УДК 004.052.4

КЭШИРОВАНИЕ ДАННЫХ В СОБЫТИЙНО-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЕ

Алексин И.Н. (Национальный исследовательский университет ИТМО)
Научный руководитель – кандидат технических наук, доцент Маркина Т.А.
(Национальный исследовательский университет ИТМО)

В статье рассматриваются методы кэширования данных в распределенных системах, содержащих элементы событийно-ориентированной архитектуры. Проводится обзор шаблонов проектирования кэша на уровне приложений и архитектур кэш-систем с учетом особенностей системы оркестрации Kubernetes. Далее разбираются способы удаления и изменения записей в кэше на основе событий, происходящих в распределенной системе. В результате, приводится архитектура приложения, обеспечивающего внедрение кэширования в программные системы с возможностью инвалидации данных при обработке соответствующих событий.

Введение. Многие современные приложения проектируются в виде распределенных систем. Такой подход позволяет создавать отказоустойчивые приложения и производить масштабирование систем при повышении нагрузки. С другой стороны, усложняется логика приложений, так как для получения и обновления данных необходимо производить множество запросов к распределенным компонентам системы вместо запроса к базе данных в случае монолитных архитектур.

Для регулирования коммуникации между микросервисами применяется событийно-ориентированная архитектура. Вместо неконтролируемых обращений друг к другу микросервисы записывают события обновления данных в очередь сообщений, откуда остальные компоненты могут прочитать события, относящиеся к их области ответственности. Основными преимуществами такой архитектуры является создание единого канала взаимодействия компонент системы и переход к асинхронным вызовам, что позволяет клиенту не ждать ответа от всех задействованных микросервисов, изменения данных происходят в фоновом режиме.

В дополнение к созданию очереди событий приложения оптимизируется с помощью кэширования данных, то есть сохранения результатов запросов к базам данных непосредственно в памяти программных компонентов или в специализированном программном обеспечении, реализующем хранение данных в оперативной памяти. В результате, доступ к данным осуществляется без обращения к микросервисам и менее скоростному основному хранилищу. Таким образом, организация кэширования является важной задачей при проектировании программных систем, так как это уменьшает время отклика приложения.

Однако, наряду с увеличением производительности внедрение кэширования усложняет программную систему, так как является альтернативным источником данных. Отсюда появляются проблемы согласования кэша и основных баз данных, задача отбора данных для кэширования, изменения и очистки записей, реагирования кэш-памяти на события, происходящие в программной системе.

Основная часть. Главной задачей этой работы является разработка архитектуры кэш-приложения, обеспечивающего внедрение кэширования в программные системы, развёрнутые в системе оркестрации Kubernetes. Основным требованиями к кэш-приложению являются

изменение или очистка данных (инвалидация кэша) при получении событий об изменении данных в основном хранилище, получаемых с помощью очереди сообщений.

Первым шагом были изучены паттерны проектирования, применимые для кэширования. В результате, для реализации логики добавления записей в кэш был выбран гибридный подход, состоящий из двух паттернов. При получении HTTP GET-запросов был выбран классический алгоритм Cache Aside, который на каждый запрос сначала проверяет кэш и обновляет его данными из основного хранилища, если ответ на запрос не найден. При получении запросов на изменение данных (POST, PUT, DELETE) применялся подход Refresh Ahead. В этом случае запросы проходят к бизнес-логике приложений, которая изменяет данные в основном хранилище, а также создаёт объект в очереди сообщений с указанием, какие именно данные были изменены. Компонент кэш-приложения (Cache Controller) реагирует на события из очереди сообщений и удаляет или изменяет соответствующие данные из кэш-памяти.

Далее были рассмотрены способы развёртывания кэша в системах оркестрации контейнеров. Наиболее простым способом является создание кэша на уровне Reverse Proxy: если ответ на запрос найден в кэше на уровне API Gateway, то он возвращается пользователю без запросов к микросервисам. Однако, такой подход не позволяет инвалидировать данные, что приводит к несогласованности между кэшем и основным хранилищем. Для решения данной проблемы предлагается вновь использовать гибридное решение, состоящее из Client-Server подхода (для создания кэш-кластера на основе Redis), а также использования паттерна Sidecar Container для управления кэшем. Таким образом, каждый микросервис разворачивается вместе с sidecar cache контейнером, и все запросы к микросервису проходят через данный cache контейнер. В результате, при GET-запросах контейнер возваращет ответ пользователя без запросов к бизнес-логике. При изменениях в данных контейнер создаёт события для Cache Controller, который изменит или удалит данные из кэша.

Выводы. Таким образом, спроектировано приложение, которое может быть интегрировано в распределённые системы с целью реализации логики кэширования данных без изменений бизнес-логики программных систем. Основным достоинством представленной архитектуры является возможность изменять данные в кэш-памяти при получении соответсвующих событий от обслуживаемой системы. Приложение, реализованное с использованием данной архитектуры, протестировано на платформе Minikube. После тестирования производительности и надёжности оно может быть интегрировано в приложения, развернутые на платформе Kubernetes.

Алексин И.Н. (автор)

Маркина Т.А. (научный руководитель)