Why mongo Db?

Open source document oriented database

High performance database highly available.

It is horizontally scalable. We can add a server to increase its capacity instead of vertically scalable which is buying a bigger server.

Where to use?

1. High demanding application (real-time application)
2. Diverse ,mixed set of data
3. Massive concurrency
4. Application which are globally deployed in multiple sites
5. no down time is tolerated
6. Able to grow with user needs should not be stuck with data models should be able to evolve as per the market need.
7. High uncertainty in sizing (not sure how much the system will grow) needs fast scaling.
8. Where ever we need seamless and consistence experience
9. High performance while read, write and update happens at the same time.
10. For Real-time analytics mongo Db is the best where Hadoop is good doing analytics on existing data (batch analysis).
11. Straight forward replication( we can create replicas of mongo db in very less time)
12. Scaling on demand (we can add or remove mongo servers on demand)
13. Location based deployment ( singe cluster can expand over entire globe)
14. Geo spatial queries (location based queries like nearest restaurant , within a circle)
15. High Availability and auto failover(if there is any failure of any datacenter then it will be automatically failover to one of the replicas so there will be no down time )
16. Mongo db can be deployed not only commodity hardware but also virtual hardware like aws.
17. Text indexing
18. Compression
19. Full table scan can disrupt if lot of operations going on.

What MongoDB is not for?

Mongo is not a good choice for:

1. Billing system or general ledger system. Example Oracle RAC
2. Search engine Example Elastic search, SOLR

OLTP and OLAP

OLTP (On-line Transaction Processing) is characterized by a large number of short on-line transactions (INSERT, UPDATE, and DELETE). The main emphasis for OLTP systems is put on very fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second. In OLTP database there is detailed and current data, and schema used to store transactional databases is the entity model (usually 3rdNormalForm).

Example: RDBMS, NoSQL DB (Mongo Db)

- OLAP (On-line Analytical Processing) is characterized by relatively low volume of transactions. Queries are often very complex and involve aggregations. For OLAP systems a response time is an effectiveness measure. OLAP applications are widely used by Data Mining techniques. In OLAP database there is aggregated, historical data, stored in multi-dimensional schemas (usually star schema).

Example: Hadoop

Why mongo Db came in to picture?

1. Usage of mobile devices increased drastically in recent years
2. Social media usage in business application increased a lot.
3. Unstructured data increasing at the rate of 2x as compared to structured data.
4. Classic database (RDBMS) does not know how to scale gracefully in a distributed fashion on commodity hardware.

Bson -> binary json in mongo db

Collection similar to tables

Document similar to record

Commands

db 🡪 shows current database

show dbs or show databsases 🡪 show databases

use dbname 🡪 creates and switches the database

> db

test

> show db

2018-10-29T16:04:27.168+0530 E QUERY [js] Error: don't know how to show [db] :

shellHelper.show@src/mongo/shell/utils.js:1055:11

shellHelper@src/mongo/shell/utils.js:766:15

@(shellhelp2):1:1

> show databases

admin 0.000GB

config 0.000GB

local 0.000GB

test 0.000GB

> switch local

2018-10-29T16:05:04.609+0530 E QUERY [js] SyntaxError: missing ( before switch expression @(shell):1:7

> db

test

> use local

switched to db local

> db

local

> use test

switched to db test

> show databases

admin 0.000GB

config 0.000GB

local 0.000GB

test 0.000GB

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

test 0.000GB

> db.testcollection

test.testcollection

> db.testcollection.insert({"name":"Max"})

WriteResult({ "nInserted" : 1 })

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

test 0.000GB

> use test

switched to db test

> db.testcollection

test.testcollection

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

test 0.000GB

test2 0.000GB

> db.dropDatabase()

{ "dropped" : "test", "ok" : 1 }

**Create collection and insert data**

db.mycollection.insert([{"firstName":"Niranjan","lastName":"Panigrahi"},{"firstName":"Amit","lastName":"Saha"}])

**Get all data from collection**

db.getCollection('testcollection').find({})

show collections

**Create collection**

db.createCollection("mycollection2")

**Drop collection**

db.mycollection2.drop()

**Creates Database**

use school

db.getCollection('mycollection').find().pretty()

db.getCollection('mycollection').findOne()

db.mycollection.findOne()

mongoimport --db dbName --collection collectionName --file fileName.json

mongoimport --db dbName --collection collectionName --file fileName.json –jsonArray

C:\project\softwares\mongo

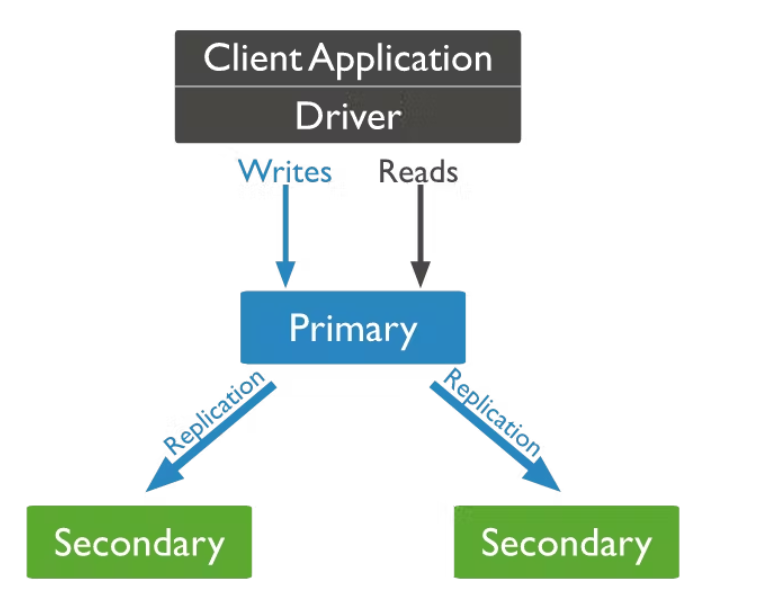
**Import documents**

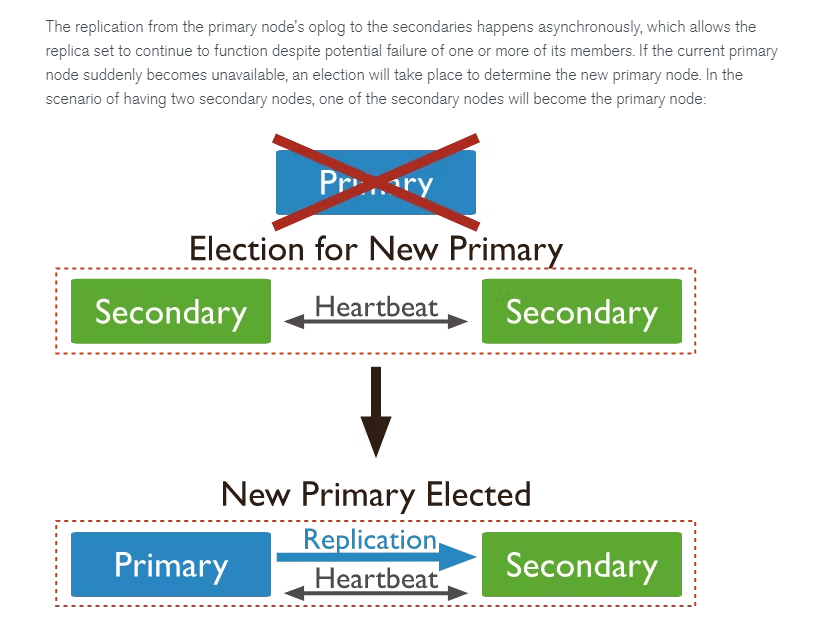
C:\project\softwares\mongo>mongoimport --db test --collection students --file C:\students.json

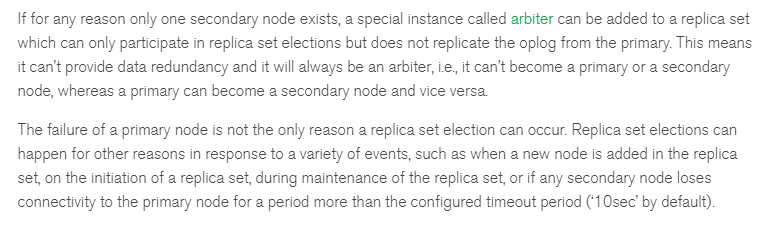
C:\project\softwares\mongo>mongorestore --drop -d test -c tweets C:\tweets.bson

**Mongo Replication**

1. Group of mongod processes that maintain same dataset
2. Redundancy and High availability across multiple server.
3. Increased read capacity
4. If replica set is enabled it ensures that data will not be lost in case primary node fails.







**Replication concept**

1. Write operations go to the primary node.
2. All changes are written recorded in operations log(oplog)
3. Asynchronous replication happens from primary oplog.

**Read write operation**

1. Read operation can optionally directed to specific replica-set.
2. By default, all read operation goes to primary node.

**Read preference:**

Configuration to read from specific secondary can be done in mongo cluster ,

Which is called read preference.

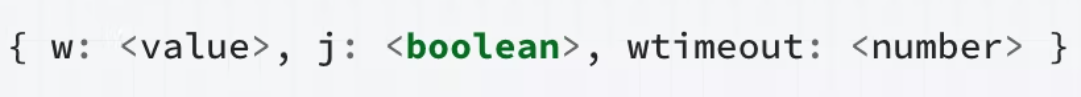
1. if a client application is configured to go directly to secondaries, then the mode parameter in the read preference should be set to [secondary](https://docs.mongodb.com/manual/core/read-preference/#secondary)
2. If there are specific needs for the least network latency irrespective of whether that happens in the primary or any secondary node, then the [nearest](https://docs.mongodb.com/manual/core/read-preference/#nearest) read preference mode should be configured.

potentially stale data comes into play (if the nearest node is a secondary node) due to the nature of [asynchronous replication](https://docs.mongodb.com/manual/replication/#asynchronous-replication) from primary to secondaries.

1. read preference mode can be set to [primary preferred](https://docs.mongodb.com/manual/core/read-preference/#primaryPreferred) or [secondary preferred](https://docs.mongodb.com/manual/core/read-preference/#secondaryPreferred). These two modes also make use of another property called [maxStalenessSeconds](https://docs.mongodb.com/manual/core/read-preference-staleness/#replica-set-read-preference-max-staleness) to determine to which node of the replica set should the read operation be directed.
2. Replica set members can lag behind the [primary](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-primary) due to network congestion, low disk throughput, long-running operations, etc. The read preference  maxStalenessSeconds option lets you specify a maximum replication lag, or "staleness", for reads from [secondaries](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-secondary). When a secondary's estimated staleness exceeds maxStalenessSeconds, the client stops using it for read operations.

**Write concern:**

  level of acknowledgement we desire to have from the cluster upon each write operation is called write concern.



1. MongoDB cluster, you can include additional options to ensure that the write has propagated successfully throughout the cluster. This involves adding a [write concern](https://docs.mongodb.com/manual/reference/write-concern/) property alongside an insert operation.

W: 0/1 by default 1

If W = 0 no write acknowledgement is needed

W= 1 only requires the primary node to acknowledge the write

W = 4 requires **3** secondaries need to signal as well as the primary node to acknowledge

The **j** value corresponds to whether MongoDB has been written on disk in a special area called the [journal](https://docs.mongodb.com/manual/reference/glossary/#term-journal). This is used from MongoDB for recovery purposes in case of a hard shutdown, and it is enabled by default.

**wtimeout** value is the time the command should wait before returning any result. If this is not specified and if for any reason the actual write has any network issues, then the command would block indefinitely, so it is a good practice to set this value. It is measured in milliseconds, and it is only applicable for **w** values greater than 1

Graphical user interface, text, application

Description automatically generated

**Sharding in MongoDB :**