目录

mongodb-src-r2.6.12\docs\errors.mid文件中有所有错误信息列表。

**为什么\_deleteList不按照大小排序DeleteRecord**

<https://github.com/mongodb/mongo/blob/v2.6/src/mongo/db/structure/catalog/namespace_details.h#L57>

**问题**：I have a question about allocing space for new record from deleteList arrays-list in MongoDB. I looked at the \_\_stdAlloc function code in namespace\_details.h and namespace\_details.cpp. Mongodb's alloc strategy is to try five times for bestmatch DeletedRecord. I don't understand why we don't sort DeletedRecord-size for list. Sorted list is quickly for allocing space for record. Why don't mongodb sort deletedRecord of deleteList backward list by size?

Notes:

1. namespace\_details.h,namespace\_details.cpp;
2. MongoDB source code version is 2.6.12.

**解答**：This is mostly a historical consideration: allocation strategies and storage engines have changed since MongoDB 2.6 (which [reached end-of-life](https://www.mongodb.com/blog/post/mongodb-2-6-end-of-life) in October, 2016). MongoDB 3.0 introduced a storage engine API and the WiredTiger storage engine which is now the default for new deployments as of MongoDB 3.2. The MMAPv1 storage engine is still available as of MongoDB 3.4, but development focus for new features and support has moved on to the WiredTiger storage engine.

The general MMAP record allocation strategy is to try to quickly find a "good enough" record allocation without necessarily finding the ideal allocation to maximise storage reuse. The free list is organised into [unsorted linked list "buckets" grouping records of similar sizes](https://github.com/mongodb/mongo/blob/v2.6/src/mongo/db/structure/catalog/namespace_details.h#L57). Maintaining a sorted list is Big-O costly vs O(1) for inserting and deleting from an unsorted linked list: the MMAP implementation is straightforward and consistently fast.

The original MMAP allocation strategy was to allocate record space for documents with a variable amount of padding based on historical document growth for the current collection. This approach can lead to a large number of record sizes in each free list bucket depending on the application use case. With this allocation strategy, sorted free lists would be helpful to ensure efficient storage reuse but the overall performance impact & benefit would be uncertain without comparing real implementations and use cases.

MongoDB 2.2 added a new Power of 2 allocation strategy (which became the default for MMAP in [MongoDB 3.0](https://docs.mongodb.com/v3.0/core/mmapv1/#power-of-2-allocation)). The Power Of 2 strategy quantizes record allocations into predetermined sizes (eg. 32, 64, 128 bytes, ...) which simplifies reuse of free space and ensures all allocations are O(1). This addresses the challenge of storage reuse in the original padding strategy while still providing predictable performance. Given fixed record sizes, I don't think sorted free lists are likely to be a worthwhile tradeoff in complexity vs performance for the current MMAP implementation.

Since MongoDB is open source, you could always [fork the repo on Github](https://github.com/mongodb/mongo) and try making & testing the implementation changes. For more information, see the guide to [Contributing to the MongoDB Server](https://github.com/mongodb/mongo/wiki).

**MongoDB源码中有\_view\_write，为什么还有\_view\_private？**

<http://stackoverflow.com/questions/40690428/in-mongodb-source-code-what-is-the-usage-of-view-private/40696251#40696251>

**问题**：I am reading mongodb source code. I have a question about memory-map-file.There are \_view\_write and \_view\_private in class DurableMappedFile. The \_view\_write is used for saving a pointer of mapped. The \_view\_private also save a pointer of mapped, but the mapped pointer is MAP\_PRIVATE type. The MAP\_PRIVATE is copy-on-write. I don't know What occasion will be used pointer of MAP\_PRIVATE. Note:The class DurableMappedFile is defined in durable\_mapped\_file.h/cpp.MongoDB source code version is 2.6.12.

**解答**：The reason that we have a private copy on write view of the memory mapped file is that we cannot risk the OS writing modified areas of the memory-mapped file back to disk before the entire write transaction (WriteUnitOfWork) is complete and written to the journal. Only after the journal is durable on disk (synced) is it safe to start writing the data to disk. We do that by copying the data back on the write view. Then periodically we remap our private view so we don't keep duplicate copies indefinitely.See [here](https://github.com/mongodb/mongo/blob/2bd286acef2fdb035f1d45253f6e6e4c24a2dc04/src/mongo/db/storage/mmap_v1/dur.cpp#L30-L52) for more detail.Doing updates this way ensures that replaying the journal after a hard crash (unclean shutdown) will leave the data in a consistent state.

we want to grab from the front so our next pointers on disk tend to go in a forward direction which is important for performance.

断句分析：

we want to grab from the front

so our next pointers on disk tend to go in a forward direction which is important for performance.

后续指针 顺序读取 对于性能很重要。

我们从头读取，这样后续的指针就可以（在磁盘上）顺序移动，这对于性能来讲非常有利。

idea is to reduce quantization overhead of large records at the cost of increasing

the DeletedRecord size distribution in the largest bucket by factor of 4.

百度人工翻译：

我们希望从前端进行操作，这样的话磁盘的后续指针就可能会往前运动，这对于性能很重要（即提升性能）。我们的想法是减少大记录（占磁盘空间大的记录）的量化开销，不过这样做的代价是需要将最大bucket（桶）中的DeletedRecord的大小增加到原来的4倍。 //Grap这里可以理解为“进行操作”，也可以表示“抓住”，视上下文而定。 size distribution 即为粒度分布或大小分布。 DeleteRecord可以保留不翻译，应该为删除链表里用来存储删除的记录的一个结构。 Buket应为为存储桶，一般为数组形式。 就原文来看，主要说明的是如何提升磁盘的（读写）性能。因删除操作可能会产生很多不能重复利用的”存储碎片”，从而导致存储空间大量浪费。

1. 参考资料

http://www.cnblogs.com/cswuyg/p/4355948.html