云数据库 MongoDB

运维开发指南

产品文档

腾讯云

文档目录

运维开发指南
开发规范

最近更新时间：2024-01-12 10:54:49

数据库设计规范

禁止类

禁止数据库中创建过多的表

在 wiretiger 引擎中，每个集合都需要创建多个文件来保存元数据、数据及索引，磁盘上过多的小文件会导致性能下降。建议单个数据库表个数控制在100个以内，整个数据库实例表数量控制在2000个以内。

建议类

建议数据库名以 db 开头，不能包含除 _ 以外的特殊字符，所有字母全部小写，数据库名不超过64个字符。
建议集合名以 t_开头，不能包含除 _ 以外的特殊字符、集合名不超过120个字符。

索引设计规范

禁止类

禁止线上库不带 background 参数建索引

MongoDB 4.2 及之前的版本，createIndex() 命令默认是 foreground 模式，这种模式下创建索引会阻塞数据库的所有操作，造成业务中断，线上业务执行 createIndex() 务必添加 background参数。

注意：

background 需要在 createIndex() 命令的 options 参数中，示例：`db.test.createIndex({a: 1}, {unique:true, background: true})` ，切勿将不同的参数分开，示例：`db.test.createIndex({a: 1}, {unique:true}, {background: true})` 。

排序字段需要放到索引中，避免业务大量在内存中排序造成数据库 OOM（Out Of Memory）

MongoDB 4.2 及之前的版本，一条 sql 默认只允许使用32MB内存进行排序，如果超出会提示 `Sort operation used more than the maximum 33554432 bytes of RAM` 错误，此时可以通过执行 `db.runCommand({ getParameter : 1, "internalQueryExecMaxBlockingSortBytes" : 1 } )` 调整排序内存。

但是线上业务禁止调整，因为这样会让数据库 OOM 的概率增大。建议将排序字段加到索引中，通过索引排序实现排序能力。

MongoDB 4.4 版本，虽然提供了磁盘排序的选项以避免排序消耗大量内存，但是建议最好使用索引排序。

禁止一个表内创建过多的索引

MongoDB 插入每条数据的时候同时需要写索引。索引越多，写入数据时就要花费更多的代价。因此禁止对索引的滥

用，如对每个字段都建立索引，哪怕不会根据该字段进行查询。建议单表索引最多不超过10个。

建议类

建议定期清理无用索引

索引在写操作时会带来额外的资源消耗，因此需要尽量精简索引。

MongoDB 4.4 之后的版本，建议使用 hidden index 先隐藏掉无用的索引，隐藏后业务确认正常再删除索引。

按照最左匹配原则，如果单列索引已经被复合索引包含，建议删除

额外的索引会造成写操作时候性能浪费。

建议尽量避免使用 $ne/$nin 等操作

和其他数据库（如 MySQL）一样，不等于及 not in 类的操作无法有效利用索引，尽量应该避免。

建议在区分度较大的字段上建立索引

如果索引字段区分度较小，查询扫描的行数依然会比较多，查询效率较低，对数据库负载影响较大。因此建立索引的字段尽量应该有较大的区分度。

数据库操作规范

禁止类

禁止上线未经过 explain() 确认执行计划的 sql

上线 sql 前需要 explain() 确认执行计划是否符合预期，否则上线可能会引起故障。

线上环境禁止关闭鉴权，特别是开放外网访问的数据库

关闭鉴权会将数据库暴露给所有人，特别是数据库服务器开通了外网。建议线上环境打开鉴权。如果一定要关闭鉴权，务必设置防火墙规则或 IP 白名单。

禁止在 admin 库、local 中存储业务数据

admin 库读写时会加 db 锁，影响性能；local库只会保存到本地，不会复制到从节点，如果发生主从切换会丢失数据。因此禁止使用 admin 库和 local 库。

禁止执行 db.dropDatabase() 命令后再创建同名的 db

MongoDB 4.0 及之前的版本，官方文件要求删除 db 并创建同名 db 后，业务读写数据前需要在所有 mongos 节点上执行重启或 flushRouterConfig 命令。

MongoDB 4.2 及之后的版本，需要对所有的 mongos 和 mongod 节点重启或执行 flushRouterConfig 命令。

因此禁止在业务代码中直接进行 db.dropDatabase() 命令后再创建同名的 db。运维人员在做该操作时，请务必按照官方文档要求进行所有必要的操作。

高并发高性能场景，禁止过度使用 in 和 or

in 或者 or 条件语句在数据库底层需要转换成多次查询，过多的 in 和 or 操作在高并发高性能场景，会严重影响请求的响应时延及数据库负载。

高并发高性能场景，禁止将复杂的运算操作交给数据库进行

MongoDB 提供了强大的计算能力（如 MapReduce 等），这些特性对开发人员非常友好，极大减轻了业务逻辑。但是这些运算不可避免是需要资源的，如果将复杂运算下沉到数据库层，高并发场景势必会给数据库造成极大的负

担，数据库一旦故障会造成整个系统雪崩。
建议在高并发高性能的场景下，数据库操作保持简单，复杂的运算交给服务器并适当在数据库前端增加缓存。

线上业务禁止直接进行批量数据 remove

remove 命令到数据库后会先查询符合删除条件记录的 _id，之后一条条按照 _id 进行删除，并记录到 oplog 中（删除每条记录都会写一条 oplog）。
当满足 remove 条件的数据较多时对数据库压力较大，且极容易引起主从延迟突然增大。
线上业务建议直接用 drop 集合或用脚本一条条删除并控制删除速度。或尽量使用 ttl 索引。

业务禁止自定义 _id 字段_id 是 MongoDB 内部的默认主键，默认这是一个自增的序列。如果自定义 _id 并且业务无法保证 _id 递增，每次插入数据后，_id 索引不可避免需要对 B 树索引进行调整，这将对数据库带来额外的负担。

副本集直连 mongod 节点的场景，使用禁止只在连接串中配置单个 IP；分片集群禁止只连接单个 mongos 地址（除非 mongos 和应用服务器部署在一起）

线上业务如果只连接副本集主节点，一旦数据库发生 HA 会造成写入中断；如果只连接单个 mongos，这个 mongos 故障后会造成业务中断。

线上业务禁止设置 Write Concern j:false

Write Concern 默认一般为 j:true，表示服务端会写入 journal log 完成后再向 client 端返回。一般请勿设置 j:false，否则进程突然故障重启后，可能会造成数据丢失。

update 语句中禁止不带条件的更新

推荐保持 multi 为默认值（false），避免程序 bug（如由于某些异常造成 query 参数传了{}）造成全表数据更新。

禁止更新数组内部分元素时，将数组全部拿出来更新后再写回去

推荐使用 arrayFileters 仅对需要的元素进行修改。

建议类

建议局部读写而不是全读全写

查询语句中应尽量使用 $projection 运算符投影出需要的字段；在 update 命令中如果只是修改某个字段，建议使用 $set，请勿将文档全部读出来修改后再全量写进去。

线上环境慎重使用 db.collection.renameCollection() 命令

renameCollection() 在4.0及之前的版本会阻塞 db 的所有操作；在4.2及其之后版本会阻塞当前表及目标表的操作。而且 renameCollection() 执行期间会造成游标失效、changeStream 失效及带 --oplog 命令的 mongodump 失败等问题。线上环境禁止高峰期直接操作。

建议核心业务配置 WriteConcern 为 {w: “majority”} 参数

默认情况下，一般驱动的 WriteConcern 配置为 {w:1}，即在主节点写入完成后认为请求成功。如果机器突然发生故障并且写入的数据还未复制到从节点，这样的配置会导致数据丢失。
因此对于线上的核心业务，建议配置 {w: “majority”}，这样的配置会等数据同步大多数节点后再返回客户端。当然可靠性和性能不能兼顾，选择了 {w: “majority”} 配置后请求的延迟也会相应的增加。

不建议类

除非必要，不要在高性能场景大量使用多文档事务

MongoDB 4.0 及之后的版本，MongoDB 提供了多文档事务。但是多文档事务只是 MongoDB 数据库能力的补充，在

高并发高性能场景下，大规模使用多文档事务需要进行充分的压测。
一般来说，多文档事务提交前需要在内存中保留快照，这可能消耗大量的 cache 从而导致性能下降。

不建议使用短连接

MongoDB 的认证逻辑是一个比较复杂的运算过程，而且默认 MongoDB 会为每个连接创建一个线程。大量短连接会对数据库产生较大的负担，特别是没有 mongos 的副本集集群。建议使用长连接，详细参考 mongodb url 中 Connection Pool 参数。

分片集群设计规范

禁止类

如果使用 _id 字段作为片键，禁止使用范围分片

id 默认是一个递增的序列，随着数据量的增加会一直增大。如果 _id 作为片键并使用范围分片，集群随着数据的插入不断的进行 balance。

分片集群禁止直连 mongod 节点写数据

分片集群应该通过 mongos 写数据，直接通过 mongod 写入的数据无路由信息，会导致访问不到。

线上环境禁止长时间关闭 balancer 和 autoSplit 配置

关闭 balance 会导致片之间数据不均衡，关闭 autoSplit 可能会产生 jumbo chunk。

分片表尽量避免不带片键的查询

分片表不带片键进行查询，需要扫描所有分片后在 mongos 聚合结果，比较消耗性能，不推荐使用。

线上环境务必设置 balancer 窗口，避免 balance 对业务造成影响

balance 过程会明显对数据库造成较大的压力，建议放在业务低峰期进行。

建议类

建议使用区分度较大的字段作为片键，最理想的情况是使用唯一主键作为片键

假设我们有一个存储人口信息的集合，其使用性别作为片键，这个片键认为区分度较低，因为集合中性别字段相同的数据理论上会有一半之多。
如果片键区分度不大，可能导致大量的记录集中在某些片上，而这种不均衡也无法添加分片进行扩展。因此建议使用区分度较大的字段作为片键。

如果使用 hash 分片，建议进行预分配，特别是表比较大且经常需要大量插入数据

shardCollection() 命令默认每个分片只会创建2个 chunk，随着数据量的越来越大，MongoDB 需要不断的 balance 和 splitChunk，这将对数据库带来较大的负担。
因此在对于大集合，建议提前进行预分片（shardCollection 命令指定 numInitialChunks 参数，每个分片最大支持8192个），特别是向大集合中批量导数据。

不建议类

没有按片建顺序扫描的强需求，不建议使用 range 分片，推荐 hash 分片

range 分片容易引起不均衡和数据热点，而且因为无法预分片所以随着数据的写入 balance 不可避免，因此不建议使

用，除非有特殊的按片键范围查询需求。

注意：

在 shardCollection 的时候，sh.shardCollection("records.people", { zipcode: 1 }) 命令中1表示范围分片，sh.shardCollection("records.people", { zipcode: "hashed" }) 命令中"hashed"表示 hash 分片。需注意不要用错。

分片集群中不建议使用非分片表

MongoDB 的分片集群如果未执行 shardCollection 命令，默认数据只存储在主分片上。大量未分片的表会造成分片和分片间的数据量不一致。集群长时间运行下去，可能会造成某些片数据量特别多甚至会打满磁盘，运维在这种情况下不得不使用 movePrimary 手动进行数据搬迁，从而增加了运维复杂度。

3.2版本分片集群命令支持情况

最近更新时间：2024-01-12 10:55:05

分片策略

支持 range 的分片机制。
支持联合字段的 shard key。
分片实例下所有数据集合必须使用分片，建议把不需要分片的数据放到单独的副本集实例下。

认证机制

完全兼容支持 SCRAM-SHA-1 和 MONGODB-CR 两种机制。

分片集群命令支持情况

分类	命令	子命令	支持情况
CRUD 基本命令	find	filter	支持
		sort	支持
		projection	支持
		hint	支持
		skip	支持
		limit	支持
		batchSize	支持
		singleBatch	支持
		comment	支持
		maxScan	支持
		maxTimeMS	不支持
		readConcern	支持

	max	支持
	min	支持
	returnKey	支持
	showRecordId	支持
	snapshot	不支持
	tailable	不支持
	oplogReplay	不支持
	noCursorTimeout	支持
	awaitData	不支持
	allowPartialResults	不支持
insert	必须带 shardkey 字段，批量 insert 时 shard key 必须一致	支持
update	更新字段不能是 shardkey	支持
delete	-	支持
findandmodify	-	支持
count	-	支持
distinct	必须带有 shard key	支持
aggregate	-	支持
group	-	不支持
mapReduce	-	不支持
getmore	-	支持
getLastError	-	不支持
getPrevError	-	不支持
resetError	-	不支持
eval	-	不支持
geoNear	-	不支持

	geoSearch	-	不支持
	parallelCollectionScan	-	不支持
Diagnostic 命令	collStats	-	支持
	dbstats	-	支持
	explain	-	支持
	listDatabases	-	支持
	serverStatus	-	不支持
	top	-	不支持
分片命令	enableSharding	-	支持
	shardCollection	-	支持
管理命令	listCollections	-	支持
	dropDatabase	-	支持
	drop	-	支持
	creareIndexes	-	支持
	listIndexes	-	支持
	dropIndexes	-	支持
	logout	-	支持
	renameCollection	-	不支持
	copydb	-	不支持
	create	-	不支持
	clone	-	不支持
	cloneCollection	-	不支持
	cloneCollectionAsCapped	-	不支持
	convetToCapped	-	不支持
	filemd5	-	不支持

	fsync	-	不支持
	clean	-	不支持
	connPoolSync	-	不支持
	connectionStatus	-	不支持
	compact	-	不支持
	collMod	-	不支持
	reIndex	-	不支持
	setParameter	-	不支持
	getParameter	-	不支持
	repairDatabase	-	不支持
	repairCursor	-	不支持
	touch	-	不支持
	shutdown	-	不支持
	logrotate	-	不支持
	killop	-	不支持
user 管理命令	-	-	不支持
role 管理命令	-	-	不支持
副本集命令	-	-	不支持

3.6版本命令支持情况

最近更新时间：2024-01-12 10:55:19

MongoDB 官方命令请参见 官网文档。

腾讯云数据库 MongoDB 3.6 版本不支持的命令如下表所示：

命令分类	不支持的命令
Sharding Commands	addShard
	removeShard
Query and Write Operation Commands	getPrevError
Role Management Commands	dropAllRolesFromDatabase
Replication Commands	replSetAbortPrimaryCatchUp
	replSetFreeze
	replSetGetConfig
	replSetGetStatus
	replSetInitiate
	replSetMaintenance
	replSetReconfig
	replSetResizeOplog
	replSetStepDown
	replSetSyncFrom
	resync
Administration Commands	cloneCollection
	cloneCollection
	cloneCollectionAsCapped
	compact
	connPoolSync

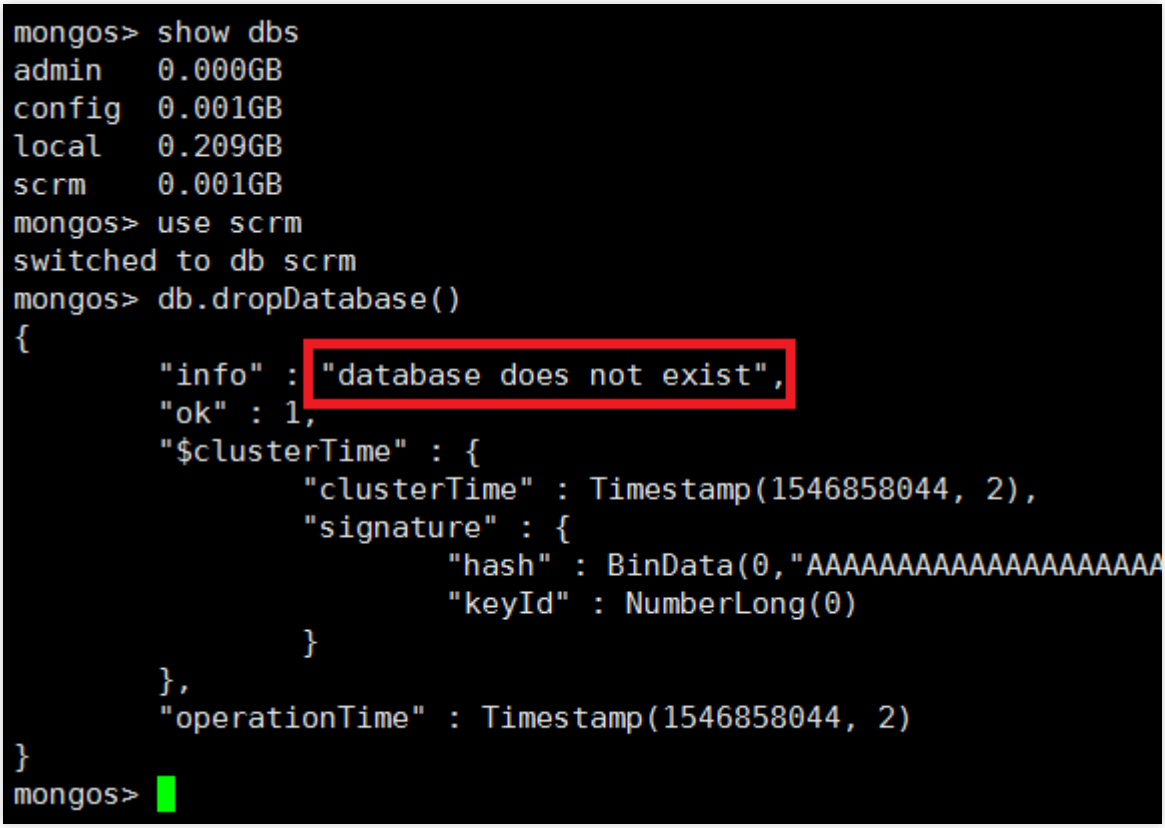
	logRotate
	setParameter
	shutdown
Diagnostic Commands	availableQueryOptions
	dbHash
	getCmdLineOpts
	getLog
	shardConnPoolStats
System Events Auditing Commands	logApplicationMessage

开发运维
3.6版本实例反复创建和删除同名数据库时报错怎么办

最近更新时间：2024-01-12 10:55:33

问题描述

MongoDB 3.6版本实例如果反复 drop 一个 Database 然后创建一个相同名字的 Database，读写或者 drop 该 Database 时可能会报错 ‘database does not exist’，具体类似下图所示：



解决办法

该问题是一个共性问题。出现该问题的原因是 mongos 可能没有更新其 metadata cache。具体请参考官方 说明。如下图所示：

> **WARNING:**
> If you drop a database and create a new database with the same name, you must either restart all `mongos` instances, or use the `flushRouterConfig` command on all `mongos` instances before reading or writing to that database. This action ensures that the `mongos` instances refresh their metadata cache, including the location of the primary shard for the new database. Otherwise, the `mongos` may miss data on reads and may write data to a wrong shard.

解决办法有两种，请选择其中一种：

1. 重启 mongos，该操作可以在 控制台 实例列表进行。
2. 或者运行 flushRouterConfig 命令，内附详细说明。

CPU 使用高问题排查

最近更新时间：2024-01-12 10:55:56

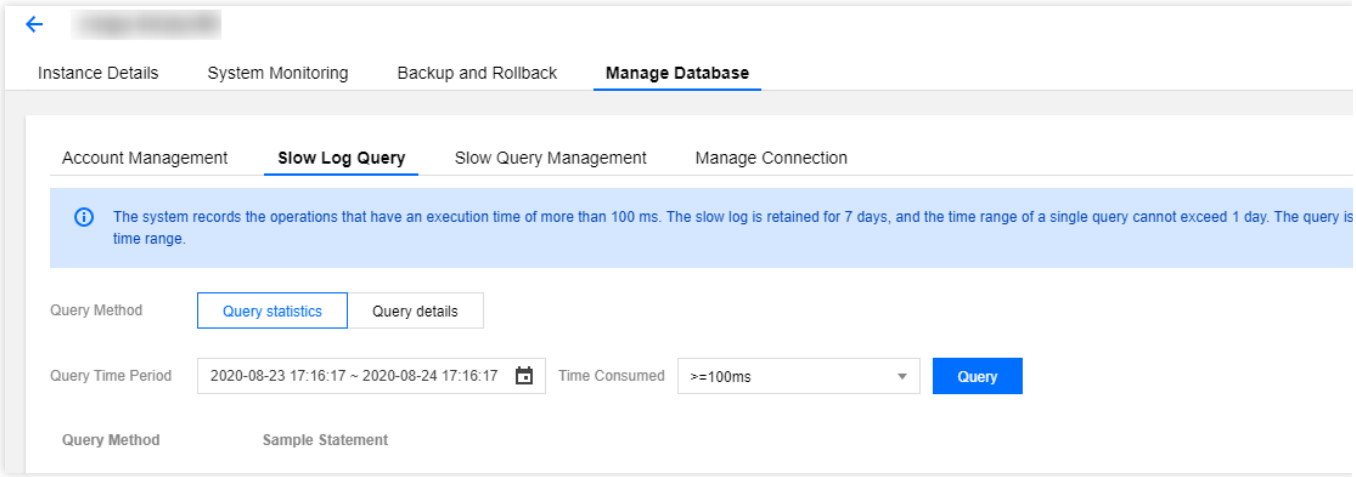
用户在使用 MongoDB 时发现 CPU 使用率高，可以从以下几个方面来排查问题。

1. 首先需要确定业务是否有很高的操作数据库频率。

登录 MongoDB 控制台，在系统监控页面查看 QPS，若业务 QPS 确实高，请评估是否需要升级实例配置。若业务 QPS 并不高，此时需要排查是不是有慢查询。

2. 查看当前 mongod 上有没有慢日志。

登录 MongoDB 控制台，查看实例的慢日志，可选择抽象查询。



请关注：command、COLLSCAN、IXSCAN、keysExamined、docsExamined 等关键字，更多的日志说明请参考 MongoDB 官网。

command 指出慢日志中记录的操作。

COLLSCAN 代表该查询进行了全表扫描，IXSCAN 代表进行了索引扫描。更多的字段描述请参考 MongoDB 官网。

keysExamined 代表索引扫描条目，docsExamined 代表文档扫描条目。keysExamined 和 docsExamined 越大代表没有建索引或者索引的区分度不高。请确认索引的创建字段。

慢查询类问题排查

最近更新时间：2024-01-12 10:56:09

您在使用 MongoDB 时发现请求延迟明显变长时，可按照如下步骤进行排查：

1. 请检查实例的请求延迟监控指标有无异常。

登录 MongoDB 控制台，单击实例 ID 进入管理页面，在**系统监控**页面，检查实例的监控数据，时延监控指标主要反馈的是从请求到达接入层直至处理完返回客户端的时间。若请求时延较高，则需检查 mongod 上是否有慢日志。

2. 查看当前 mongod 上是否有慢日志。

登录 MongoDB 控制台，单击实例 ID 进入管理页面，在**数据库管理 > 慢日志查询**页面查看实例的慢日志，可选择抽象查询。

请关注：command、COLLSCAN、IXSCAN、keysExamined、docsExamined 等关键字，更多的日志说明请参考 MongoDB 官网。

需注意的关键字如下：

command 指出慢日志中记录的操作。

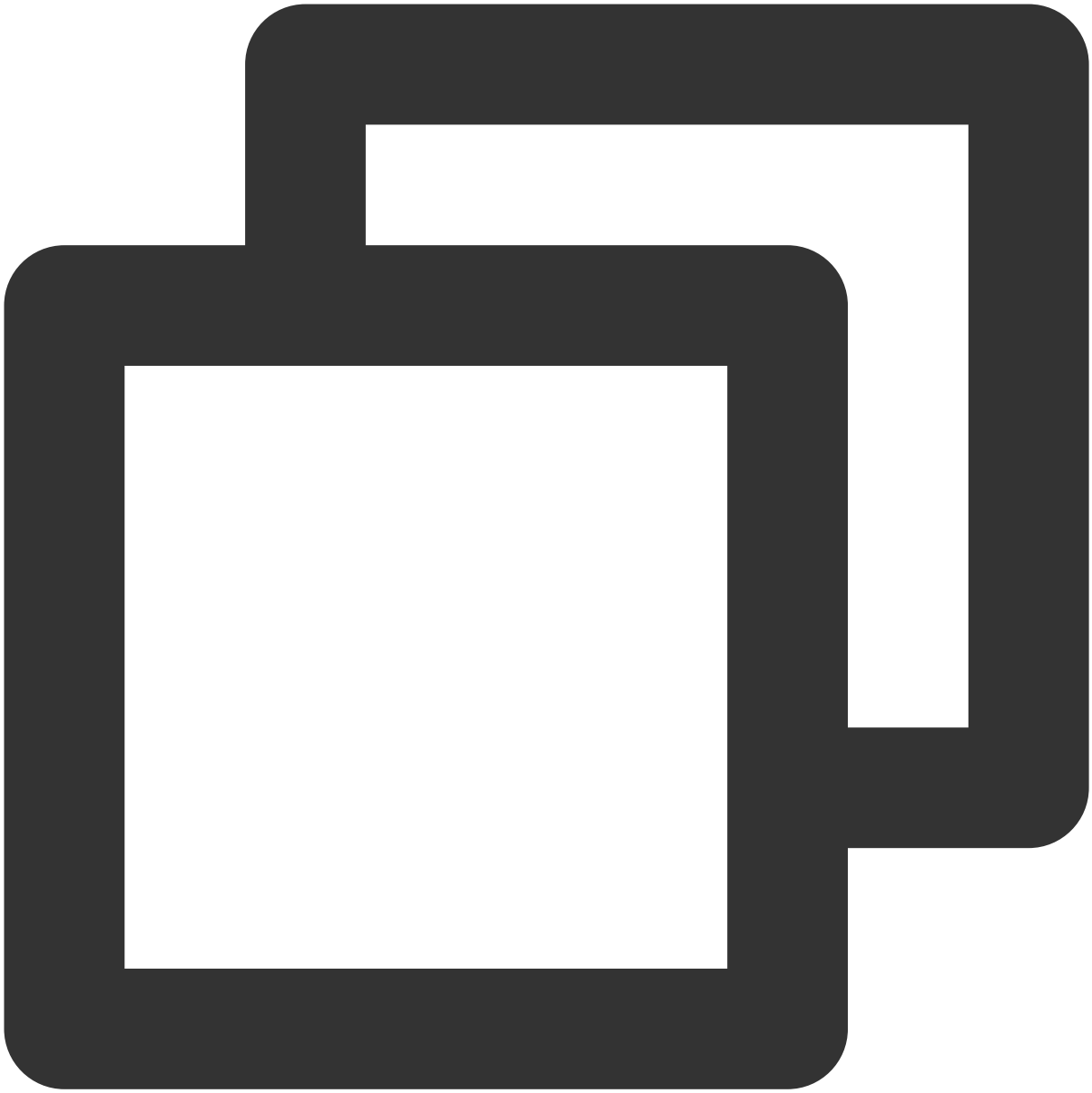
COLLSCAN 代表该查询进行了全表扫描，IXSCAN 代表进行了索引扫描。更多的字段描述请参考 MongoDB 官网。

keysExamined 代表索引扫描条目，docsExamined 代表文档扫描条目。keysExamined 和 docsExamined 越大代表没有建索引或者索引的区分度不高。请确认索引的创建字段。

3. 请确认是否在前台建索引导致请求被锁住。

如果业务查询所用索引并无问题，请确认当前是否在业务繁忙时段进行了前台建索引操作。当为一个集合创建索引时默认方式为前台方式（background 选项的值为 false）。该操作将阻塞其他的所有操作，直到前台完成索引创建。但如果采用后台方式建索引，则在创建索引期间，MongoDB 依旧可以正常提供读写操作服务，不过，后台建索引方式是有代价的，即后台方式建索引可能会导致索引创建时间变长。具体创建索引的选项请参考 MongoDB 官网。

可通过 currentOp 命令来查看当前建索引的进度，具体的命令如下：



```
db.currentOp(
     {
       $or: [
         { op: "command", "query.createIndexes": { $exists: true } },
         { op: "insert", ns: /\\.system\\.indexes\\b/ }
       ]
     }
     )
```

返回如下图所示，msg 字段代表了当前建索引的进度，locks 字段代表该操作的锁类型，更多锁的说明请参考 MongoDB 官网。

Lock Mode	Description
R	Represents Shared (S) lock.
W	Represents Exclusive (X) lock.
r	Represents Intent Shared (IS) lock.
w	Represents Intent Exclusive (IX) lock.

4. 检查是否 mongos 负载过高。

时延监控指标主要反馈的是从请求到达接入层直至处理完返回客户端的时间，若已确认 mongod 上没有慢操作，但请求时延较高，可能是 mongos 负载过高导致。造成 mongos 负载高的原因有多种，例如业务瞬间建立大量连接可能会导致 mongos 负载过高，或者，多个分片的数据需要汇总也可能会造成 mongos 负载高。此时可以进行 mongos 重启。重启 mongos 可以在 控制台 实例列表进行。

注意：

重启 mongos 会导致实例所有的连接在重启的一瞬间中断，业务直接进行重连即可，不存在持续影响业务的可能。

连接类问题排查

最近更新时间：2024-01-12 10:56:23

您在使用 MongoDB 的过程中经常会遇见两类连接问题，下面介绍可以参考的排查思路。

连接使用率高

若您在使用过程中发现实例的连接使用率高，可参考如下步骤来排查问题。

1. 确认业务是否使用了连接池。

MongoDB 的服务模型是每个网络连接由一个单独的线程（one-thread-per-connection）来处理，当网络连接数太多时，过多的线程会导致上下文切换开销变大，同时内存开销也会上涨。每次请求都建立连接和鉴权会极大的影响性能。因此，要限制实例的连接数且连接使用完毕需要释放，推荐业务使用连接池。同时，MongoDB 各个语言的 Driver 基本都会封装一个对象，应该在使用时构造一个全局的对象，然后在后续的请求中使用该全局对象来发送请求给实例。Driver 的对象有默认连接池大小，也可以在构造对象时指定 maxPoolSize 选项来设置连接池大小。

2. 如果已经设置了连接池，请检查业务是否有突发异常。

请确认业务是否有发布变更，代码逻辑缺陷导致建立大量连接。

请检查是否有连接泄漏，在相关 CVM 上统计检查连接数是否异常。

请确认业务是否有大量真实突发请求。

3. 请检查是否有慢查询导致连接被占用。

若确认业务无异常，请检查索引是否有异常，例如之前建立的索引被误删等。

若索引无异常，请检查当前是否有大量慢查询。慢查询导致连接一直占用未被释放，因此会建立更多的连接。

登录 MongoDB 控制台，单击实例 ID 进入管理页面，在**数据库管理 > 慢日志查询**页面查看实例的慢日志，可选择抽象查询。

请关注：command、COLLSCAN、IXSCAN、keysExamined、docsExamined 等关键字，更多的日志说明请参考 MongoDB 官网，

需注意的关键字如下：

command 指出慢日志中记录的操作。

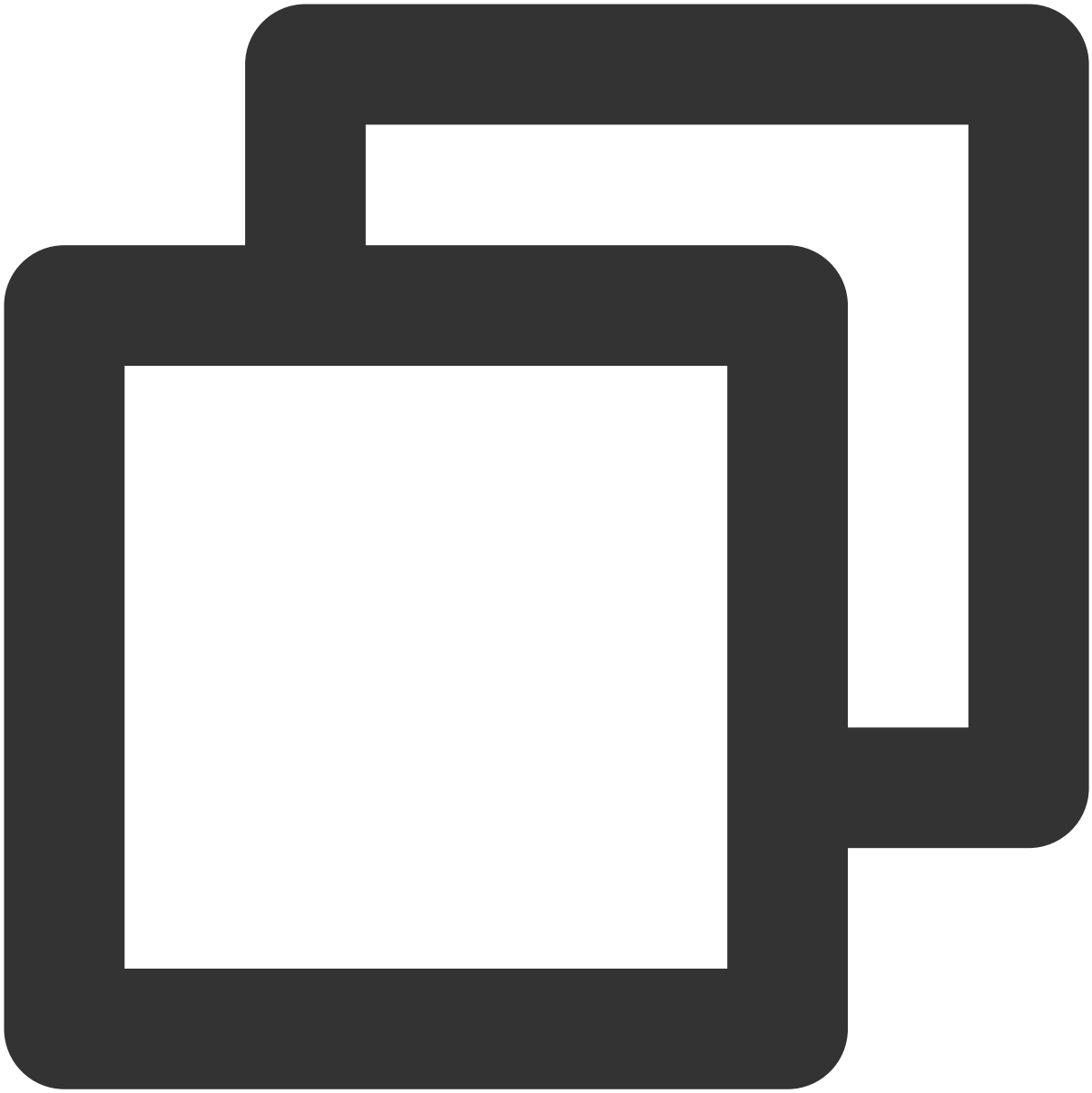
COLLSCAN 代表该查询进行了全表扫描，IXSCAN 代表进行了索引扫描。更多的字段描述请参考 MongoDB 官网。

keysExamined 代表索引扫描条目，docsExamined 代表文档扫描条目。keysExamined 和 docsExamined 越大代表没有建索引或者索引的区分度不高。请确认索引的创建字段。

4. 请确认是否在前台建索引导致请求被锁住。

若业务查询所用索引并无问题，请确认当前是否在业务繁忙时段进行了前台建索引操作。当为一个集合创建索引时默认方式为前台方式（background 选项的值为 false）。该操作将阻塞其他的所有操作，直到前台完成索引创建。但如果采用后台方式建索引，则在创建索引期间，MongoDB 依旧可以正常的提供读写操作服务，但后台建索引方式是有代价的，即后台方式建索引可能会导致索引创建时间变长。具体创建索引的选项请参考 MongoDB 官网。

可通过 currentOp 命令来查看当前建索引的进度，具体的命令如下：



```
db.currentOp(
    {
      $or: [
        { op: "command", "query.createIndexes": { $exists: true } },
        { op: "insert", ns: /\\.system\\.indexes\\b/ }
      ]
    }
    )
```

返回如下图所示，msg 字段代表了当前建索引的进度，locks 字段代表该操作的锁类型，更多锁的说明请参考 MongoDB 官网。

Lock Mode	Description
R	Represents Shared (S) lock.
W	Represents Exclusive (X) lock.
r	Represents Intent Shared (IS) lock.
w	Represents Intent Exclusive (IX) lock.

5. 评估实例配置是否满足业务要求。
若经过使用连接池和建立区分度高的索引之后从控制台监控上看业务的连接使用率仍然很高，则可能是实例的配置已经不能满足业务实际需要导致。此时，需要您根据自身业务模型，流量峰值、 QPS 和 TPS 等要求来评估实际需要的实例规格配置并根据需要升级实例。

连接拒绝

若在实际使用过程中出现了连接拒绝，请参考如下步骤排查问题。
1. 确认实例连接使用率是否达到100%。
登录 MongoDB 控制台，在系统监控页面，查看连接数和连接使用率监控指标。
若实例连接使用率100%，请排查自身业务是否有异常，另一方面紧急情况可以通过在控制台重启 mongos 来快速释放连接。
注意：
重启 mongos 会导致实例所有的连接在重启的一瞬间中断，业务直接进行重连即可，不存在持续影响业务的可能。若重启后业务连接数迅速增加又导致连接使用率100%，则说明业务确实存在大量有效连接，不属于连接泄漏的场景。此时需要业务首先排查产生大量连接的原因，可参考连接使用率高问题的排查方法。
2. 确认用户名与密码是否错误。
请确保用户名和密码正确，如有错，请登录 MongoDB 控制台，单击实例 ID 进入管理页面，在**数据库管理 > 账号管理**页面修改。
3. 确认 Mongoshell 版本。
为保障鉴权成功，请安装 Mongo Shell 3.0及以上版本。安装步骤请参考 官方文档。
4. 确认认证库是否正确。
注意：

在官网控制台创建的用户其认证库均为 admin，因此用户登录时需要制定认证库为 admin。用命令行创建的用户，例如在 test 库下建立的用户登录时指定的认证库为 test。