Лабораторная работа №1

«Сравнение клиент-серверной архитектуры и одноранговой (P2P) взаимодействия в распределённых системах»

Цель работы:

Изучить особенности клиент-серверной архитектуры и одноранговой взаимодействия в распределённых системах и провести их сравнительный анализ на основе практической реализации.

Задание на лабораторную работу:

В ходе выполнения лабораторной работы необходимо выполнить сравнительный анализ двух видов распределённых систем – клиент-серверной архитектуры и одноранговой (P2P) взаимодействия.

Для этого необходимо реализовать простейшее программное приложение, реализующее клиент-серверную модель взаимодействия, в котором клиент инициирует взаимодействие и отправляет запрос серверу, а сервер обрабатывает запрос и возвращает ответ клиенту.

Также необходимо разработать простейшее программное приложение, которое реализует модель одноранговой взаимодействия, в котором каждый узел системы должен иметь возможность как отправлять, так и принимать и обрабатывать запросы.

Затем необходимо провести сравнительный анализ данных подходов к построению распределённых систем.

Рекомендации к выполнению:

В качестве среды разработки и языка программирования рекомендуется использовать средства, поддерживающие сетевое взаимодействие (например,

Python, Java или C#). Конкретный выбор языка не влияет на содержание лабораторной работы.

Для разработки клиент-серверного приложения рекомендуется реализовать простейшее браузерное приложение, которое представляет собой простую форму с полями для ввода данных.

Для разработки приложения, которое реализует модель однорангового взаимодействия, рекомендуется реализовать простейшее консольное приложение, которое представляет собой чат для обмена текстовыми сообщениями.

Приложение позволяет двум пользователям подключаться друг к другу напрямую, без выделенного сервера. После запуска, пользователь вводит своё имя и может отправлять текстовые сообщения. Текстовые сообщения с указанием имени отправителя сразу отображаются на экране.

Для реализации такого приложения рекомендуется использовать сокеты.

Требования к отчёту.

1. Ход выполнения работы и описание разработанных приложений;

2. Снимки экрана, подтверждающие работу приложений;

3. Сравнительный анализ двух подходов к построению распределённых систем;

4. Исходный код разработанных приложений;

5. Вывод.

Лабораторная работа №2

«Разработка и исследование микросервисной архитектуры распределенного приложения»

Цель: Изучение принципов микросервисной архитектуры распределенных систем, а также формирование практических навыков проектирования слабо связанных сервисов, взаимодействующих между собой по сети и использующих независимые хранилища данных.

Задачи:

- Спроектировать распределенное приложение на основе микросервисной архитектуры;
- Разработать не менее двух независимых микросервисов;
- Обеспечить для каждого микросервиса отдельную базу данных;
- Реализовать REST-интерфейсы для доступа к функциональности микросервисов;
- Реализовать сетевое взаимодействие между микросервисами;
- Продемонстрировать базовый сценарий совместной работы сервисов.

Предметная область выбирается студентом самостоятельно (например: управление пользователями и заказами, библиотечная система, бронирование ресурсов, учет заявок и т.п.).

Распределенное приложение должно состоять как минимум из двух микросервисов, каждый из которых:

- реализует ограниченный набор бизнес-функций;
- развертывается как отдельное приложение;
- использует собственную базу данных;
- взаимодействует с другими микросервисами посредством сетевых запросов.

Требования к архитектуре системы. Проектируемая система должна соответствовать следующим требованиям:

- Использовать микросервисную архитектуру;
- Микросервисы должны быть логически и технологически независимы;
- Взаимодействие между микросервисами должно осуществляться через сеть (HTTP, REST);
- Каждая база данных должна обслуживаться только своим микросервисом.

Функциональные требования. Минимальный набор функций системы должен включать:

- операции создания и получения данных в каждом микросервисе;
- сетевое обращение одного микросервиса к другому;
- формирование итогового ответа, использующего данные нескольких сервисов.

Требования к реализации. При выполнении лабораторной работы допускается использование любых современных технологий и средств разработки. Рекомендуется применять REST-архитектуру, формат обмена данными JSON, реляционные и нереляционные базы данных. Язык программирования, фреймворки и СУБД выбираются студентом самостоятельно.

Лабораторная работа №3

«Обеспечение безопасности при межсервисном взаимодействии в распределенных системах на примере микросервисной архитектуры.»

Цель: Научиться реализовывать базовые механизмы безопасности при взаимодействии двух микросервисов: аутентификацию с использованием токенов и шифрование данных при передаче.

Задачи.

Должны быть реализованы как минимум два микросервиса:

● Один микросервис — сервис аутентификации (откуда остальные сервисы могут получать валидные токены).

● Второй микросервис — сервис данных (сервис отдает какие-то данные авторизованным пользователям или сервисам).

Требования:

Необходимо выполнить реализацию аутентификации, безопасного обмена данными между микросервисами, логирование всех запросов и попыток доступа к данным.

Лабораторная работа №4 «Исследование репликации данных и разрешения конфликтов в распределенной системе»

Цель: изучение принципов репликации данных в распределенных системах, а также анализ механизмов возникновения и разрешения конфликтов при параллельных операциях записи с использованием моделей ослабленной согласованности.

Задачи работы:

- смоделировать простую распределенную систему, состоящую из нескольких узлов;

- реализовать механизм репликации данных между узлами;
- смоделировать параллельные операции записи, приводящие к конфликту;
- выявить конфликтующее состояние реплик;
- реализовать и применить алгоритм разрешения конфликтов;
- проанализировать полученные результаты.

Описание предметной области

Рассматривается распределенная система хранения данных, состоящая из нескольких узлов, каждый из которых содержит локальную копию (реплику) одного и того же объекта данных. Узлы системы взаимодействуют между собой по сети, в которой возможны задержки передачи сообщений.

В качестве объекта данных может использоваться простая сущность, например:

- числовое значение;
- строка;
- запись с полями «значение» и «версия».

Записи и обновления могут выполняться на любом узле системы независимо, что соответствует модели leaderless replication и модели согласованности eventual consistency.

Требования к архитектуре системы

Разрабатываемая модель распределенной системы должна удовлетворять следующим требованиям:

- система должна состоять как минимум из трех узлов;
- каждый узел должен хранить собственную локальную реплику данных;
- обмен данными между узлами должен осуществляться асинхронно;
- должна быть предусмотрена возможность имитации задержек передачи сообщений;

- система не должна использовать глобальные синхронизированные часы.

Требования к функциональности

Реализуемая система должна поддерживать следующие функции:

- операцию записи данных на выбранный узел;
- операцию чтения данных с узла;
- передачу обновлений между узлами;
- обнаружение конфликтов при наличии нескольких версий данных;
- разрешение конфликтов с использованием одного из алгоритмов, например:
  - Last Write Wins;
  - векторные часы;
  - упрощенный вариант Version Vectors.

Требования к реализации

Лабораторная работа может быть реализована в виде программной модели или имитационного эксперимента. Допускается использование любого языка программирования высокого уровня (Python, Java, C++ и др.).

В рамках реализации допускается:

- использование логических или “счетчиковых” временных меток;
- хранение версии данных в виде дополнительного поля;
- вывод промежуточных состояний системы в консоль или лог-файл.

Реализация не требует использования реальных сетевых соединений и может быть выполнена в виде модели с имитацией сетевых задержек.

Ожидаемые результаты

В результате выполнения лабораторной работы студент должен:

- продемонстрировать понимание принципов репликации данных;
- показать, каким образом возникают конфликты в распределенных системах;

- реализовать и применить механизм разрешения конфликтов;
- проанализировать полученные результаты и сделать выводы о работе системы.

Выводы

По результатам лабораторной работы должны быть сформулированы выводы о влиянии выбранной модели репликации и алгоритма разрешения конфликтов на согласованность данных в распределенной системе.

Лабораторная работа №5 «Отказоустойчивость и безопасность РС на примере RPC сервиса»

Цель работы

Разработать систему, содержащую RPC-сервис с имитацией отказов и брокером сообщений.

Задание

1. Реализовать мини-RPC сервис с одним API-методом (пример: POST /orders — создать заказ, возвращает order_id и статус).

Сервис должен уметь *обнаруживать повторную отправку* запроса, например, по request_id. Если клиент повторил запрос из-за таймаута/потери, сервер не должен создавать заказ повторно.

2. Реализовать клиента с таймаутом и повторами запросов.
Клиент должен вызывать RPC и выполнять следующие действия:

- ставить таймаут;
- при отсутствии ответа *делать повторную отправку*.

3. Реализовать брокера сообщений со схемой публикации/подписки. Должна совершаться асинхронная обработка:

- Сервер после создания запроса публикует событие в topic.
- Worker (consumer) подписан на эту тему и выполняет некоторую работу, занимающую несколько секунд.

Рекомендации по выполнению

- Рекомендуемый язык программирования – Python версии 3.10 и новее;
- Предлагается использовать фреймворки FastAPI или Flask, а также библиотеки requests или urllib;
- Проект рекомендуется разбить на три сервиса (файла): RPC-сервис (server.py), клиент (client.py) и брокер сообщений (broker.py).

Требования к отчёту

1. Цель работы;
2. Ход выполнения работы и описание разработанных сервисов;
3. Примеры, демонстрирующие работу программы;
4. Исходный код разработанных приложений;
5. Вывод.