МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и информационных технологий

ОРГАНИЗАЦИЯ БЫСТРОДЕЙСТВУЮЩЕГО ЗАЩИЩЕННОГО ОБЛАЧНОГО ХРАНИЛИЩА

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 «Информатика и вычислительная техника»
факультета компьютерных наук и информационных технологий
Лукьянова Александра Юрьевича

Научный руководитель:
Доцент кафедры ДМиИТ, к.т.н., доцент_____ Соколов А.О.

подпись, дата

Зав. кафедрой:
Доцент кафедры ДМиИТ, к.ф.-м.н., доцент _____ Тяпаев Л.Б.

подпись, дата

Саратов 2019

ВВЕДЕНИЕ

Актуальность темы.

Современные технологии на сегодняшний день развиваются очень стремительно, знания увеличиваются в огромных объёмах, и хранение информации является одним из главных направлений развития человека. Использование облачных технологий охватило все слои человечества, многие пользуются ими для хранения своих данных, что позволяет сэкономить свободную память своего компьютера, научное сообщество использует облачные вычисления, для быстрого и удобного получения информации об исследованиях.

Облачное хранилище данных – модель онлайн-хранилища, в котором данные хранятся на многочисленных, распределённых в сети серверах, предоставляемых в пользование клиентам, в основном третьей стороной. Данные хранятся, и обрабатываются, в облаке, которое представляет собой, с точки зрения клиента, один большой, виртуальный сервер. Физически, такие сервера могут располагаться весьма удалённо друг от друга географически, вплоть до расположения на разных континентах [1].

Развитие облачных хранилищ приводит к определенным методам оптимизации процессов передачи данных пользователя, и их защиты.

Цель магистерской работы – является создание высокопроизводительного, защищенного облачного хранилища, включающее в себя разработку бот программы облачного хранилища

Поставленная цель определила **следующие задачи**:

1. провести анализ облачных хранилищ;
2. изучить протоколы передачи и способы защиты данных;
3. изучить технологии Long polling и Web-hooks;
4. определить структуру и содержание учебного пособия;
5. разработать и провести тестирование производительности бот программы;
6. разработать методику оптимизации облачного хранилища.

Методологические основы Разработка и оптимизация облачных хранилищ, представлены в работах Ю.Ю. Громов, Кожевников Д.О., Пустобаев А.И., Изюмов А.Е., Беккер М.Я., Моисеев А.Л., Коваленко О.С..

Структура и объём работы. Магистерская работа состоит из введения, 5 глав, заключения, списка использованных источников и одного приложения. Общий объем работы – 51 страница, из них 44 страницы – основное содержание, включая 19 рисунков и 2 таблицы, листинг программы в качестве приложения, список использованных источников информации – 21 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первая, вторая и третья главы посвящены теоретическому исследованию информации об облачных технологий и возможности взаимодействия с выбранным интерфейсом.

1 Анализ облачных хранилищ. В наше время большинство людей пользуются облачными хранилищами, которые предоставляют новый способ взаимодействия с пользовательскими данными. Существуют несколько базовых функций информационного взаимодействия пользователей с облачными хранилищами:

1. Резервное копирование данных пользователя.
2. Хранение большого объема личных и персональных данных.
3. Синхронизация важной информации между компьютерами.
4. Доступ к персональным данным из любого места.

Как можно заметить, облачные хранилища помогают пользователям в цифровую эпоху решать задачи, которые были перечислены выше.

На рынке сегодня существует множество платформ для организации облачных вычислений. Существуют как проприетарные (коммерческие), так и открытые (свободные).

Проведя анализ современных облачных платформ, можно выделить Google Drive, Яндекс.Диск, Mega, Облако Mail.ru, которые предоставляют

наибольший объём дискового пространства, но не у всех поставщиков услуг используется 256-битный ключ шифрования. Помимо этого на сегодняшний день многие пользователи работают не с одним поставщиком услуг, из этого можно выявить некоторые недостатки:

1. Отсутствие единого ПО для всех облачных хранилищах.
2. Отсутствие поддержки шифрования файлов.
3. Отсутствие стандартных функций для межоблачного взаимодействия.
4. Отсутствие унифицированного дизайна и логики.

2 Безопасность. HTTPS. Для разработки облачного хранилища, необходимо было начать с протоколов передачи и сохранения безопасности данных пользователей. Программное обеспечение, которое было выбрано, предоставляет HTTPS протокол, что удовлетворяет заявленным требованиям безопасности и передачи данных.

HTTPS (Hypertext Transport Protocol Secure) – это протокол, который обеспечивает конфиденциальность обмена данными между сайтом и пользовательским устройством. Безопасность информации обеспечивается за счет использования криптографических протоколов SSL/TLS, имеющих 3 уровня защиты.

Протоколы TLS и SSL используют асимметричную криптографию для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности для сохранения целостности сообщений. Данные протоколы широко применяются в приложениях, работающих с сетью Интернет.

3 Производительность. Long polling. Для реализации проекта, используется готовое приложение с возможностью реализации программы Бот. Приложение telegram предоставляет два метода взаимодействия бот-программы с сервером telegram, это webhooks и long polling.

Webhooks - это «пользовательские HTTP-обратные вызовы». Как правило, они инициируются каким-то событием, например, запуском кода в

репозитории или комментарием, отправленным в блог. При использовании «Short polling» клиент отправляет регулярные запросы на сервер, и каждый запрос пытается «вытащить» любые доступные события или данные. Если нет событий или данных, сервер возвращает пустой ответ, и клиент ждет некоторое время перед отправкой другого запроса опроса. Частота опроса зависит от задержки, которую клиент может терпеть при получении обновленной информации с сервера. Этот механизм имеет тот недостаток, что потребляемые ресурсы (серверная обработка и сеть) сильно зависят от приемлемой задержки при доставке обновлений от сервера к клиенту. Если допустимая частота опроса низкая (например, порядка секунд), это может привести к неприемлемой нагрузке на сервер, сеть или и то, и другое.

В отличие от «Short polling», «Long polling» пытается минимизировать как задержку в доставке сообщений сервер-бот-программа, так и использование ресурсов обработки сети. Сервер Telegram достигает этой эффективности, отвечая на запрос только тогда, когда произошло определенное событие, состояние или время ожидания. Эффективность в том, что в любой момент времени бот-программа выполняет запрос о наличии данных на сервере, вне зависимости от уже полученной информации.

Четвёртая и пятая главы посвящены реализации прототипа облачного хранилища, и применению метода оптимизации бот-программы.

4 Разработка облачного хранилища. Как ранее упоминалось, Бот-программа облачного хранилища разработана на уже реализованном приложении Telegram. Для разработки Бот-программы, необходимо было воспользоваться API приложения, предоставляемое компанией telegram.

Для создания Бот-программы, необходимо зарегистрироваться в приложении телеграм получить «токен» – уникальный id приложения. Получение токена осуществляется через запросы команд в сообщениях специальному боту @BotFather.

Первоначально необходимо добавить @BotFather в свои контакты, далее написать в сообщении команду /start, в ответе бот выдаст список всех его команд. Далее необходимо выбрать команду /newbot, после отправки, необходимо будет придумать название своего бота, которое должно оканчиваться на «…bot», и отослать это название @BotFather. Если данное название не занято, то ответе будет указа необходимый токен бота и ссылка для быстрого добавления бота в контакты. Для реализации взаимодействия Бот-программы с сервером telegram необходимо использовать команды API. Все запросы к Telegram Bot API должны осуществляться через HTTPS в следующем виде: https://api.telegram.org/bot<token>/НАЗВАНИЕ_МЕТОДА. Допускаются GET и POST запросы. Ответ придёт в виде JSON-объекта, в котором всегда будет булево поле «ok» и опциональное строковое поле «description», содержащее описание результата. Если поле «ok» истинно, то запрос прошёл успешно и результат его выполнения можно увидеть в поле «result». В случае ошибки поле «ok» будет равно «false», а причины ошибки будут описаны в поле «description». Кроме того, в ответе будет целочисленное поле error_code, но коды ошибок будут изменены в будущем.

Для получения обновлений с сервера существует способ «getUpdates». В Данном методе используется технология Long polling. Ответ возвращается в виде массива объектов.

5 Тестирование. Для подтверждения результатов применения различных методик увеличения производительности, будут производиться тестирования. Перед началом тестирования необходимо разобрать более выгодные способы тестирования программного обеспечения.

Широко используемыми методами тестирования являются модульное тестирование, интеграционное тестирование, системное тестирование, приемочное тестирование, нагрузочное тестирование, стрессовое тестирование, объемное тестирование и тестирование стабильности.

В качестве основного тестирования было выбрано тестирование производительности. Данное тестирование наглядно показывает нагрузку

облачного хранилища при определенном объеме данных. Для наглядности определения точек временного учета обработки файлов, была реализована UML диаграмма взаимодействия хранилища с сервером телеграмм и пользователем.

Входной точкой временного замера обработки файлов была выбрана точка получения обновлений о наличии файлов на сервере использованного интерфейса. Конечной точкой временного замера обработки файлов была выбрана точка отправки уведомления пользователю, что файл успешно загружен (либо произошла ошибка).

Для каждого теста, была выбрана группа файлов: 1 Kb,20Kb, 10 Mb. Данный объем файлов выбирался исходя из самого минимального, самого максимального и самого часто используемого объема файлов для хранения.

Для каждого теста планировалось провести по 20 серий тестов, с разным количеством файлов (1, 5, 25, 125). На каждом тесте замерялось время обработки файла от входной до конечной точки.

Для каждого теста был построен график зависимости суммарного времени при прохождения тестов от количества файлов. Так же была построена линия аппроксимация по полимиальной функции, которая показывает рост нагрузки, при определенном количестве обрабатываемых файлов и их размере. С определенного момента времени данная функция начинает резко возрастать, данный момент устанавливался как критический.

Для решения проблемы резкого роста времени обработки файла, был разработан метод оптимизации, путем применения алгоритма распределения приоритетов загружаемым файлам.

Блок схема алгоритма распределения приоритетов загружаемым файлам, представлена на рисунке 1.

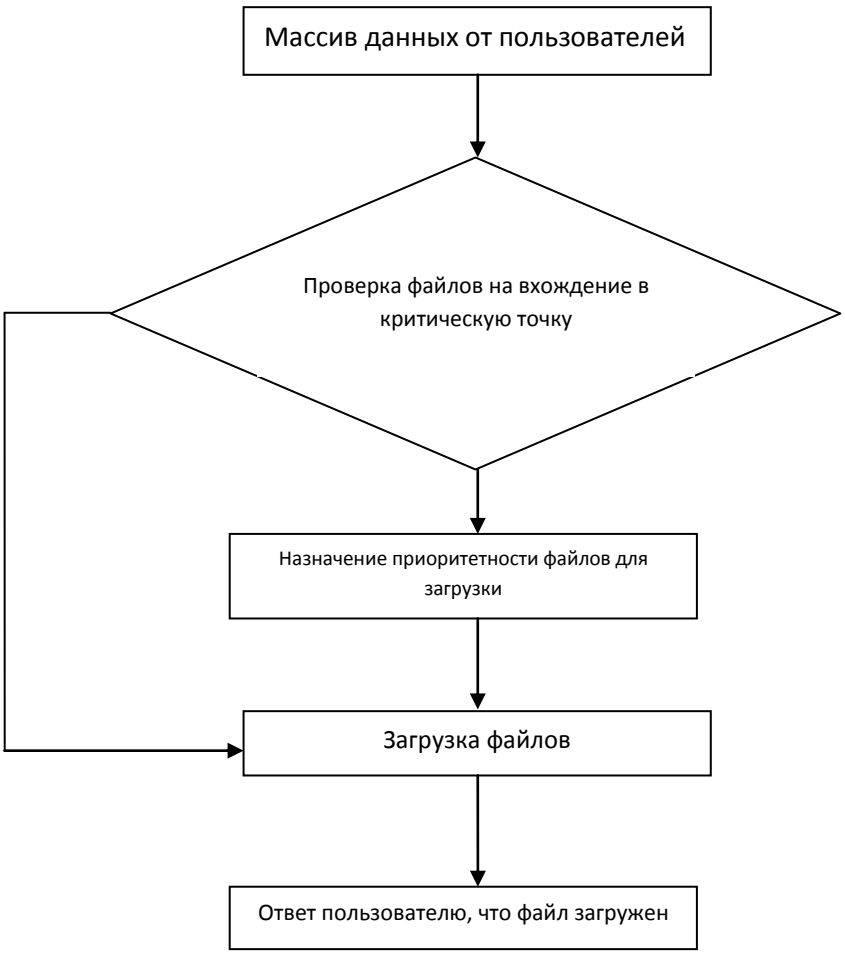

Рисунок 1 – алгоритм распределения приоритетов загружаемым файлам

Вначале получаем массив данных от пользователей. Следующим шагом происходит проверка файлов и их количества на возможность попадания в критическую точку.

Если была обнаружена совокупность файлов, которые попадают в критическую точку, то данным файлам присваивается высший приоритет загрузки. Следующим шагом будет являться загрузка файлов в облачное хранилище.

Заключительным этапом является уведомление пользователя об успешной загрузке файлов.

По разработанному методу оптимизации, примененному к облачному хранилищу, было произведено тестирование производительности, и график результатов представлен на рисунке 2.

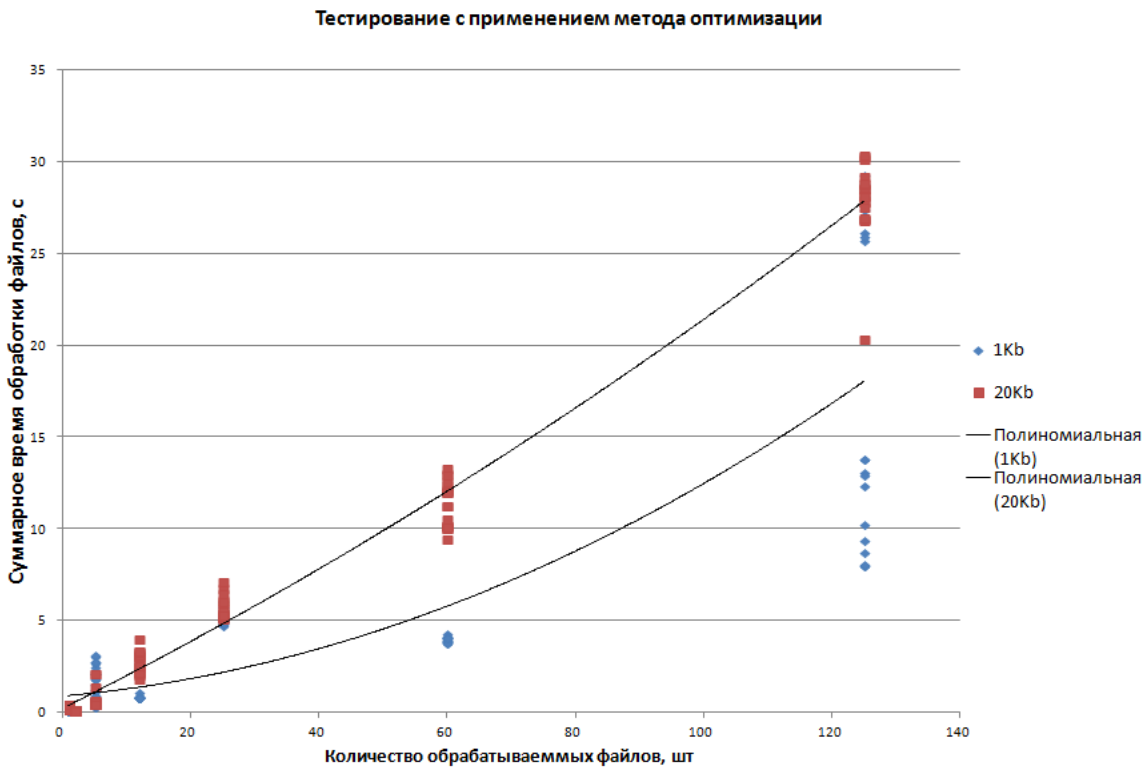

Рисунок 2 – результат тестирования с применением метода оптимизации

На графике можно заметить, что применяя данный метод оптимизации, удалось улучшить временные показатели, сократить время обработки файлов, тем самым подтвердить быстродействие облачного хранилища по сравнению со стандартным.

ЗАКЛЮЧЕНИЕ

Результатом магистерской работы является, разработанное облачное хранилище, и методика его оптимизации. Были выполнены поставленные задачи, разработан прототип бот – программы облачного хранилища и реализовано взаимное взаимодействие пользователя с облачным хранилищем, а так же проведено тестирование производительности прототипа, по результатам которого был разработан метод оптимизации, путем применения алгоритма распределения приоритетов загружаемым

файлам. Данный метод позволяет ускорить обработку загружаемых файлов облачным хранилищем. Листинг программы представлен в приложении А.

Таким образом, поставленная цель и задачи магистерской работы полностью выполнены.

Основные источники информации:

1 Облачное хранилище данных. [Электронный ресурс]: Википедия. Свободная энциклопедия / текст доступен по лицензии Creative Commons Attribution-ShareAlike; Wikimedia Foundation, Inc, некоммерческой организации. URL: http://ru.wikipedia.org/wiki/Облачное_хранилище_данных (дата обращения 25.05.2018) (Последнее изменение страницы: 16:50, 28 октября 2018). Яз. Рус.

2 Ю.Ю. Громов Моделирование протокола взаимодействия TLS на основе сети Петри при реализации защищенных соединений / Ю.Ю. Громов, А.В. Яковлев, Ю.В. Минин, Е.О. Васюкова // Вестник Воронежского государственного технического университета 2014. № 2. С. 19-23.

3 The OAuth 2.0 Authorization Framewor [Электронный ресурс]: URL: https://tools.ietf.org/html/rfc6749#section-1.1 (дата обращения 27.05.2018). Загл. с экрана. Яз. англ.

4 Кожевников Д.О. АКТУАЛЬНЫЕ ПРОБЛЕМЫ ОРГАНИЗАЦИИ МОДУЛЬНОГО ТЕСТИРОВАНИЯ КЛАССОВ ПРОГРАММНОГО КОДА // Образовательные ресурсы и технологии 2014. № 4. С. 134-141

5 Виды Тестирования Программного Обеспечения [Электронный ресурс]: URL: http://www.protesting.ru/testing/testtypes.html/ (дата обращения 27.04.2018). Загл. с экрана. Яз. рус.

6 Пустобаев А.И. О сервисе рассылки push-уведомлений: // International Journal of Open Information Technologies. 2015. № 6. С. 13-20.

7 Изюмов А.Е. Исследование безопасности протокола HTTP // Научно-технический вестник информационных технологий, механики и оптики 2005. № 3. С. 161-166.

8 Беккер М.Я. Использование цифровых сертификатов и протоколов SSL/TLS для шифрования данных при облачных вычислениях / М.Я. Беккер, А.О. Терентьев, Ю.А. Гатчин, Н.С. Кармановский // Научно-технический вестник информационных технологий, механики и оптики 2011. № 4. С. 125-129.

9 Власенко А.В. Анализ уязвимостей и моделирование атак на данные трафика “https” / Власенко А.В Дзьобан П.И. // Ежеквартальный рецензируемый, реферируемый научный журнал «Вестник АГУ» 2017. № 2. С. 109-115.

10 Моисеев А.Л. Методы тестирования и диагностирования компьютерных сетей / А.Л. Моисеев, Р.Р. Моисеева, В.В. Шаров, Ю.Н. Зацаринная // Вестник казанского технологического университета 2014. № 1. С. 315-316.

11 Коваленко О.С. Обзор проблем и состояний облачных вычислений и сервисов / О.С. Коваленко, В.М. Курейчик // Известия ЮФУ. Технические науки 2012. № 7. С. 146-153