Journaling in Mongodb

ANJU MUNOTH

Journaling

- To provide durability in the event of a failure, MongoDB uses write ahead logging to on-disk journal files
- The WiredTiger storage engine does not require journaling to guarantee a consistent state after a crash. Because the database will be restored to the last consistent checkpoint during recovery.
- However, if MongoDB exits unexpectedly in between checkpoints, journaling is required to recover writes that occurred after the last checkpoint.

What is Journaling?

- Much like one uses a journal to record thoughts and daily events, MongoDB uses a journal to ensure data integrity.
- Accomplished through writing data first to the journal files and then to the core data files. In the event of an untimely server shutdown, the data can be restored to a consistent state.
- Accomplished through MongoDB's write operation durability guarantee.
- If the mongod process stops in an unexpected manner, data from the journal will be used to re-apply the write operations when it is restarted.
- MongoDB creates, when journaling is enabled, a subdirectory for the journal data called journal. This resides under the dbPath directory and contains the write ahead logs.

Journaling and the WiredTiger Storage Engine

- WiredTiger uses checkpoints to provide a consistent view of data on disk and allow MongoDB to recover from the last checkpoint.
- However, if MongoDB exits unexpectedly in between checkpoints, journaling is required to recover information that occurred after the last checkpoint.

Journaling and the WiredTiger Storage Engine

- The WiredTiger storage engine takes a different approach to write operation data concurrency.
- WiredTiger uses checkpoints in conjunction with a journal.
- These checkpoints allow for data to be recovered after the last checkpoint.
- When a write operation is called, a snapshot is taken of the data. When data is written to disk (every 60 seconds by default), the data is written across all data files and becomes durable. This becomes a new checkpoint and can be used as a recovery point.
- This allows for WiredTiger to be covered from the last checkpoint without a journal.
- However, if an unexpected shutdown occurs between checkpoints and journaling is disabled, data will be lost. The journal in WiredTiger, therefore, utilizes a write-ahead log between checkpoints for data durability.

In-Memory

- For those that are running an Enterprise version of MongoDB, there is a storage engine that stores data in memory
- Because memory is stored in memory, the data is non-persistent. The concept of a journal does not apply in this situation.

Recovery process

With journaling, the recovery process:

- Looks in the data files to find the identifier of the last checkpoint.
- Searches in the journal files for the record that matches the identifier of the last checkpoint.
- Apply the operations in the journal files since the last checkpoint.

Recovery

- With journaling enabled, if mongod stops unexpectedly, the program can recover everything written to the journal.
- MongoDB will re-apply the write operations on restart and maintain a consistent state.
- By default, the greatest extent of lost writes, i.e., those not made to the journal, are those made in the last 50 milliseconds, plus the time it takes to perform the actual journal writes.

Journaling Process

- With journaling, WiredTiger creates one journal record for each client initiated write operation.
- The journal record includes any internal write operations caused by the initial write.
- For example, an update to a document in a collection may result in modifications to the indexes;
- WiredTiger creates a single journal record that includes both the update operation and its associated index modifications
- MongoDB configures WiredTiger to use in-memory buffering for storing the journal records.
- Threads coordinate to allocate and copy into their portion of the buffer. All journal records up to 128 kB are buffered

Journaling Process

WiredTiger syncs the buffered journal records to disk according to the following intervals or conditions:

- MongoDB syncs the buffered journal data to disk every 50 milliseconds (Starting in MongoDB 3.2)
- If the write operation includes a write concern of j: true, WiredTiger forces a sync of the WiredTiger journal files.
- Because MongoDB uses a journal file size limit of 100 MB, WiredTiger creates a new journal file approximately every 100 MB of data. When WiredTiger creates a new journal file, WiredTiger syncs the previous journal file.

Journal Files

- For the journal files, MongoDB creates a subdirectory named journal under the dbPath directory.
- WiredTiger journal files have names with the following format WiredTigerLog.<sequence> where <sequence> is a zero-padded number starting from 0000000001.

Journal Records

Journal files contain a record per each client initiated write operation

The journal record includes any internal write operations caused by the initial write.

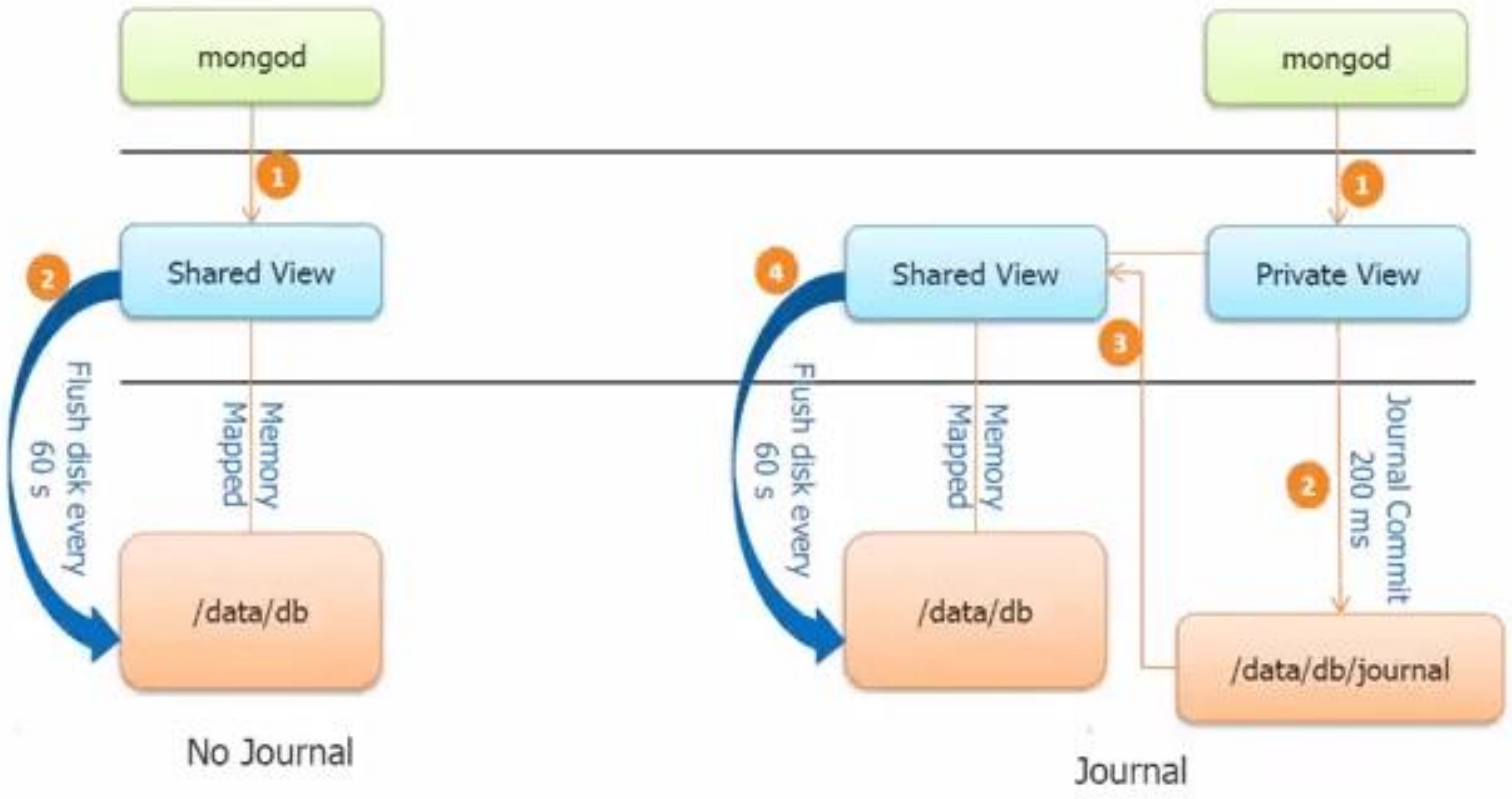
For example, an update to a document in a collection may result in modifications to the indexes;

WiredTiger creates a single journal record that includes both the update operation and its associated index modifications.

Each record has a unique identifier.

The minimum journal record size for WiredTiger is 128 bytes.

Journaling Mechanics



Working with Mongod Write Operation

Mongod primarily hosts the write operations in memory in shared view. It is called shared because it has memory mapping in actual disc.

For example, the user's data file is kept at data dd and, it has a memory mapping. Here, it first pushes all data to memory mapping and after a specified interval it flashes the data into memory, which occurs every sixty seconds and the user is not impacted in this process.

Here, this process is called the No Journal Option, which means that in case there is a 60 second delay to save data from memory to disc or abrupt shutdown, it means that whatever data is in memory may not be retrieved.

Thus, Journaling becomes relevant here.

Journaling in mongodb

In this process, a write operation occurs in mongod, which then creates changes in private view.

After a specified interval, which is called a 'journal commit interval', the private view writes those operations in journal directory (residing in the disc).

Once the journal commit happens, mongod pushes data into shared view.

As part of the process, it gets written to actual data directory from the shared view (as this process happens in background). The basic advantage is, we have a reduced cycle from 60 seconds to 200 milliseconds.

Journaling in mongodb

- In a scenario where an abruption occurs at any point of time or flash disc remains unavailable for last 59 seconds (keeping in mind the existing data in journal directory/write operations), then when the next time mongod starts, it basically replays all write operation logs and writes into the actual data directory.

Journaling in mongodb

Here, once a commit happens, the same operation is replayed in shared view and then, after sixty seconds, the flash disc happens.

After it is flashed, the data is processed. The data here is marked as processed in the journal directory, which means that every sixty seconds, it checks the data it has copied and those that are supposed to be removed from journal.

Using Journaling is like using a log, the reason being, it creates a write operation log to increase the durability. Journaling is temporary storage, which means it keeps only write operation log as pending in journal directory. Also, the shared view has the data but journal directory has the operations.

Link between Private View & Shared View

After the commit happens, it is marked as a process in journal directory, and there is another mapping done for current view of shared/private view (with no data sharing).

In the chart, all the blue items are in RAM (random access memory) and the Saffron denotes the disc.

If in case, the data is not flashed in data directory but write operations are there in data directory, then mongod will reprocess and apply write operations to the data directory.

An important point to note is that in a scenario where a crash happens before journal commit, the data which was appended within 200 milliseconds will be lost.

Points to note about journaling

One can also have Journaling in the background as an asynchronous process and not doing anything in the operations in a synchronous fashion. Journaling is recommended in production also.

Journal commit interval time frame of ‘200 milliseconds’ is configurable, which can be enabled with ‘ – – journal commit interval’ anywhere between 3 to 300 milliseconds, which all depends on the non-functioning requirements (how frequently writes are happening and how frequently one wants to write in journal directory). In case, heavy write operations are going on, then it is advisable to have lesser milliseconds.

Private view holds actual data as private is mapped with shared view. The shared view here flashes it to data directory.

In this process, the advantage we gain is, in case, we have server crashes and there is no data available that needs to be written on flashes, then the next server restarting mongod will check the journal directory for recovery. It will recover, replay and write operations in data directory and then it starts.

Compression

- By default, MongoDB configures WiredTiger to use snappy compression for its journaling data.
- To specify a different compression algorithm or no compression, use the storage.wiredTiger.engineConfig.journalCompressor setting.

Compression

- WiredTiger can compress collection data using one of the following compression library:

Snappy - Provides a lower compression rate than zlib or zstd but has a lower CPU cost than either.

Zlib -Provides better compression rate than snappy but has a higher CPU cost than both snappy and zstd.

zstd (Available starting in MongoDB 4.2) -Provides better compression rate than both snappy and zlib and has a lower CPU cost than zlib.

- By default, WiredTiger uses snappy compression library.
- WiredTiger uses prefix compression on all indexes by default.

Change WiredTiger Journal Compressor

- With the WiredTiger storage engine, MongoDB, by default, uses the snappy compressor for the journal.
- To specify a different compressions algorithm or no compression for a mongod instance
- Update the storage.wiredTiger.engineConfig.journalCompressor setting to the new value.
- If you use command-line options instead of a configuration file, you will have to update the --wiredTigerJournalCompressor command-line option during the restart below.
- Perform a clean shutdown of the mongod instance. For example, connect a mongo shell to the instance and issue db.shutdownServer():
- db.getSiblingDB('admin').shutdownServer()
- Once you have confirmed that the process is no longer running, restart the mongod instance:
- If you are using a configuration file:

```
mongod -f <path/to/myconfig.conf>
```

- If you are using command-line options instead of a configuration file, update the --wiredTigerJournalCompressor option.

```
mongod --wiredTigerJournalCompressor <differentCompressor|none>  ...
```

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.

Journaling and the In-Memory Storage Engine

- Starting in MongoDB Enterprise version 3.2.6, the In-Memory Storage Engine is part of general availability (GA).
- Because its data is kept in memory, there is no separate journal. Write operations with a write concern of j: true are immediately acknowledged.

Disable Journaling

- As a best practice Do not disable journaling on production systems.
- 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.
- To disable journaling for a standalone deployment, start mongod with the --nojournal command line option.

Monitor Journal Status

- The serverStatus command/db.serverStatus() method returns wiredTiger.log, which contains statistics on the journal.

Recover Data After Unexpected Shutdown

- On a restart after a crash, MongoDB replays all journal files in the journal directory before the server becomes available.
- If MongoDB must replay journal files, mongod notes these events in the log output.