Engineer

Черга

робота, складена по порядку, щоб її забрали пізніше

Що це таке

Черга — це проміжне сховище між двома програмами: одна сторона кладе туди повідомлення чи задачі, інша забирає їх, коли готова, зазвичай від найстаріших. Майже кожна система має кілька черг, по одній на кожен вид роботи. Програму, що додає роботу, називають виробником, програму, що її забирає, — споживачем, а те, що зберігає повідомлення між ними, — брокером. Брокером може бути окремий сервер або таблиця в базі даних, яка в системі вже є.

Сенс у тому, що дві сторони перестають чекати одна на одну. Вебзапит, якому треба надіслати лист, кладе повідомлення в чергу й одразу відповідає користувачеві; воркер забирає повідомлення за мить, а якщо їх надійде сотня разом, розбирає їх у власному темпі. Сплеск перетворюється на чергу невиконаної роботи, а залежність — на затримку.

Що черга обіцяє щодо доставки, описують два поняття. Перше — підтвердження: споживач повідомляє брокера, що закінчив із повідомленням, і лише тоді воно зникає з черги. У моделі AMQP 0-9-1, яку реалізує RabbitMQ, повідомлення, чий споживач помер, не встигнувши підтвердити, дістається іншому споживачеві. Друге — довговічність: довговічна черга з постійними повідомленнями переживає перезапуск брокера, а черга, що існує лише в пам’яті, втрачає все.

Варто розрізняти три сусідні поняття. Черга повідомлень переносить дані між сервісами. Черга задач переносить одиниці роботи для фонових воркерів, і кожна задача має аргументи, стан і лічильник спроб. Журнал, модель, на якій побудовано Apache Kafka, ззовні схожий на чергу, але

зберігає події й після прочитання, протягом строку, заданого для кожного топіка, тож кілька читачів можуть пройти ту саму історію. Читання з черги прибирає повідомлення; читання з журналу лише зсуває позицію читача.

Черга не обов’язково є окремим продуктом. PostgreSQL упорається з цією роллю за допомогою таблиці задач і запиту, який бере наступний рядок і переступає через рядки, вже заблоковані іншим воркером. Документація прямо називає таблиці на кшталт черги причиною, з якої існує `SKIP LOCKED`:

```
WITH next AS (
  SELECT id
    FROM jobs
   WHERE state = 'available'
   ORDER BY id
   LIMIT 1
     FOR UPDATE SKIP LOCKED
)
UPDATE jobs
   SET state = 'running',
       attempt = attempt + 1
  FROM next
 WHERE jobs.id = next.id
RETURNING jobs.id, jobs.args;
```

Застосуйте поняття до рішення

Застосуйте «Що таке черга?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Цифри з'являються там, де потрібні, за розкладом, і ніхто не копіює таблицю наприкінці місяця.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Черга вирішує, на що чекає користувач. Робота, покладена в чергу, виконується вже після відповіді, тож сторінка, яка запускає лист, PDF, перебудову пошукового індексу й перерахунок, може відповісти за час запису чотирьох рядків, а чотири повільні задачі доходять до кінця

згодом. Для покупця це різниця між оформленням замовлення, що відповідає за кількасот мілісекунд, і таким, що відповідає, коли поштовому серверу заманеться.

Черга вирішує й те, що переживе збій. Робота, запущена у фоновому потоці, живе лише в пам'яті процесу, і розгортання, яке перезапускає процес, забирає її з собою без жодного сліду. Робота в довговічній черзі натомість є записом: її можна повторити, переглянути й порахувати.

А ще черга розводить дві швидкості. Виробники рухаються в темпі вхідного трафіку, споживачі — в темпі найповільнішого сервісу, до якого звертаються. Черга між ними поглинає різницю, і її глибина стає одним із найчесніших чисел, які система може повідомити. Черга, що росте вже годину, каже: споживачі відстають, хоч би що показував графік процесора.

Ці переваги мають ціну. Робота тепер лежить поза полем зору між відправленням і виконанням. Доставка майже завжди відбувається щонайменше один раз, тож споживач мусить витримувати те саме повідомлення двічі. Порядок слабший, ніж здається. А окремий брокер — це ще один сервер, який треба запускати, захищати, стежити за ним і оновлювати.

Як це виглядає на практиці

Надсилання листів, перебудову пошукового індексу, генерування документів і перерахунок рейтингів на одній платформі винесли зі шляху запиту. Робота, віддана горутині, зникає, коли процес перезапускається посеред розгортання, тому роботу перенесли в чергу задач River, що живе в PostgreSQL. База даних і так була авторитетним джерелом даних, тож постановка задачі в чергу стає частиною тієї самої транзакції, що й дані, яких задача стосується. Платформа отримала повторні спроби й історію, яку можна відкрити й прочитати; заявлена межа — черга в базі даних не масштабується безкінечно.

У застосунку для macOS, збудованому над бібліотекою на C, черга мала тримати порядок. Блокувальний цикл подій бібліотеки видає події в послідовності, яка має значення: редагування мусить прийти після повідомлення, яке воно змінює. Окрема задача на кожну подію дозволяє середовищу виконання планувати їх у будь-якому порядку, і редагування може обігнати власне повідомлення. Обране рішення має один потік-

виробник, одного споживача й одну чергу між ними, а далі єдиний стрибок на головного актора. Порядок тримається, бо шлях лише один. Ціна — паралельність: усі події проходять через цього одного споживача.

Черга теж може шкодити. В одному застосунку macOS сплеск подій журналу ядра програми зберігався в необмеженому основному потоці й копіювався підписникам, чиї споживачі цих подій не використовували. Діагностичні звіти зафіксували 111 ГБ стиснутих сторінок на машині з 36 ГБ; окремий замір обсягу зайнятої пам'яті зафіксував зростання приблизно на 41 МБ за секунду під час циклу повторних спроб підключення. Для шляху обробки подій встановили явні межі, відфільтрували підписки за типом події та задали явну політику переповнення. Повторювані діагностичні події можна відкидати й оновлювати відповідний стан, замість того щоб зберігати необмежену кількість подій. Архітектурні перевірки й тести бюджетів застосунку перевіряють дотримання цієї схеми, але запис не підтверджує виміряного показника використання пам'яті після виправлення чи бенчмарку у виробничому середовищі.

Де виникають проблеми

Проблеми з чергами виринають пізно й деінде, бо робота навмисно лежить поза полем зору.

Невиконана робота накопичується, доки не скінчиться пам'ять чи диск. Черга без межі приймає роботу швидше, ніж та виконується, і зберігає різницю. Задайте кожній черзі найбільшу довжину й заздалегідь вирішіть, що відбувається, коли її досягнуто: пригальмувати виробника, відхиляти нову роботу чи викидати найстаріше. Потім стежте за двома числами, глибиною і віком найстарішого повідомлення, бо коротка черга дуже старих повідомлень не краща за довгу.

Те саме повідомлення обробляють двічі. Споживач завершує роботу й падає до того, як підтвердив; брокер цілком слушно доставляє повідомлення знову. Документація RabbitMQ радить споживачам чекати повторних доставок і писати код з думкою про ідемпотентність. Дайте кожному повідомленню ідентифікатор і записуйте вже оброблені ідентифікатори в тій самій транзакції, що й саму роботу. Нотатка на цьому сайті про те, як продати обмежений запас, не продавши більше, ніж є, будує свого записувача саме на цьому припущенні.

Повідомлення приходять не в тому порядку, в якому їх надіслали. Паралельні споживачі, попередня вибірка й повторні спроби, що ставлять невдале повідомлення за новішими, — усе це переставляє роботу. Коли порядок важливий, спрямовуйте все, що стосується одного облікового запису чи одного замовлення, через ту саму партицію або того самого споживача. Kafka робить це за ключем, і ціна в тому, що один завантажений ключ рухається не швидше за одного споживача.

Одне погане повідомлення блокує чергу або ходить по колу. Повідомлення, вміст якого щоразу валить споживача, повертається в чергу, доставляється знову й знову його валить. Обмежте кількість спроб, а повідомлення, яке вичерпало їх усі, перенесіть в окреме місце, де на нього подивиться людина. RabbitMQ називає це dead-lettering і застосовує до відхилених повідомлень, прострочених і тих, що переповнили б обмеження довжини.

Робота зникає під час перезапуску. Канал, список у пам'яті чи фоновий потік — це черга, яка нічого не зберігає між перезапусками. В AMQP довговічної черги самої теж мало: повідомлення мають публікуватися як постійні. Для кожного виду роботи вирішіть, чи прийнятно її втратити.

База даних і черга розходяться. Сервіс записує рядок, потім надсилає повідомлення й падає між цими двома кроками; тепер одне існує без іншого. Або тримайте чергу в тій самій базі, щоб задача ставала видимою для воркерів лише після коміту її транзакції, як це робить River, або записуйте повідомлення в таблицю outbox у тій самій транзакції й надсилайте його звідти.

Що почитати далі

- AMQP 0-9-1 Model Explained — обмінники, черги, прив'язки й підтвердження в поясненні RabbitMQ.
- Consumer Acknowledgements and Publisher Confirms — коли брокер вважає повідомлення доставленим і чому доставка повторюється.
- Dead Letter Exchanges — умови, за яких повідомлення відкладають убік, і куди воно потрапляє.
- The Locking Clause of SELECT — `FOR UPDATE` і `SKIP LOCKED` та таблиці на кшталт черги, яким вони пасують.
- River: Getting started — черга задач, яка зберігає задачі у власній базі даних застосунку.

- Apache Kafka: Introduction — топіки, їхні розділи (partitions) й строк зберігання в моделі журналу.

Словник

Обирайте за фактичними умовами

Надайте репрезентативний набір даних, запитання до нього та поточне вузьке місце. Узгодимо вимірювання якості, пропускної здатності та відновлення.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке черга?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/queue/