Replicación y Particionamiento

Enrique Barra Arias

Andrés Muñoz

Alejandro Pozo

Contenidos

- **Replicación**
  - **Algoritmos**

- Replicación con MongoDB
  - Descripción
  - Comandos

- Particionamiento
  - División
  - Gestión
  - Particionamiento y replicación

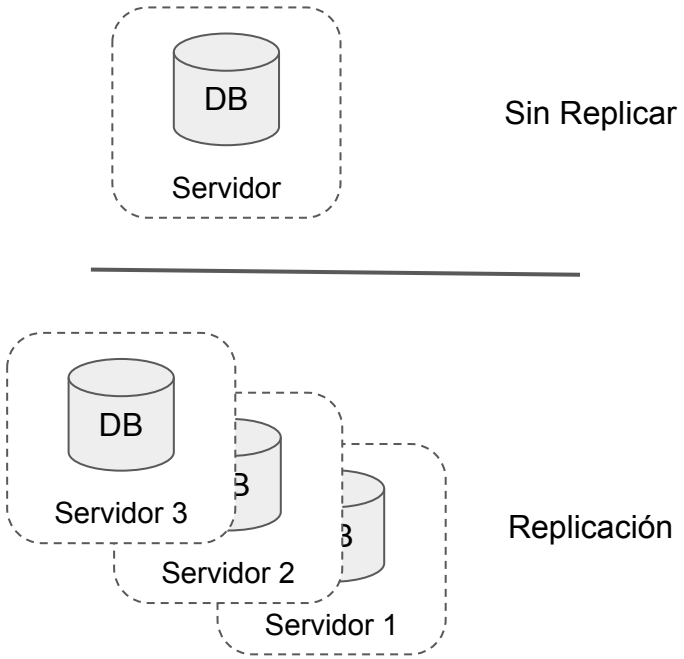
- Particionamiento con MongoDB
  - Descripción
  - Comandos

Replicación

La **replicación** dispone datos en **más de un servidor** a la vez

¿Por qué?

- Mantener **datos a salvo**
  - Recuperación frente a desastres
- **HA** (High Availability)
- Mayor rendimiento de lecturas
- Reducir la latencia
- **No** hay **downtime** de mantenimiento
  - Backups, reconstruir índices, compactación, etc



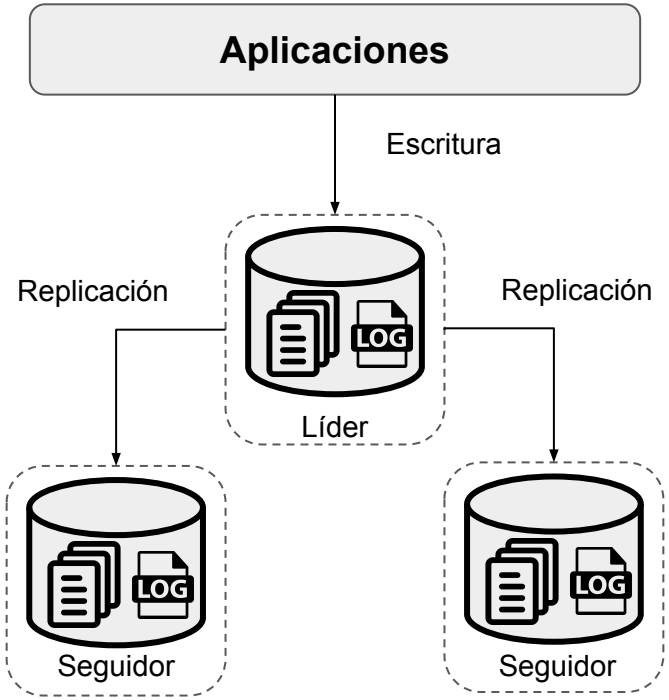
Algoritmos

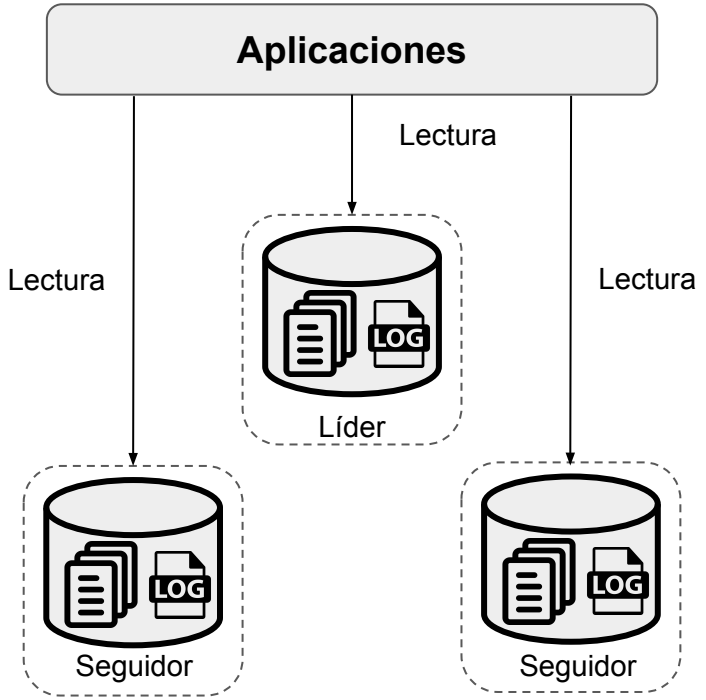
En función de cómo se **gestiona la forma de replicar** los datos

- Líder individual
- Líder múltiple
- Sin líder

Algoritmos - Líder individual

1. Solo el **líder acepta** peticiones de **escritura**
2. Los **seguidores** aplican los cambios según los **logs de replicación**
3. Las peticiones de **lectura** se pueden enviar al **líder o a los seguidores**





Algoritmos

Replicación Síncrona → Líder espera respuesta de que al menos un seguidor ha conseguido replicar los datos

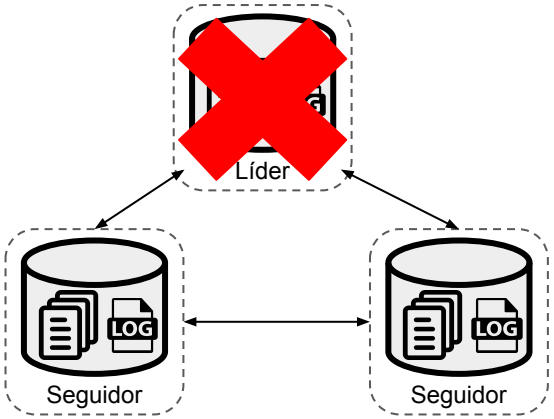
- \+ Garantia de datos en seguidores
- – Puede bloquear el flujo

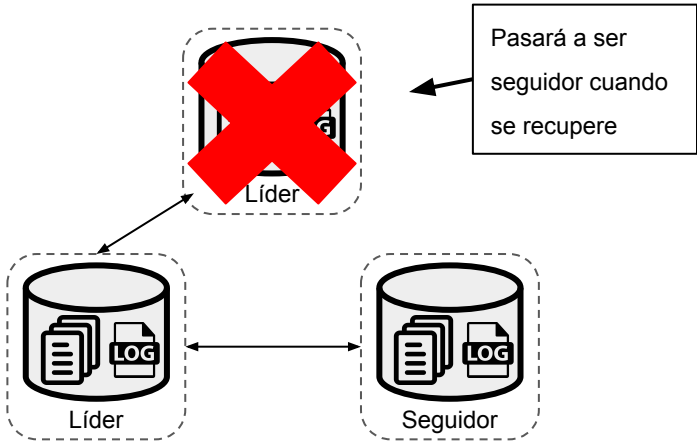
Replicación Asíncrona → Líder no espera de seguidores y sigue procesando peticiones de escritura

- \+ Líder procesa datos aún fallando seguidores
- – Pueden perderse datos

¿Qué pasa si falla el líder?

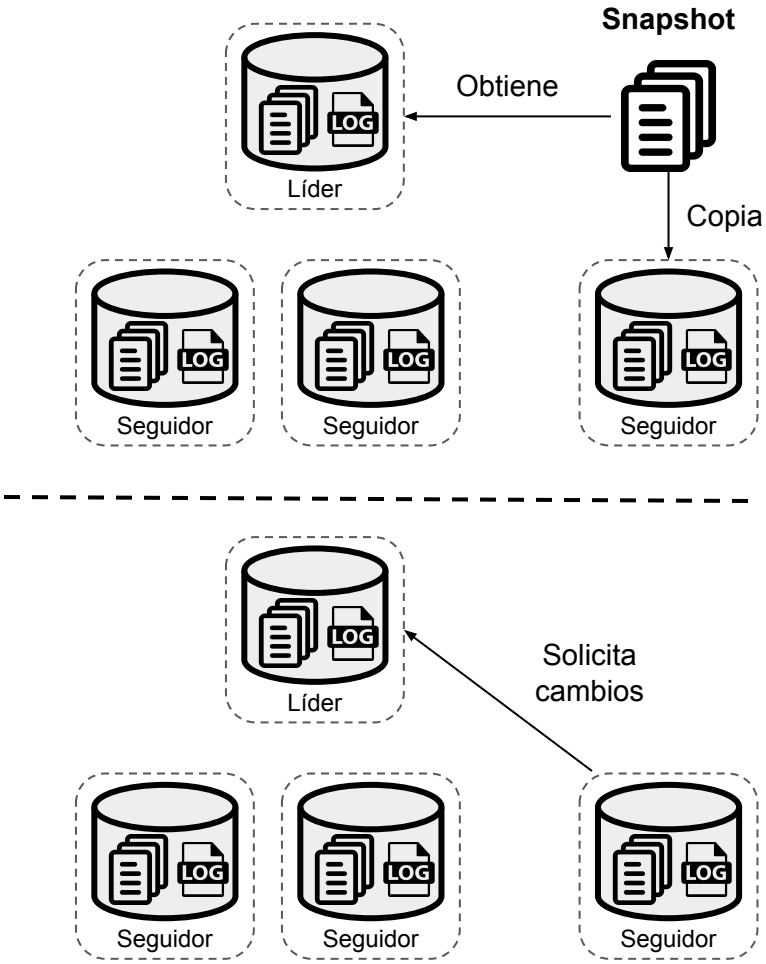
1. Determinar que el líder ha fallado
   - ↳ Timeout
2. Elección de un nuevo líder
   - ↳ Parámetros de elección
   - ↳ Votación réplicas
3. Reconfigurar el sistema para que use el nuevo líder





¿Cómo añadir un nuevo seguidor?

1. Realizar un snapshot del líder
2. Copiar Snapshot al nuevo seguidor
3. Nuevo seguidor solicita cambios desde el snapshot
4. Nuevo seguidor procesa todos los cambios



Algoritmos - Líder múltiple

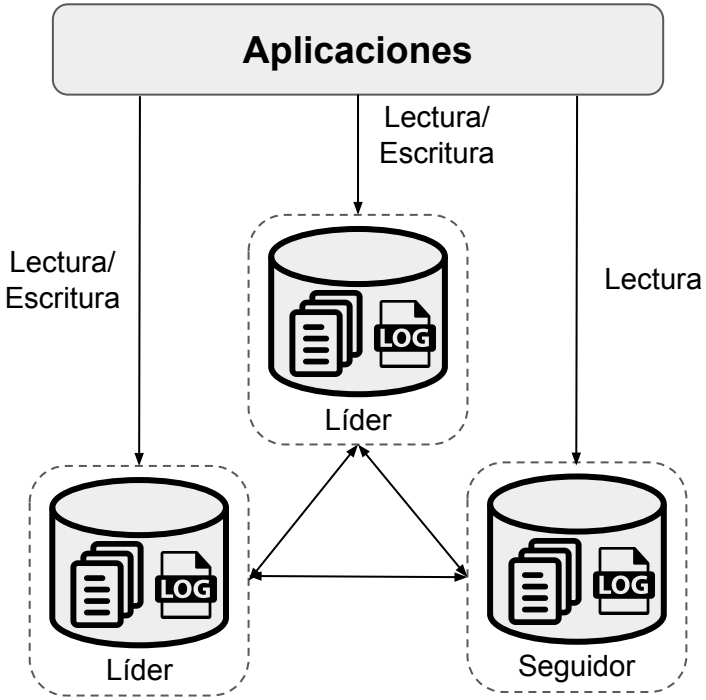
Permite que **más de un nodo** procese **peticiones de escritura**

Casos de uso

- Herramientas colaborativas

- Aplicaciones offline

- Centros de datos múltiples

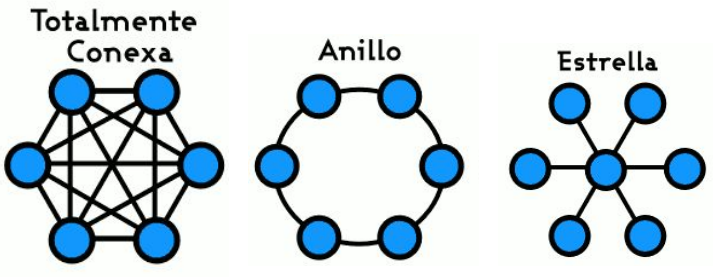


Algoritmos

La mayor **desventaja** es la **gestión de conflictos**

¿Cómo abordarlo?

- **Evitar** conflictos
- Todas las réplicas **converjan** al **mismo resultado**
  - Prioridades, unir resultados, etc
- Resolución de conflictos personalizada. Dos enfoques:
  - **On Write**
  - **On Read**
- **Topología** de las réplicas
  - All-to-all, círculo, estrella, etc

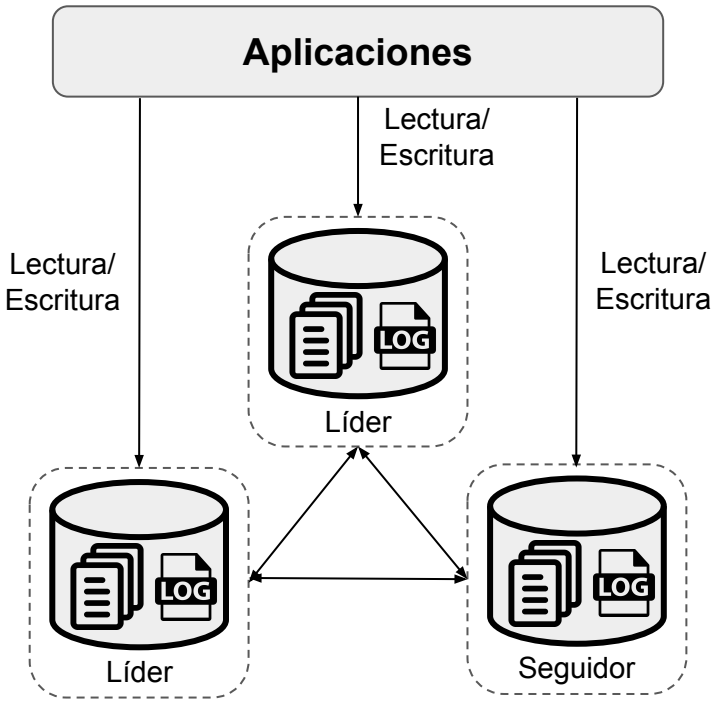


Algoritmos - Sin líder

Cualquier réplica puede aceptar **peticiones de escritura**

Buscan la consistencia de datos mediante **Quorums** de lectura (r) y escritura (w)

r + w > número de réplicas



Casos de uso

- Entornos que requieren alta disponibilidad y baja latencia
  - Ej. IoT (smart Vehicles)

Contenidos

- Replicación
  - Algoritmos

- **Replicación con MongoDB**
  - **Descripción**
  - **Comandos**

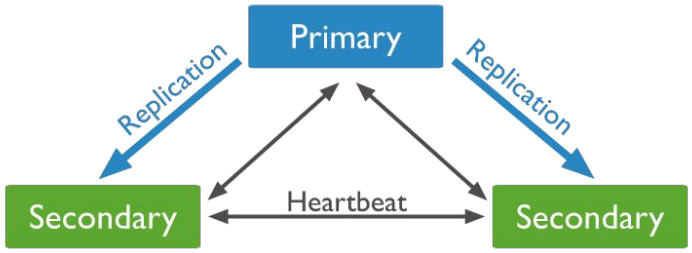
- Particionamiento
  - División
  - Gestión
  - Particionamiento y replicación

- Particionamiento con MongoDB
  - Descripción
  - Comandos

Descripción

En MongoDB, la replicación se consigue se despliega un clúster llamado **“Replica set”**

- Replicación de líder individual asíncrona
- Servidor principal (líder) + secundarios (seguidores)
- Servidores informan de su estado y versión de datos mediante latidos (pings)
- Limitación de 50 nodos réplica
- Servidores tienen prioridad de ser elegidos principal



Descripción

Servidor principal + Servidores secundario } **Dan servicio**

Servidores con prioridad 0
- ↳ Sustituto de servidores secundarios
- ↳ No puede ser elegido como principal

Servidor oculto
- ↳ Son de prioridad de 0
- ↳ Para copias de seguridad o informes

Servidor retardado
- ↳ Reflejan estado anterior de los datos

} **Backup**

Oplog
- ↳ Colección de operaciones de escritura
- ↳ Usado para replicar del primario a secundarios

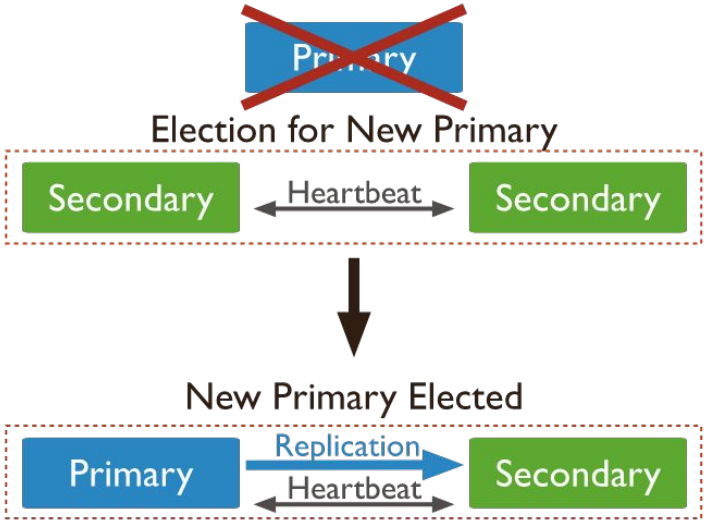
Árbitro
- ↳ Para decidir nuevo líder en caso de empate durante la elección

} **Gestión**

Descripción

¿Qué pasa si falla el primario?

1. Determinar que el primario ha fallado
   - ↳ Latidos con información de estado cada 2 segundos
   - ↳ Timeout 10 segundos
2. Elección de un nuevo líder
   - ↳ Prioridad servidor
   - ↳ Datos actualizados
   - ↳ Conexión con cluster



3. Reconfigurar el sistema para que use el nuevo primario
   - ↳ Servidor caido entrará como secundario

Comandos

Arrancar un servidor de MongoDB en modo réplica

```
mongod --replSet "my-mongo-set"
```

Conectarse al primario e inicializar replica set

```
rs.initiate({
    _id : "my-mongo-set",
    members: [ { _id : 0, host : "mongodb0.example.net:27017" }]
})
```

Añadir más nodos

```
rs.add("mongodb1.example.net:27017")
rs.add("mongodb2.example.net:27017")
```

Configuración y estado de la réplica

```
rs.status()
rs.config()
```

Comandos

Opciones de configuración al añadir un nodo

```
_id: <int>,                    // (Opcional)
host: <string>,                // (Obligatorio)
arbiterOnly: <boolean>,        // (Opcional)
buildIndexes: <boolean>,       // (Opcional)
hidden: <boolean>,             // (Opcional)
priority: <number>,            // (Opcional)
tags: <document>,              // (Opcional)
slaveDelay: <int>,             // (Opcional)
votes: <number>                // (Opcional)
```

Ejemplo

```
rs.add({ host: "mongodb.net:27017", priority: 0, hidden: true })
```

Comandos

Añadir varios nodos al inicializar

```
rs.initiate({
    _id : "my-mongo-set",
    members: [
        { _id : 0, host : "mongodb0.example.net:27017" },
        { _id : 1, host : "mongodb1.example.net:27017" },
        { _id : 2, host : "mongodb2.example.net:27017" },
    ]
})
```

O inicializar desde fichero de configuración (JSON)

```
rs.initiate(config)
```

Para realizar lecturas en línea de comandos del secundario ejecutar:

```
rs.slaveOk()
```

Comandos

Añadir árbitros a la réplica

```
rs.addArb("mongodb3.example.net:27017")
```

- O usar opción *arbiterOnly* al añadir un nuevo nodo

Eliminar una réplica del cluster

```
rs.remove("mongodb2.example.net:27017")
```

Conectarse al ReplicaSet desde **Moongose** (librería de Node.js):

```
mongoose.connect('mongodb://
    <hostMongo1>:<portMongo1>,
    <hostMongo2>:<portMongo2>,
    <hostMongo3>:<portMongo3>/<dabase_name>
       ?replicaSet=<nombre_replica_set>')
```

Contenidos

- Replicación
  - Algoritmos

- Replicación con MongoDB
  - Descripción
  - Comandos

- **Particionamiento**
  - **División**
  - **Gestión**
  - **Particionamiento y replicación**

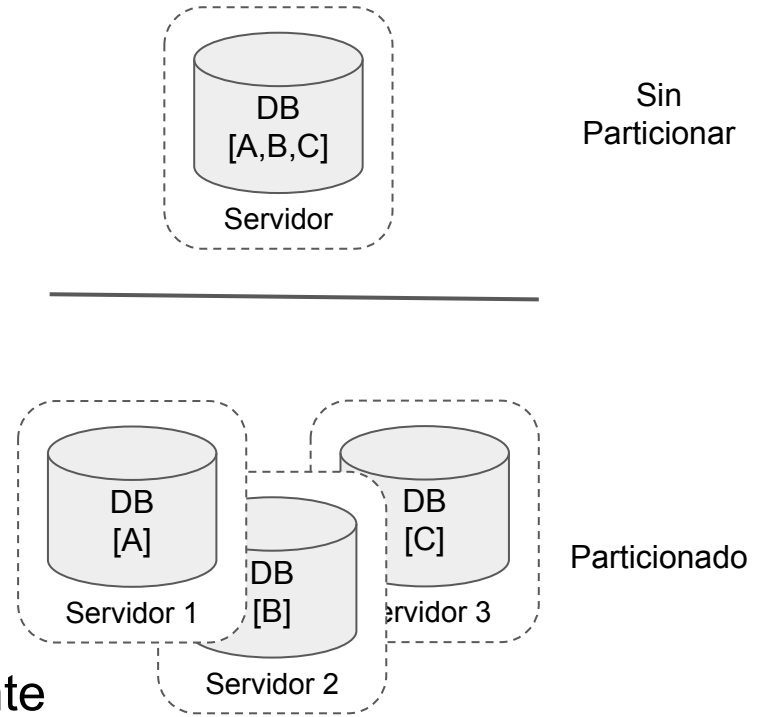
- Particionamiento con MongoDB
  - Descripción
  - Comandos

Particionamiento

El **particionamiento (o sharding)** distribuye los datos entre múltiples máquinas

¿Por qué?

- Escalabilidad horizontal
  - Más económica que la vertical
- Mejora rendimiento
  - Operaciones más rápidas



Si el particionamiento no se hace correctamente se **pierde mucha efectividad**

Particionamiento

En función de cómo se **dividen** las particiones

- Clave-Valor
  - ↳ Rango
  - ↳ Hash de clave
- Índices secundarios

En función de cómo se **gestionan** las particiones

- Algorítmico
- Dinámico

Particionamiento Clave-valor

Rango → Selecciona en qué partición se va escribir o leer en función de si la clave primaria está dentro de un rango.

- ↳ Temporal (Date)
- ↳ Alfabética (String)
- ↳ Númerica (Integer, Double)
  - – Patrones de datos → Particiones más llenas que otras

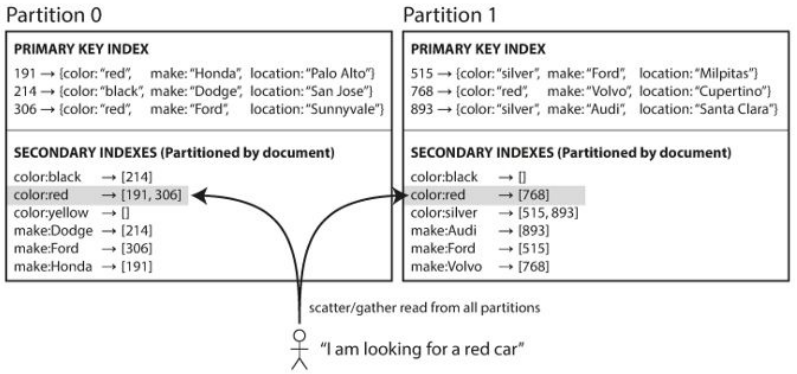
Hash → Aplica un hash a la clave antes de procesar la petición de escritura

- \+ Guarda los datos uniformemente
- − Pierde efectividad en las búsquedas

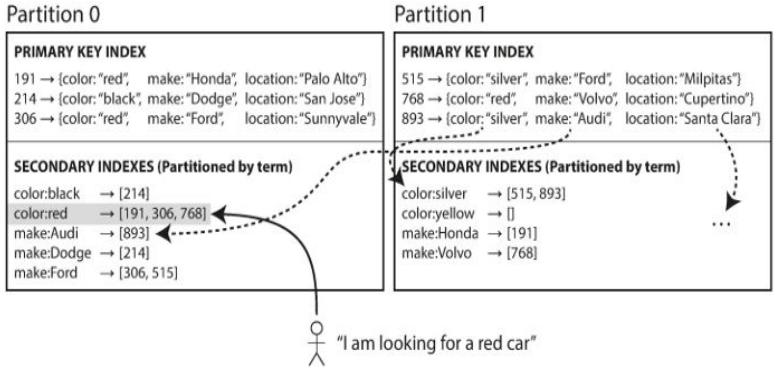
Particionamiento Índices Secundarios

Los índices secundarios permiten agilizar búsquedas pero tienen el problema de que no son únicos

- Por documento
  - Índice local
  - Resultados de todas particiones



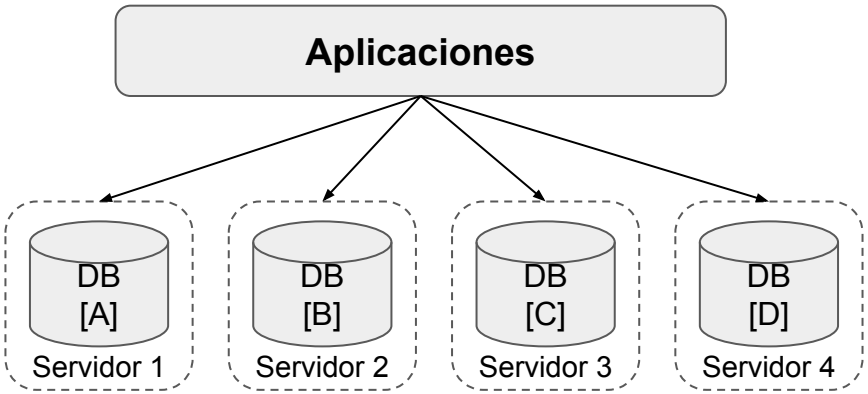
- Por objeto
  - Índice global
  - Más complejo



Particionamiento Algorítmico

En el **particionamiento algorítmico** se definen números fijos de particiones

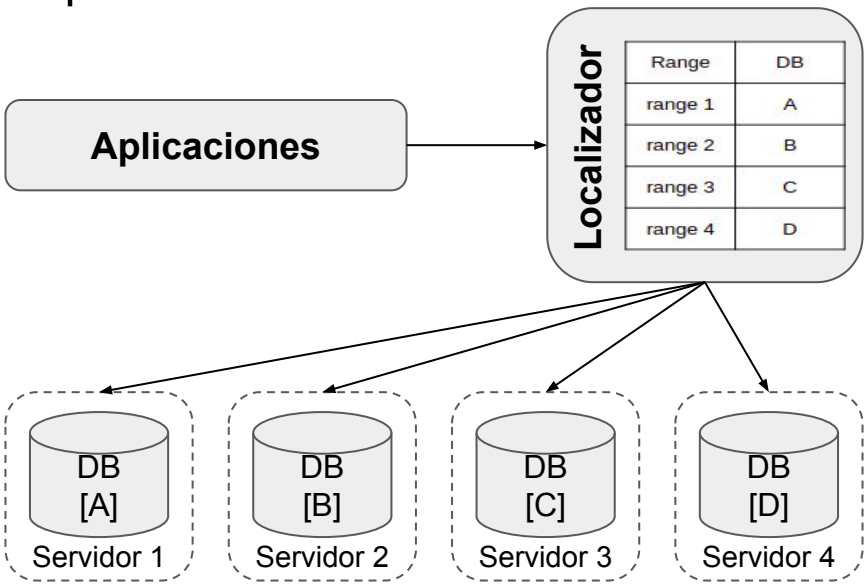
- \+ Aplicaciones determinan a qué partición dirigirse
- ~ Elección de clave es crítico → Datos homogéneos
- − Rebalanceo complicado



Particionamiento dinámico

En el **particionamiento dinámico** las particiones se ajustan al tamaño de los datos y se usa un servicio de localización para ubicar los datos

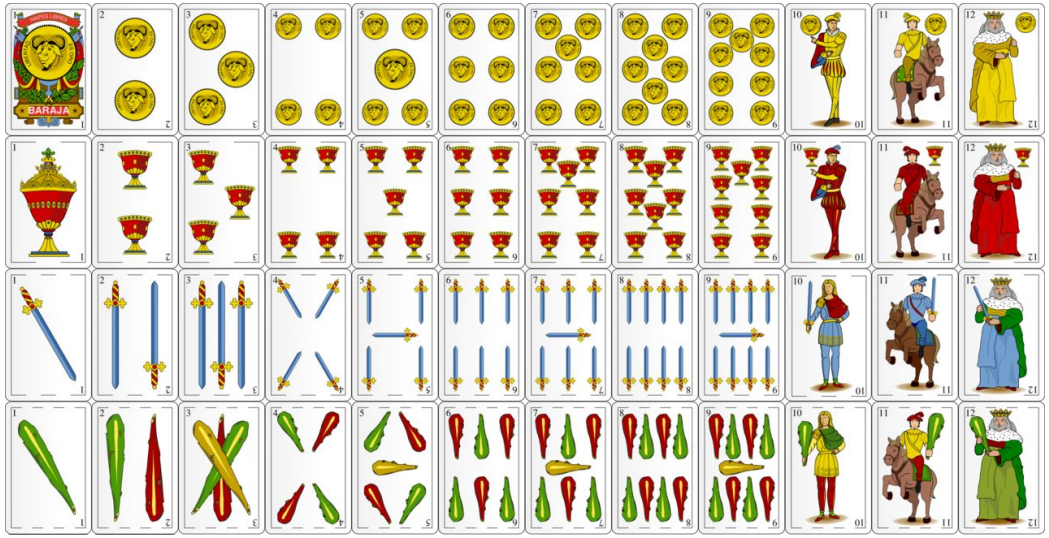
+ El balanceo entre particiones es trivial
+ Resistente a datos no uniformes
− Localizador es un punto único de fallo



Particionamiento dinámico

Ejemplo de particionamiento dinámico con cartas:

- *Baraja de 48 cartas que son los datos a insertar,*
- *4 jugadores que son los particiones,*
- *Un repartidor que es el localizador.*
- *Cada jugador admite hasta 12 cartas (tamaño máximo de la partición)*



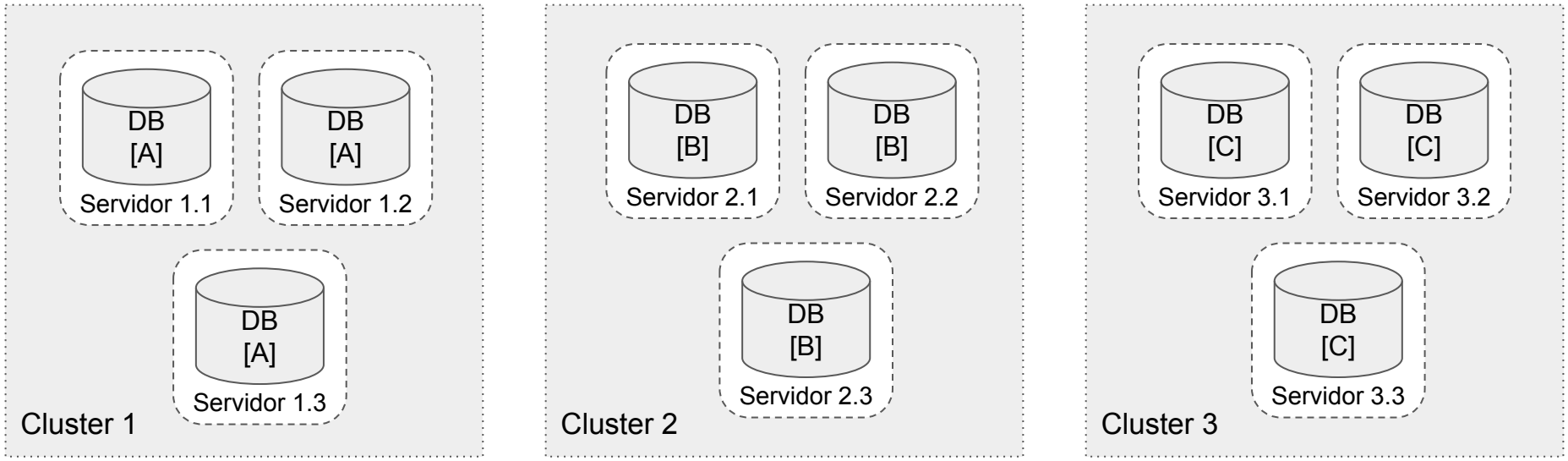
Particionamiento dinámico

1. El repartidor comienza repartiendo las cartas y se las da todas al primer jugador
2. Hasta que tiene 12 → 1,2,3,3,4,5,5,6,9,9,10,12
3. En ese momento reparte con el segundo jugador, quedando:
   - ↳ Jugador 1: 1,2,3,3,4,5,5
   - ↳ Jugador 2: 6,9,9,10,12
4. Continúa repartiendo. Hasta que otro jugador llega a 12. Momento en el que habrá que volver a rebalancear con un nuevo jugador
   - ↳ Jugador 1: 1,1,2,3,3,4,4
   - ↳ Jugador 2: 5,5,5,6,6,7,8,8
   - ↳ Jugador 3: 9,9,9,10,10,11,11,12,12
   - ↳ Jugador 4: sin cartas.
5. Y termina repartiendo el resto y tocará de nuevo rebalancear:
   - ↳ Jugador 1: 1,1,1,1,2,2,2,2,3,3,3,3
   - ↳ Jugador 2: 4,4,4,4,5,5,5,5,6,6,6,6
   - ↳ Jugador 3: 7,7,7,7,8,8,8,8,9,9,9,9
   - ↳ Jugador 4: 10,10,10,10,11,11,11,11,12,12,12,12

Particionamiento + Replicación

Generalmente se suelen **aplicar los dos procedimientos**

Una **partición** suele estar también **replicada varias veces**



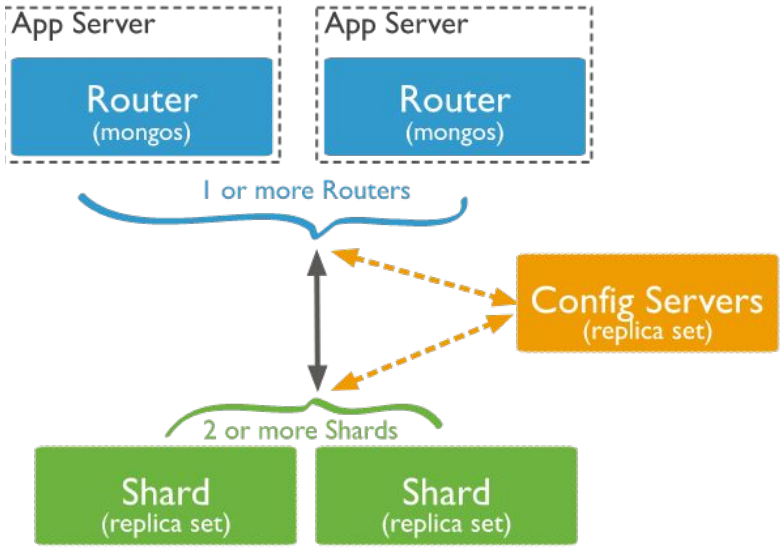
Contenidos

- Replicación
  - Algoritmos
- Replicación con MongoDB
  - Descripción
  - Comandos
- Particionamiento
  - División
  - Gestión
  - Particionamiento y replicación
- **Particionamiento con MongoDB**
  - **Descripción**
  - **Comandos**

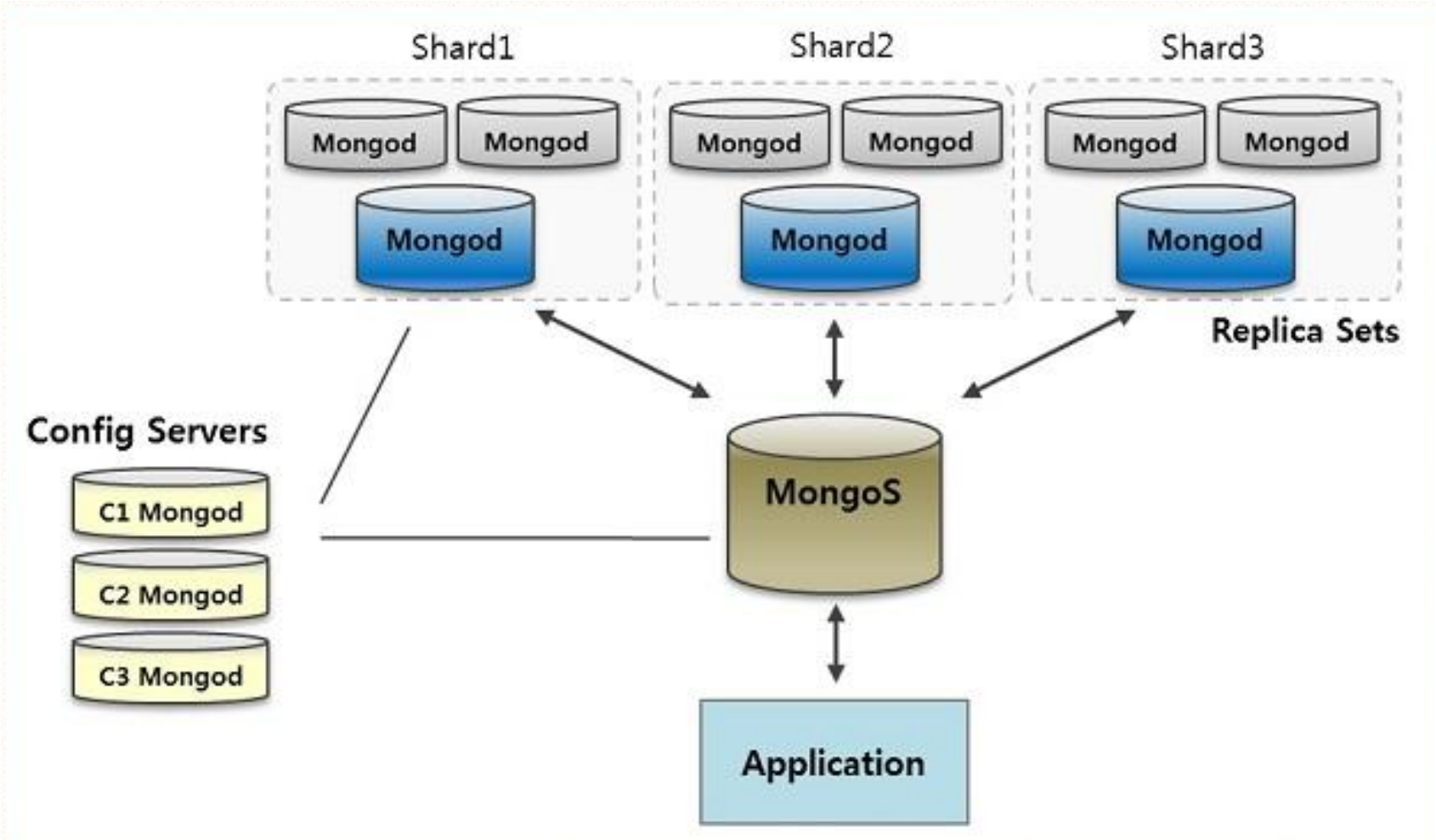
Particionamiento MongoDB

En MongoDB, la particiones se llaman **“Shards”**

- **Partición dinámica** pero permite definir un **set inicial de shards**
- El localizador es el llamado **Config Server**
- Introduce también un router **Mongos** que provee una interfaz entre los clientes y los shards
- Claves pueden estar basadas en **hash** o en **rango**



Particionamiento y Replicación



Particionamiento

Para **arrancar** un mongo como **config server**

```
mongod --configsvr --replSet <replica set name> --dbpath <path> --bind_ip
localhost,<hostname(s)|ip address(es)>
```

Para **inicializar los config servers**

```
rs.initiate(
  {
    _id: "<replSetName>",
    configsvr: true,
    members: [
      { _id : 0, host : "cfg1.example.net:27019" },
      { _id : 1, host : "cfg2.example.net:27019" },
      { _id : 2, host : "cfg3.example.net:27019" }
    ]
  }
)
```

Para **arrancar** un mongo como **sharded cluster**

```
mongod --shardsvr --replSet <replSetname>  --dbpath <path> --bind_ip
localhost,<hostname(s)|ip address(es)>
```

Para **inicializar el cluster de shards**

```
rs.initiate(
  {
    _id : <replicaSetName>,
    members: [
      { _id : 0, host : "s1-mongo1.example.net:27018" },
      { _id : 1, host : "s1-mongo2.example.net:27018" },
      { _id : 2, host : "s1-mongo3.example.net:27018" }
    ]
  }
)
```

Particionamiento

Para **añadir** nuevos shards

```
sh.addShard("replSetName/
s1-mongo1.example.net:27018,
s1-mongo2.example.net:27018,
s1-mongo3.example.net:27018")
```

Para **habilitar** sharding en una base de datos

```
sh.enableSharding("<database>")
```

Para **añadir claves** de sharding

- Hash

```
sh.shardCollection("<database>.<collection>",{<shard key field>: "hashed" })
```

- Rango

```
sh.shardCollection("<database>.<collection>",{<shard key field>: 1, ... })
```

Replicación y Particionamiento

Enrique Barra Arias

Andrés Muñoz

Alejandro Pozo