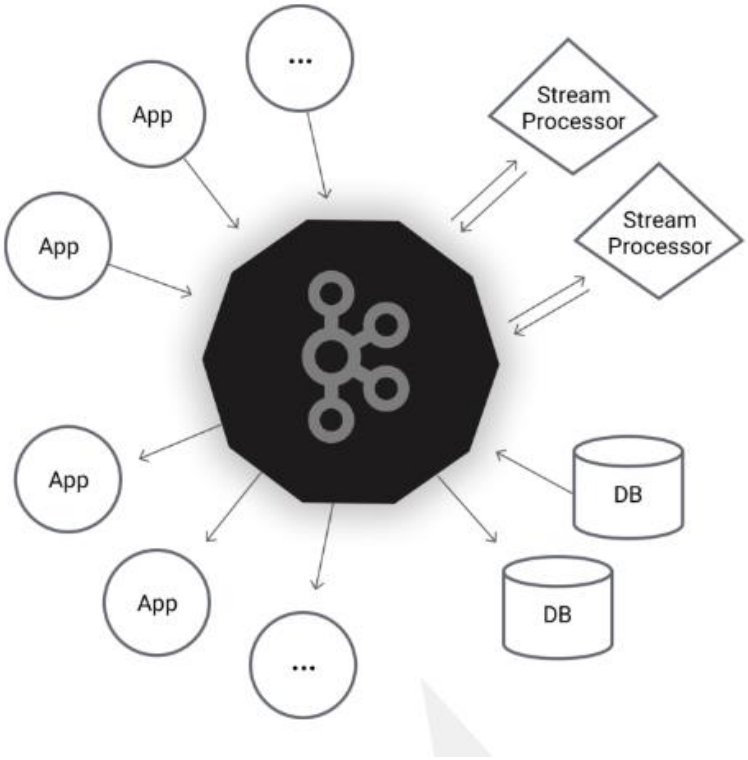
Функциональное описание Q.StreameR

Exported on Jul 08, 2019

Q.StreameR используется для построения конвейеров данных в реальном времени и потоковых приложений. Q.StreameR масштабируется по горизонтали, отказоустойчив и обладает высокой производительностью.



- Q.StreameR - это распределенная потоковая платформа. Что именно это означает?
- Топики и логи.
- Распределение
- Репликация
- Поставщики сообщений
- Подписчики
- Multi-tenancy (множественная аренда)
- Гарантии
- Q.StreameR - система сообщений
- Q.StreameR - система хранения
- Q.StreameR - потоковая обработка
- Собирая все вместе

1 Q.StreameR - это распределенная потоковая платформа. Что именно это означает?

Потоковая платформа обладает тремя ключевыми свойствами:

- Публикация и подписка на потоки сообщений (схожая с системами Message Queue или корпоративными системами обмена сообщениями).
- Долговременное и отказоустойчивое хранение потоков записей.
- Обработка потоков записей по мере их появления.

Q.StreameR может использоваться для двух больших классов приложений:

- Построение потоковых конвейеров данных в реальном времени, которые надежно передают данные между системами или приложениями.
- Создание потоковых приложений в реальном времени, которые преобразуют потоки данных или реагируют на них.

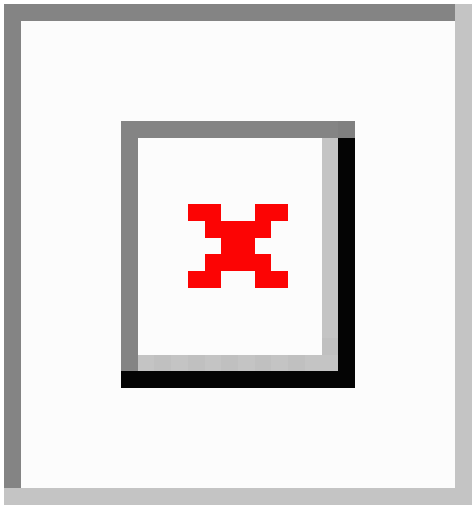
Чтобы понять, как Q.StreameR делает эти вещи, давайте углубимся и рассмотрим возможности Q.StreameR снизу вверх.

Первые несколько концепций:

- Q.StreameR запускается как кластер на одном или нескольких серверах, которые могут охватывать несколько центров обработки данных.
- Кластер Q.StreameR хранит потоки записей в категориях, называемых топиками.
- Каждая запись состоит из ключа, значения и отметки времени.

Q.StreameR имеет четыре базовых группы APIs:

- Producer API позволяет приложению публиковать поток записей в одном или нескольких топиках Q.StreameR.
- Consumer API позволяет приложению подписываться на один или несколько топиков и обрабатывать поток созданных для них записей.
- The Streams API позволяет приложению выступать в качестве потокового процессора, потребляя входной поток из одного или нескольких топиков и создавая выходной поток для одного или нескольких исходящих топиков, эффективно преобразовывая входные потоки в выходные потоки.
- The Connector API позволяет создавать и запускать повторно используемых производителей или потребителей, которые связывают топики Q.StreameR с существующими приложениями или системами данных.



В Q.StreameR связь между клиентами и серверами осуществляется с помощью простого, высокопроизводительного, независимого от языка протокола TCP. Этот протокол является версионным и поддерживает обратную совместимость со старой версией. Мы предоставляем Java-клиент для Q.StreameR, но клиенты доступны на многих языках.

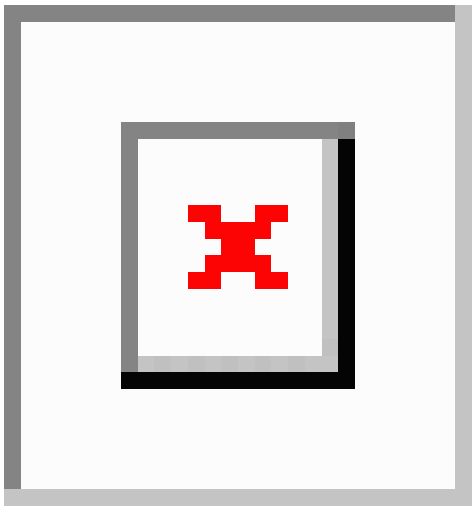
2 Топики и логи.

Давайте сначала углубимся в основную абстракцию, которую Q.StreameR предоставляет потоку записей - топику.

Топик - это категория или название канала, для которого публикуются записи. Топики в Q.StreameR всегда многопользовательские; то есть топик может иметь ноль, одного или нескольких потребителей, которые подписываются на записанные в него данные.
Для каждого топика кластер Q.StreameR поддерживает секционированный журнал, который выглядит следующим образом:

Каждый раздел представляет собой упорядоченную, неизменную последовательность записей, к которой постоянно добавляются - структурированный журнал фиксации. Каждой записи в разделах присваивается последовательный идентификационный номер, называемый смещением, который однозначно идентифицирует каждую запись в разделе.

Кластер Q.StreameR длительно сохраняет все опубликованные записи, независимо от того, были они обработаны или нет, используя настраиваемый срок хранения. Например, если политика хранения установлена на два дня, то в течение двух дней после публикации записи она будет доступна для использования. После этого она будет удалена для освобождения места. Производительность Q.StreameR фактически постоянна в отношении размера данных, поэтому хранение данных в течение длительного времени не является проблемой.



Фактически, единственные метаданные, сохраняемые для каждого потребителя, - это смещение или позиция этого потребителя в журнале. Это смещение контролируется потребителем: обычно потребитель смещает свое смещение линейно при чтении записей, но на самом деле, поскольку позиция контролируется потребителем, он может потреблять записи в любом порядке, который ему нравится. Например, потребитель может сбросить на более старое смещение, чтобы повторно обработать данные из прошлого или перейти к самой последней записи и начать потреблять с «сейчас».

Эта комбинация функций означает, что потребители Q.StreameR очень дешевы - они могут приходить и уходить без особого влияния на кластер или других потребителей. Например, вы можете использовать наши инструменты командной строки, чтобы «подогнать» содержимое любой темы без изменения того, что потребляют какие-либо существующие потребители.

Разделы в журнале служат нескольким целям. Во-первых, они позволяют масштабировать журнал за пределы размера, который поместится на одном сервере. Каждый отдельный раздел должен размещаться на серверах, на которых он размещен, но в топике может быть много разделов, поэтому он может обрабатывать произвольный объем данных. Во-вторых, они действуют как единица параллелизма - об этом чуть позже.

3 Распределение

Разделы журнала распределяются по серверам в кластере Q.StreameR, причем каждый сервер обрабатывает данные и запросы на разделение разделов. Каждый раздел реплицируется на настраиваемое количество серверов для обеспечения отказоустойчивости.

Каждый раздел имеет один сервер, который действует как «лидер», и ноль или более серверов, которые действуют как «последователи». Лидер обрабатывает все запросы на чтение и запись для раздела, в то время как последователи пассивно копируют лидера. Если лидер выходит из строя, один из последователей автоматически становится новым лидером. Каждый сервер выступает в качестве лидера для некоторых своих разделов и последователя для других, поэтому нагрузка в кластере хорошо сбалансирована.

4 Репликация

MirrorMaker Q.StreameR обеспечивает поддержку гео-репликации для ваших кластеров. С помощью MirrorMaker сообщения реплицируются в нескольких центрах обработки данных или облачных регионах. Вы можете использовать это в активных / пассивных сценариях для резервного копирования и восстановления; или в активных / активных сценариях, чтобы размещать данные ближе к вашим пользователям, или поддерживать требования к локальности данных.

5 Поставщики сообщений

Производители публикуют данные в топиках по своему выбору. Производитель несет ответственность за выбор записи, которую следует назначить тому или иному разделу в топике. Это может быть сделано в циклическом режиме, просто чтобы сбалансировать нагрузку, или это может быть сделано в соответствии с некоторой семантической функцией разделения (скажем, на основе некоторого ключа в записи).

6 Подписчики

Потребители помечают себя именем группы потребителей, и каждая запись, опубликованная в топике, доставляется одному экземпляру потребителя в каждой подписавшейся группе потребителей. Потребительские экземпляры могут находиться в отдельных процессах или на отдельных машинах.

Если все экземпляры потребителей имеют одну и ту же группу потребителей, тогда записи будут эффективно сбалансированы по нагрузке по экземплярам потребителей.

Если все экземпляры потребителей имеют разные группы потребителей, то каждая запись будет транслироваться всем процессам потребителей.

Кластер Q.StreameR с двумя серверами, на котором размещены четыре раздела (P0-P3) с двумя группами потребителей. Группа потребителей A имеет два экземпляра потребителей, а группа B - четыре.

Чаще, однако, топики имеют небольшое количество групп потребителей, по одной на каждого «логического подписчика». Каждая группа состоит из множества пользовательских экземпляров для масштабируемости и отказоустойчивости. Это не что иное, как семантика публикации-подписки, где подписчик - это кластер потребителей, а не отдельный процесс.

Реализация потребления в Q.StreameR осуществляется путем разделения разделов в журнале на экземпляры-получатели, так что каждый экземпляр является исключительным потребителем «справедливой доли» разделов в любой момент времени. Этот процесс поддержания членства в группе динамически обрабатывается протоколом Q.StreameR. Если новые экземпляры присоединяются к группе, они принимают некоторые разделы от других членов группы; если экземпляр выходит из строя, его разделы будут распределены между оставшимися экземплярами.

Q.StreameR обеспечивает только общий порядок записей в разделе, а не между различными разделами в топике. Порядок разделений в сочетании с возможностью разделения данных по ключам достаточен для большинства приложений. Однако, если вам требуется общий порядок записей, это может быть достигнуто с топиком, имеющим только один раздел. Хотя это будет означать только один процесс потребителей на группу потребителей.

7 Multi-tenancy (множественная аренда)

Вы можете развернуть Q.StreameR как multy-tenancy решение. Многопользовательский режим включается путем настройки того, какие разделы могут создавать или потреблять данные. Также есть поддержка квот для операций. Администраторы могут определять и применять квоты по запросам для управления ресурсами брокера, которые используются клиентами.

8 Гарантии

На высоком уровне Q.StreameR дает следующие гарантии:

- Сообщения, отправленные производителем в конкретный раздел топика, будут добавляться в том порядке, в котором они были отправлены. То есть, если запись M1 отправляется тем же производителем, что и запись M2, и M1 отправляется первым, тогда M1 будет иметь более низкое смещение, чем M2, и появится раньше в журнале.
- Пользовательский экземпляр видит записи в том порядке, в котором они хранятся в журнале.
- Для раздела с коэффициентом репликации N мы допустим сбои сервера N-1 без потери записей, зафиксированных в журнале.

9 Q.StreameR - система сообщений

Как понятие потоков Q.StreameR можно сравнить с традиционной системой корпоративного обмена сообщениями?

Обмен сообщениями традиционно имеет две модели: *queuing* (организация очередей) и *publish-subscribe* (публикация-подписка). В очереди пул потребителей может читать с сервера, и каждая запись отправляется одному из них; при публикации-подписке запись передается всем потребителям. Каждая из этих двух моделей имеет свои сильные и слабые стороны. Сила очереди состоит в том, что она позволяет разделить обработку данных на несколько пользовательских экземпляров, что позволяет масштабировать обработку. К сожалению, очереди не являются несколькими подписчиками - как только один процесс успешно считывает сообщение,данные удаляются. Публикация-подписка позволяет транслировать данные нескольким процессам, но не имеет способа масштабирования обработки, поскольку каждое сообщение отправляется каждому подписчику.

Концепция группы потребителей в Q.StreameR обобщает эти две концепции. Как и в случае с очередью, группа потребителей позволяет разделить обработку по совокупности процессов (членов группы потребителей). Как и в случае публикации-подписки, Q.StreameR позволяет передавать сообщения нескольким группам потребителей.

Преимущество модели Q.StreameR состоит в том, что каждая тема обладает обоими этими свойствами - она может масштабировать обработку, а также использовать несколько подписчиков - нет необходимости выбирать одно или другое.

Q.StreameR также имеет более строгие гарантии упорядочивания сообщений, чем традиционная система обмена сообщениями.

Традиционная очередь упорядоченно сохраняет записи в на сервере, и если несколько потребителей потребляют из очереди, то сервер передает записи в порядке их хранения. Однако, хотя сервер раздает записи по порядку, они доставляются потребителям асинхронно, поэтому они могут поступать не по порядку для разных потребителей. Это фактически означает, что порядок записей теряется при наличии параллельного потребления. Системы обмена сообщениями часто обходят это, имея понятие «исключительный потребитель», которое позволяет потреблять только один процесс из очереди, но, конечно, это означает, что при обработке нет параллелизма.

Q.StreameR делает это лучше. Имея представление о параллелизме - разделении - по топикам, Q.StreameR может предоставить как гарантии упорядочения, так и балансировку нагрузки по пулу пользовательских процессов. Это достигается путем назначения разделов в теме потребителям в группе потребителей, чтобы каждый раздел потреблялся ровно одним потребителем в группе. Делая это, мы гарантируем, что потребитель является единственным читателем этого раздела и использует данные по порядку. Поскольку существует много разделов, это по-прежнему балансирует нагрузку на множество пользовательских экземпляров. Однако обратите внимание, что в группе потребителей не может быть больше экземпляров потребителей, чем разделов.

10 Q.StreameR - система хранения

Любая очередь сообщений, которая позволяет публиковать сообщения, отделенные от их потребления, эффективно действует как система хранения сообщений. Отличие Q.StreameR в том, что это очень хорошая система хранения.
Данные, переданные в Q.StreameR , записываются на диск и реплицируются для обеспечения отказоустойчивости. Q.StreameR позволяет производителям ждать подтверждения, так что запись не считается завершенной, пока она не будет полностью реплицирована и гарантированно сохранится, даже если сервер записал ошибку.
Структуры дисков Q.StreameR хорошо масштабируется - Q.StreameR будет работать одинаково независимо от того, есть ли у вас 50 КБ или 50 ТБ постоянных данных на сервере.
В результате серьезного отношения к хранилищу и предоставления клиентам возможности контролировать свое положение чтения, вы можете думать о Q.StreameR как о некой специализированной распределенной файловой системе, предназначенной для высокопроизводительного хранения, репликации и распространения журналов фиксации с низкой задержкой.

11 Q.StreameR - потоковая обработка

Недостаточно просто читать, записывать и хранить потоки данных, цель состоит в том, чтобы включить обработку потоков в реальном времени.
В Q.StreameR потоковый процессор - это все, что берет непрерывные потоки данных из входных топиков, выполняет обработку этих входных данных и создает непрерывные потоки данных для исходящих топиков.
Например, приложение розничной торговли может принимать входные потоки продаж и поставок и выводить поток повторных заказов и корректировок цен, рассчитанных на основе этих данных.
Можно выполнить простую обработку напрямую, используя API производителя и потребителя. Однако для более сложных преобразований Q.StreameR предоставляет полностью интегрированный Streams API. Это позволяет создавать приложения, которые выполняют нетривиальную обработку, которая вычисляет агрегации из потоков или объединяет потоки вместе.
Это средство помогает решить сложные проблемы, с которыми сталкивается данный тип приложений: обработка неупорядоченных данных, повторная обработка ввода при изменении кода, выполнение вычислений с учетом состояния и т. д.
API потоков основан на базовых примитивах, которые предоставляет Q.StreameR: он использует API-интерфейсы производителя и потребителя для ввода, использует Q.StreameR для хранения с сохранением состояния и использует тот же групповой механизм для отказоустойчивости среди экземпляров потокового процессора.

12 Собирая все вместе

Такая комбинация обмена сообщениями, хранения и потоковой обработки может показаться необычной, но она важна для роли Q.StreameR в качестве потоковой платформы.
Распределенная файловая система, такая как HDFS, позволяет хранить статические файлы для пакетной обработки. Такая система позволяет хранить и обрабатывать исторические данные из прошлого.
Традиционная корпоративная система обмена сообщениями позволяет обрабатывать будущие сообщения, которые будут поступать после подписки. Созданные таким образом приложения обрабатывают будущие данные по мере их поступления.
Q.StreameR объединяет обе эти возможности, и эта комбинация имеет решающее значение как для использования Q.StreameR в качестве платформы для потоковых приложений, так и для потоковой передачи данных.
Комбинируя хранилище и подписки с малой задержкой, потоковые приложения могут одинаково обрабатывать как прошлые, так и будущие данные. То есть одно приложение может обрабатывать исторические, хранимые данные, но вместо завершения, когда оно достигает последней записи, оно может продолжать обрабатывать по мере поступления будущих данных. Это обобщенное понятие потоковой обработки, которое включает в себя пакетную обработку, а также приложения, управляемые сообщениями.
Аналогично для потоковых конвейеров данных сочетание подписки на события в реальном времени позволяет использовать Q.StreameR для конвейеров с очень низкой задержкой; но способность надежно хранить данные позволяет использовать их для критически важных данных, для которых необходимо гарантировать доставку данных, или для интеграции с автономными системами, которые загружают данные только периодически или могут отключаться в течение продолжительных периодов времени для обслуживания. Средства обработки потоков позволяют преобразовывать данные по мере их поступления.