MongoDB 3.4 版本配置详解

配置说明

在Mongod安装包中，包含2个进程启动文件：mongod和mongos；其中mongd是核心基础进程，用来接收读写请求、负责存储实际数据，mongod实例是构成集群的基本单位，比如Replication set、Sharding Cluster、Config Servers等；mongos是Sharding Cluster架构模式中的“路由”进程，即客户端请求访问mongos，然后有mongos将请求转发给合适的sharding server执行操作，并将result返回给客户端，所以mongos基本不存储数据，只是在内存中缓存部分shard key与sharding server的对应关系，便于路由。

在配置文件方面，mongod和mongos有很多相同之处，下文中如有区别之处将会特别指出。

在一个节点上，通常同时启动mongod和mongos进程是正常的，他们之间不存在资源竞争，但是为了避免冲突，我们希望它们使用各自的配置文件，比如mongod.conf、mongos.conf；mongodb的某些平台下的安装包中没有自带配置文件，需要开发者自己创建。

重要配置参数讲解如下：

- **processManagement:**

fork: <true | false>
描述：是否以fork模式运行mongod/mongos进程，默认为false。
pidFilePath:<路径>
描述：配合"fork:true"参数，将mongod/mongos进程ID写入指定的文件，如果不指定，将不会创建PID文件。

- **net:**

bindIp: <127.0.0.1>
描述：mongod/monogs进程绑定的IP，application通过此IP、port建立链接。可以绑定在任意网卡接口上，如果你的mongos/mongod只需要内网访问，可以绑定在内网IP(例如：192.168.1.100)，如果需要外网访问，那么则绑定外网IP，如果此值为“0.0.0.0”，则绑定到所有接口即内网、外网IP均可以访问。（不建议）可以绑定都多个ip上，ip地址之间用“,”分割。
port: 27017
描述：mongod/mongos侦听端口，默认为27017；不过因为mongodb有2种典型的架构模式：replica set和sharding，如果开发者在一个节点上部署多个mongod实例，需要注意修改此端口以避免冲突。
maxIncomingConnections: 65536
描述：mongod/mongos进程允许的最大连接数，如果此值超过操作系统配置的连接数阀值，将不会生效(ulimit)；默认值为65536。通常客户端将会使用连接池机制，可以有效的控制每个客户端的链接个数。

wireObjectCheck: true
描述：当客户端写入数据时，mongos/mongod是否检测数据的有效性(BSON)，如果数据格式不良，此insert、update操作将会被拒绝；默认值为true
ipv6: false
描述：是否支持mongos/mongod多个实例之间使用IPV6网络，默认值为false。此值需要在整个cluster中保持一致。

- **storage:**

dbPath: db
描述：mongod进程存储数据目录，此配置仅对mongod进程有效。默认值为：/data/db。

indexBuildRetry: true
描述：当构建索引时mongod意外关闭，那么再次启动是否重新构建索引；索引构建失败，mongod重启后将会删除尚未完成的索引，但是否重建由此参数决定。默认值为true。

repairPath: _tmp
描述：配合--repair启动命令参数，在repair期间使用此目录存储临时数据，repair结束后此目录下数据将被删除，此配置仅对mongod进程有效。不建议在配置文件中配置，而是使用mongod启动命令指定。

engine: mmapv1
描述：存储引擎类型，mongodb 3.0之后支持“mmapv1”、“wiredTiger”两种引擎，默认值为“mmapv1”；官方宣称wiredTiger引擎更加优秀。

journal:
enabled: true
描述：是否开启journal日志持久存储，journal日志用来数据恢复，是mongod最基础的特性，通常用于故障恢复。64位系统默认为true，32位默认为false，建议开启，仅对mongod进程有效。
directoryPerDB: false
描述：是否将不同DB的数据存储在不同的目录中，dbPath的子目录，目录名为db的名称。对已经存储数据的mongod修改此值，需要首先使用mongodump指令将数据导出，然后关闭mongod，再修改此值和指定新的dbPath，然后使用mongorestore指令重新导入数据。（即导出数据，并使用mongorestore将数据重新写入mongod的新目录中）对于replica set架构模式，只需要在每个secondary依次操作：关闭secondary，然后配置新的dbPath，然后启动即可（会执行初始化sync，从primary中将数据去完全同步到本地）。最后操作primary。**此参数仅对mongod进程有效，默认值为false，不建议修改此值**

syncPeriodSecs: 60
描述：mongod使用fsync操作将数据flush到磁盘的时间间隔，默认值为60（单位：秒），强烈建议不要修改此值；mongod将变更的数据写入journal后再写入内存，并间歇性的将内存数据flush到磁盘中，即延迟写入磁盘，有效提升磁盘效率。此指令不影响journal存储，仅对mongod有效。
mmapv1:（如下配置仅对MMAPV1引擎生效）
quota:
enforced: false
描述：配额管理，是否限制每个DB所能持有的最大文件数量，仅对mongod有效，默认值为false，建议保持默认值。
maxFilesPerDB: 8
描述：如果enforce开启，每个DB所持有的存储文件不会超过此阀值。仅对mongod进程有效。
smallFiles: false
描述：是否使用小文件存储数据；如果此值为true，mongod将会限定每个数据文件的大小为512M（默认最大为2G），journal降低到128M（默认为1G）。如果DB的数据量较大，将会导致每个DB创建大量的小文件，这对性能有一定的影响。在production环境下，不建议修改此值，在测试时可以设置为true，节约磁盘。
journal:
commitIntervalMs: 100
描述：mongod进程提交journal日志的时间间隔，即fsync的间隔。考虑到磁盘效能，mongod间歇性的flush日志数据；此值越小，数据丢失的可能性越低，磁盘消耗越大，性能越低。如果希望write操作强制立即写入journal，可以传递参数选项“j:true”（在客户端write操作中指定此选项即可），此操作（包

括此前尚未提交的）将会立即fsync到磁盘。仅对mongod有效，单位：毫秒

nsSize:

每个database的namespace文件的大小，默认为16，单位：M；最大值可以设置为2048，即dbpath下".ns"后缀文件的大小。16M基本上可以保存24000条命名条目，新建一个collection或者index信息，即会增加一个namespace条目；如果你的database下需要创建大量的collection（比如数据分析），则可以适度调大此值。

wiredTiger:（如下配置仅对wiredTiger引擎生效（3.0以上版本）

engineConfig:

cacheSizeGB: 8

描述：wiredTiger缓存工作集（working set）数据的内存大小，单位：GB，此值决定了wiredTiger与mmapv1的内存模型不同，它可以限制mongod对内存的使用量，而mmapv1则不能（依赖于系统级的mmap）。默认情况下，cacheSizeGB的值为假定当前节点只部署一个mongod实例，此值的大小为物理内存的一半；如果当前节点部署了多个mongod进程，那么需要合理配置此值。如果mongod部署在虚拟容器中（比如，lxc，cgroups，Docker）等，它将不能使用整个系统的物理内存，则需要适当调整此值。默认值为物理内存的一半。

journalCompressor: snappy

描述：journal日志的压缩算法，可选值为"none"、"snappy"、"zlib"。

directoryForIndexes: false

描述：是否将索引和collections数据分别存储在dbPath单独的目录中。即index数据保存"index"子目录，collections数据保存在"collection"子目录。默认值为false，仅对mongod有效。

collectionConfig:

blockCompressor: snappy

描述：collection数据压缩算法，可选值"none"、"snappy"、"zlib"。开发者在创建collection时可以指定值，以覆盖此配置项。如果mongod中已经存在数据，修改此值不会带来问题，旧数据仍然使用原来的算法解压，新数据文件将会采用新的解压缩算法。

indexConfig:

prefixCompression: true

描述：是否对索引数据使用"前缀压缩"（prefix compression，一种算法）。前缀压缩，对那些经过排序的值存储，有很大帮助，可以有效的减少索引数据的内存使用量。默认值为true。

- **operationProfiling:**

slowOpThresholdMs: 100

描述：数据库profiler判定一个操作是"慢查询"的时间阀值，单位毫秒；mongod将会把慢查询记录到日志中，即使profiler被关闭。当profiler开启时，慢查询记录还会被写入"system.profile"这个系统级的collection中。请参看mongod profiler相关文档。默认值为100，此值只对mongod进程有效。

mode: off

描述：数据库profiler级别，操作的性能信息将会被写入日志文件中，可选值：

1）off：关闭profiling

2）slowOp：on，只包含慢操作日志

3）all：on，记录所有操作

数据库profiling会影响性能，建议只在性能调试阶段开启。此参数仅对mongod有效。

- **replication:（复制集架构模式配置，如果只是单点，则无需配置）**

oplogSizeMB: 10240
描述：replication操作日志的最大尺寸，单位：MB。mongod进程根据磁盘最大可用空间来创建oplog，比如64位系统，oplog为磁盘可用空间的5%，一旦mongod创建了oplog文件，此后再次修改oplogSizeMB将不会生效。此值不要设置的太小， 应该足以保存24小时的操作日志，以保证secondary有充足的维护时间；如果太小，secondary将不能通过oplog来同步数据，只能全量同步。此值仅对mongod有效。

enableMajorityReadConcern: false
描述：是否开启readConcern的级别为"majority"，默认为false；只有开启此选项，才能在read操作中使用"majority"。（3.2+版本）

replSetName: <无默认值>
描述："复制集"的名称，复制集中的所有mongd实例都必须有相同的名字，sharding分布式下，不同的sharding应该使用不同的replSetName。仅对mongod有效。

secondaryIndexPrefetch: all
描述：只对mmapv1存储引擎有效。复制集中的secondary，从oplog中运用变更操作之前，将会先把索引加载到内存中，默认情况下，secondaries首先将操作相关的索引加载到内存，然后再根据oplog应用操作。可选值：
1）none：secondaries不将索引数据加载到内容
2）all：sencondaries将此操作有关的索引数据加载到内存
3）_id_only：只加载_id索引
默认值为：all，此配置仅对mongod有效。

localPingThresholdMs: 15
描述：ping时间，单位：毫秒，mongos用来判定将客户端read请求发给哪个secondary。仅对mongos有效。默认值为15，和客户端driver中的默认值一样。当mongos接收到客户端read请求，它将：
1、找出复制集中ping值最小的member。
2、将延迟值被此值允许的members，构建一个列表
3、从列表中随机选择一个member。
ping值是动态值，每10秒计算一次。mongos将客户端请求转发给延迟较小（与此值相比）的某个secondary节点。仅对mongos有效。

- **sharding:（仅对sharding架构模式下有效）**

clusterRole: <无默认值>
描述：在sharding集群中，此mongod实例的角色，可选值：
1、configsvr：此实例为config server，此实例默认侦听27019端口
2、shardsvr：此实例为shard（分片），侦听27018端口
此配置仅对mongod有效。通常config server和sharding server需要使用各自的配置文件。

archiveMovedChunks: true
描述：当chunks因为"负载平衡"而迁移到其他节点时，mongod是否将这些chunks归档，并保存在dbPath下"moveChunk"目录下，mongod不会删除moveChunk下的文件。默认为true。

autoSplit: true
描述：是否开启sharded collections的自动分裂，仅对mongos有效。如果所有的mongos都设定为false，那么collections数据增长但不能分裂成新的chunks。因为集群中任何一个mongos进程都可以触发split，所以此值需要在所有mongos行保持一致。仅对mongos有效。

configDB: <无默认值>
描述：设定config server的地址列表，每个server地址之间以“,”分割，通常sharded集群中指定1或者3个config server。（生产环境，通常是3个config server，但1个也是可以的）。所有的mongos实例必须配置一样，否则可能带来不必要的问题。仅对mongos有效。

chunkSize: 64
描述：sharded集群中每个chunk的大小，单位：MB，默认为64，此值对于绝大多数应用而言都是比较理想的。chunkSize太大会导致分布不均，太小会导致分裂成大量的chunk而经常移动
##整个sharding集群中，此值需要保持一致，集群启动后修改此值将不再生效。仅对mongos有效。

- **sytemsLog:（系统日志，必须配置）**

verbosity: 0
描述：日志级别，0：默认值，包含“info”信息，1~5，即大于0的值均会包含debug信息
quiet: true
描述："安静"，此时mongod/mongos将会尝试减少日志的输出量。不建议在production环境下开启，否则将会导致跟踪错误比较困难。

traceAllExceptions: true
描述：打印异常详细信息。

path: logs/mongod.log

logAppend: false
描述：如果为true，当mongod/mongos重启后，将在现有日志的尾部继续添加日志。否则，将会备份当前日志文件，然后创建一个新的日志文件；默认为false。

logRotate: rename
描述：日志“回转”，防止一个日志文件特别大，则使用logRotate指令将文件“回转”，可选值：
1）rename：重命名日志文件，默认值。
2）reopen：使用linux日志rotate特性，关闭并重新打开此日志文件，可以避免日志丢失，但是logAppend必须为true。

localPingThresholdMsdestination: file
描述：日志输出目的地，可以指定为“ file”或者“syslog”，表述输出到日志文件，如果不指定，则会输出到标准输出中（standard output）。

- **与安全有关的配置（摘要介绍）**

```
security:
    authorization: enabled
    clusterAuthMode: keyFile
```

```
    keyFile: /srv/mongodb/keyfile
    javascriptEnabled: true
setParameter:
    enableLocalhostAuthBypass: true
    authenticationMechanisms: SCRAM-SHA-1
```

1）authorization：disabled或者enabled，仅对mongod有效；表示是否开启用户访问控制（Access Control），即客户端可以通过用户名和密码认证的方式访问系统的数据，默认为“disabled”，即客户端不需要密码即可访问数据库数据。（限定客户端与mongod、mongos的认证）

2）clusterAuthMode：集群中members之间的认证模式，可选值为“keyFile”、“sendKeyFile”、“sendX509”、“x509”，对mongod/mongos有效；默认值为“keyFile”，mongodb官方推荐使用x509，不过我个人觉得还是keyFile比较易于学习和使用。不过3.0版本中，mongodb增加了对TLS/SSL的支持，如果可以的话，建议使用SSL相关的配置来认证集群的member，此文将不再介绍。（限定集群中members之间的认证）

3）keyFile：当clusterAuthMode为“keyFile”时，此参数指定keyfile的位置，mongodb需要有访问此文件的权限。

4）javascriptEnabled：true或者false，默认为true，仅对mongod有效；表示是否关闭server端的javascript功能，就是是否允许mongod上执行javascript脚本，如果为false，那么mapreduce、group命令等将无法使用，因为它们需要在mongod上执行javascript脚本方法。如果你的应用中没有mapreduce等操作的需求，为了安全起见，可以关闭javascript。

“setParameter”允许指定一些的Server端参数，这些参数不依赖于存储引擎和交互机制，只是微调系统的运行状态，比如“认证机制”、“线程池参数”等。参见【parameter】

1）enableLocalhostAuthBypass：true或者false，默认为true，对mongod/mongos有效；表示是否开启“localhost exception”，对于sharding cluster而言，我们倾向于在mongos上开启，在shard节点的mongod上关闭。

2）authenticationMechanisms：认证机制，可选值为“SCRAM-SHA-1”、“MONGODB-CR”、“PLAN”等，建议为“SCRAM-SHA-1”，对mongod/mongos有效；一旦选定了认证机制，客户端访问databases时需要与其匹配才能有效。

- 与性能有关的参数

```
a. setParameter:
b.     connPoolMaxShardedConnsPerHost: 200
c.     connPoolMaxConnsPerHost: 200
d.     notablescan: 0
```

1）connPoolMaxShardedConnsPerHost：默认值为200，对mongod/mongos有效；表示当前mongos或者shard与集群中其他shards链接的链接池的最大容量，此值我们通常不会调整。连接池的容量不会阻止创建新的链接，但是从连接池中获取链接的个数不会超过此值。维护连接池需要一定的开支，保持一个链接也需要占用一定的系统资源。

2）connPoolMaxConnsPerHost：默认值为200，对mongod/mongos有效；同上，表示mongos或者mongod与其他

mongod实例之间的连接池的容量，根据host限定。

配置样例

普通mongod节点

```
systemLog:
   quiet: false
   path: /data/mongodb/logs/mongod.log
   logAppend: false
   destination: file
processManagement:
   fork: true
   pidFilePath: /data/mongodb/mongod.pid
net:
   bindIp: 127.0.0.1
   port: 27017
   maxIncomingConnections: 65536
   wireObjectCheck: true
   ipv6: false
storage:
   dbPath: /data/mongodb/db
   indexBuildRetry: true
   journal:
      enabled: true
   directoryPerDB: false
   engine: mmapv1
   syncPeriodSecs: 60
   mmapv1:
      quota:
         enforced: false
         maxFilesPerDB: 8
      smallFiles: true
      journal:
         commitIntervalMs: 100
   wiredTiger:
      engineConfig:
         cacheSizeGB: 8
         journalCompressor: snappy
         directoryForIndexes: false
      collectionConfig:
         blockCompressor: snappy
      indexConfig:
         prefixCompression: true
operationProfiling:
   slowOpThresholdMs: 100
   mode: off
```

如果你的架构模式为replication Set，那么还需要在所有的“复制集”members上增加如下配置：

```
replication:
   oplogSizeMB: 10240
   replSetName: rs0
   secondaryIndexPrefetch: all
```

如果为sharding Cluster架构，则需要在shard节点增加如下配置：

```
sharding:
   clusterRole: shardsvr
   archiveMovedChunks: true
```

当然，一个mongod实例即可以为“复制集”的member之一，也可以作为sharding集群中的一个分片，这取决你的架构模式。

mongod进程可以做为“config server”实例，只需要将“clusterRole: configsvr”即可；由此可见，一个mongod实例可以

为“单点实例”、“config server”、“sharding server” + “replication set member”其中一个角色，建议使用不同的配置文件启动它。

路由节点

```
systemLog:
    quiet: false
    path: /data/mongodb/logs/mongod.log
    logAppend: false
    destination: file
processManagement:
    fork: true
    pidFilePath: /data/mongodb/mongod.pid
net:
    bindIp: 127.0.0.1
    port: 37017
    maxIncomingConnections: 65536
    wireObjectCheck: true
    ipv6: false
replication:
    localPingThresholdMs: 15
sharding:
    autoSplit: true
    configDB: m1.com:27018,m2.com:27018,m3.com:27018
    chunkSize: 64
```

mongos实例不需要存储实际的数据，对内存有一定的消耗，在sharding架构模式下使用；mongos需接收向客户端请求（后端的sharded和replication set则不需要让客户端知道），它可以将客户端请求转发到一个分片集群上（分片集群基于复制集）延迟相对较小的secondary上，同时还负责chunk的分裂和迁移工作。

其他

repair

“修复”数据库，当mongodb运行一段时间之后，特别是经过大量删除、update操作之后，我们可以使用repair指令对数据存储进行“repair”，它将整理、压缩底层数据存储文件，重用磁盘空间，相当于数据重新整理了一遍，对数据优化有一定的作用。

如果mongod没有开启journaling日志功能，repair指令可以在系统异常crash之后，用于整理数据、消除损坏数据；如果开启了journaling日志功能，我们则需不要使用repair来修复数据，因为journal就可以帮助mongod恢复数据。在replication set模式下，可以使用repair，但是通常可以直接删除旧数据，使用“数据同步”操作，即可达到“恢复”、“整理”数据的目的，效果和repair一样，而且效率更高。

repair需要磁盘有一定的剩余空间，为当前database数据量 + 2GB，可以通过使用“--repairpath”来指定repair期间存储临时数据的目录。repair指令还会重建indexes，可以降低索引的数据大小。

如果mongod意外crash，需要首先正常启动mongod，让根据journal日志恢复完数据之后，才能执行repair；如果journal日志有数据尚未恢复，那么使用repair指令启动mongod将会失败。

repair时需要关闭mongod进程，执行完毕后再启动。

```
mongod --dbpath=/data/mongodb/db --repair
```

mongodb比较倾向于使用shell来repair特定的database，这个操作相对比较可控，其内部工作机制一样。

```
1. >./mongo
2. >user mydatabase;
3. >db.repairDatabase();
```

mongodump与mongorestore

我们通常会使用到mongodb数据的备份功能，或者将一个备份导入到一个新的mongod实例中（数据冷处理），那么就需要借助这两个指令。

mongodump将整个databases全部内容导出到一个二进制文件中，可以在其他mongod使用mongorestore来加载整个文件。需要注意mongodump不会导出"local"数据库中的数据，当然这个local库对恢复数据也没有太大意义。

"-u"参数指定访问database的用户名，"-p"指定密码，"--host"和"--port"指定mongod实例的位置，"--db"指定需要dump的数据库，如果不指定则dump所有数据库，"--collection"指定需要dump的集合表，如果不指定则dumpl整个db下的所有collections；"--query <json>"指定dump时的查询条件，"--out"指定结果输出文件的路径：

```
1. >./mongodump --host m1.com --port 27017 -u root -p pass --out /data/mongodb/backup/dump_2015_10_10
```

mongorestore则将dump的数据文件导入到database，mongorestore可以创建新的database或者将数据添加到现有的database中。如果将数据restore到已经存在的database中，mongorestore仅执行insert，不会执行update，如果数据库中已经存在相同的"_id"数据，mongorestore不会覆盖原有的document。mongorestore会重新创建indexes，所有的操作都是insert而不会update。

基本指令类似于mongodump，"--db"指定需要将数据restore到哪个db中，如果此db不存在，则创建；如果没有指定"--db"，mongorestore则根据原始数据所属的db重新创建，这可能会导致数据覆盖。"--drop"表示在restore数据之前，首先删除目标db中原有的collections，--drop不会删除那些在dump文件中没有的collection。"--stopOnError"表示出错时强制退出。

```
1. ./mongorestore --db mydatabase /data/mongodb/backup/dump_2015_10_10
```

mongoimport和mongoexport

mongoexport将数据导出为JSON或者CSV格式，以便其他应用程序解析。

因为mongodb数据是BSON格式，有些数据类型是JSON不具有的，所以导出JSON格式会仍然会丢失数据类型；所以如果导出的数据是准备给其他mongodb恢复数据，那么建议使用mongodump和mongorestore。

mongostat指令可以间歇性的打印出当前mongod实例中"数据存储"、"flush"、读写次数、网络输出等参数，是查看mongod性能的有效手段。mongotop可以根据查看各个database下读写情况。