Engineer

cola

Trabajo en fila para que otro programa lo recoja

Qué es

Una cola es un lugar de espera entre dos programas: uno introduce mensajes o trabajos y el otro los retira cuando está preparado, normalmente empezando por los más antiguos. La mayoría de los sistemas ejecutan varias colas, una por tipo de trabajo. El programa que añade trabajo es el productor, el que lo retira es el consumidor, y aquello que almacena los mensajes entre ambos es el intermediario, o broker, que puede ser un servidor dedicado o una tabla de una base de datos que el sistema ya tenga.

Esta disposición permite que ambos lados dejen de esperarse mutuamente. Una petición web que necesita enviar un correo añade un mensaje y responde al usuario de inmediato; un trabajador recoge el mensaje momentos después y, si llegan cien juntos, los va procesando a su propio ritmo. Una ráfaga se convierte en trabajo pendiente y una dependencia se convierte en un retraso.

Dos ideas describen lo que promete una cola sobre la entrega. La primera es la confirmación: un consumidor avisa al intermediario cuando ha terminado con un mensaje, y solo entonces se elimina el mensaje. En el modelo AMQP 0-9-1, que implementa RabbitMQ, un mensaje cuyo consumidor muere antes de confirmarlo pasa a otro consumidor. La segunda es la durabilidad: una cola durable de mensajes persistentes sobrevive a un reinicio del intermediario, mientras que una cola mantenida solo en memoria no lo hace.

Conviene distinguir tres términos cercanos. Una cola de mensajes transporta datos entre servicios. Una cola de trabajos transporta unidades de trabajo para trabajadores en segundo plano, cada una con sus argumentos, un estado y un recuento de intentos. Un registro de eventos, el modelo de Apache Kafka, parece una cola desde fuera, pero conserva los eventos durante un periodo de retención

elegido por tema incluso después de que se hayan leído, de modo que varios lectores pueden volver a recorrer el mismo historial. Leer de una cola elimina el mensaje; leer de un registro solo desplaza la posición del lector.

Una cola no tiene por qué ser un producto aparte. PostgreSQL puede desempeñar esa función con una tabla de trabajos y una consulta que reserva la siguiente fila mientras se salta las que otro trabajador ya ha bloqueado. El manual cita las tablas que funcionan como colas como razón de ser de la cláusula `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;
```

Utilice el concepto en una decisión

Aplique «¿Qué es una cola?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Las cifras llegan donde hacen falta según un calendario, sin que nadie tenga que copiar una hoja de cálculo al terminar el mes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Un plan de infraestructura para 14 departamentos — 14 departamentos de investigación con un único plan de infraestructura

Un plan de infraestructura para 14 departamentos Póngase en contacto →

Por qué importa

Una cola determina por qué tareas tiene que esperar un usuario. El trabajo que se coloca en ella ocurre después de enviar la respuesta, de modo que una página que provoca un correo, un PDF, una reindexación de búsqueda y un recálculo puede responder en el tiempo que tarda en escribir cuatro filas, y las cuatro

tareas lentas terminan después. Para un comprador, es la diferencia entre un proceso de compra que responde en unos cientos de milisegundos y otro que responde cuando el servidor de correo está preparado.

También determina qué sobrevive a un fallo. El trabajo iniciado en un hilo en segundo plano vive en la memoria de ese proceso, y un despliegue que reinicia el proceso se lleva el trabajo por delante sin dejar constancia de que debía hacerse. En cambio, el trabajo de una cola durable es un registro: puede reintentarse, inspeccionarse y contarse.

Además, separa dos velocidades. Los productores avanzan a la velocidad del tráfico entrante; los consumidores, a la del servicio más lento al que llaman. Una cola entre ambos absorbe la diferencia y su profundidad se convierte en una de las cifras más reveladoras que puede comunicar un sistema. Una cola que lleva una hora creciendo indica que los consumidores pierden terreno, diga lo que diga el gráfico del procesador.

Los costes llegan con las ventajas. Ahora el trabajo queda fuera de la vista entre su envío y su finalización. La entrega casi siempre es al menos una vez, por lo que un consumidor tiene que tolerar recibir un mensaje dos veces. El orden es menos firme de lo que parece. Y un intermediario dedicado es otro servidor que ejecutar, proteger, vigilar y actualizar.

Cómo se aplica en la práctica

En una plataforma, el envío de correo, la reconstrucción de un índice de búsqueda, la generación de documentos y el recálculo de clasificaciones salieron todos de la ruta de la petición. El trabajo entregado a una goroutine desaparece cuando el proceso se reinicia a mitad de un despliegue, así que pasó a una cola de trabajos River almacenada en PostgreSQL. La base de datos ya era el registro de referencia del sistema, lo que permite que encolar un trabajo forme parte de la misma transacción que los datos a los que afecta. La plataforma obtuvo reintentos y un historial que alguien puede abrir y leer; el límite declarado es que una cola dentro de la base de datos no escalará indefinidamente.

En una aplicación macOS construida sobre una biblioteca C, la cola servía para mantener el orden. El bucle de eventos bloqueante de la biblioteca emite eventos en una secuencia que importa, porque una edición tiene que llegar después del mensaje que edita. Iniciar una tarea por evento permite al entorno de ejecución programarlas en cualquier orden, de modo que una edición puede

adelantar al mensaje al que se refiere. El diseño que se entregó tiene un hilo productor, un consumidor y una cola entre ambos, seguidos de un único salto al actor principal. El orden se mantiene porque hay exactamente un camino de paso. El precio es el paralelismo: todos los eventos pasan por ese único consumidor.

Una cola también puede perjudicar. En una aplicación macOS, una ráfaga de eventos de registro del núcleo de la aplicación se retenía en un flujo principal sin límite y se copiaba a suscriptores cuyos consumidores no utilizaban esos eventos. Los informes de diagnóstico registraron 111 GB de páginas comprimidas en un equipo con 36 GB; una muestra independiente de la huella de memoria registró un crecimiento de unos 41 MB por segundo durante un bucle de reintentos de conexión. La ruta de eventos recibió límites explícitos, con suscripciones filtradas por tipo de evento y una política explícita de desbordamiento. Se pueden descartar eventos de diagnóstico repetitivos y actualizar el estado afectado en vez de conservar eventos sin límite. Las comprobaciones de arquitectura y las pruebas de presupuestos de recursos de la aplicación protegen ese diseño, pero el registro no establece una cifra de memoria medida después de la corrección ni una medición comparativa en producción.

Dónde surgen los problemas

Los problemas de una cola aparecen tarde y en otro lugar, porque el trabajo queda fuera de la vista por diseño.

El trabajo pendiente crece hasta agotar la memoria o el disco. Una cola sin límite acepta trabajo más rápido de lo que se completa y conserva la diferencia. Asigne a cada cola una longitud máxima y decida de antemano qué ocurre al alcanzarla: ralentizar al productor, rechazar trabajo nuevo o descartar el más antiguo. Después vigile dos cifras, la profundidad y la antigüedad del mensaje más viejo, porque una cola corta de mensajes muy antiguos es tan mala como una larga.

El mismo mensaje se procesa dos veces. Un consumidor termina el trabajo y muere antes de confirmarlo; el intermediario, correctamente, vuelve a entregar el mensaje. La documentación de RabbitMQ indica a los consumidores que deben prever entregas repetidas y estar escritos teniendo en cuenta la idempotencia. Asigne un identificador a cada mensaje y registre los ya procesados en la misma

transacción que el propio trabajo. Una nota de este sitio sobre vender existencias limitadas sin vender de más diseña el componente de escritura basándose precisamente en esa premisa.

Los mensajes llegan en un orden distinto del de envío. Los consumidores paralelos, la precarga y los reintentos que colocan un mensaje fallido detrás de otros más recientes cambian el orden del trabajo. Cuando importe el orden, envíe todo lo relativo a una cuenta o a un pedido por la misma partición o el mismo consumidor. Kafka lo hace por clave, a costa de que una clave muy activa solo avance a la velocidad de un consumidor.

Un mensaje defectuoso bloquea la cola o se repite indefinidamente. Un mensaje cuyo contenido hace fallar al consumidor cada vez vuelve a ponerse en cola, se entrega otra vez y vuelve a hacerlo fallar. Limite los intentos y traslade un mensaje que los haya agotado a un lugar aparte donde una persona pueda examinarlo. RabbitMQ lo llama dead-lettering y lo aplica a mensajes rechazados, caducados y a los que harían superar un límite de longitud.

El trabajo desaparece al reiniciar. Un canal, una lista en memoria o un hilo en segundo plano son colas que no conservan nada tras un reinicio. En AMQP tampoco basta una cola durable por sí sola: los mensajes deben publicarse también como persistentes. Decida por tipo de trabajo si es aceptable perderlo.

La base de datos y la cola discrepan. Un servicio escribe una fila, después publica un mensaje y falla entre ambas operaciones; ahora uno existe sin el otro. Mantenga la cola en la misma base de datos, de modo que un trabajo solo sea visible para los trabajadores cuando se confirme su transacción, como hace River, o escriba el mensaje en una tabla de salida, u outbox, dentro de esa transacción y publíquelo desde allí.

Lecturas adicionales

- Explicación del modelo AMQP 0-9-1 — Intercambiadores, colas, enlaces y confirmaciones, explicados por RabbitMQ.
- Confirmaciones de consumidores y de publicación — Cuándo considera un intermediario que se ha entregado un mensaje y por qué vuelve a entregarlo.
- Intercambiadores de mensajes no entregables — Las condiciones que apartan un mensaje y adónde se envía.
- La cláusula de bloqueo de SELECT — `FOR UPDATE` y `SKIP LOCKED`, y las tablas similares a colas para las que resultan adecuados.

- River: primeros pasos — Una cola de trabajos que almacena sus trabajos en la propia base de datos de la aplicación.
- Apache Kafka: introducción — Temas, particiones y retención en el modelo de registro.

Glosario

Elija según sus restricciones reales

Traiga un conjunto de datos representativo, las preguntas que debe responder y el cuello de botella actual. Podemos acordar cómo medir la calidad, la capacidad de procesamiento y la recuperación.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una cola?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/queue/