Clase 2 – Arquitectura Distribuida

Distribución (Sharding)

Replicación (repaso)

- La replicación se realiza a través de estructuras denominadas **Replica Set**.
- Un **Replica Set** es un grupo de procesos de mongo que mantienen el mismo conjunto de datos. Estos proveen **redundancia** y **alta disponibilidad**.

- Un **nodo** (proceso mongod) se define como **Primario** y recibe todas las **operaciones de escritura**.
- Sólo puede haber un **único nodo primario**, por lo tanto se proporciona una **consistencia** estricta en la **escritura**.
- Los nodos **secundarios replican el log de operación** del nodo primario y aplican las operaciones en sus datos **asincrónicamente**.
- Si el nodo primario no está disponible, se elige uno nuevo por votación de entre los secundarios.

Replicación (repaso)

27058

27059

27060

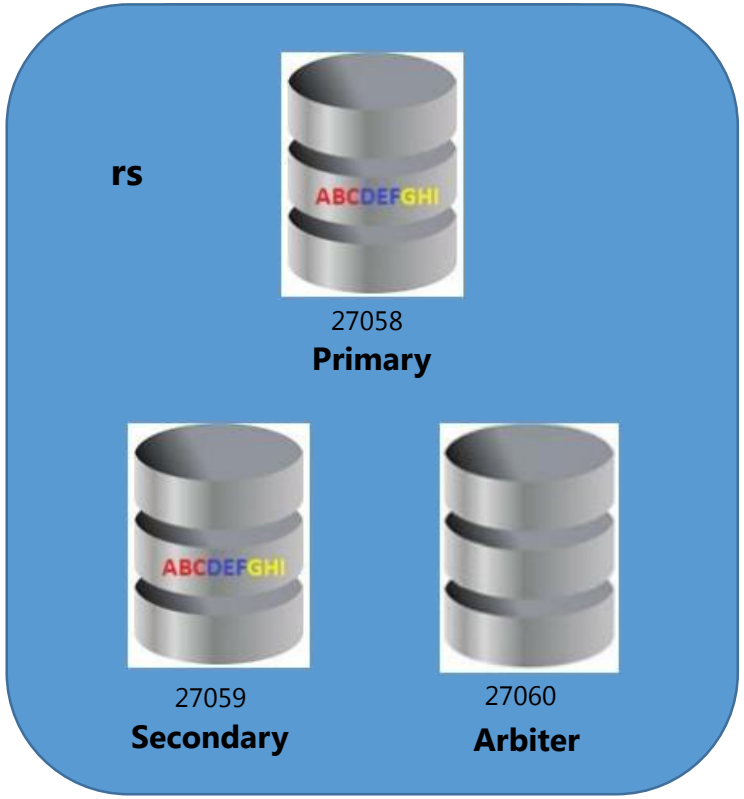
Creamos un servidor levantando una instancia

mongod **--replSet rs** --port 27058.....

Creamos 2 nuevas intancias de mongod

mongod **--replSet rs** --port 27059.....
mongod **--replSet rs** --port 27060.....

Replicación (repaso)



- Configuramos un ReplicaSet

```
mongo --port 27058
        cfg={_id:"rs",
                members: [
                        {_id:0, host:"localhost:27058“, priority: 2},
                        {_id:1, host:"localhost:27059"},
                        {_id:2, host:"localhost:27060“,
                             arbiterOnly: true}
                ]
        }

        rs.initiate( cfg )
```

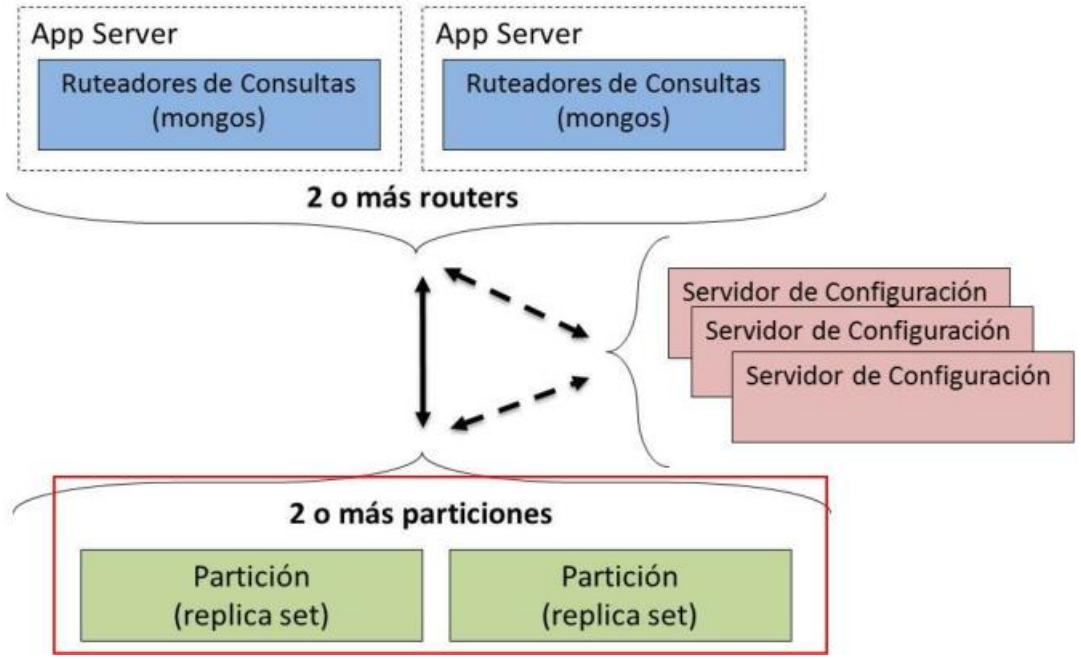
Sharding

MongoDB soporta particionamiento de datos a través de un cluster de servidores.
Una vez definido el criterio de particionamiento, MongoDB se ocupará de particionar los datos y distribuirlos de la forma más equitativa posible.
Los datos serán divididos en porciones llamadas **chunks** y serán almacenados en un nodo o conjunto de nodos llamados **shards**.

Un cluster particionado tiene tres componentes básicos:

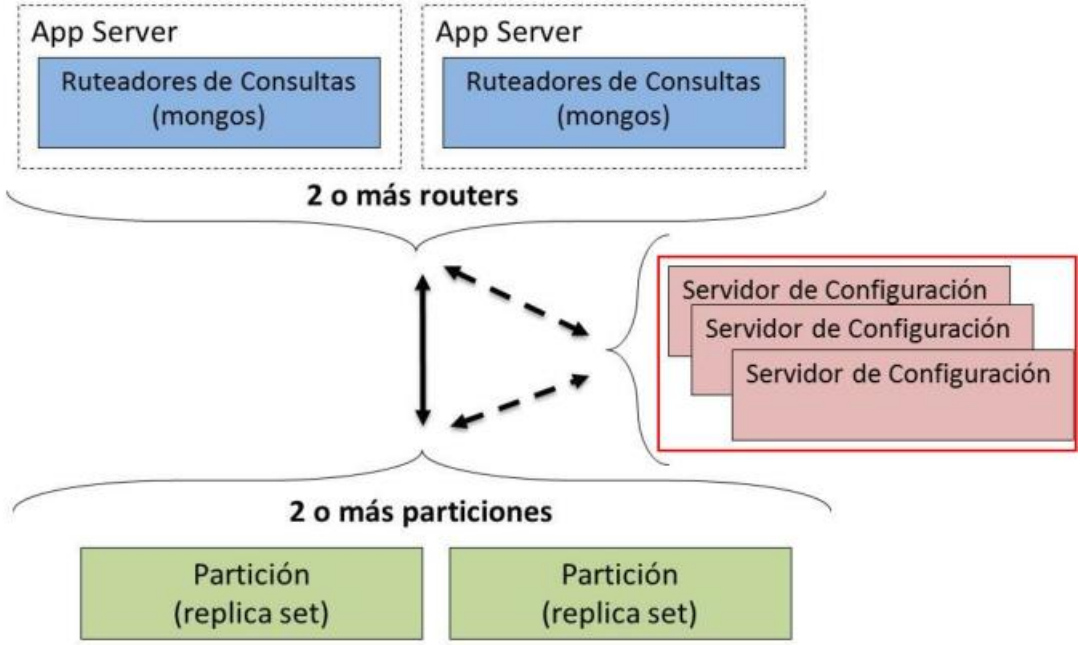
- Particiones (**mongod --shardsvr --replSet** …)
- Ruteadores de Consultas (**mongos --configdb** …)
- Servidores de Configuración. (**mongod --configsvr** …)

Particiones (Shards)



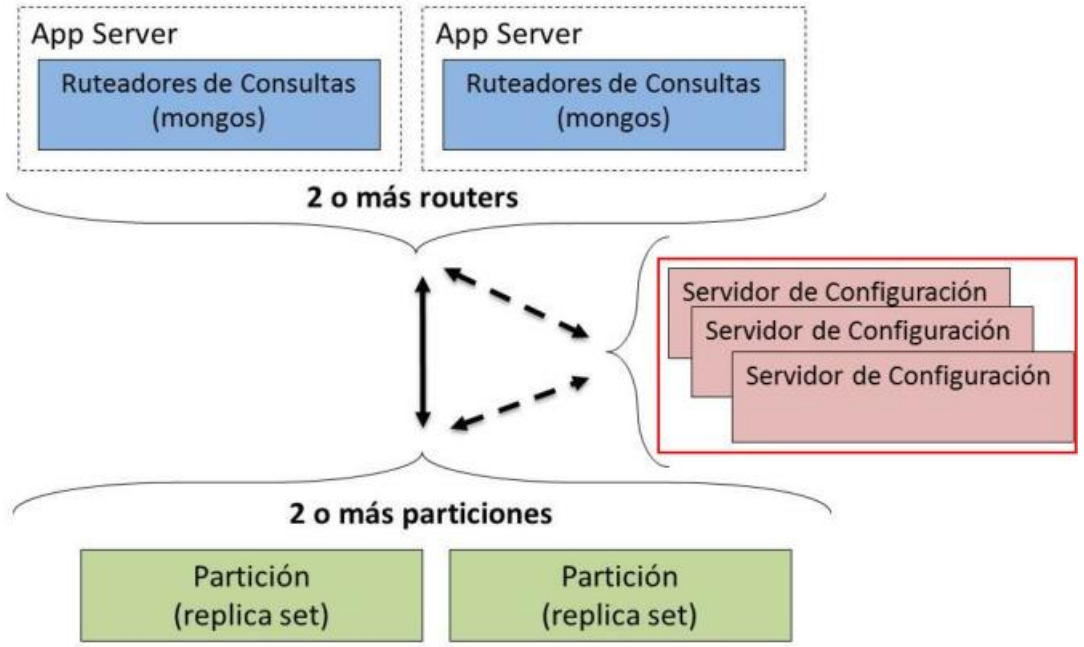
Cada partición, llamada **Shard**, almacena una porción de los datos del cluster. Para proveer alta disponibilidad y consistencia de datos cada partición debe ser un Replica Set, ya que si se tuviera un solo servidor y este fallara, se perdería esa parte de los datos.
Cada **Shard** estará constituido por instancias **mongod**, a las que habrá que agregarle la opción **--shardsvr** (junto con el resto de las opciones deseadas, como ser **--replSet <nombre_replica_set>**) al momento de levantarla.

Servidores de Configuración



Almacenan la metadata del cluster, que incluye las db, las colecciones, los **shards** y los **chunks**. Hay un documento por cada chunk, en el que se indica que rango de valores incluye y en que shard se encuentra. El ruteador utiliza esta información para direccionar las operaciones al shard que corresponda.
Estos servidores también son instancias **mongod**, a las que habrá que agregarle la opción **--configsvr** (junto con el resto de las opciones deseadas) al momento de levantarla.

Servidores de Configuración



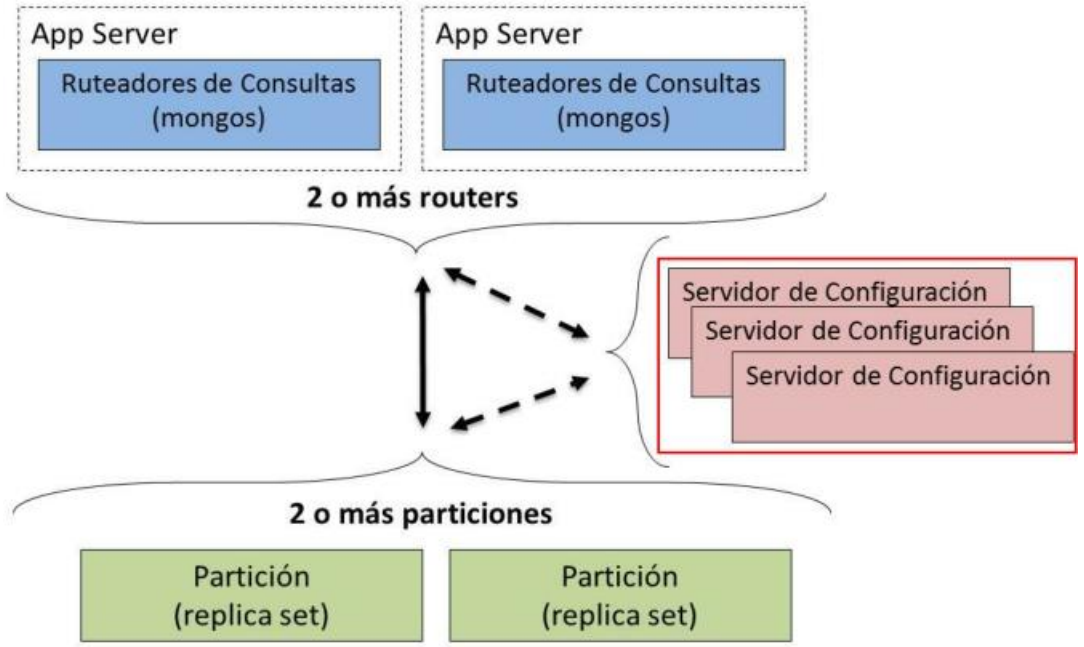
Las bases de datos `config` y `admin` son almacenadas en ellos.

El ruteador **mongos cachea la información de los servidores de configuración**:

- al iniciar
- cuando hay un cambio en la metadata del cluster

Servidores de Configuración (Replica Set)

Opcional en MongoDB v3.2 y Obligatorio en MongoDB v3.4



Los servidores de configuración pueden implementarse como un **Replica Set** de instancias que deben tener a **WiredTiger** como motor de almacenamiento.

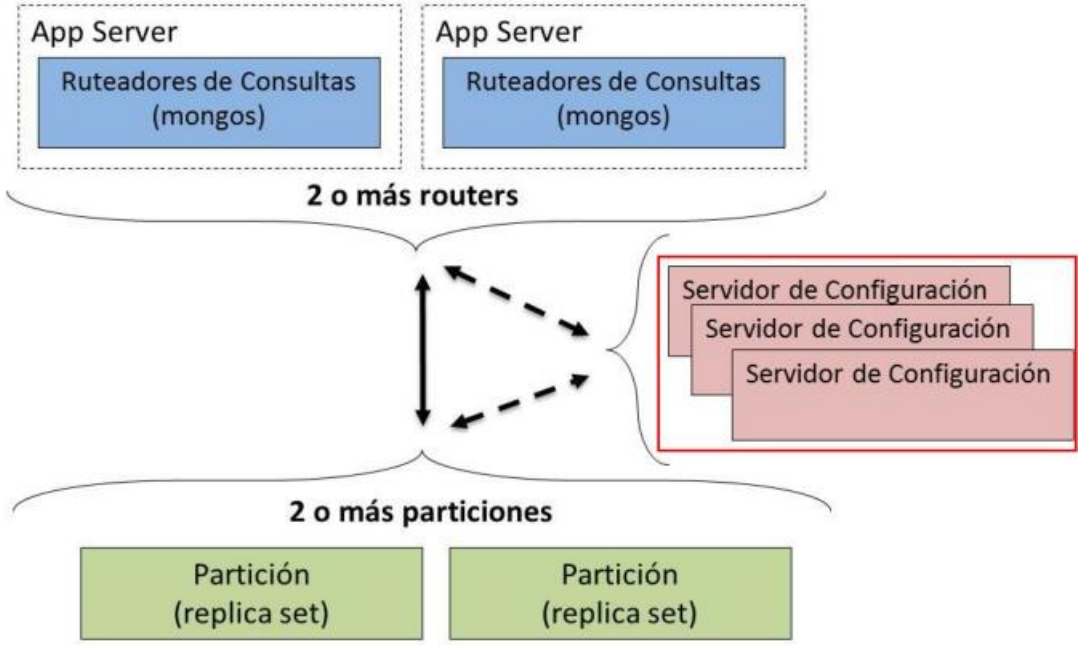
Usar un Replica Set mejora la consistencia entre los servidores, ya que puede tomar ventaja de los protocolos de lectura y escritura de estos.

Un Replica Set usado como config server:

- No puede tener Arbiters.
- No puede tener miembros delayed.
- Deben construir índices.

Servidores de Configuración (Replica Set)

Opcional en MongoDB v3.2 y Obligatorio en MongoDB v3.4

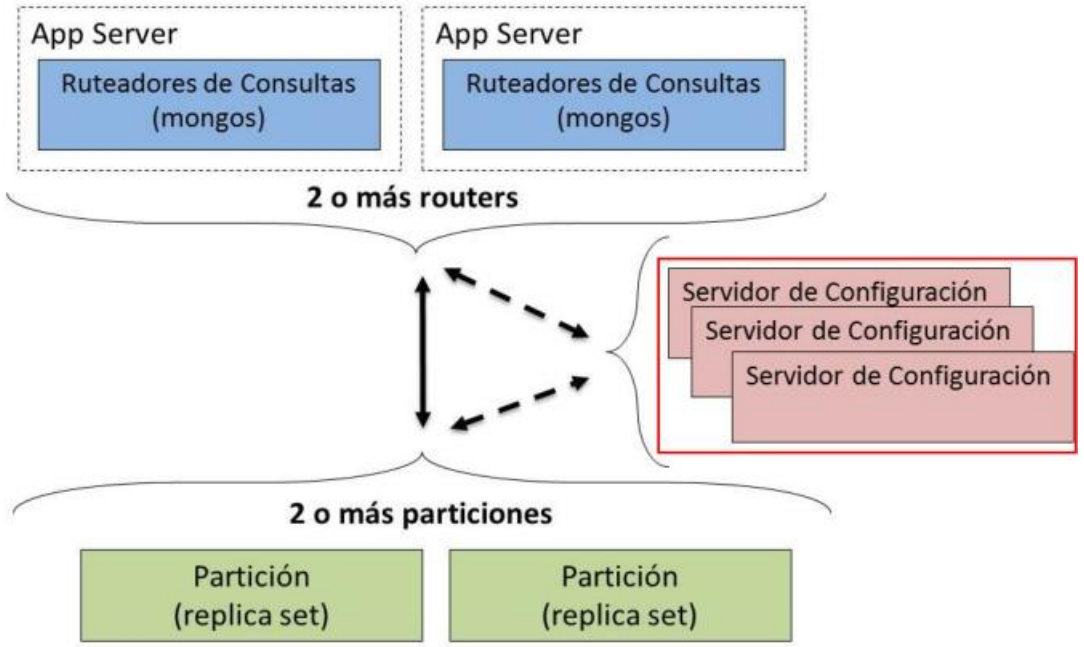


A partir de la versión 3.4, el proceso **balancer** corre sobre el Primary del Replica Set de configuración.

Si el Replica Set no tiene un Primary, la metadata del cluster pasa a ser read only.

Al **modificar la data** de los servidores de configuración se usa **majority** como **write concern** y al **leer** de ellos se utiliza un majority como **read concern**.

Servidores de Configuración

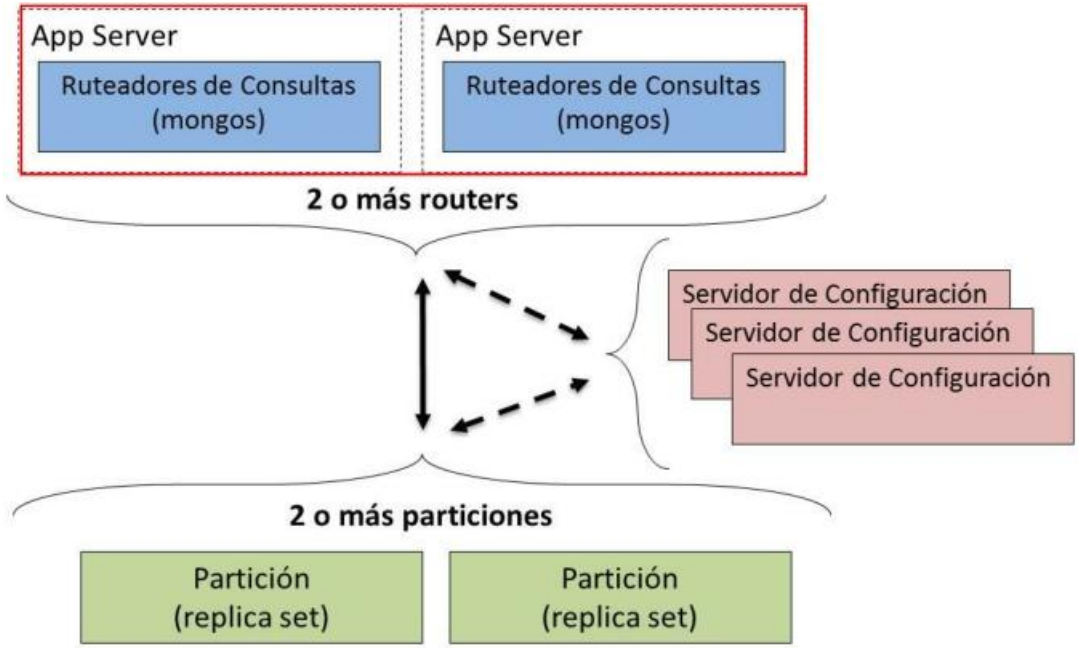


MongoDB v3.2 y anteriores

En **producción**, los clusters deberán tener **3 servidores de configuración**.

Si se **cae uno** de los servidores, la **metadata** del cluster pasará a ser de **sólo lectura** hasta que vuelvan a ser 3.

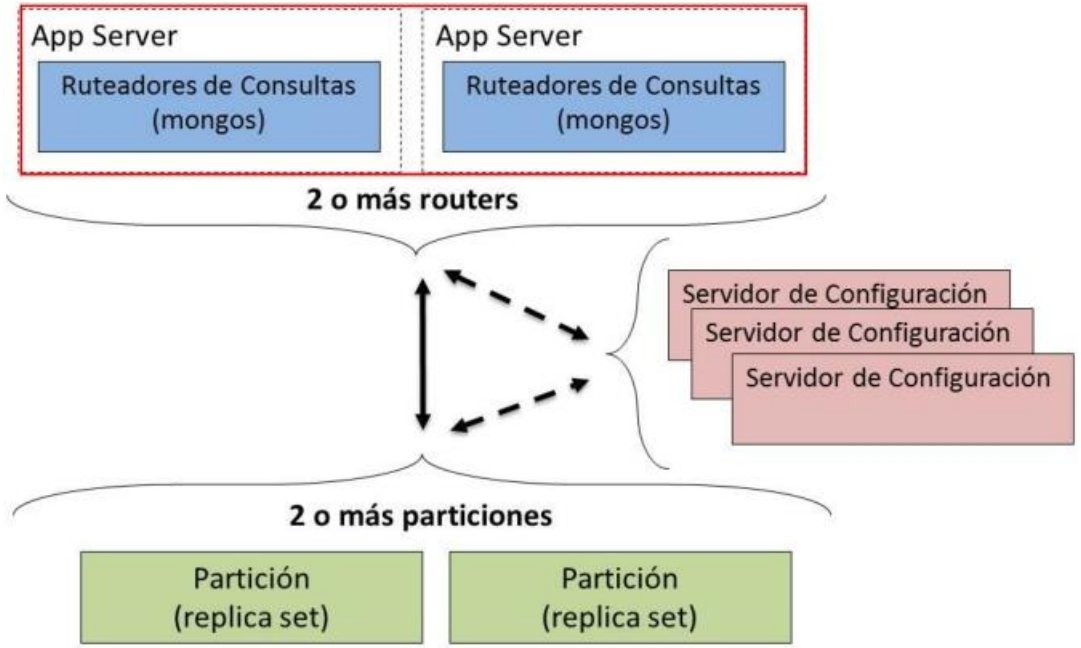
Ruteadores



Son la **interfaz con las aplicaciones, procesan y dirigen cada operación a la partición correspondiente**. Para saber hacia donde rutear las consultas y escrituras, cada **mongos** debe conocer en que nodo se encuentra cada porción de los datos, información que obtendrá de los servidores de configuración. Sumado a rutear las consultas y escrituras, el trabajo de cada **mongos** será dividir los chunks cuando pasen de su tamaño límite (*splitting*).

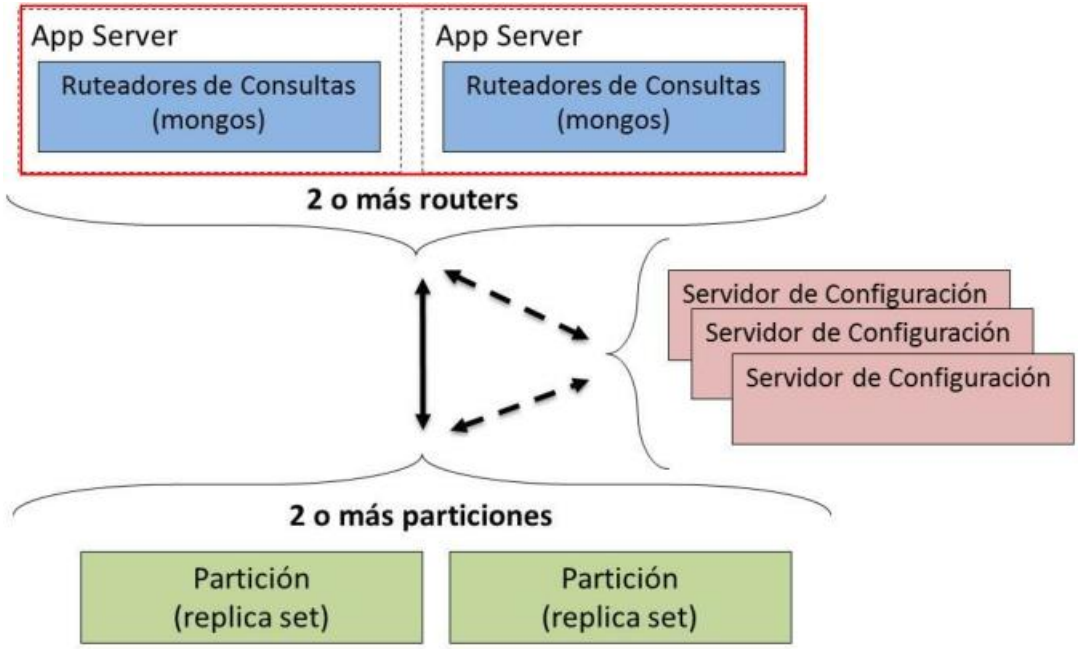
En versiones anteriores a la 3.4, también se encargaba de mantener balanceada la cantidad de **chunks** entre los **shards** (*balancer*).

Ruteadores



Convenientemente, un cluster puede tener **más de un ruteador de consultas** para **distribuir la carga** de los clientes a los ruteadores y para asegurarse de que este **no sea un SPOF**, ya que sin este no nos podremos comunicar con el cluster.

Ruteadores (3.4)



Para levantar una instancia mongos se le debe especificar en la opción configdb el nombre del Replica Set y al menos uno de los miembros del Replica Set, el mongos se encargará de averiguar la lista completa:

mongos --configdb <rs>/<cfgsvr1>

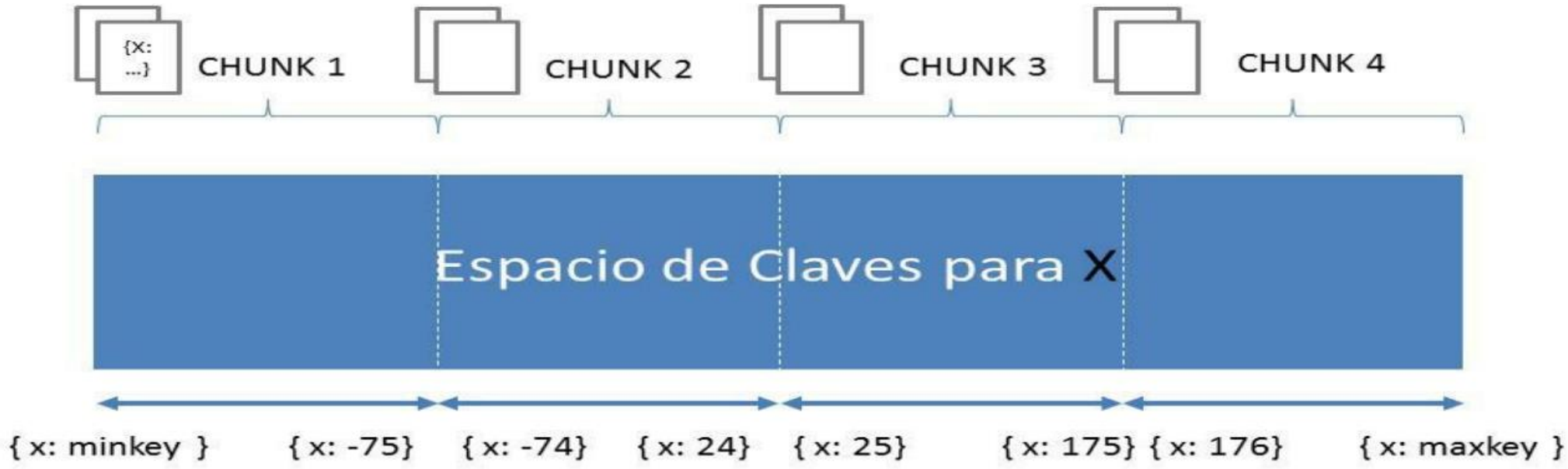
Sharding

- Para particionar los datos es necesaria una clave de partición.
- La clave de partición o **shard key** es un **campo indexado existente** en cada documento de un agregado.
- MongoDB **crea los chunks** en base a **conjuntos de valores de la shard key** y **distribuye los chunks** en **cada partición**.
- Para dividir las claves de partición en chunks, MongoDB usa dos posibles esquemas:
  - Particionamiento basado en rangos
  - Particionamiento basado en hash.

Particionamiento basado en rangos

- Divide el conjunto de datos en rangos disjuntos determinados por los valores de las claves de la partición.
- Definimos un **chunk** como un **rango de valores con un mínimo y un máximo**.
- Dado un sistema con particiones basadas en rangos, los **documentos** que tienen **valores de clave de partición cercanos** tienen **mayor probabilidad** de estar en el **mismo chunk**, y de esta manera en la **misma partición**.

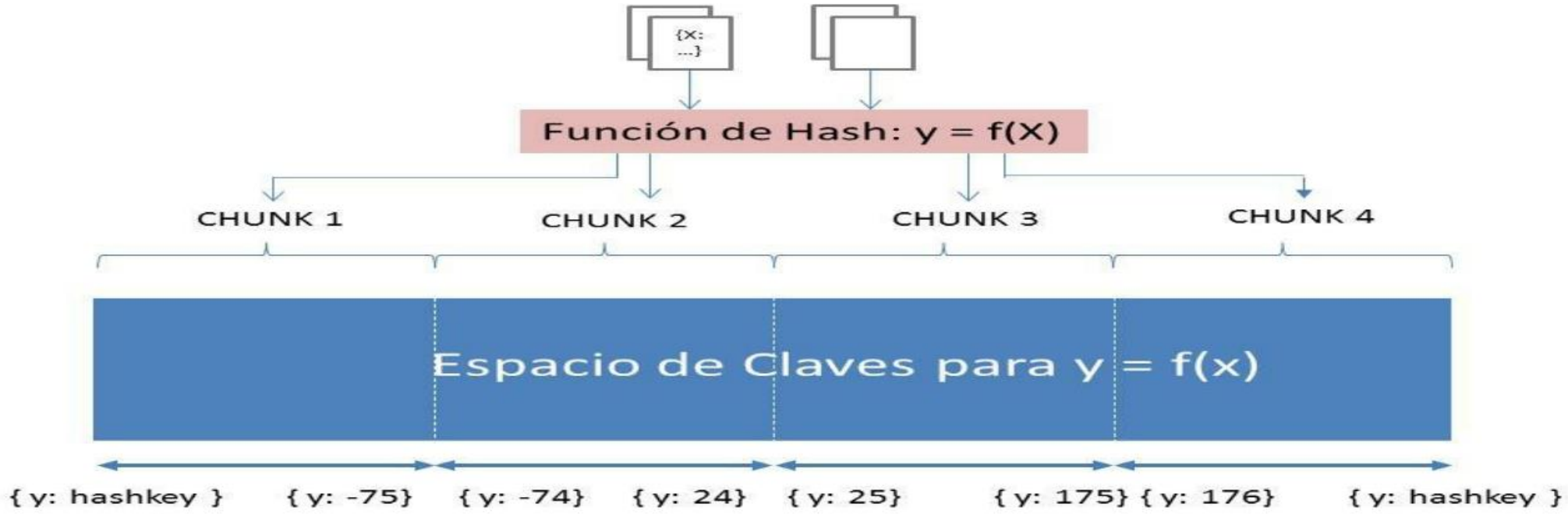
Particionamiento basado en rangos



Particionamiento basado en hash

- En este esquema de particionamiento, MongoDB calcula el valor de hash de una clave y utiliza ese valor para la división de chunks.
- Con este esquema, **dos documentos** con **valores de clave cercanos** tienen **baja probabilidad** de ser parte del **mismo chunk**.
- Esto asegura una **distribución aleatoria y equitativa** de los documentos entre los shards del cluster.

Particionamiento basado en rangos



Diferencias entre ambos esquemas

Particionamiento por Rangos	Particionamiento por Hash
Soporta consultas basadas en rangos más eficientes.	Mayor ineficiencia en las búsquedas por rango.
Ante una consulta basada en rangos el ruteador de consulta puede fácilmente determinar cuál/es chunk/s están vinculados con ese rango.	Consulta basada en rangos no es resuelta con un solo chunk sino que requiere ir a diferentes chunks en los que los datos se encuentran distribuidos.
Puede resultar en una distribución de datos no uniforme, siendo negativo para la estrategia de particionamiento.	Asegura una distribución uniforme de los datos.
Algunos shards pueden resultar con una mayor cantidad de carga de operaciones debido a una distribución no uniforme de los datos.	Garantiza una mejor distribución de la carga de operaciones dirigidas a los diferentes shards.

Mantenimiento de una Distribución Balanceada

- La adición de **nuevos datos** o la adición de nuevos servidores pueden resultar en una **distribución de datos desbalanceada** dentro del cluster.
- Podría suceder que **una shard** particular contenga significativamente **más chunks que otro**.
- MongoDB **asegura un cluster balanceado** usando **dos procesos** que corren de fondo (en background): **splitting** y **balancer**.

Splitting

- Cuando un **chunk crece más** allá de un **tamaño especificado**, MongoDB dividirá el chunk en dos partes. (Valor entre 1 y 1024 MB. Default: 64 MB)
- Estas **divisiones de chunks** son generadas por operaciones de **inserción y actualización**.
- Para dividir a los chunks, MongoDB **no migra datos o afecta los shards**. Lo único que hace es modificar la metadata en los servidores de configuración, agregando un nuevo documento por el chunk extra y modificando el rango de la shard key de ambos chunks.

Balancer

- Es un proceso que **administra las migraciones de chunks**. A partir de la versión 3.4, este proceso corre en el Primary del Replica Set de los servidores de configuración. En versiones anteriores, este proceso estaba dentro de los ruteadores de consultas (mongos).

- Cuando la **distribución** de una colección particionada en un cluster **no es equilibrada**, **migra chunks** hasta que quede lo más balanceado posible.

- Los shards administran las **migraciones de chunks** como una **operación que se ejecuta de fondo**.

Shard Key

Se deberá tener especial consideración al elegir la clave de particionamiento (Shard key), ya que en base a esta se distribuirán los datos en el cluster. Para esto se deberán tener en cuenta 3 aspectos:

- **Cardinalidad**: Los distintos valores posibles de la Shard key no deben ser muy limitados para que los **chunks** sean fácilmente **divisibles**.
- **Aleatoriedad**: Una Shard key con un alto grado de aleatoriedad podrá garantizar una **distribución** adecuada de **escrituras** a lo largo del cluster y **no permitirá** que un Shard se convierta en **cuello de botella**. De esta forma se puede conseguir **escalamiento de escrituras**.
- **Aislamiento de Consulta**: Si la Shard key es el campo mayormente **usado en los queries**, el mongos podrá rutear la consulta a un **único Shard** o un conjunto limitado de Shards.

Considerar el uso de Shard key compuestas o basadas en hash para adecuarse a estos aspectos.

Creando el Cluster

Luego de haber levantado todas las instancias necesarias, es decir, los shards, los ruteadores de consultas y los servidores de configuración, se deberá conectar con el shell al **mongos** y agregar los shards al cluster.
Para agregar shards al cluster se usará la función: **sh.addShard("<host>")**

Si es una única **instancia**:
<host> = nombreHost**:**puerto
Si es un **Réplica Set**:
<host> = replicaSet**/**nombreHost**:**puerto

Particionar una colección

Primero se deberá habilitar el Sharding para la base de datos deseada:
sh.enableSharding(“<database>”)

Luego de decidir la Shard key adecuada para dividir los datos del cluster, se deberá crear su índice:
db.colección.createIndex(<shard_key>)

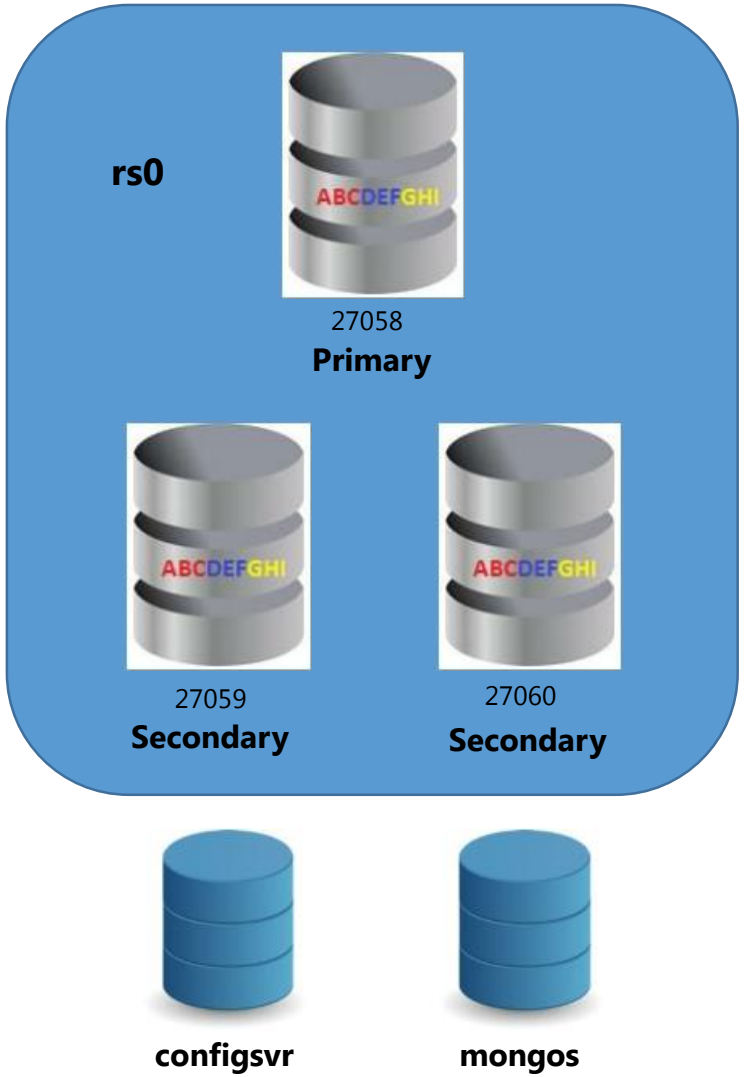
Y finalmente indicamos la colección a particionar junto con la Shard key:
sh.shardCollection(“<db>.<coleccion>”, <shard_key>, <unique>)
Si se pone unique en true, se asegurará que el índice tenga unique:true.

Sharding

VEAMOS UN CASO PRÁCTICO

DE SHARDING

Práctica de Sharding

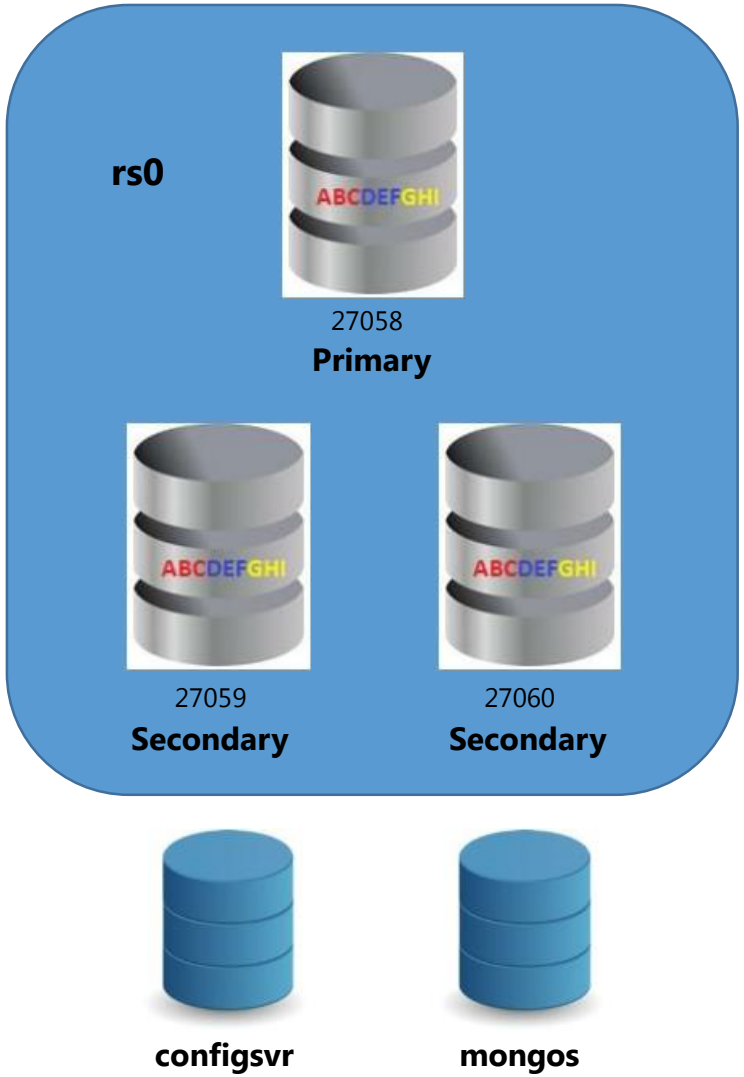


1. Levantamos las instancias de MongoDB con el parámetro --shardsvr
   mongod --shardsvr --replSet rs0 --port 27058 …

2. Levantamos un servidor de configuración y creamos el Replica Set.
   mongod --configsvr --replSet c --port 27500 …

```
mongo --port 27500
        rs.initiate({
                _id: "c",
                members: [{
                        _id:0,
                        host: "localhost:27500"
                }]
        })
```

Práctica de Sharding

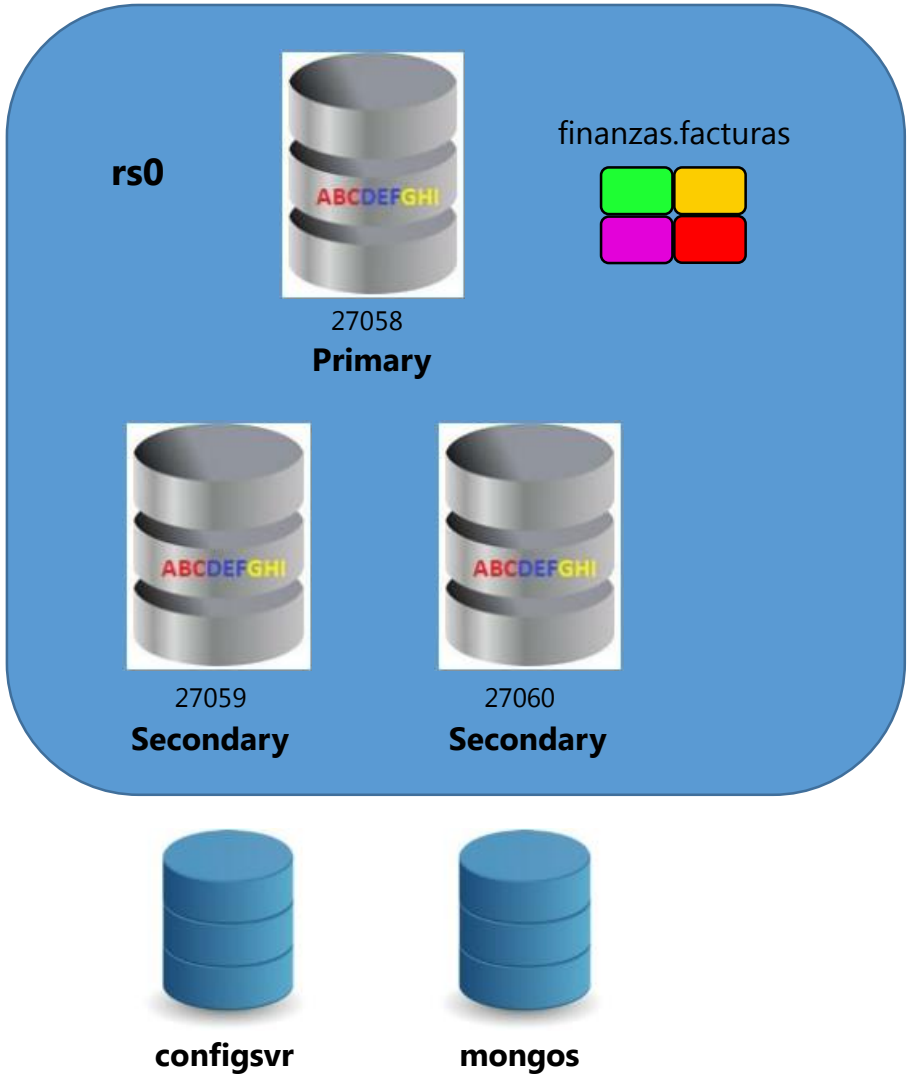


3. Levantamos un servidor de Ruteo y lo apuntamos hacia el Replica Set de configuración
   mongos --configdb c/localhost:27500

4. Realizamos luego el agregado del primer shard, que en este caso será el Replica Set "rs0".
   mongo (nos conectamos al mongos)

   sh.addShard("rs0/localhost:27058")

Práctica de Sharding



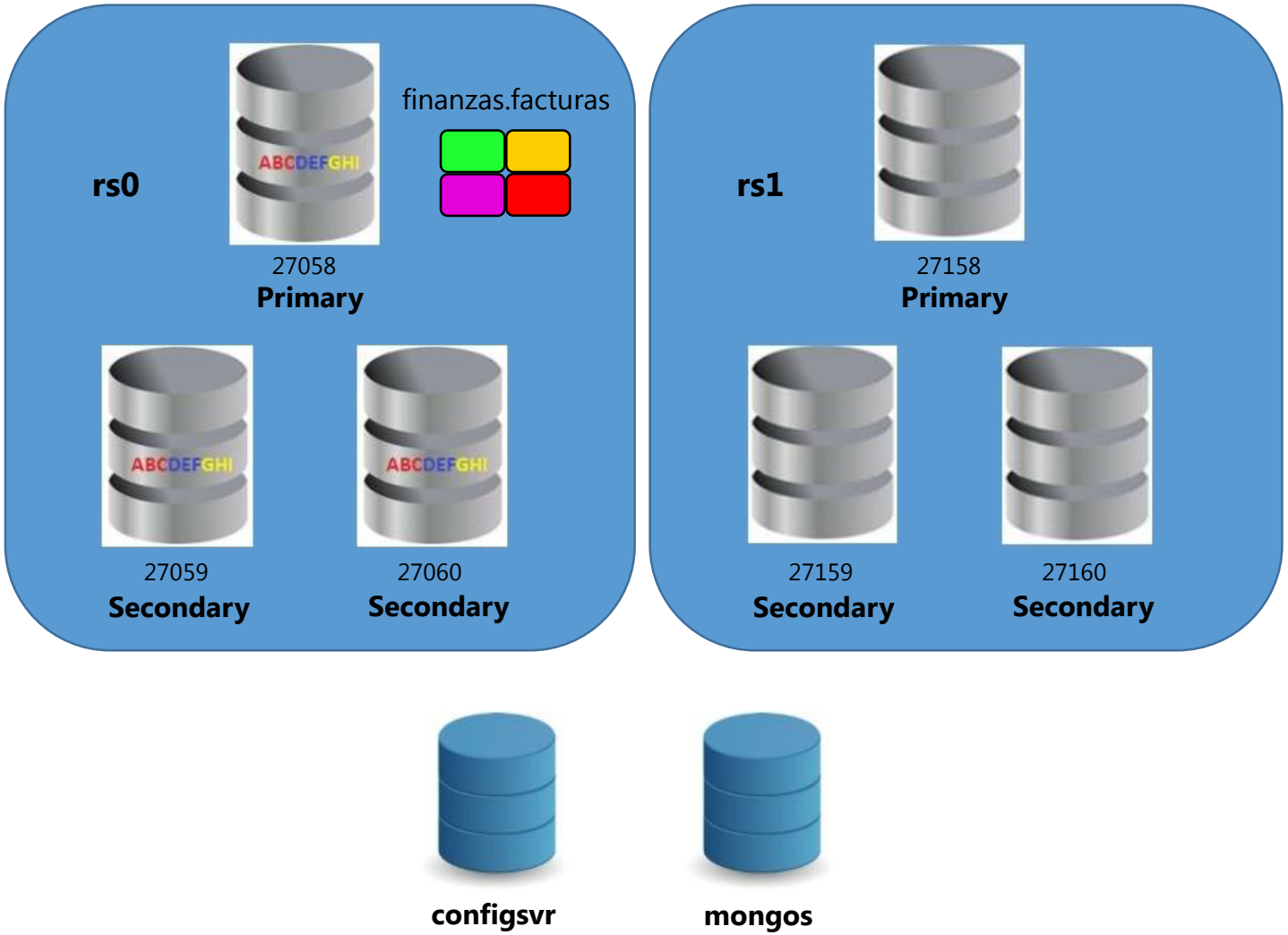
1. Definimos la clave para realizar sharding y creamos el índice necesario.
   db.facturas.createIndex({"cliente.region": 1, "cliente.nombre": 1})

2. Activamos el Sharding para la base de datos finanzas.
   sh.enableSharding("finanzas")

   sh.shardCollection("finanzas.facturas", {"cliente.región":1,"cliente.nombre":1})

Distribuye los datos de la colección en chunks.
Cada chunk tiene los documentos entre un mínimo y un máximo.

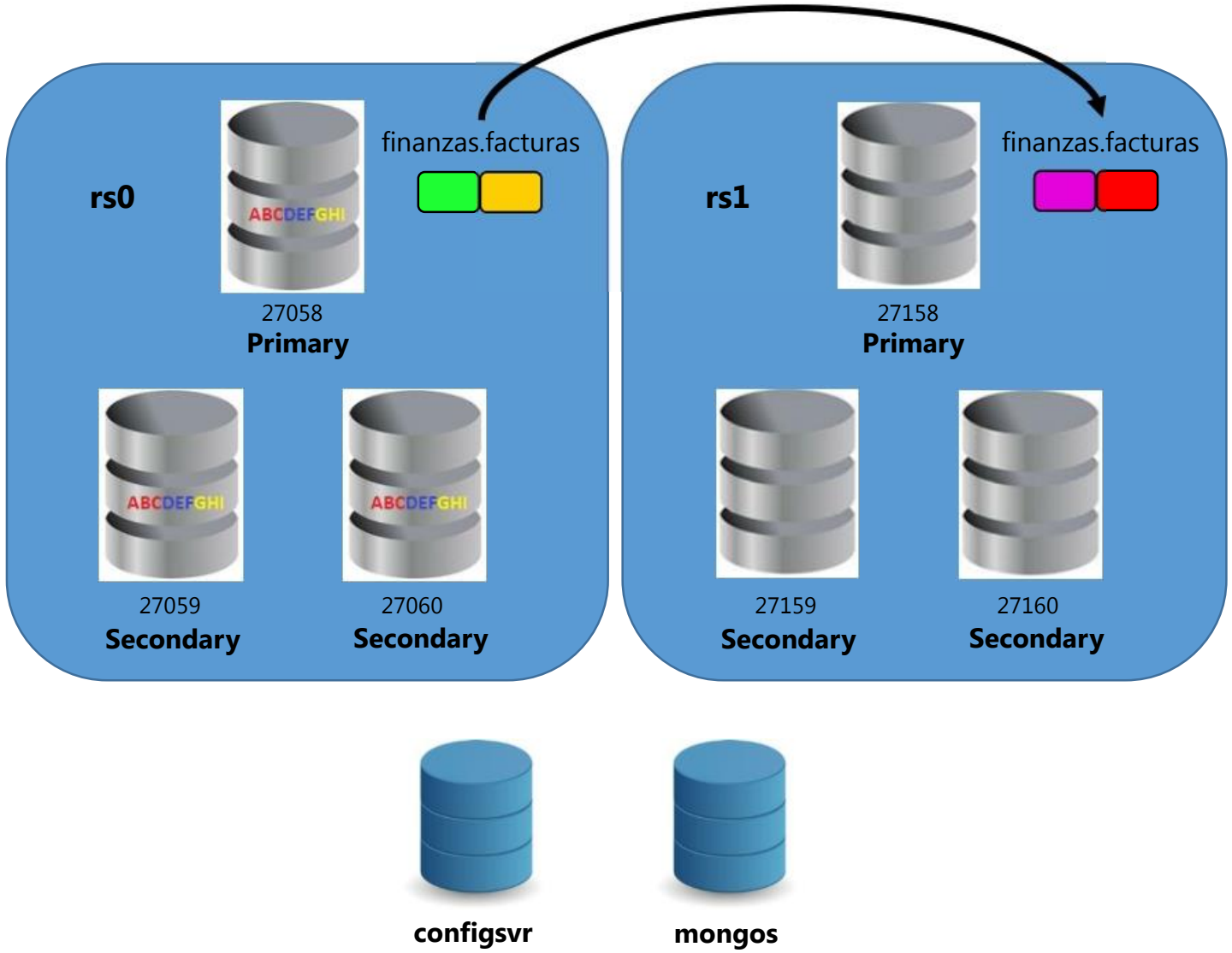
Práctica de Sharding



5. Configuramos un segundo Replica Set (**idem rs0**)
6. Lo agregamos como un nuevo Shard

mongo
sh.addShard("rs1/localhost:27158")

Práctica de Sharding



Una vez realizado este ultimo paso, el **balanceador** comenzará a distribuir los chunks hasta que los shards queden equiparados.