MONGODB 技术文档

PCONLINE.COM.CN

Dba/lsliang

2011-06-07

1.mongodb 的来历：

Mongodb　名字来源于 humongous (巨大无比的）

高可用，可扩展，高性能的面向文档数据库，用 c++开发。

主要功能：

1）面向文档的存储

2）全索引支持

3）　复制及高可用支持

5)　　支持类 sql 语句的查询

6）快速替换更新，原子操作

7）支持 map/reduce

8）　gridFS

9) 社区支持一般不会超过 24 小时。

2. 安装：

1） 下载地址： www.mongodb.org/dowloads 当前最新版本 1.8.1

2） 展开的指定目录 ：

cd /data

Tar -xf mongodb-linux-x86_64-1.8.1.tar

Ln -s mongodb-linux-x86_64-1.8.1 mongodb

Cd mongodb

Mkdir data log

Bin/mongod --fork --dbpath /data/mongodb/data --logpath /data/mongodb/log

3.生产环境最佳实践：

1）linux 系统：

1】 关闭文件系统/分区的 atime 选项

Vi /etc/fstab

在对应的分区项后面添加 noatime ,nodiratime

LABEL=/1 / ext3 defaults 1 1

LABEL=/data1 /data ext3 defaults,noatime,nodiratime 1 2

2】 设置文件句柄上线 4 k＋

Vi /etc/security/limit.conf

* soft nproc 65536

* hard nproc 65536

* soft nofile 65536

* hard nofile 65536

3】 不要使用 large vm page （不要使用大内存页选项）

Linux 大内存页 参考：http://linuxgazette.net/155/krishnakumar.html

4】用 dmesg 查看主机的信息。

2） linux 文件系统的选择：

Mongodb 采用预分配的大文件来存储数据

1】 ext4

2】 xfs

3）内核版本：

网络上对 2.6.33-31 以及 2.6.32 的表现持换衣态度， 而强力推荐 2.6.36

4.命令行参数：

1） 一般命令行参数：

-h | --help	Shows all options
-f | --config <file>	Specify a configuration file to use
--port <portno>	Specifies the port number on which Mongo will listen for client connections. Default is 27017
--dbpath <path>	Specifies the directory for datafiles. Default is /data/db or c:\data\db
--fork	Fork the server process
--bind_ip <ip>	Specifies a single IP that the database server will listen for
--directoryperdb	Specify use of an alternative directory structure, in which files for each database are kept in a unique directory. (more details) (*since 1.3.2*)
--quiet	Reduces amount of log output (more details)
--nohttpinterface	Disable the HTTP interface (localhost:28017)
--rest	Allow extended operations at the Http Interface
--logpath <file>	File to write logs to (instead of stdout). You can rotate the logs by sending SIGUSR1 to the server.
--logappend	Append to existing log file, instead of overwritting
--repairpath <path>	Root path for temporary files created during database repair. Default is dbpath value.
--cpu	Enables periodic logging of CPU utilization and I/O wait
--noauth	Turns off security. This is currently the default
--auth	Turn on security
-v[v[v[v[v]]]] | --verbose	Verbose logging output (-vvvvv is most verbose, -v == --verbose)
--objcheck	Inspect all client data for validity on receipt (useful for developing drivers)
--quota	Enable db quota management. This limits (by default) to 8 files per DB. This can be changed through the --quotaFiles parameter
--quotaFiles	Maximum number of files that will be opened per Database. See --quota
--syncdelay arg (=60)	seconds between disk syncs (0=never, but not recommended)
--diaglog <n>	Set oplogging level where n is 0=off (default) 1=W 2=R 3=both 7=W+some reads
--nocursors	Diagnostic/debugging option
--nohints	Ignore query hints
--noscripting	Turns off server-side scripting. This will result in greatly limited functionality
--notablescan	Turns off table scans. Any query that would do a table scan fails
--noprealloc	Disable data file preallocation
--smallfiles	Use a smaller default file size
--nssize <MB>	Specifies .ns file size for new databases

--sysinfo	Print system info as detected by Mongo and exit
--nounixsocket	disable listening on unix sockets (will not create socket files at /tmp/mongodb-<port>.sock)
--upgrade	Upgrade database files to new format if necessary (required when upgrading from <= 1.0 to 1.1+)

2） master、slave 参数：

--master	Designate this server as a master in a master-slave setup
--slave	Designate this server as a slave in a master-slave setup
--source <server:port>	Specify the source (master) for a slave instance
--only <db>	Slave only: specify a single database to replicate
--arbiter <server:port>	Address of arbiter server
--autoresync	Automatically resync if slave data is stale
--oplogSize <MB>	Custom size for replication operation log

3）replica sets 参数：

--replSet <setname>[/<seedlist>]	Use replica sets with the specified logical set name. Typically the optional seed host list need not be specified.
--oplogSize <MB>	Custom size for replication operation log
--fastsync	If the node is started with a recent copy of data from another node in the set (including the oplog) it won't need a full resync. You can let it know to skip the resync by running the mongod with --fastsync.

5.主从切换：

Master/slave 模式 mongodb 不支持链式复制 a--b--c--d 这种模式可以支持 a--b, a-c,a-d 这种一主多从的模式。

1） 从库的 local 库会存放主库以及复制的一些信息，命令行参数：

```
Use local
Db.sources.find()
{
"_id" : ObjectId("4dc901bc28debd38d08d3105"),
"host" : "192.168.74.7",
"source" : "main",
"only" : "price_front_cache",
"syncedTo" : { "t" : 1305278821000, "i" : 1 },
"localLogTs" : { "t" : 0, "i" : 0 }
}
```

查看复制状态的命令：

主库执行：

```
> db.printReplicationInfo()
configured oplog size:   2048MB
log length start to end: 2404212secs (667.84hrs)
oplog first event time:  Tue May 10 2011 19:54:51 GMT+0800 (CST)
oplog last event time:   Tue Jun 07 2011 15:45:03 GMT+0800 (CST)
now:                     Tue Jun 07 2011 15:45:06 GMT+0800 (CST)
```

从库执行：

```
> db.printSlaveReplicationInfo()
source:   192.168.74.7
 syncedTo: Fri May 13 2011 17:27:01 GMT+0800 (CST)
```

= 2153939secs ago (598.32hrs)

db._adminCommand({ serverStatus : 1 , repl : N})

其中 N 的取值：

0 - none 只打印本机信息 serverstatus

1 - local (doesn't have to connect to other server) 0 + repl 信息 只在本地采集不会连到 master

2 - remote (has to check with the master) 0 + repl 信息 登录到 master 上检测。

db._adminCommand({ serverStatus : 1 , repl : 2}) 的实例：


```
{
        "host" : "test740-10.pconline.com.cn",
        "version" : "1.8.0",
        "process" : "mongod",
        "uptime" : 2408222,
        "uptimeEstimate" : 2408248,
        "localTime" : ISODate("2011-06-07T07:54:36.500Z"),
        "globalLock" : {
                "totalTime" : 2408221511065,
                "lockTime" : 977471906,
                "ratio" : 0.00040588953362837775,
                "currentQueue" : {
                        "total" : 0,
                        "readers" : 0,
                        "writers" : 0
                },
                "activeClients" : {
                        "total" : 1,
                        "readers" : 0,
                        "writers" : 1
                }
        },
        "mem" : {
                "bits" : 64,
                "resident" : 2651,
                "virtual" : 12601,
                "supported" : true,
                "mapped" : 12315
        },
        "connections" : {
                "current" : 3,
                "available" : 253
        },
        "extra_info" : {
                "note" : "fields vary by platform",
```

```
                "heap_usage_bytes" : 2249584,
                "page_faults" : 231124
        },
        "indexCounters" : {
                "btree" : {
                        "accesses" : 534363,
                        "hits" : 534363,
                        "misses" : 0,
                        "resets" : 0,
                        "missRatio" : 0
                }
        },
        "backgroundFlushing" : {
                "flushes" : 40135,
                "total_ms" : 801276,
                "average_ms" : 19.96451974585773,
                "last_ms" : 7,
                "last_finished" : ISODate("2011-06-07T07:54:22.341Z")
        },
        "cursors" : {
                "totalOpen" : 0,
                "clientCursors_size" : 0,
                "timedOut" : 1
        },
        "network" : {
                "bytesIn" : 1389663948,
                "bytesOut" : 4430888859,
                "numRequests" : 4311894
        },
        "repl" : {
                "ismaster" : false,
                "sources" : [
                        {
                                "host" : "192.168.74.7",
                                "syncedTo" : {
                                        "time" : ISODate("2011-05-13T09:27:01Z"),
                                        "inc" : 1
                                },
                                "masterFirst" : ISODate("2011-05-10T11:54:51Z"),
                                "masterLast" : ISODate("2011-06-07T07:54:34Z"),
                                "lagSeconds" : 2154453
                        }
                ]
        },
        "opcountersRepl" : {
                "insert" : 4300718,
```

```
                "query" : 0,
                "update" : 0,
                "delete" : 4803717,
                "getmore" : 0,
                "command" : 0
        },
        "opcounters" : {
                "insert" : 0,
                "query" : 83,
                "update" : 0,
                "delete" : 0,
                "getmore" : 1056,
                "command" : 1249
        },
        "asserts" : {
                "regular" : 0,
                "warning" : 0,
                "msg" : 0,
                "user" : 4310615,
                "rollovers" : 0
        },
        "writeBacksQueued" : false,
        "ok" : 1
}
```


2） 主从切换步骤：

1】Halt writes on A (using the fsync command)
2】Make sure B is caught up
3】Shut down B
4】Wipe local.* on B to remove old local.sources
5】Start up B with the --master option
6】Do a write on B (primes the oplog to provide a new sync start point).
7】Shut down B. B will now have a new set of local.* files.
8】Shut down A and replace A's local.* files with a copy of B's new local.* files. Remember to compress the files before/while copying them – they can be quite large.
9】Start B with the --master option
10】Start A with all the usual slave options plus --fastsync

3） 用主库的一个物理备份快速启动、构建从库。

主库只读情况下，做一个目录镜像，然后 copy 到从库所对应的数据目录，
从库启动的时候指定参数 --fastsync

4) Oplog 尺寸：

Mongodb 的 oplog 是循环使用的，（实质上是一个环形队列，fixed length collection）

如果从库停机维护或从新从主库同步数据，要确保 oplog 的尺寸 能够装的下期间生成的数据变更的日志。

Mongodb 的 oplog 一旦建成不能修改其大小，需要停库，删除旧日志，然后指定新的尺寸。

我们的原则： oplog 的大小要为从库预留2-4小时个宕机时间，目前推荐位6G

Oplog 为预分配文件，如果指定的 oplog 尺寸很大，在启动时，会花费很长时间来建立日志文件，可以考虑手工先建立好这些文件：

假设我们指定 oplog 为10g

```
For i in （0 ..5)
Do
Head -c 2147483648 /dev/zero > local.$i
Done
```

把这些文件挪到数据目录，然后命令行启动参数指定 --oplog 10000

5） 为从库重新指定主库：

从库不带 --slave 和 --source 启动，

从库启动 shell 修改 local 数据库的 sources 表

```
> db.sources.update({host : "prod.missisippi"}, {$set : {host : "prod.mississippi"}})
```

从库用新的命令行参数 --slave --source 启动。

6.replica sets　模式：

1）　replica sets 模式是支持自动 failover ，读写分离的。

2）　启动选项：

bin/mongod --rest --replSet myset --dbpath /data/mongodb/mongodb/data_19 --port 27019 --fork --logpath /data/mongodb/mongodb/data_19/std.log

--rest　-- replSet　$setname 是必须的参数

$setname 是复制组的标示。 因为我们的局域网事互通的，所以线上配置的时候切不可与现有的复制组同名，否则会导致加入到现有的复制组。

3）　replica set 的驱动程序连接池的写法：

Mongodb://username:password@host1:port1,host2:port2.....hostn:portn/database?options

Python 的一个写法如下：

Import　pymongo as pm

Conn = pm.Connection("mongodb://lsl:passwd@192.168.10.239:27017,102.168.10.239:27018/lsl?connect=replicaSet;slaveok=true")

Mongodb 的驱动程序在本地实现了一个连接池，所以应用程序本身不需要关系那台机器是 master 那台是 slave 。而主机端口的列表不需要全部列出来，需要列出其中的几个就可以，驱动连上数据库后，会对主机发出一个 isMaster（）的查询，数据库会返回复制组里德所有的主机的列表以及主机的角色，驱动程序会缓存这些数据。

一个实例：

```
myset:SECONDARY> db.isMaster()
{
        "setName" : "myset",
        "ismaster" : false,
        "secondary" : true,
        "hosts" : [
                "192.168.10.239:27021",
                "192.168.10.239:27020",
                "192.168.10.239:27017",
                "192.168.10.239:27019",
```

```
                "192.168.10.239:27018"
        ],
        "primary" : "192.168.10.239:27020",
        "maxBsonObjectSize" : 16777216,
        "ok" : 1
}
myset:SECONDARY>
```

4） 向现有的复制组中添加新成员：

1】新机器用--rest --replSet $setname 选项启动，$setname 指定为组的 setname ，然后再 primary 机器上执行 rs.add('host:port") 新机器会自动同步全部的数据。

2】以用其他备机的一个物理全备份来制作新的成员机，

其中一台备机重新启动的时候，加上命令行参数 --fsync 或者直接在从机上执行：

```
myset:SECONDARY> db.runCommand({fsync:1})
{ "numFiles" : 10, "ok" : 1 }
```

然后开直接将这个从机的数据目录复制到目标机的相应的目录下，

复制完后， 从机执行：

```
myset:SECONDARY> db.runCommand({fsync:0})
{ "numFiles" : 10, "ok" : 1 }
```

目标机器用命令 1】中的参数启动，需要增加额外的命令行昂参数--fastsync 在 primary 中执行 rs.add() 完成。

3】主意事项： 要确保复制组中的 oplog 的尺寸要足够大，能够容纳，从机只读/备份、或者全同步过程新产生的数据变更的日志。

5）primary 的选举：

主库一旦宕机，复制组会自动重新选举 primary 数据库。

原则：

1】取得 maxlocalOpordinal （本地 oplog 的最大序列号）

2】如果半数以上的从机没有启动，则当前机保持 slave 状态并停机等待人工干预

3】如果 maxlocalOpordinal 很旧，停机等待人工干预。

4】否则，启用一致性协议，选出有最大 maxlocalOpordinal 的从机当选为主库

一致性协议的原则：

1】查询所有 slave 的 maxapliiedoptime

2】选举自己为 master： 有最新时间，并且能够看到超过半数的机器

A）如果阻塞在最新时间这里，延时一小会，再在投票自己

B）广播选举自己为 master 的信号

3】如果我收到了别人的广播信息，并且我的时间最新，回复 no

4】我必须收到别人对我的投票超过半数为 yes

5】如果收到别人的回复为 yes ，则对所有的主机的回复为 no，选举自己为 yes

6】如果必要，延迟一会，重复上述动作。

6）数据丢失与 oplog:

replica set 中，如果 primary 在进行大数据量的导入，这时 slave 可能无法跟上 master 的 oplog 的进度，如果此时 master 宕机，系统会重新选出 master ，

系统有个原则： 任何比 master 的数据更新的数据都是无效的。

根据这个原则，如果宕机的 master 重新接入系统机会出现这部分数据无效，

从 1.7.5 以后的系统里，已经增加了这部分数据的处理能力，会在数据目录下生成一个 rollback 的目录，里面的文件名为 db.collection.$timestamp.bson

这部分数据时可以用 mongorestor 导入到系统中去的，这样就为这部分数据的手工恢复提供了一种方式。

建议不要导入的原表，而是导入到一个临时表中，这样可操作性会更高。

示意图：

Master ： 宕机

|——|————>

|————————————————————————————————————》|

Slave： 当前时间

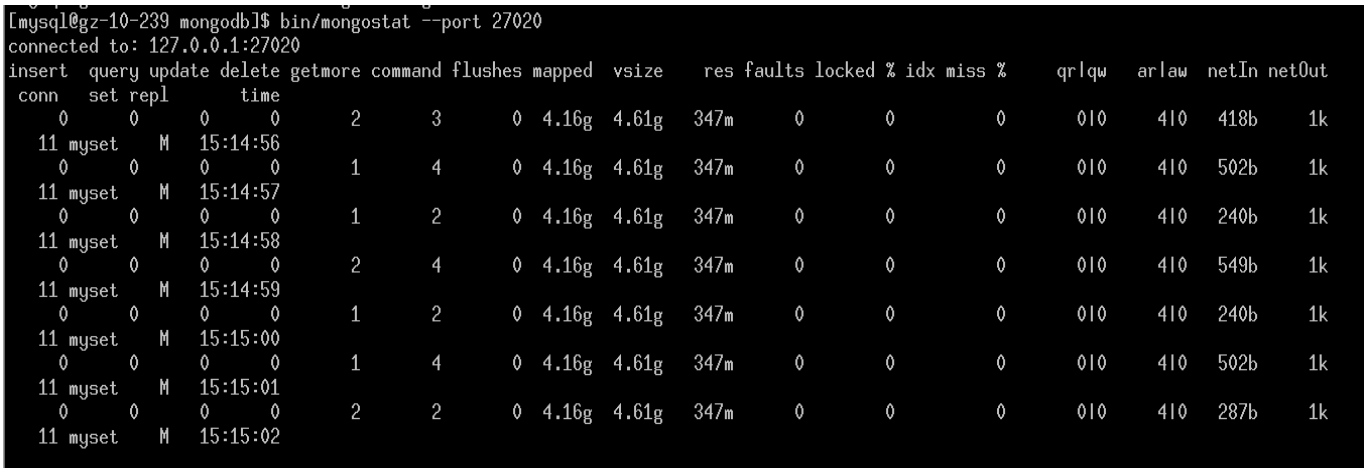
|————————|

数据丢失

当原来主库重新接入复制组后，这部分数据会写到一个 roallback 目录中，这部分数据需要人工干预。

7.系统监控：

1) mongostat



```
[mysql@gz-10-239 mongodb]$ bin/mongostat --port 27020
connected to: 127.0.0.1:27020
insert  query update delete getmore command flushes mapped  vsize    res faults locked % idx miss %     qr|qw   ar|aw  netIn netOut
 conn   set repl       time
    0      0     0      0       2       3       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   418b     1k
   11 myset    M   15:14:56
    0      0     0      0       1       4       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   502b     1k
   11 myset    M   15:14:57
    0      0     0      0       1       2       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   240b     1k
   11 myset    M   15:14:58
    0      0     0      0       2       4       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   549b     1k
   11 myset    M   15:14:59
    0      0     0      0       1       2       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   240b     1k
   11 myset    M   15:15:00
    0      0     0      0       1       4       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   502b     1k
   11 myset    M   15:15:01
    0      0     0      0       2       2       0  4.16g  4.61g   347m      0        0          0       0|0     4|0   287b     1k
   11 myset    M   15:15:02
```

各个字段的定义：

Fields:

insert - # of inserts per second (* means replicated op)

query - # of queries per second

update - # of updates per second

delete - # of deletes per second

getmore - # of get mores (cursor batch) per second

command - # of commands per second (on a slave, it's local|replicated)

flushes - # of fsync flushes per second

mapped - amount of data mmaped (total data size) megabytes

vsize - virtual size of process in megabytes

res - resident size of process in megabytes

faults - # of pages faults/sec (linux only)

locked - percent of time in global write lock

idx miss - percent of btree page misses (sampled)

qr | qw - queue lengths for clients waiting (read|write)

ar | aw - active clients (read|write)

netIn - network traffic in - bits

netOut - network traffic out - bits

conn - number of open connections

set - replica set name

repl - replication type

M - master

SEC - secondary

REC - recovering

UNK - unknown

SLV - slave

2） database profile ：

这个我也没有搞明白，暂时先空着吧，

3）http console 与 mongo shell

启动端口+1000 是 mongodb 的 http console 可以简单的查看一些性能数据

Mongodb shell mongo 的一些监控命令：

db.serverStatus() See the serverStatus Command command page.

db.stats() Stats on the current database. Command takes some time to run, typically a few seconds unless the .ns file is very large (via use of --nssize). While running other operations may be blocked.

db.foo.find().explain() explain plan

help db.help()

db.foo.help()

4）Database Record/Replay （类似 oracle 的那个功能差不多）

可以记录数据库的操作，并且记录下来，以后的某个时间可以重放这些操作，可以重现某些问题。

启动日志记录：

db._adminCommand({ diagLogging : 1})

关闭日志记录：

db._adminCommand({ diagLogging : 0 })

日志文件会生成在 $dbpath 下面 的 dialog.bin_

数字参数的取值：

0 off. Also flushes any pending data to the file.

1 log writes

2 log reads

3 log both

Note: if you log reads, it will record the findOnes above and if you replay them, that will have an effect!

日志数据的重演：

nc "database_server_ip" 27017 < "somelog.bin" | hexdump -c

nc 是 linux 命令 netcat

5） 检查 mongodb 的内存使用情况：

> db.serverStatus()

> db.serverStatus().mem

> db.serverStatus().extra_info

可以通过查看 mem.virsual 与 mem.mapped 的大小来来确认是否有内存泄漏。一般来说，两者的差距相对于整个主机的内存来说是微不足道的，可以通过一段时间来取他们的变化量，如果变化量一致增大，则基本可以确认是内存泄漏了。

6）collstat 表的统计信息：

```
> db.foo.stats()
{
        "ns" : "test.foo",
        "count" : 9,                    // number of documents
        "size" : 432,                   // collection size
        "avgObjSize" : 48,              // average object size in bytes
        "storageSize" : 3840,
        "numExtents" : 1,               // extents are contiguously allocated
                                        // chunks of datafile space
        "nindexes" : 2,
        "lastExtentSize" : 3840,
        "paddingFactor" : 1,      // padding can make updates faster
                                        // if documents grow
        "flags" : 1,
        "totalIndexSize" : 16384,
```

```
        "indexSizes" : {
                "_id_" : 8192,
                "x_1" : 8192
        },
        "ok" : 1
}
```

7）mongoniff　tcp/ip 包的一个 dump 工具：

```
[root@gz-10-239 mongodb]# bin/mongosniff    --help
Usage: mongosniff [--help] [--forward host:port] [--source (NET <interface> | (FILE | DIAGLOG) <filename>)] [<port0> <port1> ... ]
--forward           Forward all parsed request messages to mongod instance at
                      specified host:port
--source            Source of traffic to sniff, either a network interface or a
                      file containing previously captured packets in pcap format,
                      or a file containing output from mongod's --diaglog option.
                      If no source is specified, mongosniff will attempt to sniff
                      from one of the machine's network interfaces.
--objcheck          Log hex representation of invalid BSON objects and nothing
                      else.    Spurious messages about invalid objects may result
                      when there are dropped tcp packets.
<port0>...          These parameters are used to filter sniffing.    By default,
                      only port 27017 is sniffed.
--help              Print this help message.
```

8）　> db.serverStatus() 命令：　各个字段的意义：

serverStatus 命令的内容相当丰富:

Field	Example Value	Explanation
host	my.node.com	The hostname of this server
version	1.8.0-rc1-pre-	The version number of this server
process	mongod	What is the process? (mongod, mongos)

uptime	14143	Uptime in seconds
uptimeEstimate	12710	Uptime based on MongoDB's internal coarse grained timer
localTime	ISODate("2011-03-01T05:30:16.682Z")	The local time at this server (time is in UTC).
globalLock.totalTime	14143457121	The number of microseconds since startup that the global lock was created
globalLock.lockTime	17166	The number of microseconds that the global lock has been held since it was created
globalLock.ratio	0.0000012137060870720337	The ratio between lockTime & totalTime
globalLock.currentQueue.total	12	The current number of operations queued waiting for the global lock
globalLock.currentQueue.readers	10	The current number of operations queued waiting on a read lock
globalLock.currentQueue.writers	2	The current number of operations queued waiting for a write lock
globalLock.activeClients.total	3	Total number of active clients connected to this server
globalLock.activeClients.readers	2	The total number of active clients currently performing read operations
globalLock.activeClients.writers	1	The total number of active clients currently performing write operations
mem.bits	64	Is this a 32 or 64 bit architecture?
mem.resident	20	number of megabytes resident. It is typical over time, on a dedicated database server, for this number to approach the amount of physical ram on the box.
mem.virtual	2502	virtual megabytes for the mongod process. Generally virtual should be a little larger than mapped, but if virtual is many gigabytes larger, that could indicate a memory leak
mem.supported	true	Whether or not this machine supports extended memory info. If this is false, other values in 'mem' may not be present
mem.mapped	80	As MongoDB memory maps all the data files, this number is likely similar to your total database(s) size.
connections.current	23	The number of currently active connections to this server
connections.available	50	The number of available connections remaining
extra_info.heap_usage_bytes	234342	The number of bytes of heap used by this process. Only available on linux
extra_info.page_faults	2523	The number of page faults on this process. Only available on linux
indexCounters.btree.accesses	2342	The number of times the btree indexes have been accessed
indexCounters.btree.hits	2340	The number of times a btree access found the

		document it was looking for
indexCounters.btree.misses	2	The number of times a btree access could not find the document it was looking for
indexCounters.btree.resets	0	The number of times the index counters have been reset to 0
indexCounters.btree.missRatio	0	The ratio of misses to hits on the btree
backgroundFlushing.flushes	214	The number of times the database has flushed writes to disk
backgroundFlushing.total_ms	1357	The total number of ms that the database has spent flushing data to disk
backgroundFlushing.average_ms	6.341121495327103	The average number of ms it takes to perform a single flush
backgroundFlushing.last_ms	7	The number of ms that the last flush took to complete
backgroundFlushing.last_finished	ISODate("2011-03-01T05:29:44.124Z")	The timestamp from when the last flush was completed
cursors.totalOpen	0	The total number of cursors that the server is maintaining for clients
cursors.clientCursors_size	[deprecated] Same as cursors.totalOpen	
cursors.timedOut	0	The number of cursors that have timed out since this server was started
network.bytesIn	1430	The total number of bytes sent to this database
network.bytesOut	2140	The total number of bytes sent from this database
network.numRequests	20	The total number of requests that have been sent to this database
repl.setName	mySet	The name of the replica set that this server is a part of
repl.ismaster	true	Whether or not this node is the master of the replica set
repl.secondary	false	Whether or not this node is a secondary of the replica set
repl.hosts	["my.node.com:27017", "other.node.com:27017", "third.node.com:27017"]	The set of hosts in this replica sets
opcounters.insert	0	The total number of inserts performed since this process started
opcounters.query	9	The total number of queries performed since this process started
opcounters.update	0	The total number of updates performed since this process started
opcounters.delete	0	The total number of deletes performed since this process started
opcounters.getmore	0	The total number of times getMore has been called on any cursor since this process started

command	13	The total number of other commands performed since this process started
asserts.regular	0	The number of regular asserts raised since this process started
asserts.warning	0	The number of warnings raised since this process started
asserts.msg	0	The number of message asserts. These are internal server errors that have a well defined text string. Stack traces are logged for these
asserts.user	0	The number of user asserts. These are errors that can be generated by a user such as out of disk space or duplicate key
asserts.rollovers	0	The number of times the assert counters have rolled over since this process started
writeBacksQueued	false	Whether or not there are any journal entries waiting to be flushed to disk
dur.commits	0	The number of commits to the journal that have occured in the last interval
dur.journaledMB	0	MBs of data written to the journal in the last interval
dur.writeToDataFilesMB	0	MBs of data written from journal to data files in the last interval
dur.commitsInWriteLock	0	The number of commits in the last interval which were in a write lock. Commits in a write lock are undesirable
dur.earlyCommits	0	Number of times a commit was requested before the scheduled time
dur.timeMs.dt	3011	The time length of the interval over which the dur statswere collected
dur.timeMs.prepLogBuffer	0	The amount of time spent preparing to write to the journal
dur.timeMs.writeToJournal	0	The amount of time spent actually writing to the journal
dur.timeMs.writeToDataFiles	0	The amount of time spent writing to datafiles after journaing
dur.timeMs.remapPrivateView	0	The amount of time spent remapping copy-on-write memory mapped views
ok	1	Whether or not serverStatus returned correctly

myset:SECONDARY> db.serverStatus({repl:2})

{

"host" : "gz-10-239.pconline.ctc:27021",

"version" : "1.8.1",

```
"process" : "mongod",
"uptime" : 101274,
"uptimeEstimate" : 101034,
"localTime" : ISODate("2011-06-08T08:38:12.049Z"),
"globalLock" : {
        "totalTime" : 101273228908,
        "lockTime" : 1012792,
        "ratio" : 0.000010000589602214168,
        "currentQueue" : {
                "total" : 0,
                "readers" : 0,
                "writers" : 0
        },
        "activeClients" : {
                "total" : 0,
                "readers" : 0,
                "writers" : 0
        }
},
"mem" : {
        "bits" : 64,
        "resident" : 37,
        "virtual" : 3586,
        "supported" : true,
        "mapped" : 3231
},
"connections" : {
        "current" : 7,
        "available" : 812
},
"extra_info" : {
```

```
        "note" : "fields vary by platform",
        "heap_usage_bytes" : 469136,
        "page_faults" : 292
},
"indexCounters" : {
        "btree" : {
                "accesses" : 1,
                "hits" : 1,
                "misses" : 0,
                "resets" : 0,
                "missRatio" : 0
        }
},
"backgroundFlushing" : {
        "flushes" : 1687,
        "total_ms" : 76,
        "average_ms" : 0.045050385299347954,
        "last_ms" : 0,
        "last_finished" : ISODate("2011-06-08T08:37:21.701Z")
},
"cursors" : {
        "totalOpen" : 0,
        "clientCursors_size" : 0,
        "timedOut" : 0
},
"network" : {
        "bytesIn" : 26523460,
        "bytesOut" : 26983320,
        "numRequests" : 202507
},
"repl" : {
```

```
        "setName" : "myset",
        "ismaster" : false,
        "secondary" : true,
        "hosts" : [
                "192.168.10.239:27021",
                "192.168.10.239:27020",
                "192.168.10.239:27017",
                "192.168.10.239:27019",
                "192.168.10.239:27018"
        ],
        "primary" : "192.168.10.239:27020"
},
"opcountersRepl" : {
        "insert" : 30,
        "query" : 0,
        "update" : 0,
        "delete" : 0,
        "getmore" : 0,
        "command" : 0
},
"opcounters" : {
        "insert" : 0,
        "query" : 9,
        "update" : 0,
        "delete" : 0,
        "getmore" : 0,
        "command" : 202500
},
"asserts" : {
        "regular" : 0,
        "warning" : 0,
```

```
                "msg" : 0,
                "user" : 1,
                "rollovers" : 0
        },
        "writeBacksQueued" : false,
        "ok" : 1
}
```

8.一般的数据库操作：

Note one may also create .js scripts to run in the shell for administrative purposes.

help show help

show dbs show database names

show collections show collections in current database

show users show users in current database

show profile show most recent system.profile entries with time >= 1ms

use <db name> set curent database to <db name>

db.addUser (username, password)

db.removeUser(username)

db.cloneDatabase(fromhost)

db.copyDatabase(fromdb, todb, fromhost)

db.createCollection(name, { size : ..., capped : ..., max : ... })

db.getName()

db.dropDatabase()

db.printCollectionStats()

db.currentOp() displays the current operation in the db

db.killOp() kills the current operation in the db

db.getProfilingLevel()

db.setProfilingLevel(level) 0=off 1=slow 2=all

db.getReplicationInfo()

db.printReplicationInfo()

db.printSlaveReplicationInfo()

db.repairDatabase()

db.version() current version of the server

db.shutdownServer()

db.foo.drop() drop the collection

db.foo.dropIndex(name)

db.foo.dropIndexes()

db.foo.getIndexes()

db.foo.ensureIndex(keypattern,options) - options object has these possible fields: name, unique, dropDups

db.foo.find([query] , [fields]) - first parameter is an optional query filter. second parameter is optional set of fields to return.

e.g. db.foo.find(

{ x : 77 } ,

{ name : 1 , x : 1 })

db.foo.find(...).count()

db.foo.find(...).limit(n)

db.foo.find(...).skip(n)

db.foo.find(...).sort(...)

db.foo.findOne([query])

db.foo.getDB() get DB object associated with collection

db.foo.count()

db.foo.group({ key : ..., initial: ..., reduce : ...[, cond: ...] })

db.foo.renameCollection(newName) renames the collection

db.foo.stats()

db.foo.dataSize()

db.foo.storageSize() - includes free space allocated to this collection

db.foo.totalIndexSize() - size in bytes of all the indexes

db.foo.totalSize() - storage allocated for all data and indexes

db.foo.validate() (slow)

db.foo.insert(obj)

db.foo.update(query, object[, upsert_bool])

db.foo.save(obj)

db.foo.remove(query) - remove objects matching query
remove({}) will remove all

内置的操作：

$inc　自增加

{ $inc : { field : value } }

$set　修改

{ $set : { field : value } }

$unset　删除某个字段

{ $unset : { field : 1} }

$push　追加一个值

{ $push : { field : value } }

$pushAll　如果 field 是个数组，则把数组的值追加进去，字段不存在也生成新字段，如果不是数组则报错。

{ $pushAll : { field : value_array } }

$addToSet　字段为数组，如果不存在给定的值，则追加进去，字段不存在则创建，不是数组则报错。

{ $addToSet : { field : value } }

$rename　字段改名字

Version 1.7.2+ only.
{ $rename : { old_field_name : new_field_name } }

$pop　从数组里删除一个值

{ $pop : { field : 1　} }
removes the last element in an array (ADDED in 1.1)
{ $pop : { field : -1　} }
removes the first element in an array (ADDED in 1.1) |

The $ positional operator　位置操作符

Version 1.3.4+ only.
The $ operator (by itself) means "position of the matched array item in the query". Use this to find an array member and then manipulate it. For example:

```
> t.find()
{ "_id" : ObjectId("4b97e62bf1d8c7152c9ccb74"), "title" : "ABC",
   "comments" : [ { "by" : "joe", "votes" : 3 }, { "by" : "jane", "votes" : 7 } ] }

> t.update( {'comments.by':'joe'}, {$inc:{'comments.$.votes':1}}, false, true )
```

```
> t.find()
{ "_id" : ObjectId("4b97e62bf1d8c7152c9ccb74"), "title" : "ABC",
   "comments" : [ { "by" : "joe", "votes" : 4 }, { "by" : "jane", "votes" : 7 } ] }
```

参考： http://www.mongodb.org/display/DOCS/Updating

9. 数据库碎片整理：

Mongodb 数据库频繁更新删除，insert 后会产生碎片，影响的数据的查询性能，

而数据库本身没有提供太合适的碎片这里工具。

可以用 db.repaireDatabase() 来整理数据库，这个过程非常的慢

另外一个方式是停掉数据库，然后删除数据目录，从新从复制复制组中全同步数据，这个时候要考虑 oplog 的尺寸。

一个大体的步骤：

1.】先调用 rs.freeze(1200),将每个不想让它成为 primary 的机器让它在 1200 秒内无法成为 primary(这步也可以不做)

2. 】将 primary stepDown,不出意外新的 primary 会起来.

3. 】将原 primary kill 掉.

4. 】删掉所有 data 数据 (调用 repair 很慢,真不如干掉重新来)

5. 】再重启动原 primary 的进程

6. 】以此循环完成整个复制组的全部重建。

10.备份系统:

1) Mongodb 发行版提供了两个备份工具，一个 mongodump /mongorestore

一个是 mongoexport /mongoimport

2) mongodump/mongorestore: 数据位 bson 格式

Mongodump 可以备份一整个数据库，也可以备份一个表 mongodump 也可以直接访问数据目录，而不需要通过连接到 mongodb 数据库 dump 数据，但是数据目录需要有写锁，

Mongorestore 则把这些备份集恢复到数据库中。

Mongorestor 还可以恢复我们前面讲到的， （6） 中 master 宕机再回来时，丢失的那部分数据 rollback 目录里德东西。

Bin/mongrestore rollback/xxx.bson

3）mongoexport/mongoimport : 备份数据位 json 格式

Mongoexport 可以备份一个数据库，或者一个 biao ，也可以直接访问数据目录，而不需要通过连接到 mongodb 数据库 dump 数据，但是数据目录需要有写锁，

Mongoimport 则把这些备份恢复到数据库中。

Mongoimport 也可以恢复我们前面讲到的， （6） 中 master 宕机再回来时，丢失的那部分数据 rollback 目录里德东西。 但是需要先用 bsondump 转换一下

Bin/bsondump rollback/xxx.bson | bin/mongoimport --port xx -d dbname -c collname

11.与 mysql 的一些对比：

Mongodb 的备份系统，目前没有在线备份的工具，需要阻塞写，才能做到一致性备份

Mysql 则有开源，和收费的在线备份工具。

mongo单机是不安全的，master-slave 的切换 不能自动切换，需要人工干预　如果数据量很大，切换的时间和重新建库的时间也很长

Mysql 的主从切换配合我们的ha系统是可以做到的，大约会有5-10分钟的宕机时间

合适场景下性能比mysql 高很多，这是可以确定的

如果是大数据量，频繁更新的话，空间碎片的处理也是一个问题

目前没有很好的方式repairdatabase（） 就是个一个重新建库的过程，还需要源库一倍的空间

现行的版本，如果备份从库用dump ，大库的话，不太靠谱

如果用物理备份的话，又不能清理碎片

mysql 的碎片可以单表处理 + 主从切换 解决掉 有一定的宕机时间

mongo 如果用复制组，不存在宕机时间