Mongo Introduction

openinx@gmail.com

1. Mongo As A Product

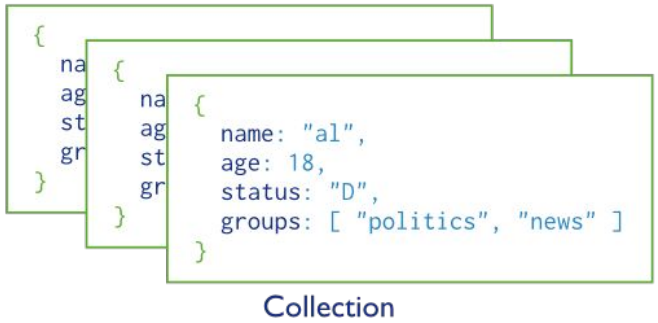
A Distribute , Document NoSQL Database

Mongo User Case

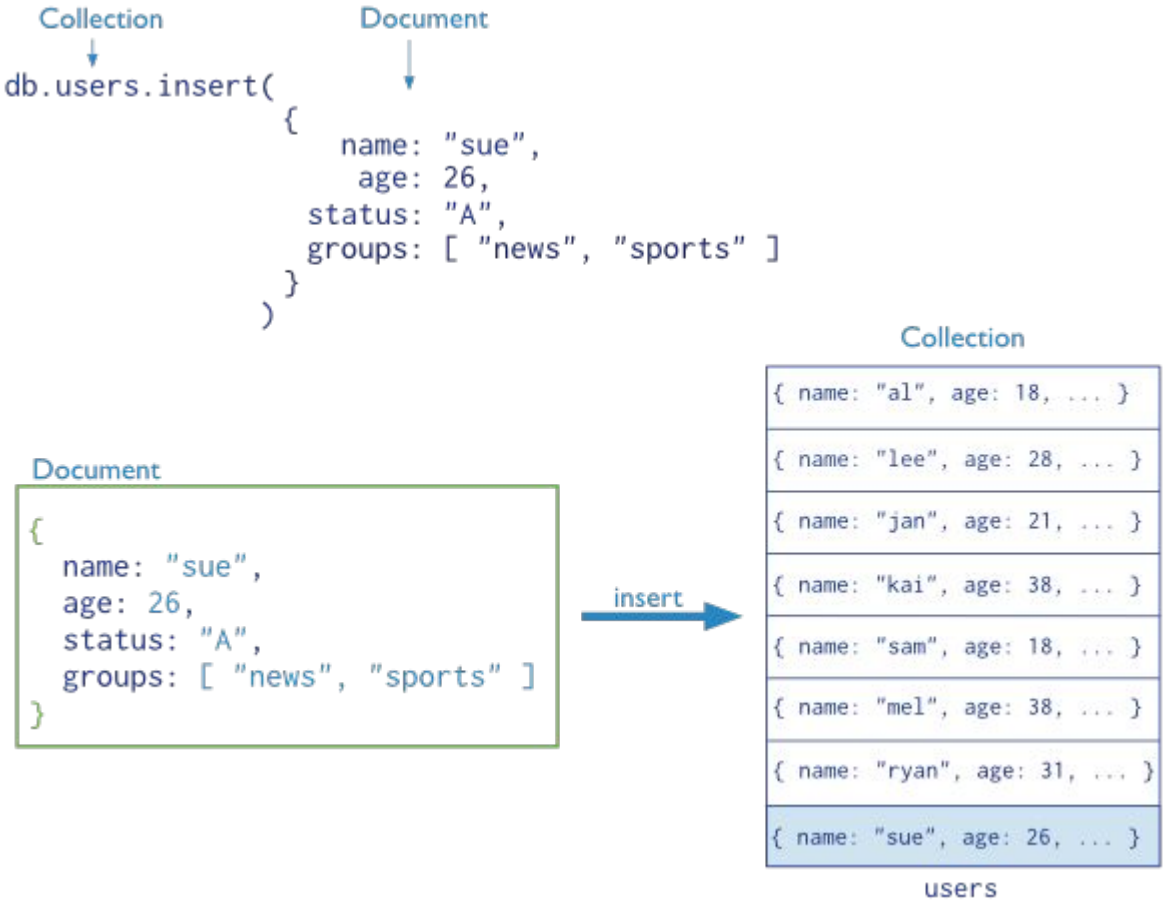
```
{
  name: "sue",                          <-- field: value
  age: 26,                              <-- field: value
  status: "A",                          <-- field: value
  groups: [ "news", "sports" ]          <-- field: value
}
```

Document

Collection



Mongo User Case



Mongo User Case

Mongo Read Operation

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

```
SELECT _id, name, address
FROM   users
WHERE  age > 18
LIMIT  5
```

← projection
← table
← select criteria
← cursor modifier

← MySQL Read Operation

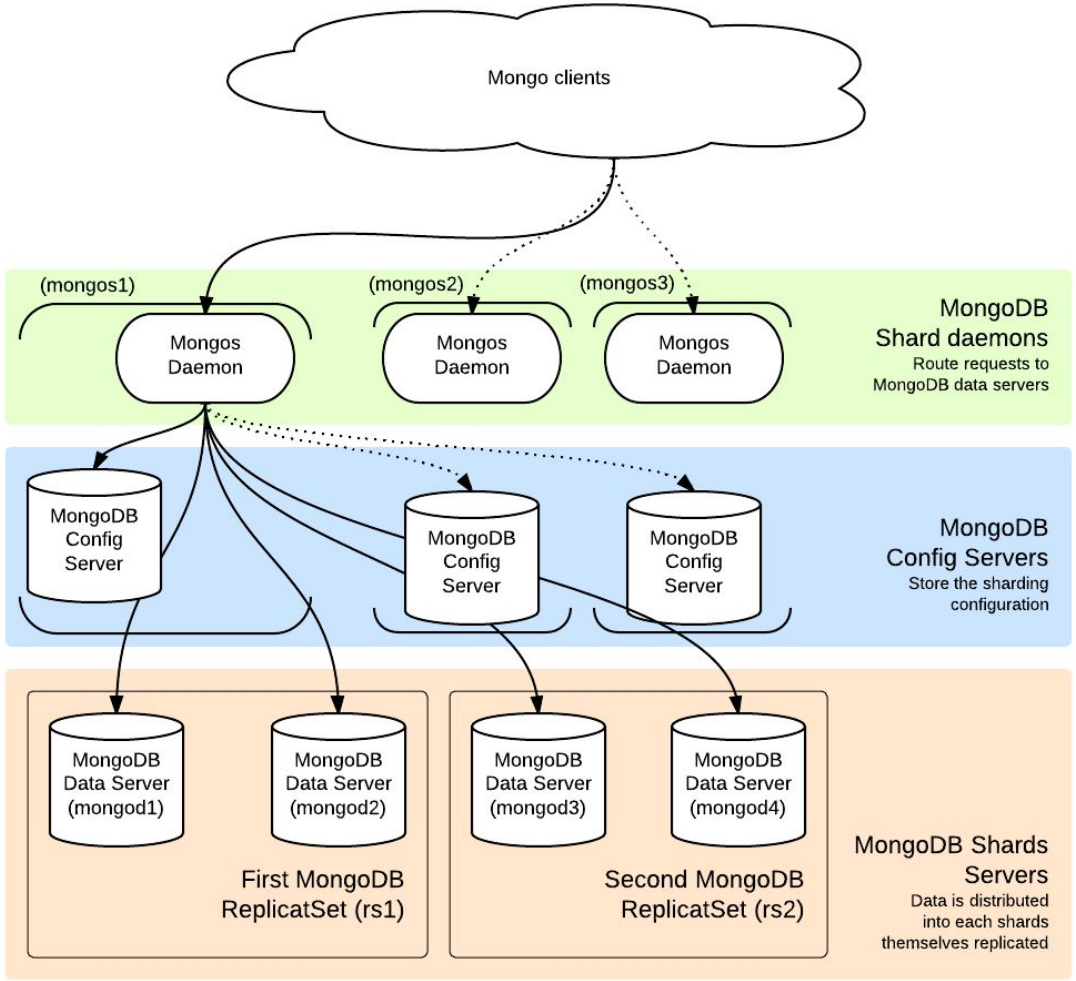
Mongo User Case


```
const MongoClient = require('mongodb').MongoClient;
const assert = require('assert');

// Connection URL
const url = 'mongodb://localhost:27017';

// Database Name
const dbName = 'myproject';

// Use connect method to connect to the Server
MongoClient.connect(url, function(err, client) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  const db = client.db(dbName);

  const col = db.collection('removes');
  // Insert a single document
  col.insertMany([{a:1}, {a:2}, {a:2}], function(err, r) {
    assert.equal(null, err);
    assert.equal(3, r.insertedCount);

    // Remove a single document
    col.deleteOne({a:1}, function(err, r) {
      assert.equal(null, err);
      assert.equal(1, r.deletedCount);

      // Update multiple documents
      col.deleteMany({a:2}, function(err, r) {
        assert.equal(null, err);
        assert.equal(2, r.deletedCount);
        client.close();
      });
    });
  });
});
```

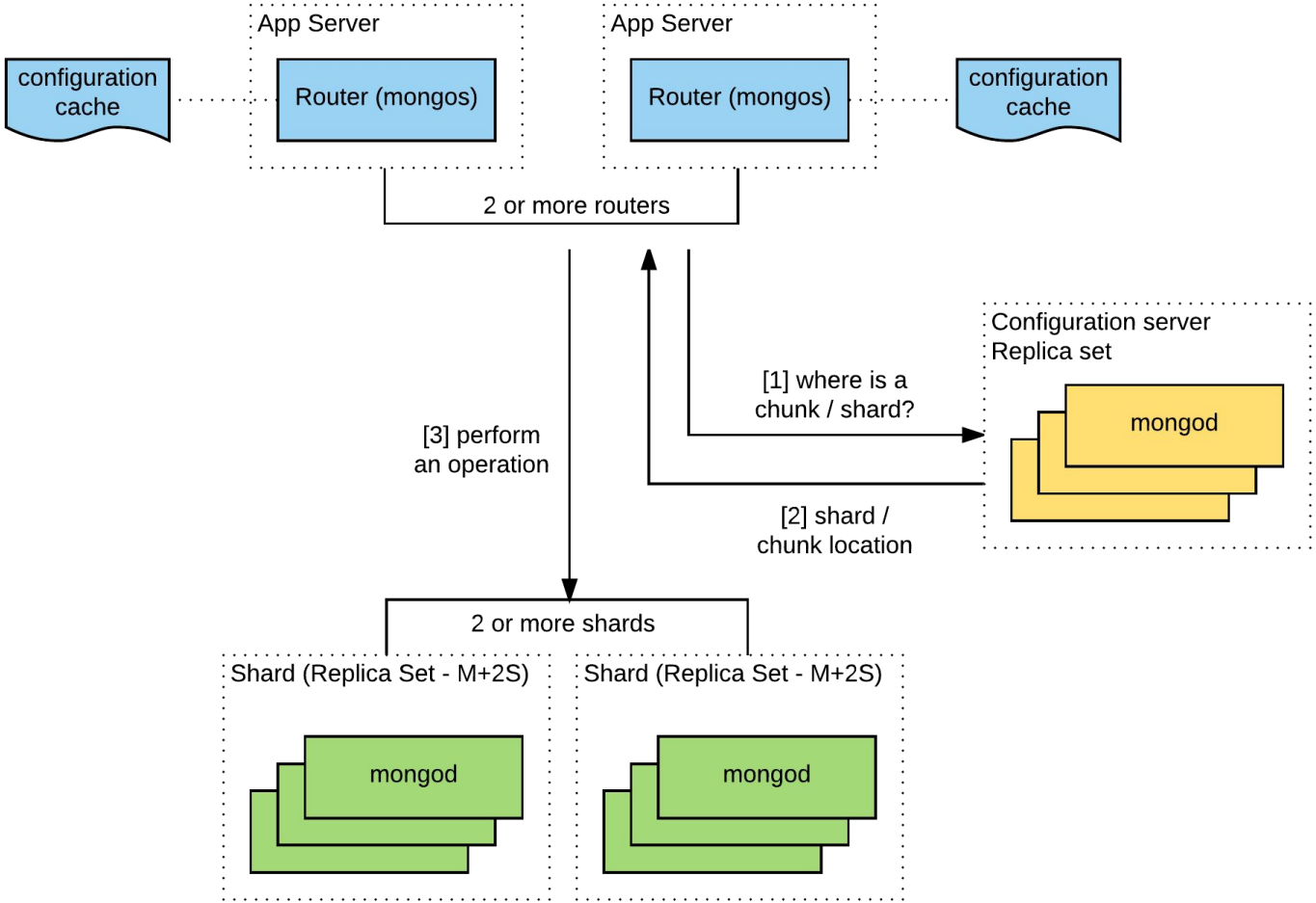
Mongo Shell Gif

2. Mongo Architecture

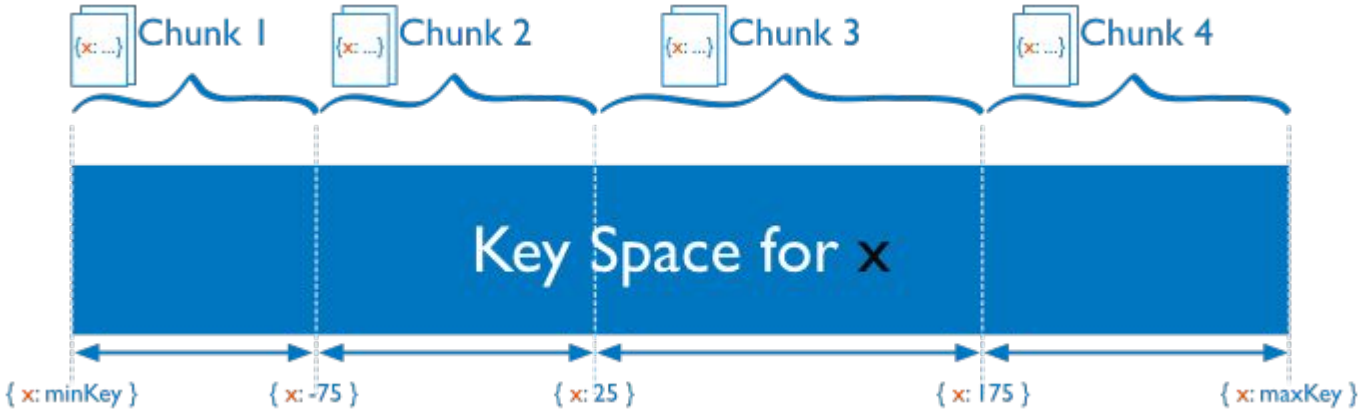
Mongo Architecture



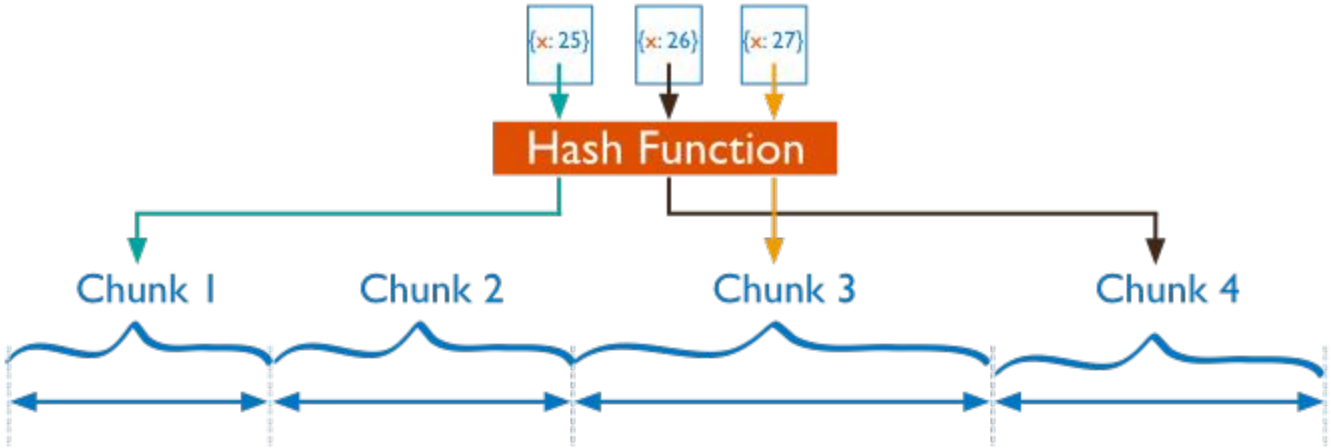
Mongo Architecture



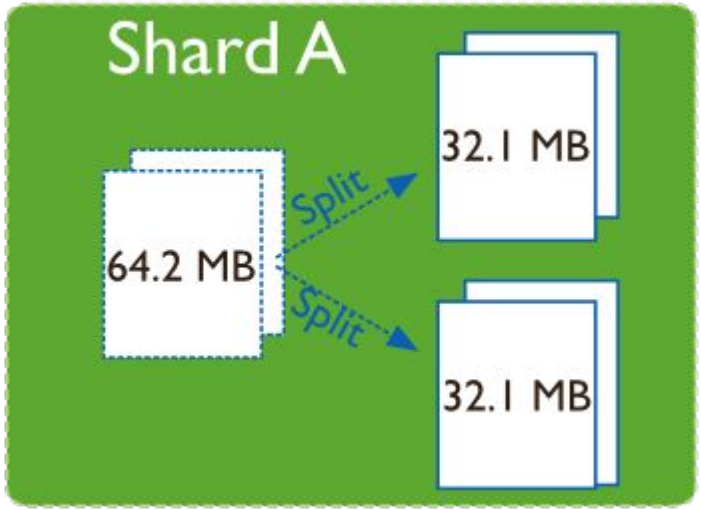
Data Partitioning -- Range Based Sharding



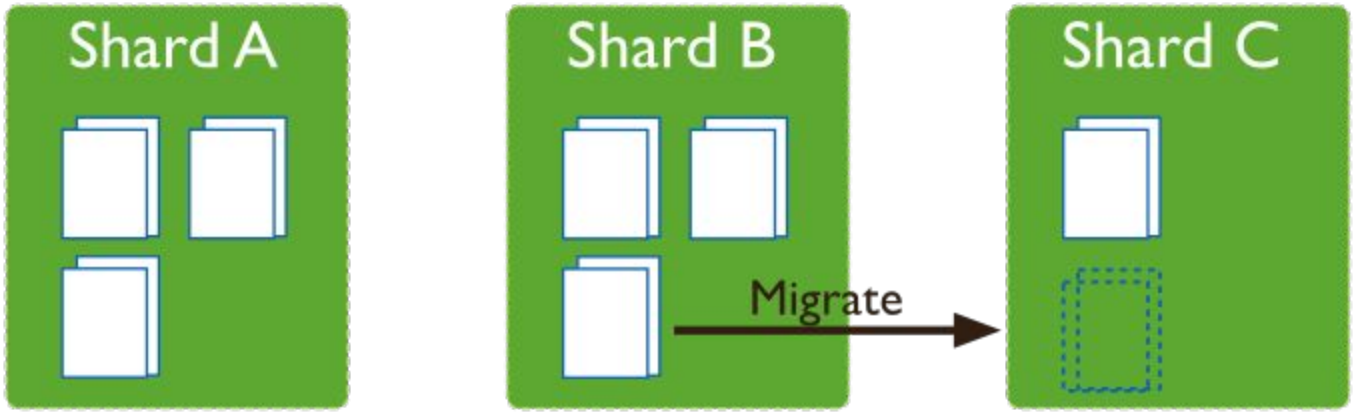
Data Partitioning -- Hash Based Sharding



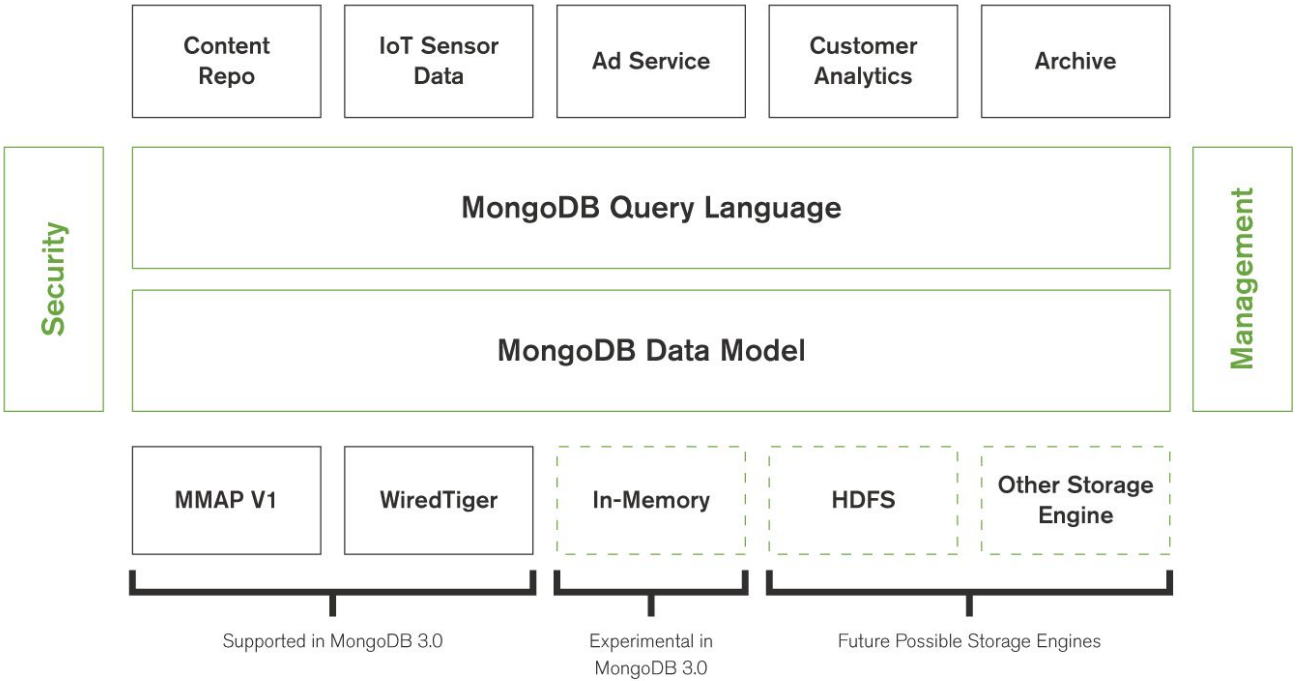
Shard - Splitting



Shard - Splitting



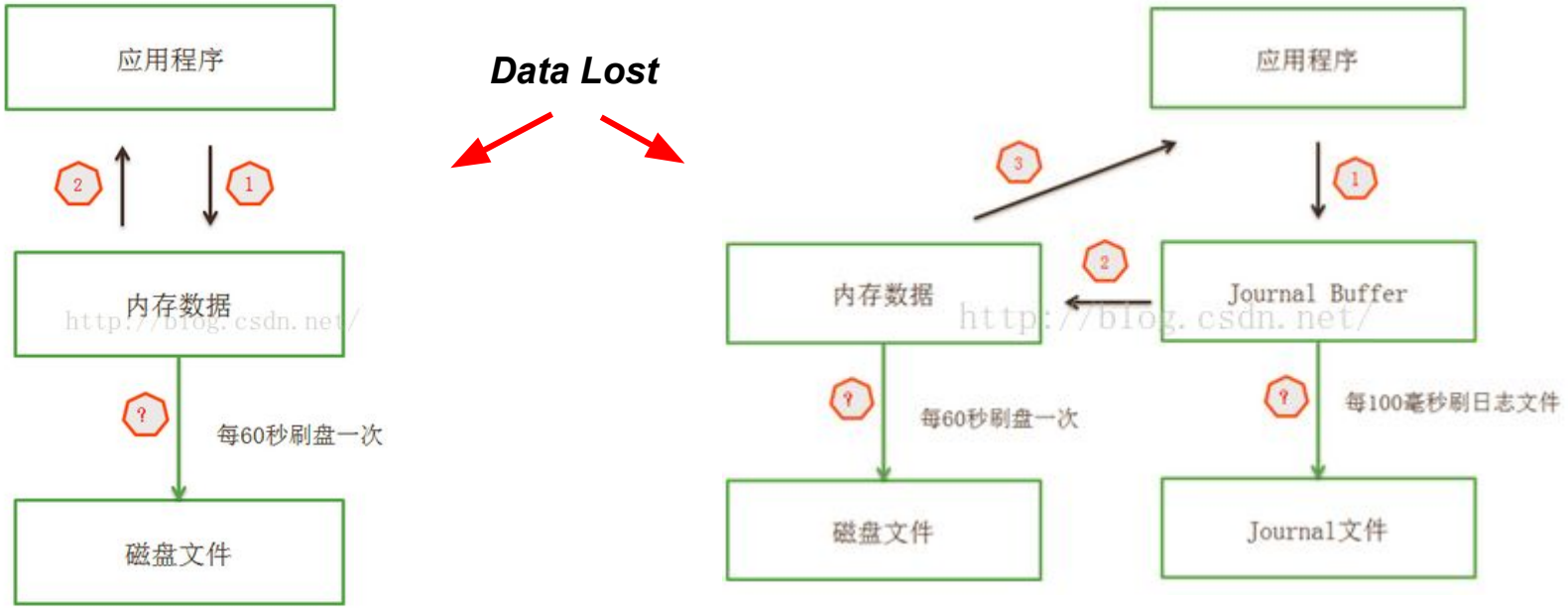
Storage Engine



Storage Engine

	MongoDB WiredTiger	MongoDB MMAPv1
Write Performance	Excellent Document-Level Concurrency Control	Good Collection-Level Concurrency Control
Read Performance	Excellent	Excellent
Compression Support	Yes	No
MongoDB Query Language Support	Yes	Yes
Secondary Index Support	Yes	Yes
Replication Support	Yes	Yes
Sharding Support	Yes	Yes
Ops Manager & MMS Support	Yes All features including deployment, upgrade backup, restore, and monitoring	Yes All features including deployment, upgrade backup, restore, and monitoring
Security Controls	Yes	Yes
Platform Availability	Linux, Windows, Mac OS X	Linux, Windows, Mac OS X, Solaris (x86)

3. Mongo Data Safety

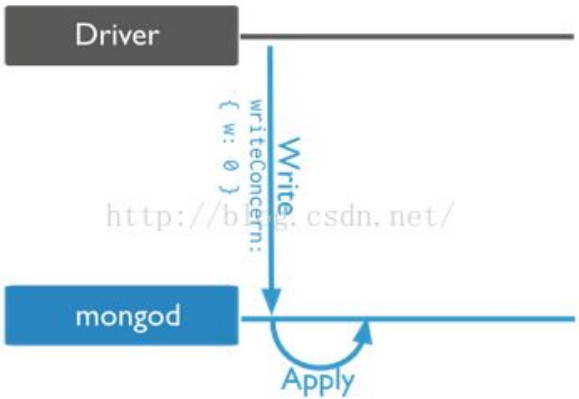


Before 2.0 (Default Journal=OFF)

2.0+ (Default Journal=ON)

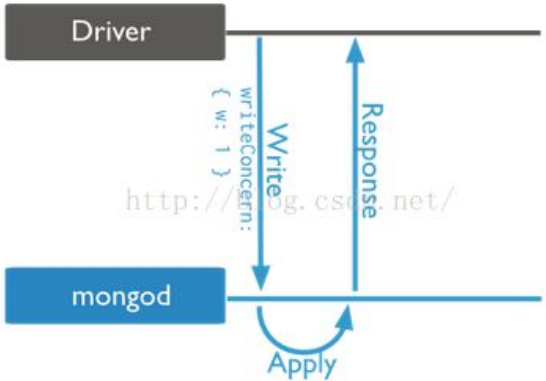
Write Concern

Data Lost



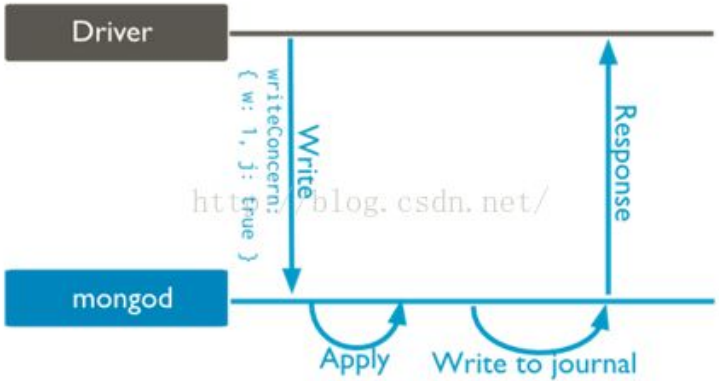
{w: 0} Unacknowledged

Before 2.4



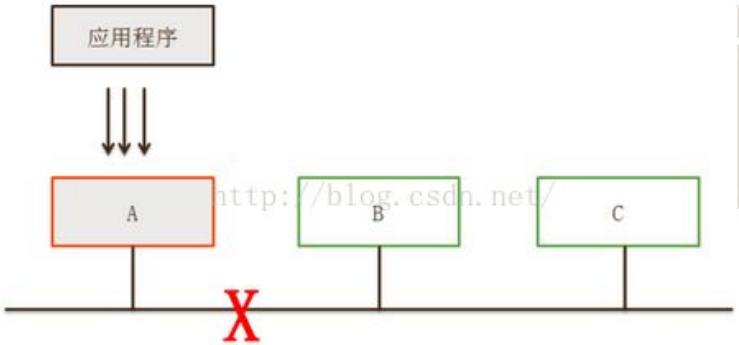
{w: 1} Acknowledged

2.4 +



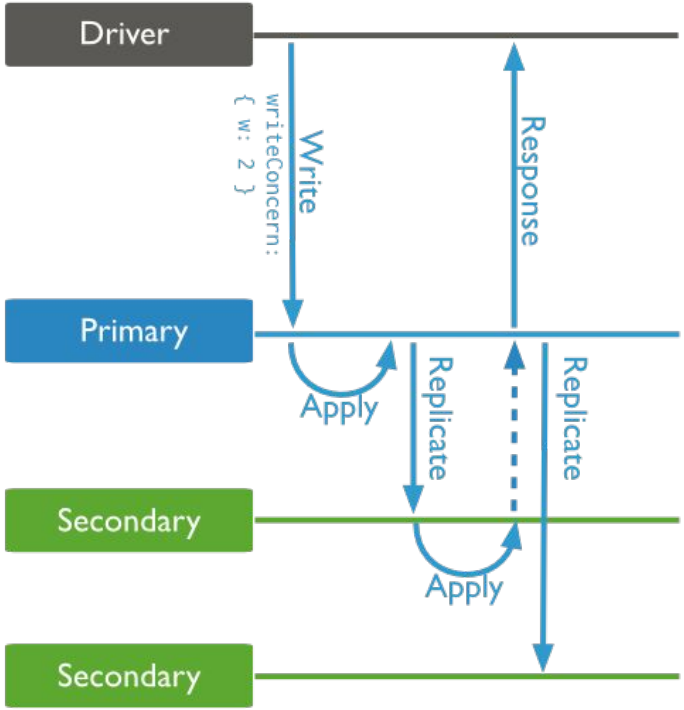
Standalone Data Safe
But Replica Set is not safe

{j:1} Journaled (Batch Sync)



01:00:00 A -> B Network broken
01:00:01 Application write an record x to A.
01:00:02 A -> reject write. A-> Slave
01:00:05 B -> Leader
01:00:08 Network recovered, A re-join replicat set. A will remove the oplog of x.
01:00:08 Application will find x does not exist any more.

Data inconsistent


Driver
writeConcern:
{ w: 2 }
Write
Response
Primary
Apply
Replicate
Replicate
Secondary
Apply
Secondary

{w: "majority"}

4. Mongo VS HBase

Mongo VS HBase

	Data Model	Data Type	Query Model	Write Perf	Hadoop Ecosytem	LBS
MongoDB	Document	int/string etc..	JSON Query	Medium	Non Apache Project	Native supported
HBase	Wide Column	byte[]	Scan + Filter	Very Faster	Apache Top Project	GeoHash
Score	?	?	M > H	H > M	H > M	M > H

MongDB And HBase Compared

Mongo VS HBase

	Spark	MapReduce	GC	Throughput	Consistence	…..
MongoDB	-	-	-	-	Strong OR Eventual	…..
HBase	-	-	-	-	Strong consistent	…..
Score	H+	H+	M+	H+	?	…..

MongDB And HBase Compared

4. Mongo Case

Mongo Case

MongoDB 特性	优势
事务支持	MongoDB 目前只支持单文档事务，需要复杂事务支持的场景暂时不适合
灵活的文档模型	JSON 格式存储最接近真实对象模型，对开发者友好，方便快速开发迭代
高可用复制集	满足数据高可靠、服务高可用的需求，运维简单，故障自动切换
可扩展分片集群	海量数据存储，服务能力水平扩展
高性能	mmapv1、wiredtiger、mongorocks（rocksdb）、in-memory 等多引擎支持满足各种场景需求
强大的索引支持	地理位置索引可用于构建 各种 O2O 应用、文本索引解决搜索的需求、TTL索引解决历史数据自动过期的需求
Gridfs	解决文件存储的需求
aggregation & mapreduce	解决数据分析场景需求，用户可以自己写查询语句或脚本，将请求都分发到 MongoDB 上完成

Mongo Case

应用特征	**Yes / No**
应用不需要事务及复杂 join 支持	必须 Yes
新应用，需求会变，数据模型无法确定，想快速迭代开发	?
应用需要2000-3000以上的读写QPS（更高也可以）	?
应用需要TB甚至 PB 级别数据存储	?
应用发展迅速，需要能快速水平扩展	?
应用要求存储的数据不丢失	?
应用需要99.999%高可用	?
应用需要大量的地理位置查询、文本查询	?

如果上述有1个 Yes，可以考虑 MongoDB，2个及以上的 Yes，选择MongoDB绝不会后悔。

Mongo Case

- 游戏场景，使用 MongoDB 存储游戏用户信息，用户的装备、积分等直接以内嵌文档的形式存储，方便查询、更新
- 物流场景，使用 MongoDB 存储订单信息，订单状态在运送过程中会不断更新，以 MongoDB 内嵌数组的形式来存储，一次查询就能将订单所有的变更读取出来。
- 社交场景，使用 MongoDB 存储存储用户信息，以及用户发表的朋友圈信息，通过地理位置索引实现附近的人、地点等功能
- 物联网场景，使用 MongoDB 存储所有接入的智能设备信息，以及设备汇报的日志信息，并对这些信息进行多维度的分析
- 视频直播，使用 MongoDB 存储用户信息、礼物信息等
- .....