гибко описать поведение системы, учесть вероятность отказов, задержки операций, лимиты конкурентности и различные сценарии нагрузки. Благодаря настраиваемым параметрам интенсивности запросов, вероятностей сбоев, времен отклика и количеству параллельных пользователей удалось провести серию экспериментов, выявить закономерности и определить критические условия работы системы.

Проведённые эксперименты показали, что рост числа пользователей и интенсивности запросов повышает нагрузку на хранилище S, способствуя образованию очередей и увеличению времени отклика. Введение вероятностей отказов в компонентах T или S приводит к росту числа ошибок и снижению стабильности системы. Изменение параметров задержек и конкурентности в S серьёзно сказывается на пропускной способности и качестве обслуживания. Таким образом, поведение системы чувствительно к целому ряду параметров, и их оптимальный подбор позволяет обеспечить баланс между пропускной способностью, временем ответа и устойчивостью к сбоям.

Использование конфигурационного файла для задания параметров и создание удобного графического интерфейса на базе Streamlit упростили настройку экспериментов и анализ полученных данных. Визуализация результатов в виде графиков, гистограмм и диаграмм облегчила интерпретацию наблюдаемых эффектов.

Предложенная модель и полученные результаты представляют практическую ценность для предварительной оценки поведения микросервисных систем до их фактического развертывания. Итоги работы могут быть применены при принятии инженерных решений об оптимизации

архитектуры, выборе стратегий кеширования и балансировки нагрузки, а также при планировании мер по повышению надёжности и отказоустойчивости. В дальнейшем возможны расширения модели за счёт более сложных стратегий маршрутизации, динамического масштабирования ресурсов и учета сетевых параметров, что позволит ещё точнее оценивать границы производительности и надёжности микросервисных решений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python Software Foundation. Python 3.11 Documentation. https://docs.python.org/3/.
2. Matplotlib Developers. Matplotlib: Visualization with Python. https://matplotlib.org/stable/contents.html.
3. NumPy Developers. NumPy User Guide. https://numpy.org/doc/stable/.
4. Pandas Developers. Pandas Documentation. https://pandas.pydata.org/docs/.
5. SimPy Documentation. SimPy: Discrete-Event Simulation for Python. https://simpy.readthedocs.io.
6. Banks, J., Carson, J. S., & Nelson, B. L. Discrete-Event System Simulation. Pearson Education, 2010.
7. Fishman, G. S. Discrete-Event Simulation: Modeling, Programming, and Analysis. Springer, 2001.
8. Pidd, M. Tools for Thinking: Modelling in Management Science. Wiley, 2009.
9. Sterman, J. D. Business Dynamics: Systems Thinking and Modeling for a Complex World. McGraw-Hill, 2000.
10. Hitesh Bhatia, Abhishek Bhardwaj, Rajesh K. Gupta VoIP Traffic Analysis for Network Planning, 2015.
11. A. G. Gohil, M. V. A. Rao, S. K. Bhatia Video Conferencing and Video Streaming: A Survey, 2013.
12. Hillier, F. S., & Lieberman, G. J. Introduction to Operations Research. McGraw-Hill, 2010.
13. Mahmassani, H. S. Dynamic Network Simulation and Control. Springer, 2013.
14. Y. Zhan, L. Li, Y. Zhang Performance of Video Streaming and Conferencing Systems: A Survey, 2019.