Разработка программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения

Е.Е. Истратова

***Аннотация* — В статье представлены результаты разработки программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения. Для реализации указанной задачи был проведен анализ предметной области, выполнено исследование существующих аналогов, определена архитектура программного обеспечения, разработан алгоритм работы, на основе которого было реализовано программное обеспечение. В результате выполнения работы была спроектирована и настроена система для хранения и визуализации метрик, которая в сочетании с разработанным модулем для сбора и передачи метрик представляет собой готовое программное обеспечение для мониторинга параметров сетей связи и серверного аппаратного обеспечения. Программное обеспечение было написано на функциональном языке программирования Erlang с применением программной системы для визуализации данных Grafana. Ключевыми функциями программы являются: соединение с сервером; получение метрик в режиме слежения; временное хранение метрик; передача метрик в систему мониторинга; визуализация метрик. Готовый программный продукт может быть использован в учебных и научно-исследовательских целях, а также в промышленности для повышения эффективности процессов поддержки и тестирования различных сетевых систем и сервисов. Практическая значимость разработанной программы заключается в возможности ее применения как для автоматизации сбора и хранения метрик, так и для повышения эффективности процессов поддержки и тестирования программных продуктов.**

***Ключевые слова* — программное обеспечение, мониторинг, серверное оборудование, сети связи, метрики, анализ.**

I. Введение

Современные корпоративные сети должны поддерживать активный рост объема трафика и подключенных устройств, чтобы обеспечить возможности для доступа и обмена информацией в сети. Причем масштаб развития сетей и высокая степень неопределенности усложняют такие задачи управления трафиком, как: управление перегрузками, прогнозирование трафика, классификация и маршрутизация, а также подверженность сбоям и атакам. Несмотря на активный рост и развитие масштабов современных корпоративных сетей, вычислительные ресурсы зачастую остаются ограниченными, из-за чего время от времени специалисты в работе сталкиваются с такими проблемами, как несвоевременное реагирование, связанное с необходимостью быстрой диагностики, а также отсутствие стабильности в работе сетей и информационных систем за счет увеличения перерывов в обслуживании. При этом применение инструментов для мониторинга означает, что специалисты по обслуживанию сетей могут оперативно находить неисправности и предотвращать возникновение нештатных ситуаций. Именно поэтому применение инструментов мониторинга для последующего сбора и анализа различных характеристик систем является актуальным направлением развития современных информационных технологий. Целью работы являлось проектирование программного обеспечения для автоматизации процессов сбора, хранения и визуализации метрик с удаленных серверов для их последующего анализа пользователем.

II. Анализ программного обеспечения для сбора и визуализации метрик

В настоящее время существует несколько подходов, основанных на применении различных инструментов для мониторинга параметров сетей связи и серверного аппаратного обеспечения. Так, в литературных источниках [1,2] описана технология использования облачных систем мониторинга.

Облачные вычисления — неотъемлемая часть современного мира, что объясняется постоянно увеличивающимся объемом вычислений, а также различными требованиями к ресурсам, размещаемым в крупных центрах обработки данных. Различия между вычислительными задачами как в требованиях к ресурсам, так и во времени обработки позволяют оптимизировать использование физического оборудования за счет применения облачных технологий.

В работе [1] показана разработка прототипа системы для управления виртуальными машинами на основе нагрузки в вычислительном кластере OpenStack. В основе предложенной идеи лежит идея «упаковки» простаивающих виртуальных машин в специальные парковые серверы, оптимизированные для этой цели. Оценка эффективности данного метода была доказана при запуске реального программного обеспечения в тестовом кластере OpenStack и моделирования того же принцип с помощью программного обеспечения-симулятора Cloudsim.

В исследовании [2] описана разработка облачной

Истратова Евгения Евгеньевна, Новосибирский государственный технический университет, istratova@mail.ru

системы мониторинга на языке Java, предназначенная для работы на платформе J2EE. Системная архитектура предложенного облачного механизма для анализа и мониторинга производительности включала три ключевых компонента: средство для мониторинга производительности серверных ресурсов, инструментарий для мониторинга систем корпоративных приложений и систему оповещения об аномальных событиях. Механизмы анализа производительности и мониторинга при этом были интегрированы и включали в себя активную систему диагностики и модуль обслуживания для выдачи уведомлений в случае обнаружения неисправностей. Данное решение позволило повысить стабильность работы корпоративной сети, снизить частоту нештатных операций и эксплуатационные расходы на предоставление информационных услуг.

В ряде других публикаций в качестве основной архитектуры программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения предлагается использование клиент-серверной архитектуры [3-5]. В статье [5] приведен пример спроектированной системы удаленного мониторинга параметров сети на основе применения клиент-серверного приложения. Протокол управления передачей данных, основанный на интерфейсе программирования сокетов, используется для обмена между локальным исполнителем и облачным сервером. Архитектура браузер/сервер выбирается для обмена данными между веб-сервером и удаленным клиентом. Взаимодействие данных между веб-сервером и удаленным клиентом осуществляется с помощью Ajax (асинхронный JavaScript и XML).

Современные сетевые коммуникационные технологии нередко используют встроенные системы микроконтроля в сочетании с технологией Интернета вещей. При этом, благодаря совершенствованию аппаратных и программных технологий, базовая работа со связью стала основной частью реализации сетевой связи, однако реальной проблемой часто является передача информации. Например, для оценки и оптимизации беспроводного канала на основе встроенной системы необходимо, насколько это возможно, устранить влияние непреодолимых факторов, рассмотреть дефекты на уровне алгоритма и на уровне логического анализа с момента развития технологии сетевой связи, определить влияние модуля сетевой связи встроенной системы мониторинга на практике насколько это возможно.

В исследовании [6] предложен метод оценивания канала связи на основе изучения сетевой связи встроенных устройств за счет применения алгоритмов LS и MMSE. По результатам экспериментов предложенные алгоритмы оказались эффективнее традиционного подхода по таким показателям, как: оценка средней пропускной способности, задержки передачи, времени отклика, скорости передачи в статической среде.

Часть публикаций посвящена вопросу применения нейронных сетей к задаче мониторинга сетевого трафика. В области управления сетевым трафиком методы машинного обучения используются в основном для прогнозирования, классификации и маршрутизации трафика, которые имеют основополагающее значение для предоставления дифференцированных и приоритетных услуг. В области оптимизации производительности обсуждается применение методов машинного обучения в контексте управления перегрузкой, корреляции QoS/QoE, а также управления ресурсами и неисправностями. Помимо этого, данные методы применяются для обеспечения сетевой безопасности [7].

Несмотря на то, что решения на основе применения методов машинного обучения показали достойные результаты для решения многих задач управления трафиком, их масштабируемость необходимо оценивать с учетом предполагаемого объема данных, количества устройств и приложений. Это объясняется тем, что существующие основанные на машинном обучении подходы к управлению сбоями и безопасностью в основном ориентированы на однопользовательские и одноуровневые сети. Чтобы разработать структуру управления неисправностями и безопасностью для современных гетерогенных сетей, существующие подходы к машинному обучению следует расширить или перестроить, чтобы учесть понятие многопользовательской аренды в многоуровневых сетях.

Метрики программного обеспечения (от англ. software metric) – это численная мера, позволяющая выделить и оценить определенные свойства программного обеспечения, либо его спецификаций [8]. Исходя из определения, применение метрик необходимо для того, чтобы можно было бы количественно оценить те или иные показатели работы программного обеспечения. На их основании можно строить статистические модели и графики, визуализирующие текущее состояние системы, а также предсказывать вероятные сбои и нагрузки. Корректное использование данного инструмента способно значительно упростить процесс поддержки программного продукта, заблаговременно выявить возможные ошибки и ускорить их исправление, а также повысить уровень понимания работы системы для разработчиков [9].

Метрики используются не только для мониторинга систем для разработчиков, но и для анализа экономических и бизнес показателей проекта. Например, в статье [10] описывается использование метрик на разных стадиях развития компании, для того чтобы контролировать ресурсы и оценивать качество продукта по кварталам. Автор также выделяет несколько типов метрик по их назначению, а именно: прогнозирующие – для того, чтобы попытаться предсказать проблемы; диагностические – для мониторинга текущего состояния проекта; ретроспективные – для того, чтобы попытаться избежать неблагоприятных событий, с которыми команда разработки сталкивалась в предыдущих проектах.

Одним из преимуществ применения системы сбора

метрик является ее гибкость. Практически любую исчислимую информацию можно преобразовать в метрику, хотя иногда для этого может потребоваться провести некоторые дополнительные математические вычисления. Например, метриками может быть количество и тип поступающих на сервер запросов, степень загрузки каждого ядра системы, занятая память препроцессора, ошибки при обработке данных. Это позволяет использовать одну систему метрик в различных проектах, просто изменяя типы данных и способы их сбора. Особенно актуальным это свойство становится при работе с комплексными серверными системами, состоящими из нескольких программных продуктов предназначенными для решения различных задач [11].

Однако использование метрик сопряжено с определенными сложностями. Во-первых, ни одна из известных на сегодняшний день метрик не может обладать достаточной значимостью и точностью. Во-вторых, метрики не дают объективной и полной картины происходящего, они всего лишь предоставляют некоторые показатели, вычисленные по заданному алгоритму. Особенно это касается тех случаев, когда для оценки какого-либо параметра вводится только одна метрика. Более-менее полную картину может дать только использование целого комплекса метрик.

В статье [12] приводится описание основ мониторинга и различных типов объектов мониторинга с соответствующими им метриками. Согласно статье, выделяют три уровня мониторинга. Мониторинг оборудования – самый низкий уровень мониторинга, он включает в себя наблюдение за такими параметрами, как нагрузка на сеть и процессор, количество производимых операций и запущенных задач, занятое этими процессами место в ОЗУ и другие параметры, связанные с физическим оборудованием [13-15].

На следующем уровне размещен мониторинг приложений, где параметры находятся от них в зависимости, к ним относятся: количество запросов к серверу, число новых записей в базе данных, количество ошибок в системе за последний час, число активных пользователей. Именно этот тип метрик является самым важным с точки зрения программистов, так как он позволяет оценить состояние самого программного обеспечения. Часто при незначительных нагрузках на оборудование система может работать некорректно, что не будет отображено на уровне мониторинга оборудования, но уровень мониторинга приложений покажет данные проблемы [16].

Следующим является мониторинг бизнес-метрики, который предназначен не для оценки состояния системы, а для получения понимания об экономических показателях. Примеры таких метрик: общая выручка и прибыль от приложения; особенности использования приложения пользователями; процент пользователей, которые дошли до покупки товаров в приложении; количество недавно зарегистрированных пользователей. Эти метрики позволяют руководителям верно корректировать курс развития проекта, чтобы максимизировать получаемую от него выгоду и удовлетворить клиентов [17,18].

Для корректного составления метрики необходимо в ней указать следующие параметры:

1. Цель измерения. Показатель метрики должен быть связан с бизнес целью, то есть нужно дать краткую и конкретную цель создания данного измерения.

2. Для кого метрика предназначается. Нужно точно знать, кто будет получателем данной метрики, им может быть менеджер проекта, разработчик, тестировщик, причем у каждого из них свои цели и задачи, значит измерение должно соответствовать этим задачам.

3. На какой вопрос пользователя отвечает метрика. Нужно точно знать, чем именно метрика поможет заданному пользователю.

4. Формула расчета. Необходимо ответить на вопрос, как именно метрика будет собираться и обрабатываться для последующего отображения. Например, количество запросов в минуту можно собирать в виде счетчика запросов, а затем делить на время.

5. Периодичность. Отвечает на вопрос, как часто нужно собирать метрику. Так как метрики могут быть очень разными, нужно четко определиться с периодичностью, дабы не создавать дополнительную нагрузку на сервер за счет сбора излишних данных. Например, не имеет смысла часто собирать метрики о проданных за смену продуктах, для этого будет достаточно частоты раз в день или в несколько часов. В то время, как метрику о текущей нагрузке на сервер нужно собирать раз в несколько секунд.

Очевидно, что процесс составления метрики позволяет точно определить ее полезность и ценность для конкретного пользователя, а также для компании в целом [19].

Таким образом, в результате проведения анализа предметной области было установлено, что системы мониторинга имеют высокую ценность для ИТ-компаний, их функционал помогает быстрее определить проблемы, предсказывать будущие изменения, следить за состоянием системы и составлять планы по улучшению ее функционирования. Хотя метрики в данном вопросе и не дают настолько точных данных о функционировании программного обеспечения как трейсы, они являются более гибкими и ориентированными на больший спектр заинтересованных лиц, что делает их относительно простым и незаменимым инструментом для мониторинга. Из рассмотренных литературных источников были получены сведения об основных методах построения систем мониторинга (USE, RED и UCA методы), на основании которых был сделан выбор в пользу применения методов USE и RED, так как они предполагают мониторинг именно состояния системы, в то время как последний метод UCA рассчитан на бизнес метрики, которые не будут использованы в рамках учебного некоммерческого проекта. Также в результате предпроектного исследования была выявлена необходимость разработки time series базы данных для хранения метрик и веб-интерфейса для их визуализации.

III. Выбор программной среды для реализации программного обеспечения

На основании результатов анализа существующих решений в области мониторинга параметров сетей были сделаны выводы о том, что реализация проекта должна включать два основных этапа. Первый из которых связан с проектированием сервера для сбора и отправки собираемых метрик. Предполагается, что данный сервер будет размещен на стороне клиента, что существенно снижает вариативность выбора среды реализации программного обеспечения. Второй этап связан с разработкой системы сбора, хранения и визуализации метрик. Указанная система должна включать два ключевых модуля, отвечающих за хранение данных (база данных) и их отображение (веб-интерфейс). При этом оба компонента системы должны быть совместимы и легко запускаться на стороне клиента.

Программный модуль для сбора и отправки метрик размещен на стороне клиента, то есть фактически на серверном оборудовании. Современные сервера практически всегда работают на базе операционных систем под руководством Linux. Это связано с тем, что Linux-подобные системы обладают рядом преимуществ по сравнению с Windows при работе в качестве сервера.

Одно из таких преимуществ – простота и прозрачность файловой системы. Принцип организации дискового пространства и работы с файлами отличается от представленного в Windows. Благодаря этому, подобная система обеспечивает больший контроль и гибкую настройку прав пользователей, что очень важно, когда с одной системой работает множество пользователей. За счет этой особенности Linux обеспечивает дополнительную безопасность и повышенную защиту прав доступа к хранящейся на сервере информации.

Кроме того, серверные Linux-системы являются очень легковесными и производительными, что позволяет реализовать их на различных аппаратных ресурсах. С такими системами часто работают без интерфейса, посредством удаленного подключения по консоли. За счет своих специфичных особенностей, данные системы стали очень популярны для использования на серверах, рабочих станциях и микрокомпьютерах, и для них было специально написано огромное количество программного обеспечения, связанного с типичными требованиями соответствующих сфер применения.

Таким образом, наиболее оптимальным вариантом при выборе операционной системы для реализации программного обеспечения является Linux. В данной работе была использована версия операционной системы Ubuntu 20.04, так как она является наиболее стабильной, поддерживает большое количество пакетов программ и обладает пользовательским интерфейсом, что положительно влияет на удобство и скорость разработки.

Разрабатываемое программного обеспечение для мониторинга предназначается для диагностики систем с высокой нагрузкой, а также серверов, работающих в сфере телекоммуникаций и Интернет-технологий. Для организации таких систем лучше всего подходят функциональные языки программирования. Самыми известными представителями данного класса являются: Haskell, Erlang и Clojure. Все перечисленные языки программирования обладают преимуществом, связанным с возможностью решения таких проблем, как: взаимные блокировки и потокобезопасность. Это объясняется тем, что они ориентированы на работу с большим количеством потоков данных, а также просты в тестировании, что является значительным преимуществом при разработке больших проектов.

Функциональные языки программирования — это языки, в которых процессы представлены как математические функции. Таким образом, каждая функция в них определяется не как подпрограмма, а как соответствие между множествами. Функциональный подход — противоположность императивному, в котором программист задает программе четкий порядок действий по шагам. При этом программа решает, как и в каком порядке исполнять действия, а программист описывает правила взаимодействия и связи между компонентами. Благодаря своим особенностям, функциональное программирование применяется при работе с большими объемами данных, а также для решения задач, связанных с выполнением сложных вычислений.

В ходе исследования трех наиболее популярных функциональных языков программирования, было установлено, что язык Clojure является новым и понятным современному пользователю, однако в России он практически не используется крупными компаниями и, соответственно, не очень популярен. Несмотря на то, что язык Haskell обладает значительно большей популярностью чем Erlang и Clojure, его область применения в основном сосредоточена на скриптах, инструментах обработки текста и фильтрующих системах, что не подходит под задачу мониторинга, так как в ней подразумевается отслеживание именно состояния серверов. Таким образом, в результате исследования было решено выбрать в качестве языка реализации проекта Erlang. Такой выбор был связан с тем, что данный язык изначально разрабатывался для работы с телекоммуникациями и многопоточными серверными процессами, и поэтому идеально подходит под область применения разрабатываемого программного обеспечения для мониторинга.

Erlang — функциональный язык программирования с сильной динамической типизацией, который отлично подходит для разработки распределенных вычислительных систем. Erlang был разработан компанией Ericsson в 1987 году для использования в телекомуникационных системах. В экосистеме Erlang есть в наличии стандартная библиотека модулей и библиотека шаблонных решений — фреймворк OTP (Open Telecom Platform). Программы на Erlang компилируются в байт-код, исполняемые виртуальной машиной BEAM, причем виртуальных машин может быть несколько на различных узлах распределенной вычислительной сети. Erlang-системы поддерживают горячую замену кода, что исключает остановку системы

при установке обновлений. В последние годы Erlang используется не только в телекоммуникационных системах, но и при разработке высоконагруженных распределенных веб-приложений. Например, чат Facebook и WhatsApp написаны именно на Erlang. Также Erlang применяется в нескольких NoSQL-базах данных высокой доступности.

Таким образом, именно язык Erlang является наиболее подходящим вариантом для разработки библиотеки для работы с метрическими данными, а также для проектирования сервера сбора и отправки метрик.

Базы данных временных рядов представляют собой системы баз данных, специально предназначенные для обработки информации, связанной со временем. Основным отличием данного вида от классических реляционных баз данных является наличие временной метки. Поэтому при добавлении новых записей в реляционную базу данных и при наличии в таблице индексов система управления базами данных будет многократно переиндексировать данные для быстрого и эффективного доступа к ним. Как следствие, производительность со временем снижается. При этом увеличивается нагрузка, что приводит к трудностям при чтении данных. База данных временных рядов оптимизирована для быстрого приема данных. Такие системы используют индексацию данных, объединенных со временем. Как следствие, в подобных системах скорость загрузки не уменьшается со временем и остается достаточно стабильной.

Еще одной особенностью баз данных временных рядов является возможность их совместного применения с рядом других приложений, среди которых могут быть как приложения-источники данных, так и приложения-потребители. Например, в данном проекте в качестве источника данных будет выступать сервер, состояние которого необходимо периодически исследовать, а потребителем данных будет являться сервис визуализации метрик. В первой главе работы были рассмотрены различные базы данных временных рядов и визуализаторы, среди них был проведен сравнительный анализ. В результате было решено использовать для реализации текущего проекта базу данных Prometheus. Для визуализации данных из базы данных был выбран веб-сервис Grafana.

Для удобства и быстроты запуска сервиса, необходимо использовать среду виртуализации. Это позволит легко объединить базу данных и сервис для визуализации метрик вместе со всеми необходимыми для работы настройками. Поместив оба компонента в виртуальную среду, будет возможно запустить их на любом устройстве одной командой, как если бы это было единое приложение. Чтобы определиться с выбором среды виртуализации, необходимо предварительно определить принцип работы подобного сервиса.

Контейнеры и виртуальные машины – это технологии, которые делают приложения независимыми от ресурсов ИТ-инфраструктуры. Контейнер – это пакет программного кода, содержащий код приложения, его библиотеки и другие компоненты. Контейнеризация делает приложения переносимыми, благодаря чему один и тот же код может работать на любом устройстве. Виртуальная машина – это цифровая копия физической машины. И контейнеры и виртуальные машины являются технологиями механизма развертывания, который обеспечивает эффективное выполнение приложения на сервере или устройстве. Приложению требуются некоторые дополнительные программные компоненты, называемые зависимостями, которые тесно связаны с базовой операционной системой сервера. Все эти уровни программного обеспечения между кодом приложения и физическим устройством называются средой приложения.

Технология виртуальных машин разработана для эффективного использования растущего объема оборудования и вычислительных мощностей. Применение одной среды приложения на отдельном физическом сервере означает недостаточное использование аппаратных ресурсов. Виртуальные машины позволяют устанавливать несколько сред на одной и той же физической машине.

Контейнеры созданы для упаковки приложений и их прогнозируемого и воспроизводимого выполнения в нескольких средах. Вместо того, чтобы создавать среду заново, можно упаковать приложение, чтобы затем запускать его во всех типах физических или виртуальных сред.

Контейнеры и виртуальные машины позволяют упаковать инфраструктуру программного обеспечения в отдельный файл, который называется файлом образа. Контейнеры виртуализируют операционную систему, чтобы приложение могло независимо выполняться на любой платформе. Виртуальные машины выходят за эти пределы: они виртуализируют физические машины для эффективного использования аппаратных ресурсов.

Ключевой задачей в работе было создание виртуального образа, который включает в себя базу данных Prometheus, визуализатор данных Grafana, а также все необходимые для их работы и взаимодействия зависимости. Так как разрабатываемое приложение должно быть портативным и легковесным с возможностью запуска одной командой, то выбор между виртуальной машиной и виртуальным контейнером был сделан в пользу именно контейнера. Это объясняется тем, что контейнеры намного меньше весят, потребляют очень мало ресурсов системы сами по себе, просты в запуске и распространении, а также не содержат в себе ничего лишнего.

Во всей индустрии приложений для контейнеризации приняты единые стандарты, вследствие чего для запуска контейнеров применяется один и тот же механизм, поэтому контейнеры любого типа совместимы с любой системой управления кластерами контейнеров. В связи с этим, в работе был использован наиболее популярный инструмент для работы с контейнерами – Docker, в основе работы которого лежит стандартизированный способ исполнения кода. Docker – это операционная система для контейнеров. Подобно тому, как виртуальная машина создает виртуальное

представление аппаратного обеспечения сервера, то есть устраняет необходимость непосредственно управлять таковым, контейнеры создают виртуальное представление серверной операционной системы. После установки на каждый сервер Docker предоставляет доступ к простым командам, необходимым для сборки, запуска или остановки контейнеров.

IV. Реализация программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения

Программный комплекс состоит из трех основных элементов: модуль сбора метрик, база данных временных рядов, веб-интерфейс визуализации метрик. Первым этапом разработки программного обеспечения являлось проектирование модуля для обработки сырых данных. Его основная задача заключалась в преобразовании данных системы в метрики, готовые для использования базой данных. К дополнительному функционалу модуля относится возможность временного хранения метрик перед выгрузкой из базы данных и передачей для формирования отчета.

Разработка модуля для обработки данных включала следующие этапы:

1. Изучение форматов данных, применяемых в Prometheus.
2. Выбор способа временного хранения метрик.
3. Реализация механизма трансформации данных в метрики.

Согласно проведенному анализу литературных источников, Prometheus поддерживает три основных формата данных:

1. Gauge – числовое значение, свободно изменяющееся с течением времени.
2. Counter – счетчик, значение которого увеличивается со временем.
3. Histogram – гистограмма, представляющая собой графическую интерпретацию выбранных наблюдений с возможностью настройки квантилей.

Данные для обработки могут быть получены программой из тела простого TCP-запроса. Благодаря этому, весь набор метрик за текущий период можно отправлять в виде строки, размещаемой в запросе. Разделителем при этом выступает знак переноса строки. Метрики типа Counter и Gauge записываются в произвольном порядке, при том, что сначала идет тип метрики, затем ее название, список тегов и описание, а в конце каждой строки находится само значение метрики в виде числа с плавающей точкой. Метрики типа Histogram имеют более строгую структуру. Для каждого квантиля гистограммы выделяется отдельная строчка, в которой указываются метрики типа Счетчик со значением квантиля. Строки гистограммы располагаются в порядке от наименьшего квантиля к наибольшему. Также гистограмма содержит дополнительные строки с сумой и количеством метрик. Такие особенности связаны с тем, что гистограмма – это комплексная метрика, в состав которой могут входить несколько обычных метрик типа Счётчик.

Для временного хранения метрик было использовано программное обеспечение Erlang Term Storage (ETS), представляющее собой инструмент для хранения объектов Elixir и Erlang в памяти. Отличительной особенностью данного инструмента является его способность хранить значительные объемы данных и предоставлять к ним доступ за фиксированное время. При разработке модуля было использовано понятие регистра – специальной метки, необходимой для разделения данных от разных модулей системы и контроля их отправки. Каждый регистр является таблицей с метриками, регистров может быть несколько. Ключи метрик в таблице каждого регистра представляют собой название метрики. Благодаря этому, поиск любой метрики в базе данных будет занимать фиксированное время. Так как это не база данных временных рядов, то существует ограничение на хранение метрик. Так, предусмотрено хранение только одной метрики каждого типа без сохранения истории.

Для записи данных в таблицы были созданы специальные модули с наборами методов под каждый тип метрик. Разработанные методы выполняют следующий функционал:

1. Declare объявляет новую метрику и добавляет ее в ETS-таблицу.
2. Deregister удаляет метрику.
3. Inc увеличивает значение метрики на указанное значение.
4. Set устанавливает значение метрики равным указанной величине.
5. Reset сбрасывает значение метрики до начального состояния.
6. Value возвращает текущее значение метрики.

При объявлении метрики необходимо задать такие ее параметры, как: название, регистр и перечень тегов. При использовании любых других операций, кроме удаления метрики, применяются эти же параметры. Удаление требует наличия таких параметров, как: название и регистр.

Для вывода метрик в текстовый формат из ETS-таблиц был реализован отдельный метод – Format, который принимает только название регистра. В процессе работы данный метод применяет ко всем элементам таблицы указанного регистра одну функцию свёртывания, которая переводит объекты Erlang в строки, а затем результаты свёртывания могут быть собраны вместе. После выполнения данных действий полученная строка с метриками готова для передачи в базу данных Prometheus.

В результате работы бал спроектирован модуль для обработки метрик с возможностью временного хранения данных, преобразования их в подходящий для Prometheus формат и формирования из них метрических показателей для вывода в виде текста.

В качестве основных функций модуля сбора для сбора и передачи метрик были выделены следующие:

1. Формирование ответов на запросы приложения.
2. Пересылка метрик.
3. Конфигурирование метрик.
4. Вывод информации об источниках метрик.

Для реализации указанных функций был разработан

алгоритм для инициализации и обработки сообщений, основанный на работе стандартного gen-сервера Erlang. Модуль для сбора и передачи метрик является одной из составных частей программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения. При запуске модуля при помощи команды Declare из разработанного модуля для обработки и временного хранения метрик объявляются необходимые для исследования метрики. После этого модуль для сбора и передачи метрик запускает таймер, который будет срабатывать через определенные промежутки времени и присылать сообщения о необходимости сбора метрик. Затем открывается сокет для приема запросов по сети. Через этот сокет с модулем будет связываться база данных и опрашивать его о наличии поступающих новых метрик. В ответ на запрос базы данных модуль отправляет сообщение, в теле которого записаны все собранные метрики, скомпонованные при помощи метода Format из разработанного модуля для обработки и временного хранения метрик.

Модуль для сбора и передачи метрик поддерживает следующие параметры конфигурации:

1. Частота опроса источников метрик.
2. Номер порта для сокета связи с базой данных.
3. Путь, по которому метрики будут передаваться в базу данных.
4. Состояние метрик (вкл/выкл).

Эти параметры заданы заранее и могут изменяться во время работы. Перед запуском модуля для сбора и передачи метрик также объявляется менеджер событий, который реагирует на изменения в файле конфигурации и отправляет сообщение о том, что конфигурация изменилась. При получении такого сообщения модуль полностью выключается и затем запускается заново с новыми параметрами.

Таким образом, модуль для сбора и передачи метрик включает следующие компоненты:

1. Скрипт для запуска приложения.
2. Супервизор, поддерживающий работу сервера и менеджера событий.
3. Файл конфигурации модуля.
4. Файл с программным кодом модуля.
5. Обработчик входящих TCP-сообщений.
6. Менеджер событий.
7. Модуль для обработки и временного хранения метрик.

Заключительными этапами разработки программного обеспечения для мониторинга параметров сетей связи и серверного аппаратного обеспечения стало проектирование модулей, которые будут заниматься хранением и визуализацией данных. В связи с этим, на данном этапе была выполнена подготовка Docker-контейнеров с развернутым сервером для приема, хранения и визуализации метрик. Для реализации указанной задачи были выполнены мероприятия, связанные с проектированием структуры виртуальных контейнеров, подключением графического визуализатора, а также установкой и настройкой базы данных временных рядов.

Развертывание базы данных временных рядов Prometheus было произведено в виртуальном контейнере Docker. Доступ к базе данных осуществлялся по номеру порта 9090, который в дальнейшем был использован при конфигурации веб-приложения.

Структура базы данных позволяет не только сохранять данные, распределенные относительно временного фактора, но и использовать гибкую структуру тегов для выделения и последующей обработки только определенных типов метрик. В результате это дает возможность делить данные по различным параметрам, и для одной метрики строить несколько вариаций применения. Например, если имеется метрика использования ресурсов процессора, то, благодаря тегам, ее можно визуализировать в виде количества ядер, нагрузки на каждое ядро отдельно, общей нагрузки, изменения нагрузки со временем и т.д. Описание служит вспомогательным параметром, позволяющим оператору сервиса определять, какая метрика за что конкретно отвечает.

Итоговая физическая модель данных содержит следующие таблицы: Tag, Metric-tags, Metric, Time, Time series. Таблица Tag включает список всех тэгов метрик, объявленных в системе. Таблица Metric представляет собой перечень объявленных метрик с названиями и описанием, то есть их краткой характеристикой. Таблица Time содержит в себе метки времени сбора метрик, которые являются основным ключом в базах данных временных рядов. Таблица Time-series объединяет в себе сущности тегов, метрик и времени, являясь основной таблицей базы данных. С точки зрения смысла данная таблица представляет собой запись метрики в конкретный момент времени с определенными тэгами. Таблица Metric-tags является вспомогательной и объединяет таблицы Time series и Tag, что необходимо для реализации связи многие-ко-многим.

Последним этапом разработки программного обеспечения стала подготовка виртуального контейнера с веб-сервисом для пользовательского интерфейса системы мониторинга. Он был выполнен при помощи инструмента Grafana. PromQL, представляющего собой функциональный язык запросов, который позволяет запрашивать и агрегировать данные временных рядов. С его помощью веб-интерфейс Grafana может извлекать хранящиеся в базе данных метрики. Причем за счет того, что обмен данными между двумя серверами будет происходить лишь в рамках локальной сети виртуальных контейнеров, то программа обеспечивает достаточно высокий уровень информационной безопасности.

Для подключения к сервису используется простая форма авторизации, данные авторизации задаются на этапе сборки контейнеров воедино, а затем передаются в виртуальную машину. Авторизация производится на основании выделенных ролей. Аккаунт Пользователя позволяет просматривать готовые наборы метрик (дашборды) и непосредственно проводить мониторинг. Дашборды могут быть разбиты по директориям и сохранены на локальной машине.

После выбора дашборда система загрузит заданный набор элементов из JSON-файла. Пользователь может задавать некоторые параметры для упрощения мониторинга, например, частоту обновления графиков, либо частоту опроса базы данных.

Аккаунт Администратора, помимо функции просмотра, также включает возможности редактирования, создания и удаления наборов метрик. Интерфейс добавления новой метрики в дашборд представляет собой форму с выбором названия метрики, тегов, фильтров и различных операций, которые можно применить для преобразования данных.

Реализованная структура виртуальных контейнеров состоит из двух сущностей. Первая сущность – контейнер на базе Ubuntu 20 для развертывания базы данных временных рядов Prometheus. Вторая сущность – контейнер на базе Ubuntu 20 с веб-сервисом Grafana, необходимый для запуска пользовательского интерфейса в браузере. Данные сущности были объединены в систему, которая запускается за счет docker-compose.

Основной скрипт запуска системы monitoring.yml содержит в себе все необходимые параметры для связывания двух сущностей воедино. Директория prometheus при компиляции будет передана в контейнер с базой данных временных рядов. Она содержит в себе файл с настройками для источников приема метрик, сами источники настраиваются в файле hosts, который можно редактировать из консоли. После настройки источники будут отображаться в интерфейсе базы данных Prometheus.

Директория Grafana при компиляции передается в контейнер с веб-сервисом пользовательского интерфейса и содержит поддиректорию dashboards, которая используется для сохранения пользовательских наборов метрик. В этой поддиректории также располагается файл конфигурации с настройками структуры дашбордов, который необходим для того, чтобы виртуальная машина с веб-интерфейсом могла бы определить, где сохранять и откуда загружать пользовательские наборы метрик.

Последний файл в директории grafana – datasource.yaml, содержит в себе конфигурацию источников сбора метрик, где указан IP-адрес и номер порта виртуальной машины с базой данных Prometheus.

Таким образом, в результате выполнения работы была спроектирована и настроена система для хранения и визуализации метрик, которая в сочетании с разработанным модулем для сбора и передачи метрик представляет собой готовое программное обеспечение для мониторинга параметров сетей связи и серверного аппаратного обеспечения [20].

V. Заключение

Актуальность выполненной работы непосредственно связана с эффектом, который оказывают системы мониторинга на производительность компании в целом. Данные программы позволяют оперативно реагировать на возникшую проблему в работе различных информационных сервисов, а также эффективно предотвращать возникновение неполадок. В связи с этим, данный вид программного обеспечения необходим для проведения постоянного мониторинга информационной инфраструктуры, обеспечивающего запуск и последующую корректную работу всех необходимых сервисов и сетевых систем вне зависимости от масштабов управляемой инфраструктуры и размеров предприятия. Все это делает разработку систем данного вида целесообразной.

Практическая значимость разработанной программы заключается в возможности ее применения как для автоматизации сбора и хранения метрик, так и для повышения эффективности процессов поддержки и тестирования программных продуктов. В результате готовый программный продукт может быть использован в научных целях и в промышленности для повышения эффективности процессов поддержки и тестирования программных продуктов.

Библиография

[1] Kommeri J., Niemi T., Nurminen J.K. Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware // Supercomput 73, 1978–2000 (2017). https://doi.org/10.1007/s11227-016-1899-0.

[2] Peng Y., Wu I.C. A cloud-based monitoring system for performance analysis in IoT industry // Supercomput 77, 9266–9289 (2021). https://doi.org/10.1007/s11227-021-03640-8.

[3] Амельченко А.О., Антонянц Е.Н., Истратова Е.Е. Программа для сбора количественных характеристик трафика в корпоративных компьютерных сетях. Свидетельство о регистрации программы для ЭВМ № 2021614232, зарегистрирована 22.03.2021. Заявка № 2021613498 от 22.03.2021.

[4] Истратова Е.Е. Программа для мониторинга характеристик трафика в корпоративных компьютерных сетях. Свидетельство о регистрации программы для ЭВМ № 2021665972, зарегистрирована 06.10.2021. Заявка № 2021665332 от 06.10.2021.

[5] Xiao Y., Zhang H., Yuan C. The Design of an Intelligent High-Speed Loom Industry Interconnection Remote Monitoring System // Wireless Pers Commun 113, 2167–2187 (2020). https://doi.org/10.1007/s11277-020-07317-y.

[6] Wang C. Research and implementation of network communication based on embedded monitoring system // Wireless Netw (2023). https://doi.org/10.1007/s11276-023-03307-7.

[7] Boutaba R., Salahuddin M.A., Limam N. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities // J Internet Serv Appl 9, 16 (2018). https://doi.org/10.1186/s13174-018-0087-2.

[8] Hua X. (2021). Application of computer remote network communication technology // Electronic Components and Information Technology, 5(12), 77–78.

[9] Буренин А.Н. О некоторых принципах управления серверным оборудованием защищенных инфокоммуникационных сетей специального назначения / А.Н. Буренин, К.Е. Легков, М.С. Первов // Наукоемкие технологии в космических исследованиях Земли. — 2018. — № 2. — С. 22-26.

[10] Федоров А.С. Разработка системы мониторинга серверного оборудования / А.С. Федоров, Р.А. Андреев // Экономика и качество систем связи. — 2020. — № 4 (18).— С. 32-42.

[11] Xiaodi Y., Yanjun Z., Jian C., Changyuan Y. (2020). Mobile channel estimation based on decision feedback in vehicle-to-infrastructure visible light communication systems // Optics Communications, 462, 125261.

[12] Симанков В.С. Программное и аппаратное обеспечения подсистем интеллектуального ситуационного центра / В.С. Симанков, В.А. Шарай // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. — 2021. — №3 (286). — С. 63-70.

[13] Воеводин В.А. Об оценке своевременности обмена данными в централизованной системе мониторинга Wi-Fi сетей / В.А. Воеводин, Д.С. Буренок // DIZWW. — 2021. — № 17. — С. 60-65.

[14] Mohanty B., Sahoo H.K., Patnaik B. (2020). Wireless channel estimation and beamforming by using block sparse adaptive filtering // Signal, Image and Video Processing, 15(4), 769.
[15] Shariq M., Singh K., Maurya P.K. (2022). AnonSURP: An anonymous and secure ultralightweight RFID protocol for deployment in internet of vehicles systems // The Journal of Supercomputing, 78(6), 8577–8602.
[16] Макеева О.В. Организация корпоративного сервера на базе Linux / О.В. Макеева, А.В. Франчук, М.Л. Рысин // Инновации и инвестиции. — 2021. — № 10. — С. 71-77.
[17] Веревкин С.А. Организация мониторинга распределенной корпоративной сети с целью ее оптимизации и использования полученных данных для обеспечения информационной безопасности / С.А. Веревкин, Р.А. Юхимук, Д.В. Кудро // Известия ТулГУ. Технические науки. — 2023. — № 2. — С. 188-192.
[18] Линдигрин А.Н. Анализ специфики и проблематики процессов поиска аномалий в сетевых данных / А.Н. Линдигрин // Известия ТулГУ. Технические науки. — 2021. — № 5. — С. 304-309.
[19] Sakthivel S., Vidhya G. (2020). A trust-based access control mechanism for intra-sensor network communication in internet of things // Arabian Journal for Science and Engineering, 46, 3147.
[20] Истратова Е.Е. Программа для мониторинга параметров сетей связи и серверного аппаратного обеспечения. Свидетельство о регистрации программы для ЭВМ № 2023662727, зарегистрирована 13.06.2023. Заявка № 2023662001 от 13.06.2023.

Истратова Евгения Евгеньевна. Новосибирский государственный технический университет, г. Новосибирск, Россия. Кандидат технических наук, доцент кафедры автоматизированных систем управления. Количество печатных работ: 121. Область научных интересов: информационные технологии, информационные сети, системы компьютерного зрения. e-mail: istratova@mail.ru (ответственная за переписку).

Development of software for monitoring the parameters of communication networks and server hardware

E.E. Istratova

***Abstract* — The article presents the results of development of software for monitoring the parameters of communication networks and server hardware. To implement this task, an analysis of the subject area was carried out, a study of existing analogues was carried out, the architecture of the software was determined, an algorithm was developed, on the basis of which the software was implemented. As a result of the work, a system for storing and visualizing metrics was designed and configured, which, in combination with the developed module for collecting and transmitting metrics, is a ready-made software for monitoring the parameters of communication networks and server hardware. The software was written in the Erlang functional programming language using the Grafana data visualization software system. The key functions of the program are: connection with the server; getting metrics in tracking mode; temporary storage of metrics; transfer of metrics to the monitoring system; visualization of metrics. The finished software product can be used for educational and research purposes, as well as in industry to improve the efficiency of support and testing processes for various network systems and services. The practical significance of the developed program lies in the possibility of its application both for automating the collection and storage of metrics, and for improving the efficiency of support and testing of software products.**

***Keywords* — software, monitoring, server equipment, communication networks, metrics, analysis.**

REFERENCES

[1] Kommeri J., Niemi T., Nurminen J.K. Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware // Supercomput 73, 1978–2000 (2017). https://doi.org/10.1007/s11227-016-1899-0.

[2] Peng Y., Wu I.C. A cloud-based monitoring system for performance analysis in IoT industry // Supercomput 77, 9266–9289 (2021). https://doi.org/10.1007/s11227-021-03640-8.

[3] Amelchenko A.O., Antonyants E.N., Istratova E.E. A program for collecting quantitative characteristics of traffic in corporate computer networks. Computer program registration certificate № 2021614232, registered on 22.03.2021. Application № 2021613498 dated 22.03.2021.

[4] Istratova E.E. A program for monitoring traffic characteristics in corporate computer networks. Computer program registration certificate № 2021665972, registered on 06.10.2021. Application № 2021665332 dated 06.10.2021.

[5] Xiao Y., Zhang H., Yuan C. The Design of an Intelligent High-Speed Loom Industry Interconnection Remote Monitoring System // Wireless Pers Commun 113, 2167–2187 (2020). https://doi.org/10.1007/s11277-020-07317-y.

[6] Wang C. Research and implementation of network communication based on embedded monitoring system // Wireless Netw (2023). https://doi.org/10.1007/s11276-023-03307-7.

[7] Boutaba R., Salahuddin M.A., Limam N. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities // J Internet Serv Appl 9, 16 (2018). https://doi.org/10.1186/s13174-018-0087-2.

[8] Hua X. (2021). Application of computer remote network communication technology // Electronic Components and Information Technology, 5(12), 77–78.

[9] Burenin A.N. On some principles of managing server equipment for protected infocommunication networks for special purposes / A.N. Burenin, K.E. Legkov, M.S. Pervov // Science-intensive technologies in space research of the Earth. - 2018. - №. 2. - S. 22-26.

[10] Fedorov A.S. Development of a monitoring system for server equipment / A.S. Fedorov, R.A. Andreev // Economics and quality of communication systems. - 2020. - № 4 (18). - S. 32-42.

[11] Xiaodi Y., Yanjun Z., Jian C., Changyuan Y. (2020). Mobile channel estimation based on decision feedback in vehicle-to-infrastructure visible light communication systems // Optics Communications, 462, 125261.

[12] Simankov V.S. Software and hardware for subsystems of an intelligent situational center / V.S. Simankov, V.A. Sharay // Bulletin of the Adyghe State University. Series 4: Natural-mathematical and technical sciences. - 2021. - № 3 (286). - S. 63-70.

[13] Voevodin V.A. On assessing the timeliness of data exchange in a centralized monitoring system for Wi-Fi networks / V.A. Voevodin, D.S. Burenok // DIZWW. - 2021. - № 17. - S. 60-65.

[14] Mohanty B., Sahoo H.K., Patnaik B. (2020). Wireless channel estimation and beamforming by using block sparse adaptive filtering // Signal, Image and Video Processing, 15(4), 769.

[15] Shariq M., Singh K., Maurya P.K. (2022). AnonSURP: An anonymous and secure ultralightweight RFID protocol for deployment in internet of vehicles systems // The Journal of Supercomputing, 78(6), 8577–8602.

[16] Makeeva O.V. Organization of a corporate server based on Linux / O.V. Makeeva, A.V. Franchuk, M.L. Rysin // Innovations and investments. - 2021. - № 10. - S. 71-77.

[17] Verevkin S.A. Organization of monitoring of a distributed corporate network in order to optimize it and use the obtained data to ensure information security / S.A. Verevkin, R.A. Yukhimuk, D.V. Kudro // News of TulGU. Technical science. - 2023. - № 2. - S. 188-192.

[18] Lindigrin A.N. Analysis of the specifics and problems of anomaly search processes in network data / A.N. Lindigrin // News of TulGU. Technical science. - 2021. - № 5. - S. 304-309.

[19] Sakthivel S., Vidhya G. (2020). A trust-based access control mechanism for intra-sensor network communication in internet of things // Arabian Journal for Science and Engineering, 46, 3147.

[20] Istratova E.E. Program for monitoring the parameters of communication networks and server hardware. Computer program registration certificate № 2023662727, registered on 13.06.2023. Application № 2023662001 dated 13.06.2023.