Understanding Journaling in MongoDB

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 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.

It is important to know that Journaling was disabled before version 2.4.10 by default but after that it has been enabled.

The moment the mongod process starts, the following statement can be observed:

Journal dir = D:\Rana2\custom data\journal

Here, Journal Directory is a child directory inside data directory and by default is enabled.

What is Journaling in MongoDB?

In this process, a write operation occurs in mongod, which then creates changes in private view. The first block is memory and the second block is 'my disc'. 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.

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.

How it works?

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.

For example, if the user is writing some data without Journaling, then whatever data is written, its memory mapping lets the user know the location where the data is written.

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.

Also note that in the journal directory, we keep writing the actual operation.

In the example statement, such as *'Db.class.insert'* which is an insert operation, the data in inserted into class operations. So the class operation does not actually stay but the operation resides.

It must also be observed that if there is a delay in using journal, it impacts the performance.

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.

Secondly the 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.

Also note that 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.