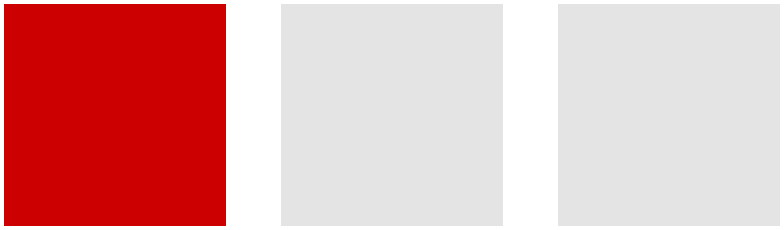


trivadis
makes IT easier.

An approach to MongoDB for (Oracle) DBAs

Trallori Stefano
Consultant, ZH-IMS
23.09.2015

Table of Contents

1 Preface

This document is meant to give a general overview of the MongoDB NoSQL database from the perspective of database administration. The intended audience is database administrators; where applicable I have underlined the possible similarities with the Oracle RDBMS.

2 General information

Whenever not explicitly specified, the whole content of this knowledge article refers to version 2.6. As of today version 3 is available: some information described here may not apply to the new version.

3 Introduction to MongoDB

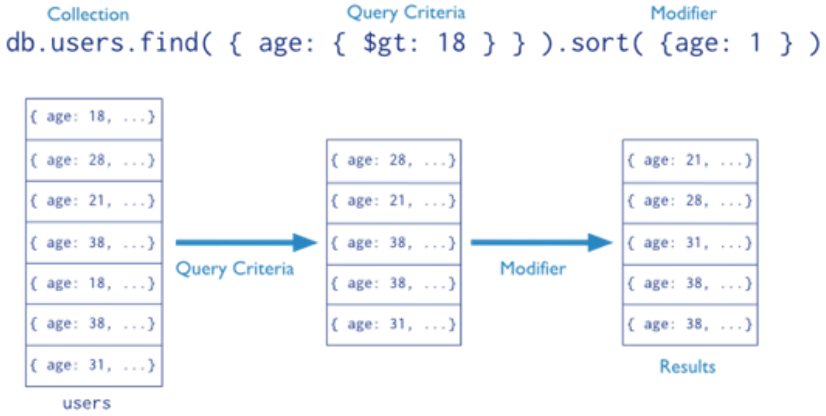
MongoDB is an open-source document-oriented database. It is available under GNU AGPL 3.0 license, but a commercial license as well as an "enterprise" edition is available. MongoDB enterprise includes more advanced security features, for example auditing and LDAP/Kerberos authentication.
MongoDB stores data in BSON format (Binary JSON) and is a NoSQL, non-ACID document-oriented database, which for an Oracle DBA makes it somehow "strange" to deal with.
The way data is organized is not by means of tables (rows and columns) and relationships like in a RDBMS: Every record in MongoDB is called "document". The "documents" are stored in "collections". A collection is a group of related documents that have a set of shared common indexes.

This is how a document looks like:

```
{
  name: "sue",                          <-- field: value
  age: 26,                              <-- field: value
  status: "A",                          <-- field: value
  groups: [ "news", "sports" ]          <-- field: value
}
```

This is a collection of said documents:

A simple query to find all documents from collection users having age greater than 18, ordered by age in ascending order looks like this:



4 The physical layer

Regarding the physical layer MongoDB has nothing particular. Each database will have one or more datafiles. The space is pre-allocated and the file size is determined automatically by MongoDB. Datafiles for all databases are created in the directory specified by the startup parameter "*dbpath*"

```
-bash-4.1$ ls -ltrh /u01/mongodb/standalone1
total 161M
-rwxrwxr-x 1 mongod mongod    6 Jan  5 14:40 mongod.lock
-rw------- 1 mongod mongod  16M Jan  5 14:40 local.ns
-rw------- 1 mongod mongod  64M Jan  5 14:40 local.0
drwxrwxr-x 2 mongod mongod 4.0K Jan  5 14:42 _tmp
-rw------- 1 mongod mongod  16M Jan  5 14:42 testdatabase.ns
-rw------- 1 mongod mongod  64M Jan  5 14:42 testdatabase.0
drwxrwxr-x 2 mongod mongod 4.0K Jan  5 14:42 journal
```

Journal files have a very important function. All changes happening on the databases are actively written (per default every 100ms) to these files (comparable to the redo log files) and later to the data files.

5 Backup and recovery techniques

There is no "*Oracle RMAN*-like" tool available. MongoDB Enterprise provides the DBAs with a cloud-based interface called *MMS* to manage backups and restores. Backup and restore of mongodb instances can be accomplished by using OS-level commands, such as LVM snapshots.

When using snapshots, be sure to have consistent snapshots of the filesystem where datafiles and journal files reside, if different.

Example:

```
-bash-4.1$ ./backup_all_db.sh
STEP 1 - create snapshot
#sudo lvcreate --size 1G --snapshot --name lv1_ext4_snap /dev/vg001/lv1_ext4
  Logical volume "lv_mongodb_snap" created
```

```
STEP 2 - archive snapshot
#dd if=/dev/vg001/lv1_ext4_snap | gzip > /tmp/db1_ext4_bck.gz
62914560+0 records in
62914560+0 records out
32212254720 bytes (32 GB) copied, 1653.09 s, 19.5 MB/s
STEP 3 - remove snapshot
#sudo lvremove --force /dev/vg001/lv_ext4_snap
  Logical volume "lv_mongodb_snap" successfully removed
```

An alternative would be "*mongodump*", which dumps a database to a binary BSON-formatted file.

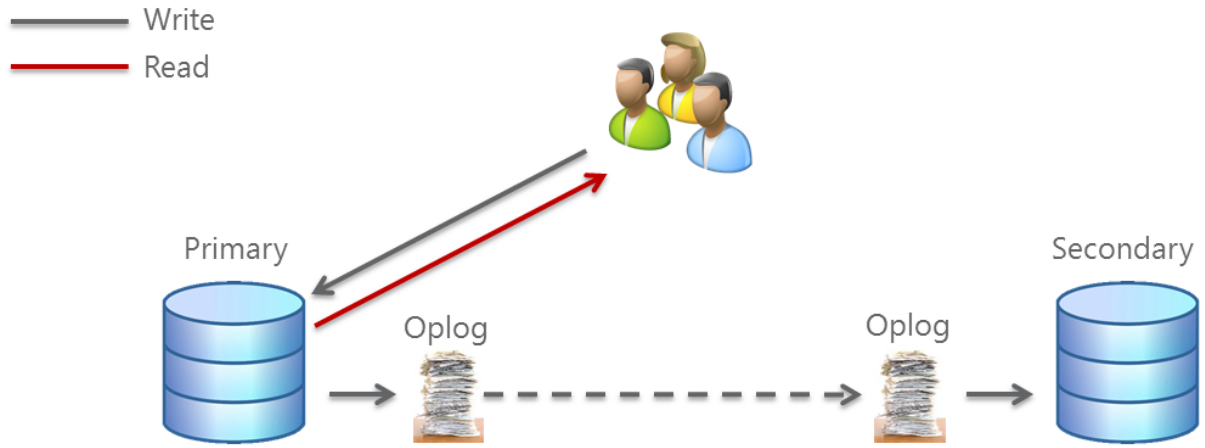
To accomplish a restore, the utility "*mongorestore*" is available to read dump files created with *mongodump*.

Snapshot backup can be restored normally using operating system commands.

Apart from using *MMS*, no point-in-time recovery is possible.

6 High availability

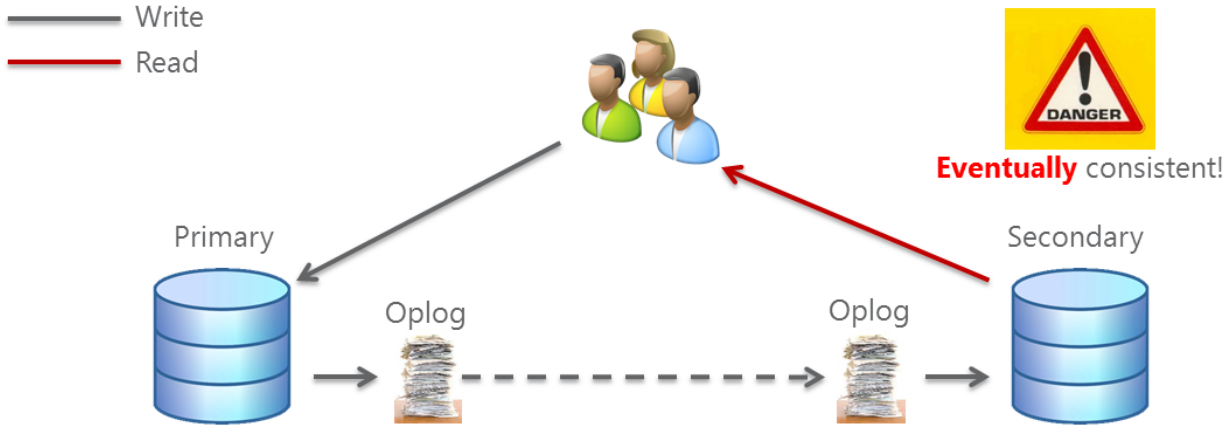
High availability in MongoDB is guaranteed by using the so-called "Replica Sets". Replica Sets are *mongodb* instances running on separate machines and having replicated data. In any given replica set there will be only one PRIMARY and one or more SECONDARY or ARBITERs. ARBITERs are used for elections of the new primary in case of an outage, much like the Oracle's "observer", and therefore do not contain any user data.



Every change made to the PRIMARY instance is logged in a capped collection called "operation log" (oplog). The oplog exists in the PRIMARY instance as well as in the SECONDARY instance(s)

Warning: By default, when a write occurs, the user is acknowledged as soon as the write is successful in the PRIMARY instance memory. This means that the application is by default unaware whether the information has also been written to the secondary instance(s) memory. This behaviour can be changed. See “Write Concern” in the MongoDB manuals for more information. In addition, the drivers used to access MongoDB can use a "Read Preference". This means they could redirect all reads to any given SECONDARY instance, possibly reading stale data.

It is possible to give different priorities to different SECONDARY instances. Members with a higher priority will more likely become primary in case of a failover. It is also possible to have DELAYED members: SECONDARY instances having changes applied with a defined delay. These will have priority 0 (=will never become primary) but will not be hidden from the clients. This means that end users can potentially read stale data from them! Hence, a good practice would be to mark DELAYED members as "HIDDEN".



The number of replica set members has to be uneven.

7 Partitioning

In MongoDB, partitioning is achieved by means of "sharding" (horizontal scaling). Each shard contains a portion of data, which will thereby be splitted amongst several instances.

In MongoDB; a shard can either be an instance or a replica set.

A sharded database (or sharded cluster) is composed of the following:

- One or more mongos processes (routers)
- Three config servers
- Two or more shards, each of them can be either a single instance or a replica set

Sharding is done at collection level. Collection's data is organized in chunks (default size 64MB) by means of a shard key (either single field, compound key or hashed key).
A chunk exceeding the chunk size will be splitted and the chunks will be migrated to other shards once the migration threshold has been reached.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "version" : 4,
        "minCompatibleVersion" : 4,
        "currentVersion" : 5,
        "clusterId" : ObjectId("5420168ed2e8674d843ae1ad")
}
  shards:
        {  "_id" : "r0",  "host" : "r0/mongodb1:27117,mongodb2:28117" }
        {  "_id" : "r1",  "host" : "r1/mongodb1:28117,mongodb2:27117" }
```

```
  databases:
        {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
        {  "_id" : "testdb",  "partitioned" : true,  "primary" : "r0" }
                testdb.users
                        shard key: { "country" : 1 }
                        chunks:
                                r1      4
                                r0      4
                        { "country" : { "$minKey" : 1 } } -->> { "country" :
"Bulgaria" } on : r1 Timestamp(2, 0)
                        { "country" : "Bulgaria" } -->> { "country" :
"Ethiopia" } on : r1 Timestamp(3, 0)
                        { "country" : "Ethiopia" } -->> { "country" :
"Ireland" } on : r1 Timestamp(4, 0)
                        { "country" : "Ireland" } -->> { "country" :
"Mauritius" } on : r1 Timestamp(5, 0)
                        { "country" : "Mauritius" } -->> { "country" :
"Pitcairn Island" } on : r0 Timestamp(5, 1)
                        { "country" : "Pitcairn Island" } -->> { "country" :
"Suriname" } on : r0 Timestamp(1, 5)
                        { "country" : "Suriname" } -->> { "country" :
"Western Sahara" } on : r0 Timestamp(1, 6)
                        { "country" : "Western Sahara" } -->> { "country" :
{ "$maxKey" : 1 } } on : r0 Timestamp(1, 7)
        {  "_id" : "test",  "partitioned" : false,  "primary" : "r1" }
        {  "_id" : "db",  "partitioned" : false,  "primary" : "r0" }
```

7.1 *mongos* processes

mongos processes are used from the clients to gain access to the data in the shards. They route requests to the right shard and fetch data back to the clients.

7.2 config servers

config servers are nothing but *mongod* instances which are used to keep metadata information regarding the sharded cluster, such as machine names, information about chunks, etc..

7.3 shards

Shards can be either single instance *mongod* (not recommended in production) or replica sets. Shards contain the actual partitioned data.

8 Monitoring

Regarding monitoring there are four possibilities:

- Operating system commands (*mongostat*, *mongotop*)
- Database commands (*db.currentOp()*, *db.serverStatus()*)
- Third party software
- Other (MMS, customized solutions)

8.1 Operating system commands (*mongostat*, *mongotop*)

mongostat is used to show how many and which operations per second are performed on a given instance.

```
-bash-4.1$ mongostat --port 30000
connected to: 127.0.0.1:30000
insert  query update delete getmore command flushes mapped  vsize    res
faults         locked db idx miss %     qr|qw   ar|aw  netIn netOut  conn
time
    *0     *0     *0     *0       0     1|0       0   160m   601m    62m
0 testdatabase:0.0%          0       0|0     0|0    62b     3k     4
14:45:54
    *0     *0     *0     *0       0     1|0       0   160m   601m    62m
0 testdatabase:0.0%          0       0|0     0|0    62b     3k     4
14:45:55
```

mongotop will show performance metrics per collection, and will only show the top used collections

```
-bash-4.1$ mongotop --port 30000
connected to: 127.0.0.1:30000
                            ns       total        read       write
2015-01-07T13:47:40
          admin.system.indexes         0ms         0ms         0ms
       admin.system.namespaces         0ms         0ms         0ms
            admin.system.roles         0ms         0ms         0ms
            admin.system.users         0ms         0ms         0ms
          admin.system.version         0ms         0ms         0ms
               config.settings         0ms         0ms         0ms
```

8.2 Database commands (*db.currentOp()*, *db.serverStatus()*)

currentOp would show information regarding operations going on in the database, such as running time, query text, locks acquired and so on

```
-bash-4.1$ mongo --port 30000
MongoDB shell version: 2.6.4
connecting to: 127.0.0.1:30000/test
> use testdatabase
switched to db testdatabase
> db.currentOp()
{ "inprog" : [ ] }
```

serverStatus returns server wide information

```
-bash-4.1$ mongo --port 30000
MongoDB shell version: 2.6.4
connecting to: 127.0.0.1:30000/test
> db.serverStatus()
{
        "host" : "mongodb1:30000",
        "version" : "2.6.4",
        "process" : "mongod",
        "pid" : NumberLong(11565),
        "uptime" : 30,
        "uptimeMillis" : NumberLong(29248),
        "uptimeEstimate" : 28,
```

```
        "localTime" : ISODate("2015-02-04T13:25:29.189Z"),
        "asserts" : {
                "regular" : 0,
                "warning" : 0,
                "msg" : 0,
                "user" : 0,
                "rollovers" : 0
        },
        "backgroundFlushing" : {
                "flushes" : 0,
                "total_ms" : 0,
                "average_ms" : 0,
                "last_ms" : 0,
                "last_finished" : ISODate("1970-01-01T00:00:00Z")
        },
        "connections" : {
                "current" : 1,
                "available" : 818,
                "totalCreated" : NumberLong(1)
        },
        "cursors" : {
                "note" : "deprecated, use server status metrics",
                "clientCursors_size" : 0,
                "totalOpen" : 0,
                "pinned" : 0,
                "totalNoTimeout" : 0,
                "timedOut" : 0
        },
        "dur" : {
                "commits" : 30,
                "journaledMB" : 0,
                "writeToDataFilesMB" : 0,
                "compression" : 0,
                "commitsInWriteLock" : 0,
                "earlyCommits" : 0,
                "timeMs" : {
                        "dt" : 3069,
                        "prepLogBuffer" : 0,
                        "writeToJournal" : 0,
                        "writeToDataFiles" : 0,
                        "remapPrivateView" : 0
                }
        },
        "extra_info" : {
                "note" : "fields vary by platform",
                "heap_usage_bytes" : 62560880,
                "page_faults" : 31
        },
        "globalLock" : {
                "totalTime" : NumberLong(29251000),
                "lockTime" : NumberLong(114043),
                "currentQueue" : {
                        "total" : 0,
                        "readers" : 0,
                        "writers" : 0
                },
                "activeClients" : {
                        "total" : 0,
                        "readers" : 0,
                        "writers" : 0
                }
```

```
        },
        "indexCounters" : {
                "accesses" : 2,
                "hits" : 2,
                "misses" : 0,
                "resets" : 0,
                "missRatio" : 0
        },
        "locks" : {
                "." : {
                        "timeLockedMicros" : {
                                "R" : NumberLong(749),
                                "W" : NumberLong(114043)
                        },
                        "timeAcquiringMicros" : {
                                "R" : NumberLong(5429),
                                "W" : NumberLong(362)
                        }
                },
                "admin" : {
                        "timeLockedMicros" : {
                                "r" : NumberLong(669),
                                "w" : NumberLong(0)
                        },
                        "timeAcquiringMicros" : {
                                "r" : NumberLong(4),
                                "w" : NumberLong(0)
                        }
                },
                "local" : {
                        "timeLockedMicros" : {
                                "r" : NumberLong(17983),
                                "w" : NumberLong(43)
                        },
                        "timeAcquiringMicros" : {
                                "r" : NumberLong(81),
                                "w" : NumberLong(2)
                        }
                },
                "testdatabase" : {
                        "timeLockedMicros" : {
                                "r" : NumberLong(46681),
                                "w" : NumberLong(126)
                        },
                        "timeAcquiringMicros" : {
                                "r" : NumberLong(4926),
                                "w" : NumberLong(8)
                        }
                },
                "test" : {
                        "timeLockedMicros" : {
                                "r" : NumberLong(34258),
                                "w" : NumberLong(41)
                        },
                        "timeAcquiringMicros" : {
                                "r" : NumberLong(25),
                                "w" : NumberLong(1)
                        }
                }
        },
        "network" : {
```

```
                "bytesIn" : 299,
                "bytesOut" : 450,
                "numRequests" : 4
        },
        "opcounters" : {
                "insert" : 1,
                "query" : 1,
                "update" : 0,
                "delete" : 0,
                "getmore" : 0,
                "command" : 6
        },
        "opcountersRepl" : {
                "insert" : 0,
                "query" : 0,
                "update" : 0,
                "delete" : 0,
                "getmore" : 0,
                "command" : 0
        },
        "recordStats" : {
                "accessesNotInMemory" : 0,
                "pageFaultExceptionsThrown" : 0,
                "admin" : {
                        "accessesNotInMemory" : 0,
                        "pageFaultExceptionsThrown" : 0
                },
                "local" : {
                        "accessesNotInMemory" : 0,
                        "pageFaultExceptionsThrown" : 0
                },
                "test" : {
                        "accessesNotInMemory" : 0,
                        "pageFaultExceptionsThrown" : 0
                },
                "testdatabase" : {
                        "accessesNotInMemory" : 0,
                        "pageFaultExceptionsThrown" : 0
                }
        },
        "writeBacksQueued" : false,
        "mem" : {
                "bits" : 64,
                "resident" : 61,
                "virtual" : 758,
                "supported" : true,
                "mapped" : 240,
                "mappedWithJournal" : 480
        },
        "metrics" : {
                "cursor" : {
                        "timedOut" : NumberLong(0),
                        "open" : {
                                "noTimeout" : NumberLong(0),
                                "pinned" : NumberLong(0),
                                "total" : NumberLong(0)
                        }
                },
                "document" : {
                        "deleted" : NumberLong(0),
                        "inserted" : NumberLong(1),
```

```
                "returned" : NumberLong(0),
                "updated" : NumberLong(0)
        },
        "getLastError" : {
                "wtime" : {
                        "num" : 0,
                        "totalMillis" : 0
                },
                "wtimeouts" : NumberLong(0)
        },
        "operation" : {
                "fastmod" : NumberLong(0),
                "idhack" : NumberLong(0),
                "scanAndOrder" : NumberLong(0)
        },
        "queryExecutor" : {
                "scanned" : NumberLong(0),
                "scannedObjects" : NumberLong(0)
        },
        "record" : {
                "moves" : NumberLong(0)
        },
        "repl" : {
                "apply" : {
                        "batches" : {
                                "num" : 0,
                                "totalMillis" : 0
                        },
                        "ops" : NumberLong(0)
                },
                "buffer" : {
                        "count" : NumberLong(0),
                        "maxSizeBytes" : 268435456,
                        "sizeBytes" : NumberLong(0)
                },
                "network" : {
                        "bytes" : NumberLong(0),
                        "getmores" : {
                                "num" : 0,
                                "totalMillis" : 0
                        },
                        "ops" : NumberLong(0),
                        "readersCreated" : NumberLong(0)
                },
                "preload" : {
                        "docs" : {
                                "num" : 0,
                                "totalMillis" : 0
                        },
                        "indexes" : {
                                "num" : 0,
                                "totalMillis" : 0
                        }
                }
        },
        "storage" : {
                "freelist" : {
                        "search" : {
                                "bucketExhausted" : NumberLong(0),
                                "requests" : NumberLong(0),
                                "scanned" : NumberLong(0)
```

```
                                        }
                                }
                        },
                        "ttl" : {
                                "deletedDocuments" : NumberLong(0),
                                "passes" : NumberLong(0)
                        }
                },
                "ok" : 1
}
```

8.3 Third-party software

A number of third party monitoring software have plugins for monitoring mongodb:

Tool	Plugin	Description
Ganglia	mongodb-ganglia	Python script to report operations per second, memory usage, btree statistics, master/slave status and current connections.
Ganglia	gmond_python_modules	Parses output from the serverStatus and replSetGetStatus commands.
Motop	None	Realtime monitoring tool for MongoDB servers. Shows current operations ordered by durations every second.
mtop	None	A top like tool.
Munin	mongo-munin	Retrieves server statistics.
Munin	mongomon	Retrieves collection statistics (sizes, index sizes, and each (configured) collection count for one DB).
Munin	munin-plugins	Ubuntu PPA Some additional munin plugins not in the main distribution.
Nagios	nagios-plugin-mongodb	A simple Nagios check script, written in Python.
Zabbix	mikoomi-mongodb	Monitors availability, resource utilization, health, performance and other important metrics.

8.4 Other (MMS, customized solutions)

The MongoDB's MMS interface has Oracle OEM-like monitoring capabilities.

Moreover, by passing the parameter -httpinterface when starting the mongod process a simple http-based monitoring console is started. The port will be the mongod port + 1000.

In addition to this, RESTful services can be used to easily query database internal tables. By means of this, a customized monitoring solution is also feasible.

9 Security

9.1 Authentication

MongoDB comes with four different authentication methods

- MONGODB-CR Authentication: the default, username/password authentication method.
- x.509 Certificate Authentication: MongoDB supports x.509 certificate authentication for use with a secure SSL connection.
- Kerberos Authentication: Only available in Enterprise
- LDAP Proxy Authority Authentication: Only available in Enterprise

9.2 Authorization

Role-Based access control is available but not enabled by default. If enabled (-auth parameter of mongod) it is possible to control privileges on objects by assigning privileges to roles, and then roles to users.

9.3 Auditing

Starting from version 2.6 auditing is available, but not enabled by default. Audit information can be written to the following destinations:

- console
- syslog
- JSON file
- BSON file

The list of events that can be audited is quite comprehensive

Auditing is only available in Enterprise

9.4 Encryption

Encryption is not directly available from MongoDB. The following third party products are recommended according to MongoDB documentation

Name	Description
Linux Unified Key Setup (LUKS)	LUKS is available for most Linux distributions. For configuration explanation, see the LUKS documentation from Red Hat.
IBM Guardium Data Encryption	IBM Guardium Data Encryption provides support for disk-level encryption for Linux and Windows operating systems.
Vormetric Data Security Platform	The Vormetric Data Security Platform provides disk and file-level encryption in addition to application level encryption.
Bitlocker Drive Encryption	Bitlocker Drive Encryption is a feature available on Windows Server 2008 and 2012 that provides disk encryption.

10 Tuning

10.1 Explain plan

Adding *.explain()* at the end of a query will execute the query and print out the explain plan as follows:

```
db.users.find( { country: { $in: [ 'Azerbaijan', 'Brazil' ] } } ).explain()

{
    "clusteredType" : "ParallelSort",
```

```
    "shards" : {
        "r1/mongodb1:28117,mongodb2:27117" : [
            {
                "cursor" : "BtreeCursor country_1",
                "isMultiKey" : false,
                "n" : 8009,
                "nscannedObjects" : 8009,
                "nscanned" : 8010,
                "nscannedObjectsAllPlans" : 8009,
                "nscannedAllPlans" : 8010,
                "scanAndOrder" : false,
                "indexOnly" : false,
                "nYields" : 62,
                "nChunkSkips" : 0,
                "millis" : 15,
                "indexBounds" : {
                    "country" : [
                        [
                            "Azerbaijan",
                            "Azerbaijan"
                        ],
                        [
                            "Brazil",
                            "Brazil"
                        ]
                    ]
                },
                "server" : "mongodb1:28117",
                "filterSet" : false,
                "stats" : {
                    "type" : "KEEP_MUTATIONS",
                    "works" : 8012,
                    "yields" : 62,
                    "unyields" : 62,
                    "invalidates" : 0,
                    "advanced" : 8009,
                    "needTime" : 2,
                    "needFetch" : 0,
                    "isEOF" : 1,
                    "children" : [
                        {
                            "type" : "SHARDING_FILTER",
                            "works" : 8012,
                            "yields" : 62,
                            "unyields" : 62,
                            "invalidates" : 0,
                            "advanced" : 8009,
                            "needTime" : 2,
                            "needFetch" : 0,
                            "isEOF" : 1,
                            "chunkSkips" : 0,
                            "children" : [
                                {
                                    "type" : "FETCH",
                                    "works" : 8011,
                                    "yields" : 62,
                                    "unyields" : 62,
                                    "invalidates" : 0,
                                    "advanced" : 8009,
                                    "needTime" : 2,
                                    "needFetch" : 0,
```

```
                                    "isEOF" : 1,
                                    "alreadyHasObj" : 0,
                                    "forcedFetches" : 0,
                                    "matchTested" : 0,
                                    "children" : [
                                        {
                                            "type" : "IXSCAN",
                                            "works" : 8011,
                                            "yields" : 62,
                                            "unyields" : 62,
                                            "invalidates" : 0,
                                            "advanced" : 8009,
                                            "needTime" : 2,
                                            "needFetch" : 0,
                                            "isEOF" : 1,
                                            "keyPattern" : "{ country: 1.0
}",
                                            "isMultiKey" : 0,
                                            "boundsVerbose" : "field
#0['country']: [\"Azerbaijan\", \"Azerbaijan\"], [\"Brazil\", \"Brazil\"]",
                                            "yieldMovedCursor" : 0,
                                            "dupsTested" : 0,
                                            "dupsDropped" : 0,
                                            "seenInvalidated" : 0,
                                            "matchTested" : 0,
                                            "keysExamined" : 8010,
                                            "children" : []
                                        }
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ]
        },
        "cursor" : "BtreeCursor country_1",
        "n" : 8009,
        "nChunkSkips" : 0,
        "nYields" : 62,
        "nscanned" : 8010,
        "nscannedAllPlans" : 8010,
        "nscannedObjects" : 8009,
        "nscannedObjectsAllPlans" : 8009,
        "millisShardTotal" : 15,
        "millisShardAvg" : 15,
        "numQueries" : 1,
        "numShards" : 1,
        "indexBounds" : {
            "country" : [
                [
                    "Azerbaijan",
                    "Azerbaijan"
                ],
                [
                    "Brazil",
                    "Brazil"
                ]
            ]
        },
```

```
    "millis" : 16
}
```

10.2 Profiling

Profiling can be enabled per-instance or per-database. Three different levels are available:

- 0 - Profiler off: No data is being collected
- 1 - Profiler enabled for slow operations only (per default <=100msec, can be adjusted)
- 2 - Profiler enabled for all operations

Profiling data is written to the capped collection *system.profile*, having a size of 1MB.

Operations, which take longer than "slow" (per default 100msec) are always logged in the instance log file.

10.3 Query optimization

The query plans are chosen on the first query execution and then saved along with the «query pattern» (query text excluding literal values)

They are revised and re-evaluated in the following situations

- Collection receives 1000 writes
- An index is rebuilt (*reIndex()*)
- An index is added or dropped
- Mongod process is restarted
- A query is run with *explain()*

Cached plans are visible using *db.collection.getPlanCache()*.

10.4 Tips and tricks

10.4.1 Use capped collection whenever possible

Capped collections are circular, fixed-size collections that keep documents well ordered, even without the use of an index. This means that capped collections can receive high-speed writes and sequential reads. Example: log files

10.4.2 Check your access paths

Ensure indexes are available where needed. Use *ensureIndex()* to create one if needed.

10.4.3 Limit the results server-side

Instead of retrieving all documents and then discarding the remaining ones, just use *limit()* at the end of a query to limit the retrieved documents server-side: saves network resources

10.4.4 Return only necessary data

Do not retrieve all fields of a document if you do not need them. In your query just select the fields you need: saves network and CPU resources.

10.4.5 Use *$hint* to choose a specific index

db.users.find().hint("age_1") will force usage of index age_1

10.4.6 Use *$inc* operator to increment or decrement values server-side

Avoids round-trips (fetch data, increment, insert data back)

11 Conclusion

The world of NoSQL databases can be interesting also for a relational-database professional. However, make sure the knowledge regarding database administration is spread before NoSQL projects start. I found MongoDB to be a good starting point to get familiar with NoSQL concepts: it is free of charge, well documented and it has interesting free online courses.

12 References

MongoDB Documentation – http://docs.mongodb.org/v2.6
MongoDB University: M102 MongoDB for DBAs – http://university.mongodb.com/courses/m102/about