Temas de Kafka: trampas y conocimientos

ThousandEyes Ingeniería Seguir
15 de febrero de 2019 · 7 min de lectura

Autor: Raúl Benencia , Ingeniero Senior de Confiabilidad de Sitios en ThousandEyes.

Hacer un seguimiento del tamaño de un tema de Kafka y observar cómo varía con el tiempo nos ayudó a descubrir patrones inesperados que podrían parecer misteriosos para el administrador de clústeres sin experiencia. Incluso con límites establecidos, el tamaño informado puede variar enormemente. Se pueden ver picos periódicos tras la eliminación de datos, con una frecuencia que varía de un tema a otro. Eliminar datos de los temas de Kafka puede ser aún más desconcertante, ya que, bajo ciertas condiciones,

la información que ha pasado mucho tiempo después de la política de retención puede desenterrarse de la profundidad del tema.

En esta publicación de blog, compartiremos algunos escenarios que nos dejaron rascándonos la cabeza. Estos no eran obvios a primera vista y, en algunos casos, nos obligó a leer el código fuente de Kafka para arrojar luz sobre lo que estaba sucediendo.

Descripción general de la arquitectura de Kafka

Antes de pasar a los jugosos detalles, repasemos rápidamente cómo funciona Kafka y almacena su información. Un clúster de Kafka está formado por uno o más corredores de Kafka. Los clústeres de Kafka contienen temas que actúan como una cola de mensajes donde las aplicaciones cliente pueden escribir y leer sus datos. Los temas se dividen en particiones y estas particiones se distribuyen entre los corredores de Kafka. Los corredores no suelen poseer todas las particiones para todos los temas. Solo pueden tener algunas particiones por tema.

Una partición se puede replicar a otros corredores, y así es como Kafka proporciona redundancia: si un corredor falla, las particiones que tenía se pueden encontrar repartidas en el resto de los corredores. Las particiones se almacenan en el disco como 'segmentos' y tienen un tamaño de 1 GB de forma predeterminada. Entonces, cuanto más grande sea la partición, más segmentos tendrá en el disco.

Hay mucha información sobre cómo Kafka funciona internamente , por lo que nuestra modesta descripción general no profundizará más.Sin más preámbulos, hablaremos ahora sobre los escenarios intrigantes que nos sorprendieron cuando nuestros temas de Kafka comenzaron a crecer en tamaño y las políticas de eliminación no estaban actuando como esperábamos.

Picos de eliminación

Comenzaremos con una situación en la que los temas experimentarían picos de eliminación que diferían en frecuencia, a pesar de que su configuración era exactamente la misma. Como estos temas están configurados para conservar datos durante una semana, esperábamos una caída constante de información en lugar de picos, similar a docenas de otros temas que tenemos en nuestro grupo. Entonces, ¿por qué estábamos viendo picos?

Notamos este escenario cuando comenzamos a acumular datos y pudimos ver las políticas de retención y limpieza en acción. En la Figura 1 mostramos el tamaño de cuatro temas, incluida la replicación. Como puede ver, dos de ellos son puntiagudos mientras que los otros dos siguen un patrón esperado.

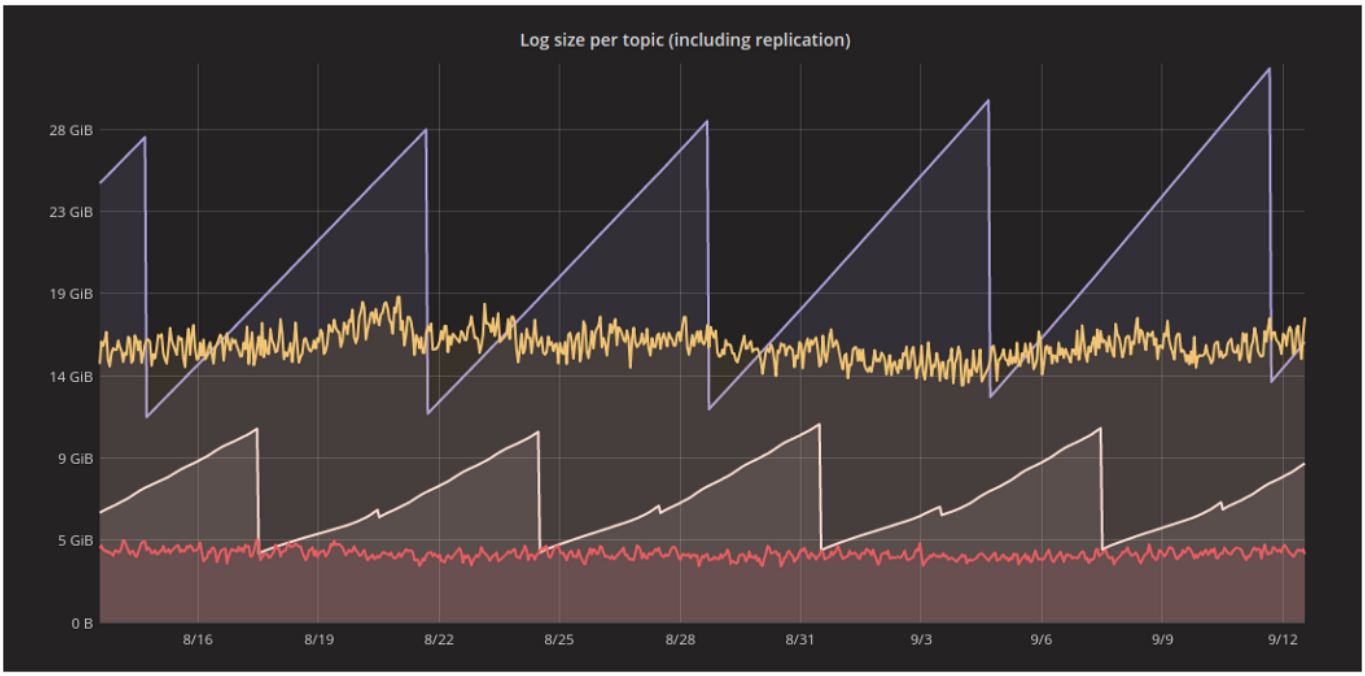

Figura 1: Dos temas con picos y dos temas con el comportamiento que esperábamos.

Originalmente, todos tenían picos, pero cuando encontramos el motivo de los picos, pudimos volver a configurar dos de los temas para obtener el comportamiento esperado.

Entonces, ¿cuál es la razón detrás de estos picos? Comprendamos esto estudiando la línea violeta en la Figura 1, que representa un tema cuyo tamaño varía de 16GB a 32GB. Es un tema con 32 particiones y un factor de replicación de 3. El resto de los parámetros tienen su valor predeterminado. De forma predeterminada, los segmentos se implementan cuando alcanzan 1 GB de tamaño o una vez a la semana (predeterminado), lo que ocurra primero.

La clave para comprender los picos es notar que las políticas de eliminación no actúan sobre los segmentos activos. Si los segmentos no se llenan, no se implementarán hasta que se cumpla el número de horas definido en **log.roll.hours** , el valor predeterminado es una semana. Entonces, los picos que vemos en la Figura 1 son la suma de los tamaños de los segmentos para todas las particiones del tema que no alcanzan 1 GB en siete días.

Vemos las grandes caídas porque todas se implementan a la misma hora una vez a la semana, cuando se cumple el período de tiempo.

Si los picos de eliminación ocurren con menos frecuencia, digamos cinco días, es porque los segmentos de ese tema alcanzaron su tamaño máximo y se implementaron. Ese fue el caso de otros temas que tenemos: la cantidad de datos que veríamos eliminados era enorme, a veces 60 GB o más. Después de investigar esto, comprendimos que se debía a que los segmentos se estaban llenando. Teníamos 32 particiones y cada una de ellas se replica 3 veces y, por lo general, todos los segmentos se despliegan al mismo tiempo.

Los picos por sí mismos son inofensivos, pero es mejor tener temas que no cambien de tamaño para que podamos usar el espacio en disco de manera más eficiente. Los picos se pueden evitar mediante la disminución del valor de la **segment.bytes** parámetro, que especifica el tamaño de segmento, a un valor de acuerdo con la cantidad de datos a ser manipulados por el tema. En la Figura 1, el tamaño del segmento para los dos temas que no tienen picos se redujo a 128 MB.

Cuando los temas crecen

La segunda situación que discutiremos está relacionada con el tamaño de ciertos temas, ya que eran más grandes de lo esperado. Algunas aplicaciones que desarrollamos necesitan cargar y procesar todos los datos de un tema antes de poder trabajar, y debido al tamaño inusual del tema, el tiempo de arranque de estas aplicaciones tomaría casi un minuto, en lugar del tiempo casi instantáneo. que estábamos esperando. Tener grandes temas fue perjudicial para el rendimiento de la aplicación. Para limpiar los registros más antiguos del tema y, por lo tanto, restringir el tamaño del tema, utilizamos la política de **eliminación** de Kafka . En este escenario en particular, aunque configuramos el tamaño máximo de un tema o el tiempo máximo de espera antes de eliminar los archivos de registro, el tamaño del tema aumentaría constantemente.

Después de investigar un poco, descubrimos que las políticas de **eliminación** no tienen una granularidad fina: pueden eliminar segmentos, pero no datos antiguos que estén dentro de un segmento. Como tal, solo eliminará los segmentos para los que la totalidad de la información sea anterior al tiempo de retención. Esto da como resultado que Kafka elimine el segmento como un todo solo cuando el último registro agregado es anterior al tiempo de retención especificado. Naturalmente, este comportamiento crea una gran fluctuación en el tamaño de un tema, lo que a veces resulta en tamaños

inesperadamente grandes. Una vez más, reducir el valor de **segmento.bytes** ayuda a lograr un tamaño constante.

Datos antiguos en temas que no se eliminan

La última situación que discutiremos es la más misteriosa. Habíamos configurado un tema con un período de retención de una semana. Durante una semana normal recibiría muy pocos datos, por lo que, en teoría, el tema siempre se cargaría rápido y mantendría solo unos pocos megabytes de espacio. Resulta que el tema no era información pequeña ni retenida durante solo una semana: ¡pudimos recuperar misteriosamente información de varios meses! El protagonista de esta anécdota es un tema configurado con una política de borrado de **compactar, borrar** .

Aunque usábamos una política de **eliminación compacta** con un período de retención de una semana, veíamos mensajes con varios meses de antigüedad. ¿Cómo pudo pasar esto? Bueno, como indica el nombre de la política, Kafka primero ejecutará el proceso compacto y luego el proceso de eliminación. La compactación funciona analizando todo el segmento y dejando solo la última entrada para cada clave de mensaje. Las entradas más antiguas se eliminan. Para evitar tener múltiples segmentos más pequeños, la política **compacta** los fusionará.

El comportamiento descrito tiene la siguiente consecuencia. Durante la fase compacta, las claves de registro no repetidas que están al principio del tema, ubicadas en los primeros segmentos de una partición, terminarán fusionándose con segmentos que contienen registros más nuevos. Los segmentos más nuevos, por supuesto, tendrán claves que aún se encuentran en el marco de tiempo de retención. Una vez finalizada la compactación, Kafka activará la política de **eliminación** . Ya mencionamos que esta política piensa en segmentos, no en mensajes individuales y que no eliminará segmentos que contengan información que no ha caducado. Como el segmento más antiguo contiene claves nuevas y antiguas, no se eliminará, por lo que el consumidor de este tema verá información ***muy***¡antiguo! Una solución alternativa para evitar este caso de esquina es, una vez más, reducir el tamaño de **segmento.bytes** .

Lecciones aprendidas

Al crear un tema nuevo, la cantidad de particiones no es el único parámetro importante a configurar. El tamaño del segmento es un parámetro clave si no se espera que la

cantidad de datos retenidos por segmento supere el tamaño del segmento predeterminado durante el período de retención. Además, al considerar el tamaño total de un tema, debe tener en cuenta el número de particiones, la política de eliminación y el factor de replicación.

Nuestra confusión inicial provino de no comprender en profundidad cómo funcionan las políticas de eliminación. Hay mucha documentación basada en cómo funciona la política **compacta** , pero no tanto con respecto a la política de **eliminación** . Principalmente, lo que hace **compacto** es marcar algunos registros en el segmento para su eliminación. Si los segmentos se vuelven demasiado pequeños, Kafka los fusionará.

La principal confusión en la política de **eliminación** proviene de asumir que Kafka siempre eliminará los datos anteriores al tiempo de retención especificado. En particular, es un error suponer que el administrador de registros eliminará los datos de los segmentos que tienen datos más recientes que el período de retención especificado.

Cada una de estas situaciones “kafkianas” se abordó y se encontró una explicación perfectamente lógica. Para algunos de ellos, la razón fue una mala interpretación de la documentación, para otros fue el comportamiento oscuro que Kafka puede tener bajo ciertas condiciones. Como era de esperar, hay mejoras continuas en el propio Kafka y es posible que estos escenarios se hayan abordado en versiones más recientes.

Publicado originalmente en blog.thousandeyes.com el 15 de febrero de 2019.

Kafka Apache Kafka Tema de Kafka Big Data