2 Storage

Concepto storage engine

El motor de almacenamiento es el componente del gestor de base de datos responsable de gestionar como los datos son almacenados tanto en memoria como en disco. MongoDB soporta múltiples motores de almacenamientos con diferentes rendimientos para cargas de datos específicas, impactando significativamente en el rendimiento de las aplicaciones.

NOTA

A partir de la versión 4.2, MongoDB ha eliminado el MMAPv1 storage engine.

WiredTiger Storage Engine (Por defecto)

Es el motor por defecto, incorporado en la versión 3.2 y es el diseñado para la mayoría de cargas de datos y recomendado para nuevos desarrollos. Proporciona entre otras características, un modelo de concurrencia a nivel de documento, checkpoints y compresión.

En la versión MongoDB Enterprise, WiredTiger también soporta encriptación.

Snapshots and Checkpoints

WiredTiger utiliza el control de simultaneidad de conversión múltiple (MVCC). Al comienzo de una operación, WiredTiger proporciona una instantánea de un punto en el tiempo de los datos de la operación. Una instantánea presenta una vista coherente de los datos en memoria.

Al escribir en el disco, WiredTiger escribe todos los datos en una instantánea en el disco de manera coherente en todos los archivos de datos. Los datos ahora duraderos actúan como un punto de control en los archivos de datos. El punto de control asegura que los archivos de datos sean consistentes hasta el último punto de control inclusive; es decir, los puntos de control pueden actuar como puntos de recuperación.

A partir de la versión 3.6, MongoDB configura WiredTiger para crear puntos de control (es decir, escribir los datos de la instantánea en el disco) a intervalos de 60 segundos.

Durante la escritura de un nuevo punto de control, el punto de control anterior sigue siendo válido. Como tal, incluso si MongoDB termina o encuentra un error al escribir un nuevo punto de control, al reiniciar, MongoDB puede recuperarse del último punto de control válido.

El nuevo punto de control se vuelve accesible y permanente cuando la tabla de metadatos de WiredTiger se actualiza atómicamente para hacer referencia al nuevo punto de control. Una vez que se puede acceder al nuevo punto de control, WiredTiger libera páginas de los puntos de control antiguos.

Usando WiredTiger, incluso sin journal, MongoDB puede recuperarse desde el último punto de control; sin embargo, para recuperar los cambios realizados después del último punto de control, ejecute con journaling.

Journal

WiredTiger utiliza un registro de escritura anticipada (es decir, journal) en combinación con puntos de control para garantizar la durabilidad de los datos.

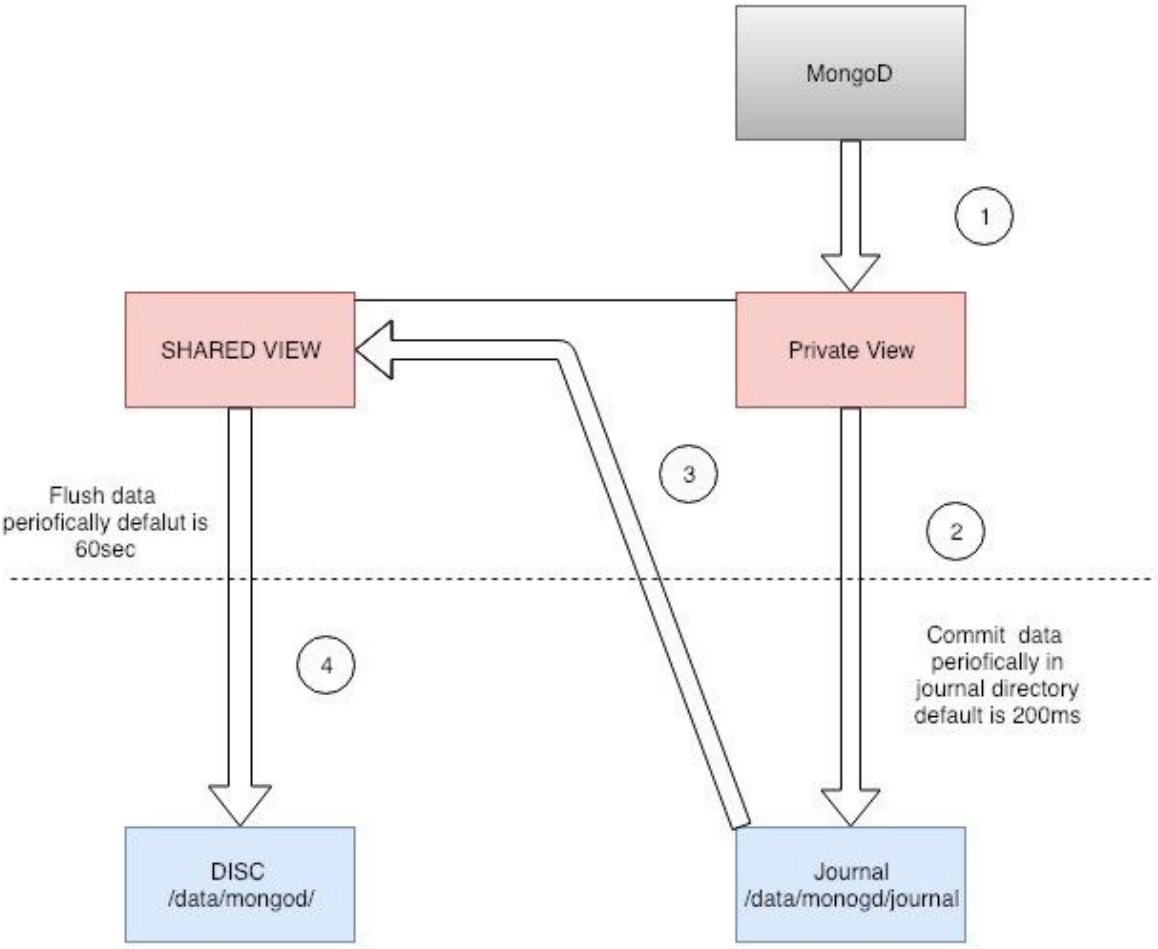
El journal WiredTiger persiste todas las modificaciones de datos entre puntos de control. Si MongoDB cae entre los puntos de control, usa el diario para reproducir todos los datos modificados desde el último punto de control.

El diario WiredTiger se comprime utilizando la librería de compresión rápida snappy. Para especificar un algoritmo de compresión diferente o sin compresión, use la configuración storage.wiredTiger.engineConfig.journalCompressor.

NOTA Si un registro de registro es menor o igual a 128 bytes (el tamaño de registro de registro mínimo para WiredTiger), WiredTiger no comprime ese registro.

Puede deshabilitar el journal para instancias independientes estableciendo storage.journal.enabled en false, lo que puede reducir la sobrecarga de mantener el diario.

Para instancias independientes, no usar el diario significa que, cuando MongoDB se cierra inesperadamente, perderá todas las modificaciones de datos antes del último punto de control.



1.- Cuando la operación de escritura se realiza en MongoDB, los primeros cambios se realizan en una vista privada.

2.- Después de un intervalo especificado, que se denomina intervalo de confirmación de journal, la vista privada escribe esas operaciones en el directorio journal (que reside en el disco).

3.- Después de que se confirma el journal, MongoDB inserta datos en la vista compartida.

4.- Después de un intervalo de tiempo especificado (60 segundos por defecto), se copia al directorio principal de MongoDB.

Después de que los datos se vacían en el directorio principal de MongoDB, se marcan como procesados y se eliminan de la memoria del journal.

La vista privada y la vista compartida tienen espacio asignado en la RAM

NOTE Starting in MongoDB 4.0, you cannot specify --nojournal option or storage.journal.enabled: false for replica set members that use the WiredTiger storage engine.

Journal File Size Limit
WiredTiger journal files for MongoDB have a maximum size limit of approximately 100 MB.

Once the file exceeds that limit, WiredTiger creates a new journal file. WiredTiger automatically removes old journal files to maintain only the files needed to recover from last checkpoint.

Compression

With WiredTiger, MongoDB supports compression for all collections and indexes. Compression minimizes storage use at the expense of additional CPU.

By default, WiredTiger uses block compression with the snappy compression library for all collections and prefix compression for all indexes.

For collections, the following block compression libraries are also available:

zlib

zstd (Available starting in MongoDB 4.2)
To specify an alternate compression algorithm or no compression, use the storage.wiredTiger.collectionConfig.blockCompressor setting.

For indexes, to disable prefix compression, use the storage.wiredTiger.indexConfig.prefixCompression setting.

Compression settings are also configurable on a per-collection and per-index basis during collection and index creation. See Specify Storage Engine Options and db.collection.createIndex() storageEngine option.

For most workloads, the default compression settings balance storage efficiency and processing requirements.

The WiredTiger journal is also compressed by default. For information on journal compression, see Journal.

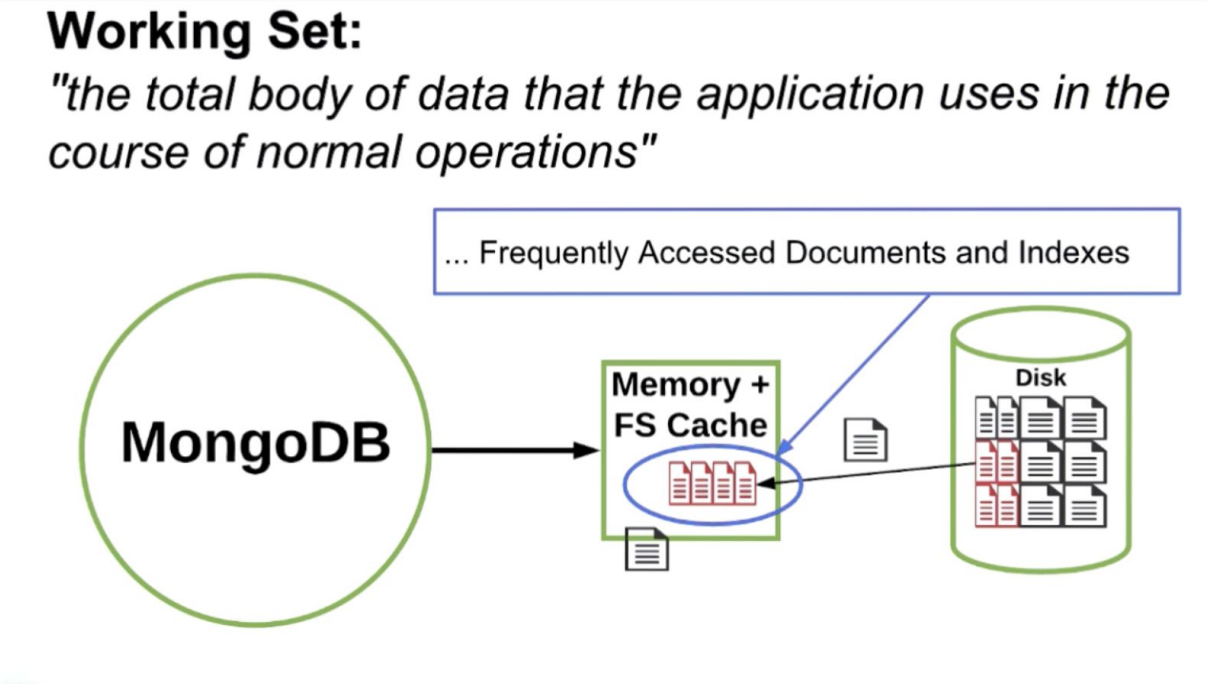
Memory Use

With WiredTiger, MongoDB utilizes both the WiredTiger internal cache and the filesystem cache.

Starting in MongoDB 3.4, the default WiredTiger internal cache size is the larger of either:

- 50% of (RAM - 1 GB),
- or 256 MB.

For example, on a system with a total of 4GB of RAM the WiredTiger cache will use 1.5GB of RAM (0.5 * (4 GB - 1 GB) = 1.5 GB). Conversely, a system with a total of 1.25 GB of RAM will allocate 256 MB to the WiredTiger cache because that is more than half of the total RAM minus one gigabyte (0.5 * (1.25 GB - 1 GB) = 128 MB < 256 MB).



NOTE In some instances, such as when running in a container, the database can have memory constraints that are lower than the total system memory. In such instances, this memory limit, rather than the total system memory, is used as the maximum RAM available.

Different representations are used for data in the WiredTiger internal cache versus the on-disk format:

- Data in the filesystem cache is the same as the on-disk format, including benefits of any compression for data files. The filesystem cache is used by the operating system to reduce disk I/O.

- Indexes loaded in the WiredTiger internal cache have a different data representation to the on-disk format, but can still take advantage of index prefix compression to reduce RAM usage.

- Collection data in the WiredTiger internal cache is uncompressed and uses a different representation from the on-disk format. Block compression can provide significant on-disk storage savings, but data must be uncompressed to be manipulated by the server.

Via the filesystem cache, MongoDB automatically uses all free memory that is not used by the WiredTiger cache or by other processes.

In-Memory Storage Engine

Este motor está disponible para la versión MongoDB Enterprise y en vez de persistir los datos en disco, utiliza la tecnología in-memory para controlar la latencia de datos.

https://docs.mongodb.com/manual/core/inmemory/