Roger Bodamer
roger@10gen.com
@rogerb

10gen mongoDB

Thoughts on Transaction and Consistency Models

10gen mongoDB

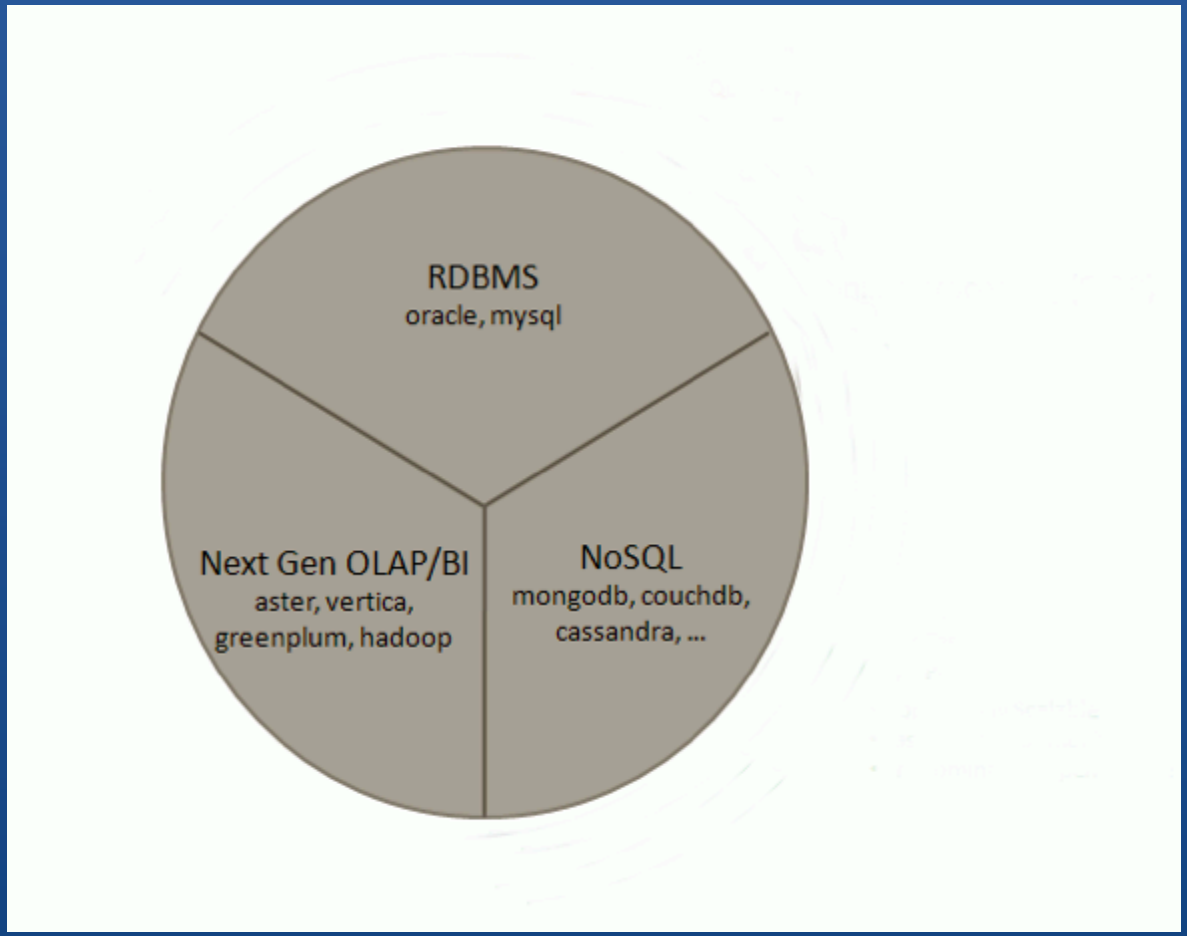
10gen mongoDB

10gen mongoDB


RDBMS
(Oracle, MySQL)
New Gen.
OLAP
(vertica, aster,
greenplum)
Non-relational
Operational
Stores
("NoSQL")
10gen
mongoDB

The database world is changing

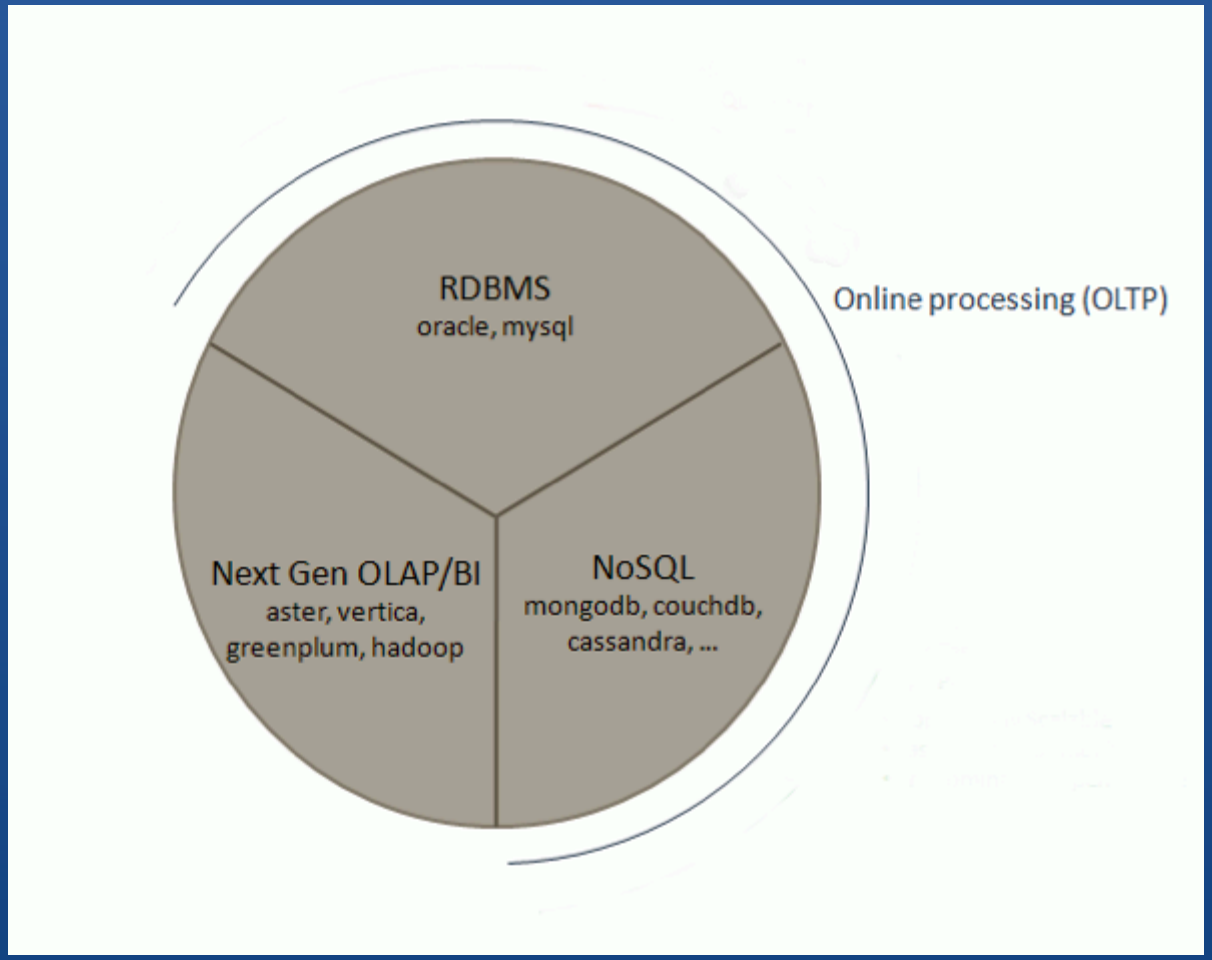
Document Datastores, Key Value, Graph databases



10gen mongoDB

The database world is changing

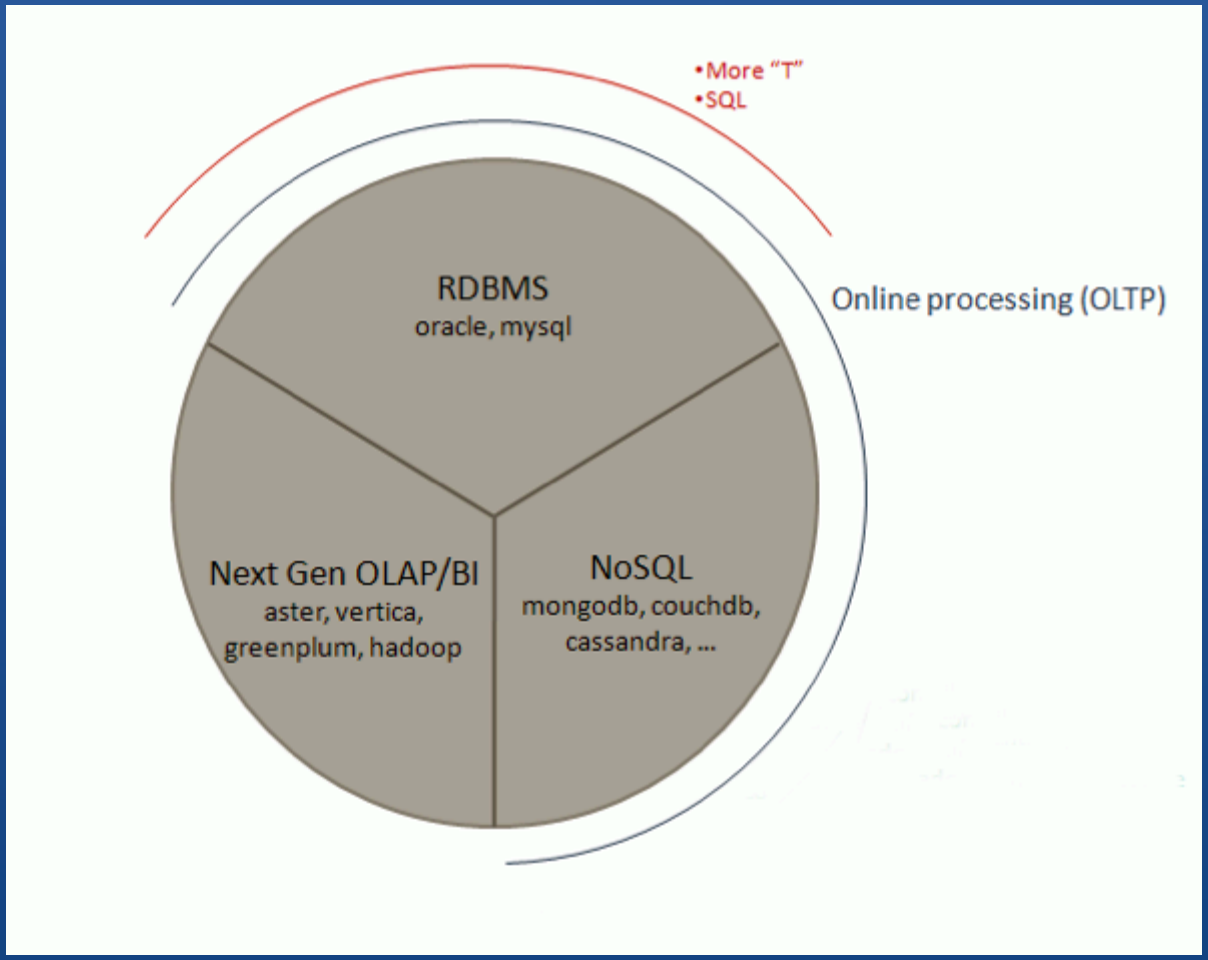
Transactional model



10gen mongoDB

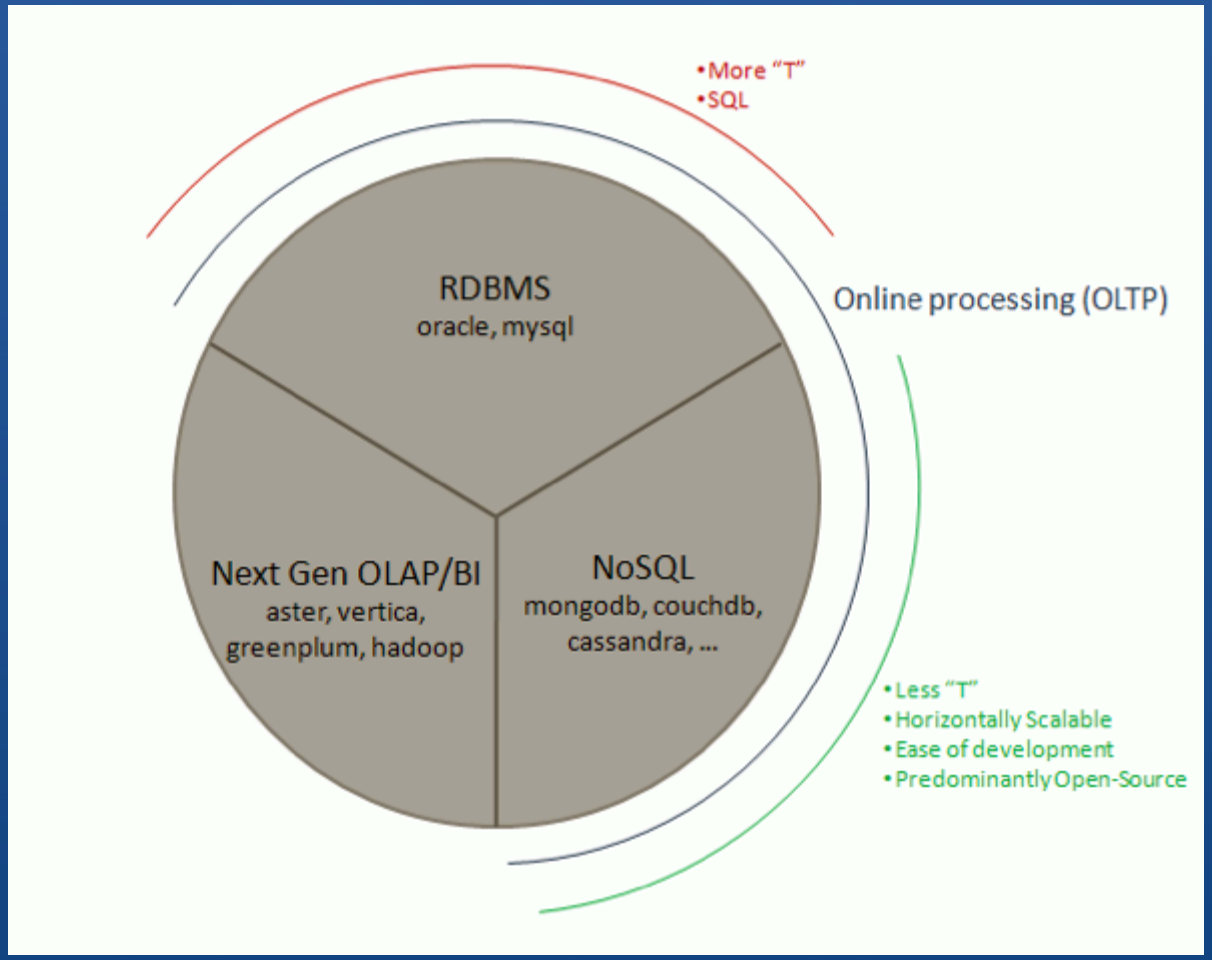
The database world is changing

Full Acid

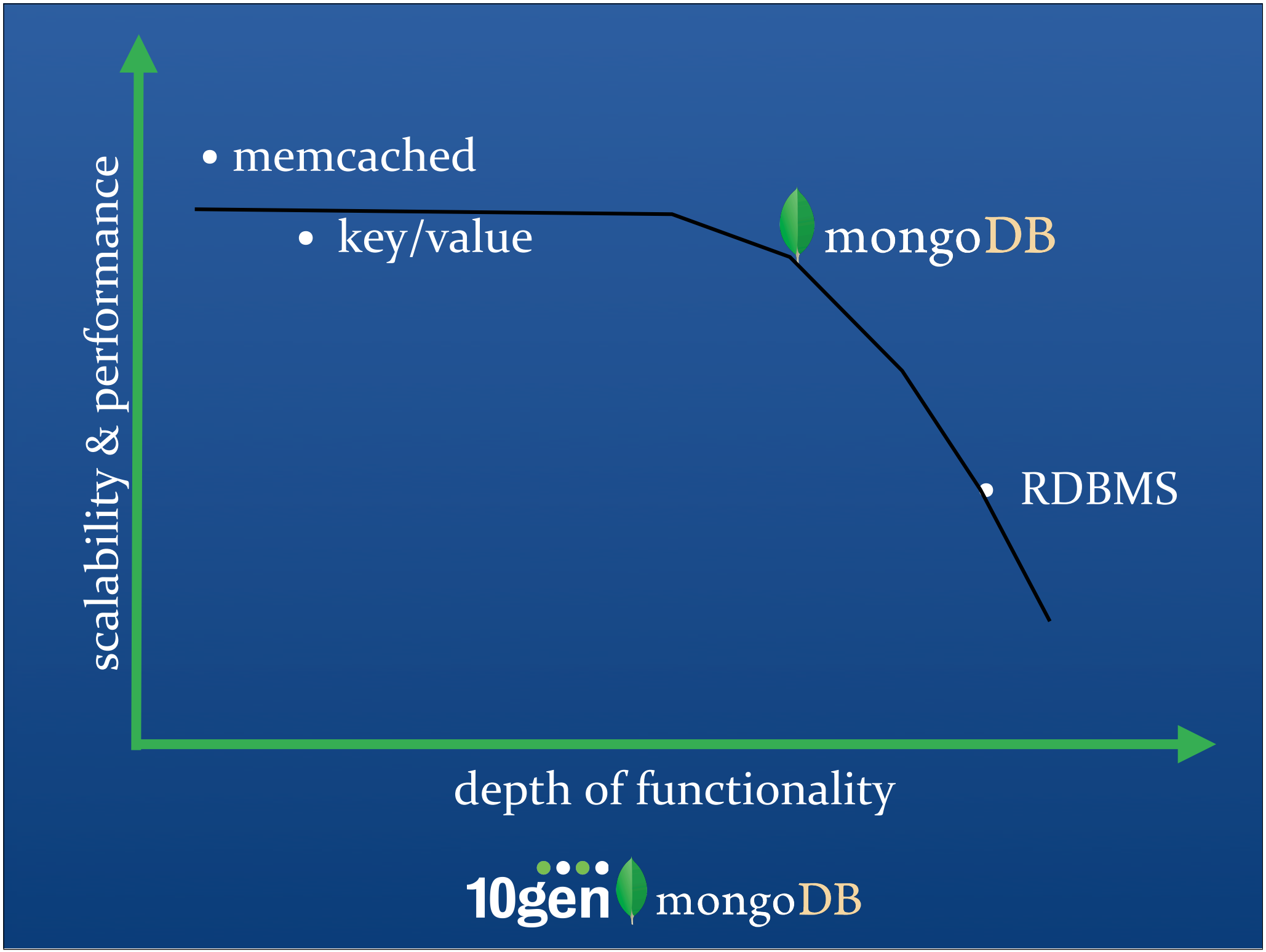


10gen mongoDB

The database world is changing



10gen mongoDB


scalability & performance
• memcached
• key/value
mongoDB
• RDBMS
depth of functionality
10gen mongoDB

CAP

It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- *Availability*
- *Atomic consistency in all fair executions (including those in which messages are lost).*

CAP

It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:

- *Availability*
- *Atomic consistency in all fair executions (including those in which messages are lost).*

Or: If the network is broken, your database won't work

But we get to define "won't work"

Consistency Models - CAP

- Choices are Available-P (AP) or Consistent-P (CP)
- Write Availability, not Read Availability, is the Main Question

- It's not all about CAP

  Scale, Reduced latency:

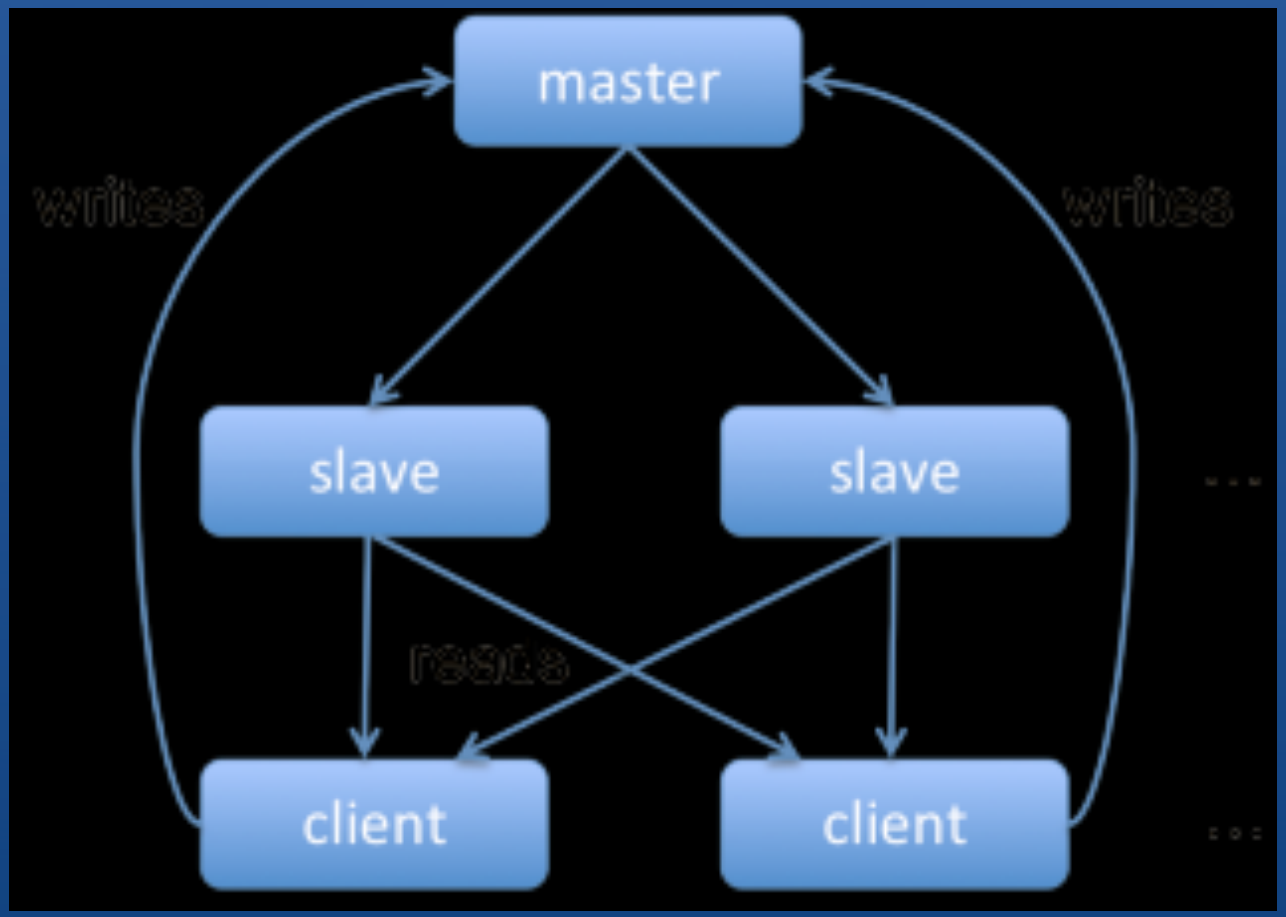
  - Multi data center
  - Speed
  - Even load distribution

Examples of Eventually Consistent Systems

- Eventual Consistency:
  - "The storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value"

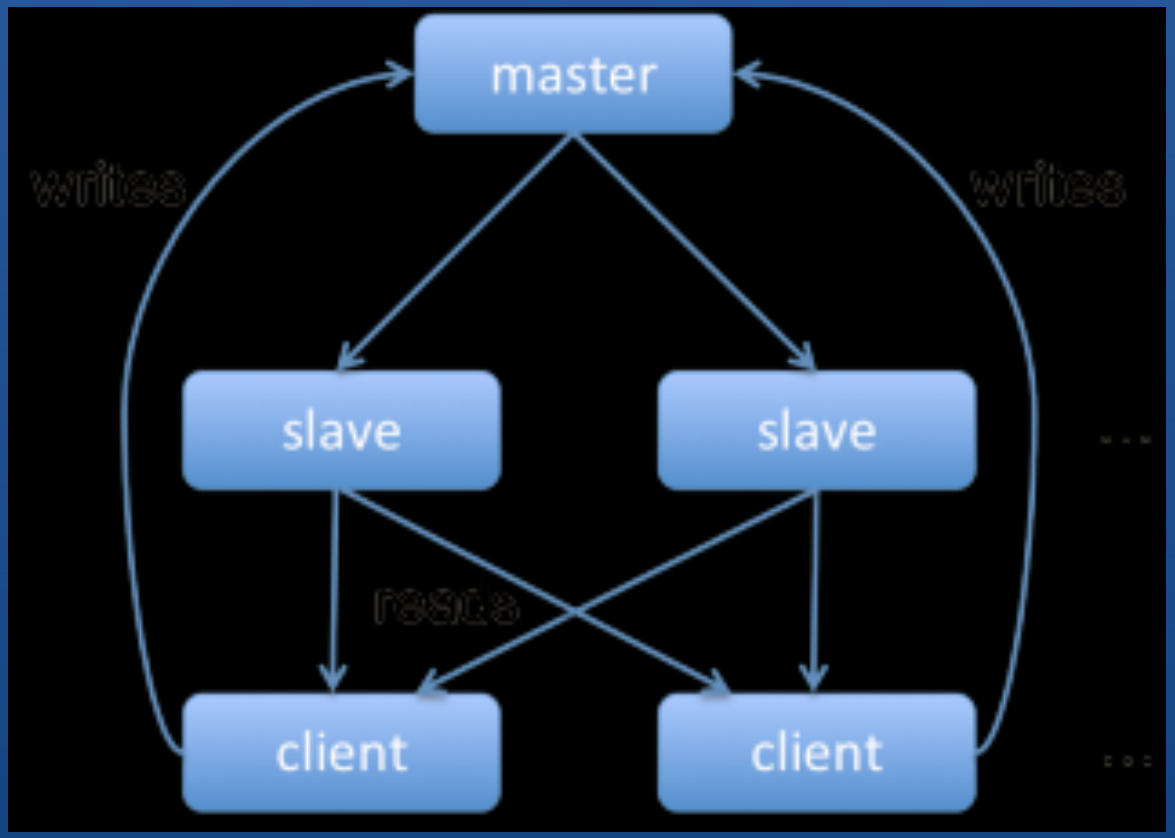
- Examples:
  - DNS
  - Async replication (RDBMS, MongoDB)
  - Memcached (TTL cache)

Eventual Consistency



10gen mongoDB

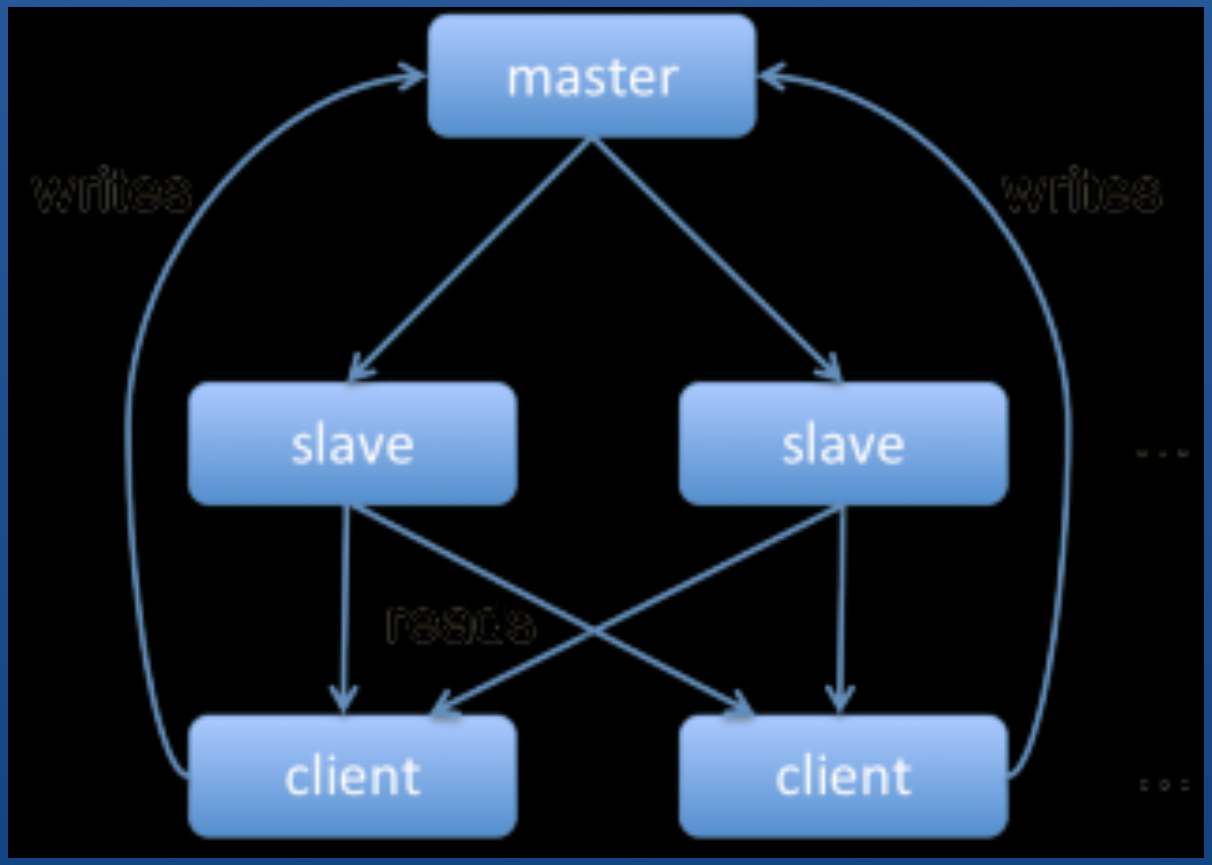
Eventual Consistency



Read(x) : 1, 2, 2, 4, 4, 4, 4 …

10gen mongoDB

Could we get this?

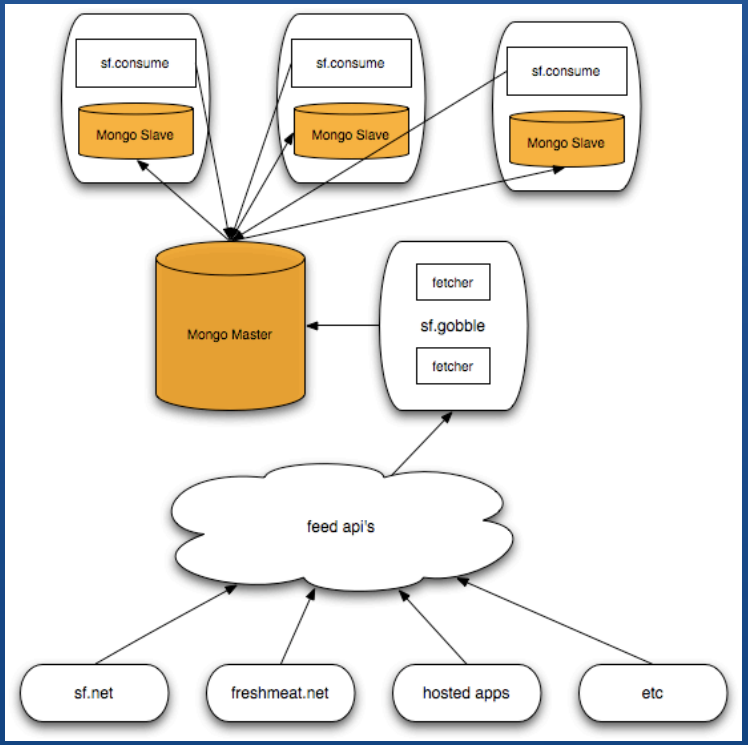


Read(x) : 1, 2, 1, 4, 2, 4, 4, 4 …

10gen mongoDB

Monotonic read consistency

Prevent seeing writes out of order



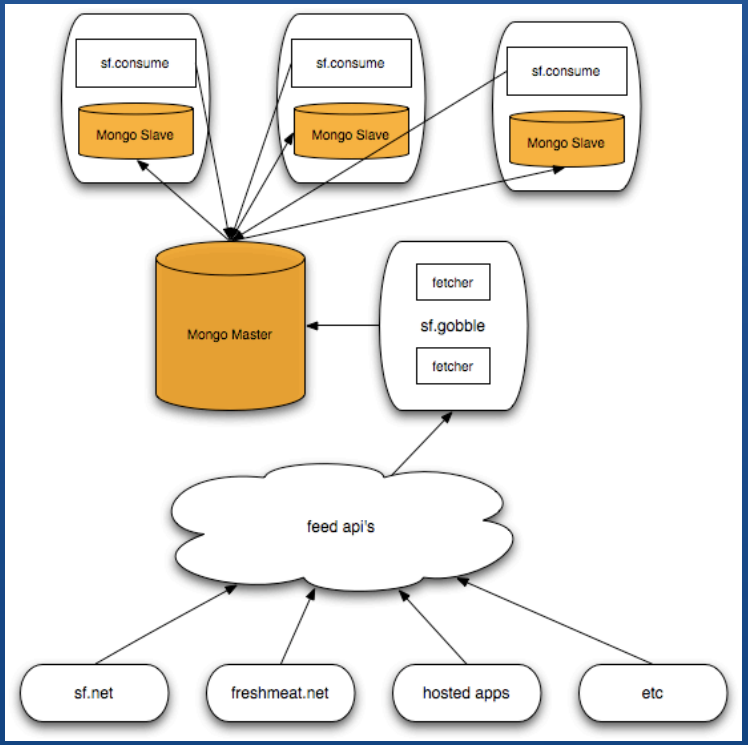
- Appserver and slave on same box
- Appserver only reads from Slave

- Eventually consistent
- Guaranteed to see reads in order

10gen mongoDB

Monotonic read consistency

Prevent seeing writes out of order



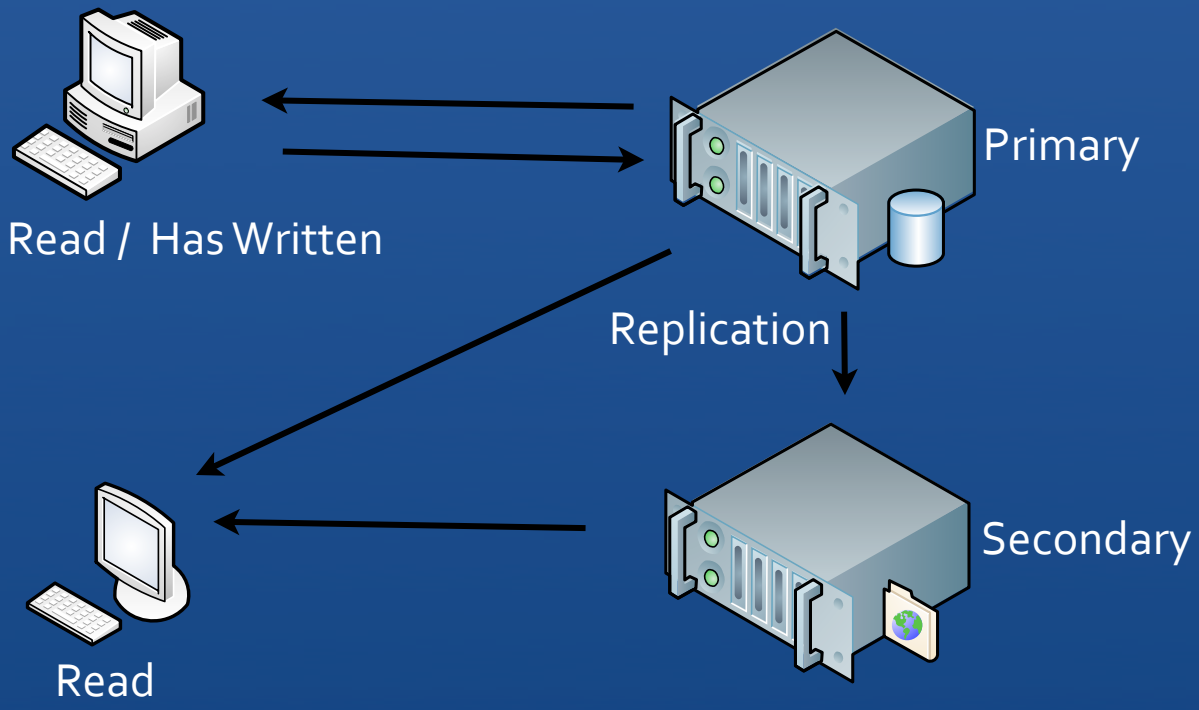
- Appserver and slave on same box
- Appserver only reads from Slave

- Eventually consistent
- Guaranteed to see reads in order

- Failover ?

10gen mongoDB

RYOW Consistency



- Read Your Own Writes
- Eventually consistent for readers
- Immediately consistent for writers

10gen mongoDB

Amazon Dynamo

- R: # of servers to read from
- W: # servers to get response from
- N: Replication factor
  - R+W>N has nice properties

Example

Example 1

R + W <= N

R = 1
W = 1
N = 5

Possibly Stale data
Higher availability

Example

Example 1

R + W <= N

R = 1
W = 1
N = 5

Possibly Stale data
Higher availability

Example 2

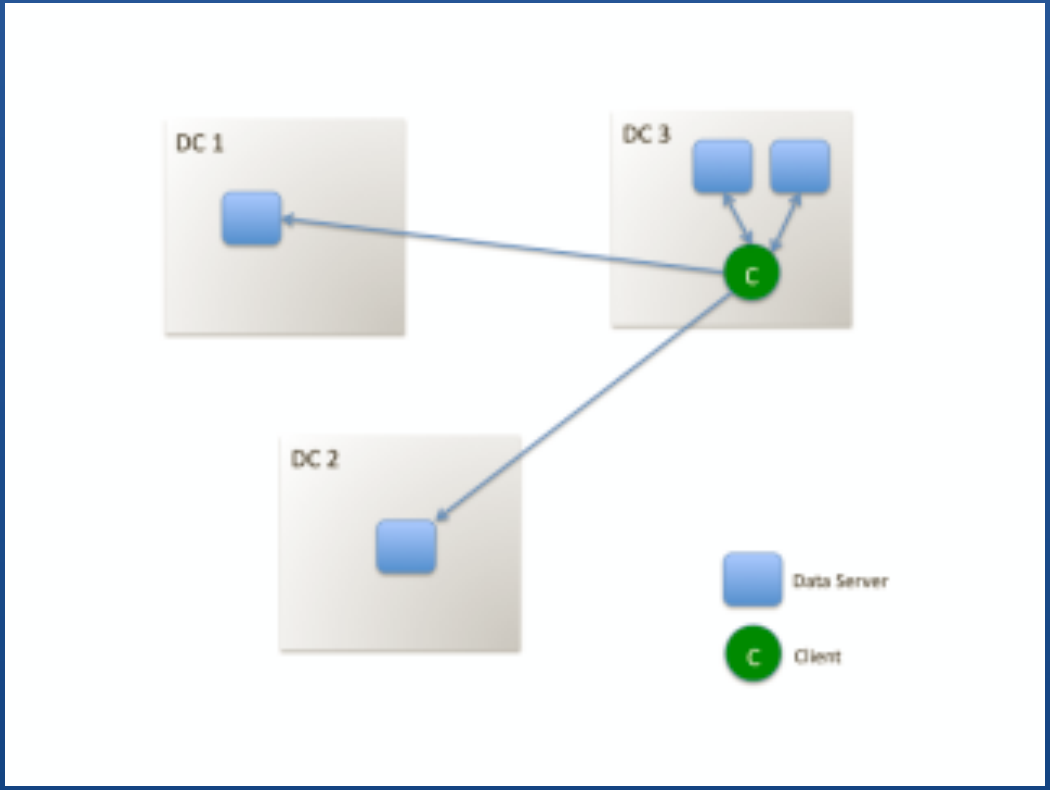
R + W > N

R = 2
W = 1
N = 2

R = 1
W = 2
N = 2

Consistent data

R + W > N



If R+W > N, we can't have both fast local reads and writes at the same time if all the data centers are equal peers

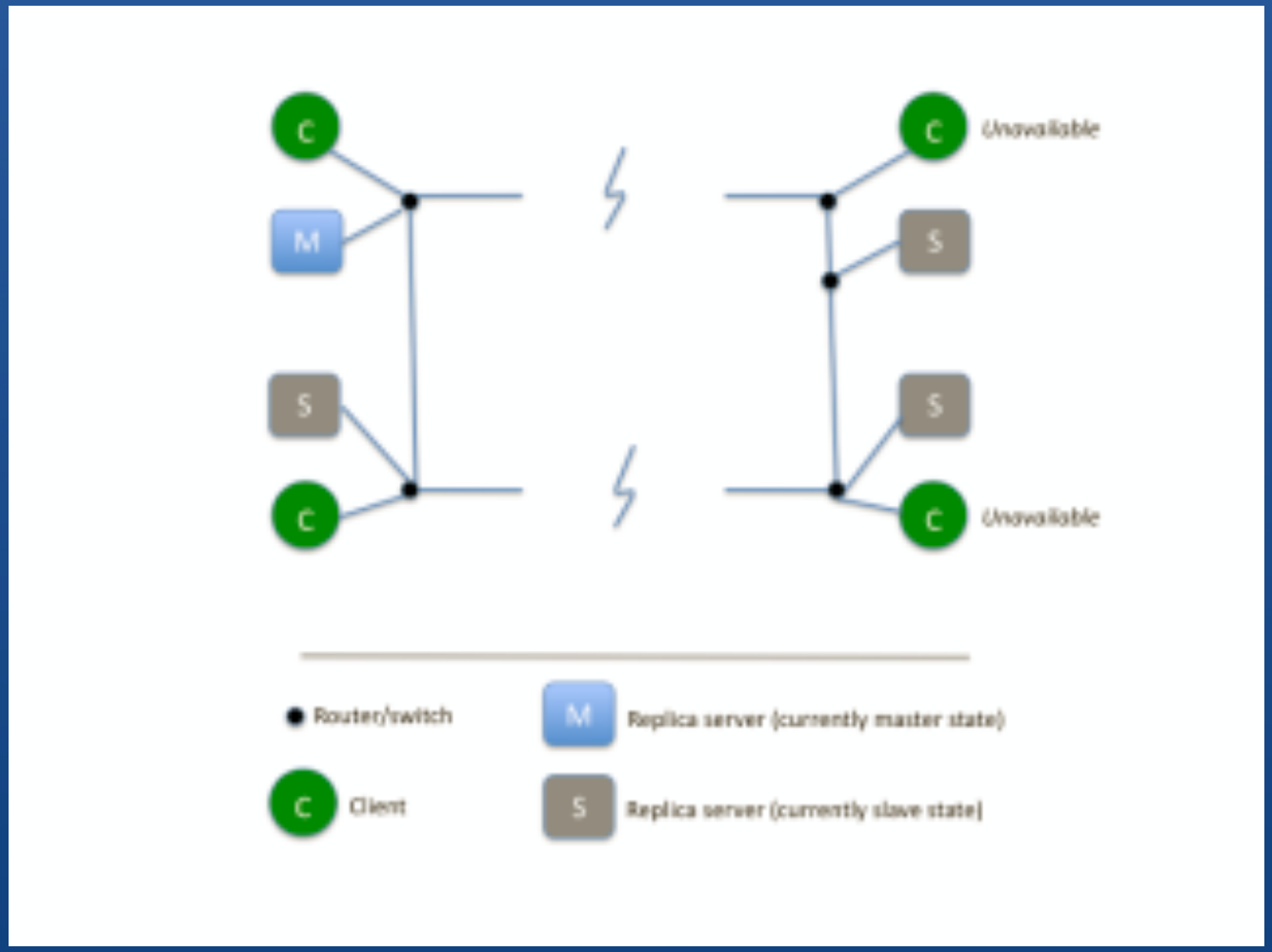
Consistency levels

- Strong: W + R > N
- Weak / Eventual : W + R <= N
- Optimized Read: R=1, W=N
- Optimized Write: W=1, R=N

Multi Datacenter Strategies

- DR
  - Failover
- Single Region, Multi Datacenter
  - Useful for Strong or Eventual Consistency
- Local Reads, Remote Writes
  - All writes to master on WAN, EC on Slaves
- Intelligent Homing
  - Master copy close to user location

Network Partitions



10gen mongoDB

Network Write Possibilities

- Deny all writes
  - Still Read fully consistent data
  - Give up write Availability

Network Write Possibilities

- Deny all writes
  - Still Read fully consistent data
  - Give up write Availability
- Allow writes on one side
  - Failover
  - Possible to allow reads on other side

Network Write Possibilities

- Deny all writes
  - Still Read fully consistent data
  - Give up write Availability
- Allow writes on one side
  - Failover
  - Possible to allow reads on other side
- Allow writes on both sides
  - Available
  - Give up consistency

Multiple Writer Strategies

- Last one wins
  - Use Vector clocks to decide latest

- Insert

  Inserts often really means

  if (!exists(x)) then set(x)

  exists is hard to implement in eventually consistent systems

Multiple Writer Strategies

- Delete

```
op1: set( { _id : 'joe', age : 40 } }
op2: delete( { _id : 'joe' } )
op3: set( { _id : 'joe', age : 33 } )
```

- Consider switching 2 and 3
- Tombstone: Remember delete, and apply last-operation-wins

- Update

update users set age=40 where _id='joe'

However:

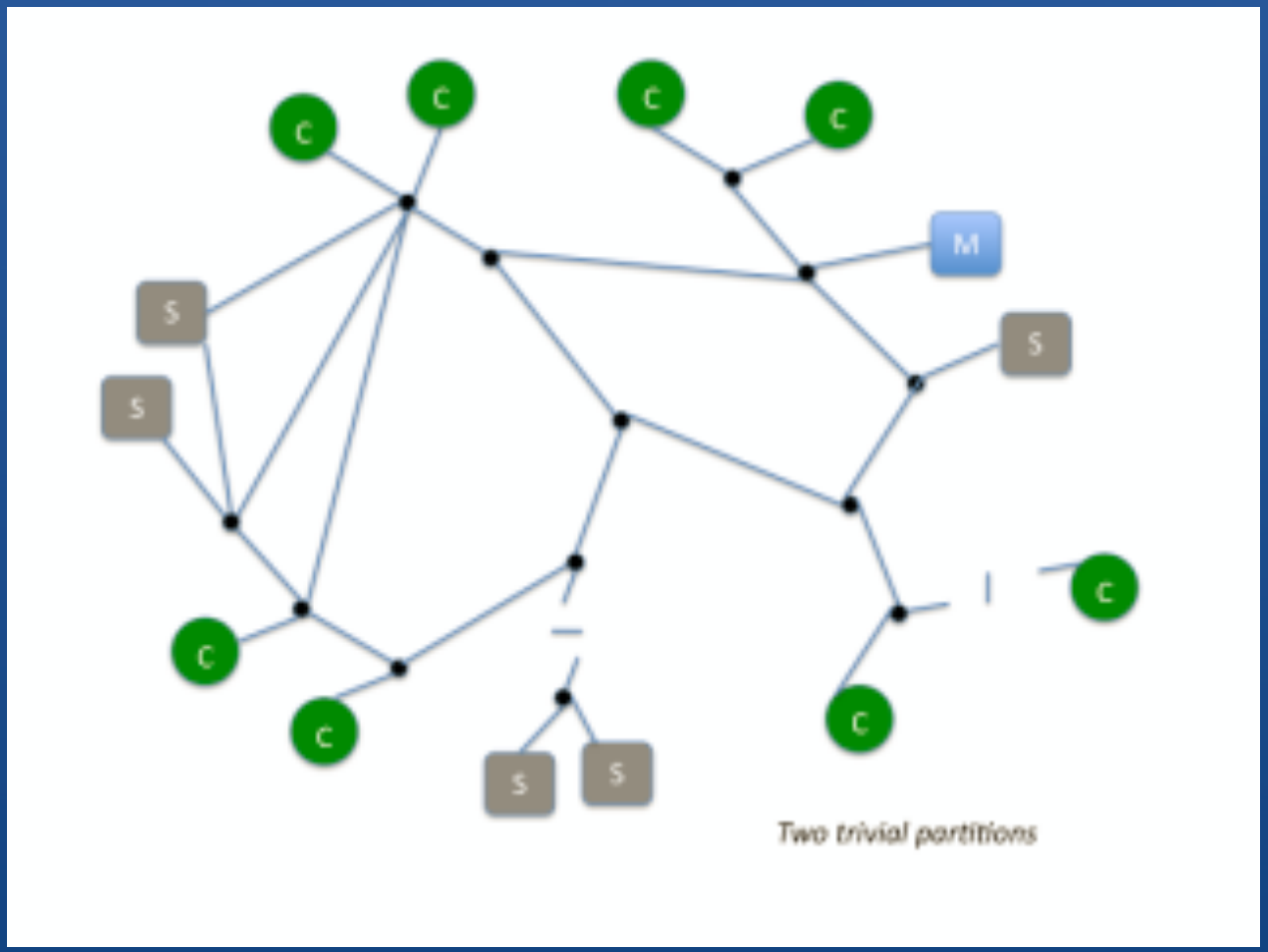
```
op1: update users set age=40 where _id='joe'
op2: update users set state='ca' where _id='joe'
```

Multiple Writer Strategies

- Programatic Merge
  - Store operations, instead of state
  - Replay operations
  - Did you get the last operation ?
  - Not immediate

- Communtative operations
  - Conflict free?
  - Fold-able
  - Example: add, increment, decrement

Trivial Network Partitions



10gen mongoDB

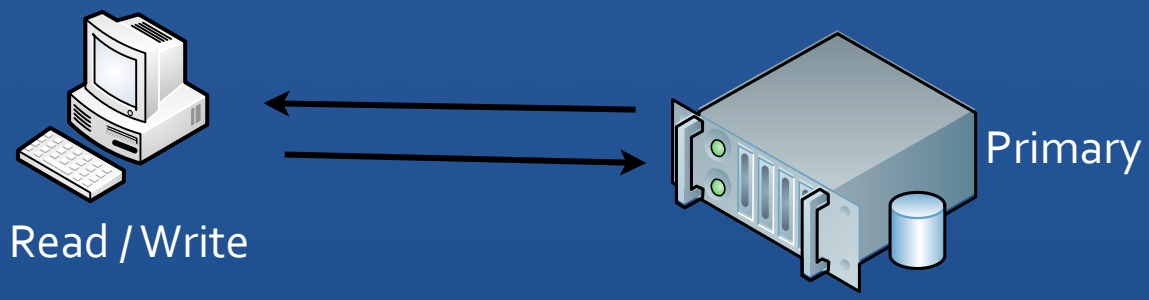
MongoDB Options

MongoDB Transaction Support

- Single Master / Sharded

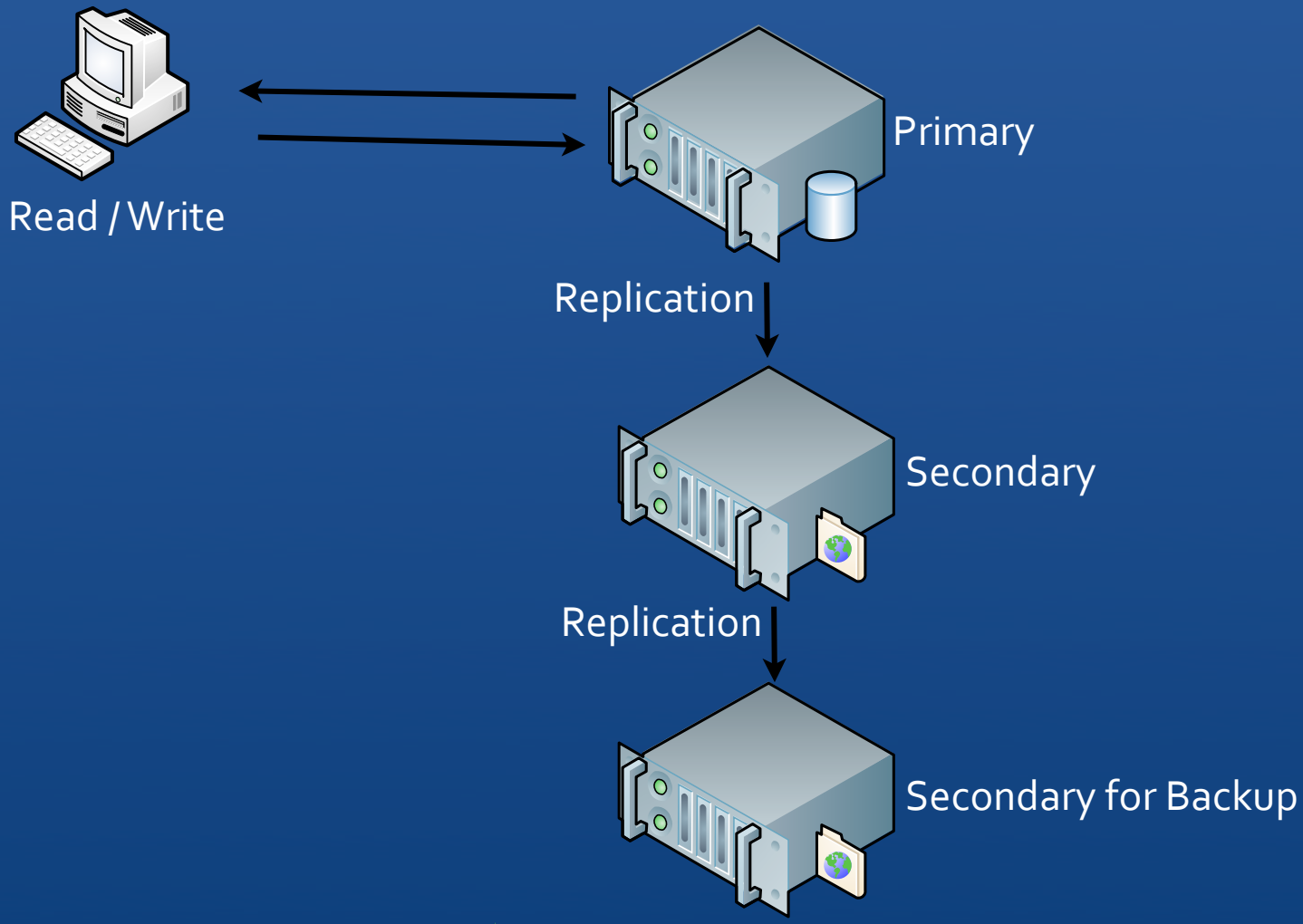
- MongoDB Supports Atomic Operations on Single Documents
  - But not Rollback
- $ operators
  - $set, $unset, $inc, $push, $pushall, $pull, $pullall
- Update if current
  - find followed by update
- Find and modify

MongoDB



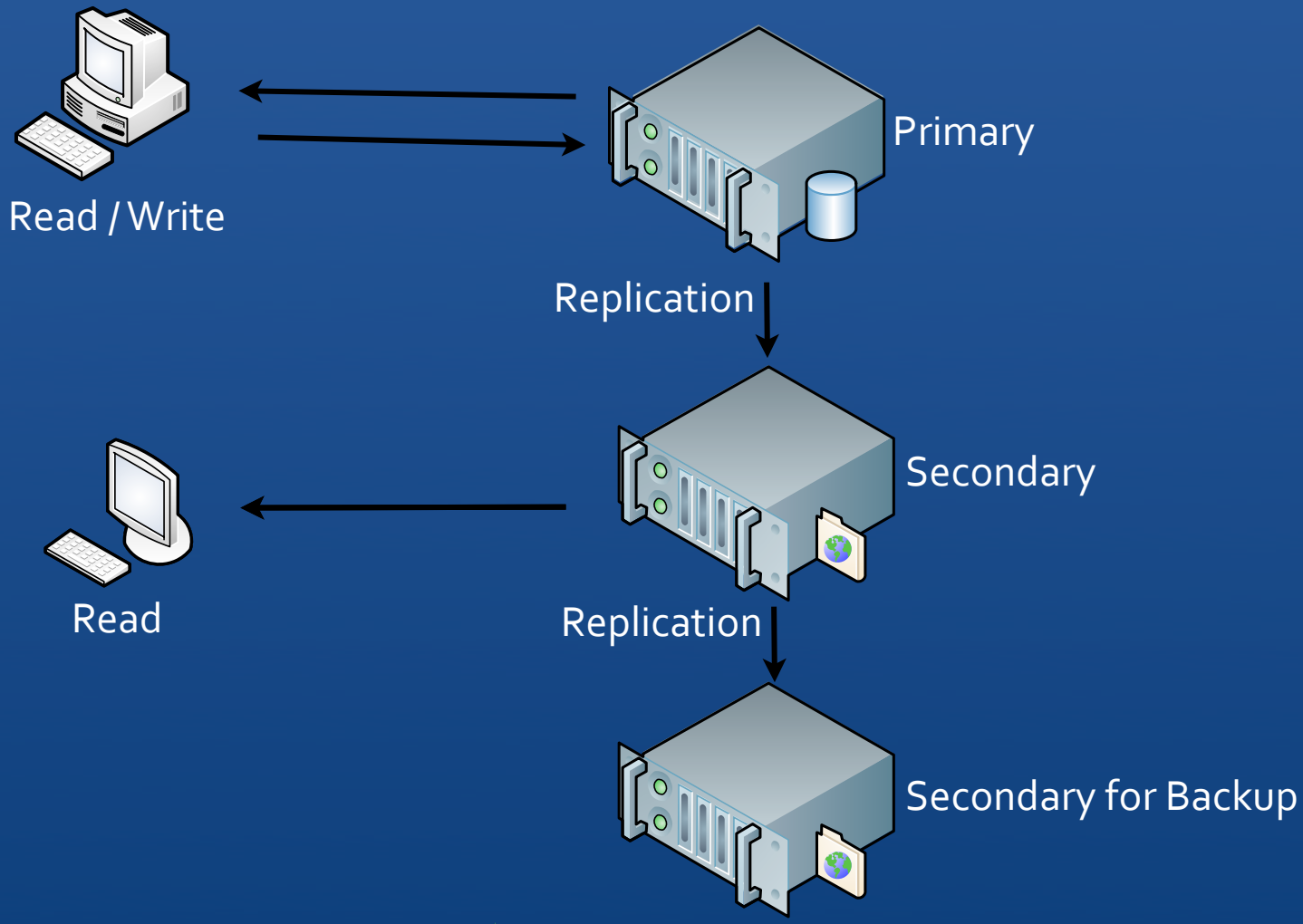
10gen mongoDB

MongoDB



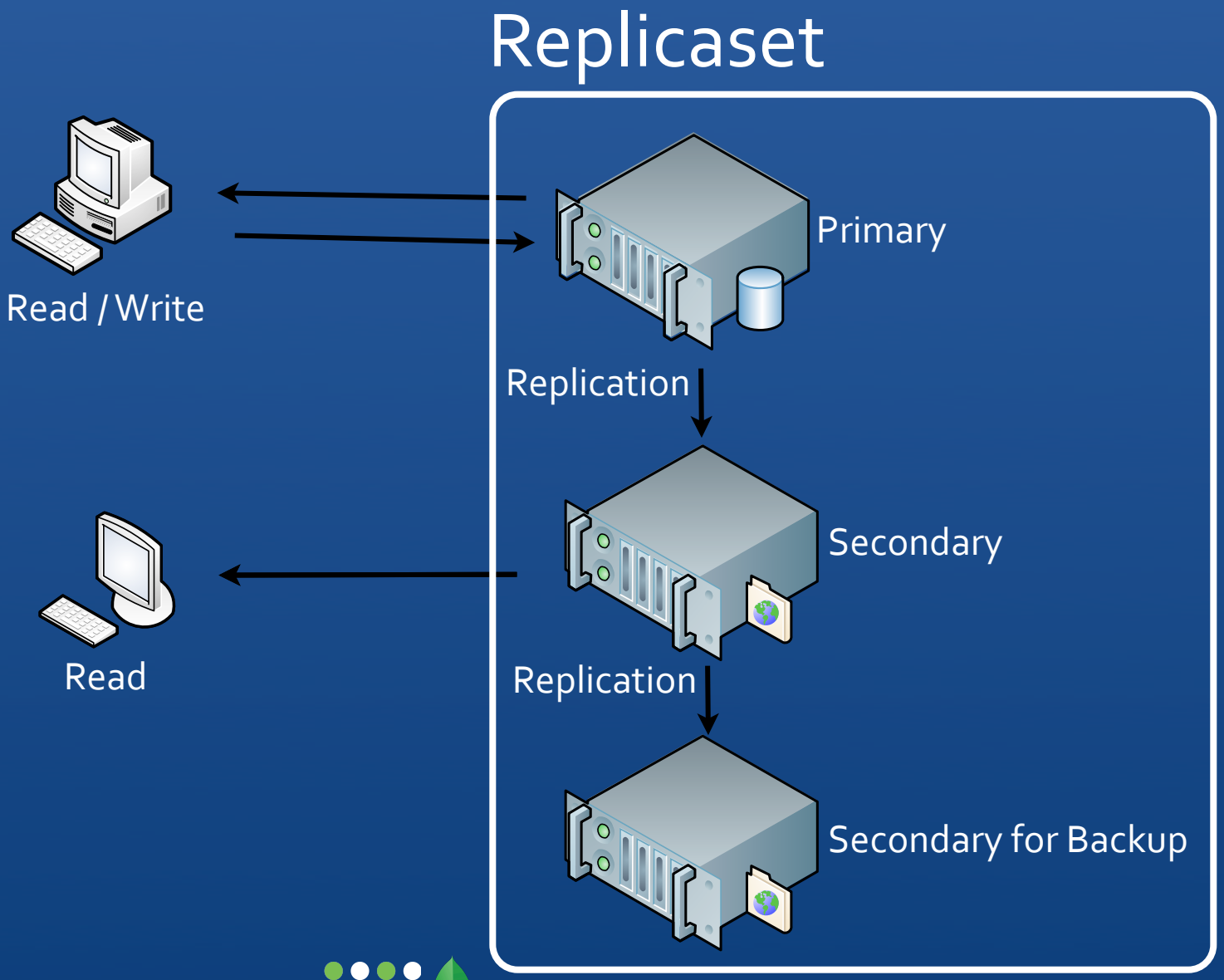
10gen mongoDB

MongoDB



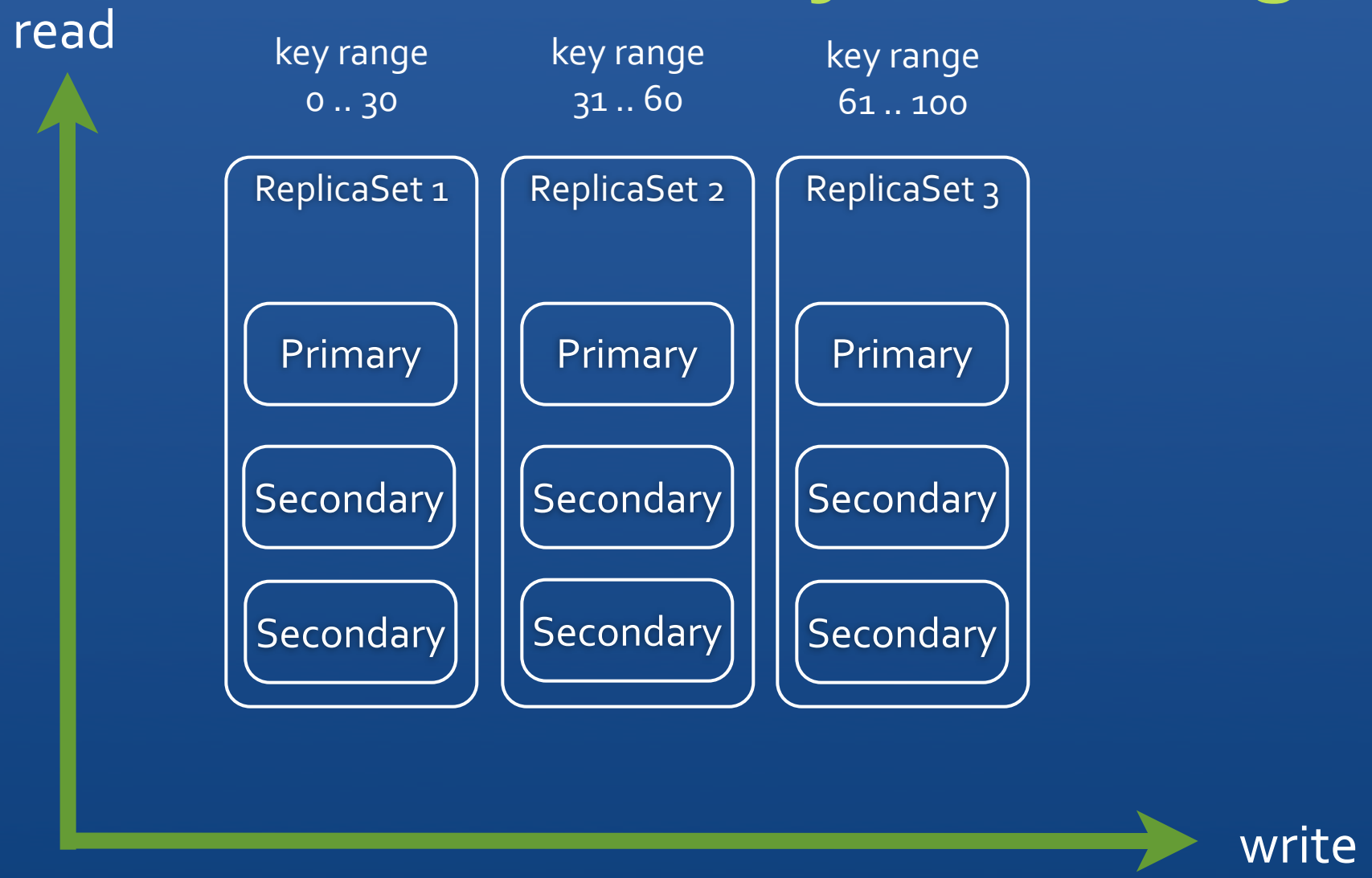
10gen mongoDB

MongoDB

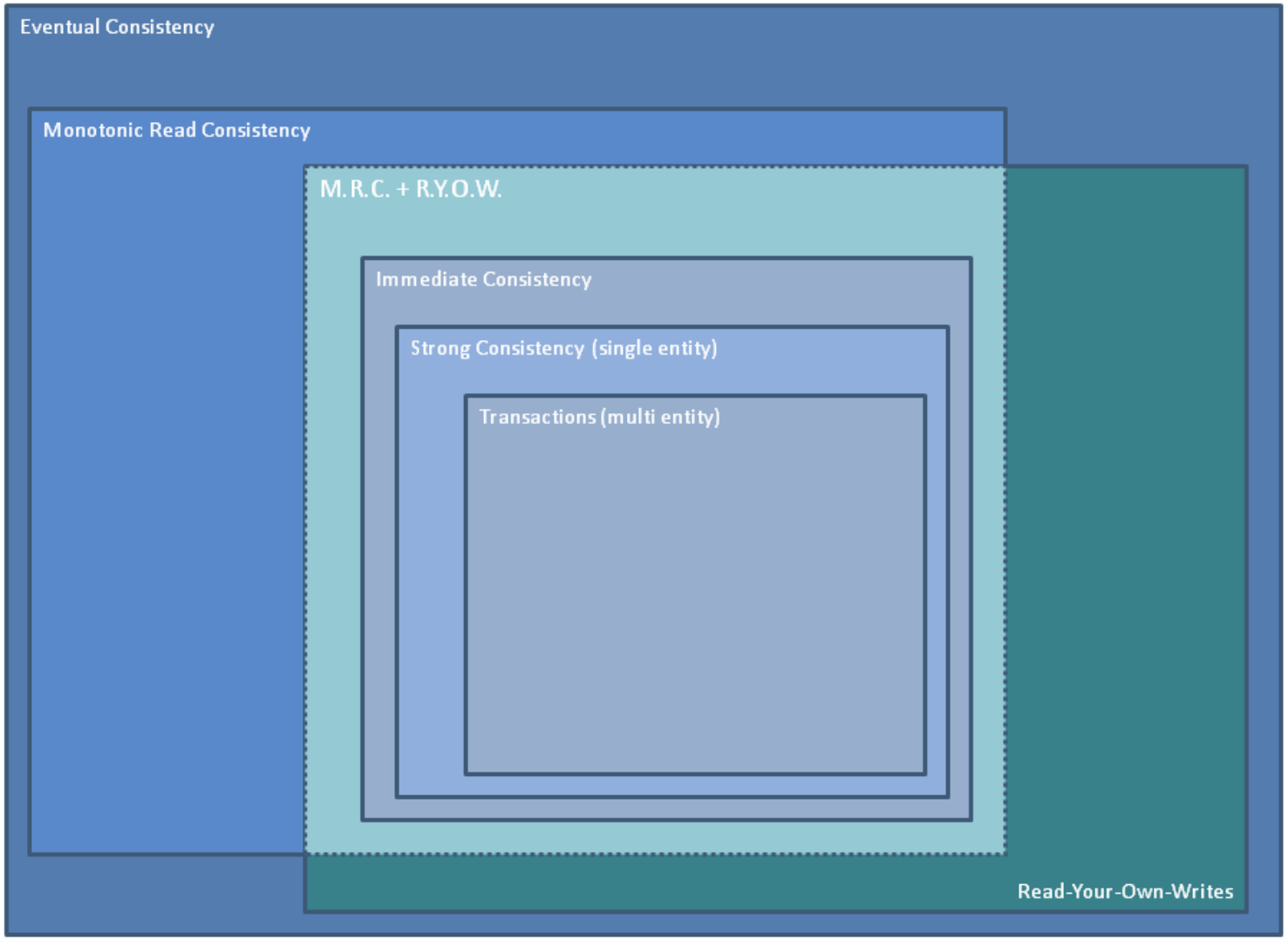


10gen mongoDB

Write Scalability: Sharding



10gen mongoDB



10gen mongoDB

Sometimes we need global state / more consistency

- Unique key constraints
  - User registration
- ACL changes

Could it be the case that…

uptime(CP + average developer)

>=

uptime(AP + average developer)

where uptime:= system is up and non-buggy?

Thank You :-)
@rogerb

download at mongodb.org

conferences, appearances, and meetups
http://www.10gen.com/events



Facebook | Twitter | LinkedIn
http://bit.ly/mongofb | @mongodb | http://linkd.in/joinmongo