

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра дискретной математики и информационных технологий

**ПРИМЕНЕНИЕ МЕТОДА РАСПРЕДЕЛЕННОГО ХРАНЕНИЯ
ДАННЫХ ДЛЯ ПОВЫШЕНИЯ ИХ ЗАЩИЩЕННОСТИ**

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Захарова Сергея Алексеевича

Научный руководитель
старший преподаватель

П. О. Дмитриев

Заведующий кафедрой
доцент, к. ф.-м. н.

Л. Б. Тяпаев

Саратов 2025

ВВЕДЕНИЕ

Современные технологии хранения данных стремительно развиваются, предлагая пользователям всё более надёжные и эффективные решения для управления информацией. Одним из ключевых направлений в данной области является облачное хранение данных, которое обеспечивает доступ к файлам с любого устройства и высокую отказоустойчивость. Однако с ростом объёмов данных и требований к их безопасности возникает необходимость в оптимизации методов хранения, включая распределение информации между разными облачными сервисами для повышения надёжности.

Методы распределенного хранения данных могут быть применены для повышения безопасности при хранении данных. Распространённым примером является хранение зашифрованного файла по методу **RAID 1**. Данный подход имеет свои преимущества и недостатки. Некоторые из этих недостатков могут быть упразднены при применении альтернативных решений. В данной работе будет рассмотрен один из возможных способов такого решения при помощи принципа **RAID 5**.

В рамках работы предполагается исследование модели для распределения данных, возможностей её модификации, а также изучение методов распределённого хранения данных при помощи облачных сервисов.

Также в ходе работы требуется провести тестирование разработанного приложения. Необходимо оценить как корректность реализованного функционала, так и его соответствие выдвинутым к системе требованиям.

Исходя из этого, целью работы является обоснование и иллюстрация возможности хранения данных по принципу **RAID 5**, повышающему отказоустойчивость, а также безопасность хранения данных. Для достижения цели были поставлены следующие задачи:

1. Исследовать архитектуру **RAID** и определить методы для повышения конфиденциальности хранимых данных;
2. Разработать пакет программ иллюстрирующих метод хранения;
3. Реализовать взаимодействие с облачными хранилищами;
4. Реализовать веб-приложение использующее облачные сервисы в качестве хранилищ;
5. Провести тестирование отказоустойчивости и корректности восстановления данных при недоступности одного из сервисов.

Структура работы включает в себя восемь глав:

1. Описание предметной области
2. Постановка задачи
3. Теоретические сведения
4. Исследование возможности дополнительной защиты данных с использованием разделенного хранения
5. Разработка решения для обработки данных на облачных сервисах
6. Реализация класса, распределяющего данные среди облачных хранилищ по принципу RAID 5
7. Разработка пользовательского интерфейса
8. Тестирование реализованного решения

При написании работы было использовано 20 источников, приведено 15 рисунков, 2 таблицы. Всего работа предоставлена в виде 60 листов.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе рассматриваются предпосылки и актуальность задачи разработки приложения для облачного хранения данных с использованием принципа RAID 5. Отмечается, что существующие решения, предлагаемые облачными провайдерами, часто недостаточно надёжны и зависят от одного поставщика. Предлагается подход программной эмуляции RAID 5 на уровне прикладной логики, позволяющий повысить отказоустойчивость и информационную безопасность за счёт распределения данных между независимыми облачными сервисами.

Раздел также включает анализ инструментов разработки, где в качестве backend-технологии выбран ASP.NET Core с языком C#, а для frontend — фреймворк Vue 3 на JavaScript. Это обосновано их производительностью и удобством интеграции с API облачных сервисов.

В завершение даётся обзор существующих решений (например, Google Drive, Dropbox, StableBit CloudDrive), выявляются их ограничения (зависимость от одного сервиса, отсутствие кроссплатформенности) и подчёркивается необходимость независимого, распределённого подхода. Упоминается пример сбоя в работе Yandex Cloud в 2024 году как подтверждение практической значимости разработки.

Во втором разделе изложена постановка задачи, решаемой в рамках выпускной квалификационной работы. Основной целью является разработка одностраничного веб-приложения для хранения и восстановления файлов, распределённых по трём облачным сервисам (Яндекс Диск, Dropbox и Google Drive) по принципу RAID 5. Такая организация хранения обеспечивает отказоустойчивость — возможность восстановления данных при недоступности одного из сервисов.

Поставлены конкретные задачи по реализации backend-части на основе ASP.NET Core (язык C#), frontend-интерфейса с использованием JavaScript и Vue.js, а также интеграции с тремя облачными хранилищами. Описаны ключевые функциональные возможности приложения: загрузка и скачивание файлов с учетом разбиения на блоки, восстановление утерянных данных при отказе одного из хранилищ, реализация механизма паритета по технологии RAID 5. Представлена архитектура решения и перечислены этапы его реализации: разработка сервисных классов, контроллеров, а также логики

взаимодействия с облачными API.

В третьем разделе рассматриваются теоретические основы технологии RAID (Redundant Array of Independent Disks) как способа организации отказоустойчивого хранения данных. Излагаются цели и принципы использования RAID-массивов, их роль в обеспечении надёжности, производительности и эффективности хранения информации за счёт чередования и избыточности.

Дается классификация уровней RAID, включая RAID 0, RAID 1, RAID 5, RAID 6 и RAID 10. Для каждого из них описаны принципы работы, преимущества, недостатки, требования к минимальному количеству дисков и области применения. Например, RAID 0 позволяет достичь высокой скорости, но не обеспечивает отказоустойчивости, RAID 1 дублирует данные для надёжности, но требует значительного объема хранилищ, тогда как RAID 5 сочетает избыточность с эффективным использованием пространства, что делает его особенно подходящим для задач, где важен компромисс между надёжностью и затратами.

Рассмотрены два метода разбиения файла на блоки для распределения: полное деление на 6 равных массивов байтов, и последовательное деление до тех пор, пока общее число блоков не станет кратным 6. Для каждого из трёх пар блоков (например, A_i и B_i , C_i и D_i , E_i и F_i) вычисляются соответствующие массивы чётности ($Parityi_1$, $Parityi_2$, $Parityi_3$), которые затем распределяются по облачным хранилищам таким образом, чтобы каждый диск хранил только одну из частей каждой тройки (два блока данных и один блок чётности).

Также приводится иллюстрация схемы распределения данных на три диска, демонстрирующую, как RAID 5 позволяет сохранить все данные даже при отказе одного из хранилищ, восстановив их с использованием паритета. В случае использования трёх облачных сервисов потери по объёму составляют объём одного виртуального "диска" однако достигается повышенная отказоустойчивость и возможность автоматического восстановления.

В четвёртом разделе рассматриваются особенности программной реализации RAID 5 в условиях работы с облачными хранилищами. Поскольку физического RAID-контроллера не используется, разработана собственная модель хранения и восстановления данных с применением побайтовой

операции XOR.

Описан процесс преобразования файла в набор массивов байтов, распределяемых по трём хранилищам: два файла содержат фрагменты исходных данных, третий — массивы чётности. Реализована возможность восстановления утраченных данных при недоступности одного из хранилищ.

Приведены два этапа программной реализации:

- Первый этап — создание консольного приложения, делящего файл на шесть частей, формирующего массивы чётности и сохраняющего три подготовленных файла.
- Второй этап — модификация модели с возможностью задания динамического размера фрагментов данных, что повышает конфиденциальность. Размер изначального файла записывается в конец файла для корректного восстановления.

В пятом разделе предоставляется разработка архитектурного решения и выбор технологий, используемых для построения отказоустойчивой системы облачного хранения данных на основе RAID 5. В этом разделе изложены программные и технические средства, обеспечивающие реализацию ключевых функций приложения: загрузки, хранения, распределения, восстановления и скачивания файлов с использованием нескольких облачных хранилищ.

В качестве серверной технологии был выбран фреймворк ASP.NET Core, предоставляющий платформу для создания веб-приложений. Его использование обусловлено необходимостью реализации вычислительно затратных операций, таких как побитовое вычисление паритета, что выгодно выполнять с помощью компилируемого языка программирования C#. Применение компилятора позволяет выполнять операции XOR эффективно, используя низкоуровневые инструкции, в отличие от интерпретируемых языков, которые требуют дополнительных накладных расходов на проверку типов и вызов методов.

Системная архитектура реализована по принципу трёхуровневого проектирования, включающего:

- Уровень представления — реализован с помощью JavaScript и фреймворка Vue.js; отвечает за взаимодействие с пользователем, отображение данных и передачу запросов к серверу.

- Уровень бизнес-логики — реализует основной функционал распределения данных, вычисления паритета, сборки и восстановления файлов, обрабатывает входящие запросы и управляет процессами хранения.
- Слой данных — содержит модули взаимодействия с облачными сервисами, включая отправку и получение файлов, а также получение метаданных.

Отдельное внимание уделяется настройке инфраструктурных компонентов, необходимых для работы веб-приложения. В частности, применяется Swagger — инструмент для автоматической генерации и тестирования API, позволяющий визуализировать маршруты, параметры и возвращаемые значения. Данный инструмент значительно упрощает процесс отладки и тестирования серверной части. Также подключена и настроена технология CORS, обеспечивающая безопасность кросс-доменных запросов между клиентом и сервером, работающими на разных доменах.

Далее последовательно рассматриваются особенности интеграции каждого из облачных сервисов:

Яндекс Диск

Для подключения к API Яндекс Диска реализован механизм авторизации пользователя через OAuth. Создано собственное приложение в Яндекс ID, позволяющее получать токен доступа и выполнять операции чтения и записи. Применяются HTTP-запросы к REST API Яндекс Диска, обеспечивающие загрузку и скачивание файлов. Используется класс `HttpClient`, а данные передаются в виде массива байт. Приведены примеры кода и логика десериализации JSON-ответов для получения адресов прямой загрузки/скачивания.

Dropbox

Для работы с Dropbox реализовано собственное приложение в консоли разработчика. Выбран режим полного доступа ко всему хранилищу пользователя. На стороне сервера подготовлены методы для отправки POST-запросов с передачей файла в формате байтов и заголовками, содержащими настройки пути, режима перезаписи и поведения при конфликтах имён. Также предусмотрен механизм загрузки и обработки метаданных файлов через HTTP-запросы с сериализацией параметров в JSON.

Google Drive

Для взаимодействия с Google Drive используется официальная библиотека `Google.Apis.Drive.v3`, подключаемая через NuGet в среде Visual Studio. Авторизация пользователя реализована через сервис Google OAuth, с использованием JSON-файла, содержащего `client_id`, `client_secret` и прочие настройки. Подключение сопровождается созданием сервиса `DriveService`, через который выполняются операции поиска, загрузки и скачивания файлов. При загрузке необходимо указать имя файла и ID папки. При скачивании сначала определяется ID нужного файла по имени, затем формируется запрос и файл сохраняется на локальное устройство. Приведены фрагменты кода с пояснением всех этапов.

Каждый облачный сервис был подключён с соблюдением требований безопасности, в том числе через авторизацию, настройку прав доступа и конфигурацию разрешений в приложениях. Все запросы выполняются в асинхронном режиме, что позволяет не блокировать основной поток выполнения и поддерживать отзывчивость интерфейса при передаче больших объёмов данных.

Таким образом, третий раздел содержит детальное описание реализации серверной части приложения, включая выбор технологий, архитектурных подходов, средств авторизации, примеры работы с API облачных сервисов и технические детали передачи и обработки файлов. Этот этап стал основой для последующей реализации логики распределения данных и взаимодействия с пользователем.

В шестом разделе предоставляется практическая реализация ключевого модуля всей системы — класса, осуществляющего распределение и восстановление данных между облачными хранилищами в соответствии с принципами RAID 5. Описывается структура этого класса, перечень реализованных методов, их назначение и алгоритмическая логика.

Реализованный модуль инкапсулирует функциональность работы с тремя подключёнными облачными сервисами (Яндекс Диск, Dropbox, Google Drive) и позволяет выполнять следующие операции:

- загрузку файла с распределением данных по всем облакам;
- скачивание полного файла при наличии доступа ко всем хранилищам;
- восстановление файла при недоступности одного из облаков (любого из трёх);

- расчёт массива чётности (паритета) по побитовой операции XOR.

Для реализации вычислений создан отдельный метод `SolveParity`, принимающий два массива байтов и возвращающий массив чётности. Этот метод используется при каждой операции распределения данных.

В подразделе, посвящённом загрузке файла с распределением, подробно описан процесс:

- Файл считывается на сервер и представляется в виде массива байтов.
- Массив делится на 6 частей с возможностью указать динамический размер частей, формируются три пары.
- Для каждой пары вычисляется чётность.
- Каждая из трёх частей (две части + чётность) сохраняется во временный файл.
- Через соответствующий сервис каждая часть отправляется на своё облачное хранилище.
- После успешной загрузки временные файлы удаляются с сервера.

Следующий подраздел описывает алгоритм загрузки и восстановления файла. В обычной ситуации, если все три облачных сервиса доступны, части файла загружаются и объединяются в исходный массив байтов. Однако, в случае недоступности одного из хранилищ, применяется механизм восстановления утраченных данных. Используются хранимые данные и массив чётности, чтобы по операции XOR восстановить недостающий блок.

Алгоритмы восстановления используют асинхронные вызовы облачных сервисов, временно сохраняют загруженные части на сервере, а затем объединяют их в исходный файл.

Таким образом, шестой раздел демонстрирует не только реализацию алгоритмов RAID 5 в распределённой среде, но и интеграцию с облачными API, обработку байтовых данных и обеспечение отказоустойчивости на уровне логики приложения.

В седьмом разделе представляется разработка пользовательского интерфейса одностраничного веб-приложения, предназначенного для взаимодействия пользователя с функциональностью распределённого облачного хранилища, реализующего принципы RAID 5. Основная задача интерфейса — предоставить пользователю удобные средства для загрузки и скачивания файлов, выбора способа восстановления данных и визуального контроля над

выполнением операций.

Интерфейс был реализован с использованием фреймворка Vue.js, который обеспечивает интеграцию с API серверной части. Также в проекте использовались HTML и CSS для создания визуального оформления и базовой структуры страниц.

В разделе описана структура пользовательского интерфейса, состоящая из следующих ключевых компонентов:

- форма загрузки файла, реализующая отправку данных на сервер для дальнейшего распределения по облачным хранилищам;
- интерактивный выпадающий список с выбором доступных на сервере файлов для скачивания;
- радиокнопки, позволяющие пользователю выбрать один из четырёх режимов скачивания файла: с использованием всех дисков, без Яндекс Диска, без Dropbox или без Google Drive;
- кнопки управления, отвечающие за запуск операций загрузки, скачивания и обновления списка доступных файлов;
- поле с отображением выбранного пользователем файла и результатов операций.

Интерфейс построен с использованием одностраничной архитектуры, что позволяет выполнять большинство операций без перезагрузки страницы, обеспечивая высокую отзывчивость и удобство использования.

Для отображения динамических данных используются директивы `v-if` и `v-for`:

- `v-if` позволяет отображать или скрывать элементы интерфейса в зависимости от состояния приложения (например, отображение списка файлов только после нажатия кнопки);
- `v-for` реализует итерацию по массиву файлов, полученных с сервера, создавая визуальный список элементов, доступных для скачивания.

Также описана подробная реализация всех элементов представленного интерфейса. На иллюстрациях, приведённых в разделе, продемонстрированы основные компоненты интерфейса: форма выбора файла, список опций восстановления, отображение выбранного файла и статусные сообщения. Интерфейс оформлен с использованием базовых CSS-стилей, позволяющих реализовать корректное отображение элементов на разных устройствах.

Раздел демонстрирует, что разработанный интерфейс интегрирован с серверной частью приложения, позволяя в полном объёме использовать возможности распределённого хранения данных по технологии RAID 5. Таким образом, пользователь получает доступ к инструменту, обеспечивающему надёжное и безопасное хранение данных с возможностью восстановления в случае сбоя одного из облачных сервисов.

В восьмом разделе проводится тестирование разработанного приложения на соответствие требованиям отказоустойчивости и корректности функционирования. Проверяются основные сценарии работы: загрузка файлов с распределением по принципу RAID 5, восстановление при недоступности одного из облачных хранилищ (Яндекс.Диск, Dropbox, Google Drive), а также сборка исходного файла.

Приложение успешно проходит тесты на целостность данных, демонстрируя корректное восстановление информации при потере одного из фрагментов.

Результаты тестирования подтверждают, что приложение соответствует поставленным задачам и может использоваться как надёжное распределённое решение для хранения данных.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была исследована модель распределенного хранения данных. Полученная модель была успешно применена при реализации системы распределённого хранения данных на основе принципа RAID 5. Основной целью разработки стало повышение отказоустойчивости и защищённости данных при размещении в среде, подверженной сбоям и внешним рискам. Для демонстрации работы модели был разработан пакет программ, иллюстрирующих метод хранения.

Также было разработано веб-приложение, использующее облачные сервисы в качестве хранилищ. Оно включает в себя пользовательский интерфейс, бизнес-логику обработки файлов и модуль взаимодействия с облачными хранилищами.

Особое внимание было уделено вопросу повышения безопасности хранения. Для этого реализован механизм динамического задания размеров частей данных при разбиении файла, что делает схему хранения менее предсказуемой и затрудняет восстановление информации при частичном доступе к данным. Также предусмотрены различные сценарии восстановления файлов при недоступности одного из облаков, что подтверждает устойчивость системы к отказам.

Проведённое тестирование подтвердило применимость модели, устойчивость к сбоям, а также корректность полученных данных. Реализация проекта показала, что подход RAID 5 может быть успешно адаптирован для распределённой облачной инфраструктуры без использования физического RAID-контроллера.

Таким образом, в рамках работы создано решение, сочетающее преимущества облачного хранения и отказоустойчивости, а также повышение защищённости при хранении данных с применением подготовленной модели, что делает его актуальным инструментом для безопасного и надёжного управления данными.

Основные источники информации:

1. Chen P.M. RAID: High-performance, reliable secondary storage / P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson // ACM Computing Surveys (CSUR). – 1994г. – Vol. 26. – P. 145–185.
2. Массивы дисков в RAID [Электронный ресурс] URL: <https://1bx.host/>

stati/obshchie-stati/massivy-diskov-v-raid0-raid1-raid5-raid10/ (дата обращения: 19.01.2025)

3. Павлова А.А. Получение доступа к данным, содержащимся в RAID 5 / Павлова А.А., Молодцова Ю.В. // Вестник Алтайской академии экономики и права. - 2022г. -№6-2. -с. 356-360.
4. ASP.NET Core [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/aspnet/core/security/cors?view=aspnetcore-9.0> (дата обращения: 04.02.2025)
5. mdn web docs HTML [Электронный ресурс] URL: <https://developer.mozilla.org/ru/docs/Web/HTML> (дата обращения: 22.03.2025)