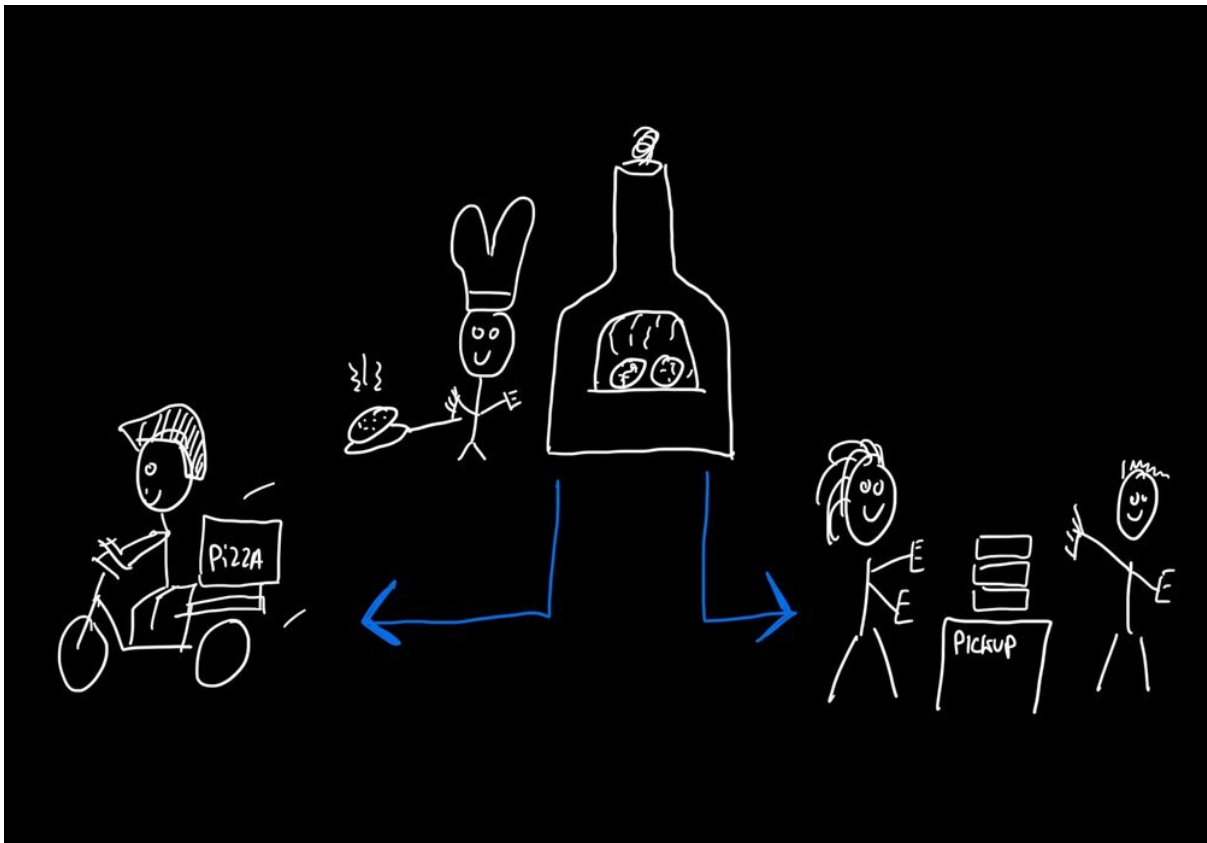


Diseño de una cola de capacidad multidimensional

Gestionar la capacidad de la cocina, los envíos y las recogidas

Willem L. Middelkoop

2 dic. 2020



Este mes necesité crear dimensiones adicionales a un mecanismo de cola de capacidad. La aplicación de pedidos de comida que creé necesitaba poder restringir la capacidad según la cantidad de pedidos, el contenido de los pedidos individuales y el tipo de envío (para llevar/entrega). Sigue leyendo para descubrir cómo utilicé una Arquitectura Lambda para hacer esto.

Restricción de capacidad

Aunque puede que no te des cuenta, una de las características más poderosas de la [food ordering app that I created](#) es, en realidad, poder restringir el número de pedidos. Esto suena contraintuitivo, pero al limitar el número de pedidos que se pueden realizar, mis clientes (los restaurantes y dueños de tiendas que usan mi aplicación) pueden asegurarse de que los pedidos se preparen y entreguen de buena manera.

La complejidad de tres pizzas

Digamos que quieres pedir tres pizzas. La aplicación de comida necesita saber:

- ¿Quieres **recogerlas** o que te las **entreguen**?
- ¿A **qué hora**? ¿Cuándo tendrán que entrar en el horno?
- ¿Hay **suficiente espacio disponible** para *tres pizzas en el horno a esa hora*?
- Si has elegido recogida: ¿**podemos recibirte con seguridad** a esa hora?
- Si has elegido entrega: ¿**hay un repartidor/a disponible** para llevarte tus pizzas?
- ¿**Sí a todo esto**? Entonces **por favor, paga** mientras **mantenemos tus pizzas programadas** en el horno y la entrega o recogida. ¿Error al pagar? ¿Te lo has pensado mejor? Entonces "deshacemos" toda la programación y esperamos a que alguien más pida pizzas.

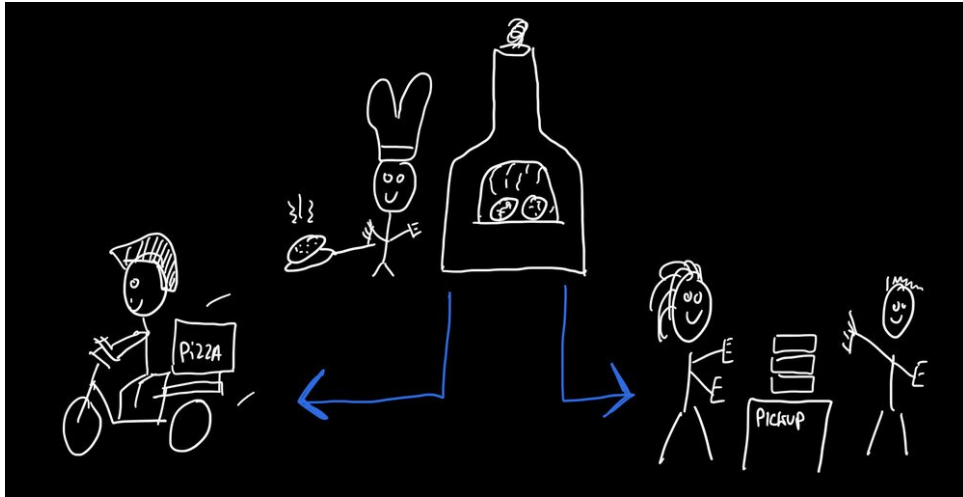


Comer pizza durante el desarrollo es cómo se "vive el algoritmo"

Tres dimensiones de capacidad

Especialmente a la hora de la cena, existe el riesgo de que demasiadas personas pidan comida a la vez. Si bien la aplicación puede manejar el tráfico adicional gracias a su [highly scalable architecture](#), las limitaciones físicas permanecen, tales como:

- **capacidad de la cocina:** la cantidad de platos que se pueden preparar a la vez
- **capacidad de recogida:** la cantidad de personas que se pueden recibir con seguridad a la vez con respecto a las restricciones de COVID-19
- **capacidad de entrega:** la cantidad de pedidos que se pueden entregar, teniendo en cuenta el tráfico, la distancia y la ruta



Tres limitaciones de capacidad: cocina / entrega / recogida

Distintas pero interdependientes

Estos tres tipos diferentes de restricciones son interdependientes. Por ejemplo, es posible que un pedido grande para recoger ocupe toda la capacidad de la cocina, lo que también genera indisponibilidad para la entrega. Todavía se podrían aceptar pedidos más pequeños para completar la capacidad de la cocina, mientras que es posible que ya no haya suficiente capacidad disponible para pedidos grandes. Esto significa que se necesita un mecanismo que considere *todas* las métricas combinadas para determinar la disponibilidad para nuevos pedidos, incluido el contenido de un pedido no realizado (por ejemplo, lo que hay en tu carrito de compras).

Diferencias en la granularidad del intervalo de tiempo

Es común definir una restricción física utilizando un intervalo de tiempo, por ejemplo *"la cocina es capaz de producir 10 platos cada 15 minutos"*. Se vuelve un desafío cuando se definen otras restricciones físicas utilizando diferentes longitudes de intervalo de tiempo. Si bien la cocina podría estar limitada por 15 minutos, la capacidad de entrega podría estar utilizando un intervalo de tiempo más largo (por ejemplo, 3 entregas cada 30 minutos). De alguna manera, el mecanismo debería ser capaz de lidiar con estas diferencias en la granularidad del intervalo de tiempo.

Pedidos por adelantado

A este desafío se suma la opción de que las personas realicen sus pedidos por adelantado. Las personas pueden seleccionar un intervalo de tiempo en el futuro y reservar la capacidad requerida. A veces veo personas que hacen su pedido con días de anticipación para asegurarse de que su comida se entregue en un día y hora específicos. Esto significa que hay (al menos) dos horas asociadas con cada pedido: la hora del pedido y la hora de entrega.

Reservar y liberar capacidad

La mayoría de los dueños de negocios requieren pagos en línea para pedidos en línea, esto es para asegurar que todos los pedidos sean legítimos y que no se desperdicie capacidad

(o comida) en ausencias o pedidos fantasmas. El [problem with online payments](#) es que no todas las transacciones son exitosas. A veces se rechaza un pago con tarjeta de crédito o débito (por ejemplo, debido a fondos insuficientes). A veces las personas simplemente cierran la aplicación antes de completar el pedido. El sistema debe reservar la capacidad requerida para un pedido dado, mientras permite que las personas realicen el pago en línea, y debe liberar la capacidad reservada en caso de que se abandone el pedido o cuando falle el pago.

Alto volumen, alta velocidad

Por último, pero no menos importante, el mecanismo debe ser confiable y eficiente. La disponibilidad del pedido debe determinarse en milisegundos mientras se maneja una gran concurrencia, ya que hay muchos clientes diferentes conectados a la aplicación de pedidos de comida durante la hora pico. Cuando alguien realiza un pedido, afecta la disponibilidad para todos los demás pedidos (potenciales). No todos estarán conectados de manera confiable, ya que algunas personas usan la aplicación en teléfonos con mala recepción o wifi. El sistema debe funcionar bien en diversas condiciones con muchas personas conectadas diferentes.

Arquitectura Lambda

La [Lambda Architecture](#) es una forma de manejar grandes cantidades de datos mediante el uso de métodos de procesamiento de flujo y procesamiento por lotes. Una arquitectura lambda depende de un modelo de datos con una fuente de datos inmutable de solo anexión, a menudo compuesta por eventos con marca de tiempo.

Datos inmutables

Esta es una base muy importante para el mecanismo: todos los datos tienen marca de tiempo y *nunca* cambian. En lugar de modificar los registros existentes, simplemente se agregan nuevos registros para sobrescribir los registros existentes. Esto permite una alta concurrencia y algoritmos distribuidos, ya que la sincronización se simplifica ya que solo hay que buscar datos nuevos (¡en lugar de nuevos y actualizados!).

Capa de velocidad/tiempo real

El procesamiento de flujo, o capa de velocidad/tiempo real, en una arquitectura lambda está diseñado para proporcionar respuestas súper rápidas a las necesidades de datos en tiempo real. Mantiene vistas y métricas basadas en el procesamiento de eventos en tiempo real.

Capa de precomputación

El procesamiento por lotes, o capa de precomputación, proporciona una base completa y precisa de todos los datos del sistema. Como el procesamiento de grandes cantidades de datos puede llevar tiempo, el procesamiento por lotes no es lo suficientemente rápido como para hacer todo al instante. Es por eso que las dos capas trabajan en conjunto en una arquitectura lambda.

Implementación

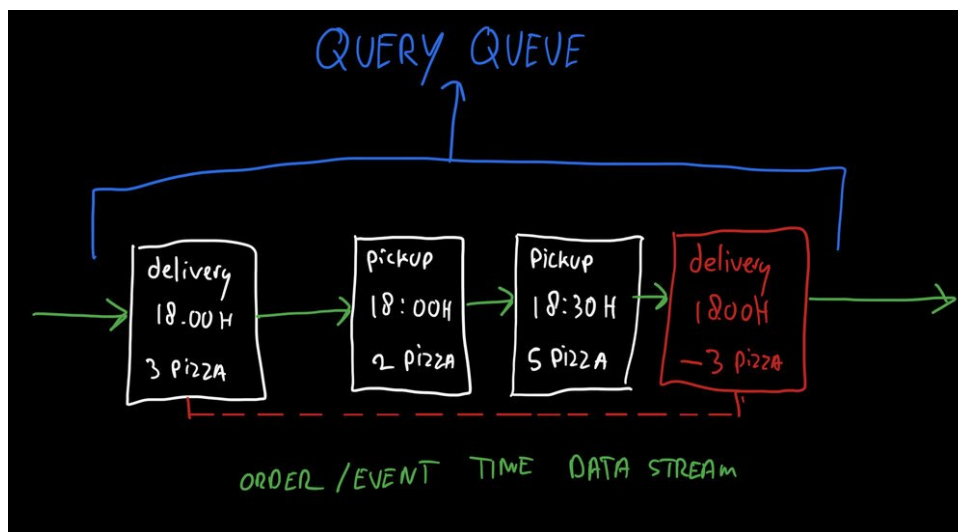
Los servidores backend que manejan los pedidos de comida de la aplicación trabajan con un flujo de datos de eventos con marca de tiempo. En esencia: cada pedido es un evento en el sistema. Cada pedido se almacena con su fecha/hora de pedido, fecha/hora de envío (hora programada de entrega o recogida), detalles del cliente y contenido del pedido (número de platos, etc.).

Reservar y liberar capacidad

Si falla el pago o si se abandona un pedido, se agrega un pedido "negativo" al flujo de datos. Neutraliza cualquier efecto sobre las restricciones de capacidad por sus métricas negativas. Por ejemplo: para cancelar un pedido de tres pizzas, se crea un pedido negativo de "menos tres pizzas" en el flujo de datos con la misma marca de tiempo de envío. Esta es una forma elegante de asegurarse de que la capacidad esté reservada y liberada.

Métricas multidimensionales en tiempo real

En lugar de abstraer la capacidad en contadores, el servidor backend utiliza los datos de flujo *reales* para responder a las consultas de métricas en tiempo real. Esto permite que el sistema maneje las diferencias en la granularidad del intervalo de tiempo.



Un flujo de eventos de pedidos es el corazón del mecanismo, las consultas se cumplen mirando este flujo: ¿cuántas pizzas estarán listas a las 18:00hrs?

La primera vez que se recibe una consulta para una métrica determinada, el sistema carga datos históricos a través de un mecanismo por lotes para inicializar el modelo que luego puede mantener actualizado en tiempo real. Esta métrica se mantiene actualizada al observar todos los pedidos nuevos que ingresan al sistema mientras lleguen consultas para una métrica determinada. Una vez que los datos ya no se solicitan, el modelo se descarga. Si se necesita nuevamente, el proceso se repite.

Lo mejor de este enfoque es que la mayoría de las consultas se responden en milisegundos, ya que se leen directamente de la memoria RAM sin interacción con la base de datos o el disco. Esto permite un rendimiento ultra alto "según sea necesario". Esto

funciona bien con la aplicación de pedidos de comida, ya que no *todos* los intervalos de tiempo son igualmente populares.

La aplicación de pedidos de comida consulta repetidamente un servidor de capacidad central que mantiene las métricas de la cola. Luego filtra los intervalos de tiempo disponibles observando las cosas reales que una persona está seleccionando para su pedido. Por lo tanto, la selección de un intervalo de tiempo (probablemente) disponible se realiza del lado del cliente, lo que reduce la carga en el backend. Una vez que se realiza un pedido, el backend reserva la capacidad actualizando las métricas de la cola central, lo que impide que otras personas reclamen el mismo lugar en la cola.

Conclusión

Los mecanismos de cola pueden ser desafiantes cuando se trata de múltiples métricas que controlan la velocidad. Realmente debería tomarse el tiempo para modelarlo correctamente, piense en el tipo de datos que *realmente* necesita para realizar comprobaciones relacionadas con la cola.

¡Ahora sabes que pedir tres pizzas no es tan simple! Garantizar que se puedan servir calientes y a tiempo es un gran desafío técnico, además de ser un buen chef. ¡Feliz por ti de que *yo* no me centre en esto último, ja!