05 Sincronización

Oplog

La replicación se refiere a mantener una copia idéntica de los datos en varios servidores. La forma en que MongoDB logra esto es manteniendo un registro de operaciones, u oplog, que contiene cada escritura que realiza un primario. Esta es una colección capped situada en la base de datos local del primario. Los secundarios consultan en esta colección las operaciones que se replicarán.

```
clusterGetafe:PRIMARY> use local
switched to db local
clusterGetafe:PRIMARY> show collections
oplog.rs
...
clusterGetafe:PRIMARY> db.oplog.rs.find()
```

Ejecutamos una operación:

```
clusterGetafe:PRIMARY> use getafeTest
switched to db getafeTest

clusterGetafe:PRIMARY> db.foo3.insert({puntuacion: 0})
WriteResult({ "nInserted" : 1 })

clusterGetafe:PRIMARY> db.foo3.update({},{$inc:{puntuacion: 1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Comprobamos:

```
clusterGetafe:PRIMARY> use local
switched to db local
clusterGetafe:PRIMARY> db.oplog.rs.find().sort({$natural: -1})

...
{ "ts" : Timestamp(1581454938, 1), "t" : NumberLong(36), "h" : NumberLong(0), "v" : 2, "op" : "u",
"ns" : "getafeTest.foo3", "ui" : UUID("b2a05066-8096-4327-8ac3-42006cf613bb"), "o2" : { "_id" :
ObjectId("5e43160d2837798f4a9a44a9") }, "wall" : ISODate("2020-02-11T21:02:18.390Z"), "o" : {
"$v" : 1, "$set" : { "puntuacion" : 1 } } }

...
{ "ts" : Timestamp(1581454861, 2), "t" : NumberLong(36), "h" : NumberLong(0), "v" : 2, "op" : "i",
"ns" : "getafeTest.foo3", "ui" : UUID("b2a05066-8096-4327-8ac3-42006cf613bb"), "wall" :
ISODate("2020-02-11T21:01:01.823Z"), "o" : { "_id" : ObjectId("5e43160d2837798f4a9a44a9"),
"puntuacion" : 0 } }
```

Cada secundario mantiene su propio registro de operaciones, registrando cada operación que replica desde el primario. Esto permite que cualquier miembro se use como fuente de sincronización para cualquier otro miembro, como se muestra en la Figura 11-1. Los secundarios obtienen operaciones del miembro con el que se están sincronizando, aplican las operaciones a su conjunto de datos y luego escriben las operaciones en su registro de operaciones. Si falla la aplicación de una operación (que solo debería ocurrir si los datos subyacentes se han dañado o de alguna manera difieren de los primarios), el secundario se cerrará.

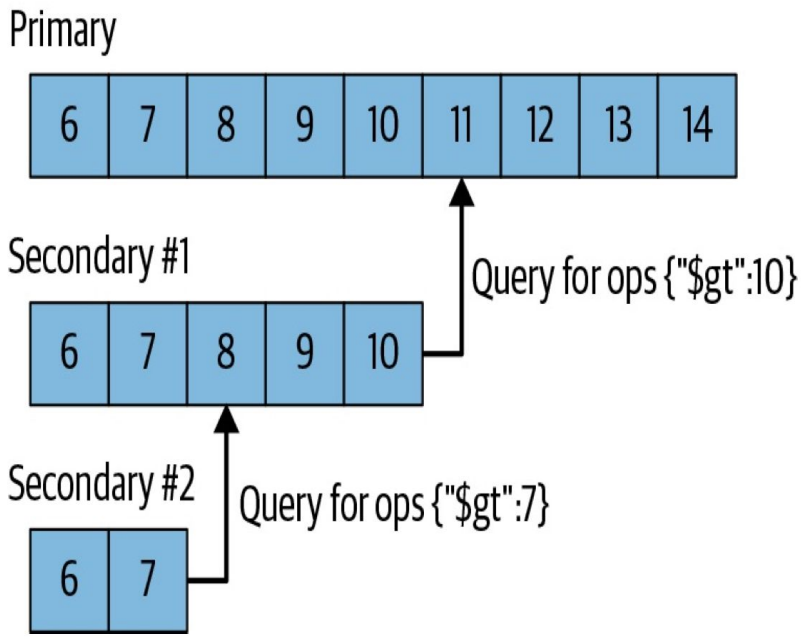

Figure 11-1. Oplogs keep an ordered list of write operations that have occurred; each member has its own copy of the oplog, which should be identical to the primary's (modulo some lag)

Si un secundario deja de funcionar por algún motivo, cuando se reinicie, comenzará a sincronizarse desde la última operación en su registro de operaciones. A medida que las operaciones se aplican a los datos y luego se escriben en el registro de operaciones, el secundario puede reproducir operaciones que ya ha aplicado a sus datos. MongoDB está diseñado para manejar esto correctamente: reproducir las operaciones de oplog varias veces produce el mismo resultado que reproducirlas una vez. Cada operación en el oplog es idempotente. Es decir, las operaciones de oplog producen los mismos resultados si se aplican una o varias veces al conjunto de datos de destino.

Debido a que el oplog es de un tamaño fijo, solo puede contener un cierto número de operaciones. En general, el oplog utilizará el espacio aproximadamente a la misma velocidad que las escrituras que ingresan al sistema: si escribe 1 KB / minuto en el primario, su oplog probablemente se llene a aproximadamente 1 KB / minuto.

Sin embargo, hay algunas excepciones: las operaciones que afectan a varios documentos, como eliminaciones o actualizaciones múltiples, se explotarán en muchas entradas de oplog.

La operación única en el primario se dividirá en una operación de registro por documento afectado. Por lo tanto, si elimina 1,000,000 de documentos de una colección con db.coll.remove (), se convertirán en 1,000,000 de entradas de registro eliminando un documento a la vez. Si está realizando muchas operaciones masivas, esto puede llenar su oplog más rápido de lo que podría esperar.

En la mayoría de los casos, el tamaño de oplog predeterminado es suficiente. Si puede predecir la carga de trabajo de su conjunto de réplicas para que se parezca a uno de los siguientes patrones, puede crear un registro de operaciones que sea más grande que el predeterminado.

Por el contrario, si su aplicación realiza predominantemente lecturas con una cantidad mínima de operaciones de escritura, un registro de operaciones más pequeño puede ser suficiente.

Estos son los tipos de cargas de trabajo que pueden requerir un tamaño de registro más grande:

Actualizaciones a múltiples documentos a la vez

El oplog debe traducir las actualizaciones múltiples en operaciones individuales para mantener la idempotencia. Esto puede usar una gran cantidad de espacio de registro sin un aumento correspondiente en el tamaño de los datos o el uso del disco.

Las eliminaciones equivalen a la misma cantidad de datos que las inserciones.

Si elimina aproximadamente la misma cantidad de datos que inserta, la base de datos no crecerá significativamente en términos de uso del disco, pero el tamaño del registro de operaciones puede ser bastante grande.

Número significativo de actualizaciones en el lugar

Si una parte importante de la carga de trabajo son actualizaciones que no aumentan el tamaño de los documentos, la base de datos registra una gran cantidad de operaciones, pero la cantidad de datos en el disco no cambia.

Antes de que mongod cree un oplog, puede especificar su tamaño con la opción oplogSizeMB. Sin embargo, después de haber iniciado un miembro del conjunto de réplicas por primera vez, solo puede cambiar el tamaño del registro de operaciones utilizando el procedimiento "Cambiar el tamaño del registro de operaciones".

Formas de sincronización

MongoDB usa dos formas de sincronización de datos: una sincronización inicial para llenar nuevos miembros con el conjunto de datos completo y la replicación para aplicar cambios continuos a todo el conjunto de datos. Echemos un vistazo más de cerca a cada uno de estos.

Sincronización inicial

MongoDB realiza una sincronización inicial para copiar todos los datos de un miembro del conjunto de réplicas a otro miembro. Cuando se inicia un miembro del conjunto, verificará si está en un estado válido para comenzar a sincronizarse con alguien. Si está en un estado válido, intentará hacer una copia completa de los datos de otro miembro del conjunto. Hay varios pasos para el proceso, que puede seguir en el registro del mongod.

Primero, MongoDB clona todas las bases de datos excepto la base de datos local. El mongod escanea cada colección en cada base de datos de origen e inserta todos los datos en sus propias copias de estas colecciones en el miembro de destino. Antes de comenzar las operaciones de clonación, se eliminarán todos los datos existentes en el miembro de destino.

ADVERTENCIA. Solo realice una sincronización inicial para un miembro si no desea los datos en su directorio de datos o los ha movido a otro lugar, ya que la primera acción de mongod es eliminarlos todos.

En MongoDB 3.4 y versiones posteriores, la sincronización inicial genera todos los índices de la colección a medida que se copian los documentos para cada colección (en versiones anteriores, solo los índices "_id" se crean durante esta etapa). También extrae los registros oplog recién agregados durante la copia de datos, por lo que debe asegurarse de que el miembro de destino tenga suficiente espacio en disco en la base de datos local para almacenar estos registros durante esta etapa de copia de datos.

Una vez que se clonan todas las bases de datos, el mongod usa el registro de operaciones de la fuente para actualizar su conjunto de datos para reflejar el estado actual del conjunto de réplicas, aplicando todos los cambios al conjunto de datos que ocurrieron mientras la copia estaba en progreso. Estos cambios pueden incluir cualquier tipo de escritura (inserciones, actualizaciones y eliminaciones), y este proceso puede significar que mongod tiene que reclinar ciertos documentos que fueron movidos y, por lo tanto, perdidos por el clonador.

Dependiendo del nivel de tráfico y los tipos de operaciones que ocurrieron en la fuente de sincronización, puede que le falten o no objetos.

En este punto, los datos deben coincidir exactamente con el conjunto de datos tal como existía en algún momento en el primario. El miembro finaliza el proceso de sincronización inicial y pasa a la sincronización normal, lo que le permite convertirse en secundario.

Hacer una sincronización inicial es muy fácil desde la perspectiva del operador: simplemente inicie un mongod con un directorio de datos limpio. Sin embargo, a menudo es preferible restaurar desde una copia de seguridad, como se describe más adelante. La restauración desde una copia de seguridad suele ser más rápida que copiar todos sus datos a través de mongod.

Además, la clonación puede arruinar el conjunto de trabajo de la fuente de sincronización. Muchas implementaciones terminan con un subconjunto de sus datos a los que se accede con frecuencia y siempre en la memoria (porque el sistema operativo accede con frecuencia). Realizar una sincronización inicial obliga al miembro a buscar todos sus datos en la memoria, desalojando los datos utilizados con frecuencia. Esto puede ralentizar drásticamente a un miembro ya que las solicitudes que los datos en la RAM manejaban de repente se ven obligadas a ir al disco. Sin embargo, para pequeños conjuntos de datos y servidores con algo de espacio para respirar, la sincronización inicial es una opción buena y fácil.

Uno de los problemas más comunes que enfrentan los miembros con la sincronización inicial es que lleva demasiado tiempo. En estos casos, el nuevo miembro puede "caerse" al final del oplog de la fuente de sincronización: se queda tan atrás de la fuente de sincronización que ya no puede ponerse al día porque el oplog de la fuente de sincronización ha sobrescrito los datos que el miembro necesitaría usar para continuar replicando.

No hay forma de arreglar esto más que intentar la sincronización inicial en un momento menos ocupado o restaurar desde una copia de seguridad. La sincronización inicial no puede continuar si el miembro se ha caído del oplog de la fuente de sincronización.

Replicación

El segundo tipo de sincronización que realiza MongoDB es la replicación. Los miembros secundarios replican datos continuamente después de la sincronización inicial. Copian el oplog de su fuente de sincronización y aplican estas operaciones en un proceso asincrónico.

Los secundarios pueden cambiar automáticamente su sincronización de la fuente según sea necesario, en respuesta a los cambios en el tiempo de ping y el estado de la replicación de otros miembros. Hay varias reglas que gobiernan desde qué miembros se puede sincronizar un nodo dado. Por ejemplo, los miembros del conjunto de réplicas con un voto no pueden sincronizarse con miembros con cero votos, y las secundarias evitan la sincronización de miembros retrasados y miembros ocultos. Las elecciones y las diferentes clases de miembros del conjunto de réplicas se analizan en secciones posteriores.

Manejo de la obsolescencia

Si un secundario se queda muy atrás de las operaciones en tiempo real que se realizan en la fuente de sincronización, el secundario quedará obsoleto. Un secundario obsoleto no puede ponerse al día porque cada operación en el registro de operaciones de la fuente de sincronización está demasiado adelante. Esto podría suceder si el secundario ha tenido tiempo de inactividad, tiene más escrituras de las que puede manejar o está demasiado ocupado manejando lecturas.

Cuando un secundario se vuelve obsoleto, intentará replicarse de cada miembro del conjunto para ver si hay alguien con un oplog más largo desde el que pueda arrancar. Si no hay nadie con un registro de operaciones lo suficientemente largo, la replicación en ese miembro se detendrá y será necesario volver a sincronizarla por completo (o restaurarla desde una copia de seguridad más reciente).

Para evitar secundarios no sincronizados, es importante tener un gran registro de operaciones para que el primario pueda almacenar un largo historial de operaciones. Un oplog más grande obviamente usará más espacio en el disco, pero en general es una buena compensación porque el espacio en el disco tiende a ser barato y normalmente se

usa poco del oplog, por lo que no ocupa mucha RAM. Una regla general es que el registro de operaciones debe proporcionar cobertura (ventana de replicación) durante dos o tres días de operaciones normales.

Estados de los miembros

Los miembros necesitan saber sobre los estados de los otros miembros: quién es el principal, con quién se pueden sincronizar y quién está inactivo. Para mantener una vista actualizada del conjunto, un miembro envía una solicitud de latido a todos los demás miembros del conjunto cada dos segundos. Una solicitud de latido es un mensaje corto que verifica el estado de todos.

Una de las funciones más importantes de los latidos del corazón es informar al primario si puede alcanzar a la mayoría del conjunto. Si un primario ya no puede llegar a la mayoría de los servidores, se degradará y se convertirá en secundario.

Los miembros también comunican en qué estado se encuentran a través de los latidos del corazón. Ya hemos discutido dos estados: primario y secundario. Hay varios otros estados normales en los que a menudo verá miembros estar en:

STARTUP

Este es el estado en el que se encuentra un miembro cuando se inicia por primera vez, mientras MongoDB intenta cargar su configuración de conjunto de réplicas. Una vez que se ha cargado la configuración, pasa a STARTUP2.

STARTUP2

Este estado dura todo el proceso de sincronización inicial, que generalmente demora solo unos segundos. El miembro bifurca un par de hilos para manejar la replicación y las elecciones y luego pasa al siguiente estado: RECUPERACIÓN.

RECOVERING

Este estado indica que el miembro está funcionando correctamente pero no está disponible para lecturas. Puede verlo en una variedad de situaciones.

Al inicio, un miembro tiene que hacer algunas verificaciones para asegurarse de que esté en un estado válido antes de aceptar lecturas; por lo tanto, todos los miembros pasan brevemente por el estado de RECUPERACIÓN al inicio antes de convertirse en secundarios. Un miembro también puede entrar en este estado durante operaciones de larga ejecución, como la compactación o en respuesta al comando replSetMaintenance.

Un miembro también pasará al estado de RECUPERACIÓN si se ha quedado demasiado atrás (obsoleto) de los otros miembros para ponerse al día. Este es, generalmente, un estado de falla que requiere resincronizar al miembro. El miembro no entra en un estado de error en este punto porque vive con la esperanza de que alguien se conecte en línea con un oplog lo suficientemente largo como para que pueda reiniciarse a sí mismo.

ARBITER

Los árbitros (ver "Árbitros electorales") tienen un estado especial y siempre deben estar en este estado durante el funcionamiento normal.

También hay algunos estados que indican un problema con el sistema. Éstos incluyen:

DOWN

Si un miembro estaba arriba pero luego se vuelve inalcanzable, entrará en este estado. Tenga en cuenta que un miembro reportado como "inactivo" podría, de hecho, aún estar activo, simplemente inaccesible debido a problemas de red.

UNKNOWN

Si un miembro nunca ha podido comunicarse con otro miembro, no sabrá en qué estado se encuentra, por lo que lo informará como DESCONOCIDO. Esto generalmente indica que el miembro desconocido está inactivo o que hay problemas de red entre los dos miembros.

REMOVED

Este es el estado de un miembro que se ha eliminado del conjunto. Si un miembro eliminado se agrega nuevamente al conjunto, volverá a pasar a su estado "normal".

ROLLBACK

Este estado se usa cuando un miembro revierte los datos, como se describe en "Rollbacks". Al final del proceso de reversión, un servidor volverá a pasar al estado RECOVERING y luego se convertirá en secundario.