CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Distributed Data Management

Lecture 6: Sharding with MongoDB

Document Oriented NoSQL→ MongoDB

In the previous week…

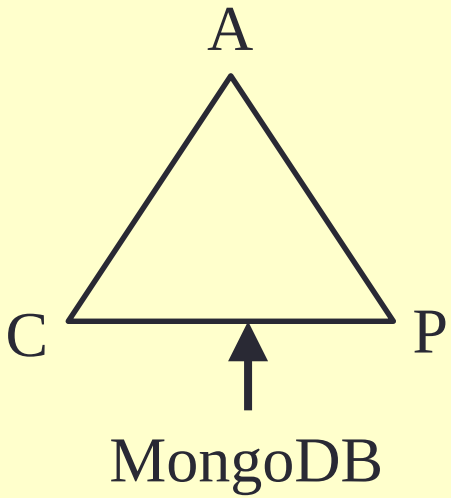
- Document Oriented DBs belong to the Key Value-based family:
  - The data is still stored in a key-value format.
  - However, values provide some structure → document format.
  - A value / document is encoded in JSON, BSON:
    - Accessible by most popular programming languages.
    - Very natural and flexible to be queried.

Document Oriented NoSQL→ MongoDB

In the previous week…

- MongoDB is the most popular document oriented DB.
  - High performance.
  - High availability.
  - Easy scalability.

- CAP theorem → MongoDB chose CP.
  - But they claim to be high available as well!



- MongoDB access using (among others):
  - JavaScript commands (native).
  - Pymongo library (from Python script).

MongoDB for Scaling Out

On the one hand, let's imagine Facebook has the following 21 DCs across the world.

MongoDB for Scaling Out

Let's imagine Facebook uses MongoDB to store a collection "users", representing info of what does this **500 million users** like:

- A document might look like this:

```
{ "name" : "Jill Bloggs",
  "country" : "Ireland",
  …,
  "interested_in_sports" : "Yes",
  …,
  "interested_in_shopping" : "No"
}
```

In reality, we would have a list property "interests", but this works as an example

MongoDB for Scaling Out

On the other hand, let's imagine that the marketing department of Zara is preparing the advertising campaign for the upcoming summer:

MongoDB for Scaling Out

- Zara marketing department wants to make good decisions:
  - How many ads do we create?
  - How much do we spend for advertising on radio and TV?

MongoDB for Scaling Out

- In order to make these decisions, Zara wants to do some data analytics. They have detected an increase of sales in Ireland.
  - So, is it real or just a lucky month of sales (like shorts in a heatwave)?
  - If it is real, how permanent is it? (i.e., will it be maintained?), or is it something that needs a big ad campaign in order to be consolidated?
  - They don't know. But Zara's data science department have come up with an idea:

Let's use Facebook and its collection "users" to see if this year there are more people living in "Ireland" and "interested in shopping" than last year.

MongoDB for Scaling Out

- So, this case study is about:
  1. How does Facebook store its collection “users”?
     (> 2 billion active users, supposedly)
  2. What happens when Zara triggers the query to MongoDB?

MongoDB for Scaling Out

- Facebook has 21 DCs.
- The collection *users* will occupy Terabytes →

  the collection is divided (spatially dispersed_ across the 21 DCs.

MongoDB for Scaling Out

- Facebook has 21 DCs.
- The collection “users” occupies terabytes → the collection is divided across nodes in the 21 DCs.

DC1	**DC2**	…	**DC21**
user1	user7		user3
user32	user9		user4
…	…		…

- Each document is in one (and only one) DC:
- From now on, we will call each DC node “a shard” (small part of a whole, just like a shard of glass from a broken window).
- We will call the set of DC nodes “the cluster”.

MongoDB for Scaling Out

How are the documents distributed?

- Shard key: A collection is said to be "sharded" if one of its fields (key) is selected as a *shard key* for the collection's document distribution across the shards of the cluster.

- Let's suppose we choose country as our shard key.

{ "name" : "Nacho",
"country" : "Ireland", ← **Shard Key**
…,
"interested_in_sports" : "Yes",
…,
"interested_in_shopping" : "No"
}

MongoDB for Scaling Out

How are the documents distributed?

- Distribution → Shard key ranges.

Metadata:	**S1**	**S2**	…	**S21**
[alb --- cro) → S1	user1	user7		user3
[cro --- lit) → S2	user32	user9		user4
…	…	…		…
[tai --- zam) → S21				

- Each user (document) belonging to the countries Albania, Brazil, Cameroon, etc. (with a shard key "country" in [alb --- cro) will be placed in Shard S1 (e.g. in the Frankfurt DC).

MongoDB for Scaling Out

Why Shard key ranges distribution?

- If I want to query the Facebook users of Argentina, then I know that, even if the collection "users" occupy terabytes and the data is distributed across 21 DCs, all users from Argentina will be in S1 (i.e., the DC of Frankfurt).

Metadata:	**S1**	**S2**	…	**S21**
[alb --- cro) → S1	user1	user7		user3
[cro --- lit) → S2	user32	user9		user4
…	…	…		…
[tai --- zam) → S21				

MongoDB for Scaling Out

Why Shard key ranges distribution?

- Problem: If we do a query that does not involve the shard key "country", then we will need to scan all 21 shards – very inefficient with latency involved across the internet.

Metadata:	**S1**	**S2**	…	**S21**
[alb --- cro) → S1	user1	user7		user3
[cro --- lit) → S2	user32	user9		user4
…	…	…		…
[tai --- zam) → S21				

MongoDB for Scaling Out

Shard key range → "Chunk" of data

- All the documents sharing a shard key range are referred to as a chunk of data (usually 64 MB, although configurable).
  - So, if the shard key range is [alb --- cro) it means that all documents (users) from Albania, Argentina, Brazil, Cameroon occupy all together about 64MB or less.
  - Note: if we wanted to ensure data stayed geographically close to users, we could use a different shard key, such as using GPS coordinates.

Metadata:	**S1**	**S2**	…	**S21**
[alb --- cro) → S1	user1	user7		user3
[cro --- lit) → S2	user32	user9		user4
…	…	…		…
[tai --- zam) → S21				

MongoDB for Scaling Out

What happens if a key range grows?

- Imagine Murphy's triggers a campaign →
  - Number of new Facebook users in Ireland grow.
  - Shard S2 grows beyond 64MB.
  - We cannot let this grow to the level of hundreds of MBs!

Metadata:	**S1**	**S2**	…	**S21**
[alb --- cro) → S1	user1	user7		user3
[cro --- lit) → S2	user32	user9		user4
…	…	…		…
[tai --- zam) → S21				

MongoDB for Scaling Out

What happens if a key range grows?

- The system does two operations in the background:
  1. Split.
  2. Migrate.

  - Split divides the key range into 2 sub-ranges (sub-chunks).
  - A balancer component will figure out what a good intermediate key is, e.g. maybe “ita”

Metadata:

[alb --- cro) → S1

[cro --- ??) → S2

[?? --- lit) → S2

[tai --- zam) → S21

S1	S2 …	S21
user1	user7	user3
user32	user9	user4
	…	…

MongoDB for Scaling Out

What does it happen if a key range grows?

- The system does two operations (background):
  1. Split.
  2. Migrate.

  - Split has not yet migrated anything; it has just updated the metadata. So, *split* is a very lightweight (cheap) operation.

Metadata:

[alb --- cro) → S1

[cro --- ita) → S2

[ita --- lit) → S2

[tai --- zam) → S21

S1	S2 …	S21
user1	user7	user3
user32	user9	user4
	…	…

MongoDB for Scaling Out

What does it happen if a key range grows?

- The system does two operations (background):
  1. Split.
  2. Migrate.
  - Migration actually migrates the new key sub-range to a different shard (i.e., move the chunk from Frankfurt DC to Helsinki DC).
  - A balancer component decides which is the best shard to be moved to (possible policy: maintain all shards with similar load of data).

Metadata:

[alb --- cro) → S1

[cro --- ita) → S2

[ita --- lit) → S3

[tai --- zam) → S21

S1	S2 …	S21
user1	user7	user3
user32	user9	user4
	… …	

MongoDB for Scaling Out

What does it happen if a key range grows?

- The system does two operations (background):
  1. Split.
  2. Migrate.
  - Migration is an expensive operation. Thus, the system will schedule the migrations so as to not to overload the cluster. In particular, not more than one migration between 2 nodes at a time.

Metadata:

[alb --- cro) → S1

[cro --- ita) → S2

[ita --- lit) → S3

[tai --- zam) → S21

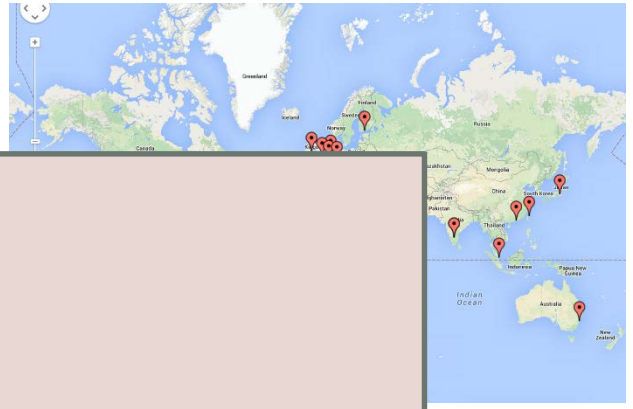
S1	S2 …	S21
user1	user7	user3
user32	user9	user4
	…	…

MongoDB for Scaling Out

Processes Involved.

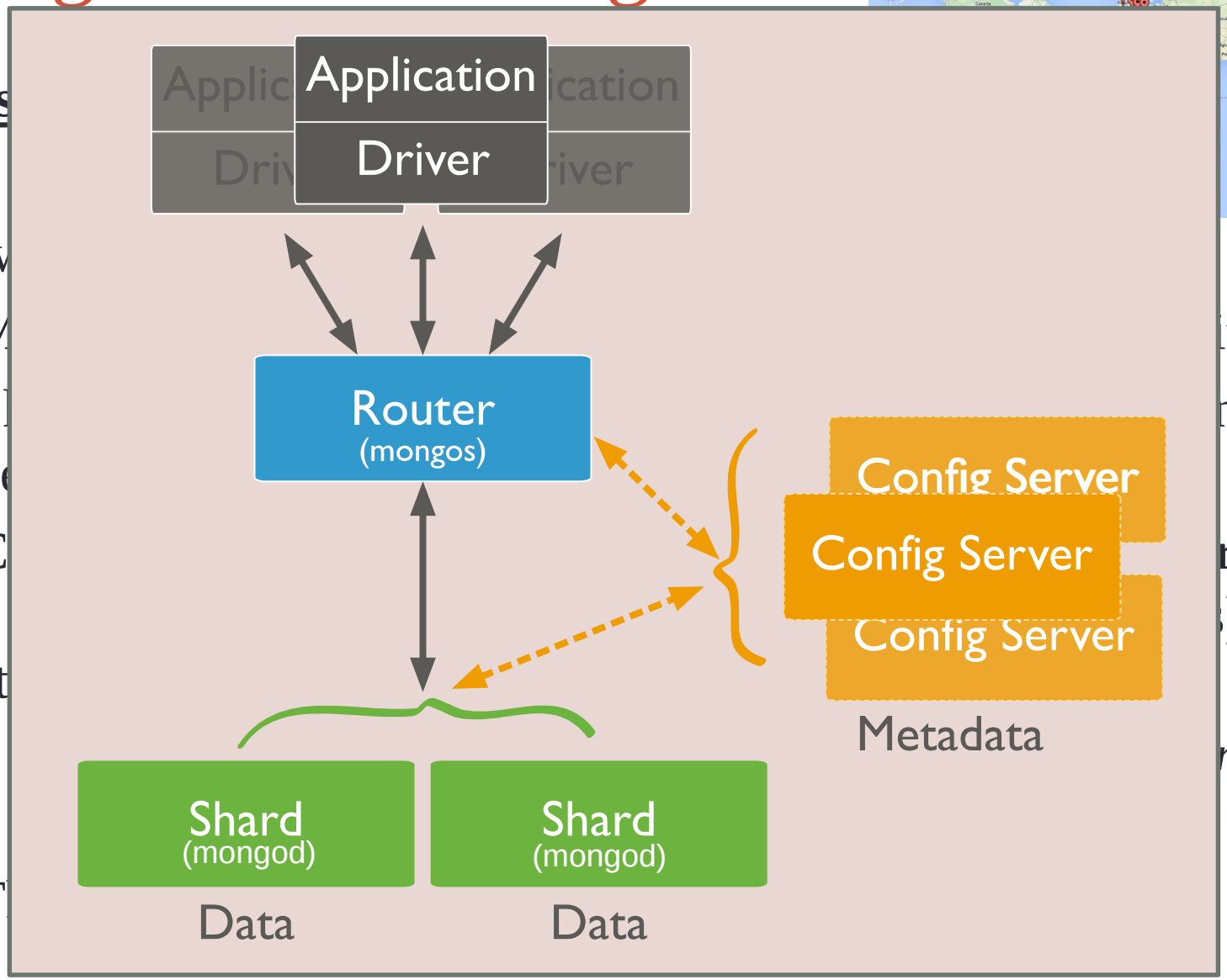
- How does Facebook sets up this environment?
  - Metadata → Configuration Server (usually 3 for availability).
  - Shards → Each shard needs at least 1 **mongod** database engine per server.
  - Client abstraction → A process, **mongos (MongoDB Shard**), is a routing service between application layer and the node(s) storing the data. **mongos** communicates with:
    - The config server: *"I have this query, where should I look for?"*
    - The mongod of concrete shard(s): *"Please trigger this query"*
  - The Zara team uses **mongo** client to connect to **mongos**

MongoDB for Scaling Out

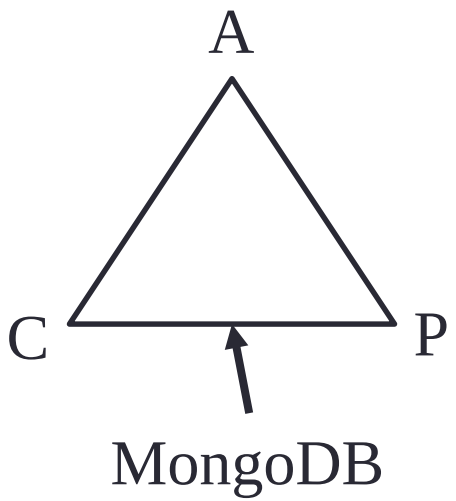


Proces

- Hov
  - o M … ity).
  - o Sh … ngine pe
  - o C … **rd**), is a … s) st
    - …
    - …
  - o T …



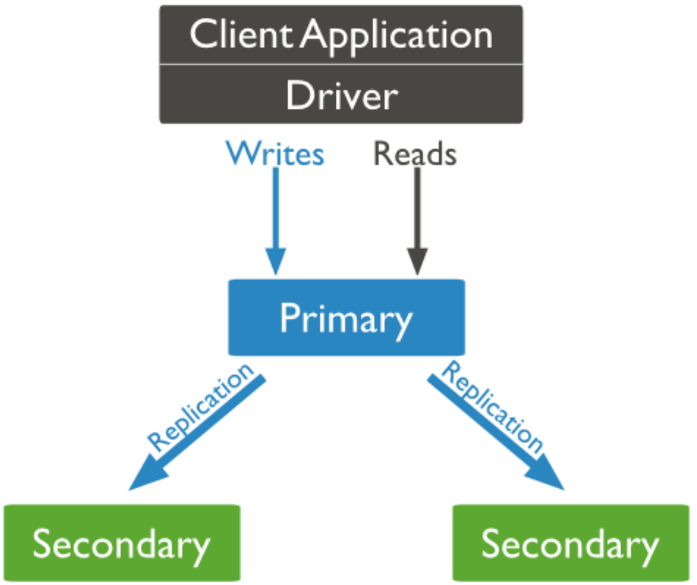
MongoDB for Scaling Out



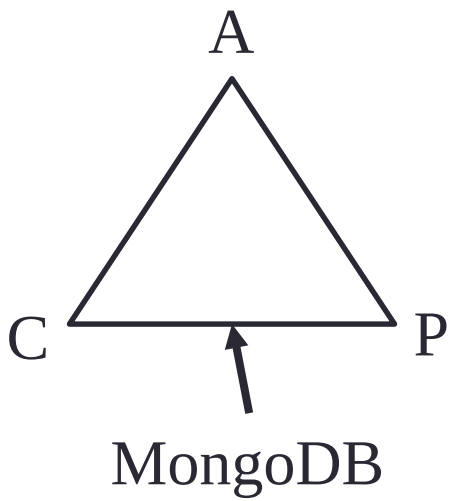
Conclusions:

…How does MongoDB achieves consistency and high availability?

- **Sharding + Replication**
  - **Sharding:** Cluster for data distribution
  - **Replication:** Each shard of the cluster contains multiple copies of data. This data is replicated / synchronised on several servers.



MongoDB for Scaling Out

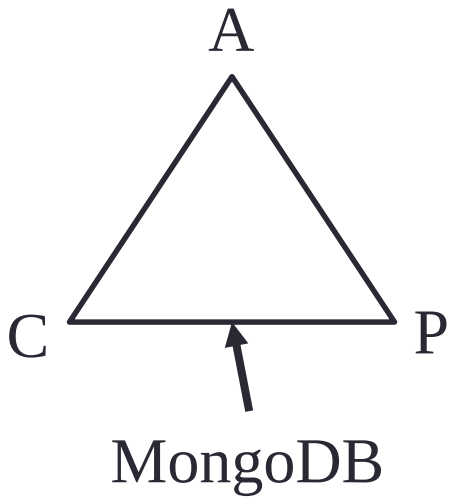


Conclusions:

…How does MongoDB achieves consistency and high availability?

- **Replication** → Synchronise data among multiple servers.
- MongoDB → Set up a **replica set** on each shard.
  - Replica Set: Multiple servers, all with the very same documents.
  - 1 server is primary, others are secondary.
  - Client always interacts with primary (read / write operations).
    - Primary automatically replicates the updates to secondary ones.
  - If primary is shut down, a secondary node *is elected* as primary.
  - When former primary recovers, continue acting as secondary.

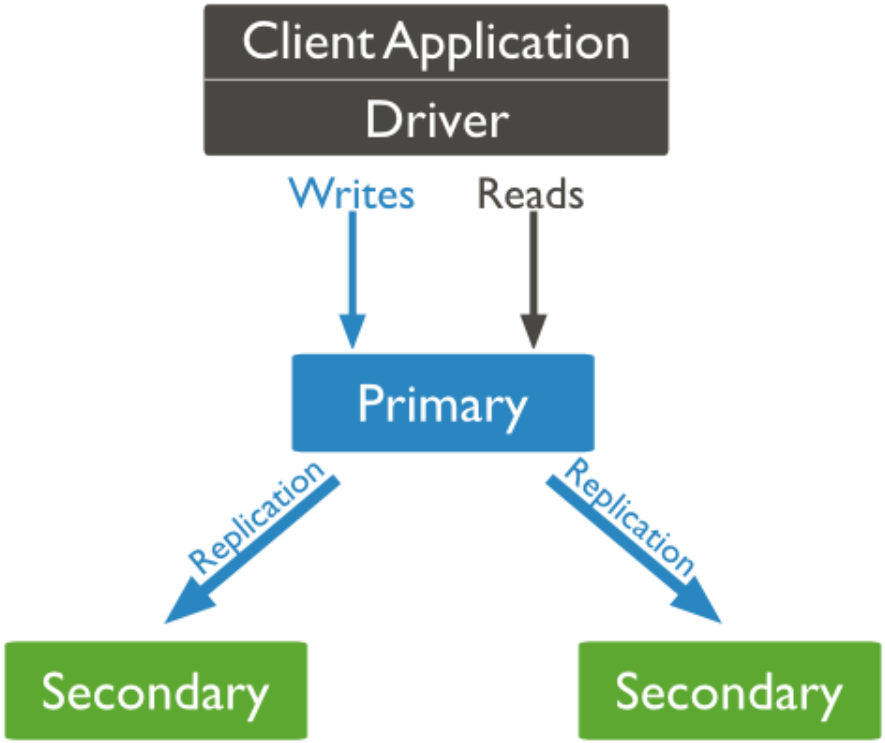
MongoDB for Scaling Out



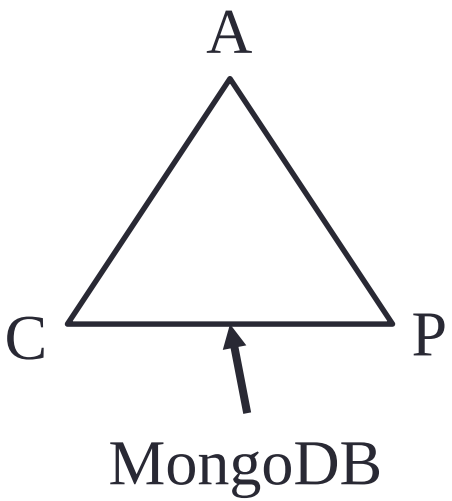
Conclusions:

…How does MongoDB achieves consistency and high availability?

- Replication allows:
  - Redundancy
  - Data availability
  - Disaster recovery



MongoDB for Scaling Out



Conclusions:

…How does MongoDB achieves consistency and high availability?

- Replication + Sharding
  - Each shard has a replica set.
  - If a client queries a shard, the sharding distribution mechanism ensures the data is consistent (the operations are performed against the primary node of the shard).
  - The primary node will be mostly fully available (replication minimises the waiting time if the primary node is shut down).

Thank you for your attention!