Практическая работа №2.

Развёртывание простого распределённого приложения.

Постановка задачи:

Создать прототип распределённого приложения согласно варианту. «Прототип» в данном случае означает, что фактическая реализация нагрузки, выполняемой узлом, может быть заменена на сопоставимые задержки (имитация работы). Т.е., например, для распределённой файловой системы можно не реализовывать хранение и передачу файлов, но необходимо хранить на каждом узле список «сохранённых» у него файлов и выдерживать необходимые задержки при получении и передаче файлов. Также нужно создать интерфейс для тестирования вашего приложения или уметь продемонстрировать его работу другим наглядным способом.

Должны быть созданы различные версии приложения в зависимости от используемого способа связи узлов. Необходимо реализовать следующие способы связи:

– REST-api;

– RPC (можно использовать любой RPC-протокол или реализовать свой);

– Kafka;

– RabbitMQ;

– * WebSocket (по желанию).

Для REST-api или RPC нужно также реализовать автоматическое обнаружение узлов (в локальной сети или через общий статичный узел).

Допустимо использование любых языков и фреймворков, но желательно уметь объяснить свой выбор.

Вариант выбирается по последним цифрам номера зачётной книжки. При желании можно взять любой другой вариант, если он никем не занят или получилось договориться об обмене.

Варианты приложений:

1. Распределённая файловая система. Обязательные операции: просмотр иерархии каталогов, добавление файла, получение файла, удаление файла.

2. DSM (виртуальная распределённая память). Обязательные операции: чтение ячейки памяти, запись в ячейку памяти, взятие и освобождение блокировки доступа к конкретной ячейке (при чтении/записи). Все ячейки должны представлять собой непрерывную «ленту» с последовательно идущими адресами.

3. Система распределённого шифрования. Обязательные методы: зашифровать файл, расшифровать файл. Шифрование отдельных блоков файла должно выполняться параллельно на разных узлах (не менее 10 узлов).

4. Распределённый чат. Обязательные методы: отправка сообщения, получение всех сообщений, адресованных пользователю, удаление сообщения, получение списка пользователей, создание нового пользователя. История сообщений должна быть распределена по узлам. Любой узел должен являться потенциальной точкой входа в чат. Реализовывать аутентификацию не обязательно.

5. Система расчёта СДУ методом Эйлера. Обязательные методы: решить СДУ для заданных коэффициентов. Вычисления должны быть распределены по узлам (не менее 5 узлов).

6. Финансовая система. Обязательные методы: получение баланса пользователя, перевод денег от одного пользователя другому, добавление денег на баланс пользователя, снятие денег с баланса пользователя. Сведения о балансе пользователей (в виде истории транзакций или простой бд) должны быть распределены по разным узлам, но не теряться при потере одного из узлов.

7. Система распределённой аутентификации. Обязательные методы: проверка пароля, получение сведений о пользователе, получение списка пользователей, регистрация пользователя, удаление пользователя.

8. Система распределённого предоставления контента. Обязательные методы: получить контент с заданным именем, добавить контент, удалить контент с заданным именем. Контент должен быть реплицирован не менее чем на два узла, при этом загрузка должна быть осуществлена с наиболее быстрого. В качестве контента можно использовать текстовые или html файлы (использование более сложных вариантов также не запрещено).

9. Система для автоматизированной отправки запросов. Обязательные методы: настроить отправку запросов на заданный локальный узел в заданном режиме, остановить отправку запросов, задать интервал отправки запроса, задать тело запроса. Должна поддерживаться два режима отправки запросов: постоянный и интервальный-синхронизированный. В первом случае все узлы непрерывно отправляют запросы на заданный узел, пока не получат команду отмены. Во втором случае узлы должны отправлять запросы одновременно с некоторым интервалом (который может быть изменён отдельным запросом). Должен быть предоставлен способ отследить успешность выполненных запросов.

10. Многопользовательская игра. Отдельные узлы реализуют разные игровые комнаты, в первую очередь занимаются свободные узлы. Обязательные методы: создать игровую комнату (сервер), присоединиться к игровой комнате по её идентификатору, удалить игровую комнату, выйти из игровой комнаты, методы для выполнения действий самой игры. Игровая механика на выбор студента, вполне можно реализовать крестики-нолики.

11. Многопользовательская игра с шардированием. Игра представляет собой текстовый рогалик (roguelike), где карта разбита на чанки, которые размещены на разных узлах. Обязательные методы: передвинуться вправо/влево/вверх/вниз, запросить чанк по координатам, запросить 9 чанков, центральным из которых будет чанк с заданными координатами, начать игру с заданным идентификатором. Одновременно на одной карте могут находиться несколько игроков (как на одном чанке, так и

на разных). Игроки должны видеть других игроков и не могут проходить их насквозь. По желанию студента могут быть реализованы дополнительные методы.

12. Система параллельного тестирования. Обязательные методы: загрузить программу, добавить тест-кейс, протестировать, очистить. Программа загружается в виде исходного кода на языке по выбору студента. Тест-кейсы представляют собой текстовые файлы, имеющие формат и возможности по выбору студента. При вызове метода «протестировать» на разных узлах создаются экземпляры программы, на которых выполняются тесты. Если во время тестирования подана новая программа или команда очистки, тестирование всё равно должно быть корректно завершено, после чего будут обработаны новые данные.

13. Система проведения электронного голосования. Обязательные методы: войти под заданным именем, отдать голос за один из вариантов (варианты могут быть заданы изначально или задаваться через интерфейс), валидировать свой голос. Голос должен быть подписан криптографически, при валидации проверяется наличие голоса с такой подписью и за что он отдан.

14. Система коллективного редактирования документа. Документ – совокупность некоторых неделимых элементов – блоков. В качестве блока для текстового документа можно взять абзац. Блоки распределены по узлам системы. Работа с документом может осуществляться несколькими пользователями одновременно, должны быть предусмотрены блокировки параллельной записи. Обязательные методы: изменить блок, добавить блок, удалить блок, просмотреть документ в целом (получить все блоки документа).

15. Система «Умный дом». В этом варианте узлы не равноправны и делятся на 4 группы: датчики, актуаторы, хранилища (координаторы) и интерфейсы. Датчики генерируют события через случайные промежутки времени и отправляют их координаторам. Координаторы обрабатывают

события и сохраняют историю этих событий. По результатам обработки они отправляют актуаторам команды (открыть окно, включить свет и т. д.). Узлы-интерфейсы предоставляют пользовательский интерфейс, через который можно посмотреть историю событий и отправить команду узлу-актуатору.

16. *(Вариант исключительно по желанию) Распределённая база данных. Обязательные методы: создать таблицу с полями заданных типов и с заданными именами, получить сведения из таблицы, изменить сведения в таблице, добавить сведения в таблицу, начать транзакцию, отменить транзакцию, закрепить транзакцию. Таблицы должны быть распределены по узлам. Транзакции должны обеспечивать ACID. При отключении любого из узлов БД должна или продолжать функционировать, предоставляя хотя бы часть данных.

В отчёте по практической работе (сдаётся в электронном виде после демонстрации результатов вашей работы на паре/консультации) должны быть приведены скриншоты работы программы и её исходный код, а также содержимое файлов dockerfile и docker-compose.yml.