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

Кафедра теоретических основ компьютерной безопасности и криптографии

Отслеживание ресурсов и состояния систем

АВТОРЕФЕРАТ
дипломной работы

студентки 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Чаркиной Елены Владимировны

Научный руководитель
ассистент _____ А. А. Лобов
22.01.2022 г.

Заведующий кафедрой
д. ф.-м. н., доцент _____ М. Б. Абросимов
22.01.2022 г.

Саратов 2022

ВВЕДЕНИЕ

В настоящее время объем мирового рынка серверов стремительно растет. По словам исследователей, рост продаж серверов по итогам 2020 года был ожидаем, учитывая влияние пандемии — из-за нее люди стали чаще работать и учиться из дома, что привело к всплеску спроса на облачные сервисы, а владельцы последних вынуждены были наращивать закупки серверов, чтобы справиться с наплывом пользователей. Это, в свою очередь, привело к расширению сетей дата-центров крупнейшими операторами.

Еще одним катализатором роста рынка эксперты считают постоянно увеличивающийся спрос на технологии безопасности и хранения данных в ключевых отраслях.

Многие крупные компании имеют сотни тысяч серверов, поэтому использование специальных технологий для мониторинга становится необходимым. В данной работе подробно изучены способы и принципы отслеживания ресурсов и состояния систем, современные системы мониторинга, а также реализована собственная программа для мониторинга работы серверов.

Дипломная работа состоит из введения, 7 разделов, заключения, списка использованных источников и 1 приложения. Общий объем работы – 89 страниц, из них 40 страниц – основное содержание, включая 11 рисунков, список использованных источников из 21 наименования.

КРАТКОЕ СОДЕРЖАНИЕ

1. Основные определения

Мониторинг инфраструктуры – это сбор метрик, описывающих состояние IT-инфраструктуры.

Метрика – это конкретная характеристика, показывающая текущее состояние по определенному параметру (пример: количество одновременных подключений).

Сервер мониторинга – ПО, которое выполняет агрегацию, хранение, пересчет данных по метрикам.

Агент мониторинга – ПО, которое выполняет сбор метрик с хоста, который подлежит мониторингу, а также отправку данных на сервер мониторинга.

2. Требования к системе мониторинга

Современная система мониторинга серверов должна предоставлять следующие возможности: мониторинг нескольких серверов, мониторинг диапазона показателей сервера, мониторинг приложений, автоматическое предупреждение о проблемах, запуск действий в ответ на предупреждения, сбор исторических данных о состоянии и поведении сервера и устройства, отображение данных, отчеты, простая конфигурируемость, масштабируемость.

3. Методологии сбора метрик

Существует три основных подхода к сбору метрик: USE (Tom Wilkie), RED (Brendan Gregg), LTES (Google SRE). Они отличаются по целям мониторинга и, естественно, включают разный набор метрик.

В методологии USE для каждого ресурса (CPU, дисковая подсистема, память и т.д.) рекомендуется снимать следующие метрики:

- Utilization — время или процент использования ресурса, занятого «полезной работой»;
- Saturation — насыщенность, то есть количество отложенной или поставленной в очередь «работы»;
- Errors — количество ошибок в работе компонента.

RED предлагает мониторить:

- Rate — количество запросов в единицу времени (например, rps на микросервис или сервер);
- Errors — количество ошибок;
- Duration (оно же latency) — время обработки одного запроса.

В LTES по аналогии с двумя предыдущими методологиями отслеживают:

- Latency — время на обработку одного запроса (с разделением на успешные и ошибочные запросы);
- Traffic — количество запросов к компоненту (для веб-сервера это могут http-запросы, для базы данных — транзакции и т.п.);
- Errors — количество ошибок;
- Saturation — здесь это количественная метрика, отражающая, насколько компонент использует свои ресурсы и сколько у него «работы в очереди».

4. Концепция алертинга

Построение алертинга — системы автоматического оповещения о том, что метрика достигла порогового значения, — основывается на концепциях SLI, SLA и SLO.

SLI (Service level indicators) — набор ключевых метрик, по которым можно определить жизненный статус сервиса, его производительность, «удовлетворенность» конечных пользователей работой сервиса.

SLA (Service level agreement) — так называемое «соглашение об уровне доступности сервиса», которое определяется как внешнее обязательство перед конечным пользователем или клиентом.

SLO (Service level objectives) — набор целевых, «желаемых» значений SLI, выход за пределы которых может привести к нарушению SLA конкретного сервиса или компонента.

5. Библиотека psutil

psutil (process and system utilities) — это кроссплатформенная библиотека для получения информации о запущенных процессах и использовании системы (ЦП, памяти, дисков, сети, датчиков) в Python. Это полезно в основном

для системного мониторинга, профилирования , ограничения ресурсов процессов и управления запущенными процессами . При реализации собственной системы мониторинга была использована библиотека gopsutil, которая является реализацией библиотеки psutil для языка Golang.

Далее будут рассмотрены наиболее полезные функции данной библиотеки.

5.1 Получение информации о ЦП

psutil.cpu_times(percpu=False)

Возвращает системное время ЦП.

psutil.cpu_percent(interval=None, percpu=False)

Возвращает текущую загрузку ЦП в процентах.

psutil.cpu_count(logical=True)

Возвращает количество логических процессоров в системе или None.

psutil.getloadavg()

Возвращает среднюю загрузку системы за последние 1, 5 и 15 минут.

5.2 Получение информации о памяти

psutil.virtual_memory()

Предоставляет информацию об общем количестве памяти, количестве используемой памяти, количестве неиспользуемой памяти, информацию о кэше и т.д..

5.3 Получение информации о диске

psutil.disk_partitions(all=False)

Возвращает информацию обо всех смонтированных разделах диска.

psutil.disk_usage(path)

Возвращает статистику использования диска для раздела, содержащего указанный путь.

psutil.disk_io_counters(perdisk=False, nowrap=True)

Возвращает общесистемную статистику дискового ввода-вывода.

5.4 Получение информации о сети

psutil.net_io_counters(pernic=False, nowrap=True)

Возвращает общесистемную статистику сетевого ввода-вывода.

psutil.net_connections(kind=«inet»)

Возвращает общесистемные соединения сокетов.

psutil.net_if_addrs()

Возвращает адреса, связанные с каждой сетевой картой, установленной в системе.

psutil.net_if_stats()

Возвращает информацию о каждой сетевой карте, установленной в системе.

5.5 Получение информации о процессах

psutil.pids()

Возвращает отсортированный список PID текущих запущенных процессов.

psutil.process_iter(attrs=None, ad_value=None)

Возвращает итератор, возвращающий экземпляр класса Process для всех запущенных процессов на локальном компьютере.

psutil.pid_exists(pid)

Проверяет, существует ли данный PID в текущем списке процессов.

6. Обзор существующих систем мониторинга

В данном разделе будут рассмотрены функциональность и интерфейс существующих систем мониторинга.

6.1 Zabbix

Zabbix — свободная система мониторинга и отслеживания статусов сервисов компьютерной сети, серверов и сетевого оборудования. Основные особенности: распределённый мониторинг, автоматическое обнаружение, централизованный мониторинг журналов, веб-интерфейс для администрирования и настройки, отчётность и тенденции, расширение за счёт выполнения внешних скриптов, гибкая система шаблонов и групп.

6.2 Prometheus

Prometheus — система мониторинга различных систем и сервисов. Это популярный проект с открытым исходным кодом, большая часть компонентов которого написана на Golang. Основные особенности: автоматическое

обнаружение сервисов, представление данных в виде временных рядов, гибкий язык запросов, метки времени имеют точность до миллисекунд. На сегодняшний день это базовая система в таких проектах, как Docker и Boxever.

6.3 Nagios

Программное решение для мониторинга компьютерных систем и сетей Nagios способно осуществлять мониторинг практически любых компонентов, включая сетевые протоколы, операционные системы, системные показатели, приложения, службы, веб-сервера, веб-сайты, связующее программное обеспечение (Middleware) и т. д.. Основные особенности: автоматический перезапуск приложений, многопользовательский доступ, ограниченный доступ, сообщество более 1 млн. активных пользователей, расширяемая архитектура.

7. Реализация системы мониторинга

В ходе данной работы была реализована программа для мониторинга работы серверов, состоящая из трех компонентов:

- Сервер мониторинга — программа, написанная на языке программирования Golang, использующая в качестве базы данных колоночную аналитическую СУБД Clickhouse и резидентную СУБД Redis в качестве кэша, которая отвечает за получение данных от агентов;
- Агент мониторинга — программа, так же написанная на языке программирования Golang, которая устанавливается на сервер и отправляет статистику Серверу мониторинга;
- Клиентский модуль — программа, предоставляющая веб–интерфейс для просмотра полученных данных, а также для наблюдения за состоянием серверов в реальном времени. Данный модуль написан с использованием следующих технологий: HTML, CSS, TypeScript, React — JavaScript библиотека для создания пользовательских интерфейсов.

Доступ к системе реализован через аутентификацию с использованием токенов JWT.

Ниже подробно рассматривается каждый компонент данной системы, а также механизм аутентификации.

7.1 Механизм аутентификации

В данной работе для доступа к системе мониторинга реализован механизм аутентификации с использованием токенов JWT. После первого входа, клиенту возвращается сгенерированный сервером токен JWT. При каждом следующем запросе клиент должен передавать JWT установленным API способом (например, через заголовок или как параметр запроса). Сервер декодирует header и payload и проверяет зарезервированные поля. Если все в порядке, по указанному в header алгоритму составляется подпись. Если полученная подпись совпадает с переданной, пользователя авторизуют.

Токены обновления пользователей сохраняются в кэше Redis.

7.2 Сбор и хранение данных

Сбор данных осуществляется Сервером мониторинга. В базе данных хранится список всех агентов мониторинга. При старте программы Сервер мониторинга получает данный список и запускает сбор данных от агентов.

Сервер собирает следующие метрики: информация об оперативной памяти, информация о процессоре, информация о диске, информация о хосте. Данные между Сервером мониторинга и Агентами мониторинга передаются в формате JSON и сохраняются в базу данных ClickHouse.

7.3 Пользовательский интерфейс

Веб-интерфейс реализован с помощью JavaScript-библиотеки React. Пользователь видит список всех агентов, в таблице выводятся состояние, название и адрес агента.

При выборе конкретного агента открывается окно просмотра информации. Пользователь может менять данные агента, а также останавливать и запускать его. Интерфейс позволяет добавлять новых агентов и удалять существующих. Область с информацией об агенте включает графики и текстовую информацию. С помощью фильтров можно задать интервал времени для отображения статистики.

Сервер мониторинга присылает мгновенные оповещения, если какой-либо агент перестает отвечать на запросы. Оповещения отображаются во

всплывающем окне. Также пользователь может посмотреть список всех пришедших ошибок.

На рисунке 1 представлен пользовательский интерфейс.

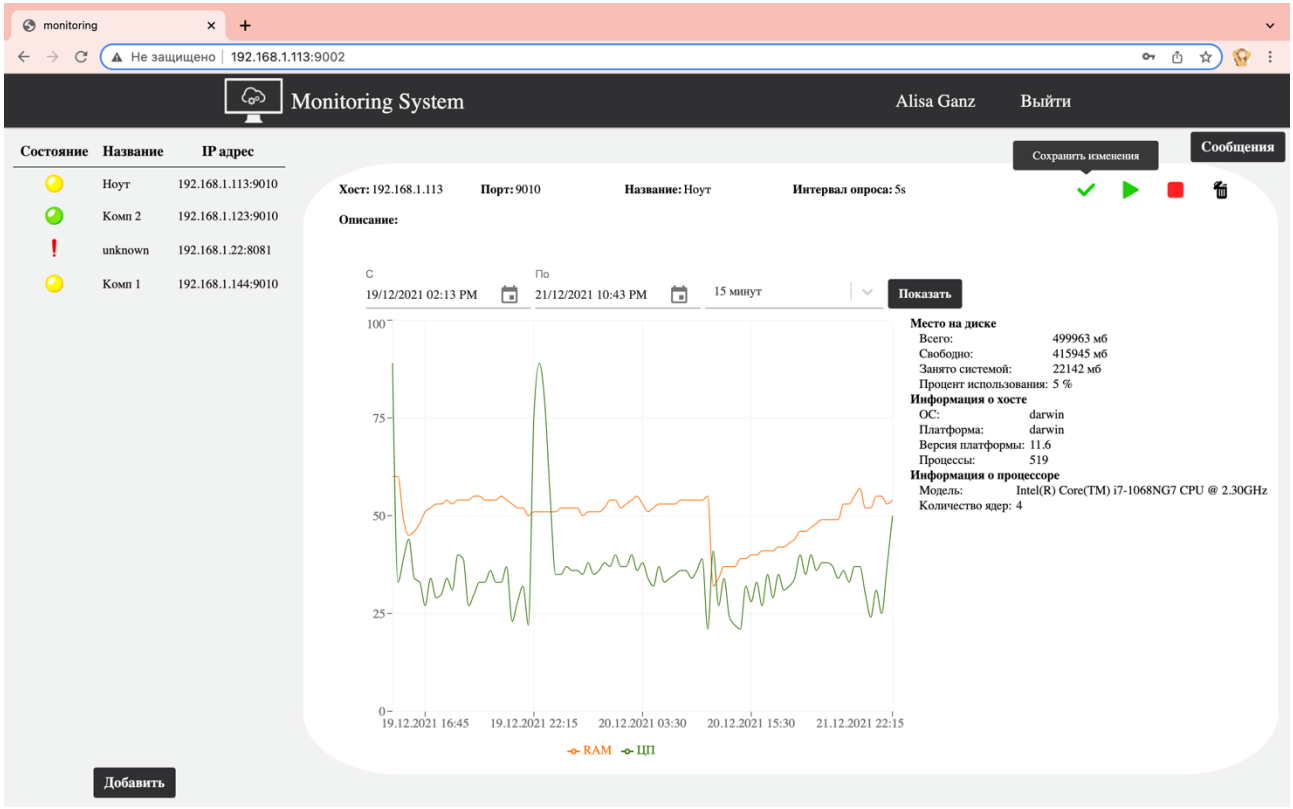

Рисунок 1 — Пользовательский интерфейс

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены способы отслеживания ресурсов и состояния систем, принципы организации систем мониторинга и самые популярные системы мониторинга. Также были реализованы следующие программы: Агент мониторинга для сбора статистики на сервере, сам Сервер мониторинга для получения и обработки данных от агентов и веб-приложение для управления и настройки агентов, а также просмотра полученных данных, отображения статистики и отслеживания работоспособности серверов.

Реализованный комплекс позволяет отслеживать состояние ЦП, памяти и диска, а также получать данные о хостах. Его можно использовать для отслеживания состояния и загруженности серверов, получения своевременных оповещений об отказах и ошибках. На основании полученной от комплекса информации возможно принятие решений об обновлении или добавлении оборудования, масштабировании сервисов и детектировании DDOS атак.