



# **THE MONGODB STRIKES BACK**

**NOSQL WARS SAGA CONTINUES**

# EPISODE 1



보이지 않는 시장

MONGODB  
STRIKES BACK



저는 스마트스터디에서 일하고 있습니다



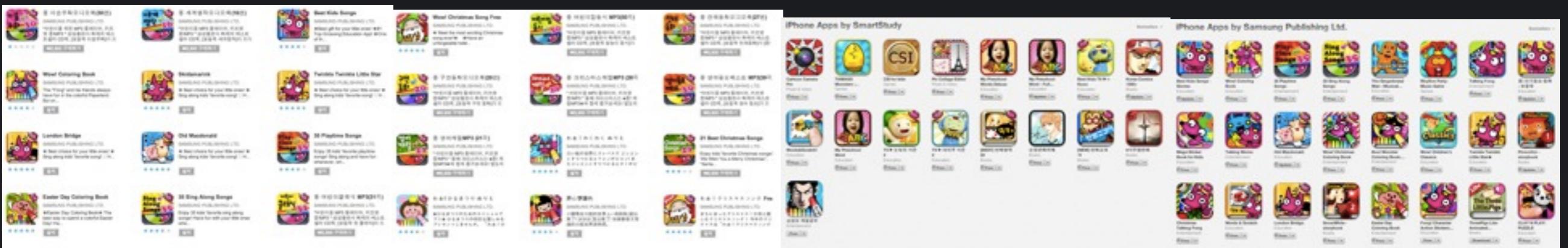
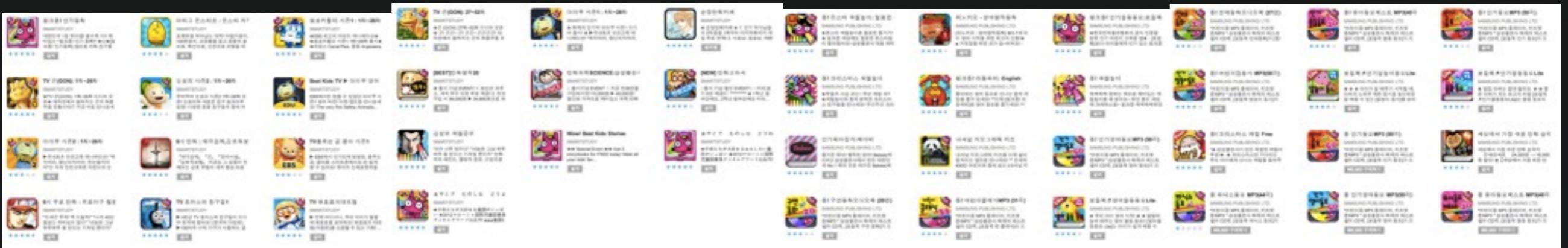
**스마트스터디는 다양한 모바일 앱을  
만들고 직접 서비스하는 회사입니다**



하필이면 2010년에 창업을 해서  
후...



다양한 앱을 만들어 서비스하면  
역시나 정신이 하나도 없습니다



iPhone Apps by SmartStudy

iPhone Apps by Samsung Publishing Ltd.



정신 없을 때에는 로그를 쌓아  
분석하느라 정신이 더 없어집니다



mongoDB

다른 할 일이 많다는 것을 핑계로  
가장 편리해 보이는 것을 골랐습니다

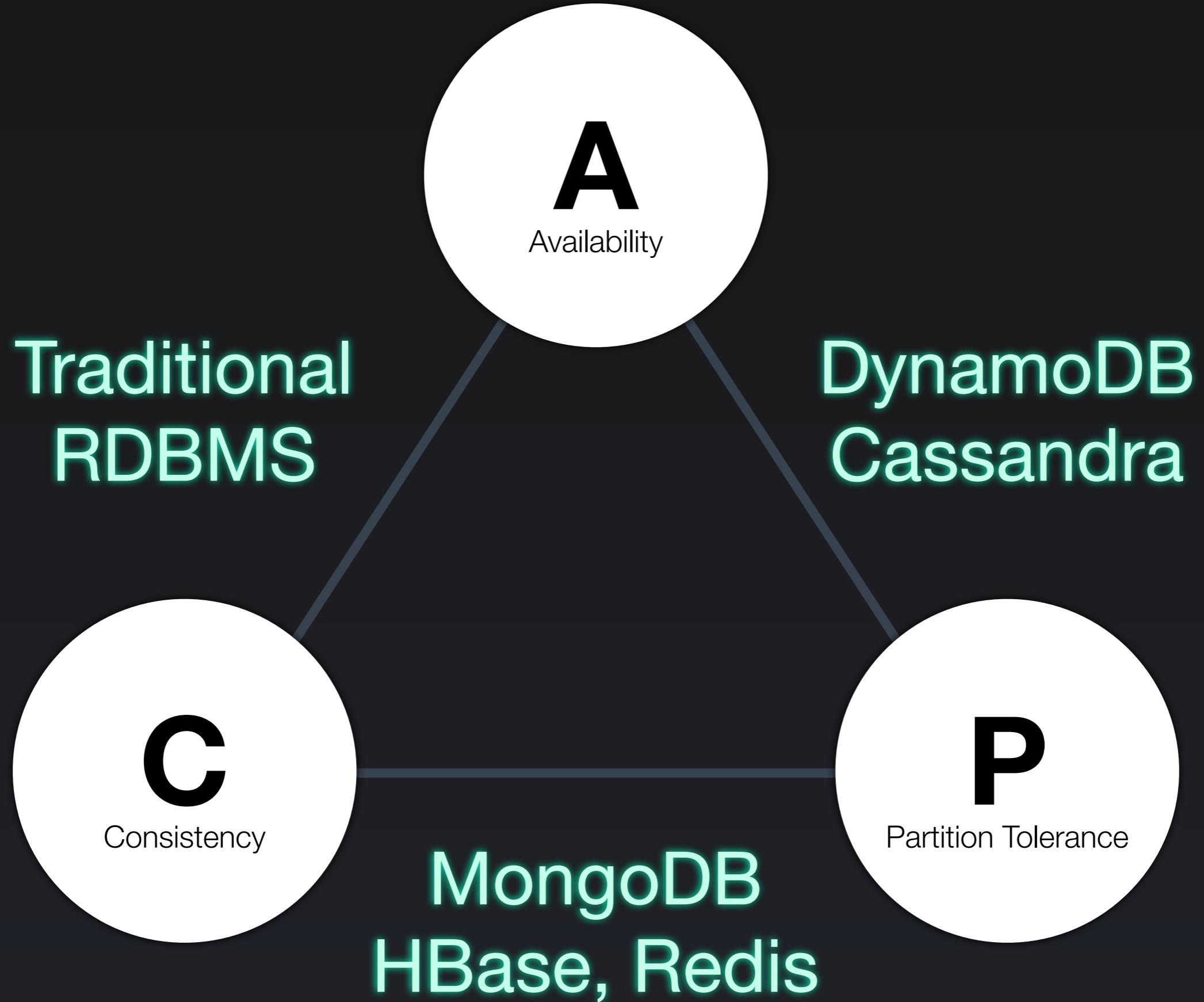
# EPISODE 2



몽고의 습격

MONGODB  
STRIKES BACK

A long time ago in a NoSQL era  
far, far away . . .



365일  
24시간  
항시 대기

Traditional  
RDBMS

한입으로  
두말하기  
없기

DynamoDB  
Cassandra

뼈와 살이  
분리되어도  
연락하기

MongoDB  
HBase, Redis

# Overview

- Document Database
- High Performance
- High Availability
- Easy Scalability

# Overview

- Document Database with JSON
- High Performance ... Really?
- High Availability by ReplicaSet
- Easy Scalability via Sharding

# Document Database

- JSON 으로 받아, BSON 으로 저장.
- 아무 생각 없이 저장해도, 일단 동작한다.
- 프로그래밍 언어에서 쓰던 데이터의 형태  
그대로 저장할 수 있는 것이 장점.

# Document Database

- MySQL과 같은 RDB 시스템 대비,
  - 테이블 스키마가 없다.
  - 즉, ALTER TABLE 이 없음.
  - 인덱스 추가 / 변경도 원하는 때에 background 로 처리 가능.

# Document Database

- MySQL과 같은 RDB 시스템 대비,
  - 콜렉션 내 데이터의 형식이 달라도 됨.
  - 애플리케이션 레벨에서 알아서 사용.
  - JOIN 대신 embedding 문서를 unwind로 풀어서 쓰는 패턴.

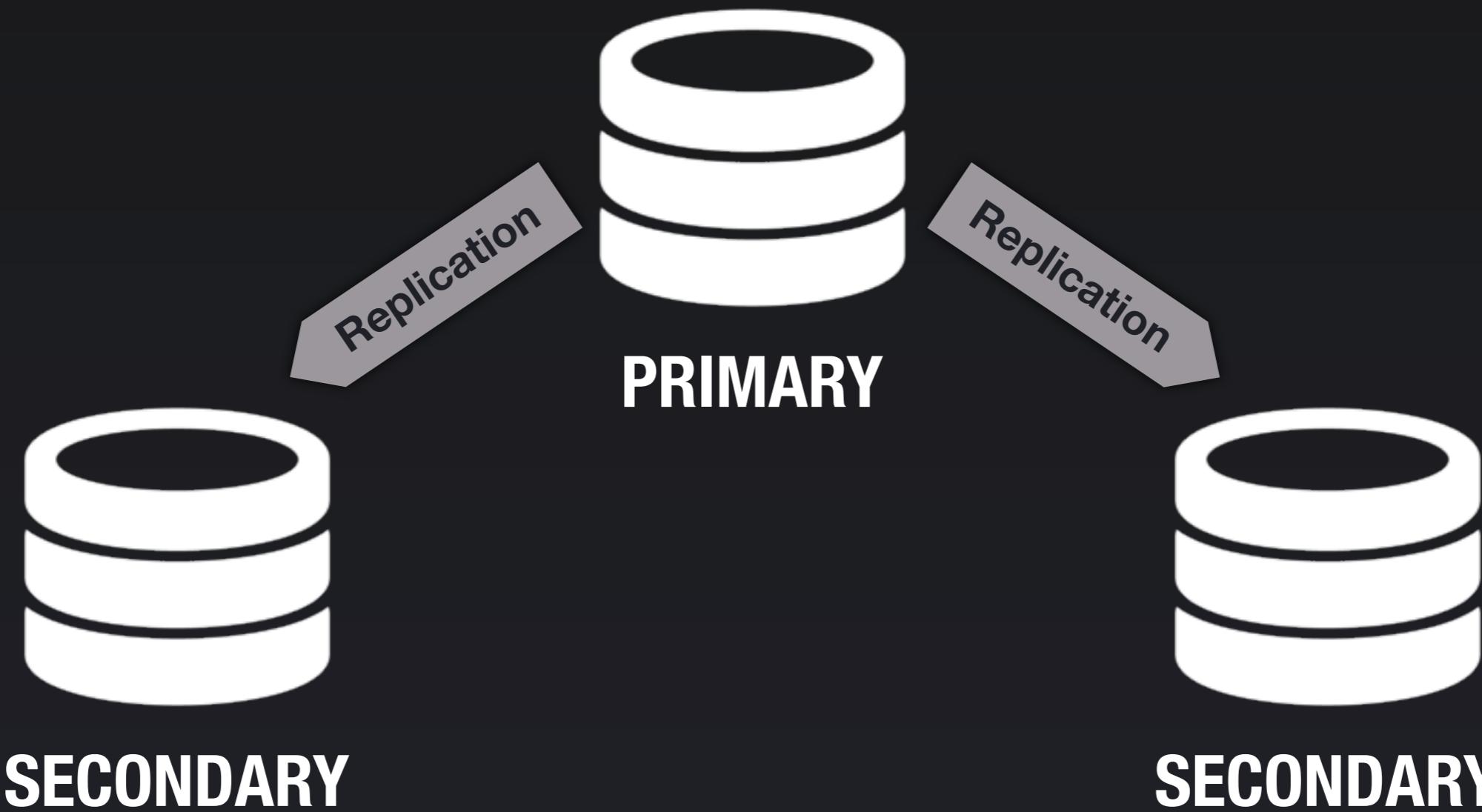
# High Performance

- 경우에 따라 다르지만, 느리지 않음.
- **Memory-mapped file** 을 그냥 사용.
- 메모리 용량 내에서는 = Page Fault 발생 전까지는 In-memory DB 굽의 성능.

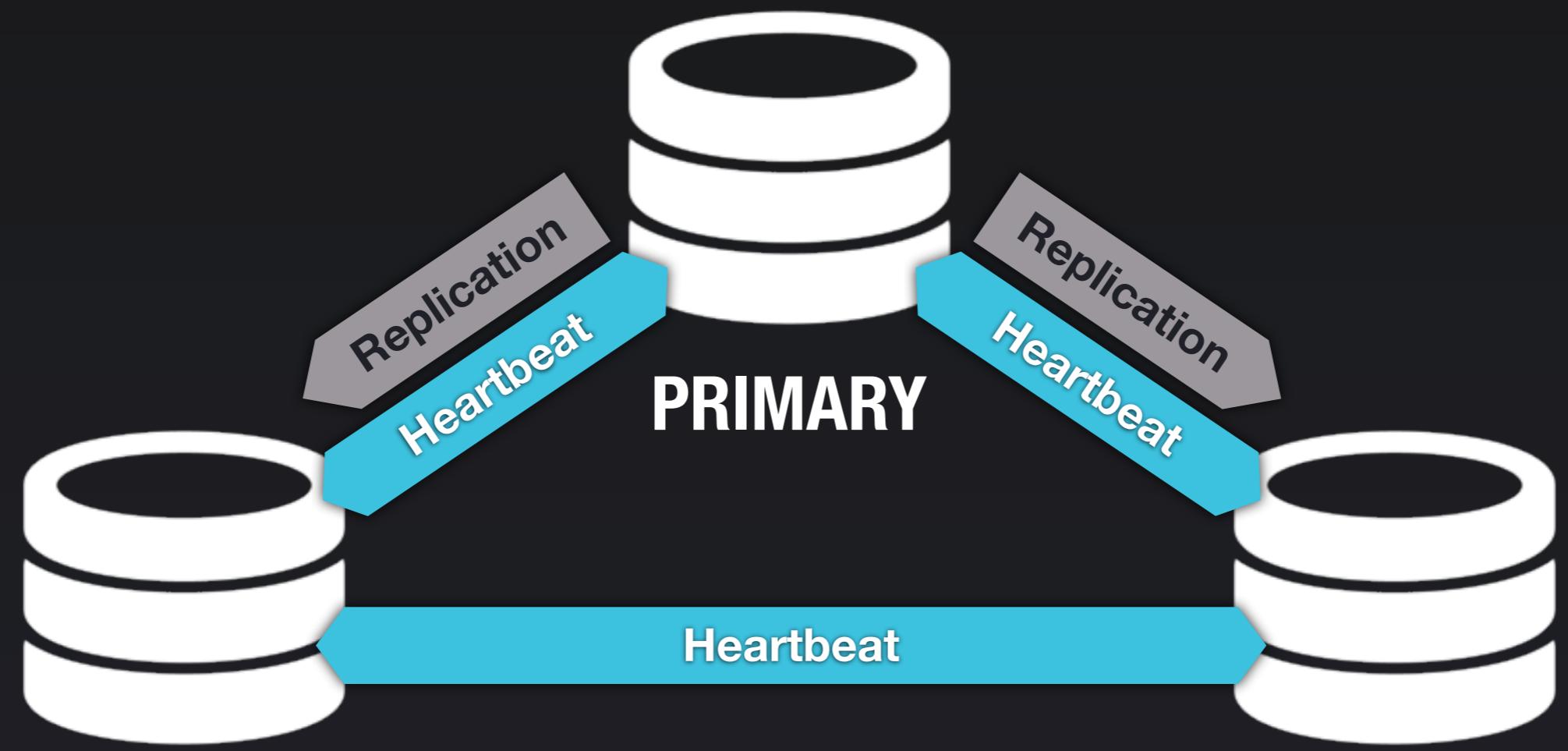
# High Availability

- 과거의 Master-Slave Replication 대신,
- 3대 이상의 노드가 결합된 보다 안정적인 ReplicaSet 지원.
- Single Primary - Multiple Secondary
- ReplicaSet 내에서 자동 Failover 지원.

# High Availability



# High Availability

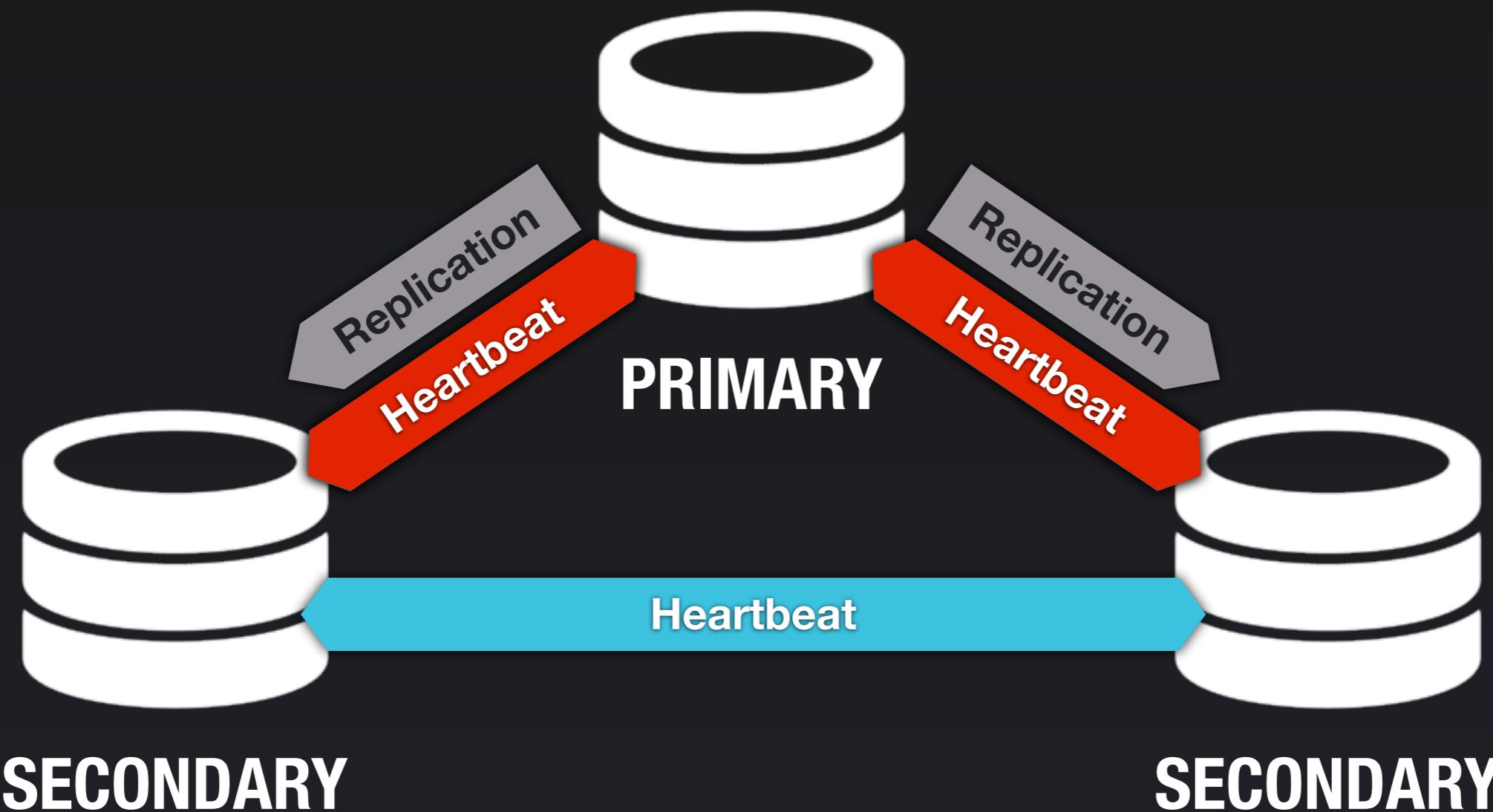


**SECONDARY**

**SECONDARY**

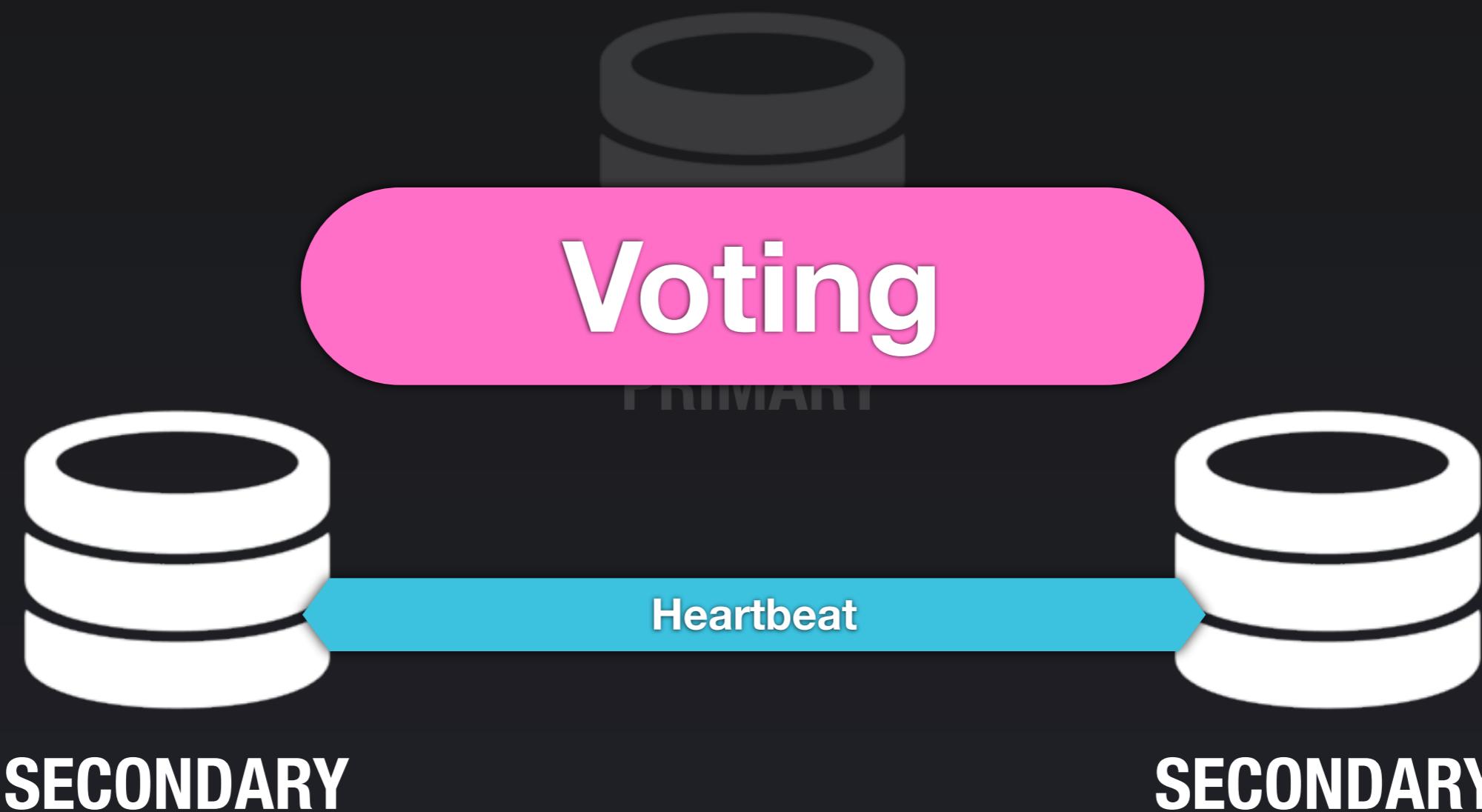
**Heartbeat** / every 2 seconds

# High Availability

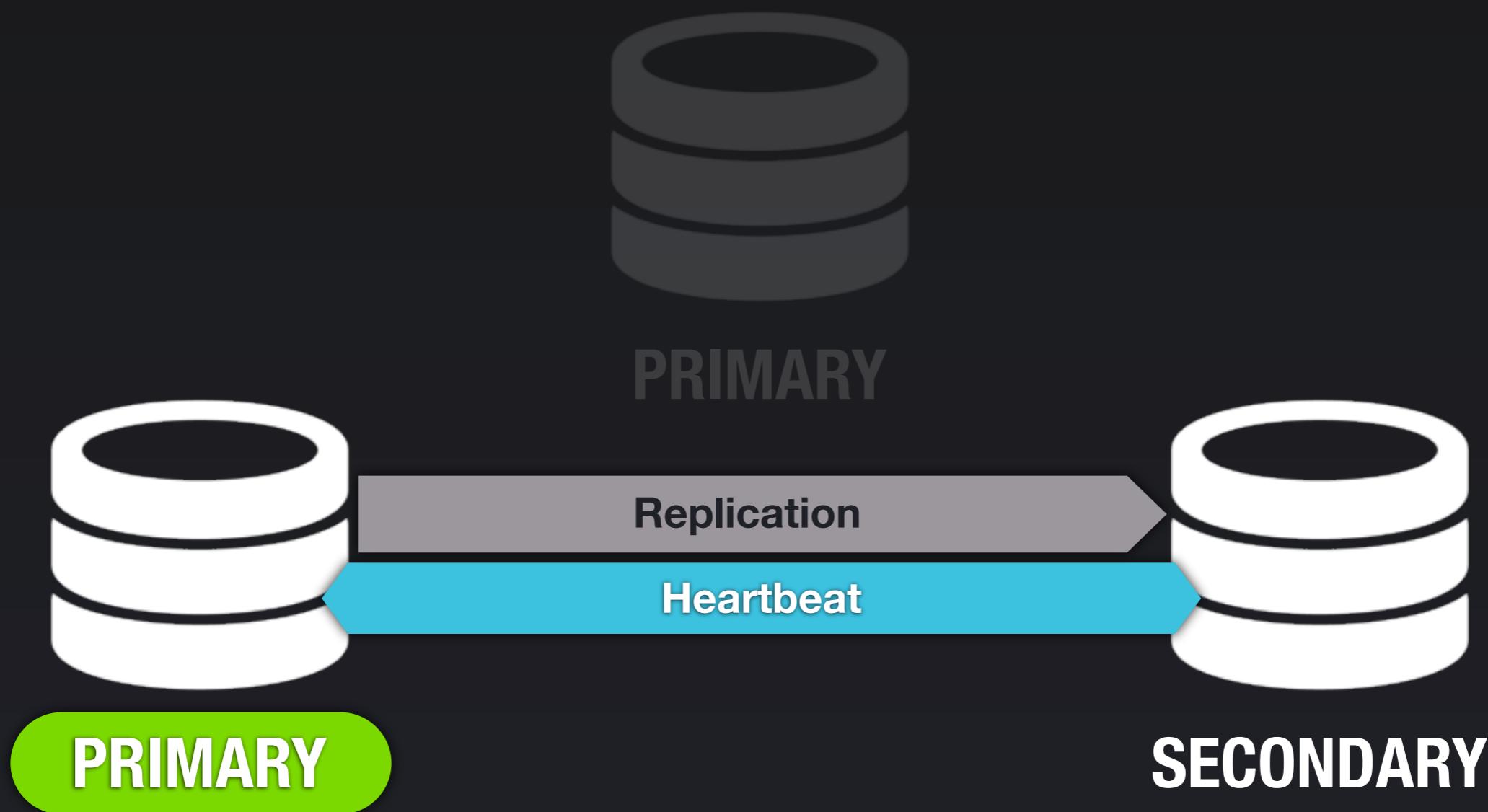


**Heartbeat failure / after 10 seconds**

# High Availability



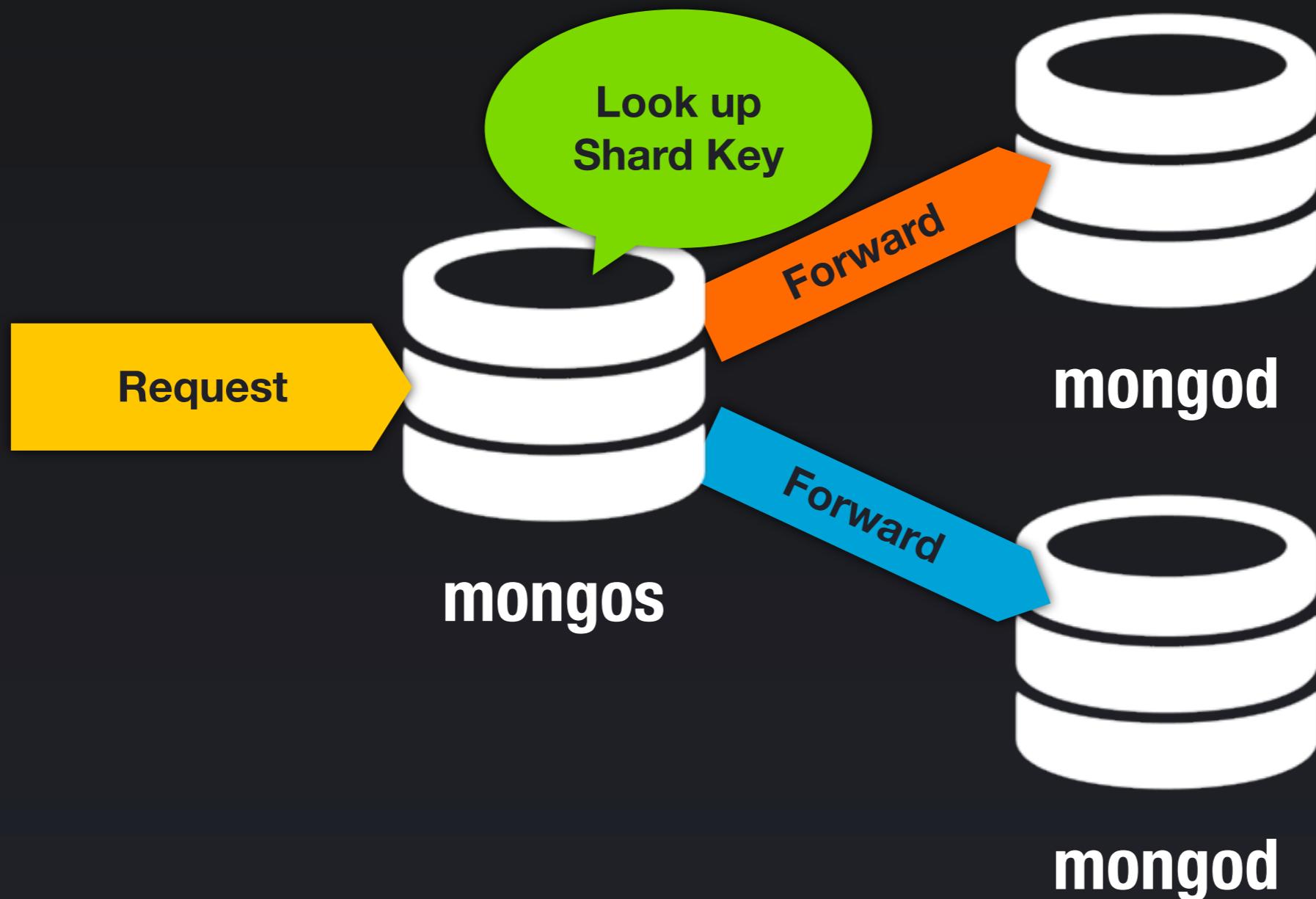
# High Availability



# Easy Scalability

- 문서의 특정 필드를 Shard key로 지정.
- N 대의 shard에 데이터를 분산 저장.
- 조회나 통계 명령어를 경우에 따라 각 shard에서 나눠서 실행 가능.

# Easy Scalability



# Easy Scalability

- Shard key로 지정된 필드의 값이 충분히 분산되지 않을 경우, 특정 shard에 데이터 (chunk)가 몰리는 현상이 생길 수 있음.
- 이를 피하기 위해 MongoDB 2.4 부터 Hashed Sharding 지원.
- 아직까지는 MD5만 사용.

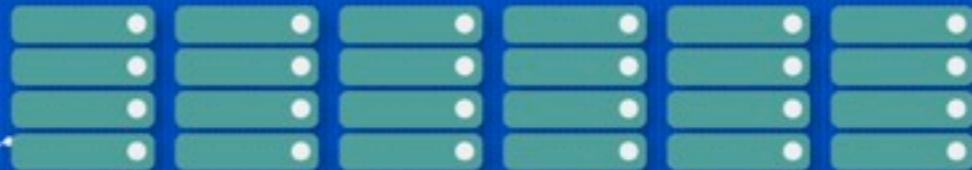


# My Overview

- 설치가 간편하다.
- All-in-One.
- Key-Value DB 들에 비해 쓰기 편하다.
- Geospatial Index / Query 지원.
- 제작사 10gen의 공식 교육 프로그램 존재.

## Free Online MongoDB Training

Classes are now in session. Sign up today!



M101J: MongoDB for Java Developers



M101P: MongoDB for Developers



M102: MongoDB for DBAs



<https://education.10gen.com/>

언제까지 셋업만 할 것인가?



이것저것  
따져보기  
귀찮을때  
몽고디비

우리는 당장 통계부

몽고디비를 설치하라!

# EPISODE 3



몽고의 함정

MONGODB  
STRIKES BACK

# Legacy strategy

- MongoDB 1.6 기반으로 작업.
- 데이터베이스를 쪼개도, 콜렉션을 쪼개도 관계 없이 모든 작업은 어쨌든 글로벌 랙.
- 이왕 이렇게 된거, 하나의 콜렉션으로!

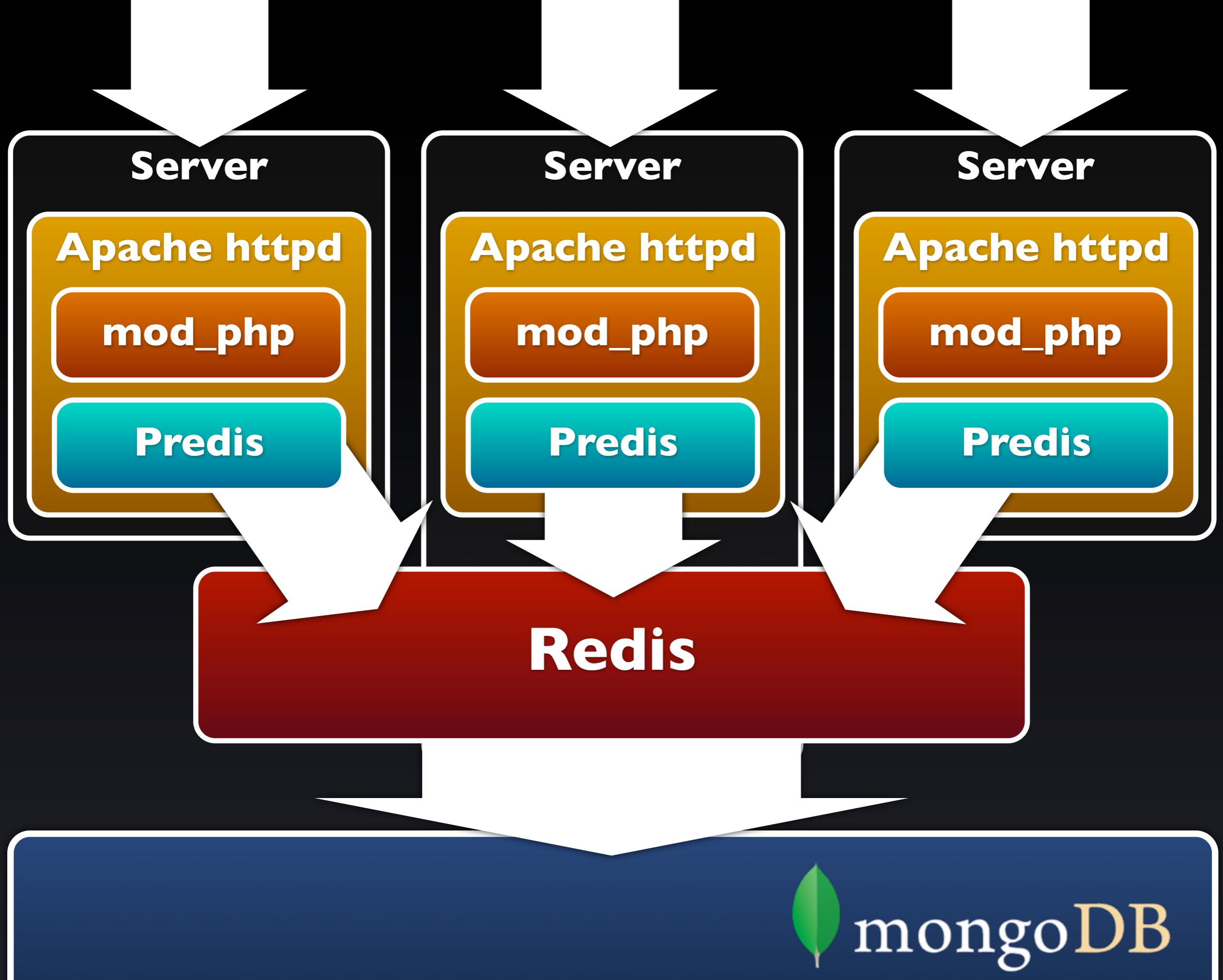


이...  
이건 정말  
듣도 보도 못한  
발상이군.

# Pitfall

- MongoDB 1.6 기준,
  - 일단 잔뜩 쌓긴 쌓았는데...
  - 그리고 쓰기가 어려운 MapReduce.
  - 통계 관련 명령어가 빈약.
  - 저장 후, 디스크에 잘 쓰였을까 불안함.







Primary

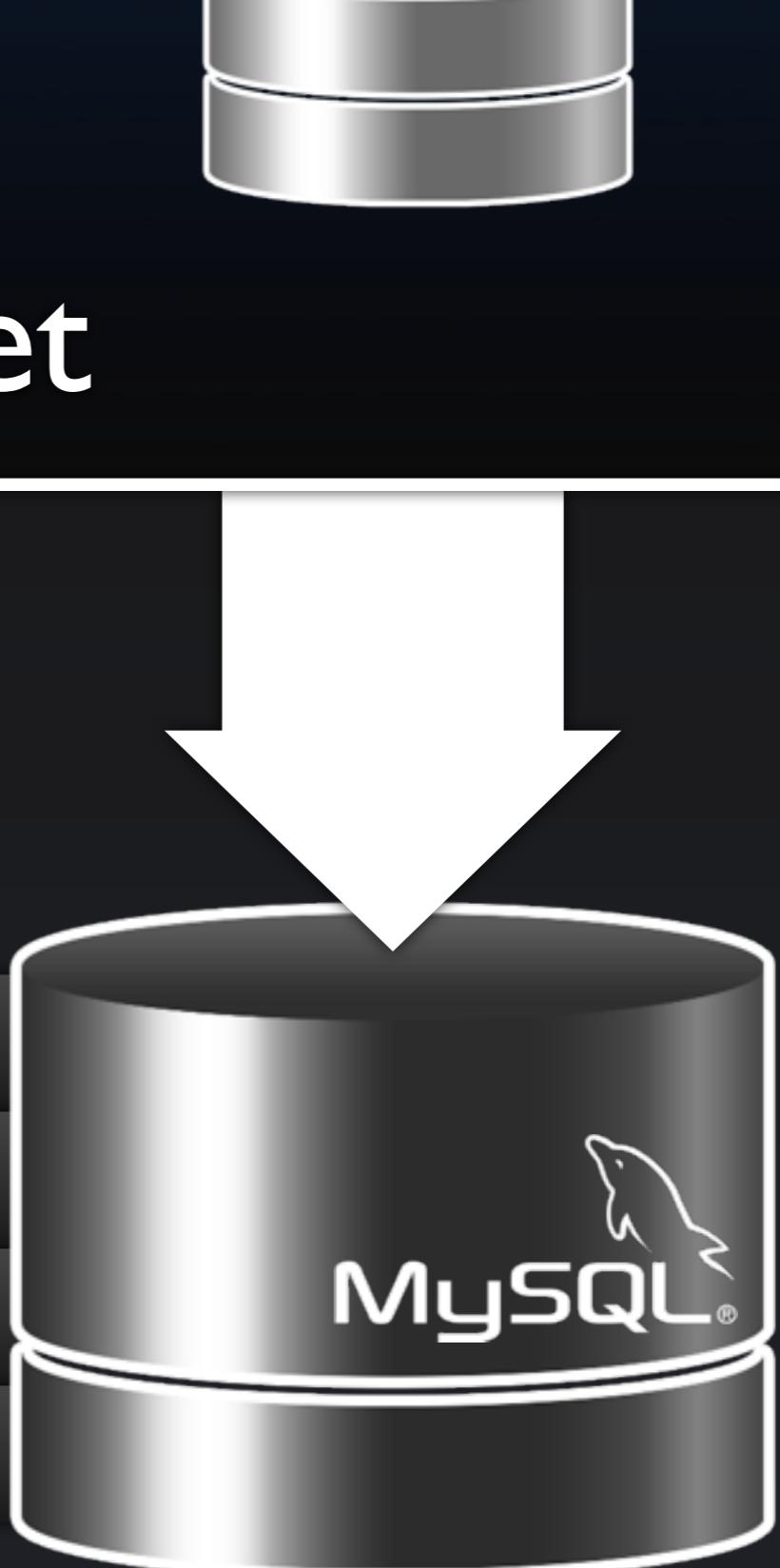
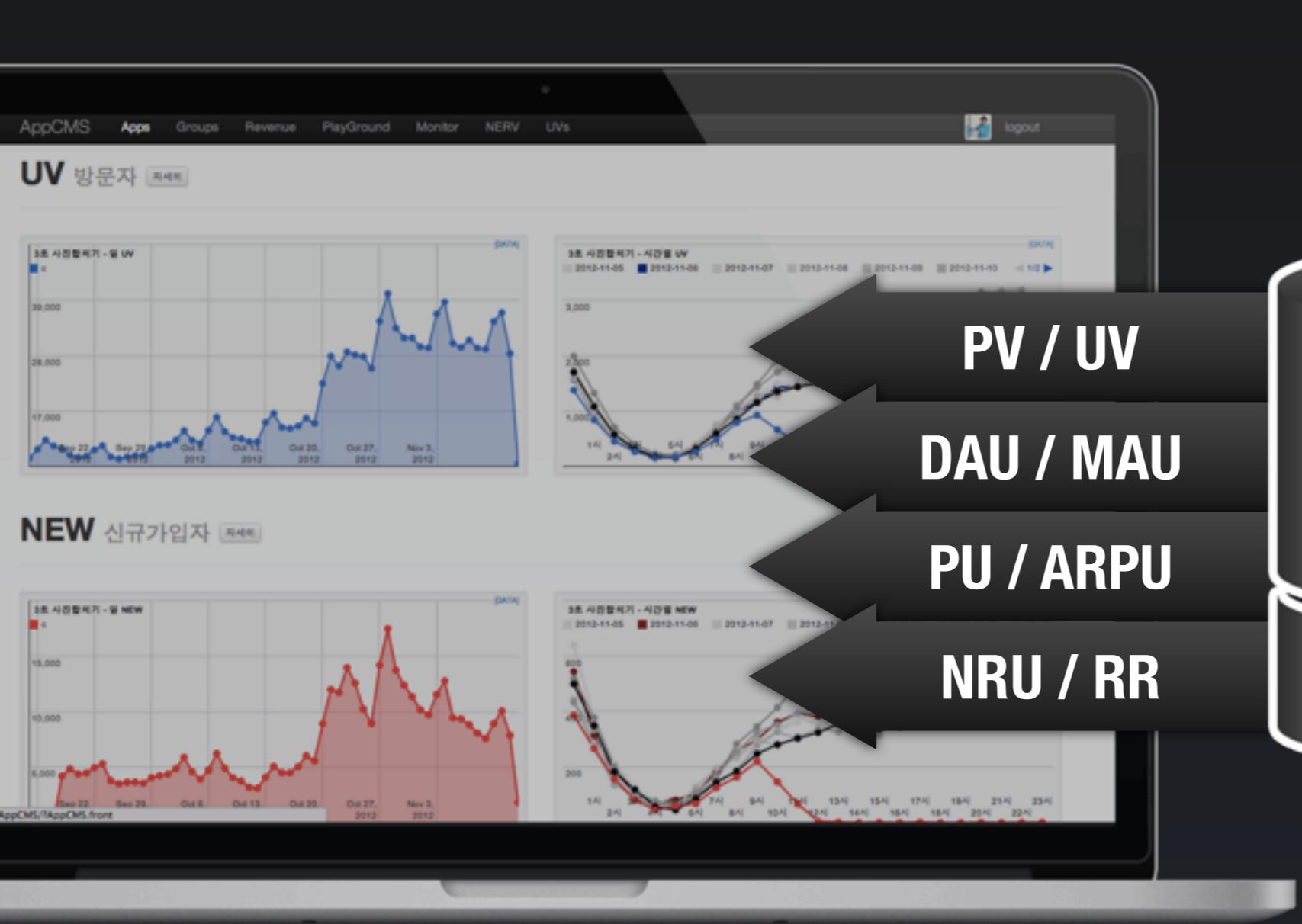


Secondary

Secondary

ReplicaSet

# ReplicaSet





# 유통기한 초과

Redis, MongoDB 그리고 MySQL  
모바일 애플리케이션 서비스에서의 로그 수집과 분석

# MongoDB today

230

20,000,000

10,000,000,000

# MongoDB today

초당 **230로그**

일간 **20,000,000로그**

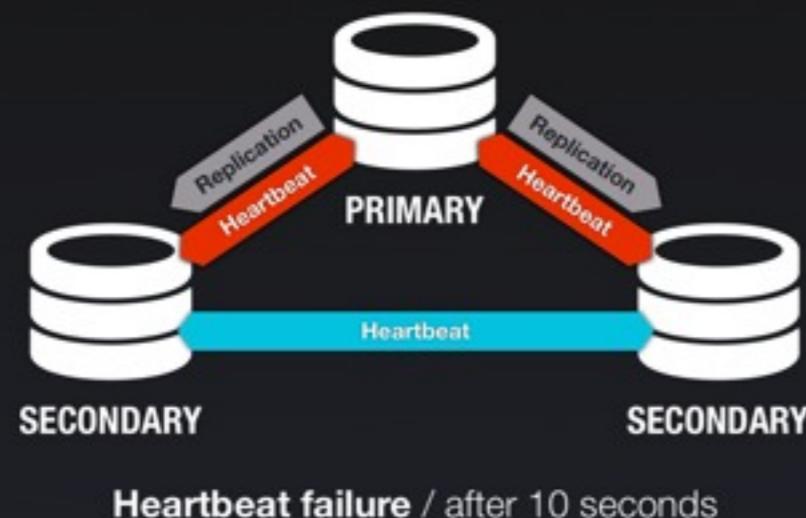
일간 **10,000,000,000바이트**

# MongoDB today

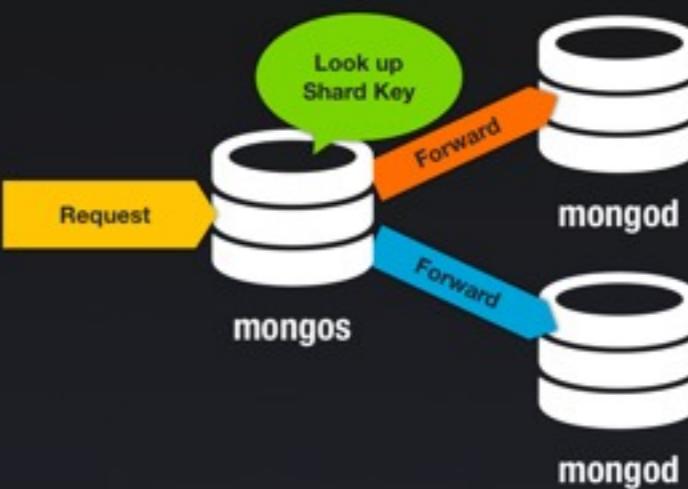
- 누적 35억 건 정도.
- 적다면 적고, 많다면 많은 양.
- 양이 많아지니, 로그를 옮기는 시간이 문제.
- 결국 실시간 통계 자료가 아니게 됨.

# No free lunch

## High Availability



## Easy Scalability



## Failover / Auto-balancing

언제나 원하는 대로 된다면야

# Deploying Hell

- 한 대에 간편하게 설치해서 쓸 수도 있지만,

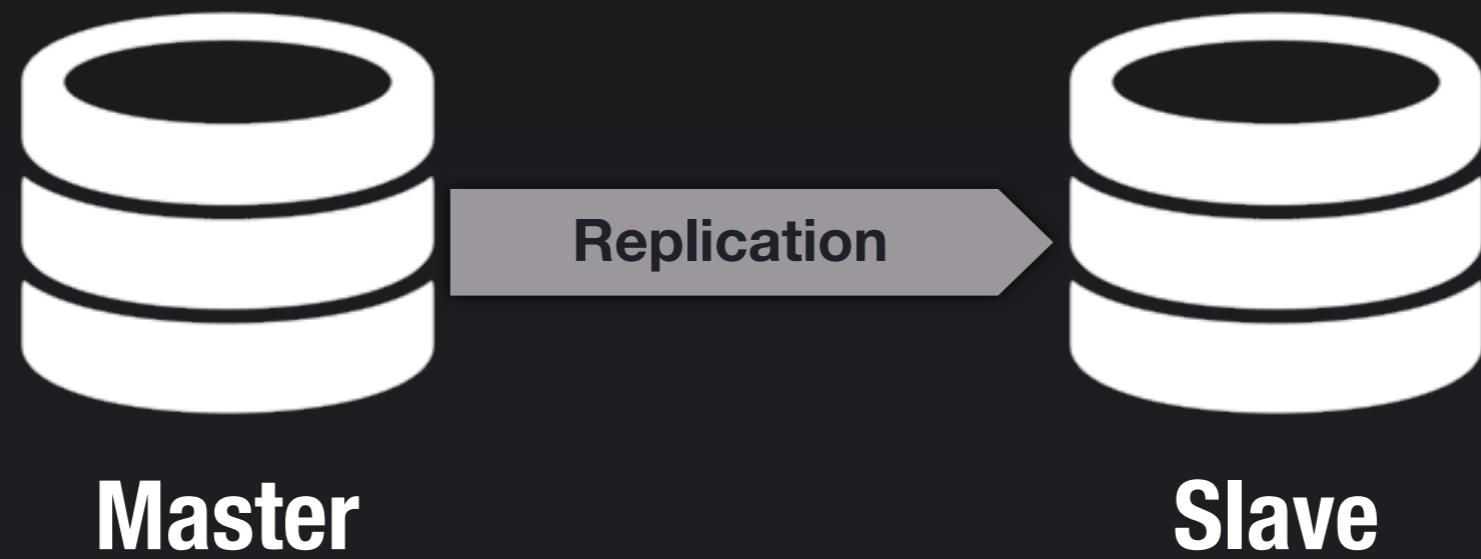
# Deploying Hell



**mongod**

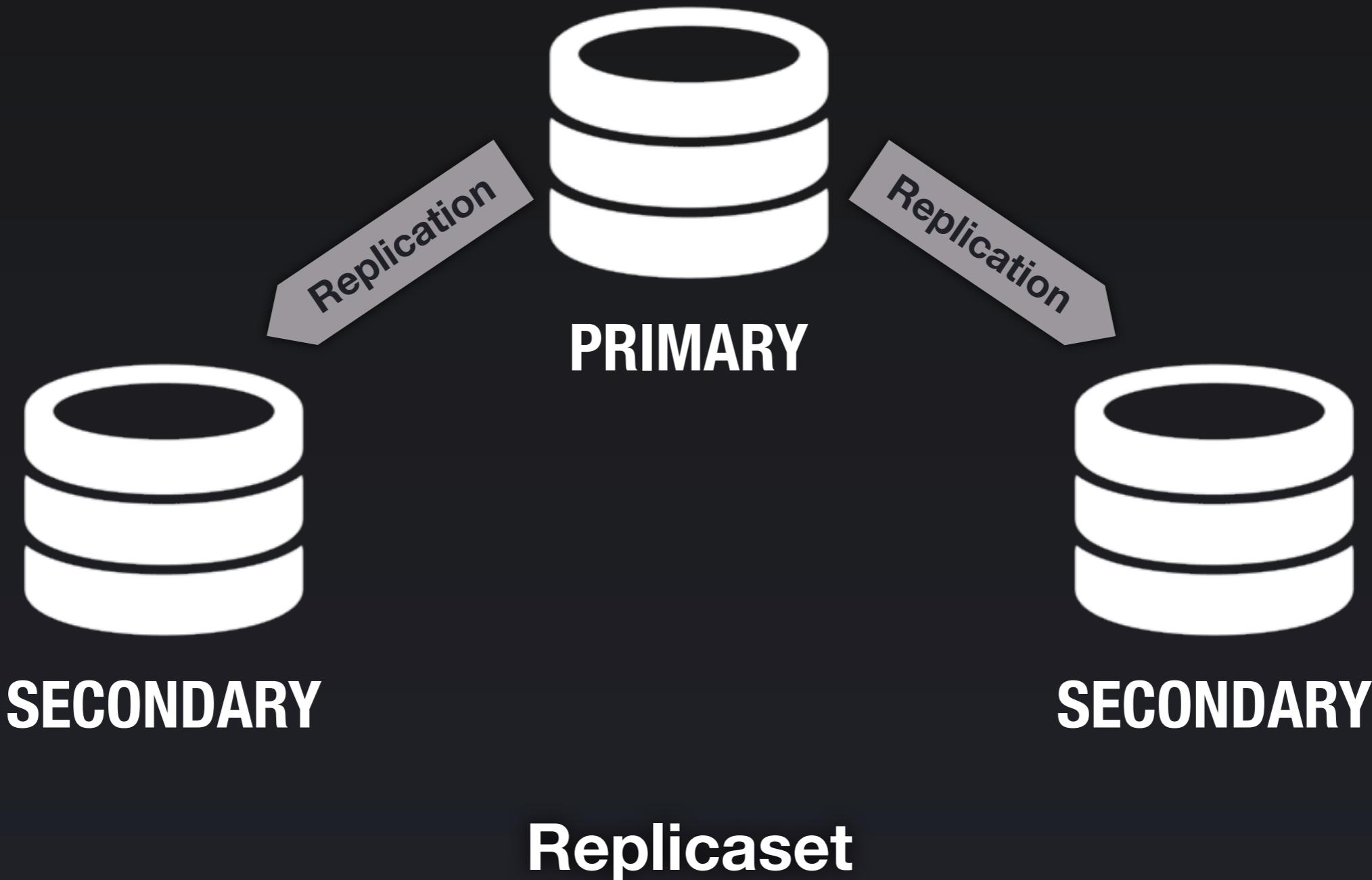
**Single**

# Deploying Hell



**Master - Slave Replication**

# Deploying Hell



# Deploying Hell



**mongos**



**mongod**



**mongod**



**mongod**

**Shard**

# Deploying Hell



**mongos**



**mongos**



**mongos**



**mongod**



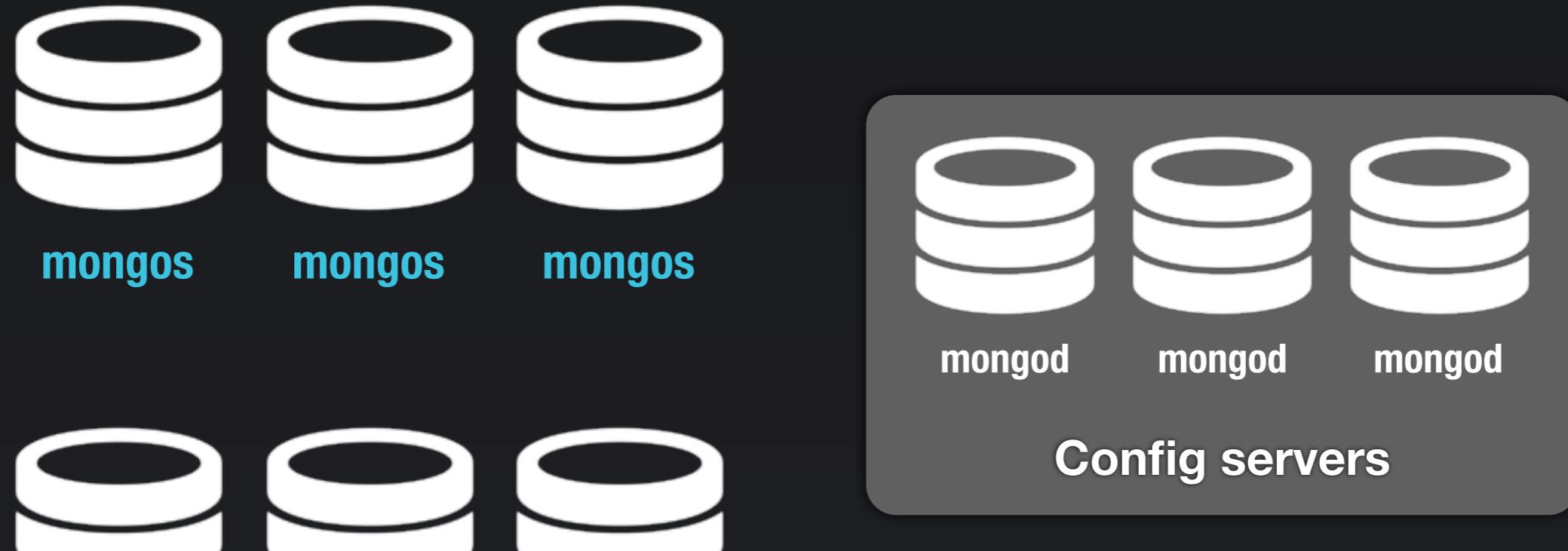
**mongod**



**mongod**

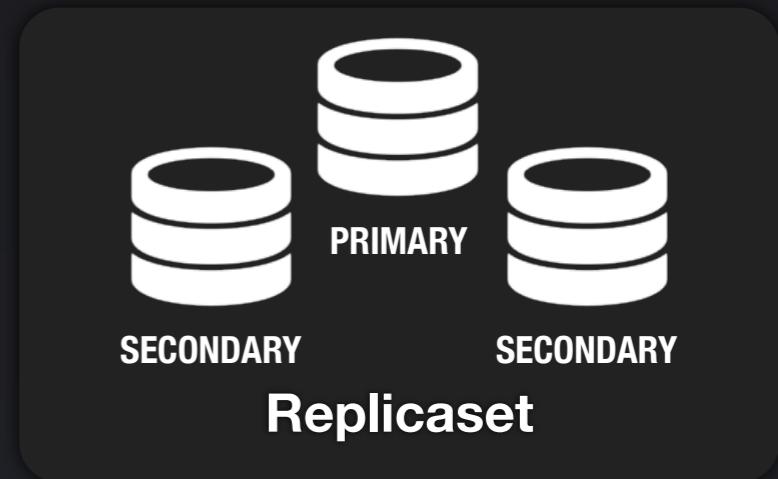
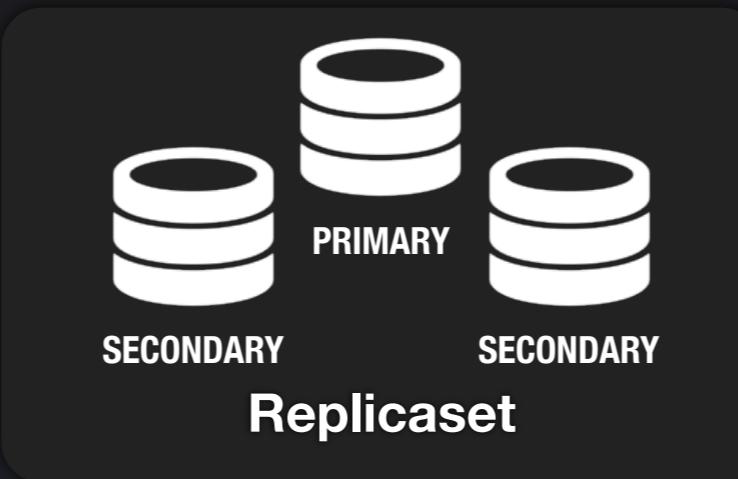
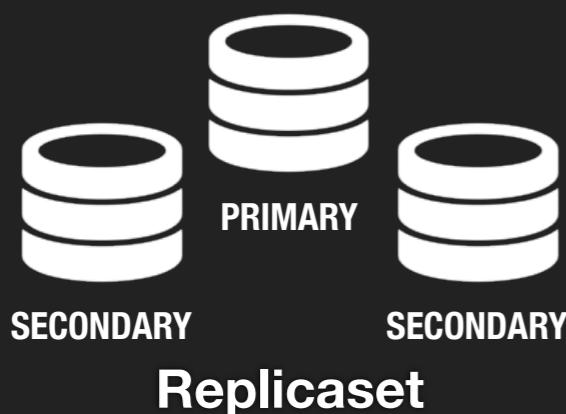
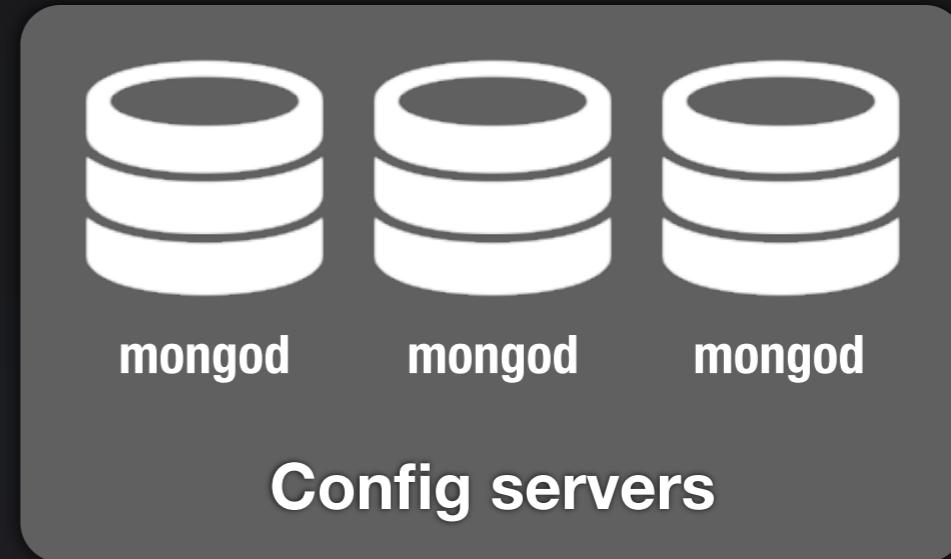
**Shard**

# Deploying Hell

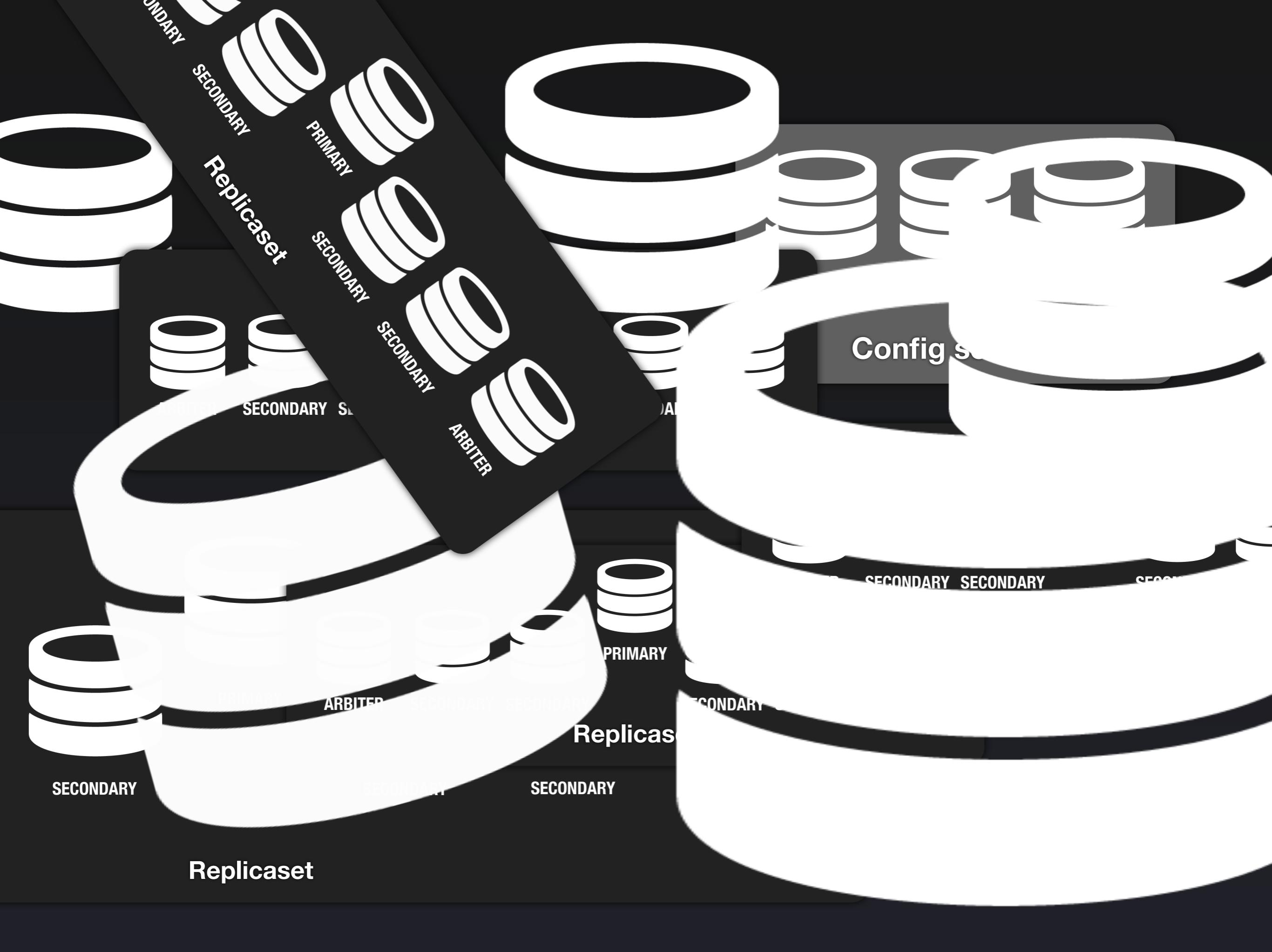


Shard

# Deploying Hell



Sharded Cluster



# Deploying Hell

- 목적에 맞는 “적절한” 설계가 필요.
- 실시간으로 처리해야 하는 데이터 크기는?
- 단순히 Shard node를 추가한다고 해서 병렬 처리량이 늘어나는 것은 아니다.
- SSD / RAM 추가 등의 하드웨어 변경에 따른 성능 향상 폭이 큼.

# EPISODE 4



새로운 희망

MONGODB  
STRIKES BACK

# New Hope

- MongoDB 버전에 따른 기능 추가

<b>Aggregation framework</b>	<b>2.1</b>
<b>Database-level lock</b>	<b>2.2</b>
<b>Yield lock on page fault</b>	<b>2.2</b>
<b>V8 JavaScript engine</b>	<b>2.4</b>

# Aggregation framework

- MongoDB 2.1
- 파이프라인을 통해 여러 조건과 값 추출을  
연쇄적으로 지정할 수 있음.
- 기존의 콜렉션 함수군에서 제공하는 것보다  
강력한 기능을 제공.

# Aggregation framework

- Map / Reduce 가 있는데 또?
  - Javascript 구현이 아닌,
  - 통계 전용 C++ 구현.
- Collection에 있던 일부 통계 함수들은?
  - 그냥 놔두고 따로 구현.

# Aggregation framework

- 기존 Collection 함수 대비 뭐가 나은가?
  - 제약 조건이 덜하다.
    - 파이프라인별 결과가 16MB 까지.
    - 그외 메모리 사용량 제한.
  - SQL 만큼은 아니지만 좀 더 편하다.

# Aggregation example

# Simple **distinct**

## **SQL**

```
SELECT DISTINCT last_name FROM users
```

## **Collection**

```
db.users.distinct('last_name')
```

## **Aggregation**

```
db.users.aggregate( [  
  { $group: { _id: '$last_name' } }  
]
```

# Group Summary

**SQL**

```
SELECT team, SUM(score) AS total, COUNT(*)  
FROM scoreboard  
GROUP BY team  
ORDER BY total DESC
```

# Group Summary

## Collection with MapReduce

```
db.scoreboard.group( {  
    key: { team: true },  
    initial: { total: 0, count: 0 },  
    reduce: function(obj, prev) {  
        prev.total += obj.score;  
        prev.count += 1;  
    }  
} )
```

# Group Summary

## Aggregation

```
db.scoreboard.aggregate( [  
  { $group: { _id: "$team",  
              total: { $sum: "$score" } } },  
  { $sort: { total: -1 } }  
] )
```

# My friends **feed**

## SQL

```
SELECT msg FROM feed  
WHERE user_id IN  
( SELECT id_to FROM friend  
    WHERE id_from = x )  
ORDER BY written_dt  
DESC LIMIT 10
```

# My friends **feed**

## Collection with MapReduce

```
db.feed.find( {  
    user_id: { $in: db.friend.group( {  
        cond: { id_from: x },  
        initial: { friends: [] },  
        reduce: function(obj, prev) {  
            prev.friends.push(obj.id_to);  
        }  
    } )[‘friends’] }, { msg:1, _id:0 } )  
.sort( { written_dt: -1 } )  
.limit( 10 );
```

# My friends feed

## Aggregation

조인, 서브쿼리  
그런거 안돼요ㅋ

# Aggregation framework

- SQL to Aggregation Framework  
Mapping Chart 참고.
- SQL 처럼 WHERE, GROUP, ORDER ...  
다 되는 것 같지만 실제로는 한계가 많음.
- 조인이나 서브쿼리가 안되므로 필요하면  
embedding.

# Aggregation framework

- 대단한 줄 알았는데 별거 없네?
  - 그래도 MapReduce 보다 몇 배 빠름.
  - Javascript 엔진이 바뀌었어도 여전히 BSON to JSON 변경에 시간이 소요.
  - 여러 Aggregation 을 동시에 수행 가능.

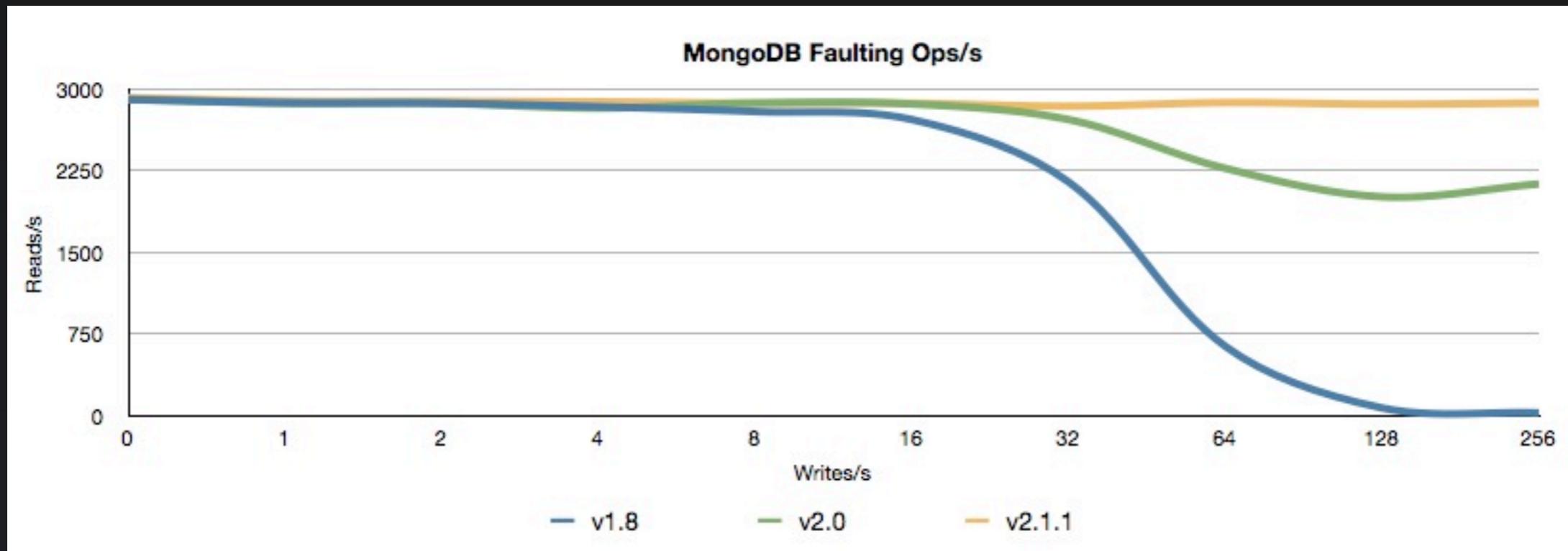
# Database-level lock

1.6	<b>Global lock</b>
2.0	<b>Global lock</b> <b>Yield the write lock</b>
2.2 ~ 2.4	<b>Database-level lock</b> <b>Yield lock on page fault</b>
Future	<b>Collection-level lock</b>

# Yield lock on page fault

- 읽고 쓰는 과정에서 데이터가 메모리에 없다고 판단되면, 락을 다른 작업에게 양보.
- 또는 읽고 쓰는 과정이 길어지면, 잠시 다른 프로세스에게 락을 양보.
- 모든 데이터가 메모리에 올라와 있는 상황에서는 별 차이 없음.

# Yield lock on page fault



- <http://blog.serverdensity.com/goodbye-global-lock-mongodb-2-0-vs-2-2/>

# V8 JavaScript engine

- MongoDB 2.4
- 기존에 사용하던 SpiderMonkey 가 싱글 스레드로 동작하는 등 성능의 제약이 많음.
- V8 으로 자바스크립트 엔진을 변경.
- MapReduce 를 비롯한 각종 통계성 함수를 복수로 실행하는 것이 가능.

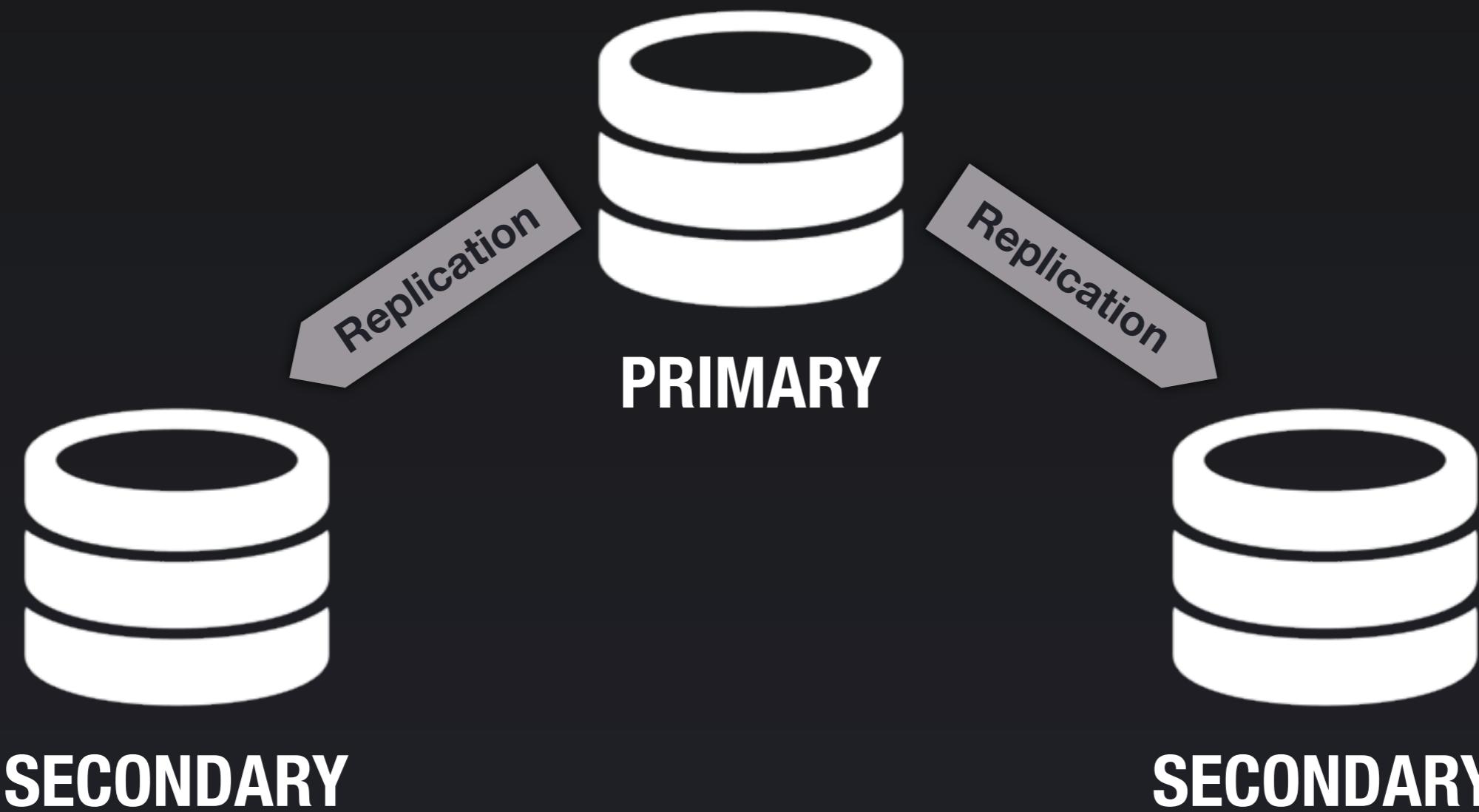
# EPISODE 5



몽고의 역습

MONGODB  
STRIKES BACK

# Production setup



# Production setup

- Dell R510 x 3
  - Intel(R) Xeon(R) L5520 @ 2.27GHz
  - 4GB RAM @ 1333MHz x 4 = 16GB
  - Perc H700 RAID 1+0
    - SATA 2TB x 6 ( 3 striped )



# Production setup

- 하드웨어는 그대로.
- MongoDB 버전업에 따라 콜렉션을 나눔.
  - 기존에는 하나의 콜렉션에 모두 넣기.
  - 로그를 남기는 앱과 로그 종류에 따라 서로 다른 콜렉션에 남기도록 변경.
- 용량을 절약하기 위해 키 이름도 줄임.

# Production setup

<b>timezone</b>	<b>tz</b>
<b>language</b>	<b>l</b>
<b>appid</b>	<b>ai</b>
<b>appversion</b>	<b>av</b>
<b>country</b>	<b>c</b>

# Production setup

로그 1개당 평균 용량

454 bytes

에서 337 bytes  
25.7% 절약

# Production setup

- 문서에 값으로 들어 있던 로그 주체와 이벤트 이름이 빠지면서 용량 절약.
- 원래는 조회를 위해 위 두 값을 포함하여 인덱스가 걸려 있었는데, 빠지면서 더 많은 인덱스를 효율적으로 구성 가능.

# Production setup

**App ID**  
64 bytes

**Event ID**  
16 bytes

**Payload**  
~350 bytes

# Production setup

**Payload**  
~350 bytes

# Production setup

<b>App ID</b> 64 bytes	<b>Event ID</b> 16 bytes	<b>Datetime</b> 8 bytes	<b>Timezone</b> 6 bytes
<b>App ID</b> 64 bytes	<b>Event ID</b> 16 bytes	<b>Datetime</b> 8 bytes	<b>Language</b> 2 bytes
<b>App ID</b> 64 bytes	<b>Event ID</b> 16 bytes	<b>Datetime</b> 8 bytes	<b>Country</b> 2 bytes

# Production setup

Datetime 8 bytes	Timezone 6 bytes
Datetime 8 bytes	Language 2 bytes
Datetime 8 bytes	Country 2 bytes

# Benchmark

- 특정 기간 동안에 수집된 로그에서, 국가별로 각 몇 회 유입되었는가를 출력.
- 기존 로그 형식 vs 신규 로그 형식
- Aggregate vs MapReduce

# Benchmark

unit = milliseconds	MapReduce	Aggregate
Old log	<b>159285</b>	<b>162257</b>
New log	<b>40402</b>	<b>6752</b>

# Benchmark

unit = seconds	MapReduce	Aggregate
Old log	<b>159.2</b>	<b>162.2</b>
New log	<b>40.4</b>	<b>6.7</b>

# Benchmark

Multiplier	MapReduce	Aggregate
Old log	24	24
New log	6	1

# Benchmark

Multiplier	MapReduce	Aggregate
1 day	24	24
1 month	6	1

# Benchmark

Multiplier	MapReduce	Aggregate
Old log	720	720
New log	6	1

# Benchmark

- In-memory 상황이 아니므로, 통계를 수행하는 과정에서 PageFault 발생.
- 전체 로그가 하나의 콜렉션에 들어있는 경우에는 PageFault도 많아질 뿐 아니라 비교해야 하는 조건도 많아짐.
- 이러한 상황에서는 인덱스에 따른 성능 향상이 거의 없음.

# Namespace problem

- MongoDB는 콜렉션 정보 저장을 위해 namespace 를 이용.
- namespace의 기본 크기는 16MB, 약 28,000개의 콜렉션과 인덱스 정보 저장.
- 콜렉션이 많아져 결국 초과하는 문제 발생.

# Namespace problem

```
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.87
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.88
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.89
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 13:46 sslog.9
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.90
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.91
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.92
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.93
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:00 sslog.94
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.95
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.96
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.97
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.98
-rw----- 1 mongodb nogroup 2.0G 2013-07-04 14:02 sslog.99
-rw----- 1 mongodb nogroup 1.0G 2013-07-04 14:02 sslog.ns
```

# Namespace problem

- **nssize** 값 변경으로 조정 가능.
- 이미 생성된 데이터베이스는 **repairDatabase** 수행을 필요.
- 콜렉션과 인덱스를 많이 사용할 것으로 예상되면 **nssize** 를 미리 늘려놓는 것이 중요.
- 용량만 늘어날 뿐, 성능에는 영향 없음.

# EPISODE 6



남은 이야기

MONGODB  
STRIKES BACK

# Push messages

- MongoDB 로 로그만 쌓는 것은 아님.
- Push token / Registration ID 보관을 위한 서비스용 데이터베이스로도 활용.

# Reinventing the wheel

- 2010년에는 적절한 푸시 메세지 관련 솔루션이나 서비스가 없었다 / 찾질 못했다.
- 일단 EasyAPNS 을 사용하고, 가내수공업으로 밑도 끝도 없이 뜯어 고치기 시작.

# Storing tokens

- <http://www.easyapns.com/>
  - PHP + MySQL
- 당장 iOS 제품에 간단히 적용 가능.

# Storing tokens

- <http://www.easyapns.com/>

- PHP + MySQL
- 당장 iOS 제품에



It works!

간단히 적용 가능

# Storing tokens

- EasyAPNS 기본 상태로는,
  - 토큰 및 기타 정보 변경시 오버헤드.
  - 서버와 커넥션을 매번 생성하는 문제.
  - 안드로이드(C2DM, GCM)에 대응은 어떻게 하지?

# Storing tokens

- 그래서,
  - MyISAM에서 InnoDB로 변경.
  - 한 프로세스 내에서는 커넥션 풀 사용.
  - C2DM, GCM의 Registration ID를 받아 처리할 수 있도록 수정.

# Storing tokens

- 수집한 토큰이 2천만 건이 넘어가니,
- 아무리 고쳐도 **INSERT ON DUPLICATE KEY UPDATE ...** 구문 최적화가 난제.
- MongoDB 에도 Unique Index 와 Upsert 가 있으니까 시도해볼까?

# Storing tokens

- 토큰 업데이트 요청을 받으면 일단 Redis에 보관하고,
- 주기적으로 MongoDB 에 Upsert.

# Storing tokens

- 토큰 업데이트 요청을 받으면 일단 Redis  
에 보관하고,
- 주기적으로 MongoDB 에 Upsert

It works!

# Storing tokens

- 특정 시간에 예약해서 단체 발송하는 경우에, 동시에 사용자가 몰려 들어오는 특성이 있어 Redis 를 버퍼로 사용.
- 2천 5백만개 가량의 토큰을 관리하는데 약 30GB 사용.
  - 국가, 언어, 타임존 등 필요 정보 포함.

# New hardware



# Reinventing the wheel

- 2013년 5월 기준 3백 만원 정도의 장비로,
- 1억 건 정도의 토큰 관리가 가능하며,
- 월 3천만건 이상의 메세지를 보내는 중.
- Urban Airship 기준으로 약 \$30,000.



# Wrap-up

- MongoDB 1.6에서 현재의 2.4 까지,
  - 동시성 문제가 많이 해결됨.
  - 요구에 따른 다양한 구성 가능.
  - 보다 빠르고 쉽게 데이터 분석 가능.

# Wrap-up

- MongoDB의 성능 향상을 위해서는,
  - 충분한 메모리와 SSD 사용은 당연.
  - PageFault 를 최소화 할 수 있는 설계.
  - 상황에 맞는 스키마 설계와 적합한 통계 명령어 사용.

그래서 누군가  
쓸만하냐 물는다면

네



# **THE MONGODB STRIKES BACK**

**FIN.**

# License

- Images, fonts and contents of **STARWARS**
  - Lucasfilm Ltd.
- Image of hardware
  - Dell Inc.
- Icons from **The Noun Project**
  - Dmitry Baranovskiy
- App icons
  - SMARTSTUDY, Samsung Publishing Ltd.