МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

СЕТЕВАЯ СИСТЕМА МОНИТОРИНГА И УПРАВЛЕНИЯ СЕРВЕРАМИ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 02.04.03 — Математическое обеспечение и администрирование информационных систем
факультета КНиИТ
Рафикова Рустана Ринатовича

Научный руководитель
доцент, к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

ВВЕДЕНИЕ

Современная IT-инфраструктура является неотъемлемой частью подавляющего большинства бизнес-процессов. Сетевые технологии позволяют создавать единые системы с обширной географией. В этих условиях критически важным становится решение задач централизованного мониторинга и управления IT-ресурсами. Предприятия, использующие крупные информационные системы, в том числе и географически распределенные, остро нуждаются в оптимизации процедур мониторинга этих систем и поддержки их работоспособности.

Целью данной работы является проектирование собственного алгоритма мониторинга состояния системы на основе существующих решений и реализация собственного решения, использующее данный алгоритм.

Для достижения данной цели были поставлены следующие задачи:

— рассмотрение задачи мониторинга серверов и их управления;
— обзор существующих подходов и готовых продуктов;
— проектирование собственного решения на основе существующих технологий;
— разработка сетевой системы мониторинга и управления серверами.

Магистерская работа содержит следующие разделы:

— введение;
— задача мониторинга серверов и их управления;
— инструменты для разработки сетевой системы мониторинга и управления серверами;
— разработка сетевой системы мониторинга и управления серверами;
— заключение.

Данная работа демонстрирует пример использования нового механизма передачи данных в задаче отслеживания серверов и их управления, что позволяет добиться повышения скорости работы системы без нарушений требований безопасности.

1 Основное содержание работы

Задача мониторинга серверов и их управления. Данная глава описывает исследуемую задачу в частности общую проблематику и существующие подходы и решения.

В первой части данной главы был рассмотрен процесс мониторинга серверов и выделены его преимущества в использовании, а именно:

- сокращение количества сбоев ИТ-инфраструктуры;
- сокращение общего времени простоя ИТ-сервисов;
- сокращение времени реакции специалиста на инцидент;
- предотвращение возможных финансовых потерь по причине простоя ИТ-систем;
- централизованный контроль параметров всей инфраструктуры на одной консоли;
- повышение качества ИТ обслуживания.

Для использования системы, включающей в себя мониторинг ОС, оповещение о событиях, формирование отчетов, панель состояния и распределенное администрирование, важными вопросами являются расположение серверов в сети и определение возможных способов обмена сообщениями. Данные вопросы были описаны в подглаве «Метод получения информации с удаленных устройств».

Были выделены следующие схемы расположения серверов:

- оба сервера находятся в одной сети. Это может быть LAN, VPN или интернет. В этом случае оба сервера между собой могут как принимать, так и отправлять данные.
- один из серверов находится в сети интернет, а другой — в частной. В этом случае возможен только односторонний обмен данными.
- серверы находятся в разных сетях и передача данных производится через прокси-сервер. В этом случае также возможен только односторонний обмен данными.
- серверы находятся в разных сетях и передача данных производится через туннели. В этом случае возможет как односторонний обмен данными, так и двусторонний, а зависимости от количества туннелей. Однако данный способ является не безопасным, так как добавляет уязвимостей, в виде возможного несанкционированного доступа на закрытый ресурс;

— серверы находятся в разных сетях и передача данных производится через дополнительный сервер находящийся в глобальной сети. В этом случае также возможен только односторонний обмен данными с задержкой.

Также было рассмотрено взаимное сетевое расположение пользователя и системы мониторинга:

— сервер в глобальной сети в интернете.
— сервер в той же сети что и пользователь.
— сервер в другой сети, но есть VPN.
— пользователь находится в другой сети, однако есть прокси сервер или сетевой туннель.
— сервер и пользователь в разных сетях.

Во второй части данной главы был проведен обзор существующих решений задачи мониторинга серверов и управления ими. Среди множества различных решений для анализа были выделены следующие:

— соединение по SSH — подход, не требующий наличия промежуточного сервера и позволяющий использовать все возможности операционной системы. Для получения доступа к командной оболочке системы используется протокол SSH, который поддерживается всеми современными серверными операционными системами;
— Zabbix— программное обеспечение с открытым исходным кодом для мониторинга для различных IT-компонентов. Zabbix отслеживает и отображает различные метрики, такие как использование сети, нагрузку процессора, потребление дискового пространства и др. Настройка мониторинга Zabbix может быть выполнена с использованием шаблонов на основе XML, которые содержат элементы для мониторинга;
— Amazon CloudWatch — это сервис мониторинга и управления серверами на площадке Amazon Web Services (AWS). Система предоставляет данные и аналитические сведения для мониторинга приложений, реагирования на изменения производительности в масштабах системы, оптимизации использования ресурсов и получения единого представления о работоспособности системы. CloudWatch собирает данные мониторинга и операционные данные в виде журналов, метрик и событий, помогая получить единое представление приложений, сервисов и ресурсов AWS,

работающих на платформе AWS, а также в локальной среде.;

— Site24x7 — ориентирован на управление серверами, что в основном требует мониторинга всей инфраструктуры как на месте, так и за ее пределами. Пакет включает в себя модули мониторинга сети, сервера и приложений. Стандартные задачи мониторинга Site24x7 охватывают более 30 метрик, включая производительность процессора, объем памяти, использование диска, показатели активности сетевого интерфейса и производительность программного обеспечения;
— Server Density специализируется на мониторинге облачных серверов, но также охватывает локальные серверы. Инструмент специализируется на защите от DDoS-атак и брандмауэрах веб-приложений.;
— Anturis — веб сервис для мониторинга и управления сервером. У компании много ценовых точек в зависимости от количества необходимых метрик. Система Anturis может контролировать любой сервер под управлением Windows или Linux, а также может отслеживать облачные серверы, такие как AWS, Azure и Rackspace.

Инструменты для разработки сетевой системы мониторинга и управления серверами. Данная глава разделена на три секции:

— инструменты для разработки веб-приложений;
— средства управления доступом;
— автоматизация процессов.

Поскольку основными архитектурными компонентами веб-приложения являются клиентская и серверная стороны, обзор инструментов разработки веб-приложений включает в себя:

— HTML, CSS, JavaScript;
— современные фреймворки и библиотеки для разработки клиентских интерфейсов;
— средства реализации адаптивного дизайна;
— языки и фреймворки для разработки серверной части приложений;
— шаблон MVC.

В секции средств управления доступом были рассмотрены элементы данного процесса, а именно:

— логическое управление доступом и его назначение;

— отношение «субъект-объект»;
— матрица доступа.

В секции автоматизации процессов выявлены процессы, сопутствующие процессу разработки ПО, которые требуют частого выполнения и, следовательно, могут быть автоматизированы. К таким процессам относятся:
— интеграция;
— тестирование и в частности подготовка тестовых данных;
— сборка артефакта.

Также были рассмотрены существующие платформы, которые используются для запуска автоматизированных сценариев согласно настроенным параметрам. Для рассмотрения были выбраны следующие платформы:
— Travis CI — сервис непрерывной интеграции, который используется для тестирования и сборки артефактов из репозиториев, размещенных в GitHub и Bitbucket. При каждом обновлении репозитория запускаются команды описанные в файле `.travis.yml`. Результат каждой сборки отображается в панели репозитория, с ссылкой на полный отчет действий. На странице Travis доступны результаты предыдущих сборок, а также время затраченное на выполнение;
— GitLab CI — встроенный механизм, который имеет полную интеграцию с другими компонентами системы и автоматически включен в сборку, при размещении gitlab на внутренних серверах;
— Jenkins — современная программная система, предназначенная, обеспечивающая процесс непрерывной интеграции программного обеспечения. Её преимуществами является отсутствие привязки к платформе, возможность запуска задач по различным критериям (запуск по расписанию, внешний вызов, webhook и т.д.), встроенное хранилище параметров доступа и возможность их использования в различных проектах, наличие множества готовых плагинов, которые предоставляют элегантные решения для подавляющего набора функционала.

Разработка сетевой системы мониторинга и управления серверами. Данная глава полностью описывает процесс разработку сетевой системы мониторинга и управления серверами и имеет следующие разделы:
— проектирование системы;

— реализация серверной части;
— реализация агента;
— реализация клиентской части;
— реализация управления доступом;
— непрерывная интеграция.

В разделе, посвященном проектированию системы, описан протокол двустороннего соединение WebSocket и показано каким образом данный протокол может быть применен. Его использование позволяет обеспечить быстрое и безопасное соединение с удаленными устройствами, которое остается открытым на протяжении всего сеанса работы приложений.

Данный раздел также содержит информацию о настройке WebSocket на уже работающих веб-серверах.

Также в данном разделе была продемонстрирована схема передачи данных между сервером, отслеживаемом устройством и пользователем, основными этапами которой являются идентификация устройства и пользователя с помощью REST и открытие WebSocket соединения для последующей передачи требуемой информации.

В конце раздела резюмированы достоинства использования протокола WebSocket, а так же описано решение использования терминала операционной системы как способа управления, поскольку он имеет огромную поддержку на серверах и не требует дополнительных знаний со стороны администратора системы.

В разделе «Реализация серверной части» описаны используемые технологии для реализации серверной части приложения и обеспечения инфраструктуры, а также показано почему именно они были выбраны.

Также в данном разделе описывается сам процесс реализации и продемонстрирован исходный код отдельно взятых классов и методов. Помимо этого раздел содержит требования к запуску приложения и необходимые инструкции.

В следующем разделе, посвященном реализации агентского приложение, также описаны используемые технологии. Помимо этого в нем отдельное внимание уделено механизмам передачи данных с помощью протокола WebSocket и сбору данных о состоянии системы, которые сопровождаются примерами.

Раздел «Реализация клиентской части» содержит описание функционала, реализуемого клиентским приложением. Так же как и в предыдущих разделах имеется перечень используемых технологий. Данный раздел помимо исходного кода сопровождается изображениями, иллюстрирующими внешний вид интерфейса приложения с учетом различных условий:

— различные экраны приложений;
— отклик приложения на внешние воздействия;
— отображение интерфейса приложения на различных размерах окна, что показывает изменение общего вида приложения при открытии с различных типов устройств.

Для обеспечения корректного отображения интерфейса приложения на экранах различных устройств были выделены диапазоны ширины экрана соответствующие смартфонам, планшетам и персональным компьютерам.

Раздел «Реализация управления доступом» содержит описание внедренных методов распределения доступов. В нем выделены пользовательские права, согласно возможному объекту воздействия.

Помимо этого в разделе демонстрируется как именно права могут быть добавлены в систему и как интерфейс системы может изменяться в зависимости от наличия тех или иных прав.

Проверка прав пользователя производится сразу в нескольких компонентах системы, что обеспечивает большую защиту контроля доступов.

Последный раздел данной главы посвящен автоматизации процессов, не относящихся к написанию кода, но требующих неменьше времени при разработке приложения, а именно тестирования и обновления.

Описан процесс тестирования серверной части приложения и продемонстрирован исходный код тестов и вспомогательных классов.

Перед описанием процесса обновления демонстрируется способ развертывания приложения. Для этого используется механизм контейнеризации и демонстрируются описания контейнеров серверной и клиентской части, которые содержат файл, описывающий каждый слой контейнера, цель каждого контейнера и файл, показывающий как именно контейнеры совместно запускаются и обмениваются информацией.

Далее данный раздел содержит механизм обновления приложения, а именно:

— условия сборки новой версии;
— описание процесса получения исходного кода на платформе Jenkins;
— процесс сборки контейнеров;
— тестирование приложения;
— сборку артефакта;
— отправку артефакта на сервер;
— развертку артефакта на сервере.

Данный механизм сопровождается изображениями и примерами исходного кода конфигураций.

ЗАКЛЮЧЕНИЕ

С увеличением количества устройств с возможностью выхода в глобальную сеть, стремительно увеличилось число пользователей, которым требуется провести диагностику того или иного устройства удаленно. Наиболее частыми случаями возникновения данной проблема являются ситуации, когда физического доступа к системе нет или он сложен в осуществлении.

В настоящее время существуют различные подходы, для того чтобы обеспечить возможность мониторинга состояния системы. Однако можно выделить следующие недостатки:

— отсутствует необходимая гибкость при эксплуатации;
— понижается надежность системы, в следствие появления дополнительных уязвимостей (со стороны кибератак);
— высокая стоимость решения.

В качестве альтернативного решения было рассмотрено сетевое взаимодействие с использованием протокола WebSocket. Это позволяет ускорить работу системы, не нарушая требований к безопасности. Также такой подход прост в конфигурации, так как несильно отличается от стандартного взаимодействия по сети.

Это решение было использовано при разработке собственной системы, которая предоставляет удобный веб-интерфейс для отслеживания состояния серверов в режиме реального времени с любого устройства. Также приложение позволяет передавать любые команды в оболочку ОС сервера, что обеспечивает механизм удаленного управления состояния сервера.