ODF 2017 开源数据库论坛(北京)
OPEN-SOURCE DATABASE FORUM(BEIJING)

开源数据库正在改变世界

2017年8月24日-25日　北京－京仪大酒店

MongoDB Key Features

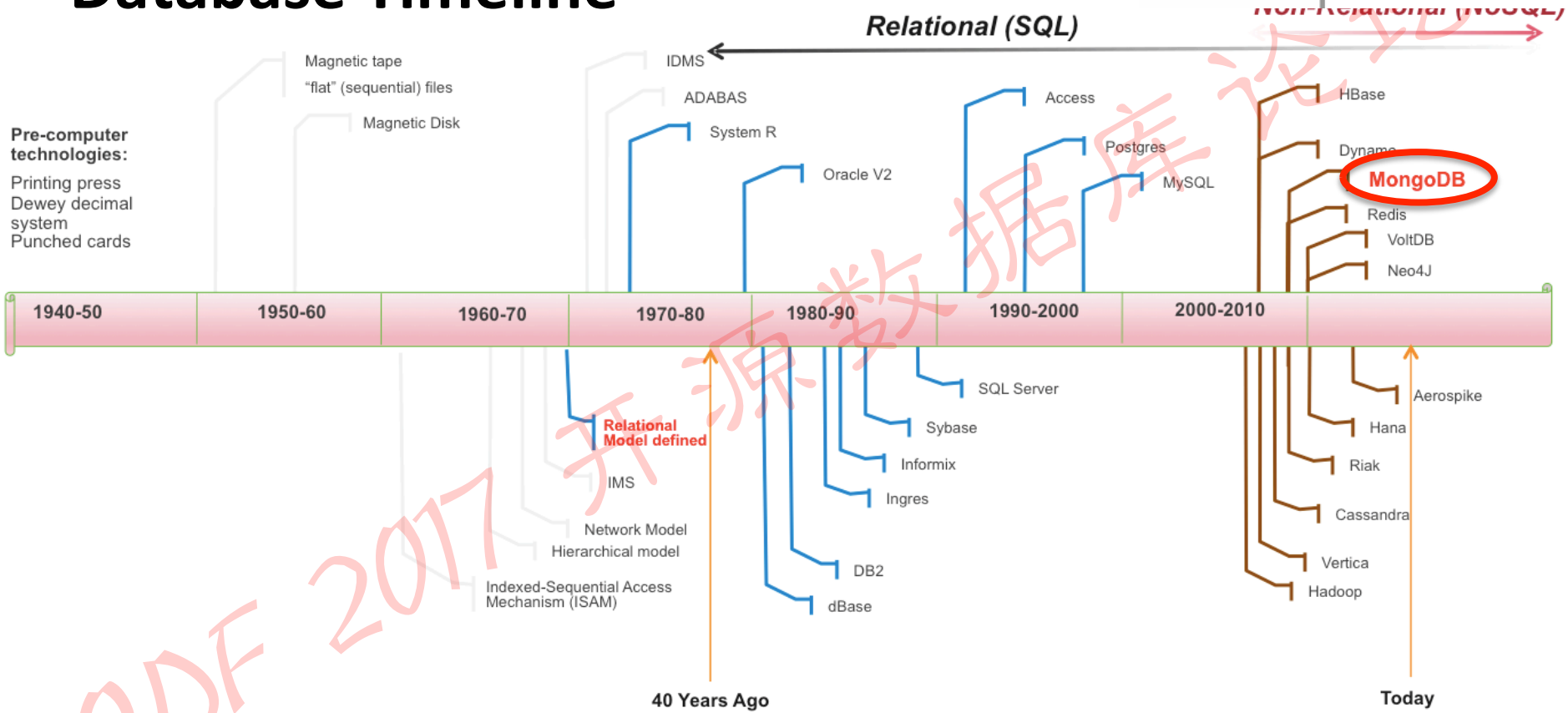
ZhangYoudong@Alibaba Cloud

2017.08.25 Beijing

About me

- 5+ years development
- Focus on distributed storage & database
- MongoDB as a Service at Alibaba Cloud
- Co-founder of mongoing.com Community

Database Timeline

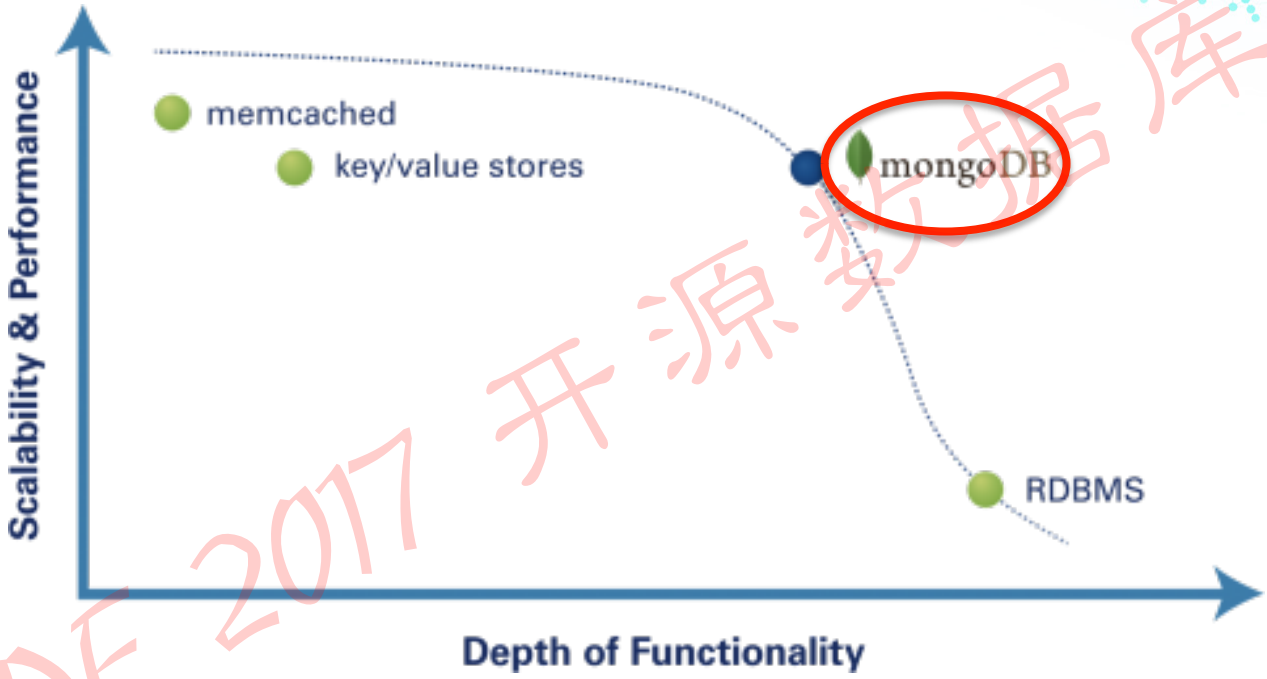


Database popularity

331 systems in ranking, August 2017

Rank			DBMS	Database Model	Score		
Aug 2017	Jul 2017	Aug 2016			Aug 2017	Jul 2017	Aug 2016
1.	1.	1.	Oracle	Relational DBMS	1367.88	-7.00	-59.85
2.	2.	2.	MySQL	Relational DBMS	1340.30	-8.81	-16.73
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1225.47	-0.52	+20.43
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.76	+0.32	+54.51
5.	5.	↓ 4.	MongoDB	Document store	330.50	-2.27	+12.01
6.	6.	6.	DB2	Relational DBMS	197.47	+6.22	+11.58
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	127.03	+0.90	+2.98
8.	8.	↓ 7.	Cassandra	Wide column store	126.72	+2.60	-3.52
9.	9.	↑ 10.	Redis	Key-value store	121.90	+0.38	+14.57
10.	10.	↑ 11.	Elasticsearch	Search engine	117.65	+1.67	+25.16

Where MongoDB Stands?



Key Features

- JSON Data Model with Dynamic Schema
- Rich, Document-Based Queries
- Flexible, Full Index Support
- GEO, text, TTL Index Support
- Built-In Replication and High Availability
- Auto-Sharding for Horizontal Scalability
- Aggregation Framework and Map/Reduce
- GridFS for Large File Storage

Key Features: MongoDB is a ???

- **Document based database**
  - JSON document
  - Flexible index & query support
- High reliablity, high availablity database
  - Replica sets
  - Ease of management
- Scalable database
  - Sharded cluster
  - Auto load balance

Documents Are Core

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

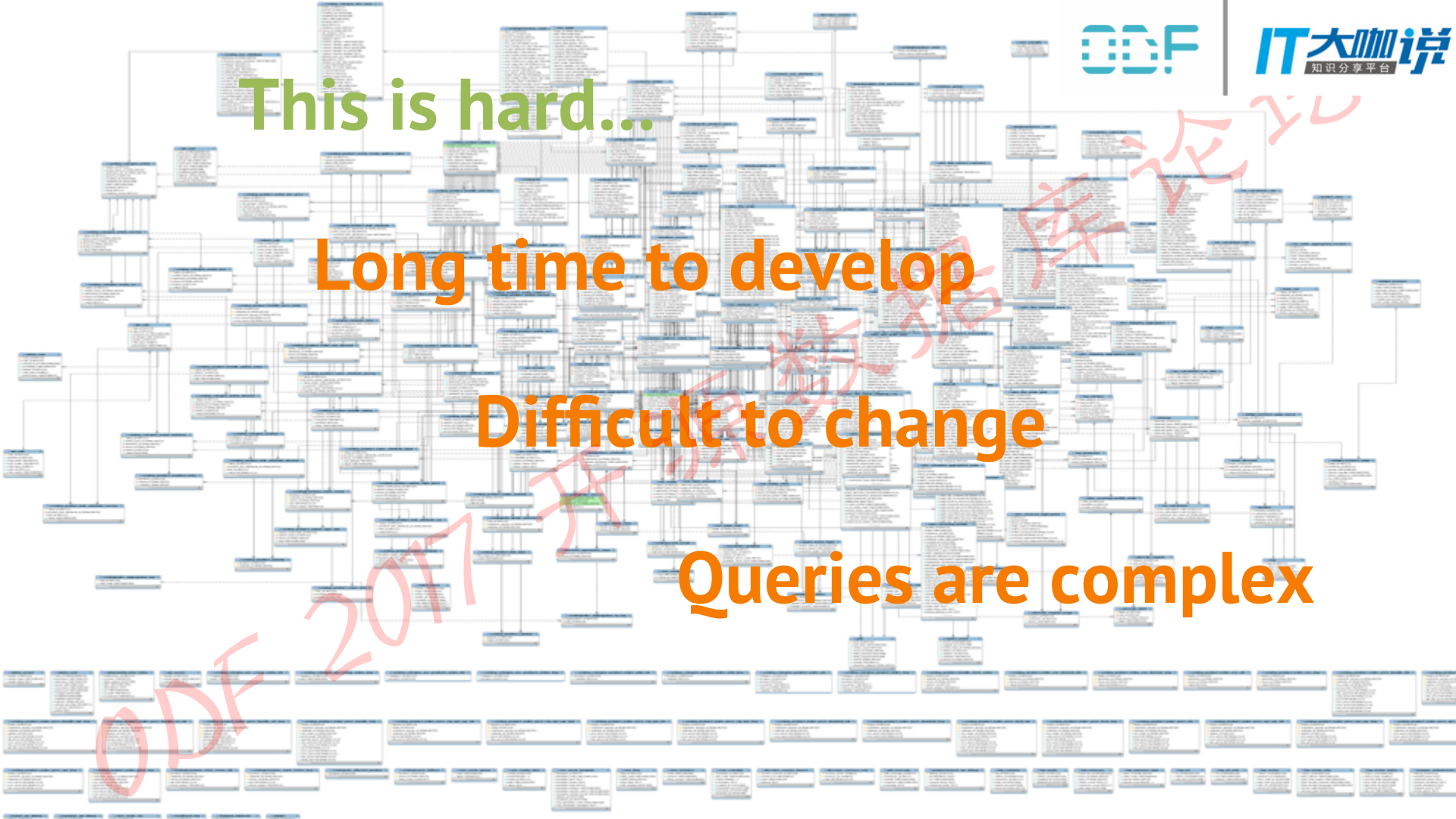
no relation

Car:

Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

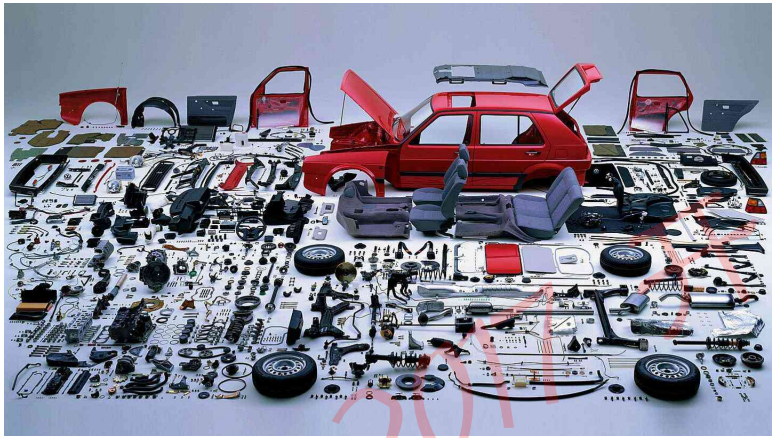
MongoDB

```
{
  first_name: "Paul",
  surname: "Miller",
  city: "London",
  location: [45.123,47.232],
  cars: [
    { model: "Bentley",
      year: 1973,
      value: 100000, … },
    { model: "Rolls Royce",
      year: 1965,
      value: 330000, … }
  ]
}
```


This is hard...
Long time to develop
Difficult to change
Queries are complex

Documents Are Core

Relational



MongoDB

Documents Advantages

Agility and flexibility

Dynamic model supports business change

Rapidly iterate to meet new requirements

Intuitive, natural data representation

Eliminates ORM layer

Developers are more productive

Reduces the need for joins, disk seeks

Gather related data together

Programming is simpler

```
{
  first_name: "Paul",
  surname: "Miller",
  city: "London",
  location: [45.123,47.232],
  cars: [
    { model: "Bentley",
      year: 1973,
      value: 100000, … },
    { model: "Rolls Royce",
      year: 1965,
      value: 330000, … }
  ]
}
```

```
{
  _id: ObjectId("5829330e159d389a461ea86d"),
  uid: NumberLong(11994734),
  name: ”Tony Zhang",
  gender: 2,
  inviteCode: "fb62c7efe6c”
  energy: 30,
  shields: 0,
  crowns: 11,
  buildings: [{…}, {…}, {…}],
  jigsaw: { … },
  friends: [{…}, {…}, {…}],
  lastReadMsgAt: NumberLong(1479101785),
  lastReadNoticeAt: NumberLong(1479120954),
  lastRecoveredAt: NumberLong(1479119453),,
  ......
}
```

Use case: Game

```
{
  _id: ObjectId("5829330e159d389a461ea86d"),
  orderId: NumberLong(3779827654),
  orderState: [
    {
        position: "Hangzhou Transfer Center",
        to: "Yunqi Town",
        date: ISODate("2017-08-11T08:49:47Z"),
        processor: "Tony Zhang",
    },
    {
        position: "Yunqi Town",
        date: ISODate("2017-08-12T10:49:47Z"),
        processor: "Jack Li",
        notes:  "Pick by self"},
    ......
  ]
}
```

Use case: Logistics

```
{
   _id: ObjectId(”2221b13ac1033f9a6b09582d"),
   cityId: 3,
   riderId: 1330234,
   point: {
       type: "Point",
       coordinates: [ 121.687679, 31.261832 ]
    },
    lastUpdateTime: ISODate("2017-03-03T13:04:05.331Z")
}

{
    _id: ObjectId(“df467fa16e33d2365f94a"),
    cityId: 3,
    shopId: 1533,
    point: {
         type: “Point”,
         coordinates: [ 121.44596, 31.322233 ]
     },
     shopTitle: “HubeiCai”,
     shopAddr: XXXXXX",
     orderTimeInLatest30Days:  ISODate("2016-03-07T12:41:54.814Z"),
     platformShopId”: 4,
}
```

Use case: O2O

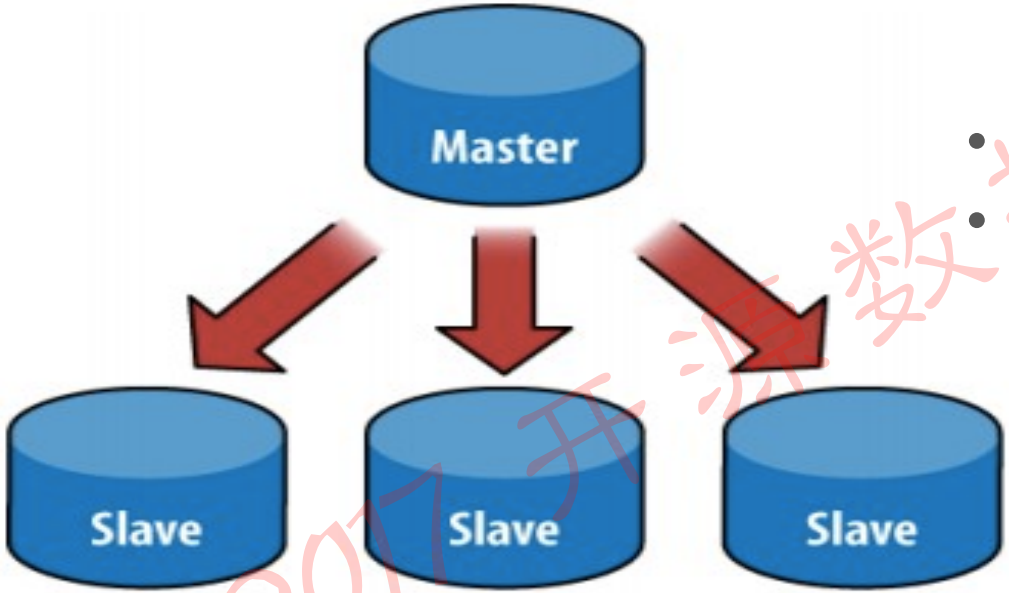
```
{
  deviceId: NumberLong(3779827654)
  date: ISODate("2017-08-11T08:45:00Z"),
  stat: [
    {
       temperature: 33
       humidity: 38
       pm25: 125
    },
    {
       temperature: 34
       humidity: 38
       pm25: 125
    },
    ......
  ]
}
```

Use case: IoT

Key Features: MongoDB is a ???

- Document based database
  - JSON document
  - Flexible index & query support
- **High reliablity, high availablity database**
  - Replica sets
  - Ease of management
- Scalable database
  - Sharded cluster
  - Auto load balance

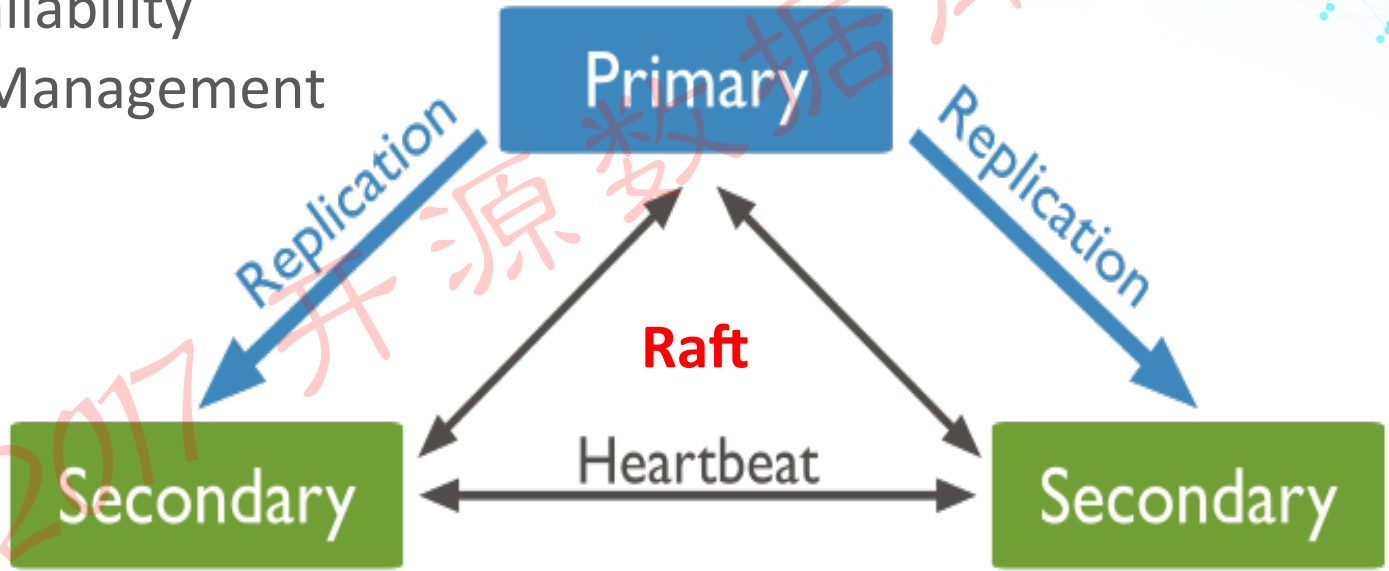
Traditional Master-slave Architectur



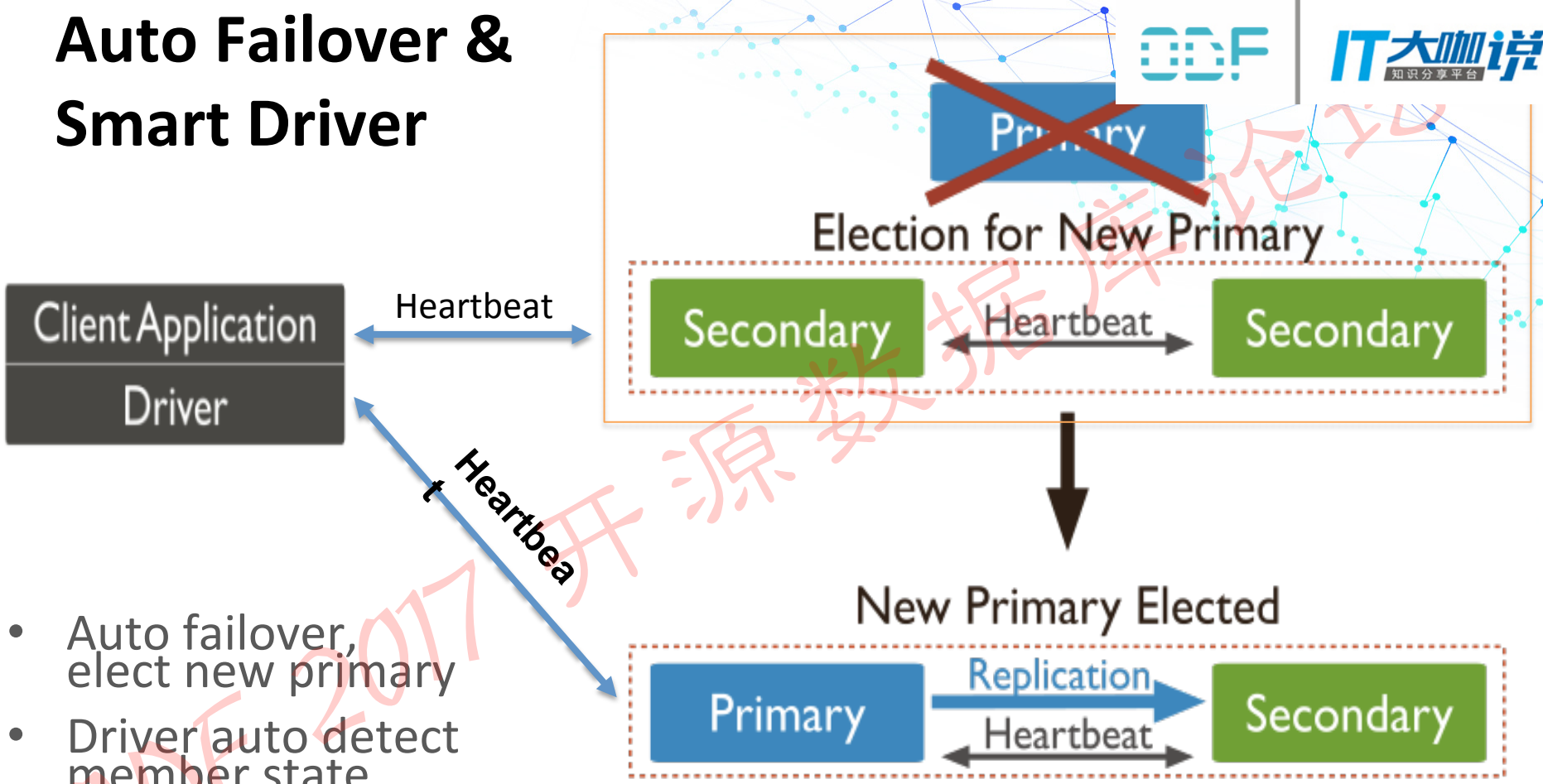
- External failover
- Split-brain problem

Replica Sets

- High Reliablity
- High Availability
- Ease of Management

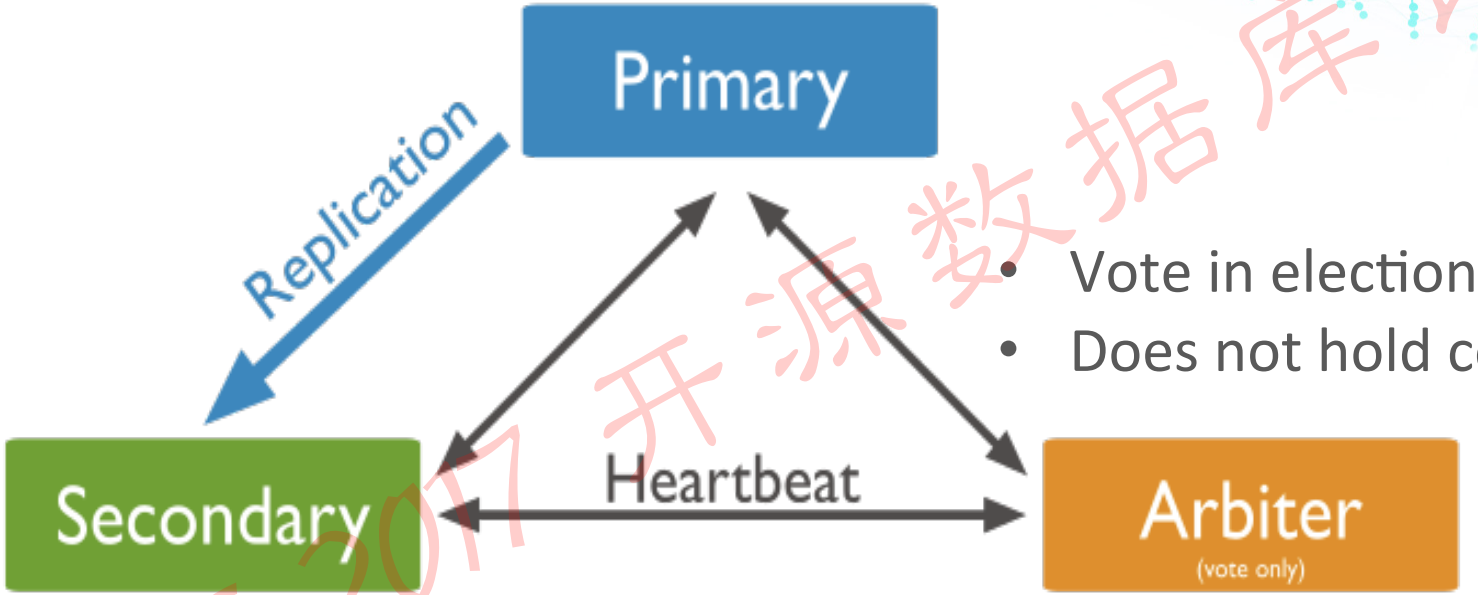


Auto Failover & Smart Driver



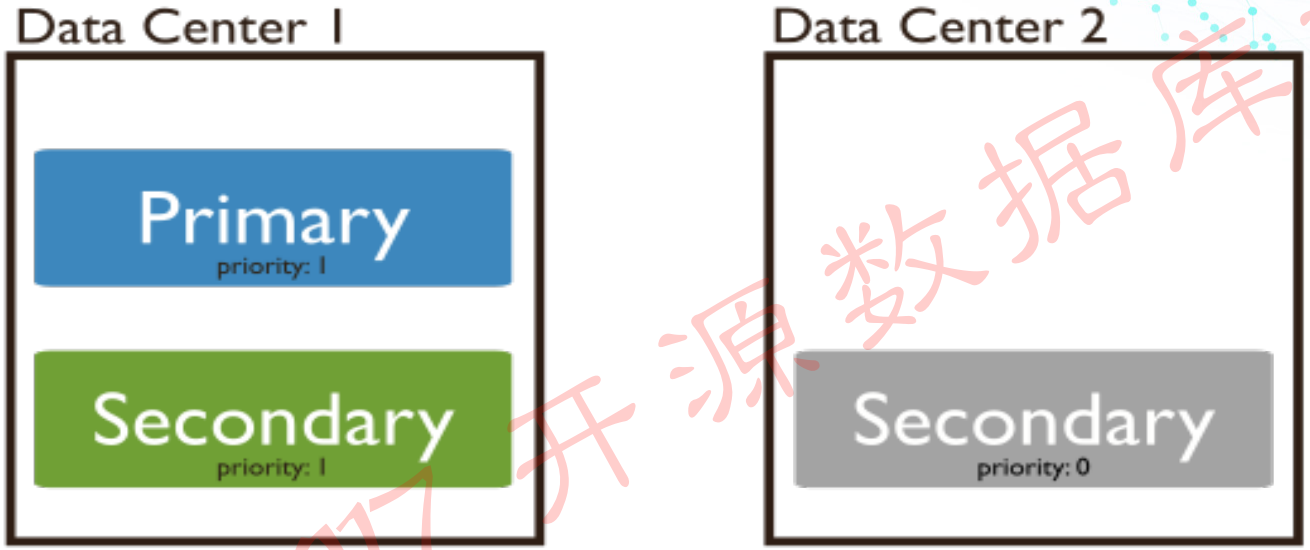
- Auto failover, elect new primary
- Driver auto detect member state

Arbiter (Vote Only Member)



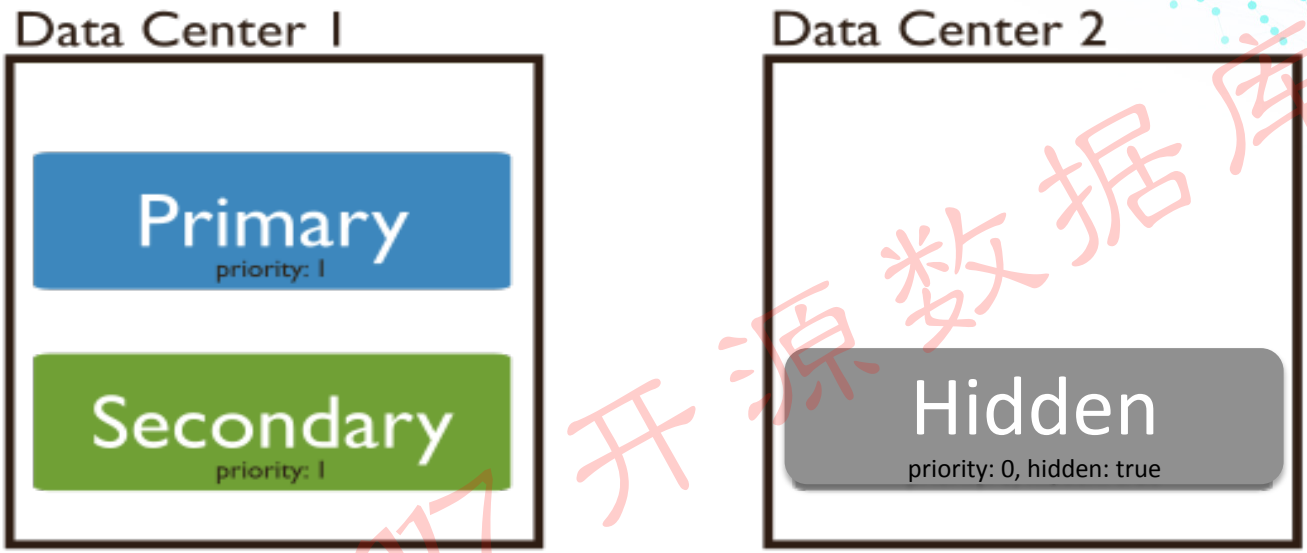
- Vote in election
- Does not hold copy of data

Priority0 Member



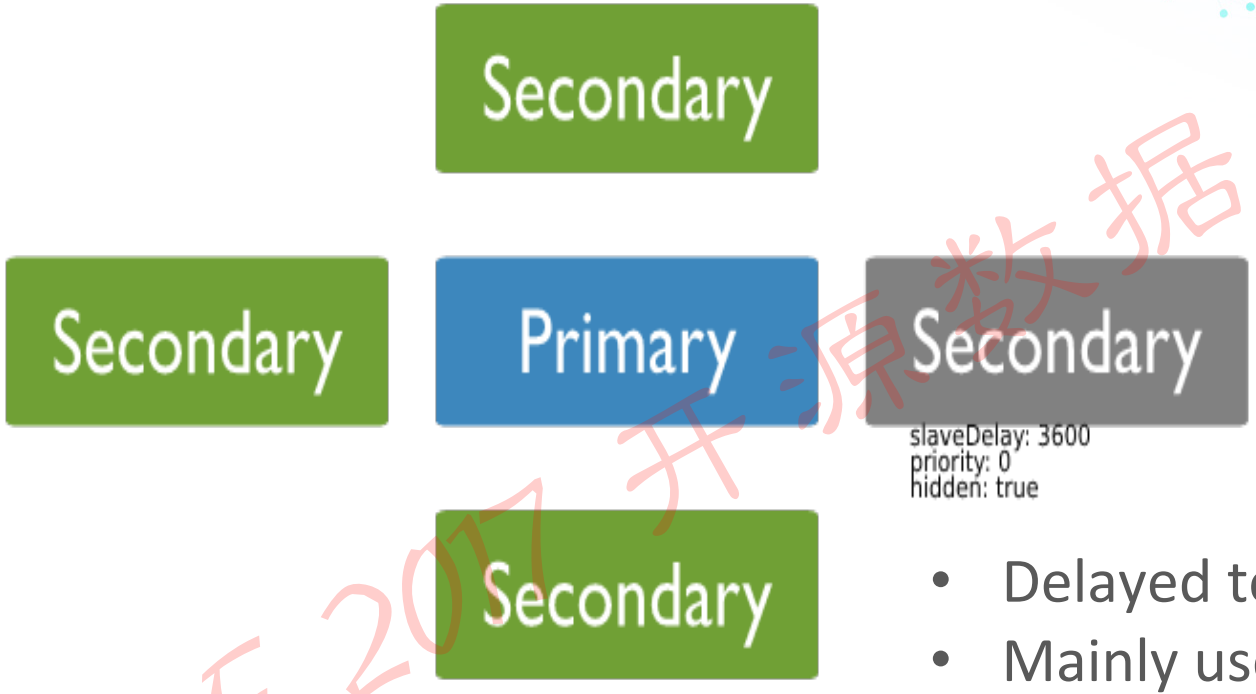
- Never becomes primary
- Not visible to application

Hidden Member



- Never becomes primary
- Visible to application

Delayed Member



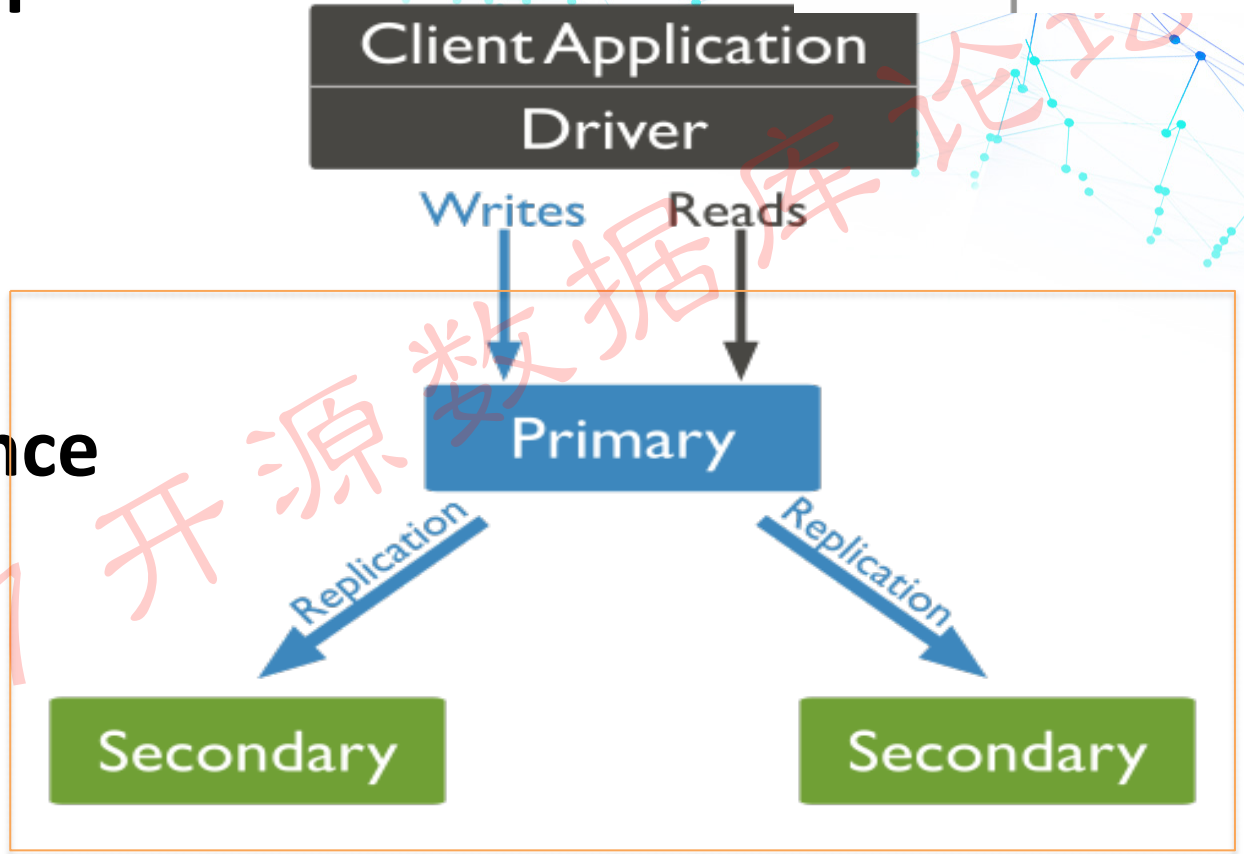
- Delayed to sync oplog
- Mainly used for backup

Write Concern

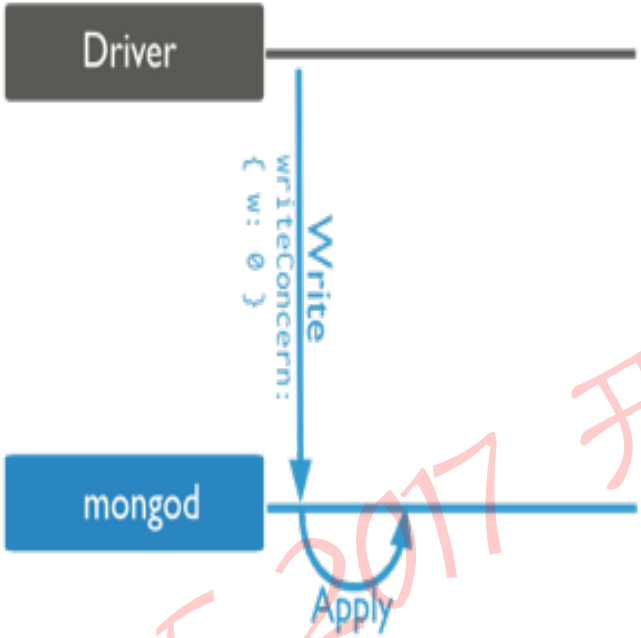
- {w: 1}
- {w: 1, j: true}
- {w: “majority”}
- {w: “tags”}

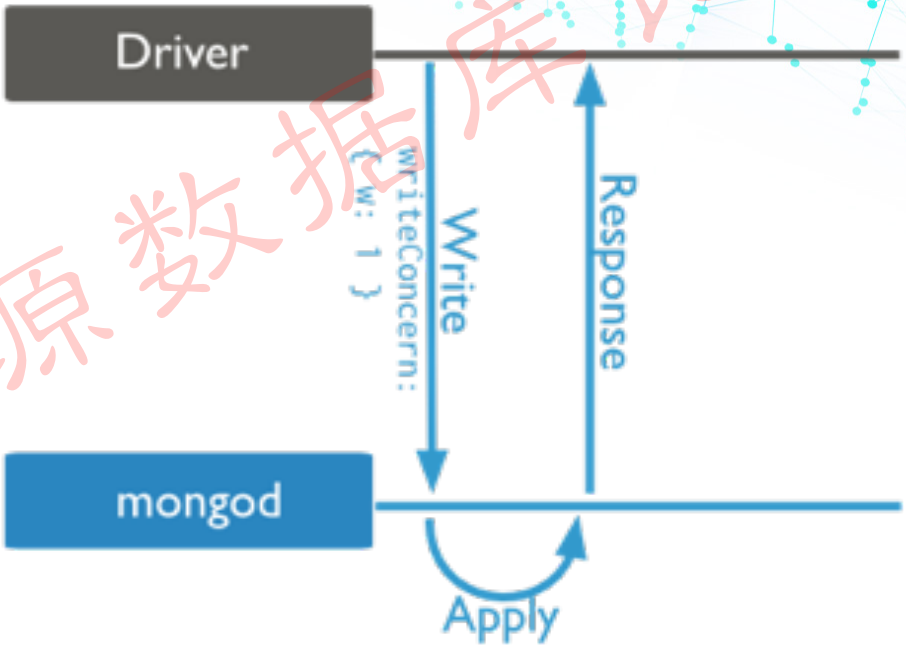
Read Preference

- primary
- secondary
- nearest
- tags

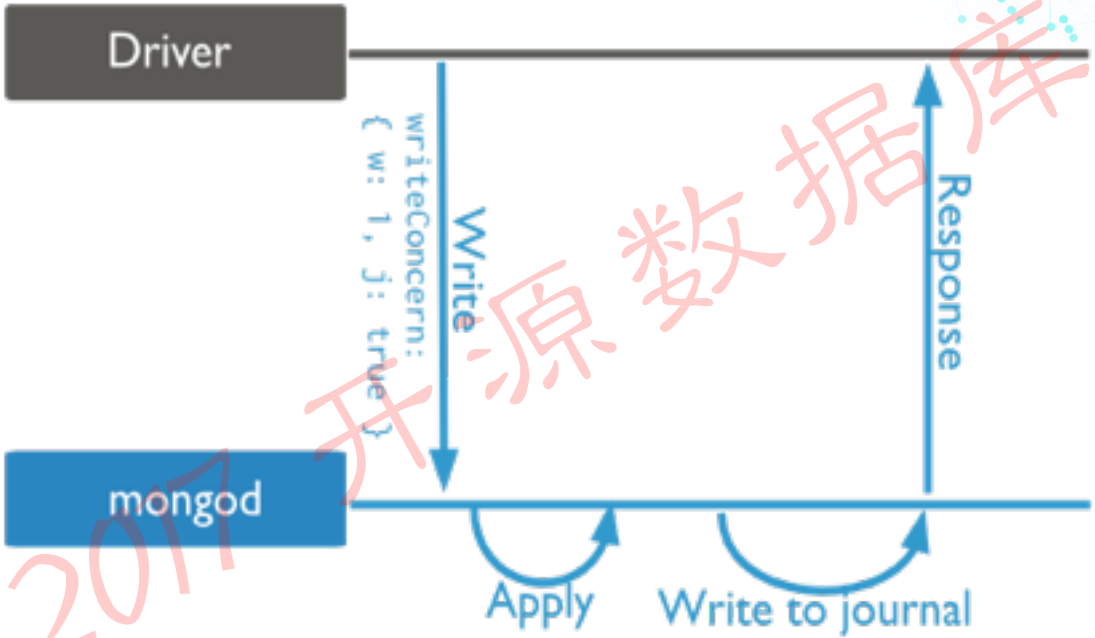


Write Concern {w: 0} vs {w: 1}

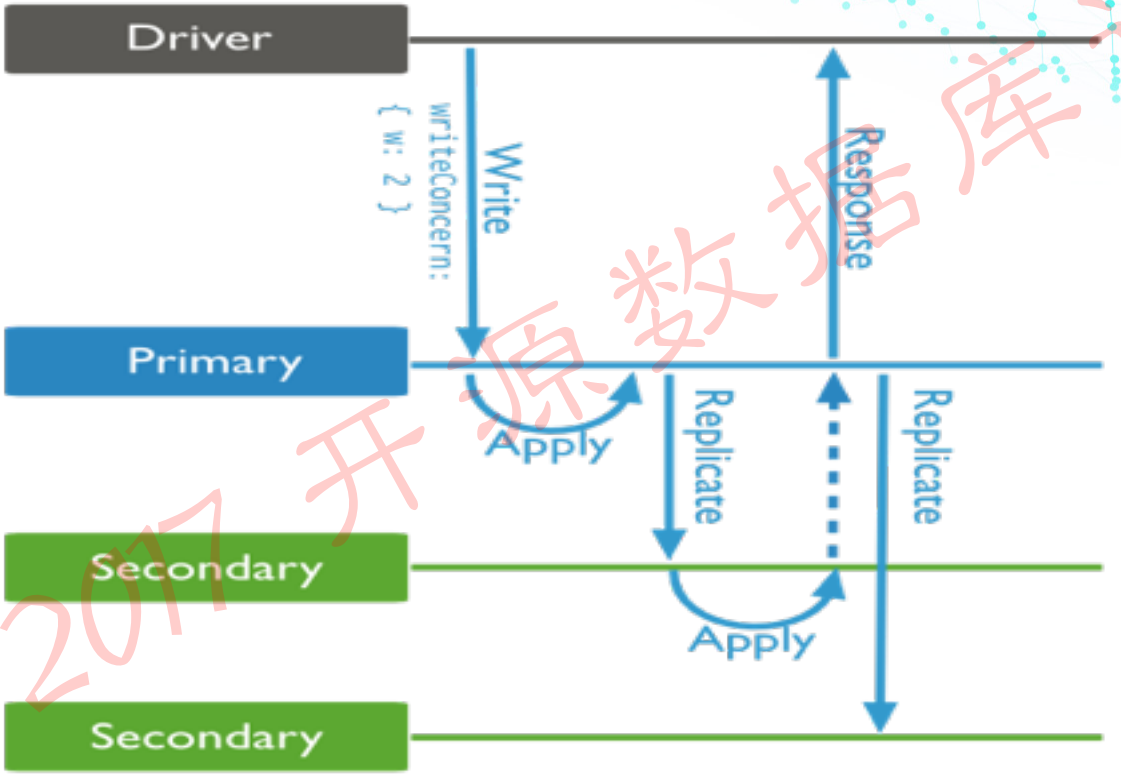




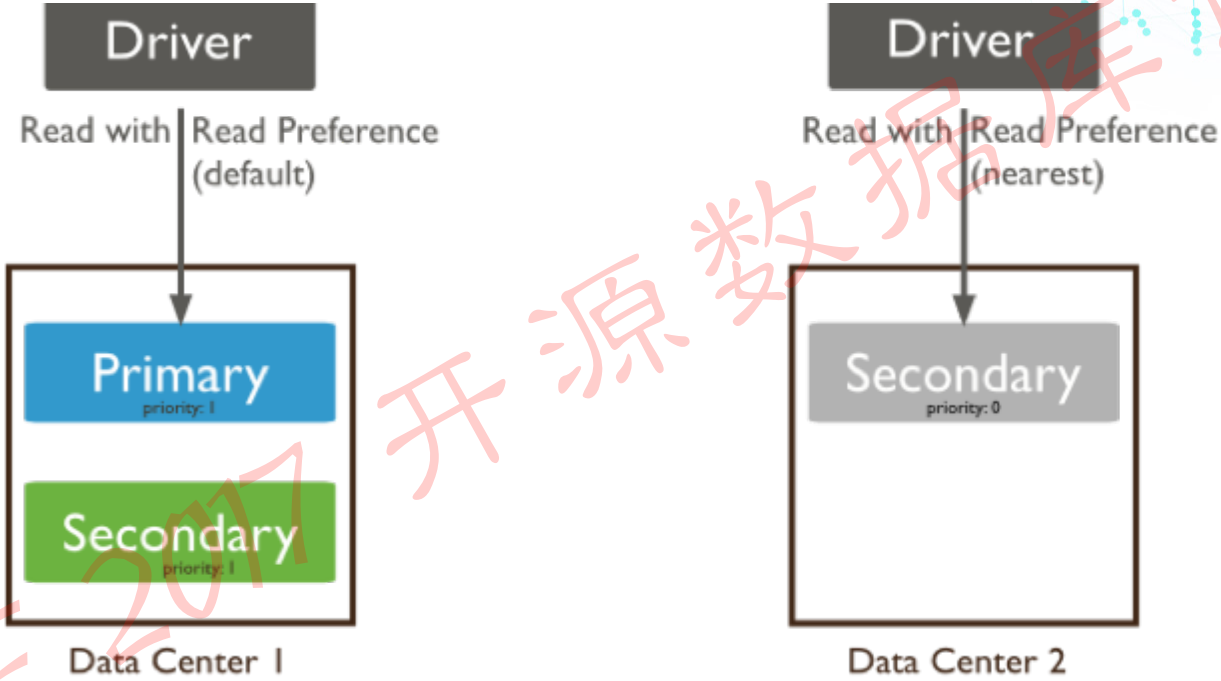
Write Concern {w: 1, j: true}



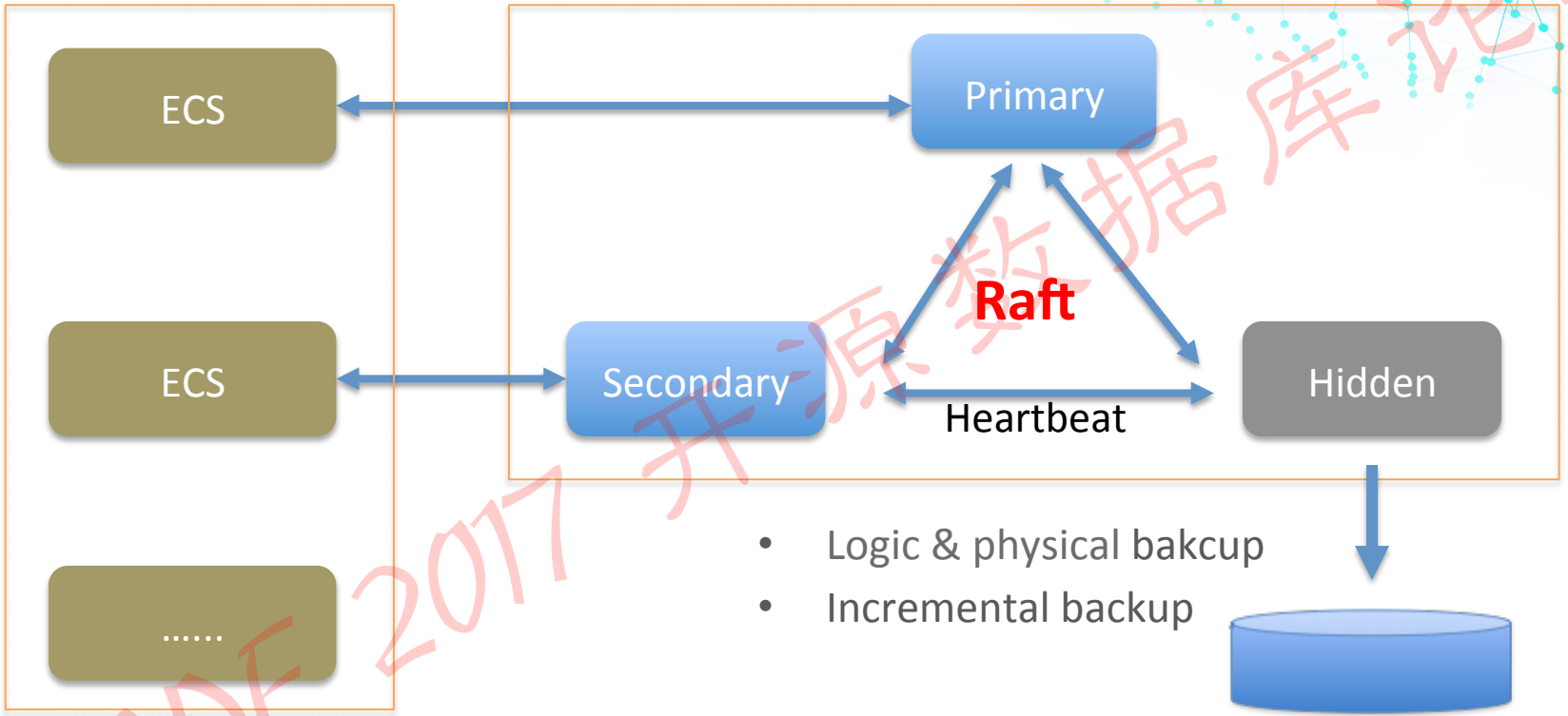
Write Concern {w: 2}



Read Preference



Use case：MongoDB as a service

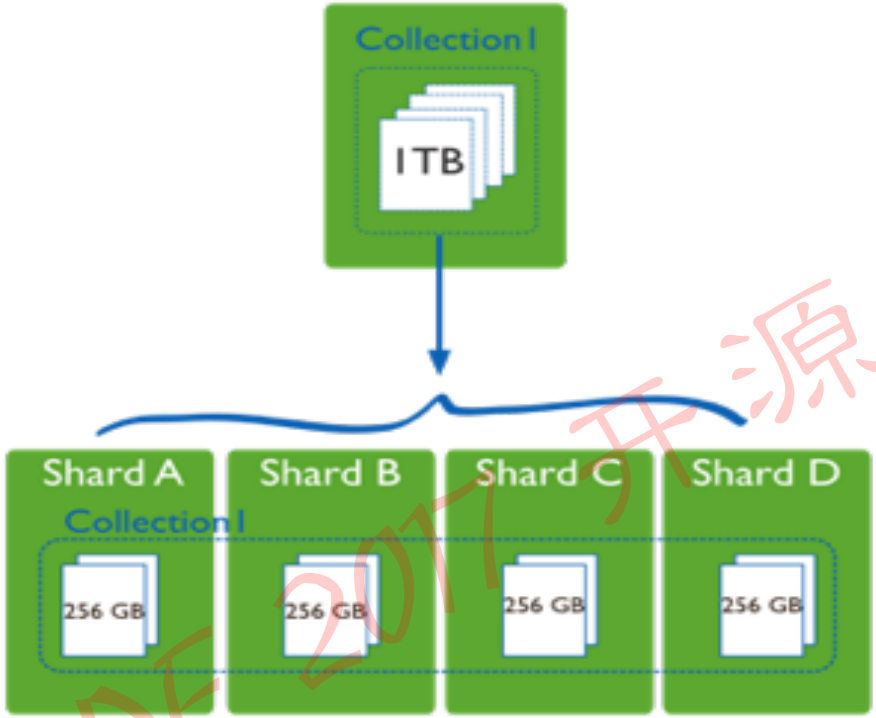


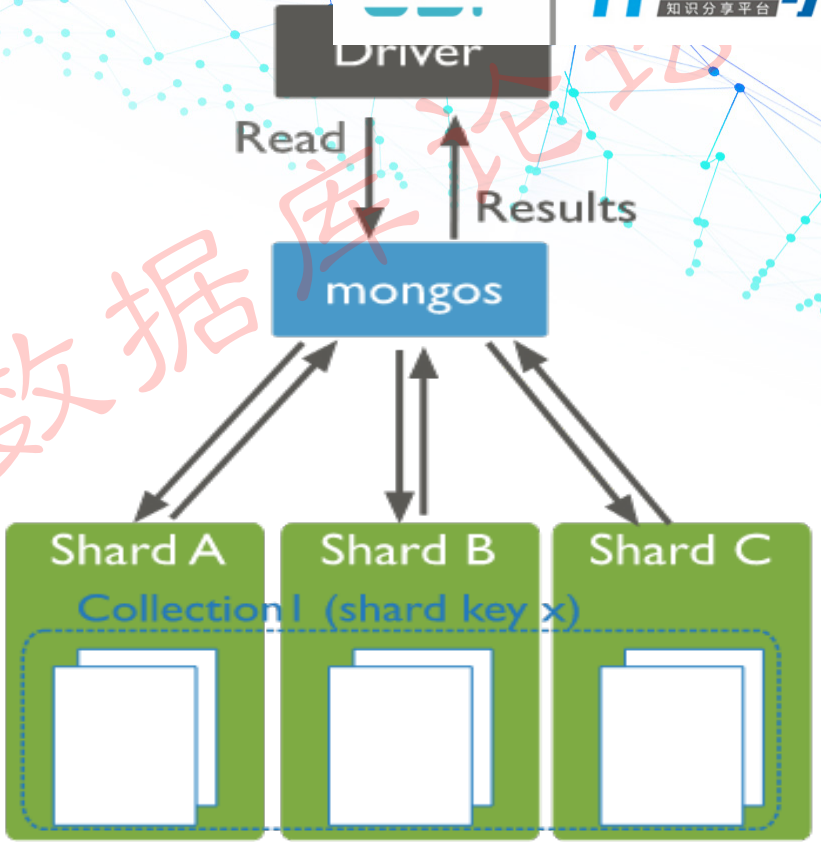
- Logic & physical bakcup
- Incremental backup

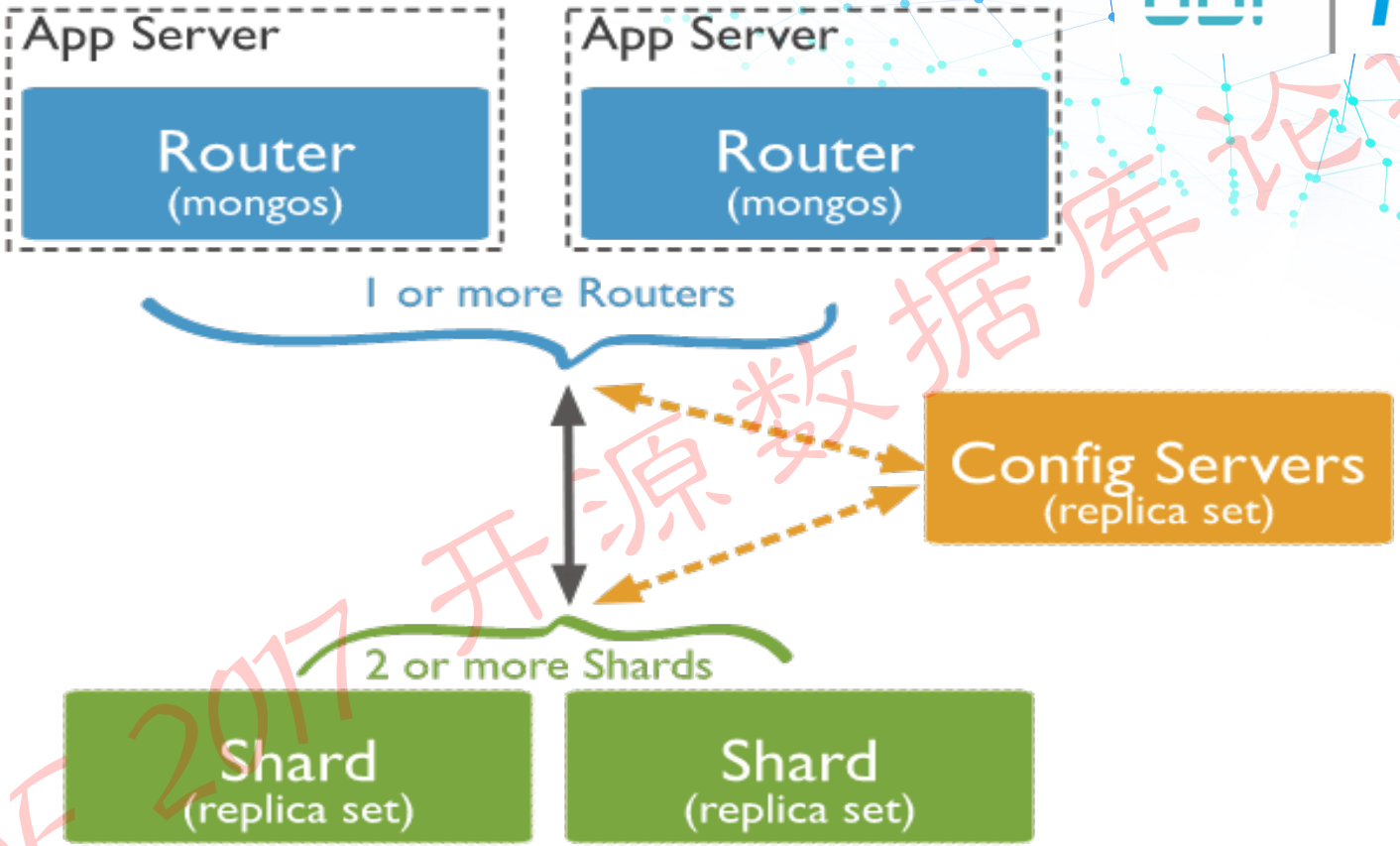
Key Features: MongoDB is a ???

- Document based database
  - JSON document
  - Flexible index & query support
- High reliablity, high availablity database
  - Replica sets
  - Ease of management
- **Scalable database**
  - Sharded cluster
  - Auto load balance

Shard Data for Scalability



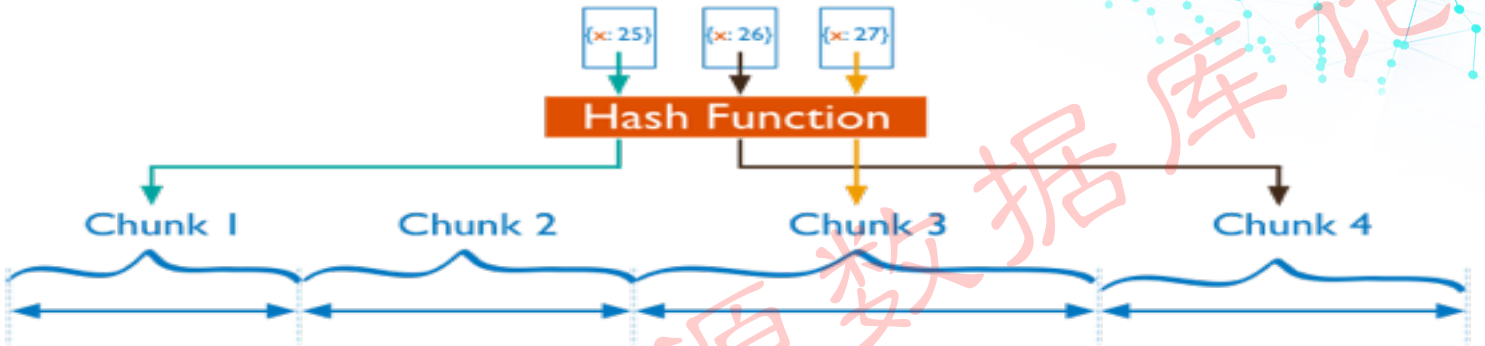




Ranged Sharding

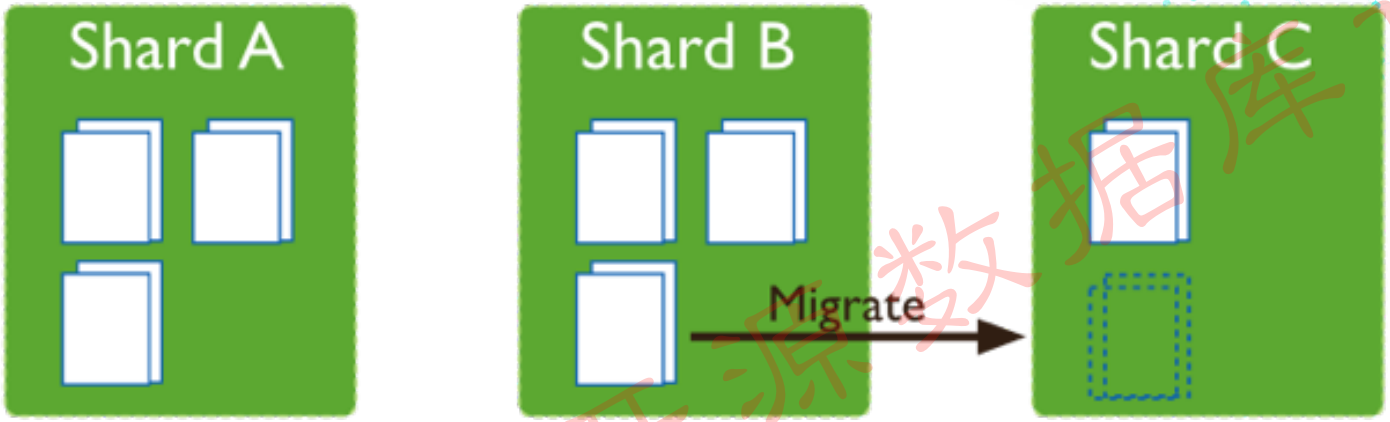
- Efficient for range query
- Write hotspot for monotonical shard keys

Hashed Sharding



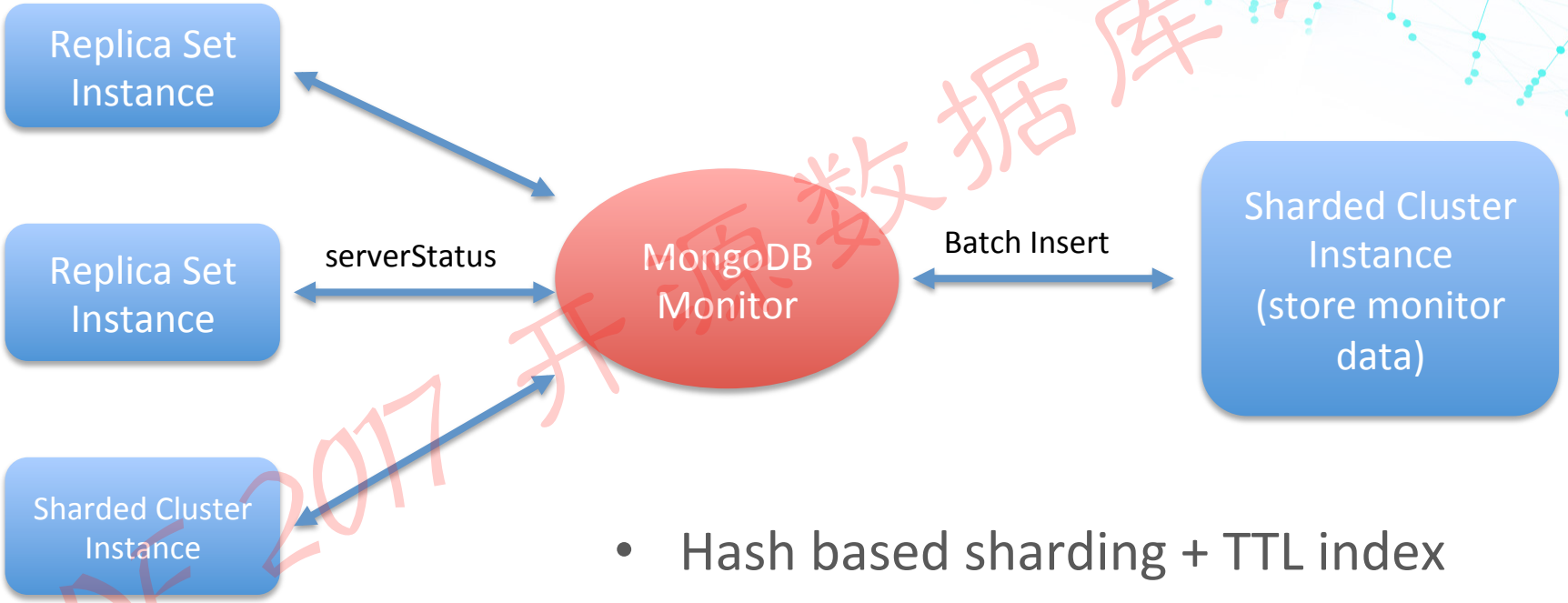
- Evenly distribute data to shards
- Cannot serve range query efficiently

Auto Load Balance



- Auto migrate based on chunk distribution
- Config balancer window
- Disable balancer during backup

Use case： MongoDB Second Level M
Base on MongoDB Shared Cluster



- Hash based sharding + TTL index

Thanks

关注开源数据库论坛