# mongoDB相关概念

### 配置文件相关

WiredTiger存储引擎

cacheSizeGB：从MongoDB 3.4开始，值取值范围为256MB至10TB，默认情况下，cacheSizeGB值为50%的RAM减去1GB或是256MB。

journalCompressor：WiredTiger采用预写事务日志联合检查站，保证数据的持久性。日志压缩算法默认为snappy。其他可选值有none或zlib。

directoryForIndexes：默认为false。当为true时，mongod会分别单独以索引命名的子目录存储索引和以集合命令的子目录存储集合数据。

blockCompressor：默认值为snappy，用于压缩集合数据的压缩的默认类型。其他可选值为none或zlib。

prefixCompression：默认为true，即使用前缀压缩索引数据。

-----------------------------------

MongoDB系列7：MongoDB存储引擎

<https://blog.51cto.com/u_15127539/2660817>

配置文件注解：  
systemLog:  
 destination: file //Mongodb日志输出为文件  
 logAppend: true //当实例重启时，不创建新的日志文件， 在老的日志文件末尾继续添加  
 path: /data/mongodb\_cluster/mongodb\_27017/logs/mongodb.log //日志路径  
  
storage:  
 journal: //回滚日志，类似于mysql的binlog  
 enabled: true //开启回滚日志  
 dbPath: /data/mongodb\_cluster/mongodb\_27017/data //数据存储目录  
 directoryPerDB: true //默认，false不适用inmemoryengine  
 wiredTiger: //存储引擎  
 engineConfig:  
 cacheSizeGB: 1 //将用于所有数据缓存的大小  
 directoryForIndexes: true //默认false索引集合storage.dbPath存储在数据单独子目录，这里必须配置为true，否则所有库的数据文件都会存放在一个目录中  
 collectionConfig:  
 blockCompressor: zlib //开启压缩  
 indexConfig:   
 prefixCompression: true //开启索引  
  
processManagement: //系统守护进程控制处理  
 fork: true //后台运行  
 pidFilePath: /data/mongodb\_cluster/mongodb\_27017/pid/mongod.pid //pid文件路径  
 timeZoneInfo: /usr/share/zoneinfo // 时区  
  
net:  
 port: 27017 //监听端口  
 bindIp: 127.0.0.1,192.168.81.210 //绑定ip

[MongoDB数据库安装部署及警告优化](https://www.jb51.net/article/231952.htm)

[MongoDB副本集、分片集的伪分布式部署(保姆级教程)](http://www.voycn.com/index.php/article/mongodbfubenjifenpianjideweifenbushibushubaomujijiaocheng)

### 角色安全控制

角色安全控制





[mogoDB思维笔记](https://www.processon.com/outline/view/5c3ededee4b08a7683a9bbc3)

### oplog简介

oplog是local库下的一个**固定集合**，Secondary就是通过查看Primary 的oplog这个集合来进行复制的。每个节点都有oplog，记录这从主节点复制过来的信息，这样每个成员都可以作为同步源给其他节点

Oplog 可以说是Mongodb Replication的纽带了

**Mongodb默认将其大小设置为可用disk空间的5%（默认最小为1G，最大为50G）**，**或也可以在mongodb复制集实例初始化之前将mongo.conf中oplogSize设置为我们需要的值**

第一次启动复制集中的节点时，MongoDB会建立Oplog,会有一个默认的大小，这个大小取决于机器的操作系统

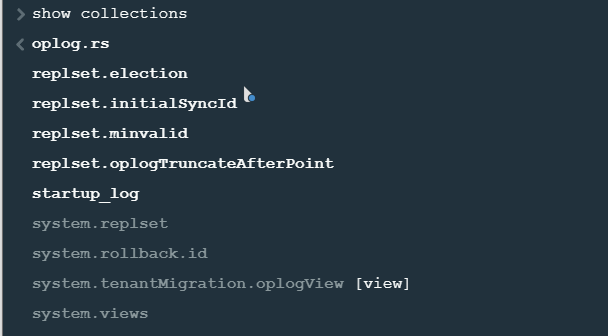
rs.printReplicationInfo() 查看 oplog 的状态，输出信息包括 oplog 日志大小，操作日志记录的起始时间。

db.getReplicationInfo() 可以用来查看oplog的状态、大小、存储的时间范围

#### oplog存储位置

oplog是存储在数据库local中，表名为“oplog.rs”

use local  
show collections



#### oplog的大小

oplog的大小

capped collection是MongoDB中一种提供高性能插入、读取和删除操作的固定大小集合，当集合被填满的时候，新的插入的文档会覆盖老的文档。

所以，oplog表使用capped collection是合理的，因为不可能无限制的增长oplog。MongoDB在初始化副本集的时候都会有一个默认的oplog大小：

* 在64位的Linux,Solaris,FreeBSD以及Windows系统上，MongoDB会分配磁盘剩余空间的5%作为oplog的大小，如果这部分小于1GB则分配1GB的空间

* 在64的OS X系统上会分配183MB

* 在32位的系统上则只分配48MB

**oplog的大小设置是值得考虑的一个问题，如果oplog size过大，会浪费存储空间；如果oplog size过小，老的oplog记录很快就会被覆盖，那么宕机的节点就很容易出现无法同步数据的现象。**

比如，基于上面的例子，我们停掉一个备份节点（port=33333），然后通过主节点插入以下记录，然后查看oplog，发现以前的oplog已经被覆盖了。

**通过MongoDB shell连接上这个节点，会发现这个节点一直处于RECOVERING状态**。

解决方法：

#### 数据同步

数据同步

在副本集中，有两种数据同步方式：

* initial sync（初始化）：这个过程发生在当副本集中创建一个新的数据库或其中某个节点刚从宕机中恢复，或者向副本集中添加新的成员的时候，默认的，副本集中的节点会从离它最近的节点复制oplog来同步数据，这个最近的节点可以是primary也可以是拥有最新oplog副本的secondary节点。

* + 该操作一般会重新初始化备份节点，开销较大

* replication（复制）：在初始化后这个操作会一直持续的进行着,以保持各个secondary节点之间的数据同步。

#### initial sync

当遇到上面例子中无法同步的问题时，只能使用以下两种方式进行initial sync了

* **第一种方式**就是停止该节点，然后删除目录中的文件，重新启动该节点。这样，这个节点就会执行initial sync

* + 注意：通过这种方式，sync的时间是根据数据量大小的，如果数据量过大，sync时间就会很长

* + 同时会有很多网络传输，可能会影响其他节点的工作

* **第二种方式**，停止该节点，然后删除目录中的文件，找一个比较新的节点，然后把该节点目录中的文件拷贝到要sync的节点目录中

通过上面两种方式中的一种，都可以重新恢复"port=33333"的节点。改变**一直处于RECOVERING状态的错误。**

[MongoDB oplog详解](http://t.zoukankan.com/Joans-p-7723554.html)

### 集群

#### 启动顺序

mongodb的启动顺序是，先启动配置服务器，在启动分片，最后启动mongos.

#### 数据库 集合开启分片功能

设置分片数据库、分片集合以及分片的key在admin数据库下进行，否则会报shardCollection may only be run against the admin database.错误