Потоковая обработка событий аудита

Оглавление

Применение

Механизм обработки записей аудита, действующий по умолчанию, спроектирован исходя из предположения о высокой доступности и производительности СУБД аудита, не оптимизирован на запись в таблицу аудита (каждая вставка идет в своей транзакции, пакетирование не используется). Помимо прочего, аудит замедляет обработку всех входящих запросов из-за синхронной записи в базу, и затрат времени на определение дополнительных данных по словарям GeoIP и UserAgent.

Существенным требованием к механизму записи аудита является синхронность, то есть в случае неуспеха фиксации события должен произойти отказ в обслуживании запроса (например, ошибка записи события об успешной аутентификации должна привести к отказу в аутентификации). Требование было реализовано как синхронная запись в СУБД. Однако, в ходе эксплуатации решения UIDM происходили случаи деградации и полного отказа сервиса аутентификации, когда инстанс СУБД начинал работать с задержками или совсем отказывал на вставку.

Для ускорения работы вышеуказанных механизмов был введен компонент надежной очереди сообщений. Компонент предназначен для того, чтобы отправлять все сообщения аудита (вместо записи их сразу в БД после генерации) в очередь сообщений, где они будут надежно сохраняться. Очереди рассчитаны на быстрое получение большого объема сообщений и их надежное хранение для последующей обработки.

Устройство

Архитектурно применен шаблон Subscribe-Publish (topic), который позволит настроить внутренний роутинг сообщений с дальнейшей обработкой.

Отдельный серверный процесс слушает подписку, берет с нее новые события аудита (пакетами по N сообщений) и записывает их в СУБД также пакетом.

Другой отдельный серверный процесс может «слушать» свою подписку и записывать события на дисковую подсистему для долговременного хранения информации (на развитие).

Реализация очереди как standalone-процесса (в противоположность in-memory в UIDM) предоставит хорошие возможности по конфигурации заказчиком самостоятельно, а так же предоставит средства мониторинга из коробки.

Задачи, решаемые асинхронным аудитом

- Увеличение пропускной способности SSO сервера при обработке сообщений, не замедляясь на обработку записей аудита
- Уменьшение нагрузки на сервер баз данных, отложив запись аудита, и производить ее из другого приложения с контролируемой скоростью и по возможности пакетно
- Предоставление возможности независимого масштабирования отдельных этапов обработки

Потенциальные цели (открытие новых возможностей)

- В случае роста нагрузки на базу аудита — возможность перенести на другой сервер без даунтайма
- Возможность временно остановить запись аудита в БД (в случае большой нагрузки на базу), при этом надежно сохраняя все сообщения в очередь
- Простая интеграция с системами Antifraud через разработку отдельного перекладчика (либо антифрод подключается подписчиком прямо в топик)
- Стриминг событий в ELK для техподдержки
- Обработка данных системами Machine Learning, predictive анализ
- Возможность изменения механизмов хранения на no-sql базы данных, или time-series базы данных.

Не являются целями данного изменения

- Изменение формата записи аудита
- Изменение логики наполнения сообщений аудита информацией

Изменение архитектуры UIDM для включения функциональности

При прямой записи событий аудита в БД («синхронный» вариант реализации), основной сервер приложения UIDM выполняет запись напрямую в СУБД аудита, гарантированно отдельно для каждого протоколируемого события.

```
sso-server -> СУБД (схема RX_AUDIT)
```

При потоковой записи событий аудита («асинхронный» вариант реализации), основной сервер приложения UIDM выполняет запись события аудита в очередь, как и ранее гарантированно и отдельно для каждого протоколируемого события; сообщения из очереди считывает компонент `audit-db-writer` и пачками сохраняет в СУБД.

```
sso-server -> очередь -> audit-db-writer -> СУБД (схема RX_AUDIT)
```

Благодаря новой архитектуре цепочка работы с событиями аудита в дальнейшем может расширяться: могут появляться новые источники событий аудита, кроме сервера UIDM; могут добавляться дополнительные приложения-обработчики событий аудита; могут добавляться дополнительные потребители.

`audit-writer`

Новый компонент в составе UIDM. Устанавливается, запускается и конфигурируется аналогично остальным

компонентам.

`audit-writer` (запись событий аудита в БД из очереди сообщений асинхронного аудита) поставляется в виде rpm-пакета `audit-writer-<версия>-rpm.rpm` и устанавливается с помощью `yum` на сервера приложений UIDM (или на другие сервера, в зависимости от проекта). На сервера должны быть также установлены пакеты конфигурации с необходимыми настройками.

Назначение — запись событий аудита в БД.

Алгоритм работы

1. Получить пакет с (N) входящих записей.
2. Произвести пакетную вставку через JdbcTemplate batchUpdate().
3. В случае если все успешно, подтвердить JMS сессию,
4. В случае ошибки откатить всю сессию.

Очередь сообщений

Решение гарантированно работает с очередями сообщений Artemis, RabbitMQ.

Возможно использование ActiveMQ, Kafka и других с небольшой доработкой. При необходимости поддержать другую очередь обратитесь к аккаунт-менеджеру.

Требования к развертыванию сервера очереди сообщений

1. Настроены повторные доставки сообщений, которые не удалось обработать с backpressure и прогрессивной задержкой, начиная с 10 секунд и до 1 часа, с максимальным количеством попыток доставки равным 10.
2. В случае невозможности обработки сообщения с очереди получателем отправлять сообщения в «очередь мертвых писем» (англ. Dead-letter queue)
3. Сервер очереди должен быть развернут в режиме failover-cluster на двух разных физических устройствах.
4. Сервер должен успешно принимать новые сообщения в очередь, даже если все читатели очереди отключены, сервер должен принимать новые сообщения, которые не считывают с очереди минимум 24 часа. Необходимо рассчитать требуемое количество дискового пространства и поставить его на мониторинг.
5. Вывести в мониторинг события, если сообщения слишком медленно считываются.
6. Предусмотреть подписчиков, в которых допустимы потери сообщений, к примеру сервисы аналитики.
7. Очередь сервиса записи сообщений в БД и постобработки не должна допускать потерь и пропуска сообщений.
8. Должен быть отработан механизм ручного добавления сообщений в очередь в случае ошибок, и необходимости повторно обработать сообщения. Возможность слать сообщения вручную прямо в подписку, а не в очередь.

Конфигурация

Настройки `sso-server`

Параметр	Описание	Значение по умолчанию
com.rooxteam.sso.configuration.auditWriterImpl	Реализация механизма сохранения событий аудита. В параметре может быть указано одно или через запятую несколько из следующих значений: - `jdbcOperationAuditDao` синхронная запись в СУБД - `amqpOperationAuditDao` запись по протоколу AMQP в очередь -	jdbcOperationAuditDao

	`loggingFirstLastOperationAuditDao` запись информации о первом и последнем входе пользователя в СУБД	
com.rooxteam.sso.audit.filter	Включает фильтр по имени событий аудита, регулярное выражение.	.*
com.rooxteam.audit.fields.excluded	Список полей аудита, которые должны быть исключены из записи в очередь, только для потоковой записи.	не задано, пишутся все поля
com.rooxteam.audit.amqp.addresses	Адрес и порт сервера очередей, например 127.0.0.1:5672	не задан
com.rooxteam.audit.amqp.username	Имя пользователя для подключения к серверу очередей	не задан
com.rooxteam.audit.amqp.password	Пароль для подключения к серверу очередей	не задан
com.rooxteam.audit.amqp.exchangeName	Имя топика	audit.raw

ПРЕДУПРЕЖДАЕМ

Если в настройках `sso-server` в качестве режима записи аудита указано одновременно `jdbcOperationAuditDao` и `amqpOperationAuditDao`, то важно убедиться, что `sso-server` и Audit DB Writer подключены к разным СУБД. В противном случае будут возникать ошибки при записи одинаковых событий в одну и ту же базу.

Все параметры требуют перезапуска сервиса `roox-sso`.

Настройки Audit DB Writer

Параметр	Описание	Значение по-умолчанию
com.rooxteam.audit.fields.excluded	Список полей аудита, которые должны быть исключены из записи в БД	не задано, пишутся все поля
com.rooxteam.audit.amqp.addresses	Адрес и порт сервера очередей, например `127.0.0.1:5672`	не задан
com.rooxteam.audit.amqp.username	Имя пользователя для подключения к серверу очередей	не задан
com.rooxteam.audit.amqp.password	Пароль для подключения к серверу очередей	не задан
com.rooxteam.audit.amqp.exchangeName	Имя подписки	audit.raw::to_enrich
com.rooxteam.audit.writer.datasource.url	JDBC URL для подключения к СУБД аудита, например `jdbc:postgresql://db.server.com:5432/RX_AUDIT`	не задан

com.rooxteam.audit.writer.datasource.driver	JDBC Driver для подключения к СУБД аудита, например org.postgresql.Driver	не задан
com.rooxteam.audit.writer.datasource.username	Имя пользователя для подключения к СУБД аудита, например `postgres`	не задан
com.rooxteam.audit.writer.datasource.password	Пароль пользователя для подключения к СУБД аудита, например `postgres`	не задан
com.rooxteam.audit.writer.queueFlushBachSize=250	Размер пакета для записи в БД	250

Все параметры требуют перезапуска сервиса `audit-db-writer`.

Настройка сервера очередей

В очереди необходимо настроить exchange со свойствами:

```
name = audit.raw
durable
```

В exchange необходима подписка с именем

```
audit.raw::to_enrich
```

Особенности и ограничения

Сервисы очередей поддерживают семантику at least once и at most once. At most once для целей обработки аудита не подходит, так как нельзя позволить терять записи аудита. Очереди позволяют транзакционно получить эффект exact once за счет использования Distributed Transactions и two-phase commit, который гарантированно работает с основными РСУБД. Но данный механизм требует детальных настроек, тестирования, и работает довольно медленно.

Ввиду этого принято решение, что сообщение может быть получено сервисом записи в БД более одного раза. В текущей версии схемы аудита отсутствует уникальный индекс на ID записи, поэтому возможно создание дубликатов. Служба эксплуатации должна быть готова к появлению дублирующих записей в таблицах аудита, и при разборе записей проверять ID на наличие дублей. Добавление сообщений в таблицу без дополнительных проверок является наиболее быстрым.

Прочтите также

- Работа с геолокацией пользователя
- Справочник событий аудита