СИСТЕМА АВТОМАТИЧЕСКОГО РЕЗЕРВНОГО КОПИРОВАНИЯ И ОБРАБОТКИ ДАННЫХ НА ЯЗЫКАХ ПРОГРАММИРОВАНИЯ JAVA, JS

Аль-Саррих З.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Видничук В.Н. – м.т.н. ассистент кафедры ПОИТ

Облачные технологии фундаментально меняют ожидания в отношении того, как должны распределяться, управляться и потребляться ресурсы вычислений, хранения и сети в глобальном масштабе. Мощные вычислительные системы и системы хранения данных, высокий уровень безопасности, легкая доступность и настраиваемость, надежная масштабируемость и функциональная совместимость, экономичные и гибкие «облачные» вычисления являются самыми востребованными в современном быстрорастущем деловом мире. Клиент, организация или предприятие, внедряющее новую облачную среду, может выбрать подходящую инфраструктуру, платформу, программное обеспечение и сетевые ресурсы для каждого бизнеса, каждый из которых обладает эксклюзивными возможностями.

Целью проекта является повысить эффективность автоматического резервного копирования и обработки данных путём разработки онлайн-системы облачного хранения на языках программирования JavaSript и Java.

Объектом исследования являются процессы резервного копирования и обработки данных, реализуемые в системе облачного хранения данных.

Как правило, компьютеры и программы, составляющие информационную систему, отличаются друг от друга. Некоторые из них имеют собственные ресурсы (файловая система, процессор, принтер, база данных и т.д.), другие могут получить доступ к этим ресурсам. Компьютер (или программа), которая управляет ресурсом, называется сервером ресурса (файловый сервер, сервер базы данных, вычислительный сервер). Клиент и сервер ресурса могут находиться на одном компьютере или на разных компьютерах, подключенных к сети.

Общие принципы взаимодействия в сети с серверами, узлами, предоставляющими определенные функции (сервисы), и клиентами, потребляющими эти функции определяет клиент-серверная архитектура.

Клиент-серверные технологии определяют свои собственные или использует существующие правила взаимодействия между клиентом и сервером, называемые протоколом обмена (протоколом связи).

Ниже описаны преимущества клиент-серверных систем.

Клиент-серверный подход является модульным, при этом компоненты программ сервера компактны и автономны.

Неисправность сервера не влияет на другие компоненты операционной системы.

Автономность компонентов позволяет запускать их на нескольких процессорах на одном компьютере (симметричная обработка) или на нескольких компьютерах в сети (распределенные вычисления).

Задача клиента, как правило, заключается в предоставлении пользовательских услуг и, прежде всего, пользовательского интерфейса, т.е. средств для получения, отображения и обработки введенных пользователем данных, которые служат основой для запроса к серверу.

Любая сеть (включая одноранговую сеть), основанная на современных сетевых технологиях, обычно имеет основанные на двухзвенной архитектуре элементы взаимодействия клиент-сервер. Данный тип сети называется двухзвенной, потому что три основных компонента должны быть распределены между двумя узлами (клиент и сервер).

Двухзвенная архитектура используется в клиент-серверных системах, где сервер напрямую и полностью отвечает на запросы клиентов, используя только собственные ресурсы. То есть сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса.

Минус двухзвенной архитектуры заключается в том, что если упал сервер или отвалилась база данных, то есть испортилось 1 звено – система не способна функционировать и клиенты не способны работать с системой.

Поэтому для предотвращения появления в работе системы аналогичных проблем в архитектуре приложения реализовывают кластер с балансировкой нагрузки, отвечающий за распределение запросов пользователей между серверами. Для распределения запросов в кластере используется один или несколько входных вычислительных узлов, через которые задачи перенаправляются с одной машины на другую.

Благодаря клиент-серверной организации, разрабатываемая нами программа позволит удаленно работать с данными, что очень важно в современных условиях, когда у специалистов есть возможность работать на легко переносимых устройствах. На сервере предусмотрена возможность параллельной обработки запросов. Система также должна обеспечивать возможность просмотра,

редактирования, удаления данных, создания новых записей. Соответствующие изменения должны быть внесены в базу данных с помощью MySQL-запросов на серверной части приложения.

Для разработки программного средства нам необходимо обратить внимание на архитектуру проектируемого программного обеспечения, которую мы проиллюстрируем с помощью диаграммы развертывания. Данная диаграмма используется для представления общей конфигурации и топологии распределенной программной системы и показывает распределение компонентов по отдельным узлам системы. Он также показывает наличие физических путей передачи информации между аппаратными устройствами, участвующими в реализации системы. Схема развертывания используется для визуализации элементов и компонентов программы, которые существуют только во время ее выполнения (выполнения). Он представляет только те компоненты, которые являются исполняемыми файлами или динамическими библиотеками. Компоненты, которые не используются во время выполнения, не отображаются на диаграмме развертывания.

При разработке архитектуры системы использовались услуги Amazon Web Services (AWS) – коммерческого общедоступному облака, поддерживаемого и разрабатываемого компанией Amazon с 2006 года. AWS предоставляет абонентам как услуги по моделированию инфраструктуры (виртуальные серверы, ресурсы хранения), так и услуги на уровне платформы (облачные базы данных, программное обеспечение для подключения к облаку, бессерверные облачные вычисления, средства разработки).

Облако развернуто в нескольких территориально распределенных центрах обработки данных, сгруппированных в группы близости, называемые «регионами». В пределах одного региона реализовано несколько «зон доступности», в пределах которых обеспечивается высокая доступность хостинг-услуг.

По состоянию на 2019 г. в 20 регионах насчитывается 60 зон доступности. Абоненты могут выбрать регион и зону доступности, а также имеют возможность организовать репликацию данных и передачу приложений между зонами доступности. В целях доступности разрабатываемого ПС как можно более большой группе лиц было принято решение арендовать сервера в 4 регионах: США, Западной Европе и 2 в Азиатско- Тихоокеанском регионе.

На рисунке 1 предоставлена диаграмма развертывания, показывающая узлы и связи между компонентами архитектуры.

Ключевая инфраструктурная услуга AWS – служба аренды виртуальных серверов EC2. Для реализации программно-аппаратной части сервиса используется два сервера: «DataBase Server» и «Backend Server». На виртуальные машины данных серверов была поставлена ОС CentOS Linux 7 x86_64. Для реализации пользовательской части ПС использованы 2 идентичных по наполнению сервера «Frontend USA Server» и «Frontend Russia Server». Данное решение используется для балансировки нагрузки между сетевыми устройствами (серверами) с целью сокращения времени обслуживания запросов пользователей. За балансировку нагрузки отвечает средство «Balance Loader».

Запросы от устройств пользователя, гостя или менеджера поступают в первую очередь на «Balance Loader» по протоколу https (протокол безопасной передачи данных, поддерживает технологию шифрования TLS/SSL). Балансировщик нагрузки направляет пользователя на один из двух серверов «Frontend USA Server» или «Frontend Russia Server» в зависимости от адреса, с которого поступил запрос. Это позволит улучшить работу пользователя с пользовательской частью системы.

Балансировщик нагрузки работает с серверами, отвечающими за отображение пользовательской части ПС, и роутером по протоколу TCP (один из основных протоколов передачи данных интернета). Соответственно таблице маршрутизации роутера, которая описывает соответствие между адресами назначения и интерфейсами, пакеты данных поступают на «Backend Server».

«Backend Server» работает с «DataBase Server» по PostgreSQL client-server протоколу и с помощью стандарта JDBC, который позволяет приложению взаимодействовать с СУБД. Стандарт JDBC основан на «концепции так называемых драйверов, позволяющих получать соединение с базой данных по специально описанному URL. Драйверы могут загружаться динамически (во время работы программы). Загрузившись, драйвер сам регистрирует себя и вызывается автоматически, когда программа требует URL, содержащий протокол, за который драйвер отвечает» [1].

Частью архитектуры является NAT, механизм в сетях TCP/IP, позволяющий преобразовывать IP-адреса транзитных пакетов. Этот механизм предотвращает внешний доступ к внутренним хостам, одновременно разрешая внутренний доступ к внешним. Трансляция создается при инициации соединения изнутри сети. Ответные пакеты, поступающие извне, соответствуют созданной трансляции и поэтому разрешены к прохождению. Если для пакетов, поступающих извне, соответствующей трансляции не существует (а она может быть созданной при инициации соединения или статической), они не пропускаются.

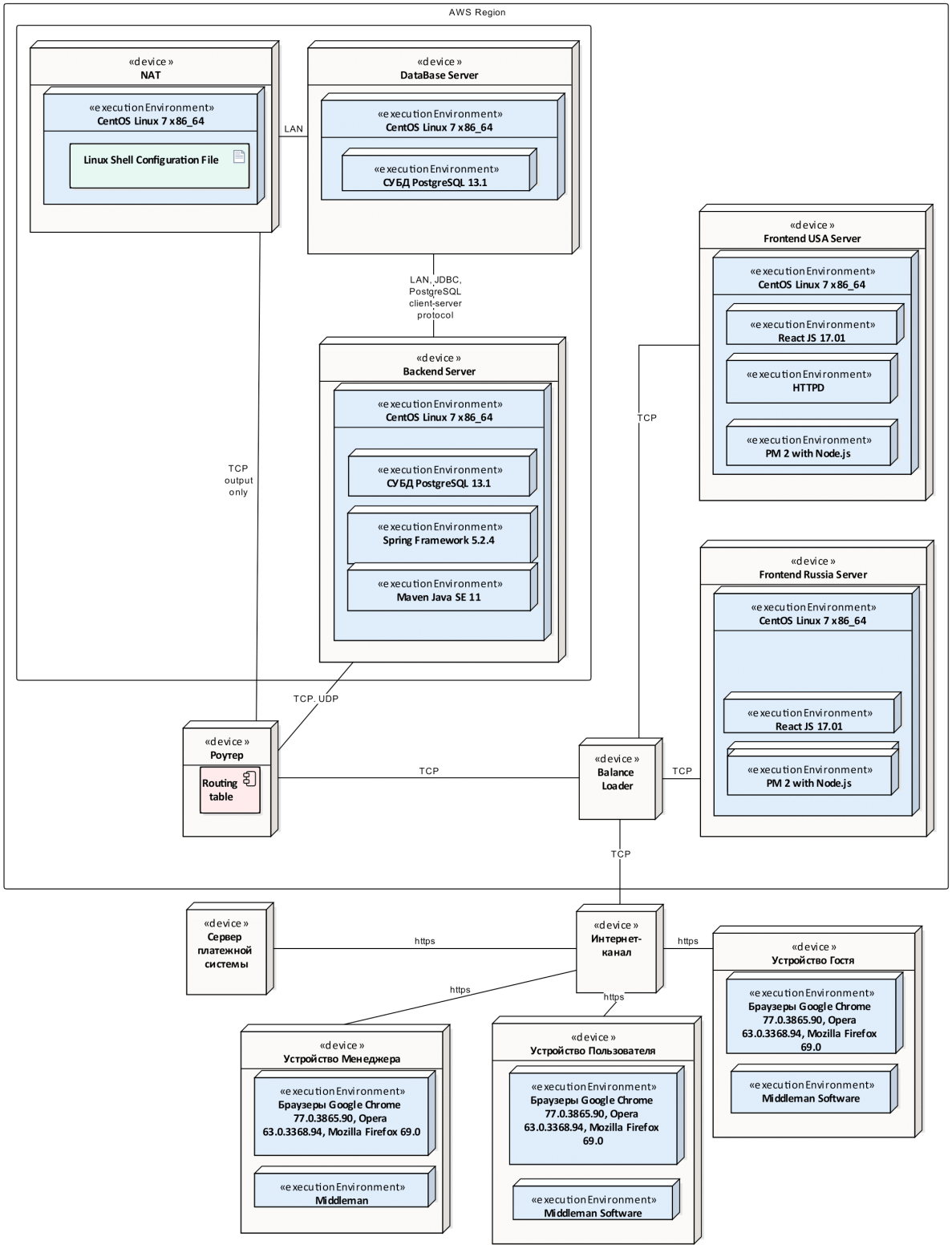

Рисунок 1 – Диаграмма развертывания

В итоге использование в системе двухзвенной клиент-серверной архитектуры с кластером с балансировкой нагрузки позволяет равномерно распределять запросы клиентов между доступными серверами и обеспечивает стабильную работу системы резервного копирования и обработки данных.

Список использованных источников:

1. Java Database Connectivity [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Java_Database_Connectivity. – Дата доступа: 03.04.2021.