UNIVERSITY OF TEXAS AT ARLINGTON

Introduction to

Adopted from slides and/or materials from mongoDB.org, W. Bo, R. Gabor, C. Scheich and K. Avery

Outline

University of Texas at Arlington

- Background
- Data model
- Comparison to RDBMS
- *CRUD* operations
- Replication
- Sharding

Background

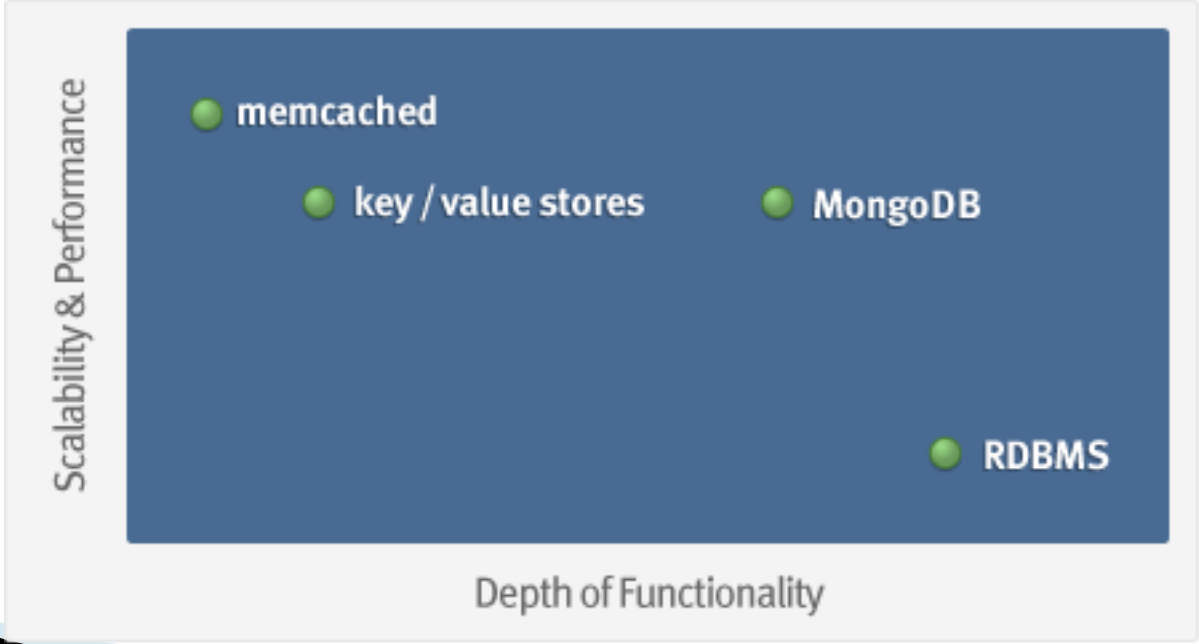
University of Texas at Arlington

- stands for humongous
- created by10gen
- open source (Implemented in C++)
- document-oriented database designed with
  - 1) scalability
  - 2) developer agility in mind
- Instead of storing data in tables and rows as we would do with a relational database, in MongoDB we store BSON (binary representation of JSON) documents with dynamic schemas (schema-less).

Background

University of Texas at Arlington

- Goal:
  - bridge the gap between key-value stores (which are fast and scalable) and relational databases (which have rich functionality).



Background

University of Texas at Arlington

- Key features:
  - scale horizontally over commodity hardware
  - support RDBMS features:
    - Ad hoc queries
    - Fully featured indexes
      - Single-field
      - Compound
      - Multi-key
      - Geospatial
      - Text
    - Aggregation operations (through map-reduce)

Data model

University of Texas at Arlington

- DB ← Collections ← Documents
- Collection = set of "related" documents sharing common indexes
- Primary key *(object id)* is automatically created and indexed for a document
  - Help narrow search span; otherwise mongoDB needs to scan all documents in every shards

Data model

University of Texas at Arlington

- Schema design:
  - Embedding
    - nesting of objects and arrays inside a BSON document (pre-joined) → fast
  - Link (referencing)
    - references between documents → requires follow-up query
- Principle:
  - embedding when you can, link when you must

Comparison to RDBMS

University of Texas at Arlington

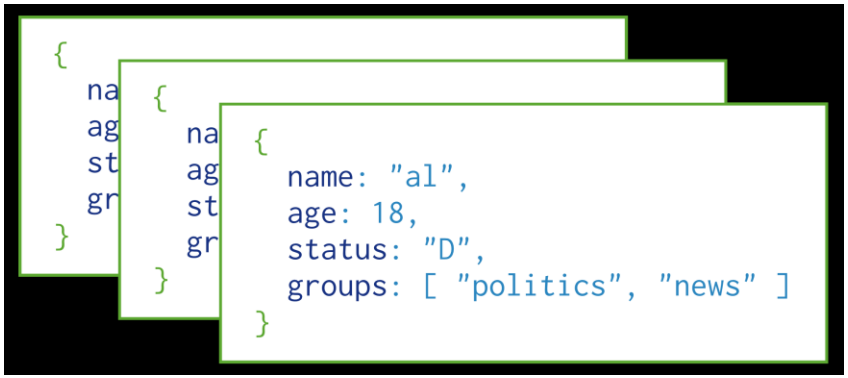
RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (BSON → JSON)
Column	Field
Index	Index
Join	Embedding/Referencing
Foreign Key	Referencing
Primary Key	ObjectID

CRUD operations

University of Texas at Arlington

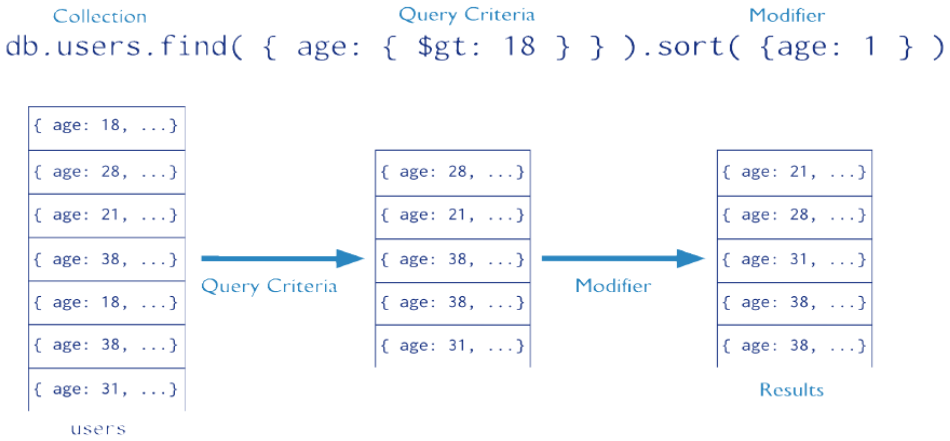
- **CRUD**
  - stands for **C**reate, **R**ead, **U**pdate, and **D**elete
  - used for reading and manipulating data
- Example: JSON documents and collection

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



CRUD operations

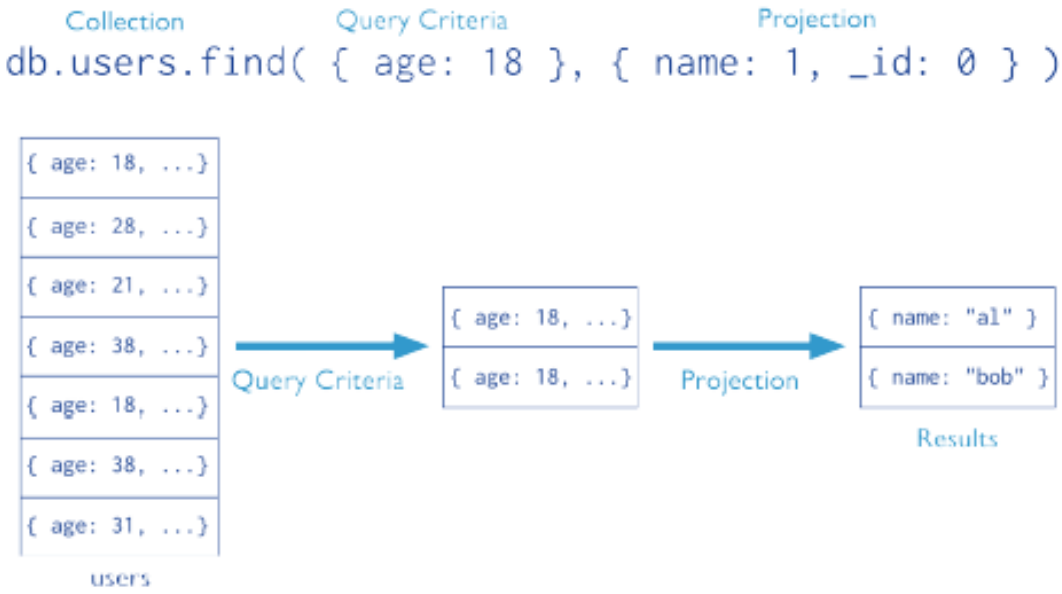
- target a specific collection at a time
  - **NOT** support JOIN operations



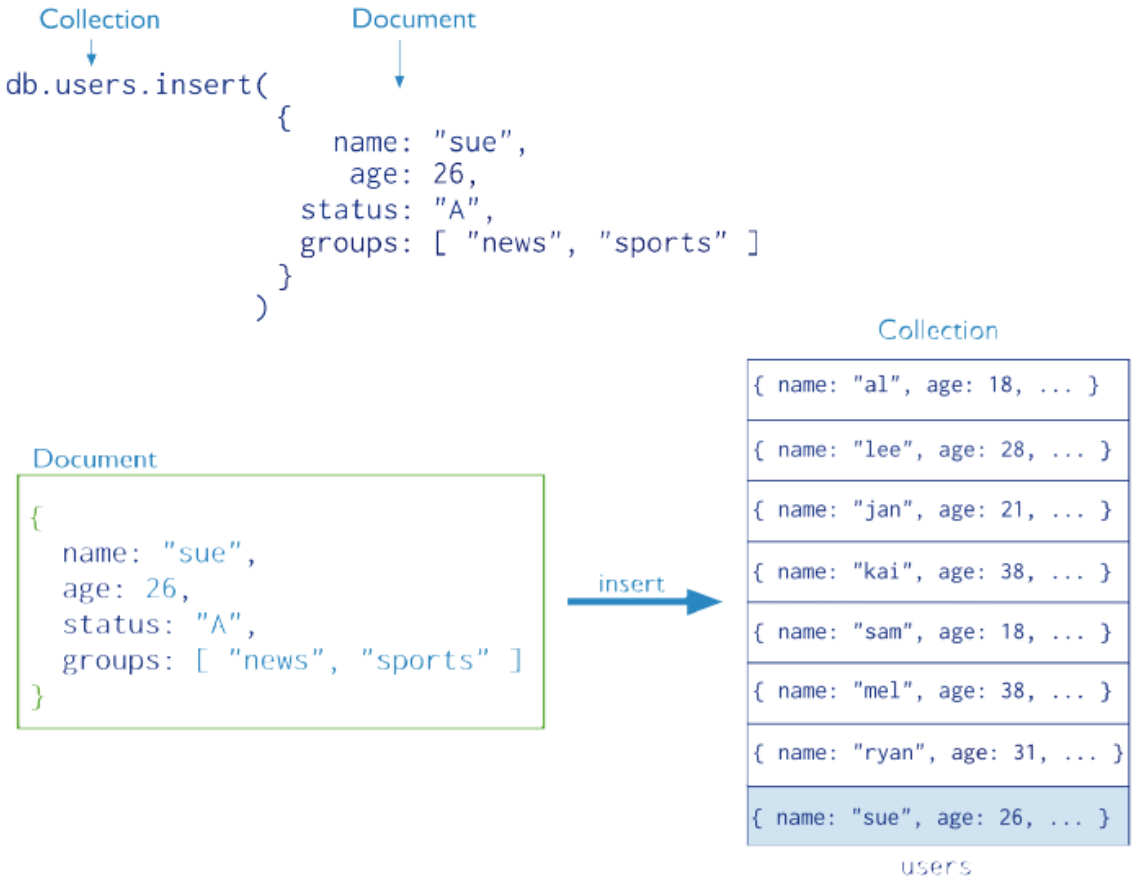
- Note:
  - Read op.: users have options to choose preferred member in a replica set; otherwise primary
  - Write op.: users specify consistency level (*write concern*)

CRUD operations

- More example:



CRUD operations



University of Texas at Arlington

CRUD operations

University of Texas at Arlington

```
SELECT _id, name, address   <-- projection
FROM   users                <-- table
WHERE  age > 18             <-- select criteria
LIMIT  5                    <-- cursor modifier
```

```
db.users.find(
   { age: { $gt: 18 } },    <-- query criteria
   { name: 1, address: 1 }  <-- projection
).limit(5)                  <-- cursor modifier
```

(collection)

- Results always include object id *(_id)* unless explicitly specify (_id:0)

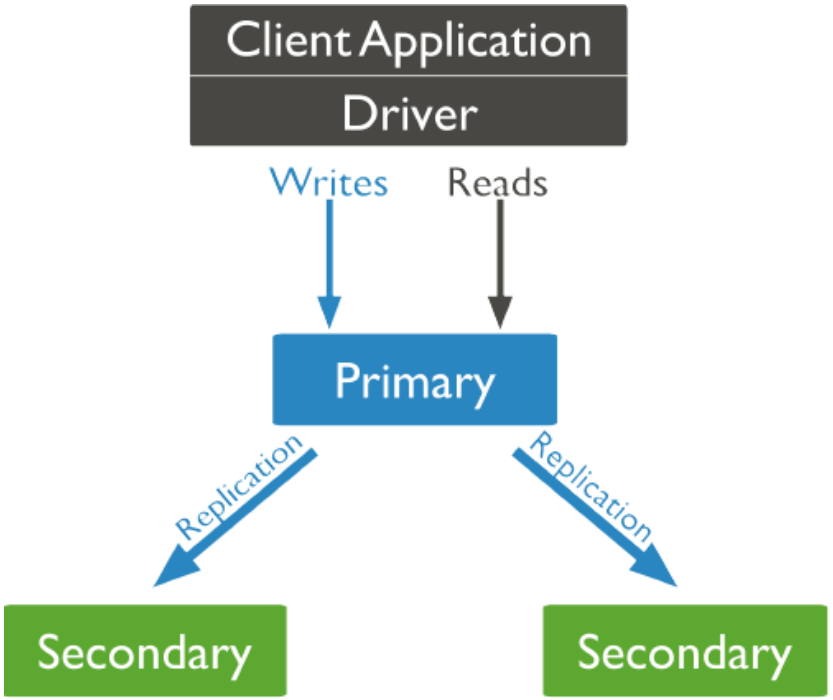
CRUD operations

University of Texas at Arlington

- Create
  - db.collection.insert(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- Read
  - db.collection.find(<query>, <projection>)
- Update
  - db.collection.update(<query>, <update>, <options>)
- Delete
  - db.collection.remove(<query>, <justOne>)

Replication

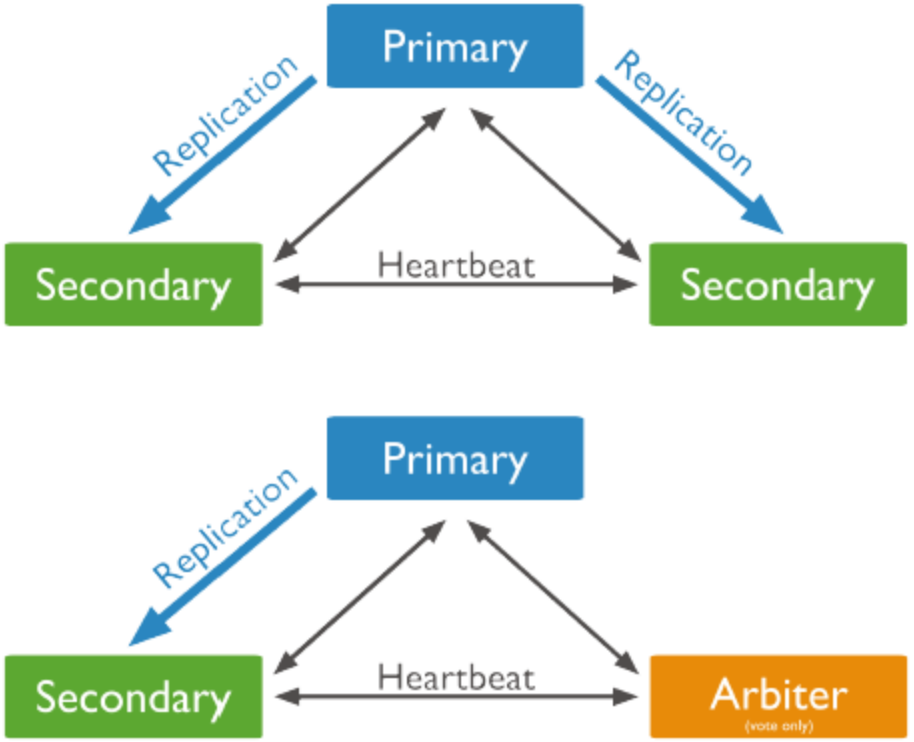
- Replica set
  - group of servers (or *mongod* processes) that maintain the same data set.
  - Only one server is active for writes (the primary, or master) at a given time



Replication

University of Texas at Arlington

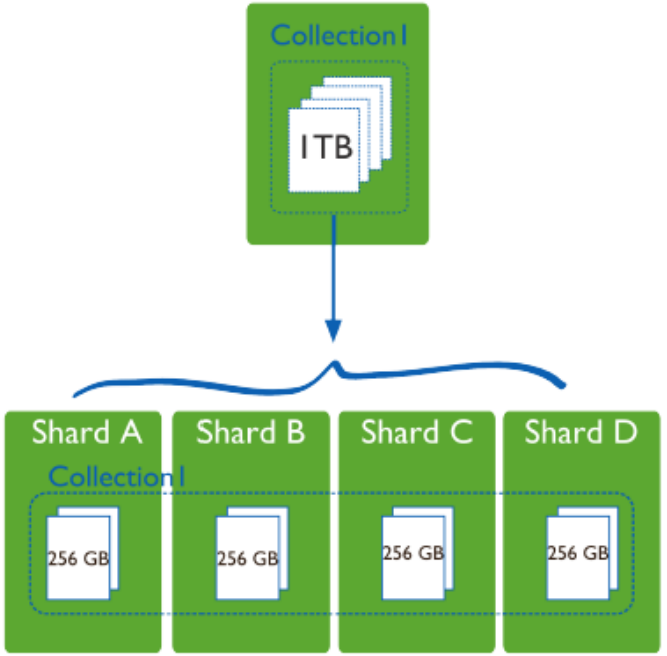
- Replica set
  - Replica set members send heartbeats (pings) to each other every two seconds. If a heartbeat does not return within 10 seconds, the other members mark the delinquent member as inaccessible.
  - Arbiter doesn't have copy of data (vote only)



Sharding

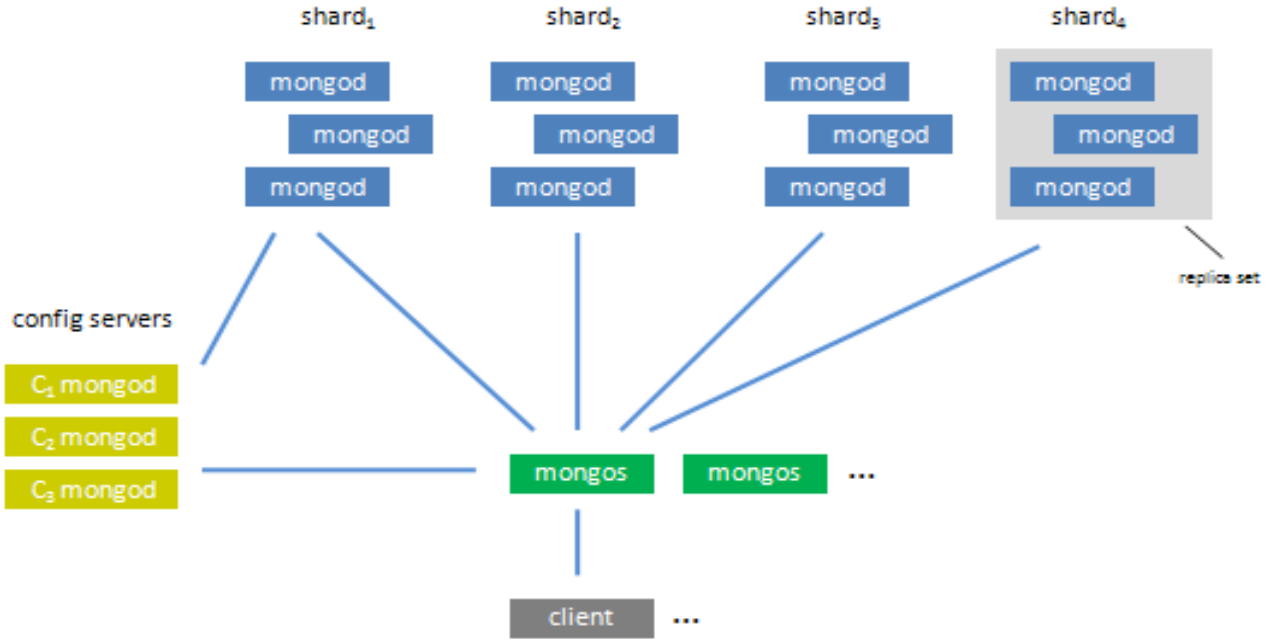
University of Texas at Arlington

- divides the data set and distributes the data over multiple servers, or shards (order-preserving manner)
- Each shard is an independent database, and collectively, the shards make up a single logical database.



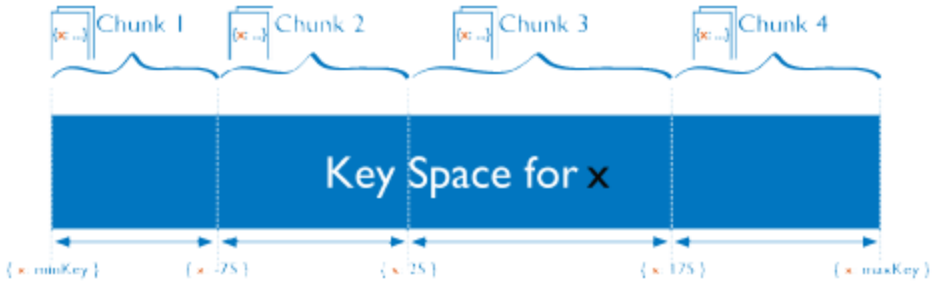
Sharding

- The set of servers/mongod process within the shard comprise a replica set



Sharding

- Example:

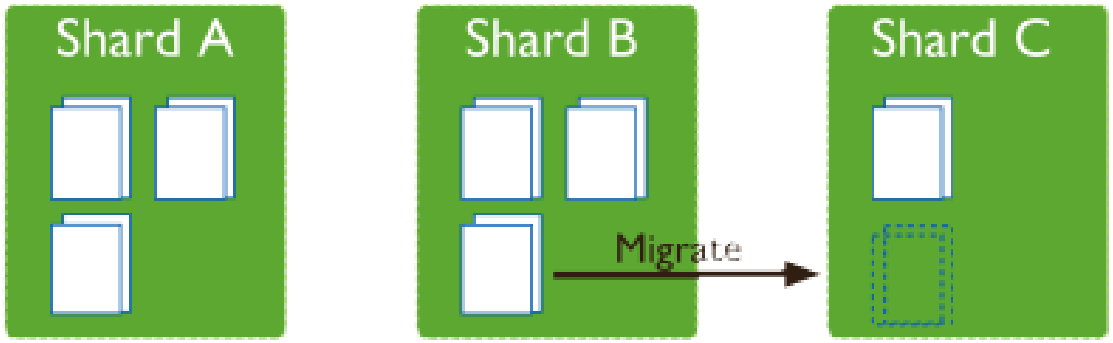


Machine 1	Machine 2	Machine 3
Alabama → Arizona	Colorado → Florida	Arkansas → California
Indiana → Kansas	Idaho → Illinois	Georgia → Hawaii
Maryland → Michigan	Kentucky → Maine	Minnesota → Missouri
Montana → Montana	Nebraska → New Jersey	Ohio → Pennsylvania
New Mexico → North Dakota	Rhode Island → South Dakota	Tennessee → Utah
	Vermont → West Virgina	Wisconsin → Wyoming

Sharding

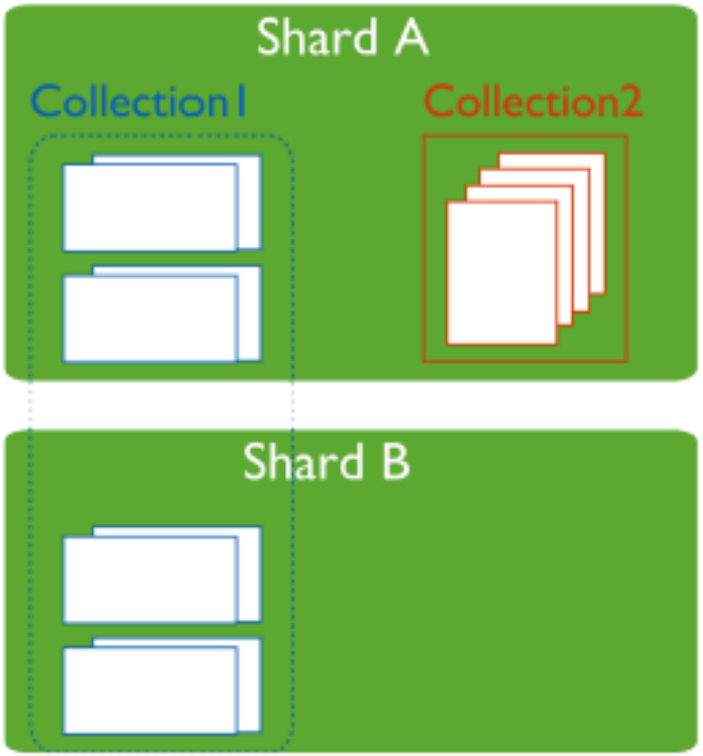
University of Texas at Arlington

- When the distribution of a sharded collection in a cluster is uneven, the *balancer* process will perform chunks migration



Sharding

- Every database has a "primary" shard that holds all the un-sharded collections in that database
- Example:
  - shard A = primary shard



Replication & Sharding

University of Texas at Arlington

- Sharding is the tool for scaling a system
- Replication is the tool for data safety, high availability, and disaster recovery.
- The two work in tandem yet are orthogonal concepts in the design.

Questions?

University of Texas at Arlington