В результате анализа предметной области и создания программного продукта были учтены основные аспекты взаимодействия пользователя с интерфейсом мониторинга данных, а также продумана технология передачи информации на интерфейс в удобном и лаконичном формате, что позволит грамотно выделять проблемные точки в машинах на металлургическом производстве.

Литература

1. WINNUM – российская платформа промышленного интернета вещей // Офиц. сайт фирмы «*CSoft*» / Информ. для пром. интернета вещей. – 2023. – Режим доступа: https://winnum.csoft.ru. – Дата доступа: 22.03.2023.
2. IntelliJ IDEA – ведущая IDE для разработки на Java и Kotlin. DE, которая делает программирование творческим и увлекательным для разработчиков по всему миру / «JetBrains Distributions s.r.o», 2000–2023. – Режим доступа: https://www.jetbrains.com/ru-ru/idea. – Дата доступа: 22.03.2023.

ПРИНЦИПЫ ПОСТРОЕНИЯ И ТЕХНОЛОГИИ РЕАЛИЗАЦИИ АРХИТЕКТУРЫ НУЛЕВОЙ ЗАДЕРЖКИ ДЛЯ ОБРАБОТКИ ПОТОКА МЕТРИК

Д. И. Тарелко

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель Н. В. Ермалинская

Описаны принципы работы архитектуры нулевой задержки, обоснована необходимость и выбраны современные технологии ее реализации в проектируемом web-приложении для мониторинга состояния эксгаустеров агломашины.

Ключевые слова: поток метрик, нулевая задержка, технологии, web-приложение.

С развитием ИТ-сферы растут и требования к разрабатываемым программным продуктам. Всем необходим широкий функционал, адаптивная визуализация, кроссплатформенность и, конечно же, скорость.

В промышленном производстве скорость обработки информации важна не только с точки зрения планирования и правильного распределения времени, но и с точки зрения оптимизации технологических процессов и контроля технического состояния основного оборудования. В этой связи сокращение длительности нагрузки ядра процессора и концентрированное направление данных в строго определенном потоке является одной из наиболее актуальных проблем промышленной автоматизации при разработке приложений, обрабатывающих высоко нагруженный поток данных. В данном контексте ведутся постоянные работы по улучшению программных решений для достижения максимальной скорости обработки и высокой их производительности.

На промышленных производствах используются программные продукты как низкоуровневых языков, так и более комплексных, например, *C++*. Но, не взирая на простоту локального применения таких решений, не стоит забывать про возможности построения пусть и не тривиальных, но эффективных способов написания похожего рода программ. Стоит также заметить, что при более углубленном анализе задач, существует возможность создания удобных программ для интегрирования в существующие решения, для кроссплатформенности или для загрузки и обслуживания.

В настоящее время почти абсолютное большинство использует такие архитектуры, как архитектура потоковой передачи данных, лямбда-архитектура или *unifield*-архитектура, в основе которой по-прежнему лежит лямбда-архитектура. Однако и они не лишены недостатков. Так, в лямбда-архитектуре пакетный и потоковый уровни работают с разными кейсами, тогда как их внутренняя логика обработки данных почти одинакова. Это приводит к возникновению дублирования данных и кода, что становится источником ошибок. В чистой потоковой архитектуре отсутствует пакетная обработка, поэтому воспроизведение данных и историческая статистика не поддерживаются должным образом.

В силу указанных причин в основу разрабатываемого web-приложения для мониторинга состояния эксгаустеров агломерационной машины в процессе обработки железной руды было принято решение заложить архитектуру нулевой задержки для обработки потока метрик. Именно данный подход позволит наиболее эффективно удовлетворить требования заказчика программного решения.

Для проектирования архитектуры был выбран ряд современных сервисов. Сервис *Apache Kafka* позволяет за счет своего функционала создавать несколько «авторов» и их «слушателей» в открытом потоке передачи строковой информации. Использование базы данных *MongoDB* позволяет разделить процессы обработки данных на кластеры «*primary*», «*secondary*» и «*arbiter*», которые предназначены для получения первичной информации, видоизмененной информации для компоновки, распределения по сущностям, а также для удобной передачи данных без потерь в случае переполнения. Это реализовано следующим образом: когда основная библиотека «*primary*» не работает, подчиненная библиотека «*secondary*» будет выбрана в качестве главной, где «*arbiter*» будет отвечать только за подтверждение количества наборов в кластере с помощью контрольного пакета и в результате решения арбитража при выборе основного сервера. После восстановления основной библиотеки ее можно будет добавить в существующий кластер репликации [1]. *Java*/*Spring Boot* – популярный фреймворк для создания web-приложений, который включает удобные инструменты, позволяющие автоматически настраивать программный продукт на основе зависимостей, добавленных в проект [2]. Также он содержит широкий спектр технологий, что очень полезно в применении *WebSocket* для непрерывной передачи данных. При этом использование технологии *WebSocket* позволяет открывать постоянное двунаправленное сетевое соединение между браузером пользователя и сервером, а также *React.js* на основе хуков эффекта взаимодействия с *WebSocket. InfluxDB*, необходимый для прямой передачи значений по запросу на сервис *Grafana*, содержит в себе легкодоступный инструментарий отображений графиков и потоков метрик.

Таким образом, реализация технологии нулевой задержки для обработки потока метрик при проектировании приложения для мониторинга состояния эксгаустеров агломашин призвана обеспечить: высокую скорость передачи и обработки данных, корректное и своевременное отображение получаемых данных и стабильность в передаче без потери информации.

При разработке архитектуры были выделены два основных направления передачи информации (рис. 1):

1) в базу данных *InfluxDB* с поступающими запросами к ней от сервиса *Grafana* для визуализации потока метрик;

2) в базу данных *MongoDB* с последующей передачей через обработку строковых данных на *api-service.*

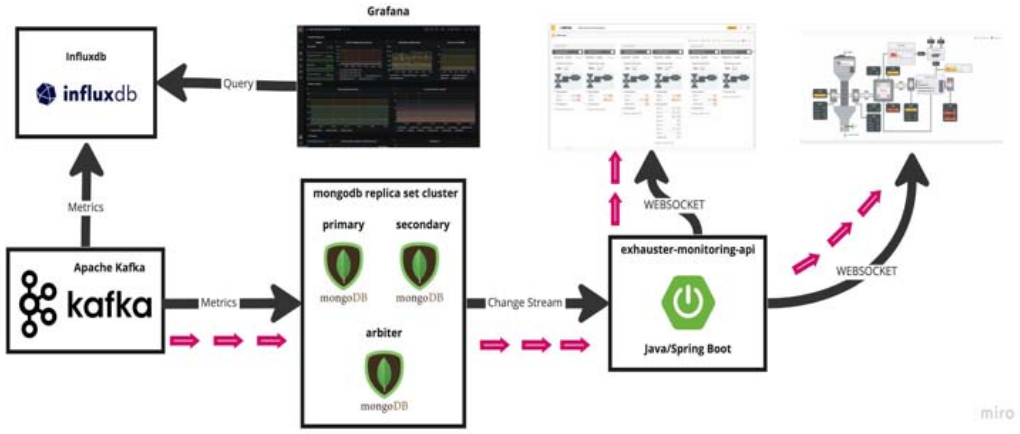

Рис. 1. Представление схемы архитектуры нулевой задержки

После обработки по *WebSocket* будет открыт доступ к представлению данных.

Основные функциональные объекты, созданные при разработке данного решения, следующие:

– *CommonConfiguration* – класс, содержащий в себе поля значений хоста, пользователя и его данных, а также функции преобразования строковой информации, функции *REST*-запросов, функцию выполнения;

– *KafkaTopicListener* – класс, предоставляющий набор функций для прослушивания сообщений и сохранений в базе данных, вывод статистики результатов испытаний по качеству передела или конечного продукта;

– *DaoImpl* – класс с функциями получения метрик;

– *MonitoringController* – контроллер методов запросов на получение информации и ее преобразованием.

Особенность данной архитектуры заключается в следующем:

– в данном решении нет циклических опросов сервисов (*long pooling*);

– метрики из *Kafka* затягиваются через *consumer'a* в *MongoDB*;

– кластер *MongoDB* через *Change Stream* «приносит» метрики API сервису;

– *API* сервис через *WebSocket* «приносит» метрики в браузер;

– передача высокочастотных метрик клиентским приложениям с минимальной задержкой (менее 500 миллисекунд).

Consumer – это необходимый функциональный интерфейс, который принимает один параметр на вход и не возвращает никаких выходных данных. Реализация этого интерфейса потребляет вводимые данные.

Пользовательский интерфейс имеет два метода: *void accept(T t)* и *default Consumer<T> andThen(Consumer<? Super T> after*. Метод *accept* является единым абстрактным методом (*SAM*), который принимает один аргумент типа *T*. Тогда как другой метод *andThen* является методом по умолчанию и используется для композиции (наличия в объекте поля ссылочного типа).

На основе описанной архитектуры при взаимодействии с визуальной отрисовкой *DOM*-элементов на странице браузера могут быть получены динамически изменяемые данные в реальном времени с максимальной задержкой в 100 мс. Фактический показатель задержки будет соответствовать заявленному при условии, что *Kafka* задерживает в потоке передачу предыдущего сообщения после запрашиваемого нового. Практическая реализация показала, что наилучшие значения показателя задержки колеблются от 10 до 30 мс.

Данные из *Kafka* параллельно с записью в *MongoDB* записываются в несколько потоков в *InfluxDB*.

Использование *InfluxDB* позволяет строить временные ряды метрик в *Grafana*. Это открывает возможности поиска аномалий в данных с помощью инструментов для *InfluxDB*.

Таким образом, на основе проведенного анализа архитектур скоростной передачи данных было выбрано оптимальное решение в виде архитектуры нулевой задержки. Данный вид архитектуры позволит передавать большие потоки данных за короткий промежуток времени без потери сообщений. Программные разработки, имеющие такую архитектуру, способны наиболее эффективно решать задачи по автоматизации мониторинга технологических процессов и рабочего оборудования промышленного производства в реальном времени.

Литература

1. Технология сегментированного кластера MongoDB // Русские Блоги / Информация для разработчиков, 2020. – Режим доступа: https://rus_sianblogs.com/ article/9754961584. – Дата доступа: 23.03.2023.
2. Spring Boot 3.0.4 / VMWare VMware, Inc. or its affiliates, 2023. – Режим доступа: https://spring.io/projects/spring-boot. – Дата доступа: 23.03.2023.

WEB-ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ СКЛАДА НЕФТЕПРОДУКТОВ

А. В. Ёжиков

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель Н. В. Ермалинская

Представлены результаты разработки функциональных требований, выбора технологий реализации и обоснования архитектуры проектируемого web-приложения для автоматизации работы склада нефтепродуктов.

Ключевые слова: web-приложение, учетные процессы, склад нефтепродуктов.

В связи с возрастающими количеством транспорта и потребностью в передвижении увеличиваются темпы потребления, а следовательно, и производства нефтепродуктов. Компании, имеющие свои нефтебазы и склады горюче-смазочных материалов, сталкиваются с тем, что большинство складов нефтепродуктов используют в своей работе оборудование и технологии, не обеспечивающие должный уровень технологической и экологической безопасности и точности учета. Внедрение систем комплексной автоматизации должно решить эти проблемы, повысить производственную и финансовую эффективность эксплуатируемых объектов и сократить до минимума экологические риски.

Таким образом, целью проводимых исследований является разработка web-приложения для автоматизации работы склада нефтепродуктов.

Одна из самых главных задач, которая осуществляется сотрудниками комплекса устройств или сооружений для хранения, приема и транспортировки нефти – это количественный учет нефтепродуктов. В обязанности специалистов по учетным мероприятиям входят: оформление документов во время приема нефтепродуктов и количественный анализ; при отгрузке или отправке нефтеперерабатывающей продукции