* [首页](https://km.woa.com/?kmref=km_header)
* [发现](https://km.woa.com/discovery?kmref=km_header)
* [悦读](https://km.woa.com/read?kmref=km_header)
* [乐问](https://km.woa.com/q?kmref=km_header)
* [直播](https://tencent.lexiangla.com/lives?company_from=tencent&kmref=km_header)
* [应用](javascript:void(0))
* [我的K吧](javascript:void(0))
* [[](https://km.woa.com/user/rickyrqzhao)](https://km.woa.com/user/rickyrqzhao)

窗体顶端



窗体底端

[](https://km.woa.com/group/tmecibdev)[TME中央信息库开发](https://km.woa.com/group/46471)

* [首页](https://km.woa.com/group/46471)
* [团队微博](https://km.woa.com/group/46471/twitters)
* [团队讨论](https://km.woa.com/group/46471/topics)
* [团队文章](https://km.woa.com/group/46471/articles)
* [团队文档](https://km.woa.com/group/46471/docs)
* [团队活动](https://km.woa.com/group/46471/events)
* [团队投票](https://km.woa.com/group/46471/surveys)
* [团队日历](https://km.woa.com/group/46471/calendars)
* [团队会议](https://km.woa.com/group/46471/meetings)

目录

1 MongoDB 特点

2 MongoDB 要素

3 MongoDB 数据库

4 MongoDB 集合

5 MongoDB 视图

6 MongoDB 索引

7 MongoDB ObjectId

8 MongoDB 复制集

8.1 复制集节点类型

8.2 复制集选主

8.3 复制集作用

9 MongoDB 分片集

9.1 分片集组件

9.2 分片键

9.3 分片策略

10 MongoDB 聚合

10.1 聚合管道

10.2 map-reduce

10.3 单一目的聚合方法

11 MongoDB 一致性

11.1 因果一致性

11.2 线性一致性

11.3 read concern

11.4 write concern

12 MongoDB WiredTiger 引擎

12.1 WiredTiger 数据结构

12.2 WiredTiger 压缩

12.3 WiredTiger 一致性原理

12.4 WiredTiger MVCC

13 MongoDB 数据读写

13.1 读偏好 ReadPerference

13.2 在复制集上进行读写操作

13.3 在分片集群上进行读写操作

14 MongoDB 事务

14.1 ACID 特性

14.2 事务的使用限制

14.3 事务与 read concern

14.4 事务与 write concern

15 MongoDB Change Stream

15.1 变更流使用场景

15.2 变更流特点

15.3 变更流监听事件类型

15.4 变更流故障恢复

15.5 变更流使用限制

16 MongoDB 性能问题定位方式

[[](https://km.woa.com/base/headlines)](https://km.woa.com/base/headlines" \t "_blank)

MongoDB 基础浅谈

* [hazenweng](https://km.woa.com/user/hazenweng)

* 2021年08月31日 12:06

* 浏览(2355)

* [收藏(233)](javascript:void(0);)

* [评论(21)](javascript:void(0);)

* [分享](javascript:void(0);)

| 导语 MongoDB 作为一款优秀的基于分布式文件存储的 NoSQL 数据库，在业界有着广泛的应用。下文对 MongoDB 的一些基础概念进行简单介绍。

**1 MongoDB 特点**

* 面向集合存储：MongoDB 是面向集合的，数据以 collection 分组存储。每个 collection 在数据库中都有唯一的名称。
* 模式自由：集合的概念类似 MySQL 里的表，但它不需要定义任何模式。
* 结构松散：对于存储在数据库中的文档，不需要设置相同的字段，并且相同的字段不需要相同的数据类型，不同结构的文档可以存在同一个 collection 里。
* 高效的二进制存储：存储在集合中的文档，是以键值对的形式存在的。键用于唯一标识一个文档，一般是 ObjectId 类型，值是以 BSON 形式存在的。BSON = Binary JSON， 是在 JSON 基础上加了一些类型及元数据描述的格式。
* 支持索引：可以在任意属性上建立索引，包含内部对象。MongoDB 的索引和 MySQL 的索引基本一样，可以在指定属性上创建索引以提高查询的速度。除此之外，MongoDB 还提供创建基于地理空间的索引的能力。
* 支持 mapreduce：通过分治的方式完成复杂的聚合任务。
* 支持 failover：通过主从复制机制，可以实现数据备份、故障恢复、读扩展等功能。基于复制集的复制机制提供了自动故障恢复的功能，确保了集群数据不会丢失。
* 支持分片：MongoDB 支持集群自动切分数据，可以使集群存储更多的数据，实现更大的负载，在数据插入和更新时，能够自动路由和存储。
* 支持存储大文件：MongoDB 中 BSON 对象最大不能超过 16 MB。对于大文件的存储，BSON 格式无法满足。GridFS 机制提供了一个存储大文件的机制，可以将一个大文件分割成为多个较小的文档进行存储。

**2 MongoDB 要素**

* database: 数据库。
* collection: 数据集合，相当于 MySQL 的 table。
* document: 数据记录行，相当于 MySQL 的 row。
* field: 数据域，相当于 MySQL 的 column。
* index: 索引。
* primary key: 主键。

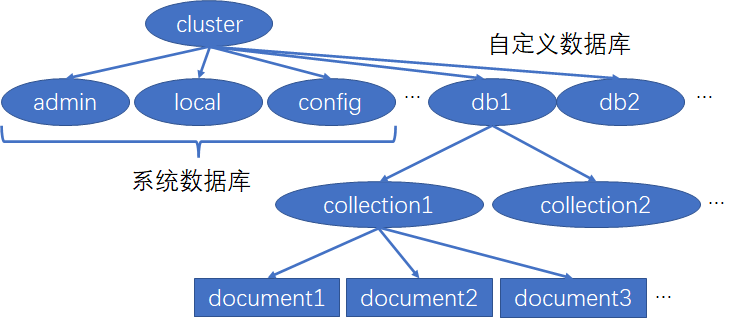
**3 MongoDB 数据库**

一个 MongoDB 实例可以创建多个 database。连接时如果没开启免认证模式的话，需要连接到 admin 库进行认证。如果开启免认证模式，若不指定 database 进行连接，默认连接一个叫 db 的数据库，该数据库存储在 data 目录中。通过 show dbs 命令可以查看所有的数据库。数据库名不能包含空字符。数据库名不能为空并且必须小于 64 个字符。

MongoDB 预留了几个特殊的 database。

* admin: admin 数据库主要是保存 root 用户和角色。例如，system.users 表存储用户，system.roles 表存储角色。一般不建议用户直接操作这个数据库。将一个用户添加到这个数据库，且使它拥有 admin 库上的名为 dbAdminAnyDatabase 的角色权限，这个用户自动继承所有数据库的权限。一些特定的服务器端命令也只能从这个数据库运行，比如关闭服务器。
* local: local 数据库是不会被复制到其他分片的，因此可以用来存储本地单台服务器的任意 collection。一般不建议用户直接使用 local 库存储任何数据，也不建议进行 CRUD 操作，因为数据无法被正常备份与恢复。
* config: 当 MongoDB 使用分片设置时，config 数据库可用来保存分片的相关信息。

一个 MongoDB 实例的数据结构如下图：



**4 MongoDB 集合**

MongoDB 集合存在于数据库中，没有固定的结构，可以往集合插入不同格式和类型的数据。集合不需要事先创建。当第一个文档插入，或者第一个索引创建时，集合就会被创建。集合名必须以下划线或者字母符号开始，并且不能包含 $，不能为空字符串（比如 ""），不能包含空字符，且不能以 system. 为前缀。

capped collection 是固定大小的集合，支持高吞吐的插入操作和查询操作。它的工作方式与循环缓冲区类似，当一个集合填满了被分配的空间，则通过覆盖最早的文档来为新的文档腾出空间。和标准的 collection 不同，capped collection 需要显式创建，指定大小，单位是字节。capped collection 可以按照文档的插入顺序保存到集合中，而且这些文档在磁盘上存放位置也是按照插入顺序来保存的，所以更新 capped collection 中的文档，不可以超过之前文档的大小，以便确保所有文档在磁盘上的位置一直保持不变。

**5 MongoDB 视图**

视图基于已有的集合进行创建，是只读的，不实际存储硬盘，通过视图进行写操作会报错。视图使用其上游集合的索引。由于索引是基于集合的，所以你不能基于视图创建、删除或重建索引，也不能获取视图的索引列表。如果视图依赖的集合是分片的, 那么视图也视为分片的。视图是实时计算并读取的。

**6 MongoDB 索引**

MongoDB 支持丰富的索引方式。如果没有索引，读操作就必须扫描集合中的每个文档并筛选符合查询条件的记录。索引能够在很大程度上提高查询速度。

* 单字段索引：有三种方式，（1）在单个字段上创建索引；（2）在嵌入式字段上创建索引；（3）在内嵌文档上创建索引。
* 复合索引：支持在多个字段上匹配的查询。对任何复合索引施加 32 个字段的限制。对于复合索引，MongoDB 可以使用索引来支持对索引前缀的查询。
* 多键索引：为了索引包含数组值的字段，MongoDB 为数组中的每个元素创建一个索引键。这些多键索引支持对数组字段的高效查询。
* 文本索引：支持对字符串内容的文本搜索查询。文本索引可以包含任何值为字符串或字符串元素数组的字段。一个集合最多可以有一个文本索引。
* 通配符索引：支持针对未知或任意字段的查询。例如：db.collection.createIndex( {"a.$\*\*" : 1 } ) 可支持诸如 db.collection.find({ "a.b" : 1 })、db.collection.find({ "a.c" : { $lt : 2 } }) 等查询，提高查询效率。不能使用通配符索引来分片集合。不能为通配符创建复合索引。
* 通配符文本索引：通配符文本索引不同于通配符索引。通配符索引不支持使用$text操作符的查询。通配符文本索引为集合中每个文档中包含字符串数据的每个字段建立索引。索引的创建方式示例：db.collection.createIndex( { "$\*\*": "text" } )。
* 2dsphere 索引：支持球体上的地理空间查询：包含、相交和邻近度查询。
* hashed 索引：支持使用哈希的分片键进行分片。基于哈希的分片使用字段的散列索引作为分片键，以便跨分片集群对数据进行分区。MongoDB 支持任何单个字段的哈希索引，但不支持创建具有多个哈希字段的复合索引，也不能在索引上指定唯一哈希索引。
* ttl 索引：一种特殊的单字段索引，支持在一定的时间或特定的期限后自动从集合中删除文档。TTL索引不能保证过期数据在过期时立即删除。默认每60秒运行一次删除过期文档的后台进程。capped collection 不支持 ttl 索引。
* 唯一索引：确保索引字段不会存储重复值。如果集合已经存在了违反索引的唯一约束的文档，则后台创建唯一索引会失败。
* 部分索引：只索引集合中满足指定筛选器表达式的文档。例如：db.collection.createIndex({ a:1 },{ partialFilterExpression: { b: { $lt: 100 } } }) 表示只对集合中 b 字段小于 100 的文进行索引，大于等于 100 的文档不会被索引。这可以有效提高存储效率。
* 稀疏索引：只包含有索引字段的文档的条目，即使索引字段包含空值。索引会跳过任何缺少索引字段的文档。非稀疏索引包含集合中的所有文档，为那些不包含索引字段的文档存储空值。

**7 MongoDB ObjectId**

ObjectId 可以快速生成并排序，长度为 12 个字节，包括：

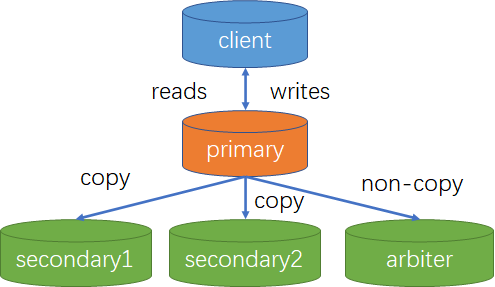
* 一个 4 字节的时间戳，表示 unix 时间戳
* 5 字节随机值
* 3 字节递增计数器，初始化为随机值

在 MongoDB 中，存储在集合中的每个文档都需要一个唯一的 \_id 字段作为主键。如果插入的文档省略了 \_id 字段，则自动为文档生成一个 \_id。

**8 MongoDB 复制集**

MongoDB 的复制集又称为副本集（Replica Set），是一组维护相同数据集合的 mongod 进程。复制集包含多个数据节点和一个可选的仲裁节点（arbiter）。在数据节点中，有且仅有一个成员为主节点（primary），其他节点为从节点（secondary）。

一个典型的复制集架构图如下：



**8.1 复制集节点类型**

* 主节点：接收所有的写操作，并将集合所有的变化记录到操作日志中，即 oplog。
* 从节点：通过复制主节点的操作来维护一个相同的数据集。从节点有几个选配项：v 参数决定是否具有投票权；priority 参数决定节点选主过程时的优先级；hidden 参数 决定是否对客户端可见；slaveDelay 参数表示复制 n 秒之前的数据，保持与主节点的时间差。从节点可以配置成 0 优先级，阻止它在选举中成为主节点，适用于将该节点部署在备用数据中心，或者将它作为一个冷节点；可以配置为隐藏复制集，防止应用程序从它读取数据，适用于在该节点上运行需要与正常流量分离的程序；可以配置为延迟复制集，保持一个历史快照，以便做按特定时间的故障恢复。
* 仲裁节点：如果将一个 mongod 实例作为仲裁节点添加到一个复制集中，该节点可以参与主节点选举，但不保存数据。仲裁节点永远只能是仲裁节点。

**8.2 复制集选主**

MongoDB 的副本集协议（又称为 pv1)，是一种 raft-like 协议，即基于 raft 协议的理论思想实现，并且对之进行了一些扩展。当往复制集添加一个节点，或当主节点无法和集群中其他节点通信的时间超过参数 electionTimeoutMillis 配置的期限时，从节点会尝试通过 pv1 协议发起选举来推荐自己成为新主节点。

在选举前具有投票权的节点之间两两互相发送心跳，以侦测节点是否存活。复制集节点每两秒向彼此发送心跳。如果心跳未在 10 秒内返回，则发送心跳的一方将被发送方标记为不可访问，也就是说，默认当5次心跳未收到时判断为节点失联。如果失联的是主节点，从节点会发起选举，选出新的主节点； 如果失联的是从节点则不会产生新的选举。选举基于 raft 一致性算法实现，在大多数投票节点存活下选举出主节点。只有能够与多数节点建立连接且具有较新的 oplog 的节点才可能被选举为主节点，如果集群里的节点配置了优先级，那么具有较高的优先级的节点更可能被选举为主节点。

复制集中最多可以有50个节点，但具有投票权的节点最多7个。

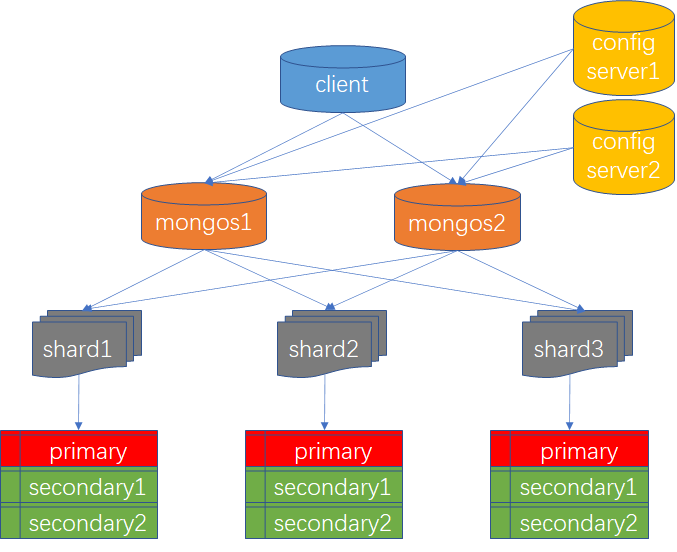
**8.3 复制集作用**

* 主节点发生故障时自动选举出一个新的主节点，以实现 failover。
* 将数据从一个数据中心复制到另一个数据中心，减少另一个数据中心的读延迟。
* 实现读写分离。
* 实现容灾，可以在数据中心故障时快速切换到同城或异地的数据中心。

**9 MongoDB 分片集**

MongoDB 支持通过分片技术来支持海量数据存储。解决数据增长的扩展方式有两种：垂直扩展和水平扩展。垂直扩展通过增加单个服务器的能力来实现，比如磁盘空间、内存容量、CPU 数量等；水平扩展则通过将数据存储到多个服务器上来实现。MongoDB 通过分片实现水平扩展。

一个典型的分片集群架构如下：



**9.1 分片集组件**

* shard：每个分片上可以保存一个集合的子集，所有分片上的子集的数据互不相交，构成完整的集合。每个分片可以被部署为复制集架构。最大为 1024 个分片。
* mongos：充当查询路由器，在客户端和分片集之间提供读写接口。mongos 提供集群单一入口，转发应用端请求，选择合适的数据节点进行读写，合并多个数据节点的返回。 mongos 是无状态的，分片集群一般需要配置至少 2 个 mongos。
* config server：存储分片集的相关配置信息。

**9.2 分片键**

MongoDB 集合若要采用分片，必须要指定分片键（shard key）。分片键由文档中的一个或多个字段组成。分片集合必须具有支持分片键的索引，索引可以是分片键的索引，也可以是以分片键是索引前缀的复合索引。要对已填充的集合进行分片，该集合必须具有以分片键开头的索引；分片一个空集合时，如果该集合还没有包含指定分片键的索引，则 MongoDB 会默认给分片键创建索引。

对于一个即将要分片的集合，如果该集合具有其他唯一索引，则无法分片该集合。

对于已分片的集合，不能在其他字段上创建唯一索引。

4.2 版本开始可以更改文档的分片键值，除非分片键字段为不可变的 \_id 字段。更新分片键时必须在事务中或以可重试写入的方式在 mongos 上运行，不能直接在分片上执行操作。在此之前文档的分片键字段值是不可变的。

4.4 版本开始，可以向现有片键中添加一个或多个后缀字段以优化集合的片键。

5.0 版本开始，实现了实时重新分片（live resharding），可以实现分片键的完全重新选择。live resharding 机制下，数据将根据新的分片规则进行迁移，不过有一些限制，比如一个实例中有且只能有一个集合在相同的时间下 resharding 等。

数据库可以混合使用分片和未分片集合。分片集合被分区并分布在集群中的各个分片中。而未分片集合仅存储在主分片中。

设置 shard key 时应该充分考虑取值基数和取值分布。分片键应被尽可能多的业务场景用到。尽可能避免使用单调递增或递减的字段作为分片键。

**9.3 分片策略**

MongoDB 将分片数据拆分成块。每个分块都有一个基于分片键的上下限范围 。分片策略包括哈希分片、范围分片和自定义 zone 分片。

* 哈希分片会计算分片键字段的哈希值，这个值被用作片键，然后根据哈希值的散列为每个块分配一个范围。
* 范围分片根据分片键的值将数据划分为多个连续范围。，然后基于分片键的值分配每个块的范围。当片键的基数大、频率低且值非单调变更时，范围分片更高效。
* 自定义 zone 分片基于 shard key 创建。每个 zone 与集群中的一个或者更多分片关联。一个分片可以和任意数目的非冲突 zone 相关联。

**10 MongoDB 聚合**

MongoDB 聚合框架（Aggregation Framework）是一个计算框架，功能是：

* 作用在一个或几个集合上。
* 对集合中的数据进行的一系列运算。
* 将这些数据转化为期望的形式。

MongoDB 提供了三种执行聚合的方法：聚合管道，map-reduce 和单一目的聚合方法（如 count、distinct 等方法）。

**10.1 聚合管道**

在聚合管道中，整个聚合运算过程称为管道（pipeline），它是由多个步骤（stage）组成的， 每个管道的工作流程是：

1. 接受一系列原始数据文档
2. 对这些文档进行一系列运算
3. 结果文档输出给下一个 stage

聚合计算基本格式如下：

pipeline = [$stage1, $stage2, ...$stageN];

db.collection.aggregate( pipeline, { options } )

**10.2 map-reduce**

map-reduce 操作包括两个阶段：map 阶段处理每个文档并将 key 与 value 传递给 reduce 函数进行处理，reduce 阶段将map 操作的输出组合在一起。map-reduce 可使用自定义 JavaScript 函数来执行 map 和 reduce 操作，以及可选的 finalize 操作。通常情况下效率比聚合管道低。

**10.3 单一目的聚合方法**

主要包括以下三个：

* db.collection.estimatedDocumentCount()
* db.collection.count()
* db.collection.distinct()

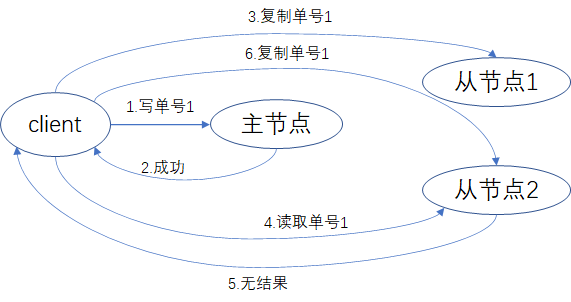
**11 MongoDB 一致性**

分布式系统有个 PACELC 理论。根据 CAP，在一个存在网络分区（P）的分布式系统中，要面临在可用性（A）和一致性（C）之间的权衡，除此之外（E），即使没有网络分区的存在，在实际系统中，我们也要面临在访问延迟（L）和一致性（C）之间的权衡。MongoDB 的一致性模型对读写操作 L 和 C 的选择提供了丰富的选项。

**11.1 因果一致性**

单节点的数据库由于为读写操作提供了顺序保证，因此实现了因果一致性。分布式系统同样可以提供这些保证，但必须对所有节点上的相关事件进行协调和排序。

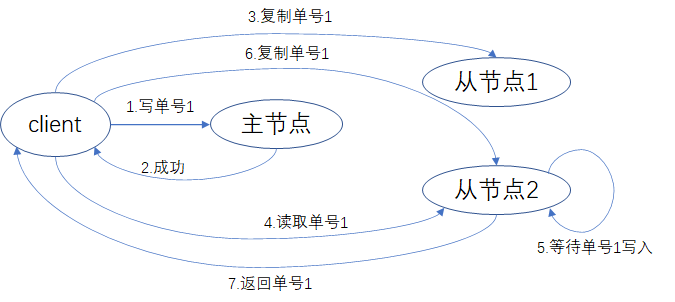
以下是一个不遵循因果一致性的例子：



为了保持因果一致性，必须有以下保证：

|  |  |
| --- | --- |
| Read your writes | 读操作必须能够反映出在其之前的写操作。 |
| Monotonic reads | 如果某个读操作已经读取到某个对象的某个值，那么任何后续访问都不应该返回在此之前的值。 |
| Monotonic writes | 如果某个写操作先于其它写操作执行，写操作顺序不应受到任何其他条件打乱。 |
| Writes follow reads | 如果某个写操作发生在读操作之后，则该写操作将等待读操作完成后进行。 |

实现因果一致性的单号读写应遵循以下流程：



为了建立复制集和分片集事件的全局偏序关系，MongoDB 实现了一个逻辑时钟，称为 lamport logical clock。每个写操作在应用于主节点时都会被分配一个时间值。这个值可以在副本和分片之间进行比较。从驱动到查询路由器再到数据承载节点，分片集群中的每个成员都必须在每条消息中跟踪和发送其最新时间值，从而允许分片之间的每个节点在最新时间保持一致。主节点将最新的时间值赋值给后续的写入，这为任何一系列相关操作创建了一个因果顺序。节点可以使用这个因果顺序在执行所需的读或写之前等待，以确保它在另一个操作之后发生。

从 MongoDB 3.6 开始，在客户端会话中开启因果一致性，保证 read concern 为 majority 的读操作和 write concern 为 majority 的写操作的关联序列具有因果关系。应用程序必须确保一次只有一个线程在客户端会话中执行这些操作。

对于因果相关的操作：

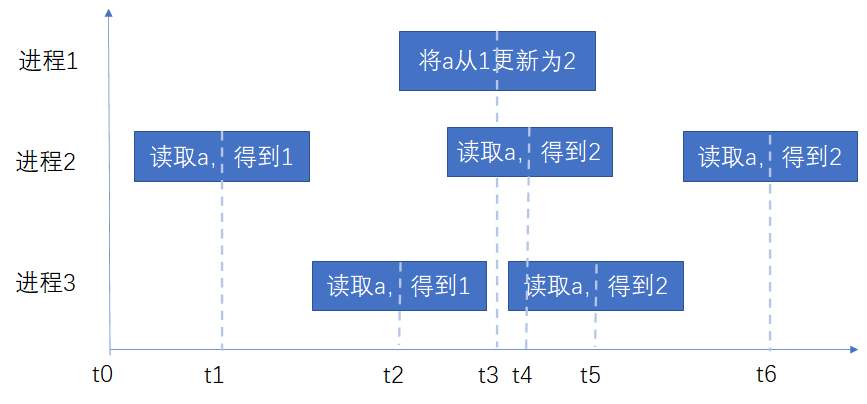
1. 客户端开启客户端会话，需满足以下条件：read concern 为 majority，数据已被大多数复制集成员确认并且是持久化的；write concern 为 majority，确认该操作已应用于复制集中大多数可投票成员。
2. 当客户端发出 read concern 为 majority 的读操作和 write concern 为 majority 的写操作的序列时，客户端将会话信息包含在每个操作中。
3. 对于与会话相关联的每个 read concern 为 majority 的读操作和 write concern 为 majority 的写操作，即使操作出错，MongoDB 也会返回操作时间和集群时间。
4. 相关的客户端会话会跟踪这两个时间字段。

**11.2 线性一致性**

线性一致性又被称为强一致性。CAP 中的 C 指的就是线性一致性。顺序一致性中进程只关心各自的顺序一样就行，不需要与全局时钟一致。线性一致性是顺序一致性的进化版，要求顺序一致性的这种偏序（partial order）要达到全序（total order）。

在实现了线性一致性的系统中，任何操作在该系统生效的时刻都对应时间轴上的一个点。把这些时刻连接成一条线，则这条线会一直沿时间轴向前，不会反向。任何操作都需要互相比较决定发生的顺序。

以下是一个线性一致性的系统示例：



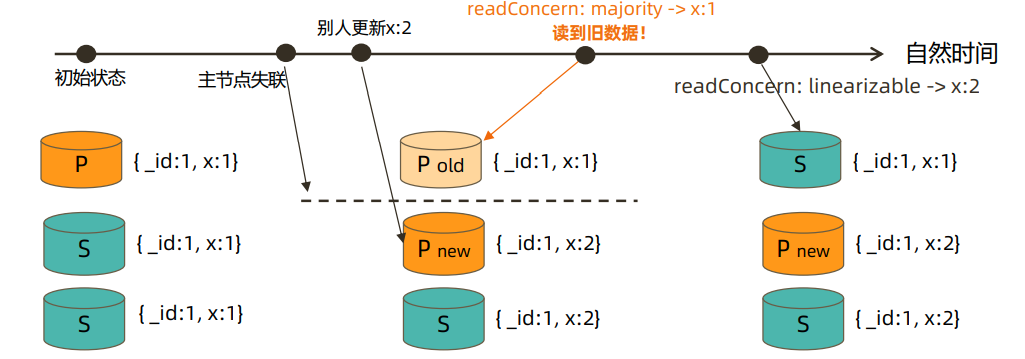
在以上系统中，写操作生效之前的任何时刻，读取值均为 1，生效后均为 2。也就是说，任何读操作都能读到某个数据的最近一次写的数据。系统中的所有进程看到的操作顺序，都遵循全局时钟的顺序。

**11.3 read concern**

read concern 是针对读操作的配置。它控制读取数据的新近度和持久性。read concern 选项控制数据读取的一致性，分为 local、available、majority、linearizable 四种，它们对一致性的承诺依次由弱到强。其中 linearizable 表示线性一致性，另外 3 种级别代表了 MongoDB 在实现最终一致性时，对访问延迟和一致性的取舍。

* local/available: 语义基本一致，都是读操作直接读取本地最新的数据，但不保证该数据已被写入大多数复制集成员。数据可能会被回滚。默认是针对主节点读。 如果读取操作与因果一致的会话相关联，则针对副节点读。唯一的区别在于，avaliable 在分片集群场景下，为了保证性能，可能返回孤儿文档。
* majority：读取 majority committed 的数据，可以保证读取的数据不会被回滚，但是并不能保证读到本地最新的数据。受限于不同节点的复制进度，可能会读取到更旧的值。当写操作对应的 write concern 配置中 w 的值越大，则写操作在扩散到更多的复制集节点上之后才返回写成功，这时通过 read concern 被配置为 majority 的读操作进行读取数据，就有更大的概率读取到最新的数据。
* linearizable：读取 majority committed 的数据，但会等待在读之前所有的 majority committed 确认。它承诺线性一致性，要求读写顺序和操作真实发生的时间完全一致，既保证能读取到最新的数据，也保证读到数据不会被回滚。只对读取单个文档时有效，且可能导致非常慢的读，因此总是建议配合使用 maxTimeMS 使用。linearizable 只能用在主节点的读操作上，考虑到写操作也只能发生在主节点上，相当于说 MongoDB 的线性一致性被限定在单机环境下实现。实现 linearizable，读取的数据应该是被 write concern 为 majority 的写操作写入到 MongoDB 集群中的、且持久化到日志中的数据。如果数据写入到多数节点后，没有在日志中持久化，当这些节点发生重启恢复，那么之前通过配置 read concern 为 linearizable 的读操作读取到的数据就可能丢失。可以通过 writeConcernMajorityJournalDefault 选项保证指定 write concern 为 majority 的写操作在日志中是否持久化。如果写操作持久化到了日志中，但是没有复制到多数节点，在重新选主后，同样可能会发生数据丢失，违背一致性承诺。
* snapshot: 与关系型数据库中的快照隔离级别语义一致。最高隔离级别，接近于 serializable。是伴随着 MongoDB 4.0 版本中新出现的多文档事务而设计的，只能用在显式开启的多文档事务中。如果事务是因果一致会话的一部分，且 write concern 为 majority，则在事务提交后，读操作可以保证已从多数提交数据的快照中读取，该快照提供与该事务开始之前的操作的因果一致性。它读取 majority committed 的数据，但可能读不到最新的已提交数据。snapshot 保证在事务中的读不出现脏读、不可重复读和幻读。 因为所有的读都将使用同一个快照，直到事务提交为止该快照才被释放。

下面借用一张图展示 majority 和 linearizable 的区别：



**11.4 write concern**

write concern 是针对写操作的配置，表示写请求对独立 mongod 实例或复制集或分片集进行写操作的确认级别。它主要是控制数据写入的持久性。包含三个选项：

* w：指定了写操作需要复制并应用到多少个复制集成员才能返回成功，可以为数字或 majority。
  + w:0 表示客户端不需要收到任何有关写操作是否执行成功的确认，就直接返回成功，具有最高性能。
  + w:1 表示写主成功则返回。
  + w: majority 需要收到多数节点（含主节点）关于操作执行成功的确认，具体个数由 MongoDB 根据复制集配置自动得出。w 值越大，对客户端来说，数据的持久性保证越强，写操作的延迟越大。w:1 要求事务只要在本地成功提交即可，而 w: majority 要求事务在复制集的多数派节点提交成功。
  + w:all 表示全部节点确认才返回成功。
* j：表示写操作对应的修改是否要被持久化到存储引擎日志中，只能选填 true 或 false。
  + j:false 表示写操作到达内存即算作成功。
  + j:true 表示写操作落到 journal 文件中才算成功。w:0 如果指定 j:true，则优先使用 j:true 来请求独立或复制集主副本的确认。j:true 本身并不能保证不会因复制集主故障转移而回滚写操作。
* wtimeout：主节点在等待足够数量的确认时的超时时间，单位为毫秒。超时返回错误，但并不代表写操作已经执行失败。跟 w 有关，比如：w 是 1，则是带主节点确认的超时时间；w 为 0，则永不返回错误；w 为 majority，表示多数节点确认的超时时间。

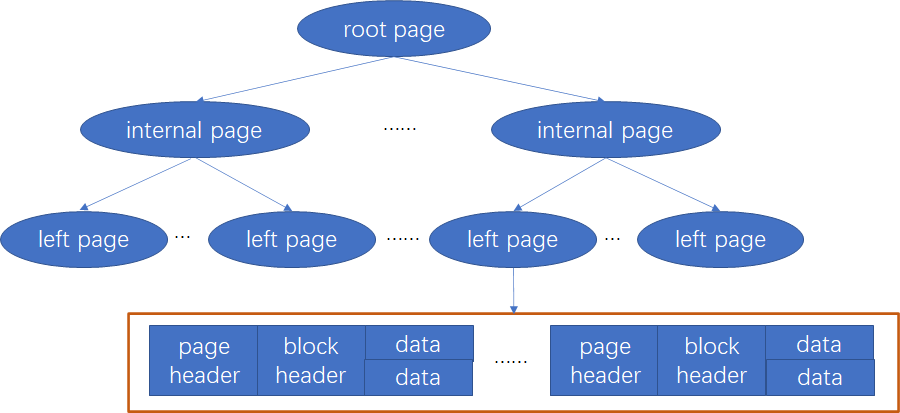
**12 MongoDB WiredTiger 引擎**

从 3.2 版本开始，默认使用 WiredTiger 存储引擎，每个被创建的表和索引，都对应各自独立的 WiredTiger 表。为了保证 MongoDB 中数据的持久性，使用 WiredTiger 的写操作会先写入 cache，并持久化到 WAL（write ahead log），每 60s 或日志文件达到 2 GB，就会做一次 checkpoint，定期将缓存数据刷到磁盘，将当前的数据持久化产生一个新的快照。

**12.1 WiredTiger 数据结构**

MongoDB 采用插件式存储引擎架构，实现了服务层和存储引擎层的解耦，可支持使用多种存储引擎。除此之外，底层的 WiredTiger 引擎还支持使用 B+ 树和 LSM 两种数据结构进行数据管理和存储，默认使用 B+ 树结构做存储。使用 B+ 树时，WiredTiger 以 page 为单位往磁盘读写数据，B+ 树的每个节点为一个 page，包含三种类型的 page，即 root page、internal page 和 leaf page。

以下是 B+ 树的结构示意图：



* root page 是 B+ 树的根节点。
* internal page 是不实际存储数据的中间索引节点。
* leaf page 是真正存储数据的叶子节点，包含页头（page header）、块头（block header）和真正的数据（key-value 对）。page header 定义了页的类型、页存储的记录条数等信息；块头定义了页的校验和 checksum、块在磁盘上的寻址位置等信息。真正的数据由一个 WT\_ROW 结构的数组变量进行存储，每一条记录还有一个 cell\_offset 变量，表示这条记录在 page 上的偏移量。WiredTiger 有一个用来为 page 分配 block 的块设备管理模块。定位文档位置时，先计算 block 的位置，通过 block 的位置找到它对应的 page，再通过 page 找到文档行数据的相对位置。leaf page 为了实现 MVCC，还会维护一个 WT\_UPDATE 结构的数组变量，每条记录对应一个数组元素，每个元素是一个链表，将所有修改值以链表形式保存。

**12.2 WiredTiger 压缩**

WiredTiger 支持在内存和磁盘上对索引进行压缩，通过前缀压缩的方式减少 RAM 的使用。

**12.3 WiredTiger 一致性原理**

WiredTiger 使用了二级缓存 WiredTiger Cache 和 File System Cache 来保证 Disk 上 Database File 数据的最终一致性。

* WiredTiger Cache：通过 B+ 树缓存未压缩的数据，并通过淘汰算法确保内存占用在合理范围内。
* File System Cache：由操作系统管理，缓存压缩后的数据。
* Database File：存储压缩后的数据。每个 WiredTiger 表对应一个独立的磁盘文件。磁盘文件划分成多个按 4 KB 对齐的 extent，并通过 3 个链表来管理：available list（可分配的 extent 列表) ，discard list（废弃的 extent 列表）和 allocate list（当前已分配的 extent 列表）

**12.4 WiredTiger MVCC**

WiredTiger 使用 MVCC 进行写操作，多个客户端可以并发同时修改集合的不同文档。事务开始时，WiredTiger 为操作提供反映内存数据的一致视图的时间点快照。 MVCC 通过非锁机制进行读写操作，是一种乐观并发控制模式。 WiredTiger 仅在全局、数据库和集合级别使用意向锁。当存储引擎检测到两个操作之间存在冲突时，将引发写冲突，从而导致 MongoDB 自动重试该操作。

使用 WiredTiger，如果没有 journal 记录，MongoDB 能且仅能从最后一个检查点恢复。如果需要恢复最后一次 checkpoint 之后所做的更改，那么开启日志是必要的。

**13 MongoDB 数据读写**

**13.1 读偏好 ReadPerference**

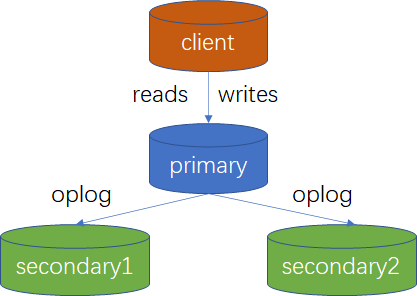
默认情况下，客户端读取复制集主节点上的数据。但客户端可以指定一个 read perference 改变读取行为，以便对复制集上的其他节点进行直接读操作。可选值包括：

|  |  |
| --- | --- |
| primary | 默认模式。从当前复制集主节点读取 。  包含读取操作的多文档事务必须使用该选项，给定事务中的所有操作都必须路由到同一个成员。 |
| primaryPreferred | 在大多数情况下从主节点读取，但如果不可用，则读取从节点上的数据。 |
| secondary | 所有操作都从复制集的从节点读取。 |
| secondaryPreferred | 在大多数情况下读取从节点数据，但如果不可用，则读取主节点上的数据。 |
| nearest | 读操作会在复制集中网络延时最小的节点上进行，随机从合格的复制集成员中读取，不管是否主节点。 |

**13.2 在复制集上进行读写操作**

读操作由客户端指定的 read prefenence 选项决定。

所有的写操作都在集合的主节点上执行。主节点执行写操作并将操作记录在操作日志或 oplog 上。oplog 是 local 数据库的一个集合，叫 local.oplog.rs。这是一个 capped collection，是固定大小，循环使用的。oplog 是对数据集的可重复操作序列，其记录的每个操作都是幂等的，也就是说，对目标数据集应用一次或多次 oplog 操作都会产生相同的结果。从节点从上一次结束时间点建立 tailable cursor，不断的从同步源拉取 oplog 并重放应用到自身，且严格按照原始的写顺序对给定的文档执行写操作。mongodb 使用多线程批量执行写操作来提高并发，根据文档 id 进行分批执行。MongoDB 为了提升同步效率，将拉取 oplog 以及重放 oplog 分到了不同的线程来执行。



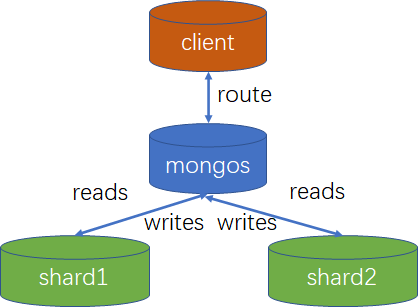
大致的写流程如下：

* producer thread 不断的从主节点上拉取 oplog，并把它加入到一个 blockQueue 里，blockQueue 不是无限容量的，当超过最大存储容量，producer thread 就必须等到 oplog 被 replBatcher thread 从队列里取出后才能继续拉取 oplog。
* replBatcher thread 不断从 producer thread 对应的 blockQueue 里取出 oplog，放到自己的内存队列里，内存队列也不是无限容量，一旦满了，就需要等待被 oplogApplication thread 消费。
* oplogApplication thread 不断取出 replBatch thread 内存队列里的所有元素，分散到不同的 replWriter thread，由 replWriter thread 根据 oplog 进行写操作。等待所有 oplog 都应用完毕，oplogApplication hread 将所有的 oplog 顺序写入到local.oplog.rs 集合。

**13.3 在分片集群上进行读写操作**

对于分片集群，需要一个 mongos 实例提供客户端应用程序和分片集群之间的接口。在客户端看来，该 mongos 实例的行为与其他 MongoDB 实例是相同的。客户端向路由节点 mongos 发送请求，由该节点决定往哪个分片进行读写。对于读取操作，若能定向到特定分片时，效率最高。一般而言，分片集合的查询应包含集合的分片键，以避免低效的全分片查询。在这种情况下，mongos 可以使用配置数据库 config 中的集群元数据信息，将查询路由到分片。如果查询不包含分片键，则 mongos 节点必须将查询定向到集群中的所有分片，然后在 mongos 上聚合所有分片的查询结果，返回给客户端。

对于写操作， mongos 定向到负责数据集特定部分的分片，config 数据库上有集合相关的分片键信息，mongos 从中读取配置，并路由写操作到适当的分片。



**14 MongoDB 事务**

**14.1 ACID 特性**

MongoDB 在一定程度上支持了事务的 ACID 特性。MongoDB 4.0 版本开始支持复制集上的多文档事务，4.2 版本引入了分布式事务，它增加了对分片群集上多文档事务的支持。

* 原子性：成功提交事务时，事务中所有数据更新将完全进行成功，并在事务外部可见。在提交事务之前，事务外部看不到在事务中进行的任何数据更新。当事务被打断或终止时，事务中进行的所有数据更新都将被丢弃，对事务外部完全不可见。但是当事务写入多个分片时，并非所有事务外的读操作都需要等待事务提交后所有分片上数据完全可见。
* 隔离性：MongoDB 提供 snapshot 隔离级别，在事务开始创建一个 WiredTiger snapshot，然后在整个事务过程中，便可以使用这个快照提供事务读。
* 持久性：事务使用 write concern 指定 {j: true} 时，MongoDB 会保证事务日志提交才返回，即使发生 crash，也能根据事务日志来恢复；而如果没有指定 {j: true} 级别，即使事务提交成功了，在故障恢复之后，事务的也可能被回滚掉。
* 一致性：参考前文提到的 MongoDB 一致性。

**14.2 事务的使用限制**

* 仅 WiredTiger 引擎支持事务。
* 对集合的创建和删除操作，不能出现在事务中。
* 对索引的创建和删除操作，不能出现在事务中。
* 不能对系统级别的数据库和集合进行操作。
* 默认情况下，事务大小的限制在 16 MB。
* 默认情况下，事务操作整体不允许超过 60 秒。
* 事务不能在 session 外运行。
* 一个 session 只能运行一个事务，多个 session 可以并行运行事务。
* 不能对 capped collection 进行操作。
* 不能使用 explain 操作做查询分析。

**14.3 事务与 read concern**

事务中的操作使用事务级别的 read concern。事务内部忽略在集合和数据库级别设置的任何 read concern。事务支持设置 read concern 为 local、majority 和 snapshot 其中之一。

* 当 read concern 为 local 时，可读取节点可用的最新数据，但数据可能回滚。对于分片群集上的事务，local 不能保证数据是从整个分片的同一快照视图获取。
* 当 read concern 为 majority 时，如果在提交事务时指定了 write concern 为 majority 级别，则返回大多数副本成员已确认的数据（即无法回滚数据）。如果事务未指定 write concern 为 majority 级别，则不保证读操作可以读取多数提交的数据。对于分片群集上的事务，不能保证数据是从整个分片的同一快照视图中获取。
* 当 read concern 为 snapshot 时，如果在提交事务时指定了 write concern 为 majority 级别，则从大多数已提交数据的快照中返回数据。如果事务未指定 write concern 为 majority 级别，则不保证读操作使用了 majority commited 的数据的快照。对于分片群集上的事务，snapshot 跨分片同步。

**14.4 事务与 write concern**

事务使用事务级别的 write concern 来进行写操作提交，可以通过配置 w 选项设置节点个数，来决定事务写入是否成功，默认情况下为 1。

* w:0 表示事务写入不关注是否成功，默认为成功。
* w:1 表示事务写入到主节点就开始往客户端发送确认写入成功。
* w:majority 表示大多数节点成功原则，例如一个复制集 3 个节点，2 个节点成功就认为本次事务写入成功。
* w:all 表示所有节点都写入成功，才认为事务提交成功。
* j:false 表示写操作到达内存就算事务成功。
* j:true 表示写操作只有记录到日志文件才算事务成功。
* wtimeout: 写入超时时间，过期表示事务失败。

**15 MongoDB Change Stream**

**15.1 变更流使用场景**

MongoDB 3.6 引入了 change stream（变更流）。它的使用场景包括：

* 数据同步：多个 MongoDB 集群之间的增量数据同步。
* 审计：对 MongoDB 操作进行审计、监控。
* 数据订阅：外部程序订阅 MongoDB 的数据变更，可离线数据同步、计算或分析等。

**15.2 变更流特点**

change stream 允许外部程序访问实时数据更改，而不会增加 MongoDB 基础操作的复杂性，也不会导致 oplog 延迟的风险。应用程序可以使用 change stream 来订阅单个集合、数据库或整个集群中的所有数据变更。若要开启 change stream，必须使用 WiredTiger 存储引擎。

change stream 可应用于复制集和分片集。应用于复制集时，可以在复制集中任意一个节点上开启监听；应用于分片集时，则只能在 mongos 上开启监听。在 mongos 上发起监听，是利用全局逻辑时钟提供了整个分片上变更的总体排序，确保监听事件可以按接收到的顺序安全地解释。mongos 会一直检查每个分片，查看每个分片是否存在最新的变更。如果多个分片上一直很少出现变更，则可能会对 change stream 的响应时间产生负面影响，因为 mongos 仍必须检查这些冷分片保持总体有序。

**15.3 变更流监听事件类型**

从 change stream 中能监听到的变更事件包括：insert、update、replace、delete、drop、rename、dropDatabase 和 invalidate。

**15.4 变更流故障恢复**

MongoDB 4.0 之后，可以通过指定 startAtOperationTime 来控制从某个特定的时间点开启监听，但该时间点必须在所选择节点的有效 oplog 时间范围内。change stream 监听返回的字段中有个 \_id 字段，表示的是 resume token，这是唯一标志 change stream 流中的位置的字段。

如果 change stream 监听比中止后需要继续监听，那么可指定 resumeAfter 恢复订阅。指定 resumeAfter 为 change stream 中断处的 \_id 字段即可。

当监听的集合发生 rename、drop 或 dropDatabase 事件，就会导致 invalidate 事件；当监听的数据库出现 dropDatabase 事件，也会导致无效事件。invalidate 事件后 change stream 的游标会被关闭，这时就需要使用 resumeAfter 选项来恢复 change stream 的监听，在 4.2 版本后也可以通过 startAfter 选项创建新的更改流来恢复监听。

**15.5 变更流使用限制**

* change stream 无法配置到系统库或者 system.xxx 表上。
* change stream 依赖于 oplog，因此中断时间不可超过 oplog 回收的最大时间窗。

**16 MongoDB 性能问题定位方式**

* 可以为 mongod 实例启用数据库分析。数据库分析器既可以在实例上启用，也可以在单个数据库层面上启用。它收集在实例上执行的 CRUD 操作、游标、命令、配置等详细信息，并将它收集的所有数据写到 system.profile 集合。这是一个capped collection，默认情况下，system.profile 容量大小为 4M。开启实时数据库分析往往伴随着副作用，请谨慎使用。
* 使用 db.currentOp() 操作。它返回一个文档，其中包含有关数据库实例正在进行的操作的信息。
* 使用 db.serverStatus() 命令。它返回一个文档，提供数据库状态的概述，通过它可以收集有关该实例的统计信息。
* 使用 explain 来评估查询性能，例如 cursor.explain() 或 db.collection.explain() 方法可以用来返回关于查询执行的信息。
* 借用一些商业工具，比如 MongoDB Ops Manager、Percona 等。

最后更新于 2021-09-03 13:31

如果觉得我的文章对您有用，请随意赞赏

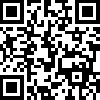
赏

[2人已赞赏](javascript:void();)

[[](https://km.woa.com/user/willzche)](https://km.woa.com/user/willzche)[](https://km.woa.com/user/wefchen)

仅供内部学习与交流，未经公司授权切勿外传

标签：[MongoDb](https://km.woa.com/group/tmecibdev/articles?tag_id=27439" \o "MongoDb" \t "_blank)(1) [mongo](https://km.woa.com/group/tmecibdev/articles?tag_id=28493)(1) [数据库](https://km.woa.com/group/tmecibdev/articles?tag_id=866)(2) [存储](https://km.woa.com/group/tmecibdev/articles?tag_id=917)(1) [事务](https://km.woa.com/group/tmecibdev/articles?tag_id=1422)(1)



本文专属二维码，扫一扫还能分享朋友圈

想要微信公众号推广本文章？[点击获取链接](javascript:void(0);)

**相关阅读**

* [TME Cmongo白皮书 - Cmongo特性介绍](https://km.woa.com/posts/show/444293?kmref=related_post)
* [TME Cmongo白皮书 - Cmongo基础使用](https://km.woa.com/posts/show/444294?kmref=related_post)
* [MongoDB Change Stream之一——上手及初体验](https://km.woa.com/posts/show/468363?kmref=related_post)
* [Mongodb索引介绍](https://km.woa.com/posts/show/480171?kmref=related_post" \o "Mongodb索引介绍" \t "_blank)
* [一文入门Mongo 事务](https://km.woa.com/posts/show/505325?kmref=related_post)
* [CMongo内核：WiredTiger 文件管理剖析](https://km.woa.com/posts/show/508918?kmref=related_post" \o "CMongo内核：WiredTiger 文件管理剖析" \t "_blank)

我顶 (53)

收藏 (233)

* [转载](javascript:void(0);)

* [收录](javascript:void(0);)
* [评论](javascript:void(0);)(21)
* [反馈](javascript:void(0);)

分享到

大家评论

[](https://km.woa.com/user/battyguo)

[battyguo](https://km.woa.com/user/battyguo)

2021-08-31 13:00:16

好文

 顶 (1) [回复](javascript:void(0);)

[](https://km.woa.com/user/songbaiwei)

[songbaiwei](https://km.woa.com/user/songbaiwei)

2021-08-31 13:03:52

顶大佬

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/frankylli)

[frankylli](https://km.woa.com/user/frankylli)

2021-08-31 14:20:18

学习

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/mathe)

[mathe](https://km.woa.com/user/mathe)

2021-08-31 14:20:37

好文

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/v_sishanhu)

[v\_sishanhu](https://km.woa.com/user/v_sishanhu)

2021-08-31 14:31:34

tnl 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/qimingchen)

[qimingchen](https://km.woa.com/user/qimingchen)

2021-08-31 14:32:06

yyds

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/allenxguo)

[allenxguo](https://km.woa.com/user/allenxguo)

2021-09-01 17:38:39

WiredTiger 引擎的数据结构是B+吗 

 顶 (1) [回复 (5)](javascript:void(0);)

[](https://km.woa.com/user/allenxguo)

[allenxguo](https://km.woa.com/user/allenxguo)

2021-09-01 17:39:06

记得是BTree和LSM

[回复](javascript:void(0);)

[](https://km.woa.com/user/hazenweng)

[hazenweng](https://km.woa.com/user/hazenweng) (楼主)

2021-09-01 21:17:48

mongodb 默认采用 WiredTiger 存储引擎，WiredTiger 支持使用 B+ Tree 和 LSM 两种结构进行数据存储管理，可配置，但默认使用 B+ 树

[回复](javascript:void(0);)

[](https://km.woa.com/user/allenxguo)

[allenxguo](https://km.woa.com/user/allenxguo)

2021-09-02 19:55:12

[@hazenweng](https://km.woa.com/user/hazenweng) 有找到相关文档吗，我找了一圈没找到文档有些。。没有去啃源码。因为看了网上都说是BTree 和 LSM，所以还挺奇怪的

[回复](javascript:void(0);)

[](https://km.woa.com/user/hazenweng)

[hazenweng](https://km.woa.com/user/hazenweng) (楼主)

2021-09-05 23:44:55

[@allenxguo](https://km.woa.com/user/allenxguo) 实际上BTree叶子节点才实际存储数据，[http://source.wiredtiger.com/3.2.1/tune\_page\_size\_and\_comp.html](http://source.wiredtiger.com/3.2.1/tune_page_size_and_comp.html" \o "http://source.wiredtiger.com/3.2.1/tune_page_size_and_comp.html" \t "_blank) 以及 <https://source.wiredtiger.com/10.0.0/tune_page_size_and_comp.html> 这里边都有提到 WiredTiger maintains a table's data in memory using a data structure called a B-Tree ( B+ Tree to be specific)

[回复](javascript:void(0);)

[](https://km.woa.com/user/allenxguo)

[allenxguo](https://km.woa.com/user/allenxguo)

2021-09-06 11:33:10

细节！

[回复](javascript:void(0);)

[](https://km.woa.com/user/simonhgao)

[simonhgao](https://km.woa.com/user/simonhgao)

2021-09-01 18:56:53

学习，先赞再看

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/chapzhang)

[chapzhang](https://km.woa.com/user/chapzhang)

2021-09-02 09:51:06

好文

 顶 [回复](javascript:void(0);)

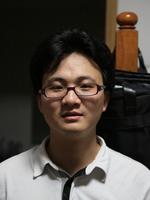
[](https://km.woa.com/user/strongwu)

[strongwu](https://km.woa.com/user/strongwu)

2021-09-02 10:01:59

建议对mangodb和mysql做个比较

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/piboyeliu)

[piboyeliu](https://km.woa.com/user/piboyeliu)

2021-09-02 10:45:46

顶大佬

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/phoenixxliu)

[phoenixxliu](https://km.woa.com/user/phoenixxliu)

2021-09-02 10:56:49

内容非常多，也比较全面哈~点赞！  
这一些地方有待商榷哈：  
3 “连接时若不指定 database，默认数据库为 db”-----集群如果没有开启免认证模式的话，是需要连到admin库上进行认证的。 “admin 数据库主要是保存 root 用户。将一个用户添加到这个数据库，这个用户自动继承所有数据库的权限。”----admin库里会保存一些系统表，比如system.users存储用户，system.roles存储角色，一般不建议用户直接操作这个数据库。如果一个user拥有admin库上的‘dbAdminAnyDatabase’role，则会拥有所有数据库的管理员权限。 “因此可以用来存储本地单台服务器的任意 collection。” —— 同样不建议业务使用local库存储任何数据，无法被正常备份/恢复。也不要修改local库下的任何表的数据。  
9.2 ---5.0开始支持live resharding  
16 大家如果使用的是云上集群，建议优先联系相关运维同事和我们值班账号h\_CMongo\_Developer，我们有丰富的问题定位经验。不建议自行开profile，在已经出问题时开启会带来负面影响。  
--------  
最后，我们在公司码客平台[https://mk.woa.com/coterie/445](https://mk.woa.com/coterie/445" \t "_blank)上也沉淀了很多相关的文章，比如本文提到的主从同步、副本集选举、change stream、慢日志、balancer等。大家如果希望了解更多，欢迎前去圈子提问或者直接咨询本人哈~

 顶 (1) [回复 (1)](javascript:void(0);)

[](https://km.woa.com/user/hazenweng)

[hazenweng](https://km.woa.com/user/hazenweng) (楼主)

2021-09-02 16:12:01

 多谢指正，我修改一下

[回复](javascript:void(0);)

[](https://km.woa.com/user/huiruan)

[huiruan](https://km.woa.com/user/huiruan)

2021-09-02 11:27:54



 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/zhenqiangyi)

[zhenqiangyi](https://km.woa.com/user/zhenqiangyi)

2021-09-02 16:54:39

顶！！！ mongodb能否改成MongoDB，哈哈，强迫症好多年了

 顶 [回复 (1)](javascript:void(0);)

[](https://km.woa.com/user/hazenweng)

[hazenweng](https://km.woa.com/user/hazenweng) (楼主)

2021-09-02 19:06:03

 已改

[回复](javascript:void(0);)

[](https://km.woa.com/user/rickyrqzhao)



[切换到更多功能](javascript:void(0);)

关于作者

[](https://km.woa.com/user/hazenweng)

[hazenweng(翁海澄)](https://km.woa.com/user/hazenweng)

公司其他组织\TME基础架构部\应用开发组员工

作者文章

* [MongoDB 基础浅谈](https://km.woa.com/posts/show/519517?kmref=author_post)
* [使用 golang 写一个 LRU cache 和 LFU cache](https://km.woa.com/posts/show/508339?kmref=author_post)
* [Elasticsearch 的个人学习与实践](https://km.woa.com/posts/show/493990?kmref=author_post)

Copyright©1998-2021 Tencent Inc. All Rights Reserved

腾讯公司研发管理部 版权所有

[广告申请](https://km.woa.com/chartlets/km/) [反馈问题](javascript:void(0);)

[463/502/359 ms]